<游戏交友>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <16/5/2023> | <1.0> | <这是第一版架构文档> | <高肖健 余润泽 陈英昊 王乐峰> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 简介

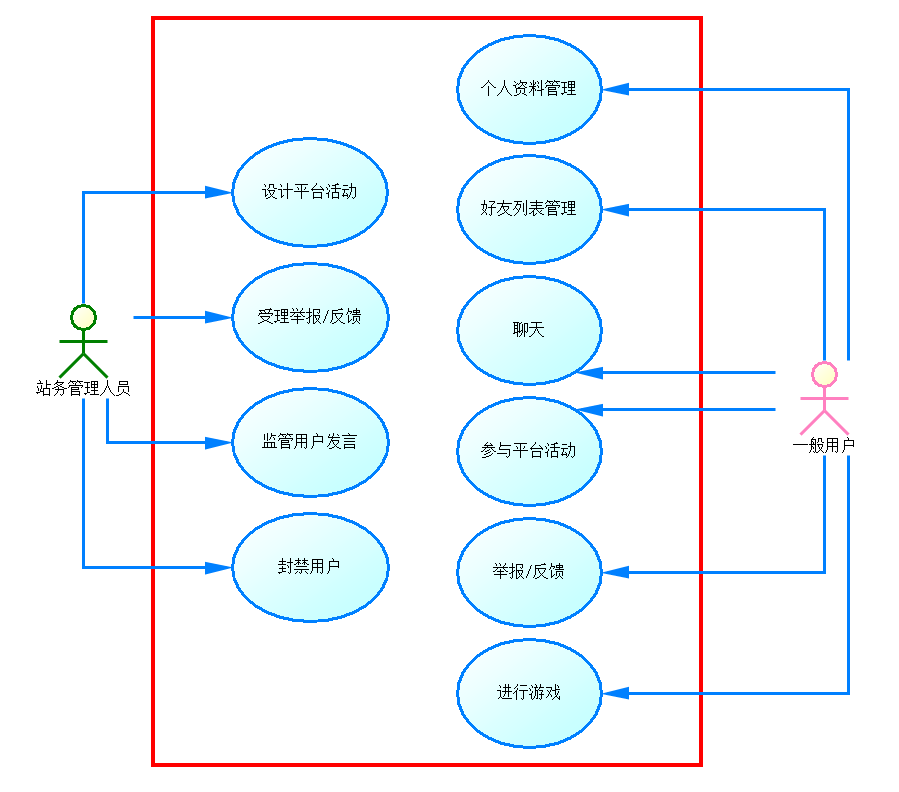
## 目的

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。

## 参考资料

无

