# 详细设计

1. 概述
   1. 系统简述

本即时通讯系统运用分层架构与数据结构，通过WEB端口完成数据转发功能，在不同客户端间实现基本的数据通信功能。是一个用于方便用户聊天、通讯的桌面应用程序

* 1. 软件设计目标

本软件的设计目标是最终实现一个具有良好用户交互体验和基本通信功能的交流软件。具体包括：用户注册与管理功能、个人信息管理功能、好友系统与实时聊天功能（仅图片）。本软件主要用于研究性学习而非实用，对安全性的要求不严格。在性能方面，应保证正常实用状态下无明显卡顿，并将信息传递与展示的时延控制在合理范围内

* 1. 参考资料

《Applying UML and Patterns》

《Software Architecture in Practice》

* 1. 修订版本记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本修订历史记录 | | | | | |
| 序号 | 修订内容 | 修订人 | 修订日期 | 版本号 | 备注 |
| 1 | 概述  与界面设计 | 何乾坤 | 2018-10-31 | 0.1 | 原型制作：  高语真 |
| 2 | 对象模型  初版绘制 | 许家乐 | 2018-11-2 | 0.2 |  |
| 3 | 顺序图绘制 | 许家乐 | 2018-11-4 | 0.3 |
| 4 | 对象模型修改  ER模型绘制  文本增补 | 何乾坤 | 2018-11-5 | 0.4 |
| 5 | 非功能设计 | 何乾坤 | 2018-11-6 | 1.0 |
| 6 | 图表增补与修改 | 何乾坤 | 2018-11-27 | 2.0 |  |

1. 术语表

（对本文档中所使用的各种术语进行说明。如果一些术语在需求规格说明书中已经说明过了，此处不用再重复，可以指引读者参考需求说明。）

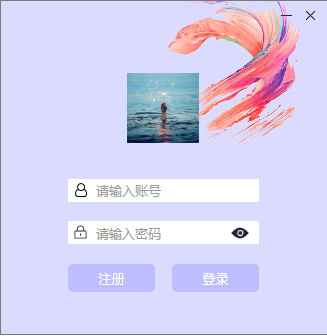
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. 设计概述
   1. 系统的复用计划

暂无复用计划

* 1. 系统接口设计
     1. 界面设计
        1. 登录界面：

登陆界面提供账号密码输入栏，显示账户对应的头像；并设计对应按钮执行登陆功能或转入注册页面



* + - 1. 注册界面

用户可以在注册界面输入注册账号的相关信息，例如用户名、密码、需要绑定的邮箱等。用户可以手动查看输入的密码。系统在执行成功注册后返回登陆界面



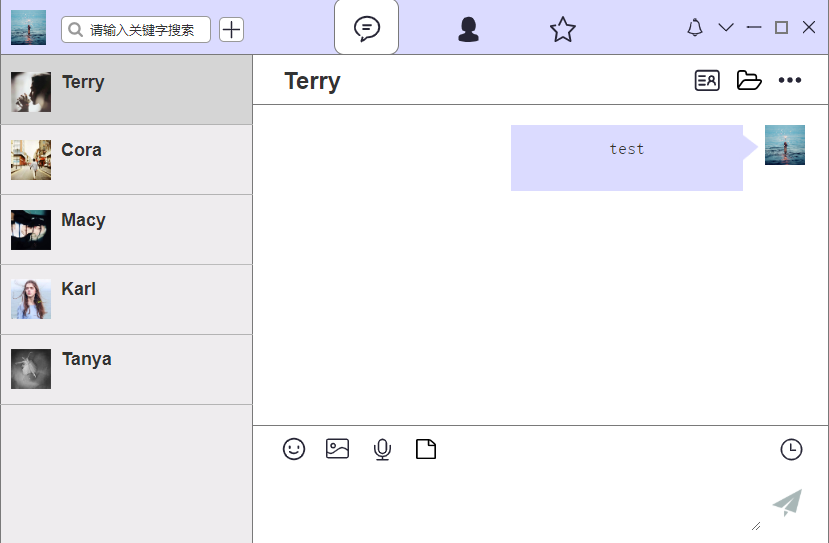
* + - 1. 主页面

主页面由顶部功能区以及下方嵌套的三个子页面组成，用户通过功能区的按钮在三个子页面间进行切换以使用所有主要功能。功能区还另外包含用户头像显示、搜索栏——意图实现统一的搜索入口。整合联系人、群、聊天记录的搜索；添加按钮——用于添加联系人或群；通知铃——整合系统提示；菜单键——转向通用设置页与注销登录。以及基本的页面控制控件



