MY PC Logo

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <8/11/2020> | <1.0> | 软件架构文档第一次迭代 | 谢厚、蒋钊、焦明胜、郑世民 |

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 5

3.1 概述 5

3.2 Application层 6

3.3 Business Service层 6

3.4 Middleware层 7

4. 进程视图 7

5. 部署视图 8

6. 实现视图 8

6.1 整体架构图 8

6.2 双人绘图结构图 9

7. 数据视图 10

8. 核心算法设计 10

8.1 命令行绘图 10

8.2 双人合作绘图 10

9. 质量属性设计战术 10

9.1 性能战术 10

9.2 可修改性战术 10

9.3 易用性战术 11

9.4 可测试性战术 11

软件架构文档

# 简介

## 目的

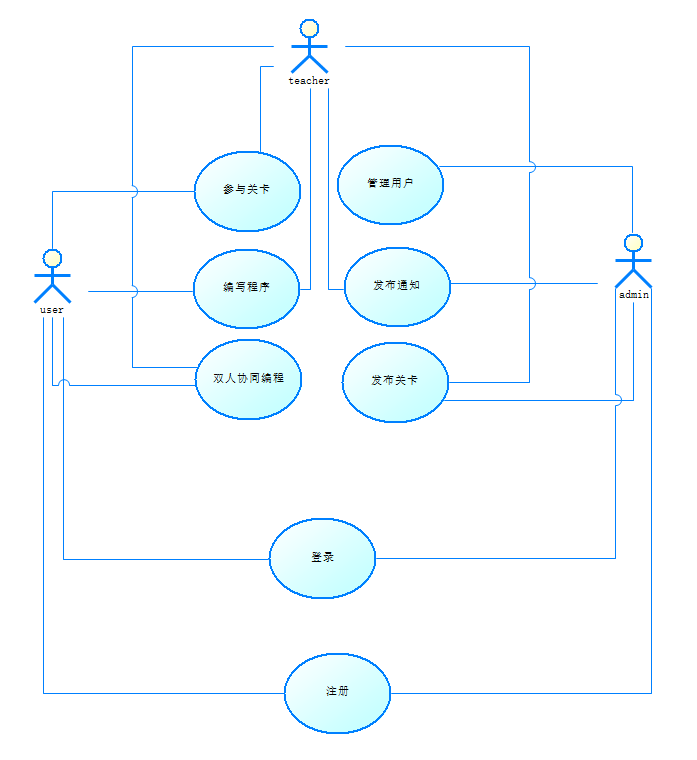
本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。并且将用于之后项目开发的基础和规范，为软件架构提供框架和重用组件，为未来业务提高效率，并提供高可用高性能，可扩展的架构设计。

## 参考资料

《软件工程原理》

选课系统软件架构文档

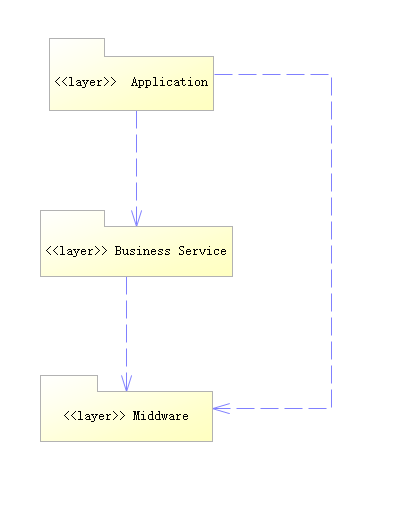
# 用例视图



# 逻辑视图

本节是对软件架构的逻辑视图的描述。主要内容包括描述重要的类，类的分包，子系统以及子系统 的分层等。另外还包括一些重要用例的实现，在构架方面具有重要意义的设计包。

## 概述



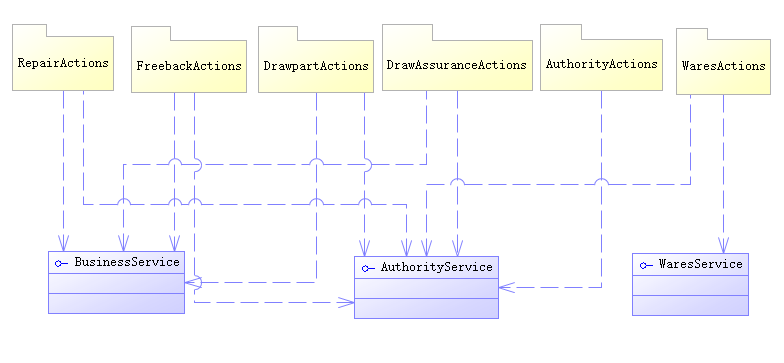
逻辑视图主要由三层组成，分别是Application层，Business Service层和Middle层。

