2023年全国职业院校技能大赛

高职组

“区块链技术应用”

赛项赛卷（8卷）

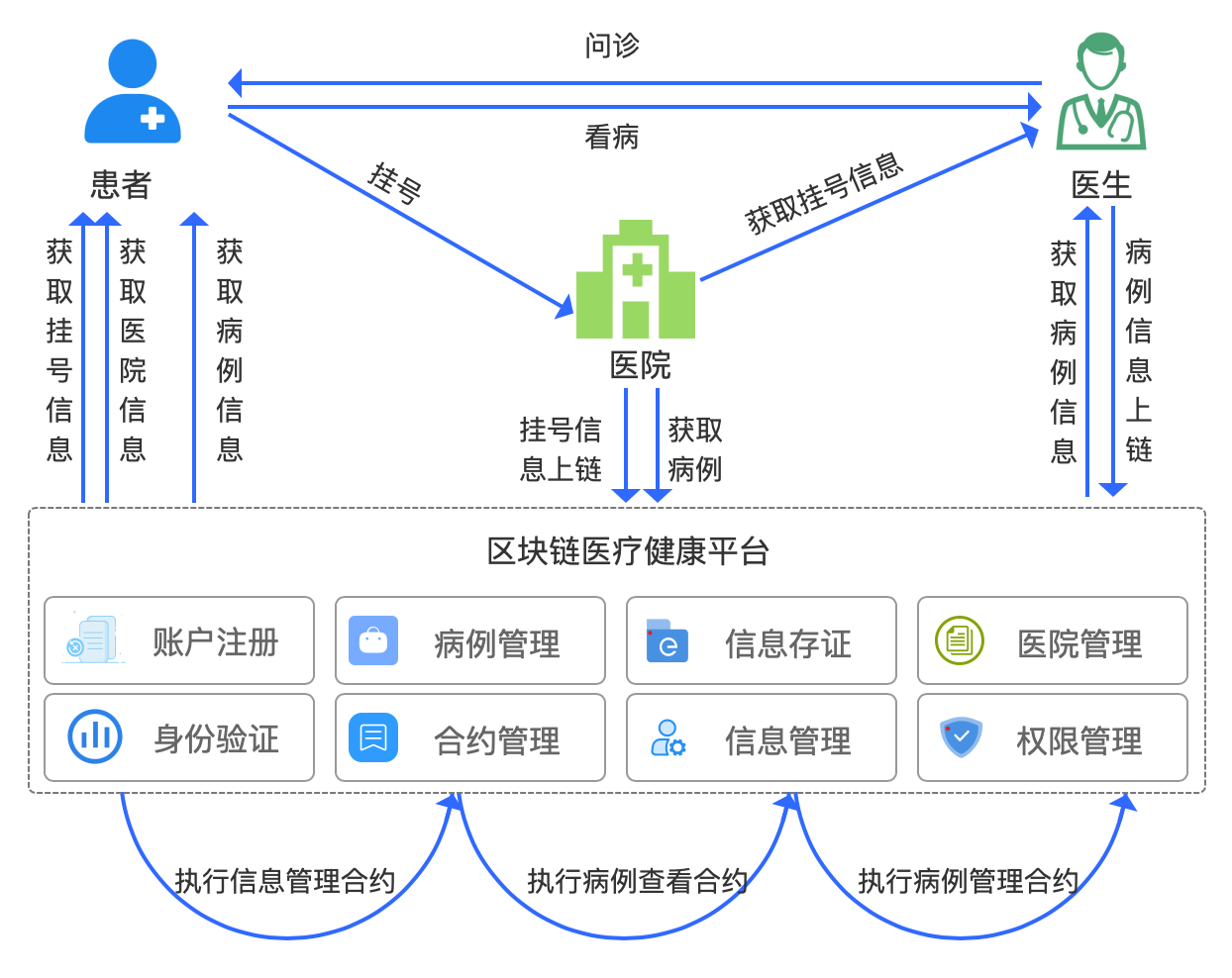
任

务

书

参赛队编号：

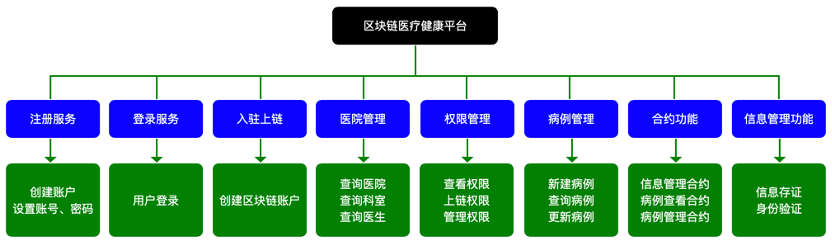
**背景描述**

现实中患者私密信息泄露情况时有发生，医疗部门的柜式存储和纸质记录已不再是最优选择。在2015-2016年间，美国的数家医院遭受黑客入侵，导致八千余万份医疗健康记录被盗。医疗数据泄漏占全球数据泄漏事件的15%，已经成为全球数据泄漏的第二大行业。并且医疗部门的柜式存储和纸质记录实现数据共享极为困难，造成病人看病难的问题。

区块链医疗健康平台业务流程图

如果能把个人的医疗检查记录上传到区块链，在这种智能化的信任机制下，不需要医患之间的相互信任，共享医疗健康数据，安全又便捷。基于区块链的医疗平台，如果患者在一家医院的检查记录，通过区块链技术的存证和追溯技术，可以受另一家医院的认可，不必重复检查，这将极大的提升患者的就医效率，减轻医院的工作负担。

现有一个基于区块链的医疗健康数据共享平台S，患者A在区块链医疗平台S中输入身份证号后进行挂号，当患者A在医院H挂科室C的号后，区块链医疗平台S将挂号信息显示给患者A。当医生D在区块链医疗平台S中点击开始就诊后，区块链医疗平台S将根据患者A的身份证号查找患者在科室C的过往所有医院的病历资料，如果有过往病历资料，调用患者A在对应科室过往的病历资料，区块链平台授权医生D查看患者病历的权限。患者就诊后医生D会在区块链医疗平台S中点击新建患者A在当前医院的病历资料。区块链医疗平台S调用患者A在医院H中的挂号信息、就诊医生的信息到区块链中进行加密上链，将相关信息添加到新建的病历资料中。医生D需要填写病历资料，如果医生D在平台不填写相关信息，系统则等待填写不能点击就诊结束动作。当医生D点击就诊结束后，医生D不可再修改患者本次的病历资料，也不可再查看患者A的病历资料。如果医生D在12小时内不点击就诊结束，区块链医疗平台S自动结束就诊。患者A可以在平台上查看自己的病历资料。



区块链医疗健康平台系统架构图