1. 聊天框子页面

聊天框子页面的左侧固定显示会话列表，用户可通过点击某一个会话在右侧打开对应的通信页面，执行基础的文字收发功能。其它设计中的功能包括表情收发、图片收发、语音收发、文件收发等



1. 联系人子页面

联系人子页面左侧固定显示联系人列表，按照姓名首字母排序。用户可通过点击某一个联系人在右侧查看对应的信息，且可直接转入通信界面



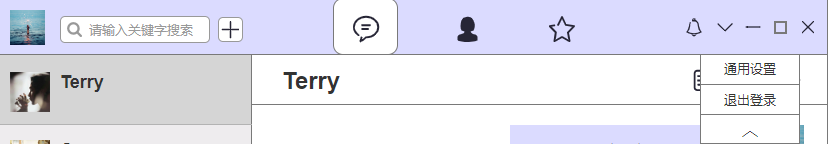
1. 工具箱子页面

工具箱子页面整合个人资料、我的群聊等功能。用户可以在个人资料页查看昵称、地区与账号等信息，并可对个人资料进行编辑。其它设计中的功能包括收藏夹、我的文件等



* + - 1. 通用设置页面

用户可以在菜单页面中选择进入通用设置页。并在其中调整一些交互特性、产品功能已经提供反馈等。此页面尚在设计中