# 用例视图

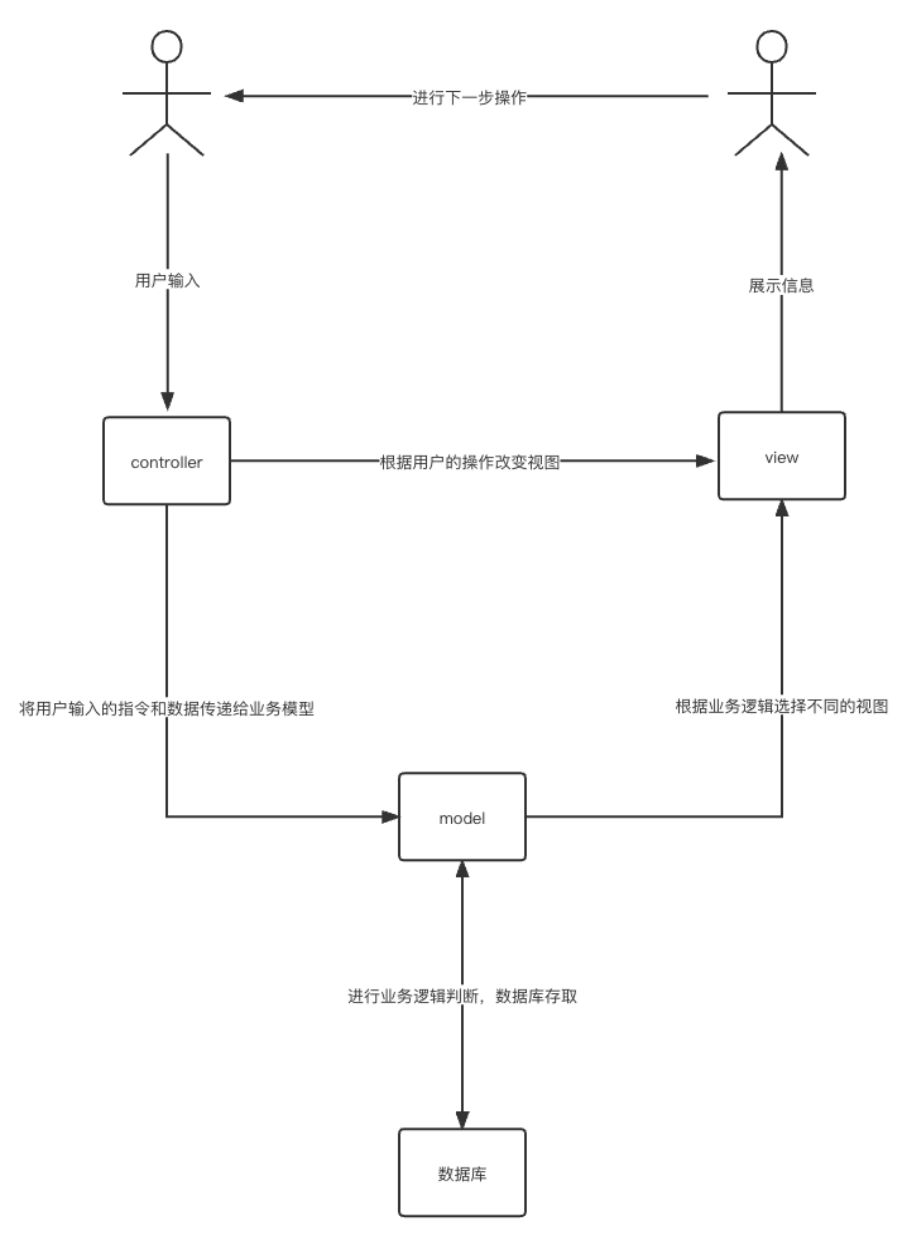


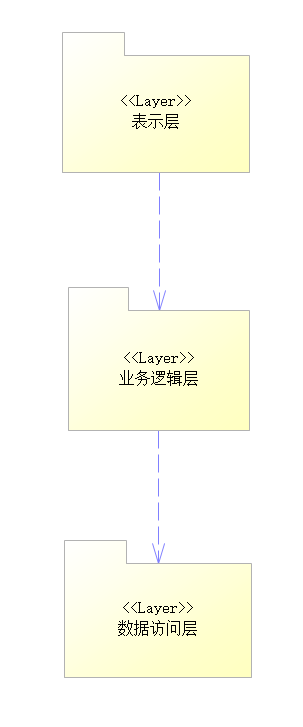
# 逻辑视图

## 概述

逻辑架构说明：使用MVC架构风格。

MVC逻辑架构是一种分层的设计模式，将应用程序分为三个独立的部分，使得各部分之间的耦合度低，易于维护和扩展。在游戏交友网站中，Model层负责管理用户信息、游戏数据等数据，View层负责展示这些数据，Controller层负责处理用户请求和数据操作，使得整个应用程序结构清晰，易于维护和扩展。





