<戒烟辅助系统>

软件架构文档

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <10/11/23> | <1.0> | <软件架构文档第一版> | <丁灿、张浩宇> |
| <20/11/23> | <2.0> | <软件架构文档第二版> | <丁灿、张浩宇> |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 5

3. 逻辑视图 5

3.1 概述 5

3.2 在构架方面具有重要意义的设计包 6

4. 进程视图 7

5. 部署视图 9

6. 实现视图 9

7. 技术视图 10

8. 数据视图（可选） 10

9. 核心算法设计（可选） 10

10. 质量属性的设计 10

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

特定读者和使用说明：

开发人员： 可以深入了解系统的技术细节，以指导他们在实现具体功能时的决策和实践。

测试人员： 可以利用文档了解系统的关键组件和接口，以编写测试用例和进行综合测试。

项目经理： 可以了解系统的整体架构，以更好地进行项目计划和管理。

维护人员： 在系统维护和升级时，可以使用文档了解系统的设计和组织结构。

使用说明：

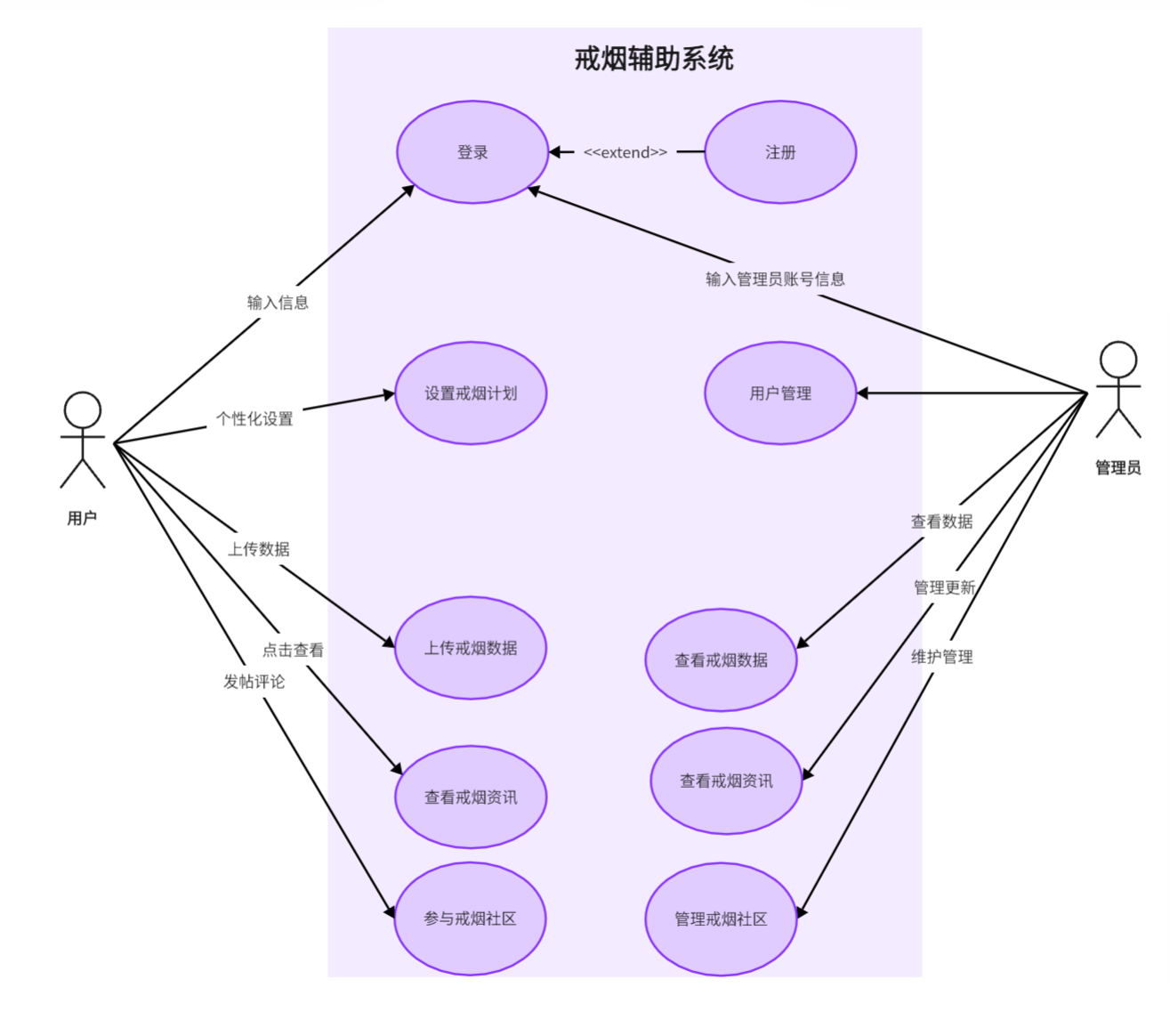
阅读者应该从文档的介绍部分开始，以获得关于文档目的和范围的概述。然后，根据其需求深入阅读关键部分，例如总体架构、模块说明和关键设计决策。对于特定任务，阅读者可以快速定位所需信息，而无需深入阅读整个文档。

