

**课程报告**

|  |  |
| --- | --- |
| **课程名称：** | **企业软件项目实训** |
| **学生姓名：** | **张扬** |
| **学生学号：** | **201630666370** |
| **学生专业：** | **软件工程卓越班** |
| **开课学期：** | **2018-2019第二学期** |

**软件学院**

**2019年6月**

**目录**

[**一、** **区块链基础知识** 1](#_Toc13319977)

[1. 区块链技术原理 2](#_Toc13319978)

[2. 联盟链和公有链的异同 2](#_Toc13319979)

[3. 分布式存储优势 3](#_Toc13319980)

[**二、** **区块链应用场景** 3](#_Toc13319981)

[**三、** **FISCO-BCOS平台** 4](#_Toc13319982)

[**四、** **智能合约** 5](#_Toc13319983)

[1. 智能合约练习 5](#_Toc13319984)

[2. Gas在智能合约中的作用 6](#_Toc13319985)

[**五、** **Spring-boot学习** 6](#_Toc13319986)

[**六、** **项目成果——MusicChain** 8](#_Toc13319987)

[1. 需求分析 8](#_Toc13319988)

[2. 用户角色分析 9](#_Toc13319989)

[3. 用户 11](#_Toc13319990)

[4. 功能性需求 12](#_Toc13319991)

[5. 非功能需求 14](#_Toc13319992)

[6. 逻辑部署方案 14](#_Toc13319993)

[7. 区块链网络部署 15](#_Toc13319994)

[8. 智能合约编写 16](#_Toc13319995)

[9. 后台实现 19](#_Toc13319996)

[10. 前端实现 26](#_Toc13319997)

[11. 软件测试 26](#_Toc13319998)

[**七、** **区块链未来展望** 26](#_Toc13319999)

[1. 隐私保护 27](#_Toc13320000)

[2. 分布式共识 27](#_Toc13320001)

[3. 交易性能 28](#_Toc13320002)

[**八、** **个人感想** 29](#_Toc13320003)

[参考文献 29](#_Toc13320004)

1. **区块链基础知识**

区块链[1][2]的出现如同一场大火席卷了全球，各行各业都前赴后继的加入到了区块链的研发中来，我们国家也不例外。随着BAT、360、京东等大互联网公司的入局，国内区块链的发展速度已经来到了一个新的高度，区块链技术的发展前景也变得越加清晰。简单来说区块链技术其实就是一种简单便捷的全民记账模式，记账中的每个模块后面都有一个数据库，并且每个数据库之间都是独立的，大家也可以把数据库当作是一个大的账本。

1. 区块链技术原理[3][4]

区块链系统由数据层、网络层、共识层、激励层、合约层和应用层组成。 其中，数据层封装了底层数据区块以及相关的数据加密和时间戳等基础数据和基本算法；网络层则包括分布式组网机制、数据传播机制和数据验证机制等；共识层主要封装网络节点的各类共识算法；激励层将经济因素集成到区块链技术体系中来，主要包括经济激励的发行机制和分配机制等；合约层主要封装各类脚本、算法和智能合约，是区块链可编程特性的基础；应用层则封装了区块链的各种应用场景和案例。该模型中，基于时间戳的链式区块结构、分布式节点的共识机制、基于共识算力的经济激励和灵活可编程的智能合约是区块链技术最具代表性的创新点。

1. 联盟链和公有链的异同

公有链是对所有人公开，用户不需要注册和授权就能够匿名访问网络和区块，任何人都可以自由加入和退出网络，并参与记账和交易。公有链是真正完全意义上的去中心化区块链，它通过密码学（非对称加密）算法保证了交易的安全性和不可篡改性，在陌生的网络（非安全）环境中，建立了互信和共识机制。在公有链中共识机制一般是工作量证明（POW）和权益证明（POS）。

联盟链仅限于联盟成员，因其只针对成员开放全部或部分功能，所以联盟链上的读写权限、以及记账规则都按联盟规则来“私人定制”。联盟链上的共识过程由预先选好的节点控制，一般来说，他适用于机构间的交易、结算、或清算等B2B场景。联盟链几乎不采用工作量证明共识机制而是采用权益证明或PBTF等共识算法。

1. 分布式存储优势

随着互联网的飞速发展、应用越来越丰富、用户数量越来越多、数据也成几何级增长，海量数据的存储给本地存储带了巨大压力，存储系统已经不堪重负，处于崩溃的边缘，因此，必须通过其他手段分散存储系统压力，分布式存储和分布式文件系统应运而生。

分布式账本[5]具有高性能。分布式存储通过将热点区域内数据映射到高速存储中，来提高系统响应速度;一旦这些区域不再是热点，那么存储系统会将它们移出高速存储。同时，其支持分级存储。由于通过网络进行松耦合链接，分布式存储允许高速存储和低速存储分开部署，或者任意比例混布。在不可预测的业务环境或者敏捷应用情况下，分层存储的优势可以发挥到最佳。

1. **区块链应用场景**

区块链技术已经从单纯的技术探讨走向了应用落地的阶段。国内外已经出现大量与之相关的企业和团队。有些企业已经结合自身业务摸索出了颇具特色的应用场景，更多的企业还处于不断探索和验证的阶段。

