软件工程课程设计

——文档汇总

# 系统名称

网上法律咨询系统

# 开发及运行环境

网上法律咨询系统的开发及运行环境如表1所示。

**表1 环境配置**

|  |  |
| --- | --- |
| 开发语言 | 后端 - Java(jdk1.8)，Python(3.7)，前端 - Html |
| 开发框架 | Spring Boot(2.1.3)，MyBatis，Junit，Pytorch(1.4) |
| 数据库服务器 | Mysql(8.0) |
| 项目管理 | Maven |
| 开发平台 | JetBrains Intelij IDEA 2020 |
| 开发系统 | Windows10 |
| 网络服务器 | Tomcat(9.0) |

# 系统简介

网上法律咨询系统是一款面向法律领域工作者以及法律问题咨询者的在线交流平台。可实现协助裁决法律纠纷，法律条文检索，热点法律条文以及案件的推送，分级管理用户信息以及匹配用户律师等功能。

* **功能性需求**

1. **用户管理：**未在本平台注册过的用户，只能进行法律条文检索，查看热点案件等基本操作。若想更加深入使用本平台，需要在本平台注册登录。在用户注册过程中，需要提供联系方式等基本信息，律师用户需要额外提供律师证明，登记律师事务所信息。登陆后的普通用户可以享受案件咨询服务，律师可以选择性地为用户咨询进行人工解答，所有用户都可以查询以及管理自己过往地咨询或者解答记录。管理员可以查看系统日志，查看法条库信息以及查看用户信息库。
2. **案件咨询：**用户在浏览器中输入想咨询的问题，服务器会利用文本挖掘以及信息检索的算法，给出用户咨询可能会涉及到的法律条文，以及刑期预测的结果。
3. **按标签检索法律条文：**用户可根据《中华人民共和国刑法》中章条标签进行检索，服务器会向用户提供所涉及法条的具体内容，并且提供《中华人民共和国刑法》文本下载功能。
4. **热点法律条文、案件推送：**系统将综合法条点击率，以及“协助裁决法律纠纷”功能中相关法条被检索次数，统计每个法条的热度，在平台首页向用户推送热点法条。此外，平台首页还会推送近期被咨询的问题，并根据点击率实时调整排序。
5. **讨论区：**用户可以在讨论区发布咨询信息，请求律师进行人工答复，律师可以在讨论区针对咨询信息与咨询者建立会话，进行一对一交流。建立会话后的咨询信息将不会在讨论区出现。双方都可以主动取消一对一交流。

* **非功能性需求**

1. 系统运行正常且稳定。
2. 为了保证数据安全、可靠，实行加密，数据备份与保护。
3. 基于一定准确率的查询（检索）结果可靠。
4. 对用户的操作进行实时的响应。
5. 界面友好，清晰简洁，符合用户操作习惯。

# 软件体系结构

采用软件体系三层架构，数据访问层（DAO），业务逻辑层（Server），业务表示层（Controller）进行开发，使得软件系统可以分散关注、松散耦合、逻辑复用、达成标准定义。软件体系结构图如图1所示。



**图1 软件体系结构图**

# 系统完成情况

网上法律咨询系统的开发历程如表2所示。

表2 开发历程

|  |  |
| --- | --- |
| 2020.05.03-  2020.05.07 | 初步完成需求分析 |
| 2020.05.15-  2020.05.17 | 复习计算机网络相关知识，了解Tomcat网络服务器，初步确定开发环境 |
| 2020.05.18-  2020.05.20 | 学习Java |
| 2020.05.21-2020.05.26 | 学习Maven项目管理，SpringBoot框架，IDEA搭建项目 |
| 2020.05.27 | 修改需求分析 |
| 2020.05.27-  2020.06.11 | 完成系统开发 |
| 2020.06.11-  2020.06.14 | 完成系统测试，修复BUG，整理文档 |

该项目的所有功能性需求都可以正常运行；在非功能性需求方面，系统能够正常稳定运行；用户的隐私保护手段一般，平台更加注重信息共享，因此用户需要自己注意不泄露隐私；案件咨询、智能预测采用的算法较为基础，效果不佳；对用户的操作可以及时响应；用户界面较为简洁，视觉效果不佳。

# 分层UseCase模型

做需求分析的时候，我们采用分层的UseCase用例建模方法。该系统中一共定义了5位角色，分别是普通用户，律师，用户信息库，法条库，日志库，系统管理员。

普通用户的角色描述如表3所示。

表3普通用户的角色描述

|  |
| --- |
| **角色：**普通用户  **角色职责：**提供咨询信息，维护个人信息，参与社区提问  **角色职责识别：**负责生成咨询信息，使用系统的主要功能，对系统运行结果感兴趣 |

律师的角色描述如表4所示。

表4 律师的角色描述

|  |
| --- |
| **角色：**律师  **角色职责：**提供咨询信息，维护个人信息，参与会话区解答  **角色职责识别：**负责生成咨询信息，参与维护社区日常运转，使用系统的主要功能，对系统运行结果感兴趣 |

用户信息库的角色描述如表5所示。

表5 用户信息库的角色描述

|  |
| --- |
| **角色：**用户信息库  **角色职责：**统计注册用户的信息，并且能分别提供律师和普通用户的信息展示功能  **角色职责识别：**负责收集用户的用户名，真实姓名，密码，身份证号，律师证明，律所所在地，律所名称信息，并且提供用户信息展示功能，对新注册用户输入信息的正确性具有一定鉴别功能 |

法条库的角色描述如表6所示。

表6 法条库的角色描述

|  |
| --- |
| **角色：**法条库  **角色职责：**维护《中华人民共和国刑法》的法条信息，实时提供推荐的法条，针对用户的咨询、检索请求，给出对应的法条或刑期预测结果  **角色职责识别：**负责收集用户的咨询、检索信息，实时更新法条数据库信息 |

日志库的角色描述如表7所示。

表7 日志库的角色描述

|  |
| --- |
| **角色：**日志库  **角色职责：**统计系统运行的状态，以及案件咨询的时间，实时提供推荐的案件，供管理员查看日志  **角色职责识别：**负责收集用户的登录、咨询、检索、会话状态，实时更新日志数据库 |

