**Socially设计文档**

1. **引言**
   1. **目的**

本设计文档介绍了Social历软件的整体框架与具体设计，旨在为开发或维护人员提供交流参考的标准，同时加快开发进度。

* 1. **文档约定**

本文档采用包图描述整体架构，采用类图描述具体设计。

* 1. **预期读者**

任何开发维护人员均可参考本文档。

1. **整体架构**

本软件采用典型的前后端架构实现，其中后端又将服务器与数据库独立分开，因此整体结构变为：前端-服务器-数据库。下面分别介绍各部分的主要功能与相互连接。

**2.1 前端**

|  |
| --- |
| 功能：与用户的交互以及简单的数据处理 |
| 连接方式：前端直接调用服务器提供的接口，而与数据库没有连接 |
| 典型场景：   1. 用户创建新的日程   2）前端接受数据，进行简单的逻辑判断  3）判断期间通过服务器调用数据库中数据  4）判断通过，加工数据，传输到后端 |

**2.2 服务器**

|  |
| --- |
| 功能：数据的调度中心，负责沟通前端与数据库，同时维护数据 |
| 连接方式：对前端提供接口，对数据库调用数据库接口 |
| 典型场景：   1. 前端发送用户编辑的邀请日程信息 2. 后端解析邀请信息，判断相关联的数据的变化情况 3. 数据改写完成后写入数据库 |

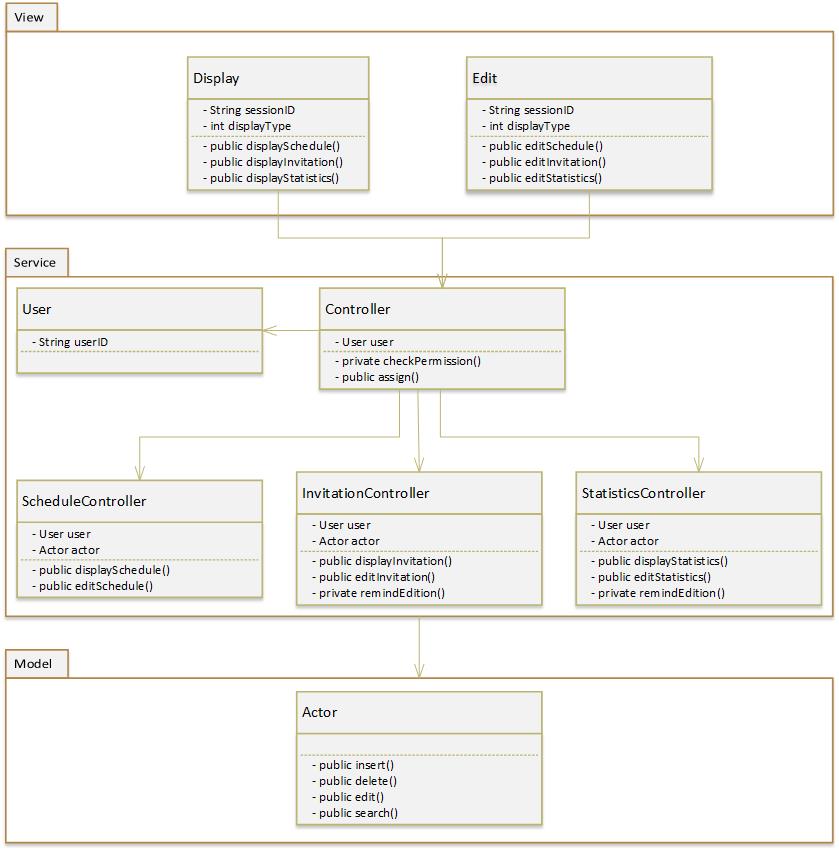
**2.3 数据库**

|  |
| --- |
| 功能：整理储存信息，方便索引查找 |
| 连接方式：向服务器提供接口 |

**2.4 包图**

整体架构以及架构之间的连接关系可以用下面的包图表示。其中，View包代表前端，Service包代表服务器，Model端代表数据库。

设计中的核心元素在于数据库中的Controller类，Controller类本身并不实现具体的功能，它充当了后端对前端的统一接口，从而在最大程度上解耦前后端的连接；同时它有效地组织本软件的多个功能，使得功能之间相互独立，方便之后的修改或添加。



