数字工程软件架构

——地图数据质检工具（WEB）

学 院：遥感信息工程学院

班 级：2006 班（20F10）

姓 名：马文卓

学 号：2020302131249

1. 概述

本次项目为 web 版的地图质检系统,其基本功能基本与之前的局域网地图质检系统类似,在此就不过多赘述,区别主要在于本次项目为开发一个互联网环境下的网络应用。

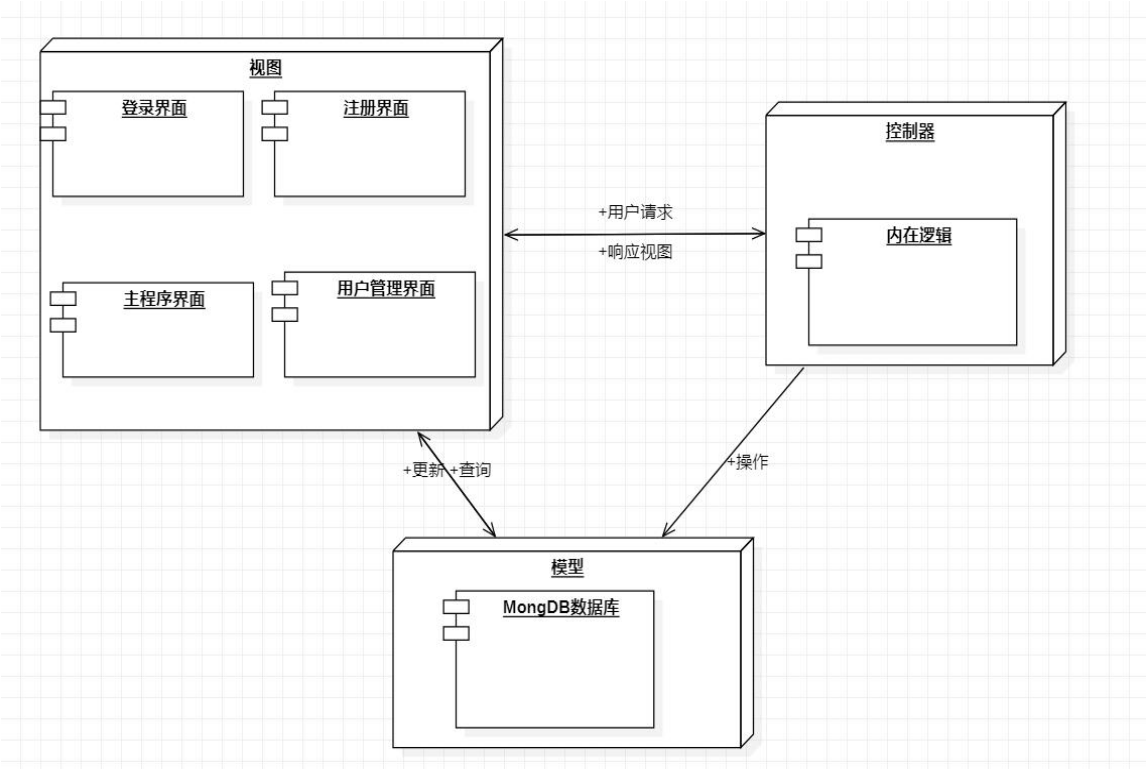
作为一个网页应用(由于没有要求具体实现,所以具体的可行性还有待参考,下面只是提出一个大的设计框架和思路),经过搜索和学习,我最后决定采用的技术如下:

前端: vue(一套用于构建用户界面的渐进式框架)+elementUI(一套基于 vue 的桌面组件库)+socket(可以实现 HTTP 的套接字工具)

后端: spring boot(开源应用框架)+MongDB(一种非关系型的数据库)+spring security(一个能够为基于 Spring 的企业应用系统提供声明式的安全访问控制解决方案的安全框架)

整体的架构风格为 MVC 风格,前端 vue 还采用了 MVVM 的模式,整个系统的运作体现了观察者模式。前端各种构件使用 elementUI,前后端之间的交互使用 socket 编写的 HTTP 实现,后端 spring boot 作为整体框架,MongDB 作为数据存储地存储用户信息和用户上传的图像\文本\错误文件等数据,并且使用 spring security 实现用户登录的验证和安全保证。

2. 整体风格(MVC)

