<spot-palm>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <2日/5月/2023年> | <1.0> | <无> | <董绍启> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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。

本软件架构文档是spot-palm项目文档中非常重要的一部分，它描述了系统的整体架构和设计，包括各个组件之间的关系、通信协议、数据流和处理流程等。它对于本项目的开发和维护具有重要的指导作用，能够帮助团队成员更好地理解和掌握系统的设计和实现细节，从而提高开发效率、降低开发成本和风险。

软件架构文档的特定读者包括但不限于以下几类人员：

开发人员（小组成员）：了解系统的总体设计和各个组件之间的关系，从而能够开发出符合要求的代码

测试人员：了解系统的总体设计和各个组件之间的关系，从而能够制定合适的测试计划和测试用例；

项目经理：了解系统的总体设计和各个组件之间的关系，从而能够更好地管理项目进度和资源，以确保项目的顺利完成；

运维人员：了解系统的总体设计和各个组件之间的关系，从而能够更好地维护和管理系统。

## 参考资料

《数理逻辑与集合论》——清华大学出版社

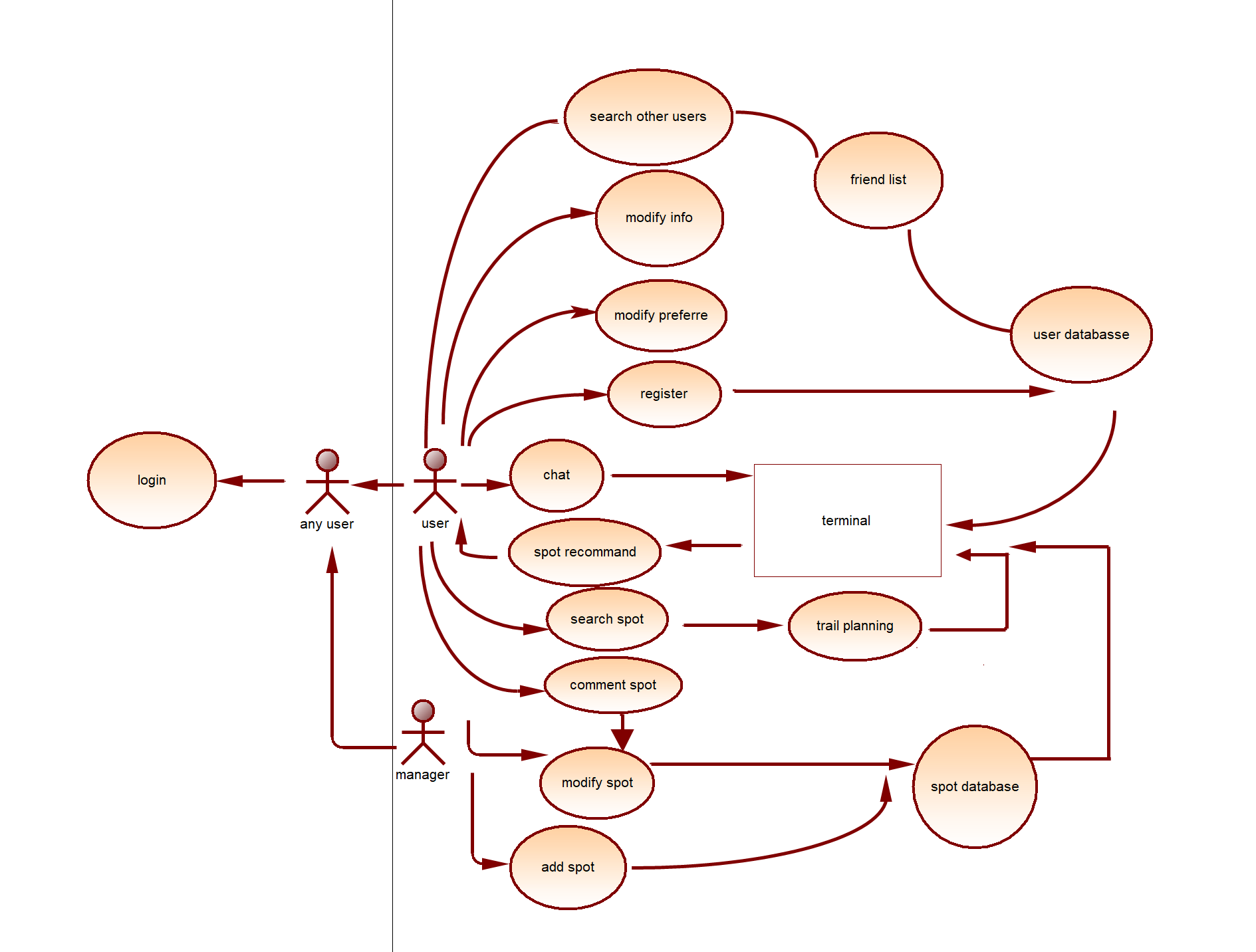
# 用例视图

此用例描述了本产品针对的两类用户，一类是管理者(manager)身份 ：该产品的运维人员；另外一类是该产品的用户(user).

任何用户均有登录功能，除了注册功能外，任何功能均需要在注册后执行。

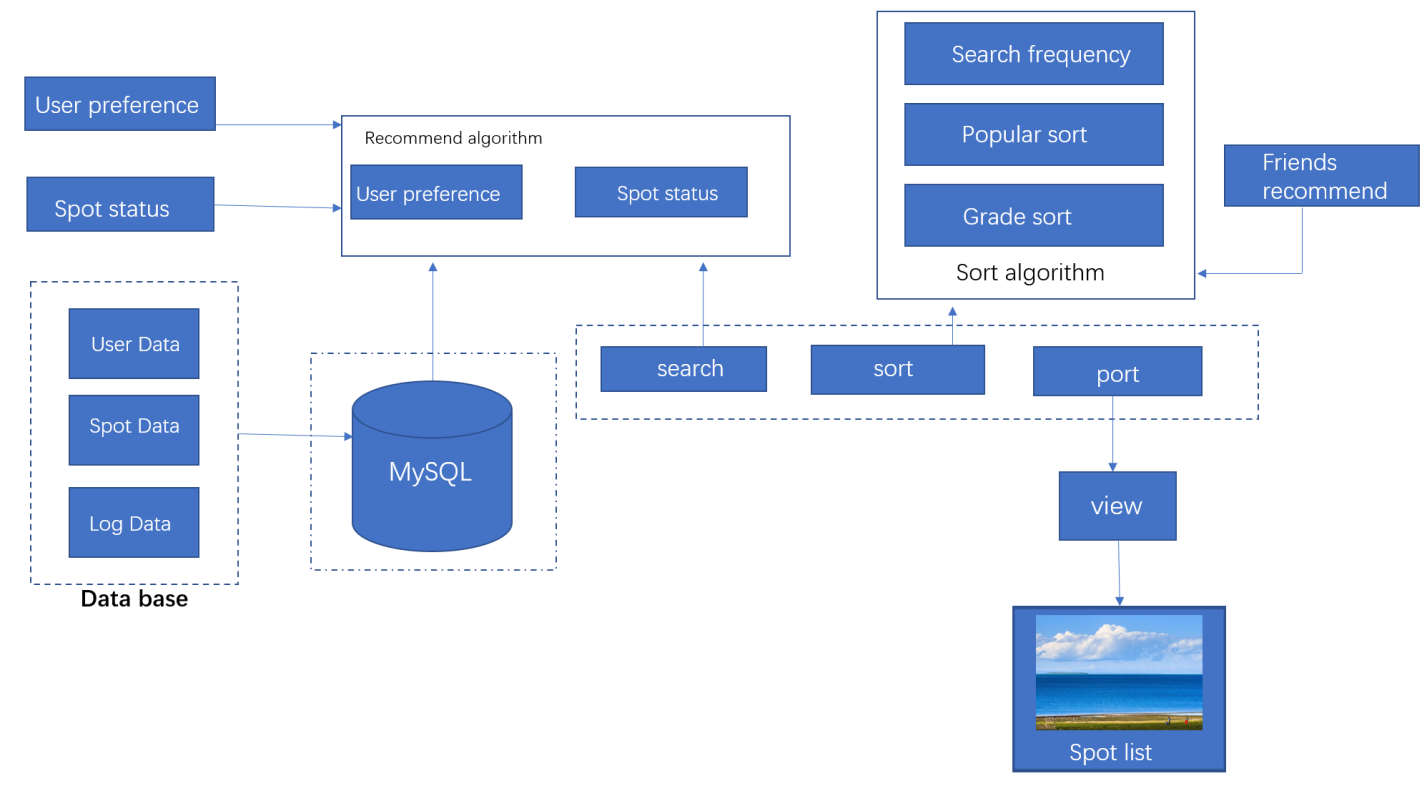
允许用户的操作有 ： 搜索其他用户并且加好友，修改个人信息，修改个人偏好，登陆，好友聊天，针对个人画板的景点推荐，搜索景点，评论景点功能。并且可以和好友组队出行。