区块链在不引入第三方中介机构的前提下，可以提供去中心化、不可篡改、安全可靠等特性保证。因此，所有直接或间接依赖于第三方担保机构的活动，均可能从区块链技术中获益。区块链自身维护着一个按时间顺序持续增长、不可篡改的数据记录，当现实或数字世界中的资产可以生成数字摘要时，区块链便成为确权类应用的完美载体，提供包含所属权和时间戳的数字证据。未来几年内，可能深入应用区块链技术的场景将包括：

金融服务：区块链带来的潜在优势包括降低交易成本、减少跨组织交易风险等。该领域的区块链应用目前最受关注，全球不少银行和金融交易机构都是主力推动者。部分投资机构也在应用区块链技术降低管理成本和管控风险。从另一方面，要注意可能引发的问题和风险。

征信和权属管理：征信和权属的数字化管理是大型社交平台和保险公司都梦寐以求的。目前该领域的主要技术问题包括缺乏足够的数据和分析能力；缺乏可靠的平台支持以及有效的数据整合管理等。区块链被认为可以促进数据交易和流动，提供安全可靠的支持。征信行业的门槛比较高，需要多方资源共同推动。

资源共享：以 Airbnb 为代表的分享经济公司将欢迎去中心化应用，可以降低管理成本。该领域主题相对集中，设计空间大，受到大量的投资关注。

1. **FISCO-BCOS平台**

FISCO BCOS平台是金融区块链合作联盟（深圳）开源工作组以金融业务实践为参考样本，在BCOS开源平台基础上进行模块升级与功能重塑，深度定制的安全可控、适用于金融行业且完全开源的区块链底层平台。

在本次实训项目中，微众银行的老师教学的平台为FISCO BCOS 2.0。此版本在原有基础上进行架构升级和优化，在可扩展性、性能、易用性等方面取得了重大突破：（1）实现群组架构，在多个节点组成的一个全局网络中，可以存在多个节点子集组成的子网络，这些子网络维护一个独立的账本。这些账本之间的共识、存储都是相互独立的，具备良好的扩展性和安全性。在群组架构中，可以更好地实现平行扩展，满足金融级高频交易场景的需求。（2）支持分布式存储，使存储突破单机限制，支持横向扩展。计算和存储分离，提高了系统健壮性，即使节点执行服务器故障，数据也不会受影响。（3）实现预编译合约框架，突破EVM性能瓶颈。支持交易并发处理，大幅提升交易处理吞吐量。

FISCO BCOS 2.0版本持续在网络传输模型、计算存储流程等方面进行优化，对性能提升提供巨大帮助。在架构方面，在存储、网络、计算三个角度，围绕高可用性和高易用性进行持续升级。基于模块化、分层、可插拔等设计原则，持续对核心模块进行重塑升级，保证系统健壮性。

1. **智能合约**

智能合约[6]是一种旨在以信息化方式传播、验证或执行合同的计算机协议。智能合约允许在没有第三方的情况下进行可信交易，这些交易可追踪且不可逆转。智能合约概念于1995年由Nick Szabo首次提出。

1. 智能合约练习

在实训的第三周，微众银行的老师带我们学习了智能合约，我们同时也自动动手实现了一份智能合约，并将其部署在区块链上，进行调用。

本次实验我们实现一个积分系统，由商家进行积分发放，用户进行积分消费。功能为总积分初始化、总积分查询、积分转账、积分查询、积分转账明细记录。具体代码不在此展示，可上Github查看。

1. Gas在智能合约中的作用

Gas是以太坊使用的特殊单位的名称。它衡量一个动作或一系列动作需要执行多少“工作”。Gas的重要性在于它有助于确保提交给网络的交易支付适当的费用。 通过要求交易支付每个操作的执行（或导致合同执行），我们确保网络不会因为执行大量对任何人无价值的密集工作而陷入困境。 这与比特币交易费用不同，它仅基于交易的千字节大小。 由于以太坊允许运行任意复杂的计算机代码，所以短的代码实际上可能导致大量计算工作的完成。 所以衡量直接完成的工作非常重要，而不是仅仅根据交易或合同的长度选择费用。

1. **Spring-boot学习**

SpringBoot是基于Spring4.0设计，不仅继承了Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了Spring应用的整个搭建和开发过程。另外SpringBoot通过集成大量的框架使得依赖包的版本冲突，以及引用的不稳定性等问题得到了很好的解决。

SpringBoot框架中还有两个非常重要的策略：开箱即用和约定优于配置。开箱即用，Outofbox，是指在开发过程中，通过在MAVEN项目的pom文件中添加相关依赖包，然后使用对应注解来代替繁琐的XML配置文件以管理对象的生命周期。这个特点使得开发人员摆脱了复杂的配置工作以及依赖的管理工作，更加专注于业务逻辑。约定优于配置，Convention over configuration，是一种由SpringBoot本身来配置目标结构，由开发者在结构中添加信息的软件设计范式。

Gradle是一个基于JVM的构建工具，是一款通用灵活的构建工具，支持maven， Ivy仓库，支持传递性依赖管理，而不需要远程仓库或者是pom.xml和ivy.xml配置文件，基于Groovy，build脚本使用Groovy编写。