系统管理员的角色描述如表8所示。

表8系统管理员的角色描述

|  |
| --- |
| **角色：**系统管理员  **角色职责：**检查系统运行日志，检查用户信息和法条信息  **角色职责识别：**负责检查日志和数据库状态，发现异常要及时处理 |

顶层UseCase模型是对每个角色的用例进行概括，我们一共定义了5个上层用例，分别是用户登录，用户管理，咨询检索，统计可视化和后台管理。

用户登录的用例描述如表9所示。顶层用例建模如图2所示。

表9 用户登陆的用例描述

|  |  |
| --- | --- |
| **用例名：**用户登录 | **执行者：**普通用户，律师，系统管理员，用户信息库，日志库 |
| **目标：**系统对用户身份进行评定，不同身份用户享有不同系统功能 | |
| **功能描述：**   1. 用户输入基本信息，完成账号申请或密码修改 2. 用户输入账号密码以及验证码，若账号密码匹配且验证码无误，系统自动认证用户身份并转入登陆完成的界面 3. 日志库记录登录异常情况 | |
| **其他非功能需求：**(针对用户登陆异常进行处理）   1. 网络波动：请求用户重新登陆 2. 账号密码不匹配：报错“账号密码不匹配”，若错误次数过多，冻结账号 | |
| **主要步骤：**   1. 用户输入账号，密码，验证码，若全部正确转3，若无账号转2 2. 用户申请账号 3. 系统认证用户身份 4. 记录登录日志 | |
| **相关用例：**用户管理，咨询检索，只有完成了用户登录以后，才能进一步完成上述两个用例 | |
| **相关信息：**（错误次数，账号冻结时间，系统管理员）   1. 错误次数：一般连续错误次数不超过5次，超过就需要当作异常处理 2. 账号冻结时间：基本的冻结时间可设为10min，若仍存在异常，可延长冻结时间 3. 系统管理员：系统管理员具有固定的账号密码 | |

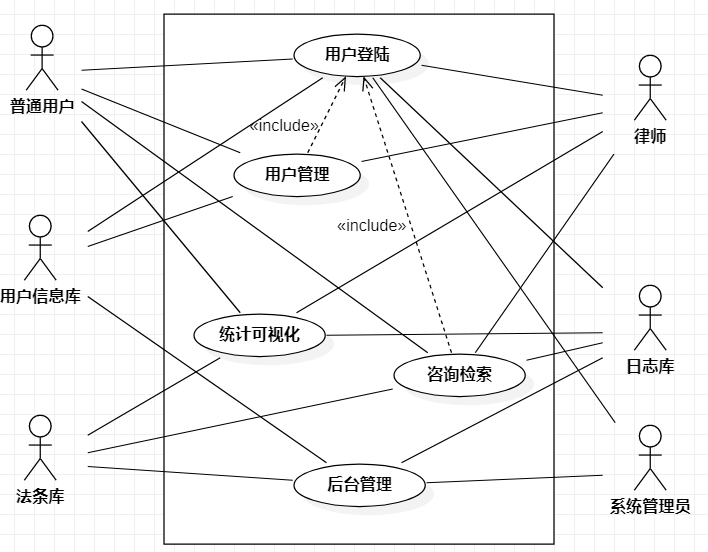


图2 顶层用例建模

用户管理的用例描述如表10所示。底层用户管理的用例建模如图3所示。

表10 用户管理的用例描述

|  |  |
| --- | --- |
| **用例名：**用户管理 | **执行者：**普通用户，律师，用户信息库 |
| **目标：**用户对隐私，个人信息进行维护，实时更新数据库信息 | |
| **功能描述：**   1. 用户输入原密码以及新密码，原密码正确则更新数据库密码 2. 用户查看自己的案件咨询记录（问题以及结果），且可以自行删除记录 3. 用户查看自己已经简历的会话（普通用户-律师，法官），且可以向会话另一方发送消息或者终止会话 4. 普通用户可以申请律师认证，律师可以修改自己的认证信息 | |
| **其他非功能需求：**（安全性，会话的及时性）   1. 安全性：用户只能对自己的信息进行查看管理 2. 会话的及时性：会话的延迟不能太低，若网络波动，需要对用户提示 | |
| **主要步骤：**  申请/修改律师认证：   1. 提供律师证 2. 提供律所所在地 3. 提供律所全称 | |
| **相关用例：**用户登陆，只有完成了用户登录以后，才能用户管理 | |
| **相关信息：**（会话延迟）   1. 会话延迟：会话延迟至多5s | |