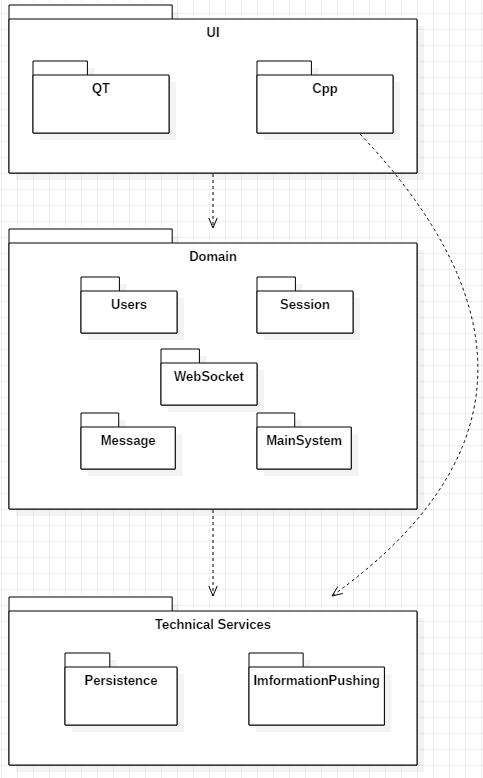
* + 1. 系统外部接口设计：与外部系统的交互设计

系统与外部系统暂无交互

* + 1. 系统内部接口设计：各子系统、各模块间的接口设计

内部接口设计见下文对象模型详述

* 1. 对象模型设计

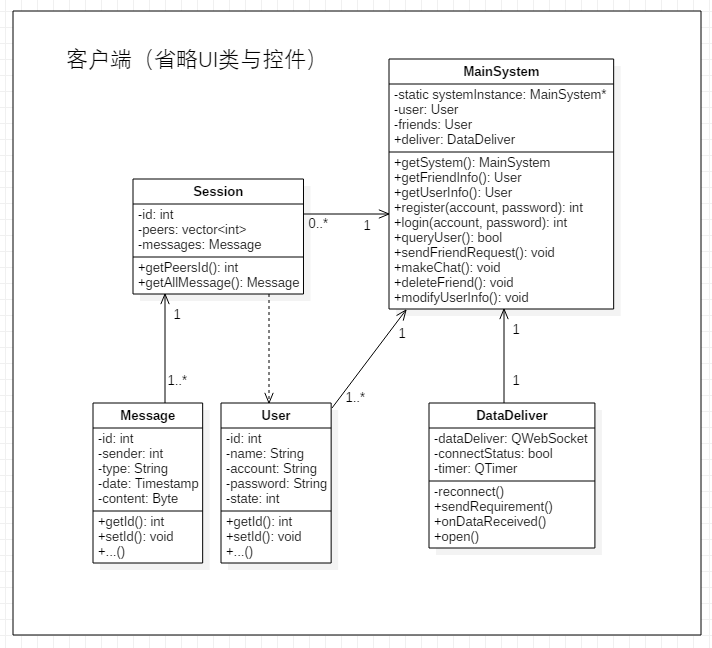


系统层级设计如上图，主要分为3层：UI层、领域层与技术支持层。

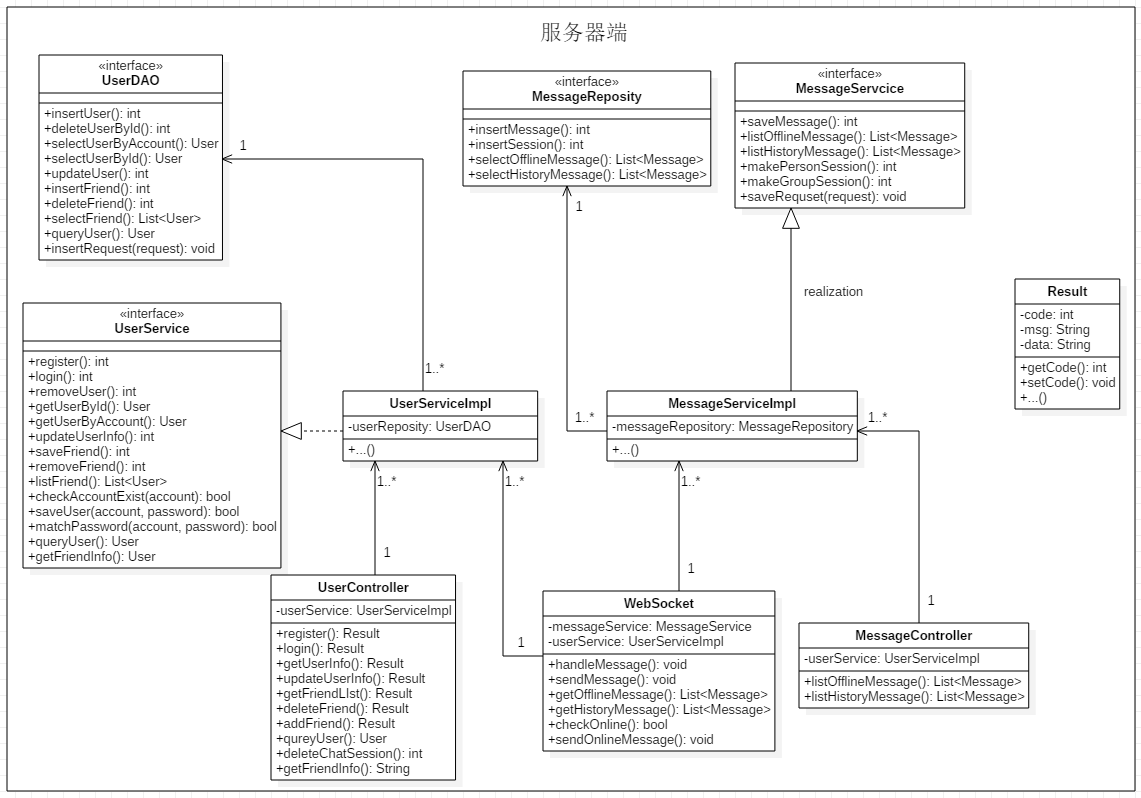
UI层主要通过QT框架、以C++语言实现，主要用于将从服务器取得的信息合理的呈现给用户

领域层揭示了本系统的功能核心：处理几个平行模块——用户、会话与消息间的相互关系，通过WebSocket实现客户端-服务器模式的通信。以及实现一定程度的错误处理功能