## 模块一：区块链产品方案设计及系统运维（35分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块一提交结果.docx】中对应的任务序号下。

### 任务1-1：区块链产品需求分析与方案设计

医疗健康平台中涉及到医院、医生、患者等参与方，他们需要在区块链医疗健康平台中完成账户注册、身份上链、挂号就诊、查询病例等多种业务活动。通过对业务活动的功能分析，可以更好的服务系统的开发流程。基于医疗健康平台系统架构，以区块链医疗健康平台为背景，结合账户注册、登录服务、入驻上链、查询病例、新建病例等核心功能描述，撰写流程图/功能图、用例图等概要设计。本任务需要依据项目背景完成需求分析与方案设计，具体要求如下:

1. 依据给定区块链医疗健康平台的业务流程图以及用例表，编制系统业务用例图，用例图中包含系统参与角色以及用例；

表1-1-1 用例表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **用例编号** | **用例标题** | **优先级** | **测试步骤** | **预期结果** | **测试结果** |
| HK-TEST-01 | 患者在区块链医疗健康平台上挂号 | 高 | 在挂号界面，患者可以选择科室、医生、时间完成挂号，弹出挂号凭证界面 | 挂号成功 | 挂号成功 |
| HK-TEST-02 | 医生在区块链医疗健康平台就诊 | 高 | 在挂号界面，拉取患者的挂号记录，在记录信息中点击就诊按钮，记录状态切换为开始就诊状态 | 开始就诊 | 开始就诊 |
| HK-TEST-03 | 医生在区块链医疗健康平台检索病例 | 高 | 医生在就诊界面，通过检索当前就诊人，查询该就诊人的过往病例，可以对检索的病例进行点击查看，点击查看验证医生的身份和权限，审核通过后可弹窗查看患者病历 | 拉取病例记录  审核医生身份  查看成功 | 拉取病例记录  审核医生身份  查看成功 |
| HK-TEST-04 | 医生在区块链医疗健康平台新建病例 | 高 | 在病例界面，展示当前就诊人的过往病例，可以点击新建，弹窗显示新建病例的页面，医生可在本页面输入病例信息进行创建 | 创建成功 | 创建成功 |
| HK-TEST-05 | 医生在区块链医疗健康平台结束就诊 | 高 | 在就诊界面，展示当前就诊人的就诊状态以及就诊时间，点击结束就诊进行二次确认，可结束本次就诊 | 结束成功 | 结束成功 |