因为在这之前从来没有接触过后台开发，更没有接触过Spring-boot，所以在第一次看见这种注释的时候还是很奇怪的，不过慢慢的也理解到注释的强大功能，很大程度上提高了程序员的开发效率。下面列出了几个常用注释。

* @SpringBootApplication：包含了@ComponentScan、@Configuration和@EnableAutoConfiguration注解。其中@ComponentScan让spring Boot扫描到Configuration类并把它加入到程序上下文。
* @Configuration 等同于spring的XML配置文件；使用Java代码可以检查类型安全。
* @EnableAutoConfiguration 自动配置。
* @ComponentScan 组件扫描，可自动发现和装配一些Bean。
* @Component可配合CommandLineRunner使用，在程序启动后执行一些基础任务。
* @RestController注解是@Controller和@ResponseBody的合集,表示这是个控制器bean,并且是将函数的返回值直 接填入HTTP响应体中,是REST风格的控制器。
* @Autowired自动导入。
* @PathVariable获取参数。
* @JsonBackReference解决嵌套外链问题。
* @RepositoryRestResourcepublic配合spring-boot-starter-data-rest使用。

1. **项目成果——MusicChain**

中国的音乐市场正在快速发展，高速增长的音乐市场规模和巨大的音乐市场体量吸引了越来越多的人和企业加入到这个战场中。自2015年中国音乐版权的觉醒以来，涌现了众多数字音乐的授权平台，如100Audio、V.Fine Musci等，他们一方面保护着音乐人的音乐版权不受侵犯，另一方面为众多的音乐人提供音乐宣传推广的平台，提高音乐人的收入。

由此，针对音乐版权纠纷问题、音乐人权益保障、音乐授权使用效率的提高等问题，我们提出了该区块链解决方案。为音乐人、版权企业、音乐使用的普通用户、音乐使用的企业和仲裁机构提供统一化的平台，此外，企业用户可另外构建系统加入该区块链中。

1. 需求分析

中国的音乐市场相较欧美国家仍然有较大的差距。我国严厉打击网络侵权以来，据不完全统计，仍然有6900多起关于音乐版权纠纷的案件发生。在音乐授权的平台上，音乐人将音乐上传到平台后，音乐的宣传力度是有限的，同时，他们所获得的收益仅为40%至70%不等。音乐人在中国音乐著作权协会中登记音乐版权后，他们的收益到账速度较慢，需要等待较长的周期才能完成收益的分配。当用户希望获得在中国音乐著作权协会中登记了版权的音乐的使用权时，需要通过协会的申请，其中需要较多次经过协会才能完成的手续。

《著作权法》中有关权利限制的一项规定：“免费表演已经发表的作品，该表演未向公众收取费用，也未向表演者支付报酬“的，可以不经著作权人许可，不向其支付报酬。从2015-2019年统计的侵权案件情况来看，知识产权的案件达六万多件，在放映权、著作权、等等权利上仍然会存在较大的纠纷。音乐侵权的行为还是常常可见，有意和无意的侵权在不停地干扰着音乐产业的发展。

音乐人生存现状：收入增长较慢，收入较低，存在较大的增长空间；收入来源中，版权收入占比低；维权难以亲自依靠法律武器进行维权，维权的消耗大。

1. 用户角色分析
   1. 角色

用户角色从整体来看，分为3中类型，音乐产生者，音乐使用者，纠纷仲裁机构。

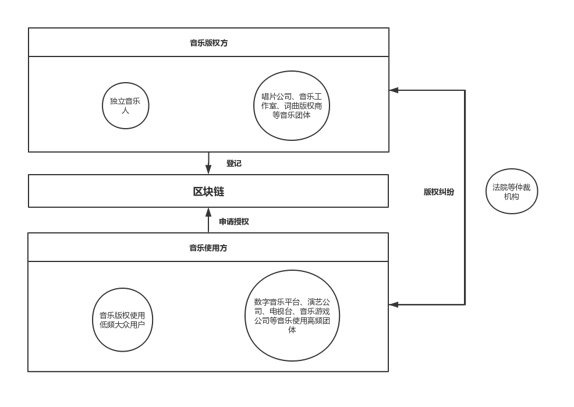


图1 角色联系

* + 1. 独立音乐人

如，网易云音乐中的独立音乐人。从调研来看，音乐人的主要需求在于音乐版权的管理，以及音乐的授权问题。音乐人能够在我们的平台上登记他们的音乐版权，能够对版权进行转让、授权，同时在必要时能够对音乐版权的注销。他们通过平台直接与版权使用方进行版权的交易，从而提高他们的收入，加快收入的到账速度。

* + 1. 版权所有企业

如，环球音乐、索尼音乐、华纳音乐等这样大型的音乐公司。对于版权所有企业来说，最重要的目标是通过版权的售卖和授权使用能够以最大效率来获得收益。因此，一方面需要为他们提供一个能够快速授权的平台，另一方面，能够为他们授予一个保护他们版权的平台。

* + 1. 版权使用企业

如，网易云音乐、QQ音乐等大型数字音乐平台以及网易游戏这样需要常常使用音乐素材制作游戏的公司。版权使用企业可能存在大量地使用音乐的情况，也可能存在各个小部门会需要使用少量音乐的情况。大量使用音乐时，企业之间可以通过区块链进行合作来共享音乐；当企业只需要使用少量音乐时，可以直接进行单曲授权，从而快速获得版权的授予。