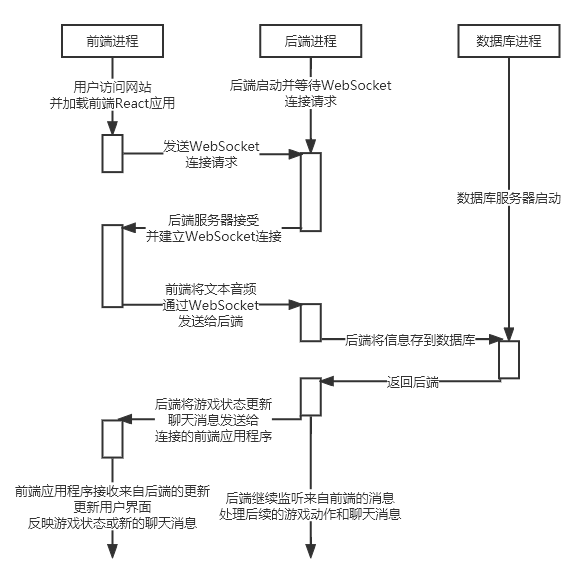
## 在构架方面具有重要意义的设计包

Model：模型层，负责数据的管理和处理，通常与数据库进行交互，包括数据的存储、读取和修改等操作。

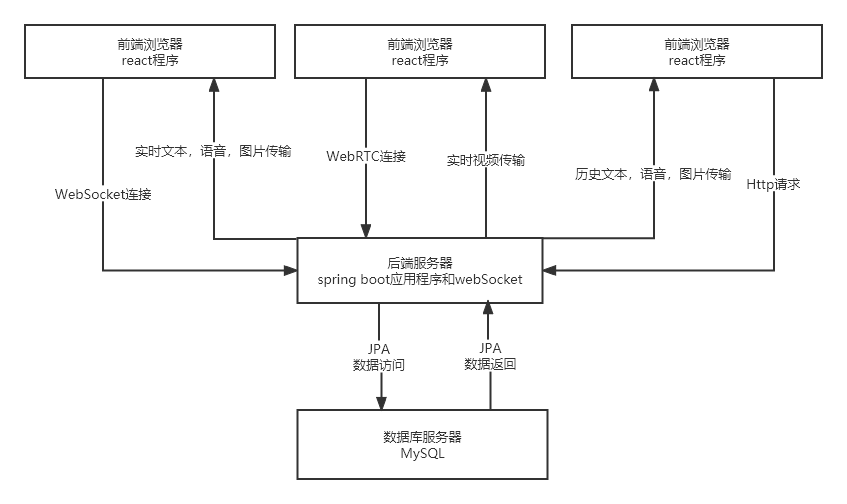
View：视图层，负责将模型层的数据呈现给用户，以及接收用户的输入操作，采用HTML、CSS、JavaScript、React等技术实现。