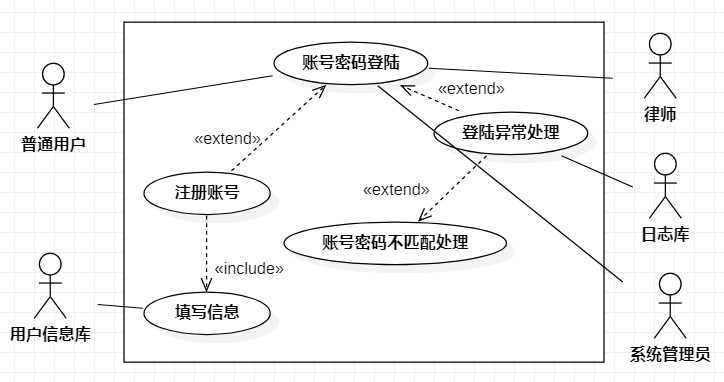


图3 底层用户管理的用例建模

咨询检索的用例描述如表11所示。底层咨询检索的用例建模如图4所示。

表11 咨询检索的用例描述

|  |  |
| --- | --- |
| **用例名：**咨询检索 | **执行者：**普通用户，律师，法条库，日志库 |
| **目标：**用户输入案件描述，系统返回相关法条或刑期预测；用户点击法律条文标签，系统展示法律条文；普通用户在讨论区发布咨询，律师法官在讨论区解答咨询；日志库记录用户资讯检索情况 | |
| **功能描述：**   1. 用户输入案件描述，服务器利用文本挖掘和信息检索算法处理用户案件，返回相关法条或者刑期预测结果 2. 用户点击法条标签，系统返回对应法条 3. 普通用户可以在讨论区新建咨询，律师法官可以选择咨询问题进行人工答复 | |
| **其他非功能需求：**（算法效率，服务器负载）   1. 算法效率：文本处理以及检索难度较高，处理速度较慢，需要选择较为合适的算法，在运行速度和准确率之间取得折中的效果 2. 服务器负载：由于存在多人同时检索的可能性，服务器需要保证用户的平均等待时间较低，使得服务器具有实用性 | |
| **主要步骤：**   1. 用户输入案件描述或者点击法条标签或者在讨论区新建咨询会话 2. 法条库接收用户输入信息并处理 3. 用户接收预测结果 4. 记录咨询日志 | |
| **相关用例：** | |
| **相关信息：**（检索效率）   1. 检索效率：用户进行案件咨询的等待时间不应该超过5秒，按标签检索法条的等待时间不应该超过1秒 | |

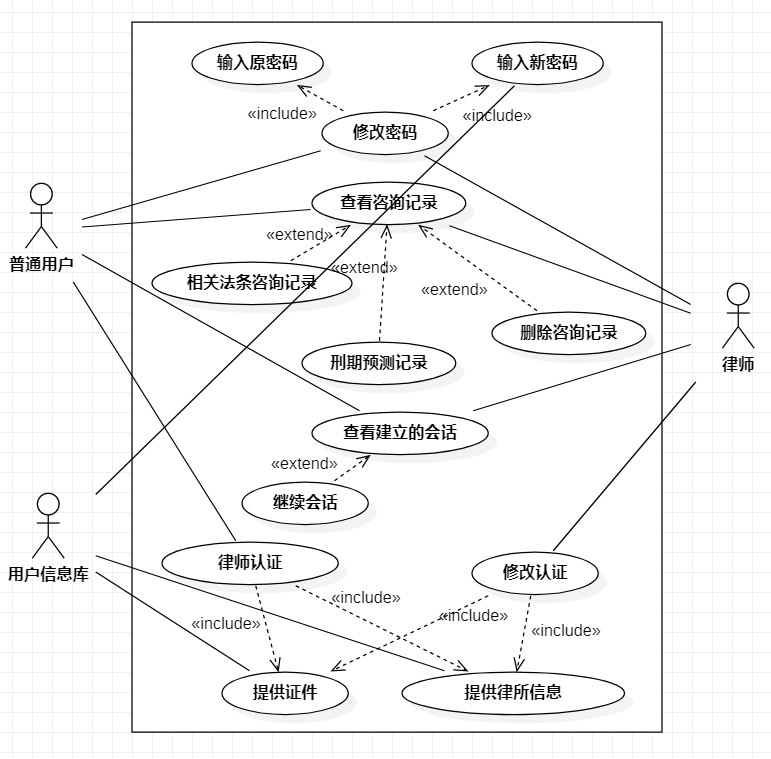


图4 底层咨询检索的用例建模

统计可视化的用例描述如表12所示。底层统计可视化的用例建模如图5所示。

表12 统计可视化的用例描述

|  |  |
| --- | --- |
| **用例名：**统计可视化 | **执行者：**普通用户，律师，法条库，日志库 |
| **目标：**服务器统计用户案件咨询，法条检索数据，在主页上推送热点案件，法条，显示在线人数 | |
| **功能描述：**   1. 日志库记录用户的案件咨询情况，推送近期热门的案件，并且后续根据点击率进行排名 2. 法条库综合案件咨询以及法条检索情况，推送热门法条 3. 显示在线人数 | |
| **其他非功能需求：**（主页推送内容的更新速度）  主页推送内容的更新速度：推送内容更新速度需根据访问人数进行修改。 | |
| **主要步骤**：   1. 用户输入案件咨询或者点击法律标签 2. 服务器统计数据 3. 服务器计算热点案件，法条并推送 | |
| **相关用例：** | |
| **相关信息：**（更新速度）  更新速度：若平台访问人数很少，可考虑每有一位顾客访问后就刷新一次推送内容；若平台访问人数多可考虑一天刷新一次推送内容 | |

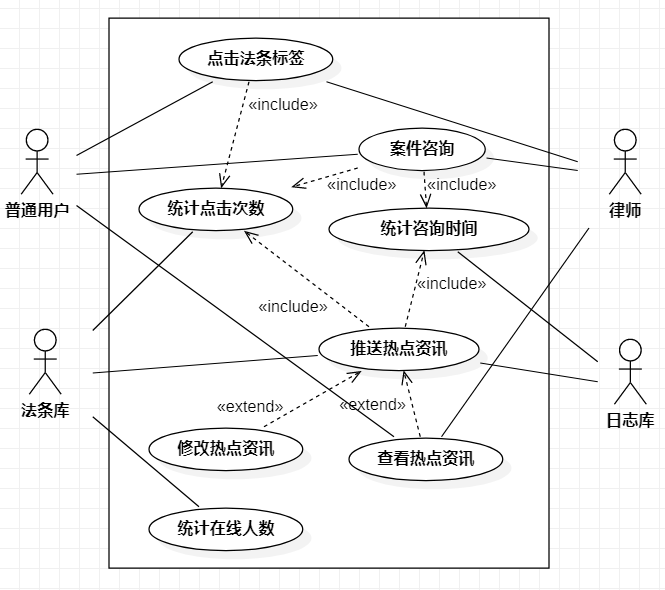


图5 统计可视化的用例建模

后台管理的用例描述如表13所示。后台管理的用例描述如图6所示。

表13 后台管理的用例描述

|  |  |
| --- | --- |
| **用例名：**后台管理 | **执行者：**用户信息库，法条库，日志库系统管理员 |
| **目标：**系统管理员查看系统日志文件，发现异常后及时进行调整 | |
| **功能描述：**   1. 查询日志文件 2. 查看用户信息 3. 查看法条信息 | |
| **其他非功能需求：** | |
| **主要步骤：**   1. 管理员登录 2. 查看日志文件，用户信息，法条信息 | |
| **相关用例：** | |
| **相关信息：** | |