* + 1. 版权使用普通用户

如抖音平台、直播平台中的网红、中小型的游戏制作方、音乐改造爱好者等普通用户。对于普通用户来说，他们使用音乐往往在数量较少，能够快速获得音乐的授权对这类用户群体来说显得更加重要。

* + 1. 仲裁机构

仲裁机构是在处理版权纠纷时，能够快速地查询并对版权的信息和历史记录进行取证，快速判案，通过区块链中的数据提高解决案件的效率。

* 1. 参与业务

独立音乐人：版权注册、版权交易、作品注销；

版权企业团体：版权注册、版权交易、作品注销、版权共享；

普通使用者：获取复制权、表演权、广播权、信息网络传播权；

企业团体：获取复制权、表演权、广播权、信息网络传播权、共享使用；

仲裁机构：获取版权信息、版权历史记录；

1. 用户

在系统中，用户设计为四种类型：普通用户、音乐人、企业、仲裁机构。以下为各种类型的用户对应的功能，其中，普通用户、企业、音乐人考虑到了他们会申请其他音乐使用权的授权，因此都提供了申请授权功能。而音乐人、企业、仲裁机构三种类型的用户需要进行认证才可确定对应的身份类型。此外，考虑到企业会购买版权、雇佣音乐人来制作音乐等问题，直接让企业版权的拥有方和使用方具有相同的功能，在使用时，他们根据自己的身份和需要进行使用即可。



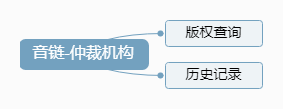


图2 用户具有功能

1. 功能性需求

|  |  |  |  |
| --- | --- | --- | --- |
| 功能名称 | 用户角色 | 描述 | 备注 |
| 账号注册 | 普通用户 | 用户输入姓名与手机号码注册，获取系统返回的公钥和私钥。 |  |
| 登录 | 普通用户、音乐人、企业用户、仲裁机构 | 各类用户使用自己的私钥进行登录。 |  |
| 音乐人认证申请 | 音乐人 | 音乐人填写姓名、身份证号码、所在地、联系电话、电子邮箱等信息，申请认证音乐人。 |  |
| 企业认证申请 | 企业用户 | 企业用户填写姓名、企业编码、所在地、联系电话、电子邮箱等信息，申请认证企业用户。 |  |
| 仲裁机构认证申请 | 仲裁机构 | 仲裁机构填写机构名称、所在地、联系电话、电子邮箱等信息，申请认证仲裁机构。 |  |
| 版权信息查询 | 任何用户 | 任何用户都可在音乐区输入音乐作品的名称、作者名字即可进行查询音乐作品的版权人信息。 |  |
| 版权授权申请 | 普通用户、音乐人、企业用户 | 用户在音乐区查询音乐后，点击申请授权，填写授权需要的购买用户、授权地域、授权期限、其他说明、授权人姓名、联系电话等信息后付款即将授权申请单提交给版权人。 |  |
| 版权详细信息查询 | 仲裁机构 | 仲裁机构登录后，输入作品名称和作者名字即可查询作品的详细信息。 |  |
| 版权历史记录查询 | 仲裁机构 | 仲裁机构登录并输入作品名称和作者名字后可查看作品交易的详细记录。 |  |
| 仲裁机构信息查看 | 仲裁机构 | 仲裁机构登录后，可查看仲裁机构的名称、所在地、联系电话等信息 |  |
| 用户信息查看 | 普通用户、音乐人、企业用户 | 用户登录后，可查看用户名称、编码/身份证号码、所在地、联系电话、电子邮箱等信息 | 普通用户仅有名字和联系电话信息 |
| 授权订单查看 | 普通用户、音乐人、企业用户 | 用户登录后，可进入授权订单中，查看用户申请的授权，当授权被同意后，电子授权证明可使用。 |  |
| 电子授权证明查看 | 普通用户、音乐人、企业用户 | 用户登录后，可进入授权订单中，查看已被授权的作品的电子授权证明。 |  |
| 版权转让 | 音乐人、企业用户 | 用户登陆后，可在音乐版权列表中，将对应的音乐进行转让给其他用户，填写其他用户的公钥后，即可将版权所属权转让给公钥对应的用户。 |  |
| 版权注销 | 音乐人、企业用户 | 用户登录后，可在音乐版权列表中，将对应的某首作品进行注销。 |  |
| 版权登记 | 音乐人、企业用户 | 用户登录后，在个人中心中，使用版权登记功能，填写作品的信息后即可申请作品的版权登记。 |  |
| 授权申请判定 | 音乐人、企业用户 | 用户登录后，用户在授权申请列表中，对应申请授权的授权单进行确定同意与否。 |  |
| 版权共享 | 企业用户 | 企业用户登录后，针对已经确定需要进行音乐版权共享的情况，将音乐版权与其他企业或用户进行共享，给予他们相应的使用权力。 |  |

表1 功能需求

1. 非功能需求
   1. 安全性

要求用户登录使用的公钥和私钥，在平台上均不作保存，全权由用户自己进行保存，确保平台的可信任度以及用户账户的安全性；平台更多发挥一种接口的作用来为用户提供服务，核心的技术均使用FISCO-BCOS的技术，确保整个系统的安全；