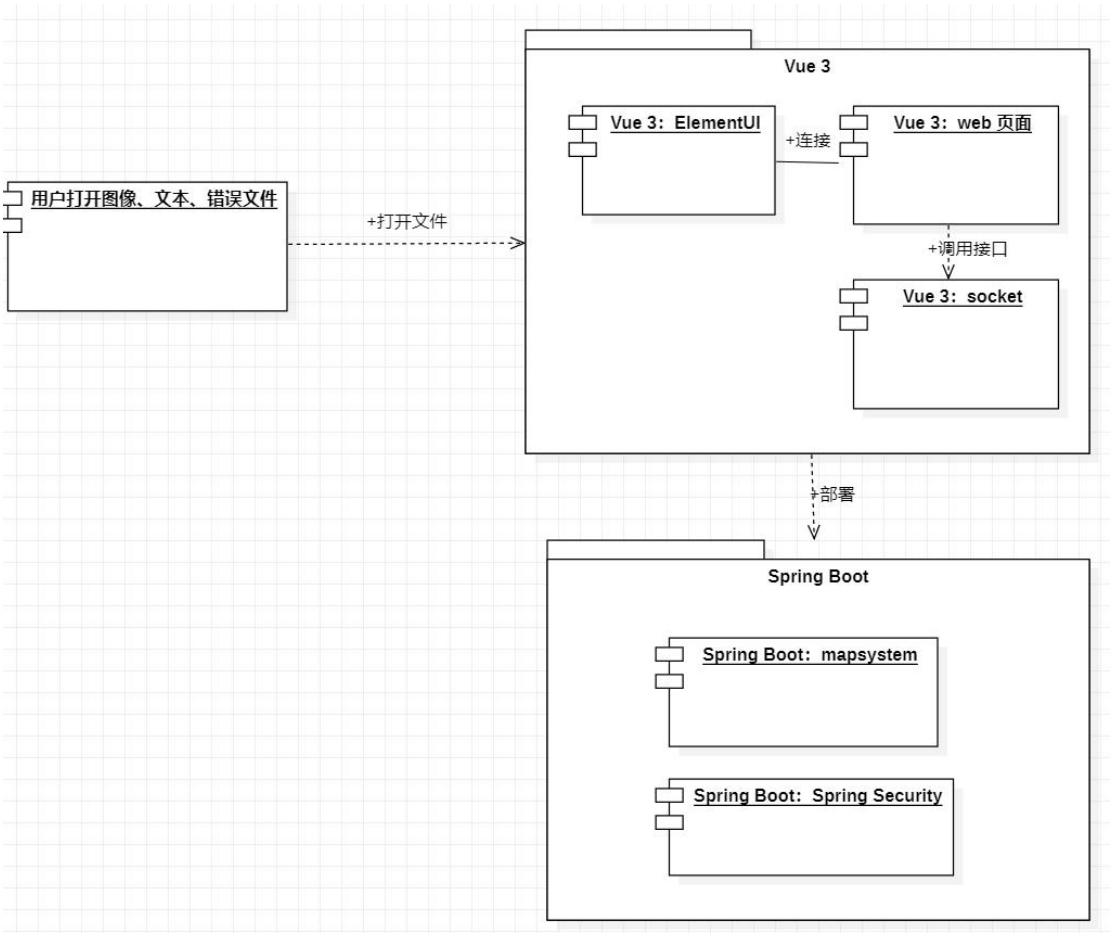


(1)概述: 本项目是典型的在 MVC 风格下的实现。视图:包括客户端网页的登录界面、注册界面、主程序界面和服务端的用户管理界面;模型:存储数据的 MongDB 数据库;控制器:封装了监听模型数据的改变和控制视图行为、处理用户交互的操作。它们三者相互联系、相互作用,实现了整个系统的功能架构。