Application层主要响应各种各种用户界面请求的动作类组成，它会调用Business Service层中的函数进行业务逻辑处理，同时根据结果显示不同的界面给用户。

Business Service层主要完成实际的业务逻辑，同时包括与数据库的表对应的实体类，以及访问数据库的DAO层。

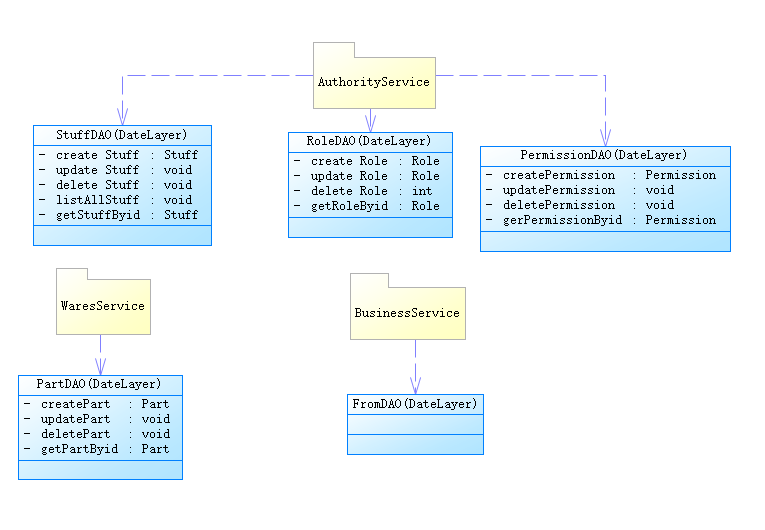
Middle层为数据库。

## Application层



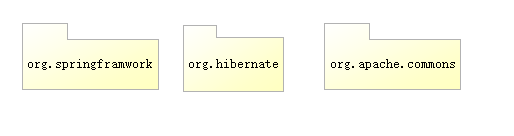
Application层主要由6个子包组成，这6个子包分别是RepairActions包，FeedbackActions包，DrawpartActions包，DrawAssuranceActions包，AuthorityActions包和WaresActions。