这样的软件构架文档将有助于确保整个团队对系统的结构和设计有共同的理解，促进协作，提高开发效率，同时也为未来的维护和升级提供了基础。

## 参考资料

1. Taylor, Richard N ,Medvidovic,et al.Software Architecture: Foundations, Theory, and Practice[C]//Software Engineering, 2010 ACM/IEEE 32nd International Conference on.ACM, 2010.DOI:10.1145/1810295.1810435.
2. Erl T .Service-Oriented Architecture: Concepts, Technology, and Design[M]. 2005.
3. Agile Software Development: Principles, Patterns, and Practices
4. Building Microservices: Designing Fine-Grained Systems

# 用例视图



# 逻辑视图

## 概述

**架构风格：分层架构**

表示层：负责用户界面和用户体验，网页界面。。

服务层：处理复杂的业务逻辑，处理应用程序的核心功能和业务规则。可能包括API调用和外部服务集成。

数据层：数据库，负责存储应用程序数据。

**模块分解和协同情况**

用户界面模块（表示层）：网页界面，负责接收用户输入和显示输出。

认证和授权模块（服务层）：处理用户登录、注册和权限管理。

戒烟数据管理模块（服务层）：负责处理用户上传的戒烟数据。

戒烟计划模块（服务层）：辅助用户制定和跟踪戒烟计划。

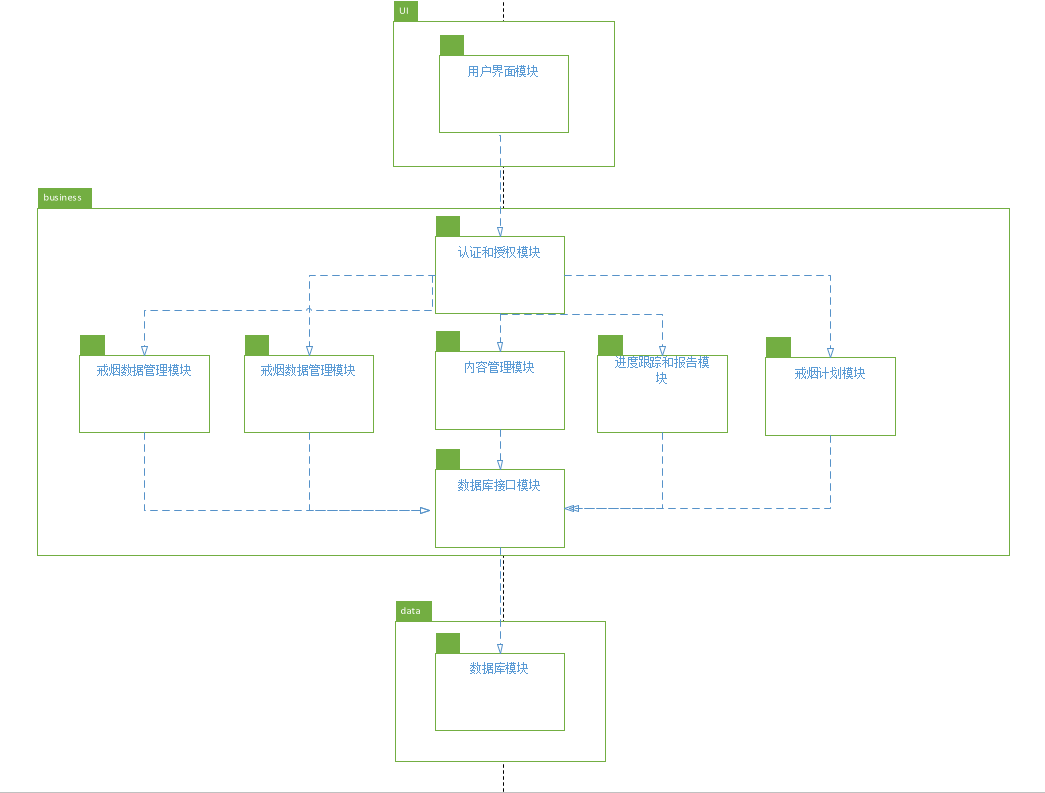
进度跟踪和报告模块（服务层）：生成用户戒烟进度报告和统计数据。

社区交互模块（服务层）：管理用户论坛和社区交互功能。

内容管理模块（服务层）：管理员用来更新戒烟资讯和教育材料。

数据库接口模块（服务层）：提供与数据库交互的接口。

数据库（数据层）：存储用户数据、戒烟数据、社区内容等。



**文字描述**

表示层（用户界面模块）：用户通过这个层级与系统交互，包括注册、登录、查看戒烟进度、上传数据等。

服务层：此层包括几个关键模块，如认证授权、戒烟数据管理、戒烟计划和进度跟踪。这些模块处理系统的核心业务逻辑。