2.依据给定的背景信息、区块链医疗健康业务流程图以及区块链医疗健康平台架构图以及给出的医疗健康业务的核心流程，使用思维导图编制业务系统功能图；

表1-1-2 医疗健康平台的核心流程

|  |  |
| --- | --- |
| 过往病例检索流程 | 区块链医疗健康平台对医生的身份证书进行验证，判断医生的身份、查看权限等，验证通过后以当前就诊人ID作为关键字进行检索，拉取所有的过往病例 |
| 新建病例流程 | 区块链医疗健康平台将对医生的身份证书进行验证，判断医生的身份、写入权限等，验证通过后可以新建病例，并签名加密广播到区块链 |

3. 按照基础层、合约层、接口层以及应用层来设计区块链系统的架构，画出系统架构图，其中在基础层需指明需要的节点、名称、协议、存储等信息；

4. 结合案例背景将区块链医疗健康平台核心功能进行划分，完成下方表格中各个主要模块要实现的功能；

|  |  |
| --- | --- |
| 病例管理模块 | 请输入要实现的功能 |
| 权限管理模块 | 请输入要实现的功能 |
| 合约功能模块 | 请输入要实现的功能 |

5. 根据用例设计以及下方提供好的承诺集，完成区块链应用系统业务流程图；

表1-1-3 承诺集

|  |  |
| --- | --- |
| 挂号承诺 | 判断患者是否在平台输入正确的信息完成挂号 |
| 就诊承诺 | 判断医生是否在规定时间内开始就诊 |
| 病例检索承诺 | 判断医生是否有查看权限，是则调用过往病例，否则等待授权 |
| 新建病例承诺 | 判断医生是否有新建权限，以及是否为患者新建，是则填写病例，否则就诊结束 |

6.整合所有内容，模块调用接口编写并形成对应的需求用例文档至【区块链技术应用赛\重命名为工位号】下。

### 任务1-2：区块链系统部署与运维

通过给定区块链项目需求，进行区块链系统的的部署，包括系统部署、节点部署等。通过监控工具完成对网络、节点服务的监控。最终利用业务需求规范，完成系统日志、网络参数、节点服务等系统结构的维护。