* 1. 隐私性

用户的公钥和私钥均有自己进行保存，平台只提供接口进行使用公钥和私钥登录，从而确保用户信息隐私；音乐人和版权企业所具有的音乐版权，只能通过查询入口一首一首地查询，而无法批量查询，确保了音乐人和企业所具有的版权的隐私；

1. 逻辑部署方案

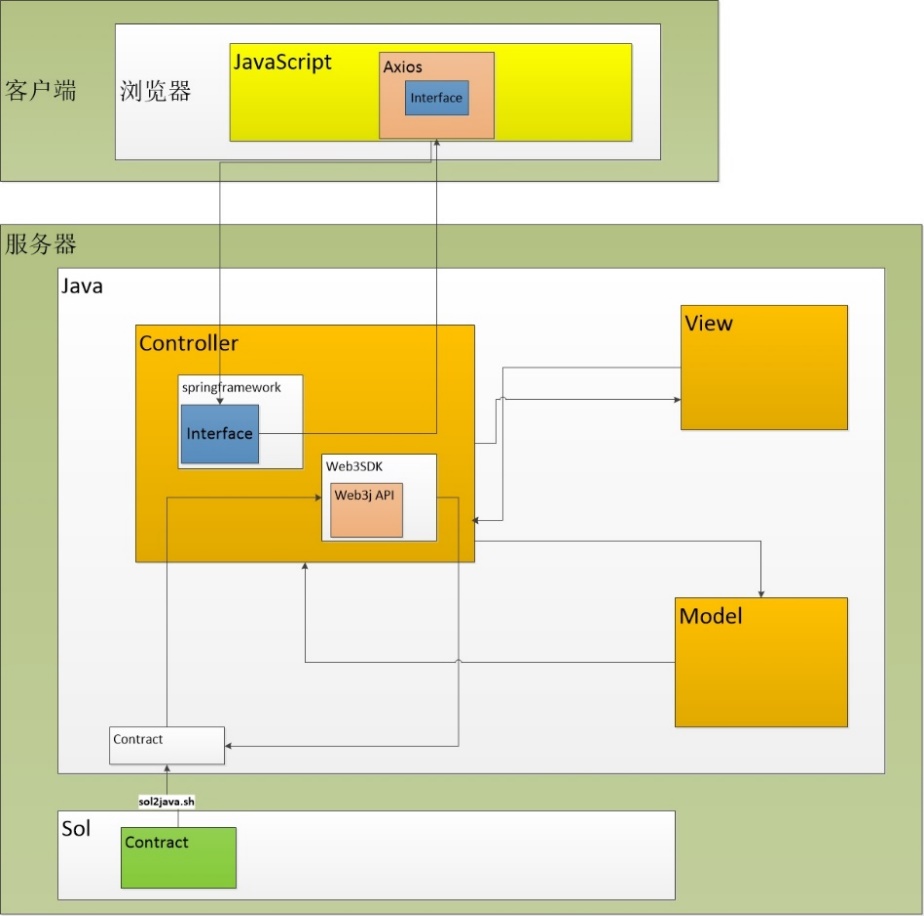
****

图3 逻辑部署方案

1. 区块链网络部署

本项目构建的4节点2结构1群组的组网模式的区块链结构如图：

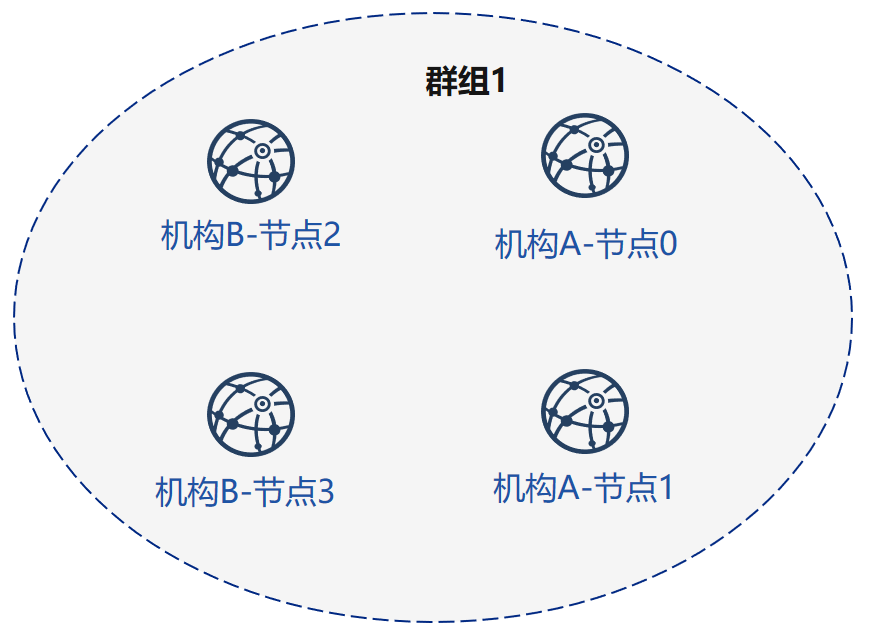


图4 组网模式

每个节点的IP，端口号及所属信息如下表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 机构 | 节点 | 所属群组 | P2P地址 | RPC/channel监听地址 |
| 机构A | 节点0 | 群组1、2 | 127.0.0.1:30300 | 0.0.0.0:8545/:20200 |
| 机构A | 节点1 | 群组1、2 | 127.0.0.1:30301 | 0.0.0.0:8546/:20201 |
| 机构B | 节点2 | 群组1 | 127.0.0.1:30302 | 0.0.0.0:8547/:20202 |
| 机构B | 节点3 | 群组1 | 127.0.0.1:30303 | 0.0.0.0:8548/:20203 |