数据层（数据库）：存储所有的用户

## 在构架方面具有重要意义的设计包

**1. 用户管理包**

名称：UserManagement

简要说明：负责处理用户注册、登录、认证和用户信息管理。

重要类：

类名：User

职责：代表系统中的用户，包含用户的个人信息。

属性：用户名、密码、电子邮箱、戒烟数据。

操作：获取和更新个人信息。

类名：AuthenticationService

职责：处理用户认证和权限控制。

操作：登录、登出、权限检查。

**2. 数据管理包**

名称：DataManagement

简要说明：管理用户上传的戒烟数据和戒烟计划。

重要类：

类名：SmokingData

职责：存储和处理用户的吸烟数据。

属性：吸烟频率、吸烟时间、戒烟日期。

类名：QuitPlan

职责：管理用户的戒烟计划。

属性：目标戒烟日期、戒烟策略。

操作：创建计划、更新计划、获取计划详情。

**3. 进度跟踪包**

名称：ProgressTracking

简要说明：用于追踪和展示用户的戒烟进展。

重要类：

类名：ProgressTracker

职责：追踪用户的戒烟进度。

属性：未吸烟天数、节约的金钱。

操作：更新进度、获取进度报告。

**4. 社区互动包**

名称：CommunityInteraction

简要说明：管理社区论坛和用户之间的互动。

重要类：

类名：Forum

职责：管理论坛帖子和讨论。

属性：帖子列表、评论。

操作：发帖、回复、删除帖子。

**5. 内容管理包**

名称：ContentManagement

简要说明：由管理员用来更新系统中的戒烟资讯和教育材料。

重要类：

类名：Article

职责：代表系统中的一篇文章或教育材料。

属性：标题、内容、作者。

操作：发布文章、更新文章。

# 进程视图

**1.Web服务进程（重量级进程）**

职责：处理来自用户界面的所有请求，是系统的主要入口点。

包含线程：

用户请求处理线程：处理用户登录、注册等请求。

数据请求处理线程：处理数据上传和查询请求。

通信模式：使用消息队列来管理来自用户界面的请求。

**2. 业务逻辑处理进程（重量级进程）**

职责：处理核心业务逻辑，如用户认证、数据处理等。

包含线程：

认证处理线程：负责用户的登录和安全验证。

数据分析线程：分析用户上传的戒烟数据。

通信模式：与Web服务进程通过远程过程调用（RPC）或RESTful API交互。