允许管理者的操作有 ： 对景点所有信息的增删改查。



# 逻辑视图

## 概述



## 在构架方面具有重要意义的设计包

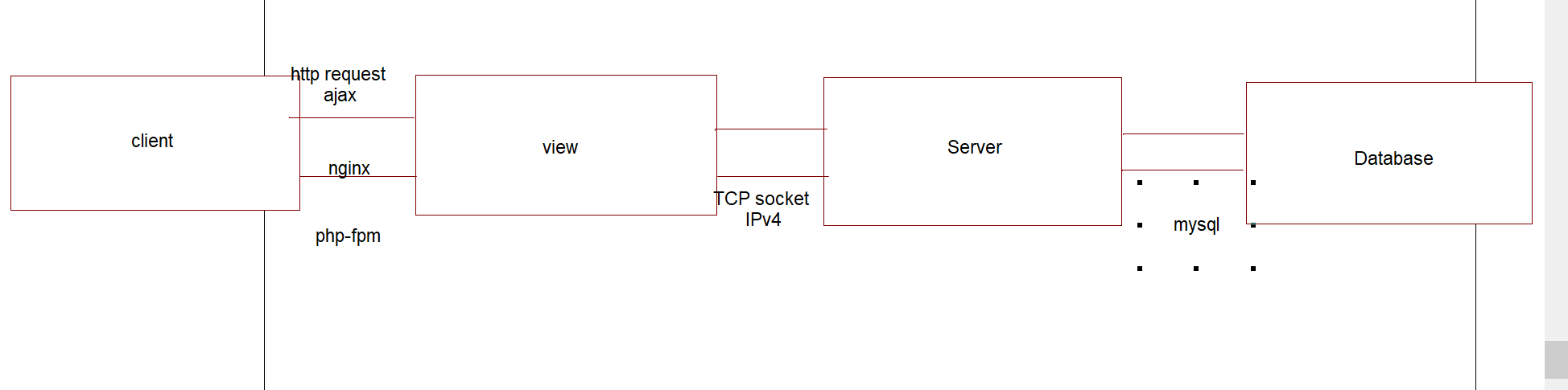
本产品采用MVC架构风格。该架构风格将应用程序的数据、用户界面和控制逻辑分离开，从而使开发过程更加灵活和可维护。本产品的MVC架构具体构成如下：

Model: 模型是应用程序的数据和业务逻辑层。它负责存储、管理和处理应用程序的数据，并提供访问和修改数据的接口。在本产品中，Model层由用户（Spot）、景点（Spot）两大模块构成。用户模块包括用户偏好（preferrence）、用户评论（comments）和用户信息（user info）三个子模块。景点模块中含有景点信息（spot info）、景点评论（comments）和景点状态（spot status）三个子模块。

View：该模块包含在用户端显示的页面，并接收用户的输入，与后端服务器进行交互。前端界面包括：1.主要界面（main page）2.用户界面（user page）3.景点页面（spot page）4.登录/注册界面（log in/sign in page）。

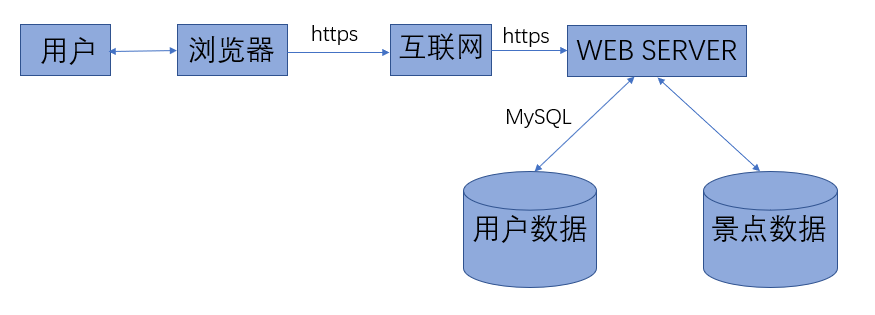
Controller：控制器是应用程序的逻辑层。该模块中有景点管理器（SpotManager）和用户管理器（UserManager），在Model层和View层之间充当中介，用于处理用户的请求并根据需要与数据库进行交互、查询、更新、删除数据模型。景点管理器用于管理景点信息、景点评论、推荐景点；用户管理器用于管理用户信息、添加景点评论、调用推荐算法得到景点推荐。