表2 节点信息

需要注意的是，为了开发人员能够在开发后台的时候可以直接调试，我们允许外网进行访问，因此监听地址设置为广播地址，而对于后台访问云服务器的时候，其设置的P2P参数则为：



图5 P2P参数设置

其中，云服务器端的公网ip为101.132.68.46。端口设置为搭建联盟链时设定的端口参数。

1. 智能合约编写
   1. 基本组件设计

参考用户角色分析，用户可以分为普通用户、音乐人、企业和仲裁机构，前者通过申请认证功能申请成为不同的用户类型。普通用户通过姓名和手机号码进行注册，生成UserEntity实例，其中的id、地址、邮箱属性在用户进行认证时接收用户对应的输入

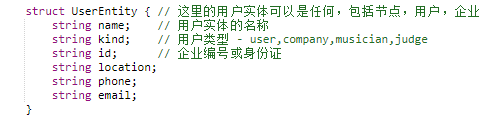


图6 用户参数

用户可以通过登记版权功能将自己的音乐上传到音链上，因此需要基本的Music架构记录用户所上传的音乐。Music通过唯一的二进制hash来对每一首不同的音乐进行标识，mname、alltime、singer等属性在用户调用register函数时输入

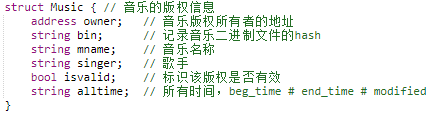


图7 音乐参数

基本架构Record用以记录用户执行的授权操作，在用户执行同意授权、版权转让和注销音乐操作时会生成相应的Record记录用户的操作，方便用户进行查询，genre记录用户操作的类型，user和author分别记录该操作涉及的两个用户的地址，music、info、alltime记录与Record生成的音乐内容、Record内容和时间信息

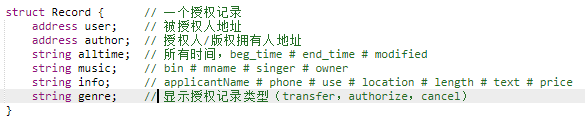


图8 授权记录参数

基本架构Notice在申请方向版权方提出授权申请时生成，在版权方登陆账号查看时提示版权方进行申请处理。start、to分别记录通知发起方和接收方的地址，music和info储存该通知涉及的音乐信息和申请人信息，最后用valid记录申请受理状况

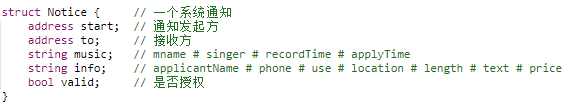


图9 系统通知参数

利用mapping将用户地址和UserEntity一一对应，另外用数组储存Music、Record和Notice记录

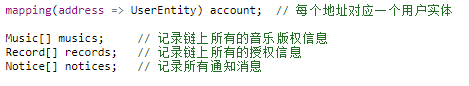


图10 其他参数

* 1. 基本注册功能

用户首次注册通过调用registerUser，输入姓名和电话，生成与其地址唯一对应的UserEntity

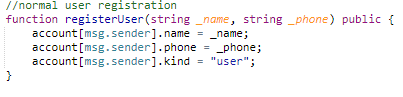


图11 用户注册函数

用户通过registerCompany、registerMusician、registerJudge申请认证为企业、音乐人、仲裁机构的一种，并在申请时完善id、地址、邮箱等信息。其函数于用户注册类似，就不在贴出。

同时，用户在上传音乐时需要调用registerMusic函数，函数从后台接收binhash、时间信息alltime以及用户输入的音乐名称，将生成的音乐存储添加进musics数组中。

申请方在填入数据并点击申请时需要调用registerNotice函数，该函数接收版权方地址、音乐信息以及申请信息作为输入，生成一条notice并加入到数组中，当版权方登陆时，在授权申请界面将会看到这条申请。

* 1. 功能函数

transferMusic实现了版权转让的功能，函数接收目标用户地址、音乐hash码和时间信息为输入，通过修改该music对象的owner地址和时间信息，记录下新的版权拥有者和版权起始、结束时间等信息，最后通过生成一条record记录下这首音乐的版权转让操作，方便用户的后续查询。

在版权方接收到申请方的notice后，他可以查看其中的申请内容，如果版权方同意将版权授予申请方，那么他在点击同意时将会调用authorizeMusic函数，该函数从后台接收申请方地址、音乐hash码、时间信息、音乐信息和申请方信息，并生成一条record记录这首音乐的授权操作。

cancelMusic对应版权撤销功能，当这首音乐的版权拥有者点击确定注销以后，cancelMusic将会更改这首音乐的valid标识符，同时生成一条record记录下版权拥有者对这首音乐的版权撤销操作。