**子任务1-2-1：区块链网络环境搭建**

在本机部署区块链网络底层环境，我们需要准备编译运行的环境，并通过区块链底层源码脚本文件编译区块链网络。

（1）使用docker安装脚本，安装docker软件，安装完成之后输出docker版本号；

（2）使用docker-compose安装脚本，安装docker-compose软件，安装完成之后输出docker-compose版本号；

（3）使用docker info命令查看docker软件相关信息，获取镜像配置信息并截图保存；

**子任务1-2-2：区块链网络系统配置**

基于原有系统配置，完成以下操作：

（1）进入区块链网络执行文件夹,使用cryptogen生产创世块文件和通道文件；

（2）进入区块链网络执行文件夹，根据docker-compose.yaml文件，通过docker容器，部署节点信息；

（3）将创建的节点加入通道中。

**子任务1-2-3：区块链网络orderer集群部署**

请使用区块链底层网络，基于kafka共识完成orderer集群部署方案，任务如下：

（1）进入区块链管理服务系统主目录，编写docker-orderer.yaml文件，并找到提三个order集群ip地址，写入到该文件中；

（2）找到三个order集群对应的服务器，检查当前服务器是否有防火墙权限，如有则关闭防火墙；

（3）进入区块链管理服务系统主目录，通过命令行方式，启动Orderer集群。

**子任务1-2-4：区块链网络运维存储加密**

（1）进入区块链存储主目录文件，运行区块链脚本，生成Key Manager管理模块，并找到生成的文件；

（2）生成Key Manager管理模块，启动Key Manager，并打印日志；

（3）启动Key Manager项目，并加密节点私钥。

### 任务1-3：区块链系统测试

结合实际业务需求，设计区块链系统的测试流程，调用部署智能合约进行系统测试、性能测试等；根据业务需求，分析并且修复给定智能合约中的安全漏洞。利用模拟业务和测试工具来完成对区块链系统服务数据的测试。

（1）编写测试智能合约，并将其部署到不同的测试网络中；

（2）发起交易操作，并发保持在其能够承受的最大范围，并查看区块链底层网络运行情况；

（3）启动区块链网络系统，将智能合约部署到区块链测试环境下。

## 模块二：智能合约开发与测试（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块二提交结果.docx】中对应的任务序号下。

### 任务2-1：智能合约设计

根据医疗健康系统需求用例文档，设计合约接口，画出各需求用例的时序图。

### 任务2-2：智能合约开发

使用Solidity语言进行智能合约开发，根据需求用例文档在待补充源码中完成程序接口功能的编码，解决代码错误和警告，正确编译合约，功能调试正确，运行合约进行业务功能的验证，成功获取合约的abi。

**子任务2-2-1：信息管理合约编码**

根据需求用例文档在待补充源码中完成信息管理合约的编码，解决代码错误和警告，正确编译合约，功能调试正确，运行合约中的检索个人信息、信息管理接口功能。

1. 编写检索个人信息接口，完成患者通过身份证号检索其姓名、性别、年龄的功能；

2. 编写信息管理接口，完成允许患者与医院和科室进行信息管理，通过身份证号检索到患者的个人信息，将预约信息显示给患者，并发送到患者的账户地址中的功能；

待补充源码：

contract MedicalDataSharingPlatform {

//用于存储患者的个人信息

struct Patient {

string name; // 姓名

string gender; // 性别

uint age; // 年龄

address accountAddress; // 账户地址

}

//将患者的ID映射到其个人信息

mapping (string => Patient) patients;

// 允许患者使用其身份证号和个人信息进行注册

function register(string memory \_id, string memory \_name, string memory \_gender, uint \_age) public {

// 将患者的个人信息存储在映射中

patients[\_id] = Patient(\_name, \_gender, \_age, msg.sender);

}

/\*\*\*\*\*\*\*\*\*\*\*检索个人信息接口开发 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 检索个人信息接口开发 \*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\* 信息管理接口开发 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 信息管理接口开发 \*\*\*\*\*\*\*\*\*\*\*/

}

**子任务2-2-2：病历管理合约编码**

根据需求用例文档在待补充源码中完成病历管理合约的编码，解决代码错误和警告，正确编译合约，功能调试正确，运行新建病历、结束就诊接口功能截图保存。

1. 编写新建病历接口，实现检索病人对应科室既往病历，授权医生查看，如果没有既往病历则创建一个新的病历功能；

2. 编写结束就诊接口，实现检查病历是否已经填写，并结束病历咨询的功能。

待补充源码：

contract MedicalRecordManagement {

// 定义一个结构体用于存储病历

struct MedicalRecord {

string patientID; // 病人ID

string hospitalName; // 医院名称

string department; // 科室

string doctorName; // 医生姓名

string registrationInfo; // 信息管理信息

string pastMedicalHistory; // 既往病史

string currentMedicalHistory; // 现病史

bool isFilled; // 是否填写完毕

}

// 定义一个映射用于存储病历

mapping(string => MedicalRecord) medicalRecords;

/\*\*\*\*\*\*\*\*\*\*\* 新建病历接口 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 新建病历接口 \*\*\*\*\*\*\*\*\*\*\*/

// 定义一个函数，用于平台更新病历的信息管理和医生信息

function updateMedicalRecord(string memory patientID, string memory hospitalName, string memory department, string memory doctorName, string memory registrationInfo) public {