技术支持层主要包含信息的持久保存与实时消息保存于推送，这一部分的实现主要依托于所使用的数据库技术（mybatis）与服务器端技术



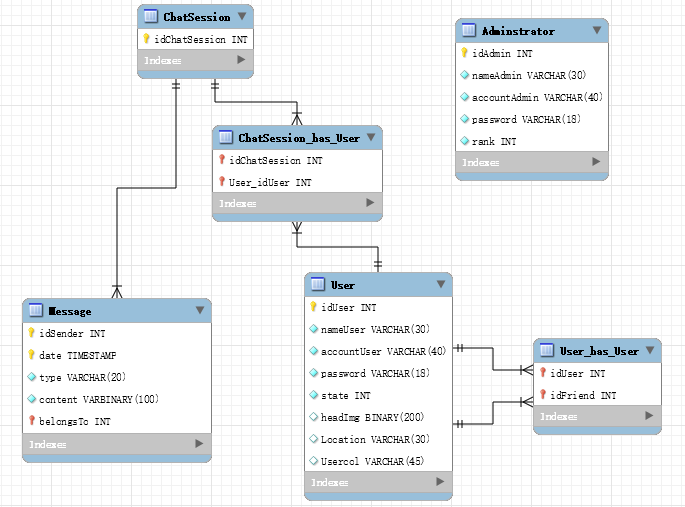
客户端类图如上：包含单一实例类MainSystem，这个类与其他几个实体类：Session、Message、User与DataDelive的数量关系如图。



服务器端类图未全部画出。以最基本的用户与消息为例。图1中一个User类在实现上分为4个部分。

1. UserDAO：这个类中的函数用于直接对数据库进行操作，并将信息返回至上层主要使用SQL语句；
2. UserService & UserServiceImpl：接口实现类中的函数用于处理基础的事务逻辑，实现诸如移除好友、登陆等的功能；
3. UserController：这个类存在的最主要目标是增加一层间接性，利于逻辑分离，它的函数功能最接近用户请求，可能是Service中事务逻辑的组合。在实现上，Controller层位于服务器端的最上部，负责接受客户端的请求并分发给对应的Service
4. 信息通过websocket类返回至客户端