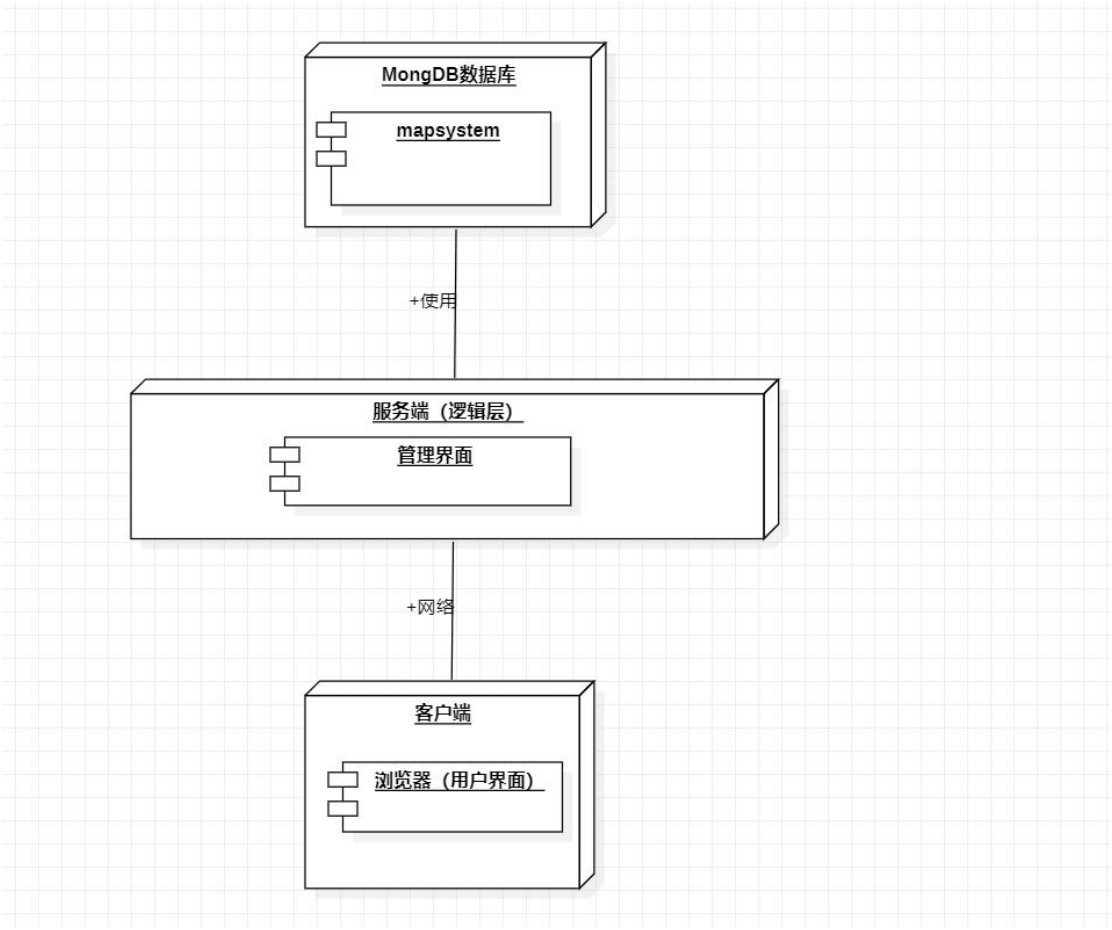
(2)分析:

优点：
①多个视图与一个模型相对应，变化－传播机制确保了所有相关视图都能够及时地获取模型变化信息，从而使得所有视图和控制器同步，便于维护。
②既有良好的可移植性
③系统被分割为三个独立的部分，当功能要求发生变化时，改变其中一个部分就能满足要求。
缺点：
①增加了系统设计和运行的复杂性
②视图与控制器连接过于紧密，妨碍了二者的独立重用
③视图访问模型的效率比较低
④频繁访问未变化的数据，降低了系统的性能