**3. 数据库管理进程（重量级进程）**

职责：负责数据存储和检索。

包含线程：

数据写入线程：处理写入数据库的请求。

数据读取线程：处理来自应用的数据查询请求。

通信模式：通过数据库连接池与业务逻辑处理进程交互。

**4. 社区管理进程（轻量级进程）**

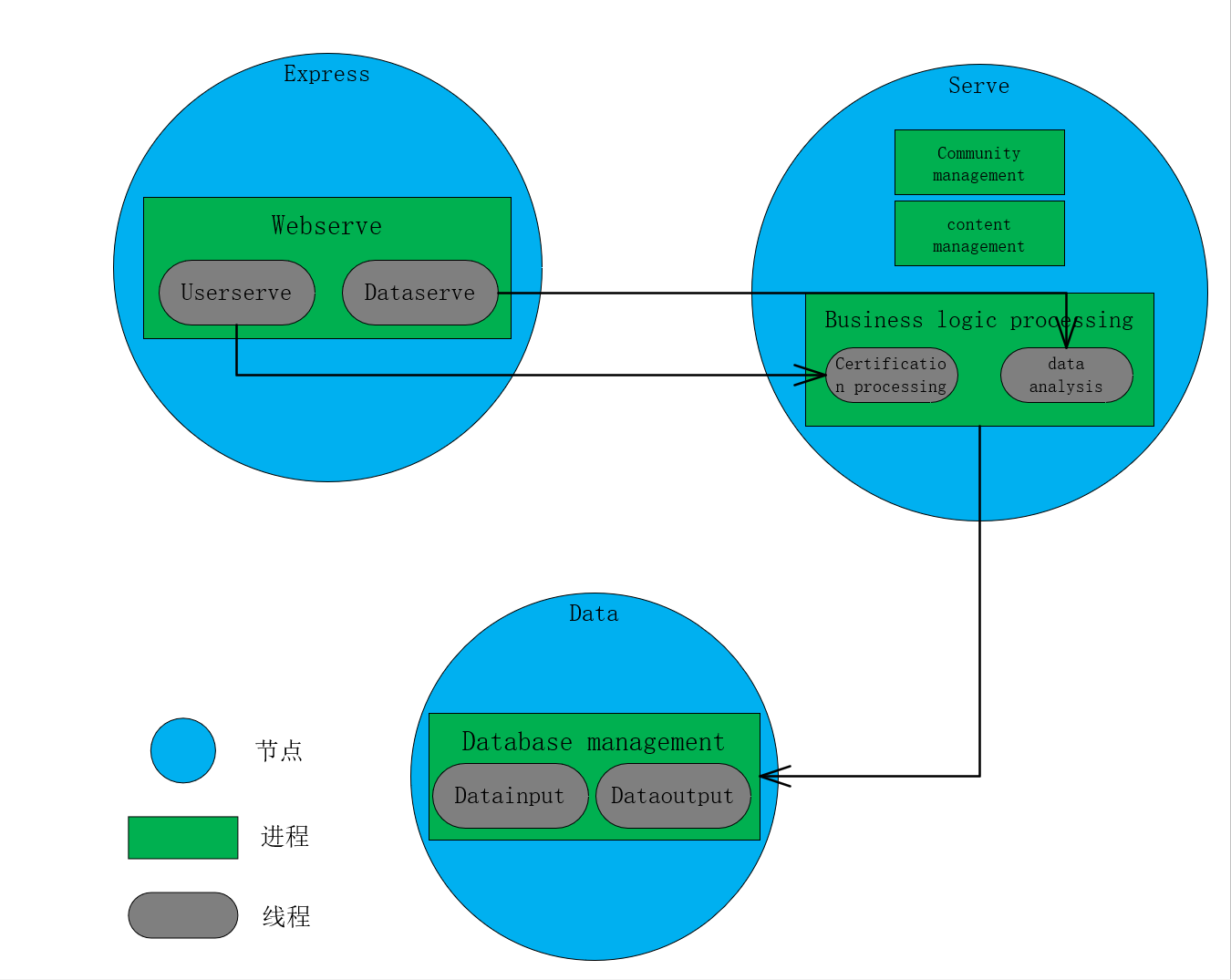
职责：管理社区论坛的操作，如帖子发布和评论。

通信模式：使用HTTP/HTTPS协议与Web服务进程交互。

**5. 内容管理进程（轻量级进程）**

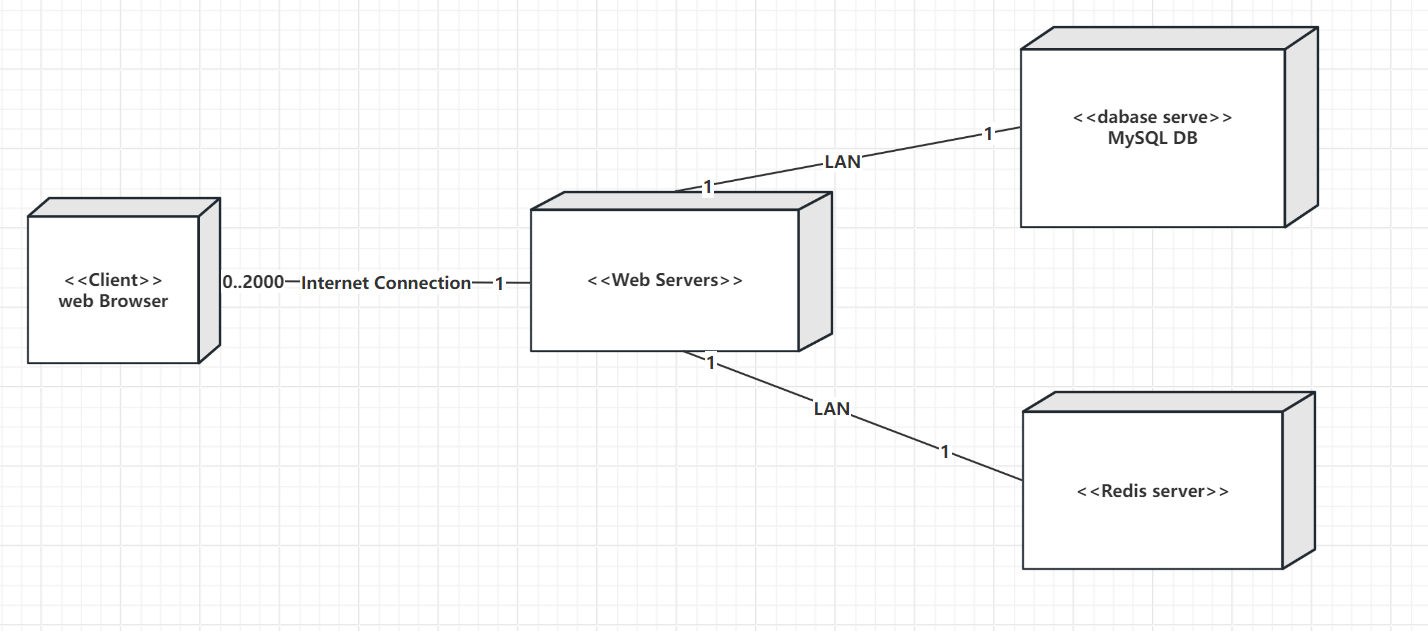
职责：处理内容更新请求，如新文章的发布和旧文章的更新。

通信模式：通过RESTful API与Web服务进程交互。



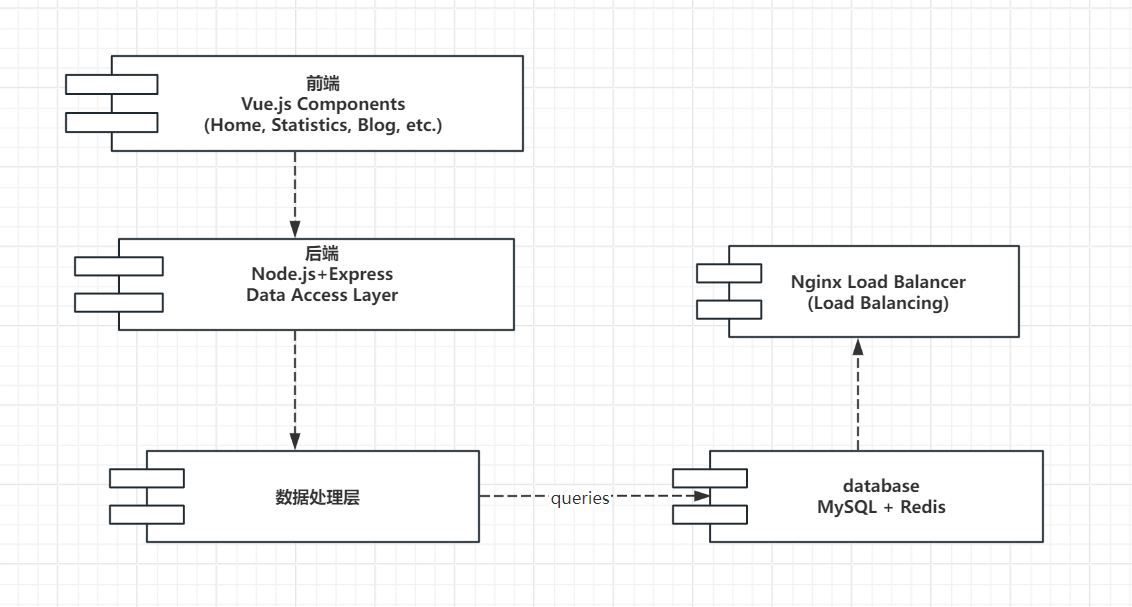
# 部署视图

[本节说明用来部署和运行该软件的一种或多种物理网络（硬件）配置。对于每种配置，它至少应该指出执行该软件的物理节点（计算机、CPU）及其互连情况（总线连接、LAN 连接、点到点连接等）。另外还要包括**进程视图**中的各进程到物理节点的映射。]