# 进程视图



用户端和服务器端通过http协议进行消息传递；服务器端收到请求后，将请求通过http协议发送给后端数据库；数据库将处理好的信息再通过http协议原路发回，直到用户端。

# 部署视图

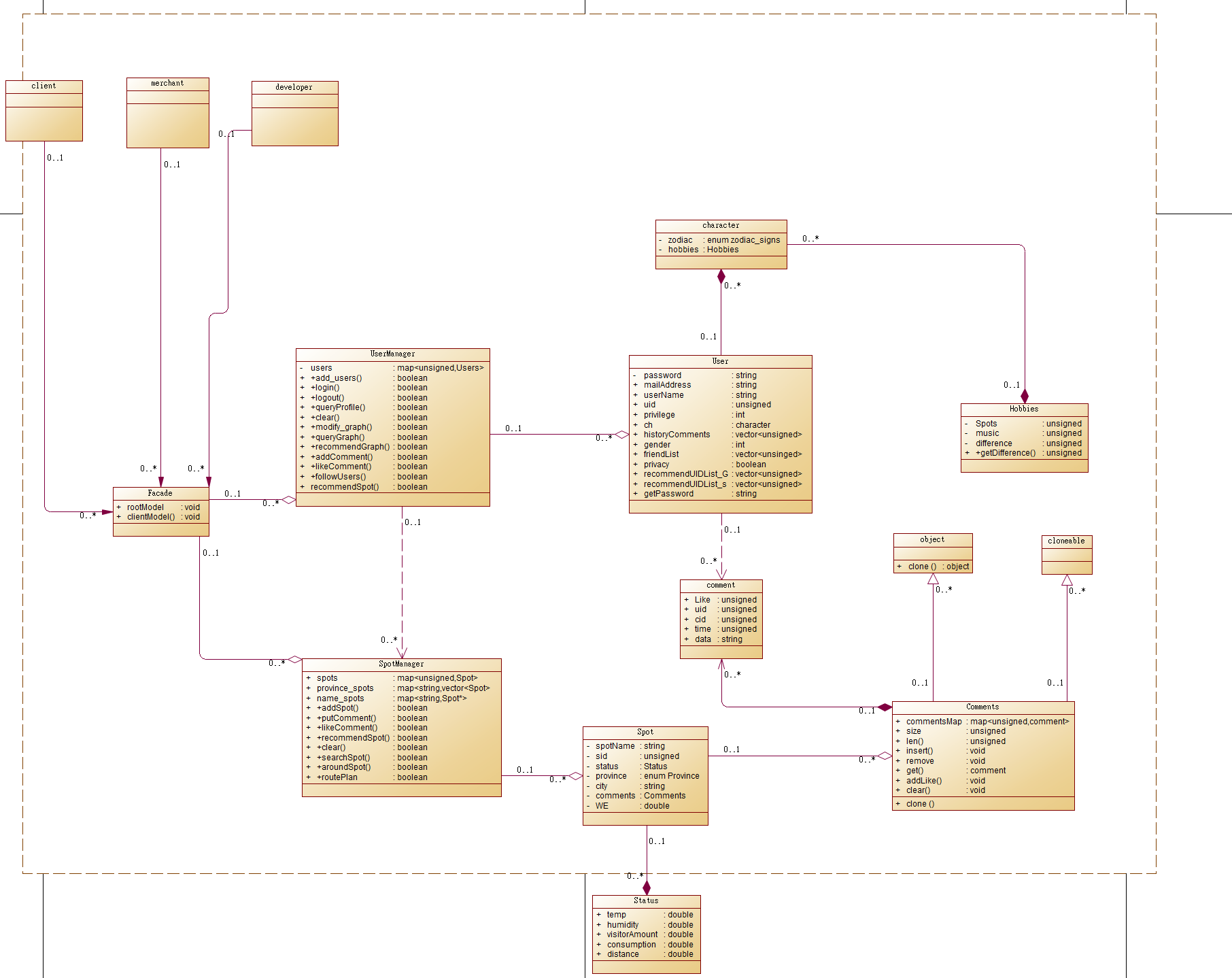


本产品使用B-S物理架构，由用户端、网页服务器、数据库服务器组成。

用户端通过发送https请求到Web服务器，Web服务器申请访问数据库服务器，数据库服务器取出数据传回给Web服务器，Web服务器结合数据和用户信息运行相应的算法，得出推荐结果等信息，传回用户端显示。