同时，我们还有一类辅助函数的主要作用是完善与后台数据的交互，get函数主要是将储存在数组中的数据返回到后台中，后台再通过调用相关函数在前端进行显示。

1. 后台实现
   1. 环境设置

Gredle：5.3.1

Jdk：1.8.0\_211

Ide: IntelliJ IDEA

* 1. 后端设计



图12 后台架构

后端的整体架构大约分为3层：数据接口、服务层和Controller层，结构如上图所示。（1）数据接口：在智能合约转化成java文件以后，对其暴露的函数接口进行封装，使接口更加美观，方便服务层进行调用。（2）服务层：利用数据接口实现需求，使其可以被Controller层所调用。（3）Controller：响应前端的请求，并返回所需的数据。

* 1. 具体结构

下面陈述具体的项目结构设计，列出具体的包以及类。

* + 1. Package：autoconfigure

存放系统所需的配置。在此不做详述。

* + 1. Package: constants

存放系统常量，具有下列class：

ConnectConstant：

CONNECT\_SECONDS = 30;

CONNECT\_SLEEP\_PER\_MILLIS = 1

TIME\_OUT = 30000;

GasConstants：

BigInteger GAS\_PRICE = new BigInteger("300000000")

BigInteger GAS\_LIMIT = new BigInteger("300000000")

* + 1. Package: contracts

用于存放已经转化好的智能合约。

* + 1. Package: controller

用于存放相关控制器，用于响应前台的响应。将在后文详细阐述。

* + 1. Package: entity

用于存放相关实体类。将在后文详细阐述。

* + 1. Package: function

用于包装智能合约的接口

* + 1. Package: service

使用function中提供的接口，提供服务。并存放其他所需函数。

* 1. 具体实例

下面是具体实例的详细内容，每个类中都要生成两个构造函数，一个接受所有参数值，一个不接受值（出Result类以外）。每个类中的每个参数都生成get和set方法。

9.4.1 Music: 歌曲

private String mName; //歌曲名称

private String singer; // 歌手

private String owner; // 版权所有者

private String bin; // 音乐二进制文件哈希

private boolean isValid; // 版权是否有效

private String alltime; //所有时间，beg\_time # end\_time # modified

9.4.2 Notice: 由用户发起的交易

private String start; // 通知发起方

private String to; // 接收方

private String music; // mname # singer # recordTime # applyTime

private String info; // applicantName # phone # use # location # length # text # price

private boolean valid; // 是否授权

9.4.3 Record: 交易成功记录

private String user; // 被授权人地址

private String author; // 授权人/版权拥有人地址

private String alltime; // 所有时间，beg\_time # end\_time # modified

private String music; // bin # mname # singer # owner

private String info; // applicantName # phone # use # location # length # text # price

9.4.4 User: 用户信息

private String name; //姓名或企业名称

private String id; //身份证号码或者企业编码

private String location; //所在地

private String phone; //联系电话

private String email; //电子邮箱

private String type; // 用于判断是那种类型的用户

9.4.5 RecordInformation: 信息综合

private Music music;

private User user;

9.4.6 Result: 返回给前端的特殊数据类型

private int success; //判断结果

private String message; //返回信息

private Object data; //可以存放不同类型的数据进行返回

* 1. 具体接口

该部分具体阐述在Controller层于前端的接口。

所有接口均返回Result类型的数据，其中包括结果信息代号、结果信息和前端需要的具体数据，若无需具体数据返回，则参数data为空。下文所指返回数据为Result类中的data参数，若没有写出则说明data为空。