## Buiness Service层

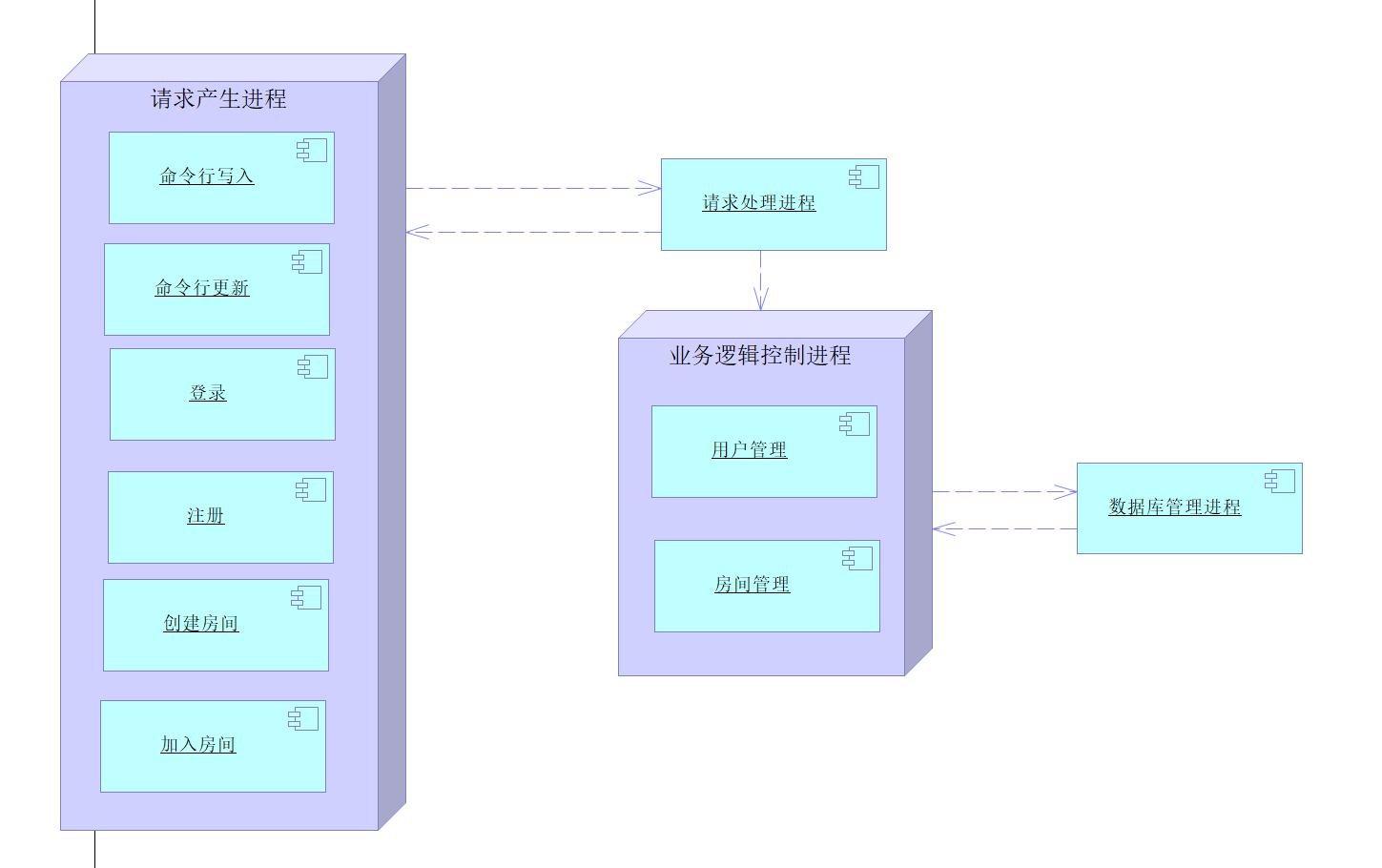


Service包主要由三个包组成，AuthorityService包负责处理用户的身份认证以及角色的权限管理，BusinessService包负责绘图功能以及绘图功能过程中的反馈处理，WaresService包负责所绘制图案的管理存贮以及用户信息的管理存贮。