// 更新病历的信息管理和医生信息

medicalRecords[patientID].hospitalName = hospitalName;

medicalRecords[patientID].department = department;

medicalRecords[patientID].doctorName = doctorName;

medicalRecords[patientID].registrationInfo = registrationInfo;

}

// 定义一个函数，用于医生D填写病历

function fillInMedicalHistory(string memory patientID, string memory pastMedicalHistory, string memory currentMedicalHistory) public {

// 检查病历是否已经填写

require(bytes(pastMedicalHistory).length > 0 && bytes(currentMedicalHistory).length > 0, "a");

// 填写病历

medicalRecords[patientID].pastMedicalHistory = pastMedicalHistory;

medicalRecords[patientID].currentMedicalHistory = currentMedicalHistory;

medicalRecords[patientID].isFilled = true;

}

/\*\*\*\*\*\*\*\*\*\*\* 结束就诊接口 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 结束就诊接口 \*\*\*\*\*\*\*\*\*\*\*/

}

**子任务2-2-3：病历查看合约编码**

根据需求用例文档在待补充源码中完成病历查看合约的编码，解决代码错误和警告，正确编译合约，功能调试正确，运行合约中的检查退款请求状态、自动批准退款请求接口功能。

1. 编写查看病人个人信息接口，实现获取指定病人个人信息功能；
2. 编写查看病人病情描述接口，实现获取指定病人病情描述功能。

待补充源码：

contract MedicalRecords {

// 定义一个结构体，用于存储病人信息

struct Patient {

string name; // 病人姓名

uint age; // 病人年龄

string personalInfo; // 病人个人信息

string medicalHistory; // 病人病史

string medicalContent; // 病人病情描述

}

//将病人地址映射到病人信息

mapping(address => Patient) public patients;

// 添加病人信息

function addPatient(string memory \_name, uint \_age, string memory \_personalInfo, string memory \_medicalHistory, string memory \_medicalContent) public {

patients[msg.sender] = Patient(\_name, \_age, \_personalInfo, \_medicalHistory, \_medicalContent);

}

/\*\*\*\*\*\*\*\*\*\*\* 查看病人个人信息接口 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 查看病人个人信息接口 \*\*\*\*\*\*\*\*\*\*\*/

// 获取病人病史

function getMedicalHistory() public view returns (string memory) {

return patients[msg.sender].medicalHistory;

}

/\*\*\*\*\*\*\*\*\*\*\* 查看病人病情描述接口 \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\* 查看病人病情描述接口 \*\*\*\*\*\*\*\*\*\*\*/

}

**子任务2-2-4：合约部署和调用**

1. 解决代码错误和警告，正确编译合约，成功获取三个合约的abi，截图代码保存；

2．将三个合约形成一个文件部署至链上合约地址中，获取部署合约的地址信息，截图编写代码保存。

### 任务2-3：智能合约测试

根据需求用例文档完成医疗健康合约中病例管理的功能测试以及性能测试，解决病例管理合约代码正确性的验证以及验证系统响应时间是否满足预期响应效果，编写功能测试用例，实现查询病历、填写病历的业务测试，使用性能测试工具，对查询接口进行性能测试，将编写完成的功能测试用例截图保存。

1.根据需求用例文档在以下测试用例表格中编写病历管理合约中的测试用例，依据病历管理就诊中病历查询、就诊中病历填写、就诊结束病历查询三种状态，分析在前置条件为患者在医疗平台挂号为真的情况下，医生查询病历、修改病历后，患者有既往病史、没有既往病史的情况下，产生的平台权限与医生权限的归属结果。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例编号 | 用例标题 | 优先级 | 前置条件 | 测试步骤 | 测试数据 | 预期结果 |
| HM-MM001 | 病例管理  （就诊中病历咨询） | 高 | 请填写 | 请填写 | 请填写 | 请填写 |
| 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 |
| 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 |
| 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 |
| 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 | 请填写 |

2.利用测试工具，模拟测试查询病历接口发送200次请求，将所需HTTP请求、协议、服务器名称或IP以及路径填到工具Web服务器中，连同请求一起发送的参数，运行成功后将汇总报告提交到指定位置。

## 模块三：区块链应用系统开发（30分）

选手完成本模块的任务后，将任务中设计结果、运行代码、运行结果等截图粘贴至客户端桌面【区块链技术应用赛\重命名为工位号\模块三提交结果.docx】中对应的任务序号下。