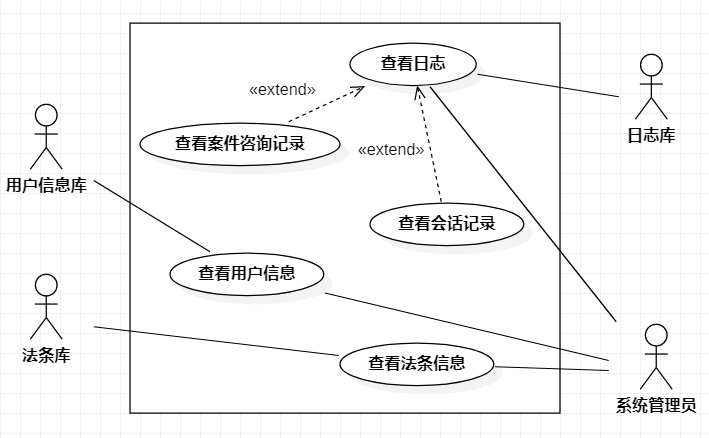


图6 后台管理的用例建模

# 类的描述

在分析确定类的过程中，我们遵守以下6点准则：

1. **去掉冗余类：**保留最具有描述能力的类。
2. **去掉不相干的类**：删除与问题无关或关系不大的类。
3. **凝聚语义：**凝聚相关类的语义。
4. **删除模糊的类或性质独立性不强的类：**有些初始类边界定义不确切，或范围太广，应该删除。
5. **删除不适宜项：**所描述的是属性或操作，不适宜作为对象类，所描述的只是实现过程中的暂时的对象，应删去。
6. **遵循软件体系结构：**在分析确定类时，要遵循三层软件习题结构，将数据访问类，业务逻辑类，业务表示类分开。

最终我们确定了如下几个类：

1. **实体类（Entity）：**用户，法条，登录记录，智能咨询记录，讨论区咨询记录，会话记录。
2. **数据访问类（Dao）：**智能咨询，会话，登录，讨论区，法条，用户。
3. **业务逻辑类（Service）：**用户登录，用户注册，文件上传，在线人数监听，统计可视化，维护刑法，讨论区，智能咨询，用户管理。
4. **业务表示类（Controller）：**登录，数据可视化，智能咨询，讨论区，用户管理。

其中业务表示层是提供给用户操作的接口，前端界面的设计也是在业务表示类的基础上进行的。我们在功能需求中提出，未登录用户智能进行登陆操作以及注册操作，将业务表示层分成上述5个类有助于权限管理。在制作分析类图的时候，我们将按照对用户暴露的5个业务表示类进行说明。

# 分析类图

根据定义好的类别，我们可以分析出类与类之间的层级关系，做出响应的分析类图。登录功能的分析类图如图7所示。

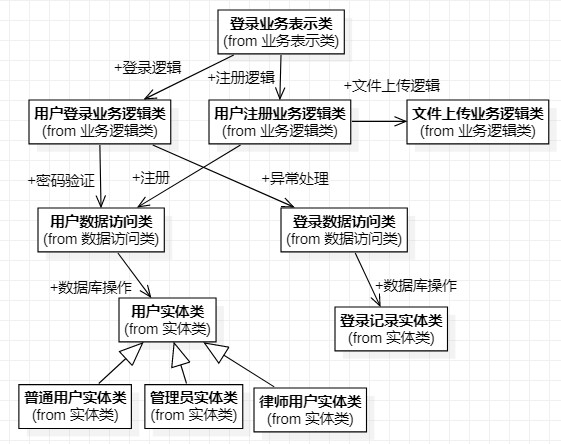


图7登录功能的分析类图

统计可视化功能的分析类图如图8所示。

![C:\Users\83470\Desktop\MobileFile\Image\O(SH10WLKY0{HZ](3HAH5`M.png](data:image/png;base64,)

图8统计可视化功能的分析类图

智能咨询功能的分析类图如图9所示。

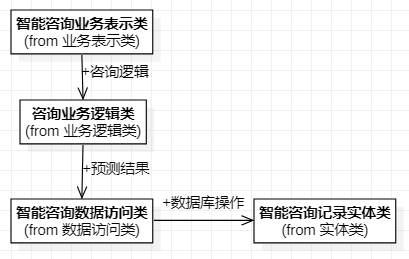


图9智能咨询的分析类图

讨论区功能的分析类图如图10所示。

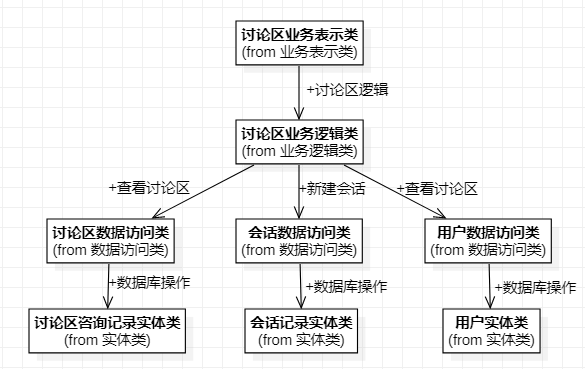


图10讨论区功能的分析类图

用户管理功能的分析类图如图11所示。

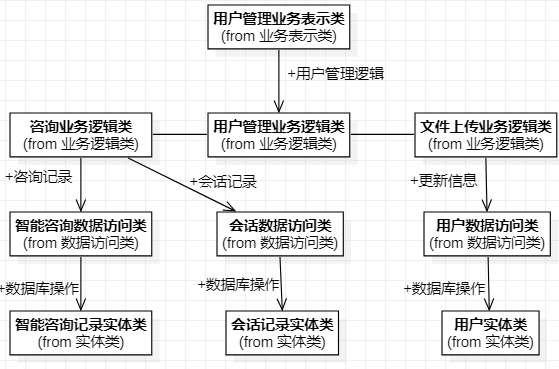


图11用户管理功能的分析类

刑法功能的分析类图如图12所示。

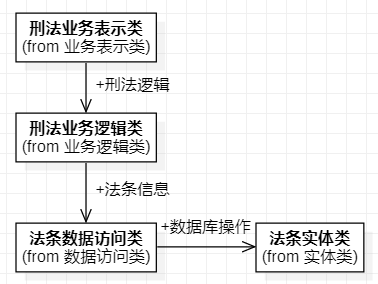


图12刑法功能的分析类图

# 静态模型

* 设计类图

在分析类图中，我们着重分析类与类之间的关系；在设计类图中，我们需要将类的属性和接口定义好，为了方便起见，在此只给出类的属性和接口的定义，类与类之间的关系如分析类图中所示。实体类的类描述图如图13所示。