## Middleware层

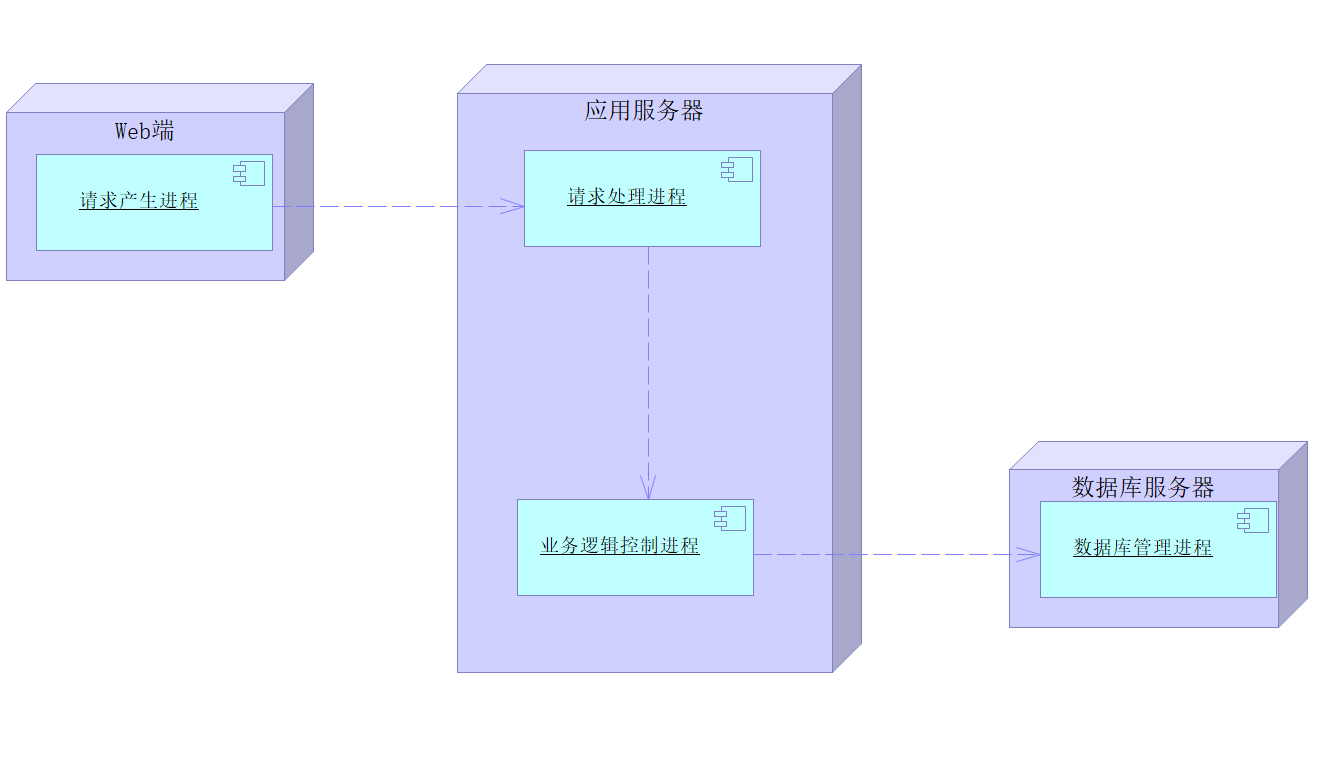


# 进程视图



Web端进程与服务器通过异步I/O通信，请求处理进程和业务逻辑处理进程通信在服务器端内核中完成，与数据库管理系统进程通过阻塞I/O通信

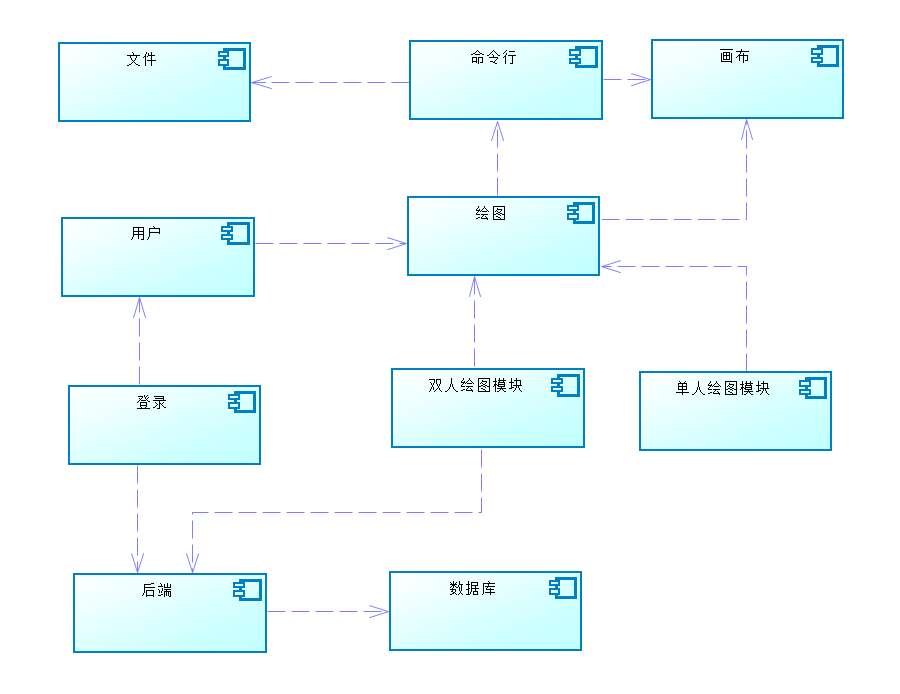
# 部署视图



My PC Logo采用传统的Client/Server Architectures 架构。Server端分为应用逻辑层和数据库层。因为PCLogo并发访问不会很大，所以将请求处理进程和业务逻辑控制进程部署在同一台服务器上。用户端通过向应用服务器发送请求，应用服务器处理请求并完成逻辑控制，由应用服务器端和数据库服务器端进行交互获取数据并返回给Client端。Web端，服务器端通过互联网连接，应用服务器和数据库服务器为LAN连接。进程试图中的请求产生进程映射到web端，请求处理进程和业务逻辑控制进程映射到应用服务器，数据库管理进程映射到数据库服务器。