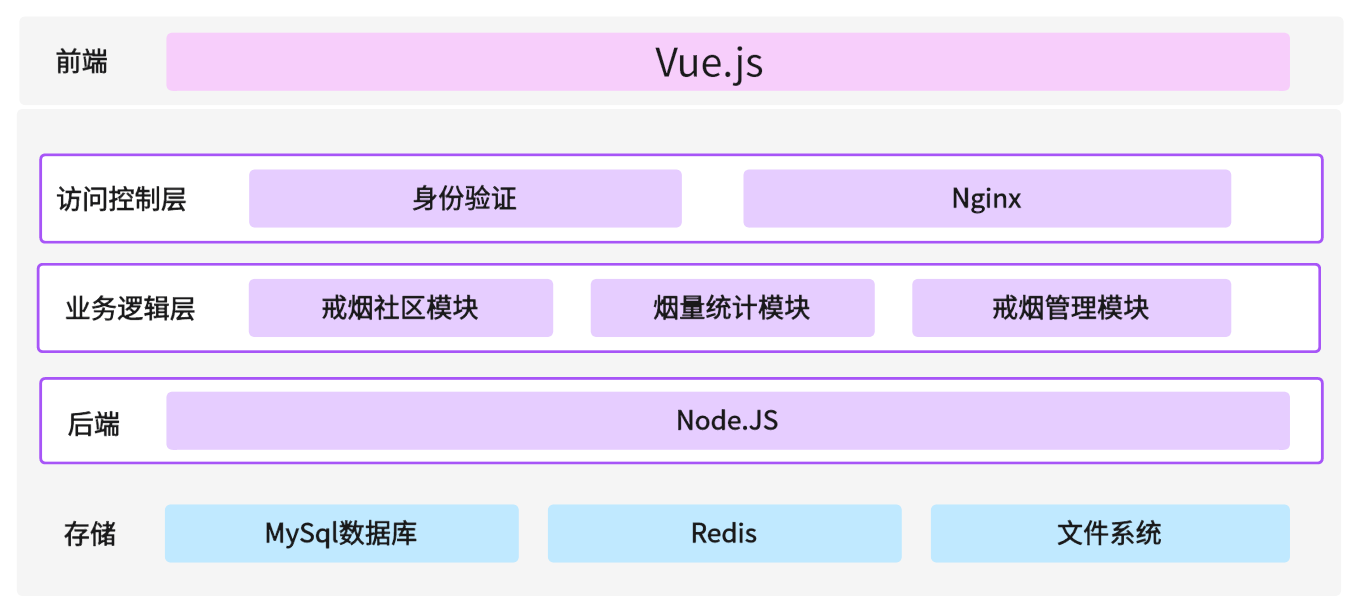


# 实现视图

[本节说明实现模型的整体结构、软件分解为实现模型中的层和子系统的情况，以及所有在构架方面具有重要意义的构件。]



# 技术视图



# 数据视图（可选）

采用关系型数据库，还未设计好数据库组织结构。将在后续补上。

# 核心算法设计（可选）

本系统主要是前后端交互以及数据库的设计，技术栈较为成熟。没有什么重要算法。

# 质量属性的设计

**1. 性能：**

前端性能：使用Vue.js作为前端框架，确保前端页面加载速度和响应时间良好，优化图片、脚本和样式表以减少加载时间。

后端性能：使用Node.js构建后端，采用异步非阻塞I/O操作，提高处理请求的效率。通过适当的数据库索引和查询优化来提高数据库性能。

负载均衡： 使用Nginx进行负载均衡，确保系统可以处理大量并发请求。

**2. 可扩展性：**

水平扩展：设计系统以支持水平扩展，通过增加服务器节点来应对用户量的增加，Nginx负载均衡可帮助实现这一点。

模块化设计：使用Vue.js组件化开发前端，使用Node.js模块化设计后端，使系统更易于扩展和维护。

**3. 可靠性：**

错误处理：实现合理的错误处理机制，通过合适的日志记录、错误信息反馈给用户，以及监控系统状态来提高系统的可靠性。

数据备份与恢复：定期对MySQL数据库进行备份，并考虑使用Redis的持久化功能，以确保数据可靠性。

**4.易用性：**

用户界面设计：采用直观友好的用户界面设计，提供清晰的信息架构和导航，以及良好的用户反馈机制。

响应式设计：使用响应式设计确保用户在不同设备上都有良好的体验，包括桌面、平板和移动设备。

**5.可移植性：**

平台无关性：使用Node.js和Vue.js等平台无关的技术，确保系统可以在不同的操作系统上运行。

容器化部署：考虑使用容器化技术（如Docker）进行部署，提高系统在不同环境中的可移植性。

**6.安全性：**

数据加密：使用HTTPS协议来保障数据在传输过程中的安全性。

身份认证和授权：使用合适的身份认证和授权机制，确保只有授权用户能够访问敏感信息。

防止SQL注入和XSS攻击：对用户输入进行有效的验证和过滤，防止恶意注入和跨站脚本攻击。

这些设计考虑可以帮助确保系统在各个方面都能够达到较高的质量标准，为用户提供稳定、高效、易用的戒烟支持服务。