3. 构件图

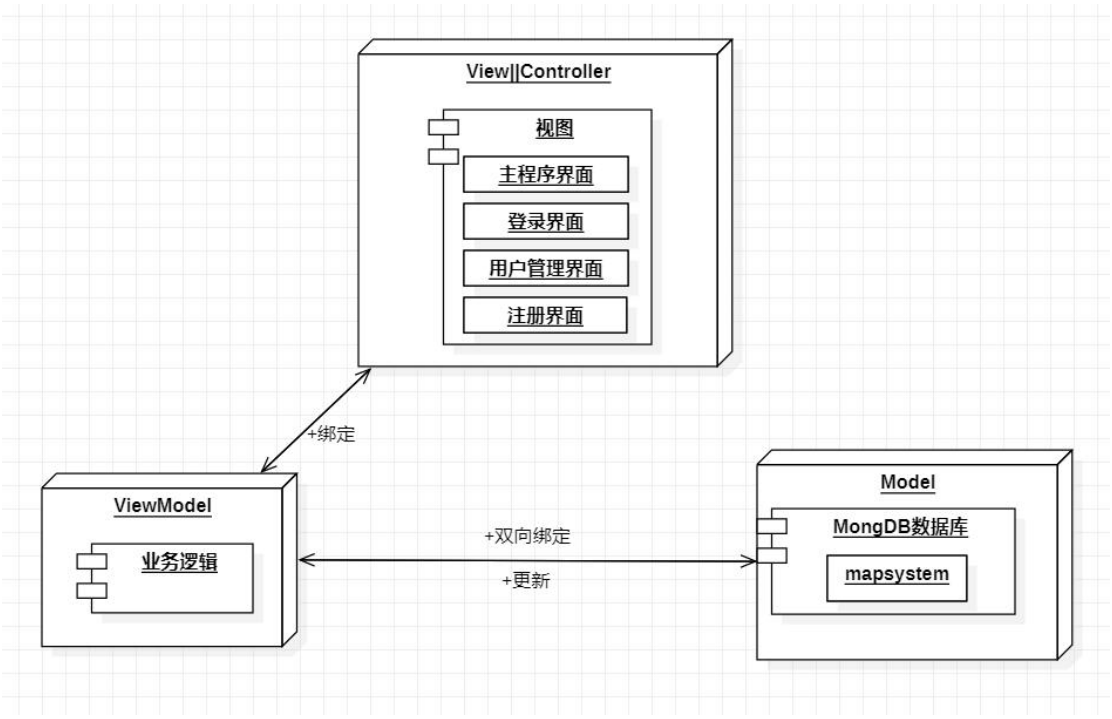


4. 部署图



5. 设计模式

（1）MVVM 模式



概述：vue 中使用了 MVVM 模式，最大的特点为数据双向绑定，大大简化了 js 的开发流程，给开发者提供了方便。MVVM 模式实际上是在 MVC 的基础上实现的，其与 MVC 有很大程度的相似之处，该模式与 MVC 模式最大的区别在于多了一个 ViewModel。ViewModel 把原来 Controller 的数据和逻辑处理部分抽离出来，单独管理，形成 View 和 Controller 之间的一座桥梁，这样就使 Controller 变得易于维护和测试了。

分析：

优点：

①低耦合：由于 View 和 ViewModel 之间松散的耦合关系，原型开发和逻辑开发可分离（即前后端可以分离开发），模块间的低耦合有利程序的扩展。

②可重用性：ViewModel 中包含一些逻辑视图，而一些 View 可以重用这些逻辑。

③可测试性：由于 ViewModel 的存在，所以测试程序可以针对 ViewModel 来写。

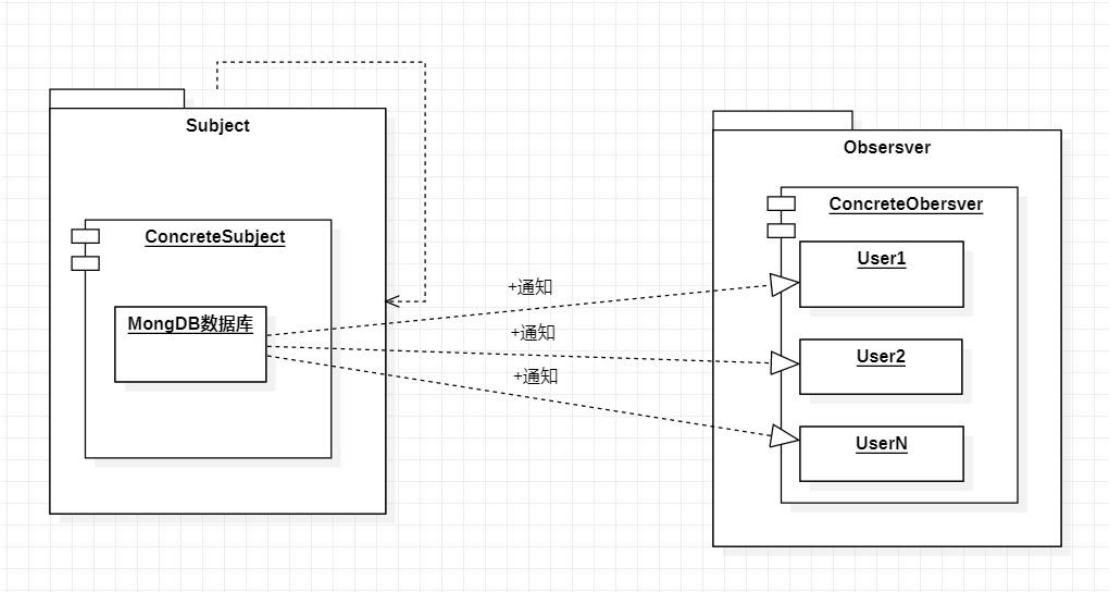
缺点：

①数据绑定使得 Bug 不易调试，也会使得一个位置的 Bug 快速传递到其他位置。

②虽然使用 Model 方便保证数据的一致性，但是大的模块中长期不释放内存会造成内存的浪费。

③数据双向绑定不利于 View 部分代码的重用。

（2）观察者模式



概述：通过本项目的功能需求，不难发现本项目为一个典型的观察者模式。其中观察目标：数据库中的相关内容（用户信息、数据文件等）；观察者：各个用户。他们之间形成了这种一对多的依赖关系，数据库中的内容发生变化就会导致用户所观察到的信息发生变化，从而影响到用户对象，这是一个典型的观察者模式。

分析：

优点：

①可以实现数据逻辑层和表现层的分离

②在观察者和观察目标之间建立一个抽象的耦合

③支持广播通信，大大简化了一对对系统

④增加新的具体观察者无需修改其他代码

缺点：

①通知所有的观察者需要花费很多时间和资源
②没有相应的机制让观察者知道被观察者发生了什么样的改变