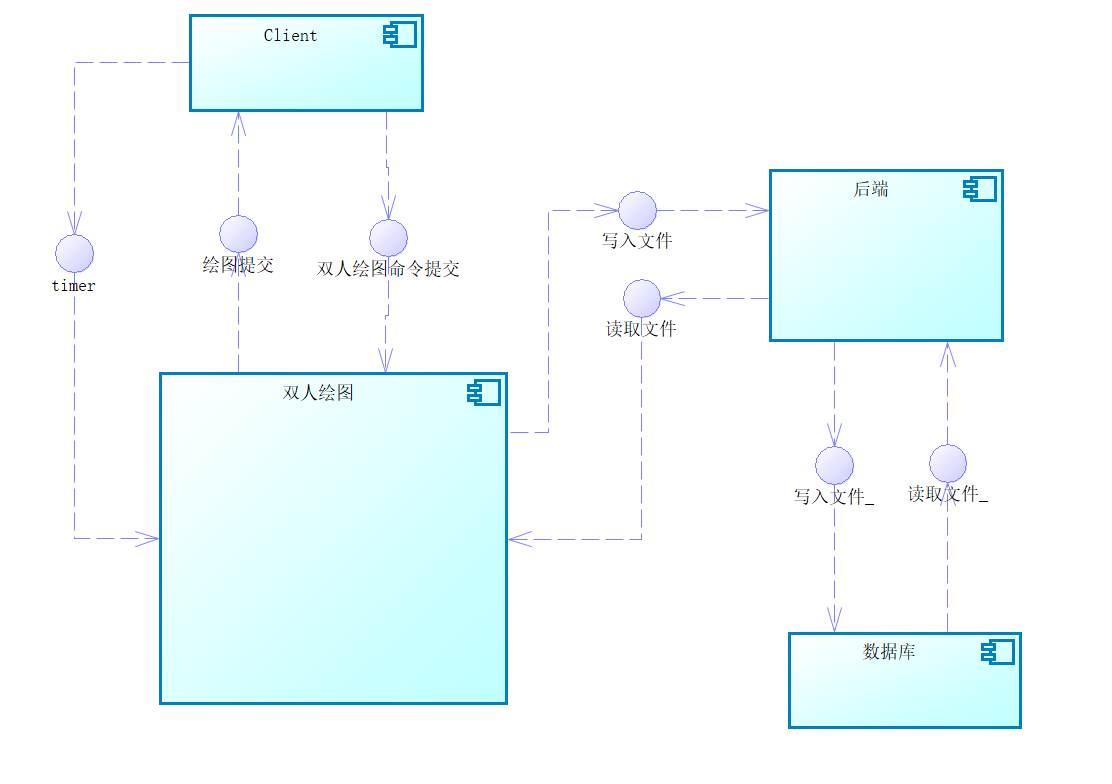
# 实现视图

**6.1 整体架构图**



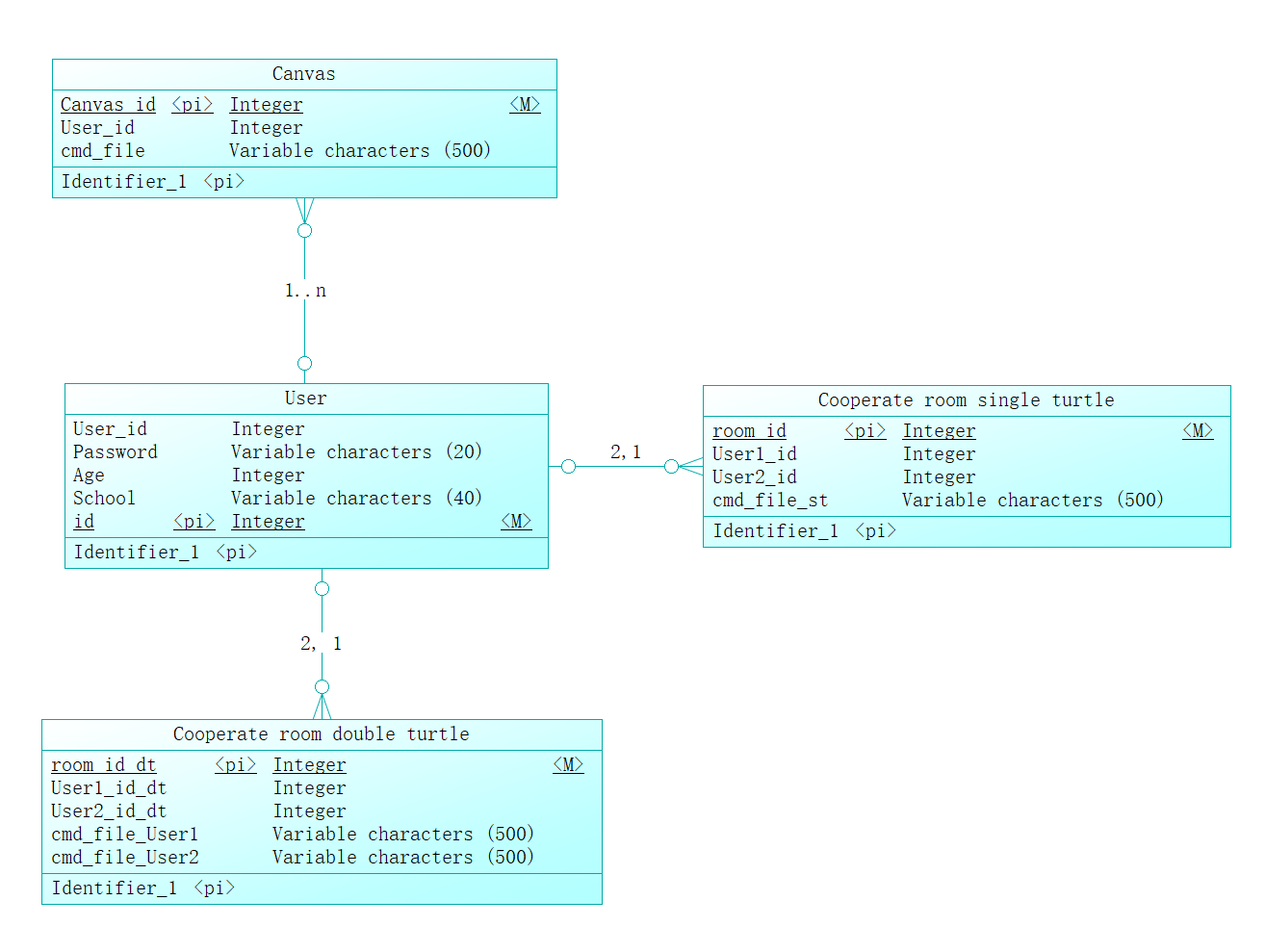
整体架构图包括了项目所含的基本功能模块和它们之间的交互做了描述。基本模块画布和命令行是绘图模块的基础，在此基础上延伸出了单人绘图和双人绘图，单人绘图模块较为简单，而双人绘图模块则需要后端、数据库的参与。

**6.2 双人绘图结构图**



双人绘图的同步依靠数据库中的文件和timer来实现，当客户端A的绘图命令在命令行提交后，客户端A会向双人绘图模块发送命令提交请求，经由后端将该命令写入到数据库文件中，而后读取数据库中的命令行文件，返回到客户端A执行绘图提交。timer控制客户端每隔一定时间向后端发送请求，后端将数据库中的双人绘图命令发送到客户端B，在画布中呈现出相应的图形，用于两个客户端上的画布同步。

# 数据视图



User实体用于存储用户，canvas实体用来实现单人绘图模式，Cooperate room single turtle和cooperat room double turtle实体分别用于实现双人单海龟和双人双海龟模式。用户与canvas为一对多关系，一个用户可以对应多个画布作品，canvas\_id为canvas的主键，同时还关联有外键User\_id来代表canvas的作者。一个cooperate room关联两个用户并将两个用户的id作为外键。Canvas均用string存储命令行文件来保存画布。

# 核心算法设计

## 命令行绘图

使用轻量级网页IDE，用解释的方式运行代码并用canvas绘图。

## 双人合作绘图

用房间实体来实现功能，实时监听命令行更新并刷新到web端，解释器找到新旧代码位置并刷新另一端提交的绘图实现交互。

# 质量属性设计战术

## 性能战术

引入并发，优化性能提高计算效率。

## 可修改性战术

模块化编程，尽量解耦，增加可维护性。

## 易用性战术

分离用户接口

## 可测试性战术

管理输入输出，内部监视，维护完备的日志。