### 任务3-1：区块链应用前端功能开发

完成区块链应用系统的医疗健康病历页面渲染，完成服务器端（后端）与Web端（前端）的接口的联调。要求如下：

1. 按照医疗健康病历详情原型图的长度、宽度、行高、间距、文字样式、颜色等，完成医疗健康病历详情页面的样式开发，将结果截图保存;

2. 完成Vue调用医疗健康病历查询接口API，获取接口返回的姓名、性别、科别、日期等信息，填充至Vue页面中。



医疗健康病历详情页面的样式开发补充源码：

|  |
| --- |
| .content{  width: 100%;  height: 100%;  background: #EEEEEE;  .contentModel{  position: absolute;  top: 50%;  left: 50%;  box-sizing: border-box;  width: 700px;  padding: 41px 30px 40px;  background: #fff;  border-radius: 8px;  box-shadow: 0 3px 12px 0 #c0c4cc;  transform: translate(-50%, -50%);  -ms-transform: translate(-50%, -50%);  -moz-transform: translate(-50%, -50%);  -webkit-transform: translate(-50%, -50%);  -o-transform:translate(-50%, -50%);  }  .case\_title{  text-align: center;  font-size: 23px;  line-height: 36px;  }  .personInfor{  // 此处代码补全:按原型图样式，进行长度、宽度、行高、间距、文字样式、颜色等样式设置；  }  .caseList{  li{  line-height: 40px;  overflow: hidden;  font-size: 15px;  span{  float: left;  width: 110px;  }  div{  float: left;  width: calc(100% - 110px);  }  }  }  .printTime{  margin-top: 11px;  padding-top: 20px;  border-top: 1px solid #222222;  }  } |

医疗健康病历详情页面的Html开发补充源码：

|  |
| --- |
| <template>  <div class="content">  <div class='contentModel'>  <div class='case\_title'>  北京市某医院<br>  门诊病例  </div>  <div class='personInfor'>  <!-- 此处代码补全:按照原型图格式，画出姓名、性别、科别、日期等信息 -->  </div>  <ul class='caseList'>  <li>  <span>主诉：</span>  <div>1</div>  </li>  <li>  <span>现病史：</span>  <div>突发症状，无治疗经历</div>  </li>  <li>  <span>既往史：</span>  <div>无，体健</div>  </li>  <li>  <span>药物过敏史：</span>  <div>否认</div>  </li>  <li>  <span>个人及家族史：</span>  <div>家族内成员，无相似病史及家族遗传病史</div>  </li>  <li>  <span>体格检查：</span>  <div>无</div>  </li>  <li>  <span>辅导检查：</span>  <div>1</div>  </li>  <li>  <span>初步诊断：</span>  <div>1</div>  </li>  <li>  <span>诊疗：</span>  <div>不适随诊</div>  </li>  </ul>  <div class='printTime'>  打印时间：2023-04-14  </div>  </div>  </div>  </template> |

### 任务3-2：区块链应用后端功能开发

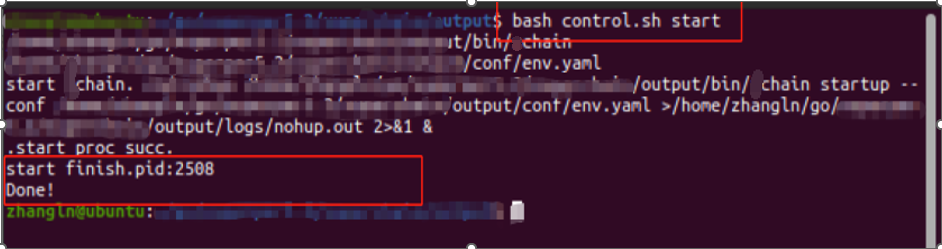
**子任务3-2-1：获取区块链交易信息**

区块链应用系统开发需要区块链底层网络进行支撑，开发链上数据交互功能，实现与节点建立链接，完成链上信息查询。要求如下：

（1）打开区块链网络文件目录，通过区块链启动脚本，输入bash control.sh start命令，启动区块链网络，将输出结果截图保存；

（2）打开（src/controller/BlockController.java）文件，在查询区块链信息接口中，使用Java-SDK获取区块链的最新高度和最新交易Hash，并将结果按十进制的整数和字符串类型返回。