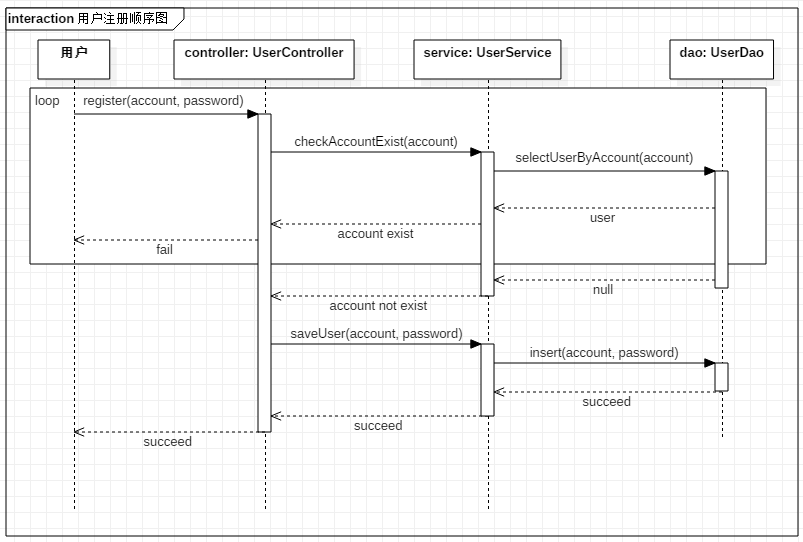
其他实体类实现逻辑与上图类似



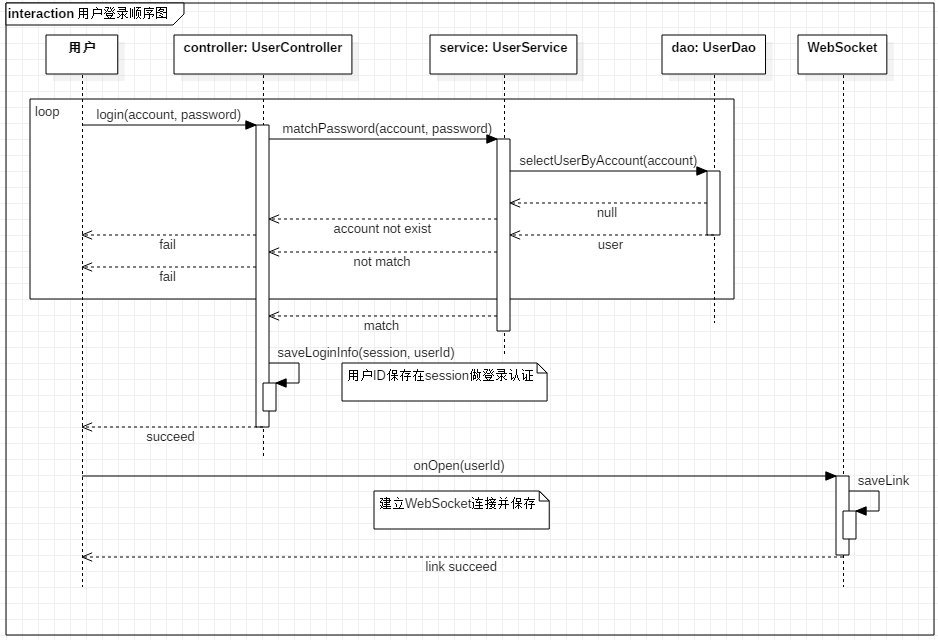
数据库设计如上图。构建满足基本的主、外键约束以及数量关系的模型，生成具体的数据库模式。黄色叹号为主键，红色叹号为有外键约束的主键，青色菱形表示该字段不可为空。实体集中，连线的叉状末端表示该实体在两者关系中可能出现多个对应项，连线的双横线末端表示该实体在两者关系中仅有出现多个对应项