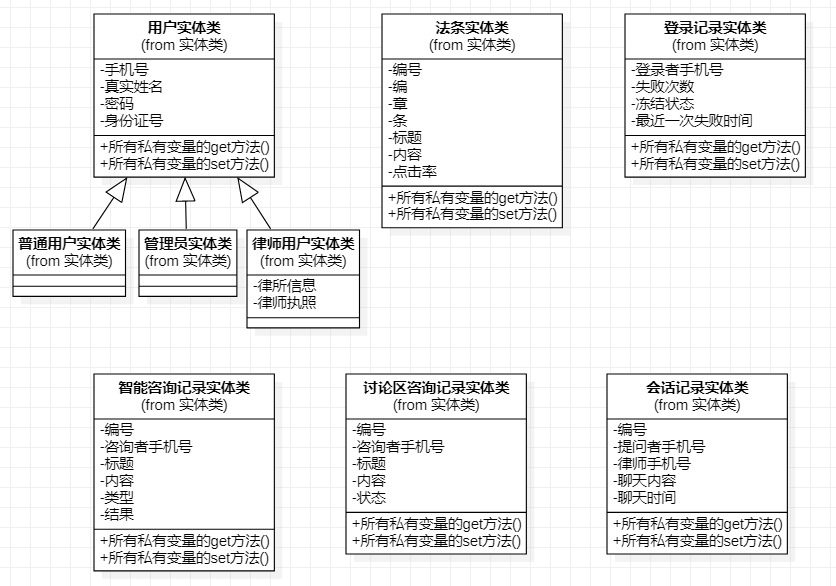


图13实体类的类描述图

数据访问类的类描述图如图14所示。

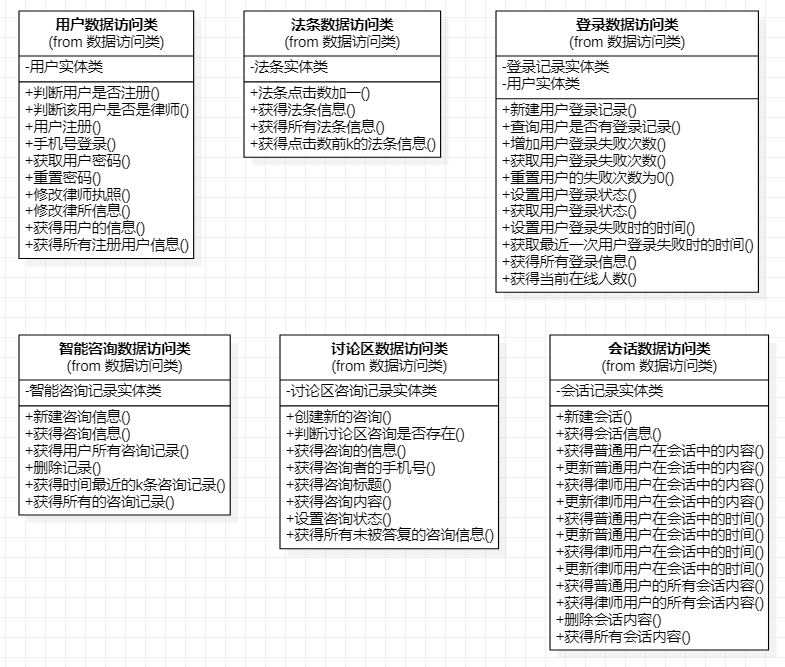


图14数据访问类的类描述图

业务逻辑层的类描述图如图15所示。



图15业务逻辑层的类描述图

业务表示层的类描述图如图16所示。

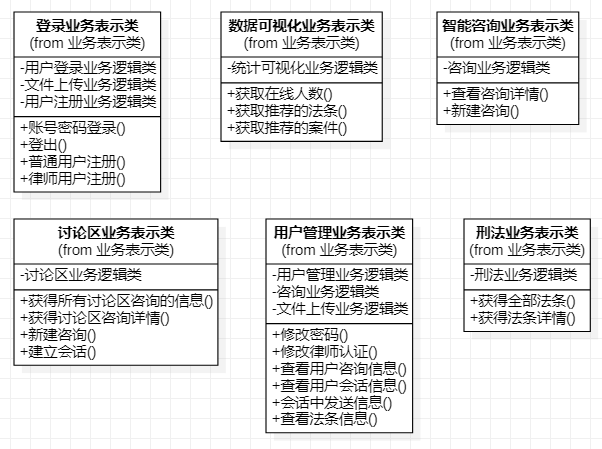


图16业务逻辑类的类描述图

# 动态模型

动态模型主要描述系统的动态行为和控制结构，其中包括4类图：状态图，描述某个对象，子系统，系统生命周期；活动图，描述操作实现中完成的工作以及用例或对象中的活动，是状态图的变种；顺序图是一种交互图，描述对象之间的动态合作关系以及合作过程中的行为次序，常用来描述一个用例的行为；合作图，用来描述相互合作的对象间的交互关系。

网上法律咨询系统服务的对象是用户和管理员，因此我们首先利用活动图描述一下客户建立会话的生命周期中，从登录到登出的过程中，系统中可能进行的活动。做出系统的活动图如图17所示。