Web服务器可以访问外部的REST，调用百度地图，实现产品的地图支持。

# 实现视图



对于三种用户 ：客户（client）， 商家（merchant）， 运维人员（developer）可以使用该产品上传 自己的指令到服务器，通过服务器为用户实现他们需要的功能。

例如客户可以实现注册、登陆、修改个人信息，修改偏好，根据偏好智能推荐好友，模糊查找景点 等... ...

User（用户类）：定义单个用户的所有信息，包含用户名、密码、邮箱地址、朋友列表、历史评论 等。

character（用户特征类）：对用户的习惯信息进行维护的接口

Hobbies（用户习惯类）：定义用户的习惯信息

comment（用户评论类）：对用户的历史评论进行维护的接口

Comments（评论类）：定义单条评论的信息以及对评论进行维护的操作接口

Spot（景点类）：定义单个景点的所有信息，包含景点名、位置、省份、所有相关评论

Status（景点状态类）：定义景点的状态信息，包括湿度、游客流量、距离等

UserManager（用户管理类）：定义所有与用户相关的操作，提供了登录、登出、添加评论等操作的 接口。

SpotManager（景点管理类）：定义所有与景点相关的操作，提供了添加景点、查找景点、周边景点 推荐等操作的接口。

CommandParser（高级指令接口）：分别用一个UserManager和一个SpotManager来配置，是所有 允许的操作的高级接口。例如，当用户发表评论时，façade将调用UserManager中的addCommend （）方法；当软件管理员添加新的景点时，façade将调用SpotManager中的addSpot（）方法。

Façade架构优势：

将用户相关操作和景点相关操作分别封装成不同的接口，再通过统一的高级接口来进行调用和维护， 方便了用户和管理员直接使用相关接口对系统内部的信息进行操作，而不必关心系统架构和实现过 程，同时也方便了软件开发人员对软件内部的相关功能进行更新和调试，而不影响外部的接口使用。

# 技术视图

技术栈选取：

1. 编程语言：
2. 前端：html、JavaScript、php
3. 后端：C++
4. 数据库：sql
5. 开发工具：
6. 前端：vscode
7. 后端：clion
8. 数据库：mysql，datagrip

# 数据视图（可选）

# 核心算法设计（可选）

在本产品中共设计了四个类别（我们称之为推荐信息维度）：温度、距景点的距离、人流量、花费。

对于某个特定的维度，设置了5个模糊集合（我们称之为五个层次）。

通过问卷收集数据，采用模糊统计的方法分别建立了各个维度的数据*x*对应模糊集合的隶属函数

再通过模糊识别：

获得每个元素对每个类别的隶属度，元素属于隶属度最大的类。公式如下，其中u0表示元素，Ai表示某个类别：

由此识别出该数据在某个维度下所属的层次。

# 质量属性的设计

1. 性能：B-S物理架构将通过将应用程序在服务端运行，可以减轻客户端的负担，提高客户端的响应速度。同时，可以采用缓存技术、负载均衡技术和分布式架构，实现高性能、高可用性和高并发访问。MVC物理架构可以采用分层设计和分布式架构，将应用程序划分为多个模块，每个模块可以独立开发和部署，实现模块化设计和快速迭代。
2. 可扩展性：B-S物理架构可以采用分布式架构和微服务架构，将应用程序划分为多个模块，每个模块可以独立部署和扩展。MVC物理架构可以采用分层设计和插件化架构，将应用程序的各个模块和功能划分为多个层次和插件，实现功能的灵活组合和扩展。
3. 可靠性：B-S物理架构可以采用分布式存储和备份机制，保证数据的可靠性和持久性。MVC物理架构可以采用分布式存储和备份机制，保证数据的可靠性和持久性。
4. 易用性：B-S物理架构可以采用友好的用户界面和交互设计，提供简单明了的操作流程和提示信息。MVC物理架构可以采用友好的用户界面和交互设计，提供简单明了的操作流程和提示信息。
5. 可移植性：B-S物理架构可以采用跨平台开发技术，实现Web和移动端的共用。