Controller：控制器层，负责处理视图层的用户输入操作，对数据进行处理和逻辑控制，调用模型层的方法获取或修改数据，并将处理结果返回给视图层。

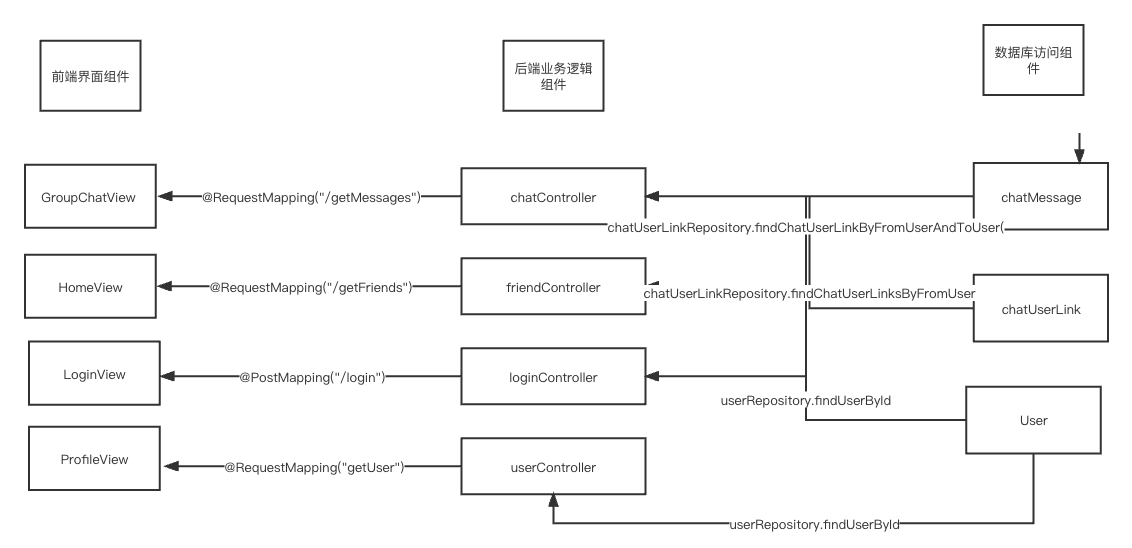
# 进程视图



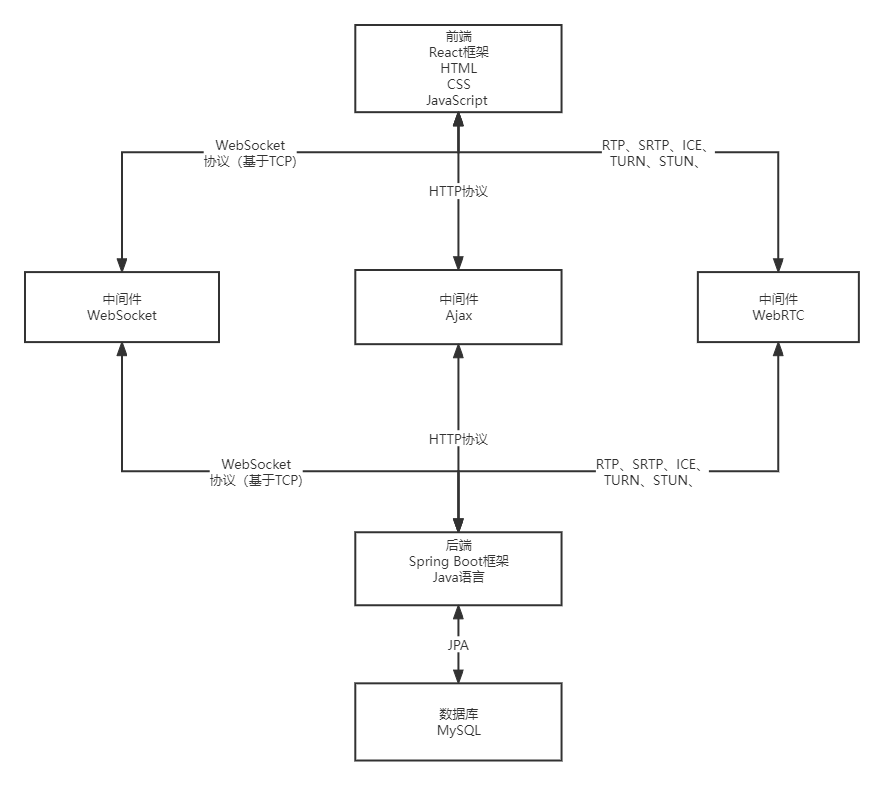
# 部署视图



# 实现视图



# 技术视图



# 数据视图（可选）

暂无

# 核心算法设计（可选）

暂无

# 质量属性的设计

前端模块化开发和后端分层MVC架构对于代码的可移植性、可扩展性和可易用性有以下帮助：

1. 可移植性：

- 前端模块化开发：通过将前端代码划分为模块，每个模块都具有清晰的职责和接口，可以降低代码的耦合性。这样的设计使得前端代码更容易在不同的环境中移植和重用，因为可以根据需要选择性地引入和配置特定的模块。

- 后端分层MVC架构：通过将后端应用程序划分为不同的层（模型、视图和控制器），可以实现业务逻辑与界面层的分离。这样的设计使得后端代码可以独立于特定的UI或其他外部因素，提高了代码的可移植性。

2. 可扩展性：

- 前端模块化开发：模块化的前端设计使得添加新功能或修改现有功能更加容易。通过新增或替换模块，可以以更小的粒度进行开发和维护。这样的设计能够更好地应对需求变化，并且支持团队的并行开发。

- 后端分层MVC架构：MVC架构将应用程序的不同关注点分离开来，使得在不影响其他层的情况下可以对其中一层进行修改或扩展。例如，可以添加新的模型类、视图或控制器来处理新的业务需求，而不会对其他层产生重大影响。

3. 可易用性：

- 前端模块化开发：模块化的前端设计使得团队成员可以更加独立地开发和测试各自的模块。模块之间的明确接口和依赖关系可以减少团队成员之间的冲突和协作成本。此外，模块化的设计也提供了更好的代码复用和维护性，使得开发人员可以更轻松地理解和修改代码。

- 后端分层MVC架构：MVC架构将不同的关注点分离开来，使得开发人员可以专注于特定层的开发。例如，前端开发人员可以专注于视图层，后端开发人员可以专注于控制器和模型层。这样的分工可以提高开发人员的专注度，降低开发的复杂性，并且更容易进行单元测试和维护。