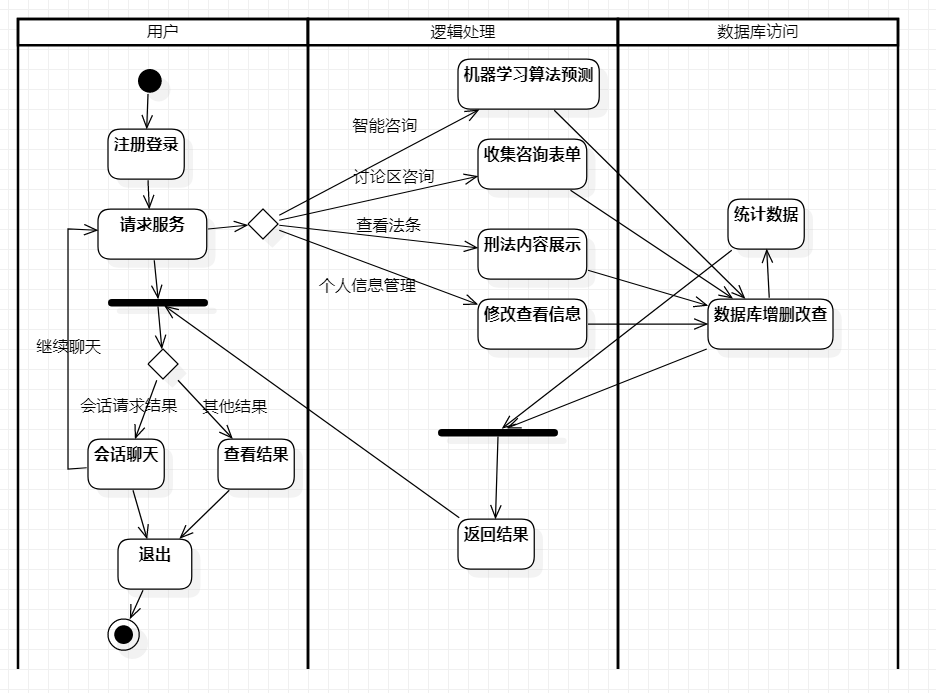


图17 系统活动图

我们的系统在用户登录模块做了较多的逻辑判断工作，这应该算是该系统逻辑上最复杂的部分，因此我们针对用户登录模块做出状态图如图18所示。

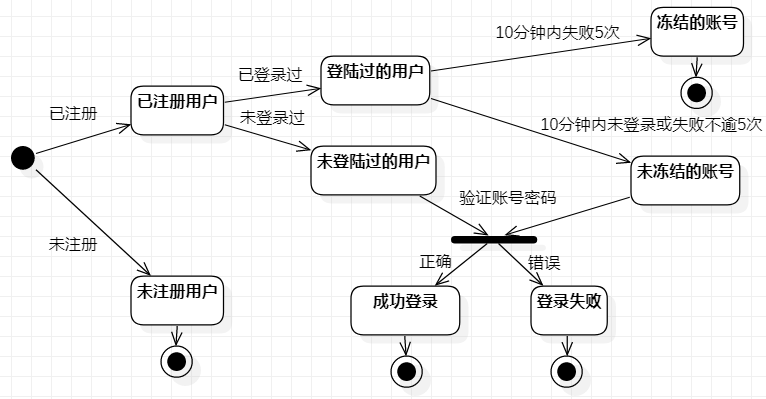


图18 用户登录状态图

由于系统大多数功能的实现只涉及到两个类之间的通信，并不存在多对象之间的动态协作，比如说用户登录用例中，表单提交的信息调用登录类的信息，通过用户类的验证，再将信息返回给登录类。因此，我们仅选择了较为复杂的两个用例—智能咨询与讨论区功能，做相应的时序图分析对象之间的交互顺序。