启动区块链源码截图：



区块链的最新高度和最新交易Hash接口补充源码：

|  |
| --- |
| /\*\*获取最新区块高度和最新交易Hash\*/  @RequestMapping(value = "/getBlockInformation")  @ResponseBody  public Map<String,Object> getBlockInformation(FlightInformation flightInformation){  Map<String,Object> resMap = new HashMap<>();  try{    // 获取最新区块高度  long blockHeight = block.getLong("height");  System.out.println("最新区块高度：" + blockHeight);  // 获取最新区块交易Hash  String txHashes = block.getJSONArray("txHashes");  System.out.println("最新区块交易Hash：" + txHash);    resMap.put("code",200);  resMap.put("message","请求成功");  resMap.put("data",null);  } catch (ClientProtocolException e) {  log.error("操作失败—log：",e);  resMap.put("code",500);  resMap.put("message","请求失败");  } catch (IOException e) {  log.error("操作失败—log：",e);  resMap.put("code",500);  resMap.put("message","请求失败");  }catch (Exception e){  log.error("操作失败—log：",e);  resMap.put("code",500);  resMap.put("message","请求失败");  }  return resMap;  } |

**子任务3-2-2：声明实体类和设计数据库**

（1）根据“查询医疗健康病历合约”中变量的字段，在Java项目中声明实体类（MedicalHealthCase），将声明代码结果截图保存；

包含字段：用户姓名（name）、年龄（age）、性别（sex）、科别（category）、时间（time）。

（2）请打开Mysql命令行，连接数据库，根据第1步声明的实体类，创建数据库表（medical\_health\_case）与表结构字段。

医疗健康病历合约实体字段补充源码：

|  |
| --- |
| /\*\*  \* 医疗健康病历实体实体  \*/  public class MedicalHealthCase {  private static final long serialVersionUID = 1L;  //在此处进行代码补全，声明医疗健康病历实体字段，并添加Get和Set方法  } |

创建medical\_health\_case表补充源码：

|  |
| --- |
| CREATE TABLE `medical\_health\_case` (  `id` INT NOT NULL AUTO\_INCREMENT COMMENT '主键id',  //在此处进行建表sql语句补全或使用工具创建表字段  `create\_time` DATETIME NOT NULL COMMENT '创建时间',  `update\_time` DATETIME NOT NULL COMMENT '修改时间',  PRIMARY KEY (`id`) USING BTREE  )  COLLATE='utf8\_general\_ci'  ENGINE=InnoDB  ROW\_FORMAT=COMPACT  ; |

**子任务3-2-3：编写调用查询医疗健康病历合约接口**

通过Java-SDK完成调用查询医疗健康病历合约接口，获取医疗健康病历信息。要求如下：

（1）接收从Web端接收对应各种参数（包括病历编号）将代码截图保存；

（2）调用Java-SDK，运行调用智能合约API，接收医疗健康病历信息传递给前端页面，将代码截图保存；

（3）医疗健康病历信息查询成功后，将获取到的医疗健康病历信息进行解析，并通过数据库依赖包（mysql-connector-java-bin.jar）存储到数据库中。

|  |
| --- |
| /\*\*获取医疗健康病历详情接口\*/  @RequestMapping(value = "/getUseMedicalHealthCase")  @ResponseBody  //此处代码补全:接收从Web端传输的参数（包含钱包地址）  public Map<String,Object> getUserMedicalHealthCase(…){  Map<String,Object> resMap = new HashMap<>();  try{  //此处代码补全:调用Java-SDK，运行调用智能合约API，接收医疗健康病历信息传递给前端页面  //此处代码补全：医疗健康病历信息查询成功后，将获取到的医疗健康病历信息进行解析，并通过数据库依赖包（mysql-connector-java-bin.jar）存储到数据库中  resMap.put("code",200);  resMap.put("message","请求成功");  resMap.put("data",result);  } catch (Exception e){  log.error("操作失败—log：",e);  resMap.put("code",500);  resMap.put("message","请求失败");  }  return resMap;  } |