* 1. 系统用例实现详细设计

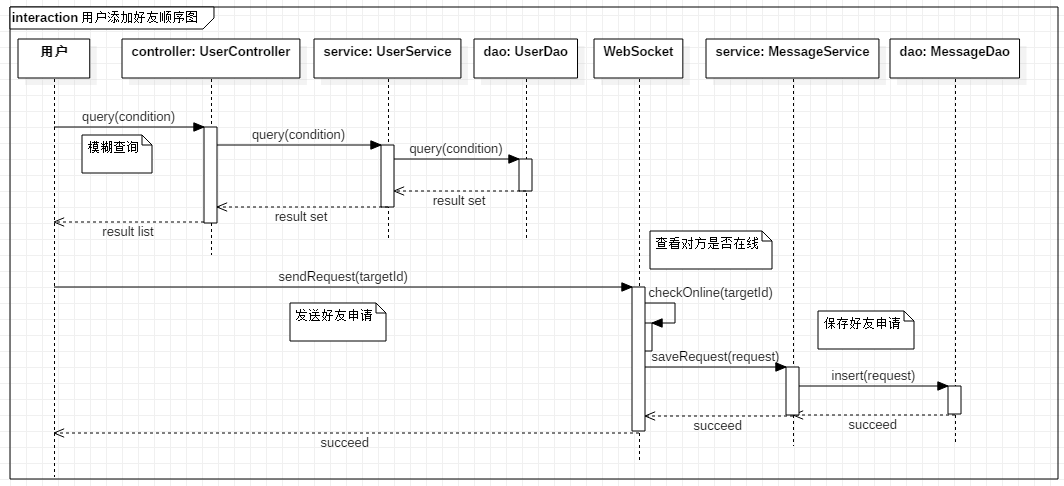
1. 用户注册顺序图



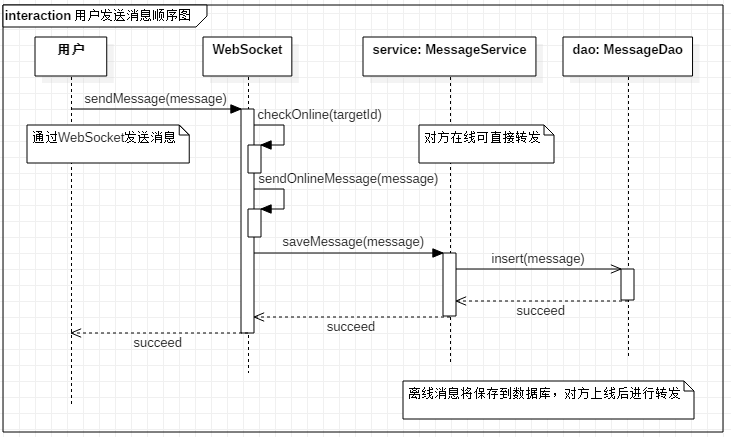
1. 用户登陆顺序图



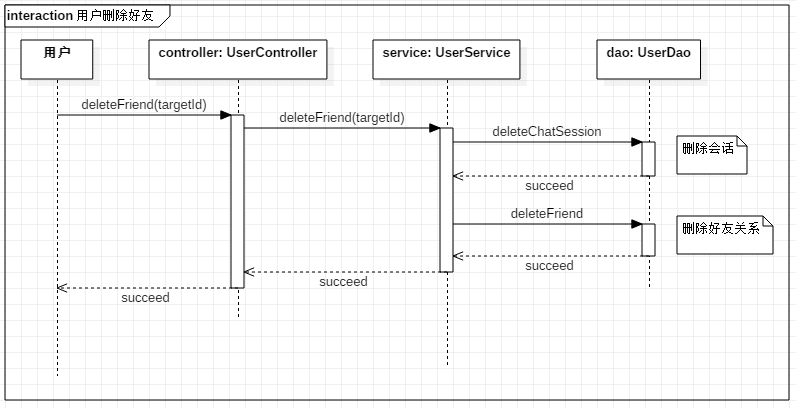
1. 用户添加好友顺序图



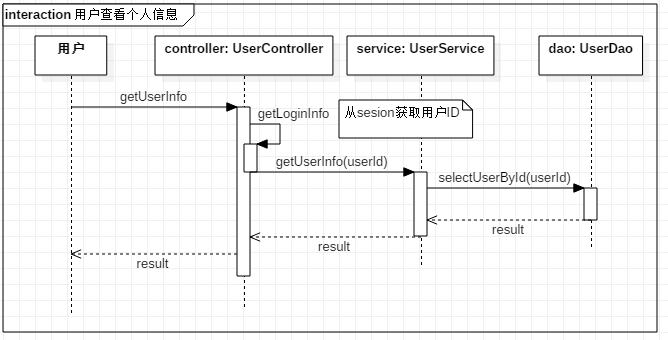
1. 用户发送消息顺序图



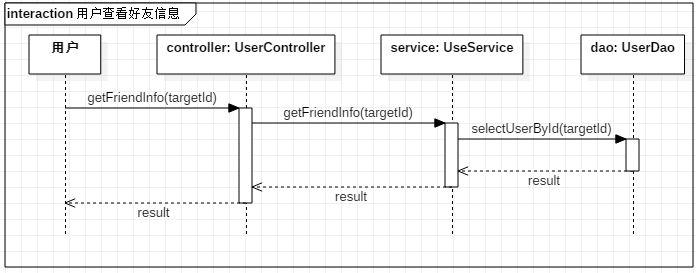
1. 用户删除好友顺序图



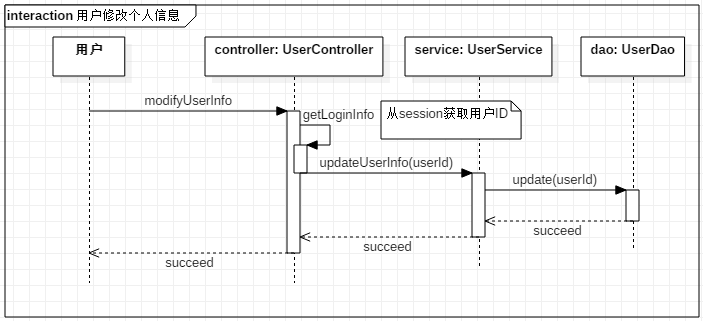
1. 用户查看个人信息



1. 用户查看好友信息



1. 用户修改个人信息



* 1. 系统非功能设计
     1. 可用性
     2. 可靠性
        1. 安全性

需求：系统应该保障用户账号密码在网络传输中的安全性。

设计：对密码进行加密处理。

* + - 1. 容灾能力

需求：系统应该具备基本的容灾策略来保证服务的正常提供。

设计：如有必要，计划使用高可用性的物理部署方案

* + - 1. 性能

1. 实时性

需求：系统需要保证时间敏感类信息的实时转发。

设计：通过较为稳定的TCP连接（WebSocket连接）转发信息。

1. 数据传输格式

需求：系统的数据传输格式应该便于团队开发、调试，同时对系统运营有帮助。

设计：在开发前期，为了方便调试，使用简单的文本协议、JSON等数据格式，后期生产部署，出于流量等考虑，可转用Protobuf等体积更小的协议