智能咨询的时序图如图19所示。

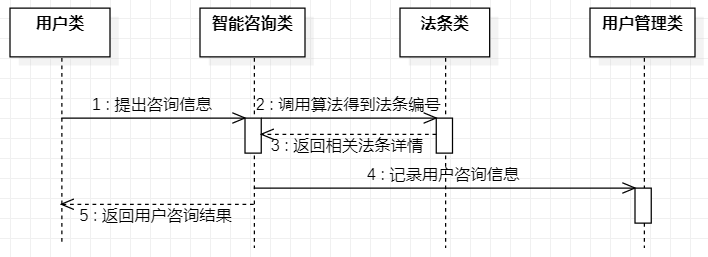


图19 智能咨询的时序图

讨论区会话的时序图如图20所示。

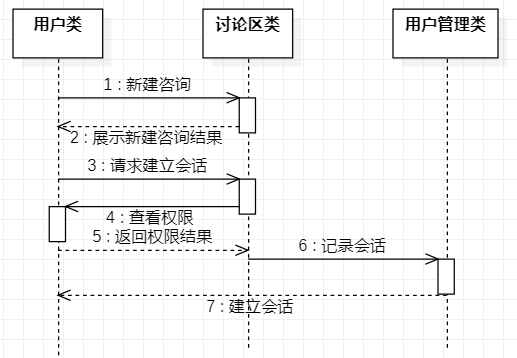


图20 讨论区会话的时序图

# 主要源代码以及可执行文件

所有源代码在附件1中可见，接下来将做出代码的目录结构使得代码逻辑能够清晰展现。

1. legalsys
2. ├─src
3. │   ├─main
4. │       ├─java
5. │           ├─com.huidong.legalsys
6. │               ├─aspect（权限管理）
7. │                   ├─LoginAspect.java
8. │               ├─configuration
9. │                   ├─DataSourceConfiguration.java
10. │                   ├─UploadFileConfiguration.java
11. │               ├─controller（业务表现层）
12. │                   ├─ConsultController.java
13. │                   ├─LoginController.java
14. │                   ├─ManageController.java
15. │                   ├─PenallawController.java
16. │                   ├─SessionController.java
17. │                   ├─StatisticsController.java
18. │               ├─dao（数据访问层）
19. │                   ├─ConsultDao.java
20. │                   ├─ConvrDao.java
21. │                   ├─LoginDao.java
22. │                   ├─SessionDao.java
23. │                   ├─StatureDao.java
24. │                   ├─UserDao.java
25. │               ├─domain（实体层）
26. │                   ├─Consult.java
27. │                   ├─Convr.java
28. │                   ├─ConvrContent.java
29. │                   ├─Error.java
30. │                   ├─Login.java
31. │                   ├─Session.java
32. │                   ├─Stature.java
33. │                   ├─User.java
34. │               ├─enumeration
35. │                   ├─ConsultTypeEnum.java
36. │                   ├─ErrorEnum.java
37. │                   ├─LoginStatusEnum.java
38. │                   ├─RegisterTypeEnum.java
39. │                   ├─SessionStatusEnum.java
40. │               ├─exception
41. │                   ├─LegalsysException.java
42. │               ├─handle
43. │                   ├─ExceptionHandle.java
44. │               ├─listener（在线人数监听）
45. │                   ├─HttpSessionListener.java
46. │                   ├─RequestContextListener.java
47. │                   ├─ServletContextListener.java
48. │               ├─service（业务逻辑层）
49. │                   ├─ConsultService.java
50. │                   ├─LoginService.java
51. │                   ├─ManageService.java
52. │                   ├─PenallawService.java
53. │                   ├─SessionService.java
54. │                   ├─StatisticsService.java
55. │                   ├─UploadService.java
56. │               ├─util
57. │                   ├─ErrorUtil.java
58. │               ├─LegalsysApplication.java（项目启动类）
59. │       ├─resources
60. │           ├─mapper（Mybatis映射）
61. │               ├─ConsultMapper.xml
62. │               ├─ConvrMapper.xml
63. │               ├─LoginMapper.xml
64. │               ├─SessionMapper.xml
65. │               ├─StatureMapper.xml
66. │               ├─UserMapper.xml
67. │           ├─python
68. │               ├─preprocess.py
69. │           ├─static
70. │               ├─css
71. │                   ├─fonts
72. │                   ├─style.css（html配置文件）
73. │               ├─images（存放图片文件）
74. │               ├─stature（存放刑法数据）
75. │           ├─templates（html文件）
76. │           ├─application.yml（配置文件）
77. │           ├─application.properties（配置文件）
78. │       ├─resourcesupload（律师执照上传路径）
79. │   ├─test（单元测试文件）
80. ├─init.sql（mysql新建数据库及表单）
81. ├─pom.xml（Maven项目配置文件）

项目完成之后，使用maven clean install，对项目进行编译、单元测试、打包，并将打包好的可执行jar包部署到本地maven仓库。

# 实现模型

构件图显示代码本身的逻辑结构，描述系统中存在的软结构以及他们之间的依赖关系。

做出系统的构件图如图21所示。

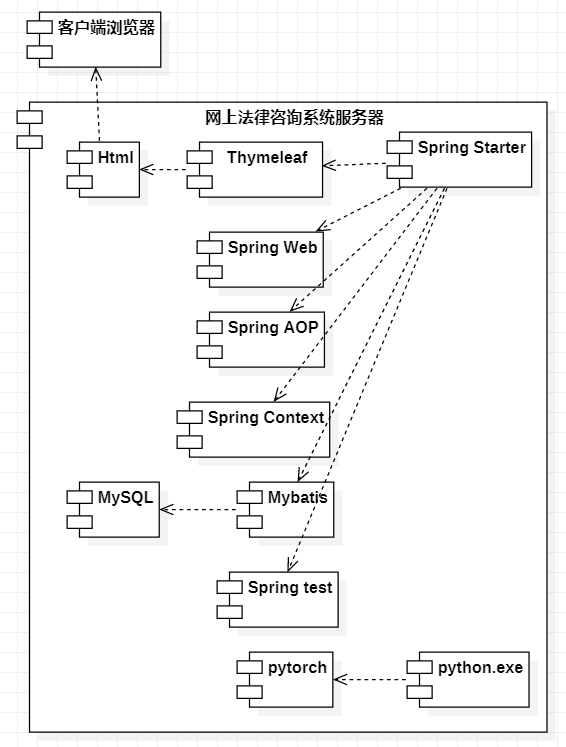


图21 系统构件图

部署图描述了系统中硬件和软件的物理配置情况和系统体系结构。做出系统的部署图如图22所示。

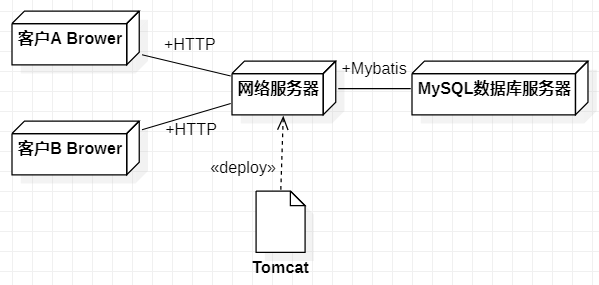


图22系统部署图

# 数据库表单

该项目所建立的数据库名称为legalsys，legalsys共设置有5个表单，分别为userTable用来记录用户个人信息。statureTable用来记录法条信息，loginTable用来记录用户登录，consultTable用来记录用户咨询，convrTable用来记录用户会话。

userTable的表单字段内容如表14所示。

表14 userTable表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| phone | varchar(255) | PRIMARY KEY |
| name | varchar(255) | NOT NULL |
| password | varchar(255) | NOT NULL |
| idno | varchar(255) | NOT NULL  UNIQUE |
| licenseurl | varchar(255) | NULL DEFAULT NULL UNIQUE |
| firmname | varchar(255) | NULL DEFAULT NULL |

statureTable的表单字段内容如表15所示。

表15 statureTable的表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| lawid | int(11) | PRIMARY KEY |
| part | int(11) | NOT NULL |
| chapter | int(11) | NOT NULL |
| article | int(11) | NOT NULL |
| title | varchar(255) | NOT NULL |
| content | varchar(255) | NOT NULL |
| clickrate | int(11) | NOT NULL |