**3具体设计**

* 1. **类图设计**

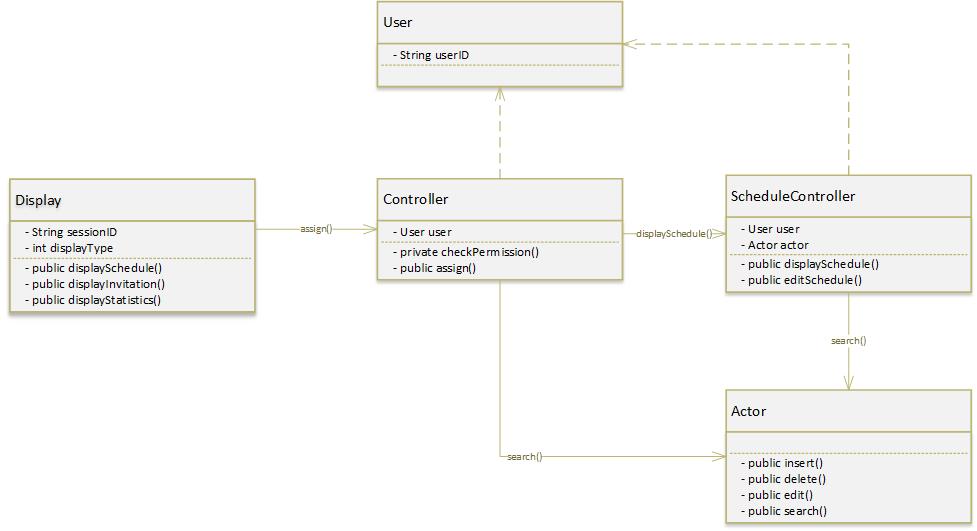
下面使用两张典型的类图来说明类之间相互关系。

**3.1.1 界面展示**

这张类图展示的是用户信息获取的流程中，类之间的相互关系：

* 前端接收信号后调用Display类中的displaySchedule方法
* 该方法中将信息传递至后端，由后端调用Controller类中的assign方法
* Controller根据信息的类型选择对应的ScheduleController类，调用其中的displaySchedule方法
* ScheduleController调用数据库接口，获取相关信息并返回

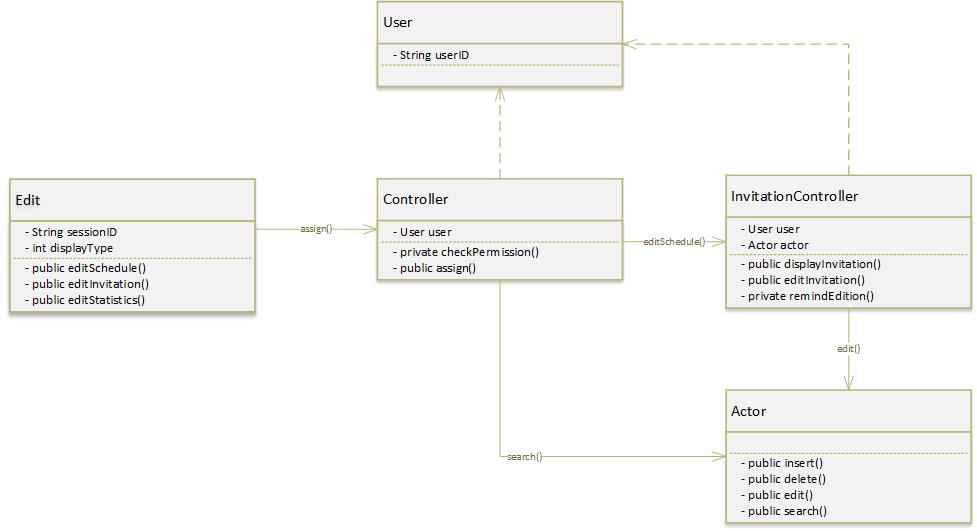
其中Display，Controller，ScheduleController与Actor类构成主要的关联关系，而Controller与ScheduleController类均依赖于User类中的信息。



**3.1.2 信息修改**

这张类图展示的是信息的修改流程，由于良好的结构设计，所有功能的替换只需要修改Controller调用的功能类，而具体的框架则不需要改变，图中的变动如下：

* 前端改变了首次调用的方法
* Controller调用的类变为InvitationController，其中InvitationController不仅要修改个人的信息，还需要在返回信息，提醒邀请人通知被邀请人

这样设计的好处在于功能的修改添加非常简便，不需要修改中间的核心框架，也不会涉及到其它功能的变化，整个软件各部分相互独立。

**3.2 UI设计**