9.5.1 CompanyController：用于企业认证

（1） RegisterCompany

作用：完成企业认证

输入：name,id,location,phone,email

9.5.2 JudgeController：用于仲裁页面

（1） Register

作用：用于完成仲裁认证

输入：name,id,location,phone,email

（2） SearchingRecord

作用：版权查询，响应点击搜索按钮

输入：musicName,singer

返回：recordInformation

（3） JudgeInformation

作用：返回仲裁机构信息

输入：无

返回：User

9.5.3 LoginController：用于登陆注册页面

（1） Register

作用：用于响应点击注册按钮

输入：name,phone

返回：List<String> 用户的私钥和地址

（2） Login

作用：响应点击登陆

输入：用户私钥

9.5.4 MainController: 用于音乐区

（1） Search

作用：搜索音乐

输入：musicName,singer

返回：Music

（2） ConfirmAuthorize

作用：响应确定申请授权

输入：music,to,info

9.5.5 MusicianController：用于响应音乐人界面

（1） RegisterMusician

作用：响应音乐人注册

输入：name,id,location,phone,email

9.5.6 UserController：用于相应个人中心

（1） enterpriseInformation

作用：显示用户信息，可用于企业、音乐人、普通用户

输入：无

返回：User

（2） ShowNoticeStartByMe

作用：显示由我发出的所有Notice

输入：无

返回：List<Notice> Notice的列表

（3） ShowMuiscOwnedByMe

作用：显示所有拥有者为我的音乐

输入：无

返回：List<Music> 音乐列表

（4） RecordTransfer

作用：应用于版权转让

输入：to,bin,alltime

（5） CancelMusic

作用：用于注销音乐版权

输入：bin,alltime

（6） MusicRegister

作用：用于响应登记版权

输入：bin，musicName,alltime

（7） ShowNotice

作用：显示所有发给我的Notice，即我是接收者

输入：无

输出：List<Notice> Notice列表

（8） AuthorizeMusic

作用：用于点击确定授权

输入：bin,alltime,to,music,info,NoticeNumber

* 1. 具体实现

因为后台代码量较大，就不在课程报告中展示，详情可以上网查看。

后台代码仓库：<https://github.com/fisco-bcos-group1/back-end>

1. 前端实现

前端主体使用Vue实现，并通过接口于后端进行数据交互。

前端代码仓库：<https://github.com/fisco-bcos-group1/front-end>

1. 软件测试

本软件主要使用了单元测试、功能测试，测试方法主要为Junit和Postman,为软件的可靠性和稳定性提供保证。

1. **区块链未来展望**

从技术角度讲，区块链所涉及到的领域比较繁杂，包括分布式系统、密码学、心理学、经济学、博弈论、控制论、网络协议等，这也意味着工程实践中大量的挑战[7]。

* 1. 隐私保护

隐私保护一直是分布式系统领域十分关键的问题。在分布式场景下，因为缺乏独立的管理机制，参与网络的各方无法保证严格遵守协议，甚至会故意试图获取网络中他人的数据，这些行为都很难进行约束。

而分布式账本要在共享协同信息和隐私保护之间达到合适的平衡，是个不小的挑战。特别随着公有账本系统屡屡出现安全漏洞，动辄造成数千万美金的风险；随着欧盟《通用数据保护条例》（General Data Protection Regulation，GDPR）的落地，隐私保护的合规要求愈加严格；传统的信息安全技术、形式化验证技术在应对新的需求时暴露出实践性不强的缺陷，都亟待解决。尤其以医疗健康领域，对数据的隐私性需求最为强烈，要求严格控制数据的来源、所有权和使用范围，传统手段很难满足这些特性，需要有机结合零知识证明、同态加密等新的密码学手段。而这些新技术在实际应用中还存在不少问题。

* 1. 分布式共识

共识是分布式系统领域经典的技术难题，学术界和业界都已有大量的研究成果（包括 Paxos、拜占庭系列算法等）。问题的核心在于确保某个变更在分布式网络中得到一致的执行结果，是被参与多方都承认的，同时这个信息是不可推翻的。

共识问题在很长一段时间内都将是极具学术价值的研究热点，核心的指标将包括容错的节点比例、决策收敛速度、出错后的恢复、动态特性等。PoW 等基于概率的系列算法理论上允许少于一半的不合作节点，PBFT 等确定性算法理论上则允许不超过 1/3 的不合作节点。

* 1. 交易性能

一般情况下，区块链并不适用于高频交易的场景，但由于金融系统的需求，业界目前也十分关心如何尽量提高区块链系统的交易性能，包括吞吐量和确认延迟两个方面。目前，公开的比特币区块链只能支持平均每秒约 7 笔的吞吐量，安全的交易确认时间为一个小时左右。以太坊区块链的吞吐量略高一些能到几十笔每秒，但交易性能也被认为是较大的瓶颈。2017 年底一款名为 CryptoKitties 的游戏应用造成以太坊网络的严重堵塞。

客观地说，目前开源区块链系统已经可以满足不少应用场景的性能需求，但离大规模交易系统每秒稳定数万笔的吞吐性能还有较大差距。

* 1. 扩展性

对于区块链网络系统来说，跟传统分布式系统不同，这个问题往往并非那么简单。实际上，大部分区块链系统的性能，很大程度上取决于单个节点的处理能力。对这些系统来说，节点需要满足 高性能、安全、稳定、硬件辅助加解密能力。

1. **个人感想**

在进行本次企业实训之前，我对区块链一无所知，只是在各种新闻中听到它的名字，微众银行的老师从最基础的部分开始讲起，循序渐进，层层深入，带我们领略了区块链的无穷魅力。在完成实训项目的过程中，对Spring-boot框架有了深入的了解，对后台开发的流程和技术更加熟悉，再一次加强了对区块链技术的理解，也很大程度上提高了自己的编码能力，收获颇丰。

# 参考文献

[1] Popper, Nathan. A Venture Fund With Plenty of Virtual Capital, but No Capitalist. The New York Times. 21 May 2016

[2] Brito, Jerry; Castillo, Andrea. Bitcoin: A Primer for Policymakers. Fairfax, VA: Mercatus Center, George Mason University. 2013

[3] 左晓静,徐晨莉,王荣.区块链原理及核心技术分析[J].计算机产品与流通,2019,0(3):85-85.

[4] 柏乐飞.区块链的核心技术分析[J].现代商贸工业,2019,40(17):84-85.

[5] Distributed Ledger Technology: beyond block chain. UK Government, Office for Science. January 2016. Retrieved 29 August 2016.

[6] Morris, David Z. "Bitcoin is not just digital currency. It's Napster for finance". Fortune. Retrieved 7 November 2018.

[7] 陈毅洋.区块链技术及其应用挑战研究[J].中国管理信息化,2019,22(9):161-163.