loginTable的表单字段内容如表16所示。

表16 loginTable的表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| phone | varchar(255) | PRIMARY KEY  FOREIGN KEY (phone) REFERENCES userTable (phone) |
| attempt | int(11) | NOT NULL |
| status | int(11) | NOT NULL |
| freezetime | varchar(255) | NULL DEFAULT NULL |

consultTable的表单字段内容如表17所示。

表17 consultTable的表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| id | int(11) | PRIMARY KEY |
| phone | varchar(255) | NOT NULL  FOREIGN KEY (phone) REFERENCES userTable (phone) |
| title | varchar(255) | NOT NULL |
| query | varchar(255) | NOT NULL |
| type | int(11) | NOT NULL |
| result | varchar(255) | NOT NULL |
| time | varchar(255) | NOT NULL |

convrTable的表单字段内容如表18所示。

表18 convrTable的表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| id | int(11) | PRIMARY KEY |
| phone | varchar(255) | NOT NULL  FOREIGN KEY (phone) REFERENCES userTable (phone) |
| lawyerphone | varchar(255) | NOT NULL  FOREIGN KEY (lawyerphone) REFERENCES userTable (phone) |
| convr | text | NOT NULL  UNIQUE |
| time | text | NULL DEFAULT NULL UNIQUE |

sessionTable的表单字段内容如表19所示。

表19 sessionTable的表单字段

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 约束 |
| id | int(11) | PRIMARY KEY |
| phone | varchar(255) | NOT NULL  FOREIGN KEY (phone) REFERENCES userTable (phone) |
| title | varchar(255) | NOT NULL |
| content | varchar(255) | NOT NULL |
| status | int(11) | NOT NULL |

# ER图

根据legalsys数据库的5个表单，可以相应构造出5个实体，做出实体之间的ER图如图21所示。

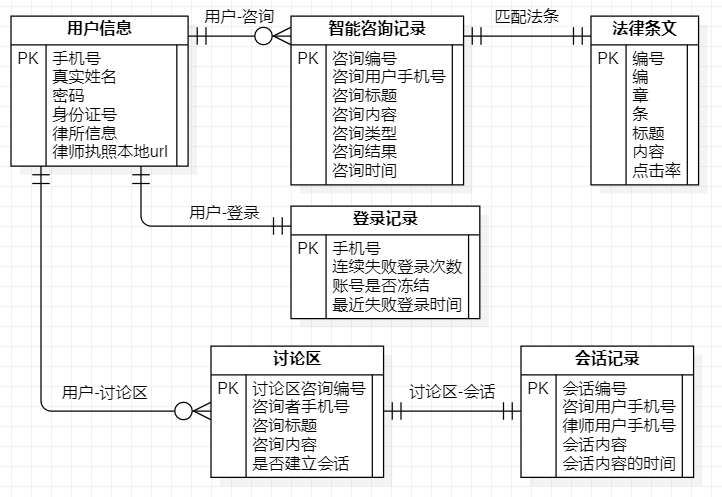


图23实体之间的ER图

# 单元测试

该项目采用Junit4进行数据访问层和业务逻辑层的单元测试，数据访问层利用assert断言来判断测试是否通过，业务逻辑层利用系统日志以及数据库结果来判断测试是否通过。由于数据访问层只涉及到数据库的增删改查，我们在测试数据访问层时选择黑盒法，只要测试用例能够通过，说明数据库的操作是没有问题的。业务逻辑层相较于数据访问层多了不少逻辑判断，因此我们采用白盒法进行测试，在设计测试用例时会根据代码语句进行调整。数据访问层的测试用例以及断言类型如表20所示，由于需要单元测试的类方法较多，在此不一一列举，仅给出用户数据访问类的测试结果。

表20 数据访问层的测试结果（部分）

|  |  |  |
| --- | --- | --- |
| 测试方法 | 测试用例 | 断言类型 |
| UserDao.isReistered | phone = "11111111111";  phone = “11111111112”; | assertEquals |
| UserDao. isRegistedLawyer | phone = "11111111111"; | assertEquals |
| UserDao. register | Phone = "15111111111";  Name = "张惠东";  Password = "222222";  Idno = "320283199903060000"; | assertTrue |
| UserDao. login | phone = "11111111111";  password = "222222";  password = "111111"; | assertNotNull |
| UserDao. getPassword | phone = "11111111111"; | assertNotNull |
| UserDao. setPassword | phone = "11111111111";  newpassword = "222222"; | assertTrue |
| UserDao. setLicenseurl | phone = "11111111111";  lincenseurl = "url1"; | assertTrue |
| UserDao. setFirmname | phone = "11111111111";  firmname = "天津市司法局"; | assertTrue |
| UserDao. getUserInfo | Na | assertNotNull |
| UserDao. getAllUsers | Na | assertNotNull |

在数据访问层进行，主要排查出一个错误：mysql在获取表单记录详情的时候，order by ××× desc可以获得表单按×××降序排列的记录，但是不能限制记录的个数。经过查阅mysql语法，发现limit可以限制获得记录的个数，该问题被解决。

业务逻辑层的测试用例以及测试结果如表21所示，由于需要单元测试的方法较多，且每个方法都需要通过白盒法测试，测试用例较多，因此在此不一一列举，仅给出用户登录业务逻辑类的测试结果。

表21 业务逻辑类的测试结果（部分）

|  |  |  |
| --- | --- | --- |
| 测试方法 | 测试用例 | 测试结果 |
| LoginService. register | phone = “11111111112”;  name = “张”;  password = “111111”;  idno = “111111111111111112”; | logger.info(“注册新的用户11111111112”) |
|  | phone = “11111111111”;  name = “惠”;  password = “111111”;  idno = “111111111111111113”; | logger.info(“注册新的用户11111111112”) |
| LoginService. registerLawyer | phone = "11111111111"; | assertEquals |
| LoginService. login | Phone = "15111111111";  Name = "张惠东";  Password = "222222";  Idno = "320283199903060000"; | assertTrue |

# 集成测试

由于业务表示层是面向用户操作的接口，是对业务逻辑层与数据访问层的概括。因此需要对该层进行集成测试。测试工具上我们采用常用的接口测试工具Postman。