**课程报告**

|  |  |
| --- | --- |
| **课程名称：** | **企业软件项目实训** |
| **学生姓名：** | **梁健恒** |
| **学生学号：** | **201630676805** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2018-2019第二学期** |

**软件学院**

**2019年6月**

**目录**

**[一、](#_Toc13319977)****[区块链技术原理](#_Toc13319977)** [1](#_Toc13319977)

**[二、](#_Toc13319981)****[联盟链和公有链的异同](#_Toc13319981)** [2](#_Toc13319981)

**[三、](#_Toc13319982)****[信任链的建立](#_Toc13319982)** [4](#_Toc13319982)

**[四、](#_Toc13319983)****[Gas与智能合约](#_Toc13319983)** [5](#_Toc13319983)

**[五、](#_Toc13319986)****[分布式存储与区块链](#_Toc13319986)** [6](#_Toc13319986)

**[六、](#_Toc13319987)****[项目成果——MusicChain](#_Toc13319987)** [6](#_Toc13319987)

[1. 需求分析 7](#_Toc13319988)

[2. 用户角色分析 8](#_Toc13319989)

[3. 用户 1](#_Toc13319990)0

[4. 功能性需求 1](#_Toc13319991)1

[5. 非功能需求 1](#_Toc13319992)2

[6. 逻辑部署方案 1](#_Toc13319993)3

[7. 区块链网络部署 1](#_Toc13319994)3

[8. 智能合约编写 1](#_Toc13319995)5

[9. 后台实现 1](#_Toc13319996)8

[10. 前端实现 2](#_Toc13319997)5

[11. 软件测试 2](#_Toc13319998)5

**[七、](#_Toc13319999)****[区块链目前的困难](#_Toc13319999)** [2](#_Toc13319999)5

**[八、](#_Toc13320003)****[个人感想](#_Toc13320003)** [2](#_Toc13320003)6

[参考文献 2](#_Toc13320004)7

**一、区块链技术原理**

区块链在本质上是一个分布式的公共账本，任何人都可以对该帐本进行核查，但并不存在单一的用户可以对它进行控制，而是区块链系统中的参与者共同维持账本的更新。

区块链借助其分布式的特点逐渐提高其在商业上的价值和地位。分布式的特点允许交易在发生时同时引入多家机构的验证和确认，省去了事后同步的操作，保证了交易记录的一致性和真实性。而联盟链在区块链的基础上引入了准入机制、监管节点、身份认证、去代币等安全保障，进一步保证了各方在进行金融交易等操作时，信息能够得到保障，而FISCO-BCOS就是一个开源的联盟链底层平台。  
 在学习中我了解到区块链的几个核心特性，首先共识协作使交易在进行时得到协同计算和群体验证，在多方的监督和验证下，交易变得更加准确、可信。其次区块链由于密码学的应用，确保了信息在传输过程中难以被破解，也使交易内容难以被篡改，进一步保证了其安全性。然后它的区块数据保证了数据的一致性，增加了篡改记录的难度性。最后分布网络的特点使得整个结构可以高效地使用各方的资源。

区块链的核心技术包括分布式账本、非对称加密、共识机制和智能合约等。分布式账本意味着交易记录不像传统的存储方式，将交易数据存储在单一的节点或数据库当中，而是由各个节点共同维护数据。因为每个节点都拥有完整的交易记录和账目，因此它们可以共同监督交易的合法性，也可以同时为其作证。[[[1]](#footnote-0)]除此之外，区块链的分布式特点还体现在它每个节点存储数据的方式：每个节点按照块链式结构存储数据，而不是将数据分成多份进行存储。每个节点在区块链中的地位都是相同的，并依靠共识机制保证存储的一致性。在这种条件下，不会存在单一节点单独记录账本数据，避免了单一节点中的账目被修改的风险。加上足够数量的节点可以保证即使在部分节点被破坏的情况下，账目也不会消失，从而保证了账目数据的安全性。

非对称加密的特点是虽然存储区块链上的交易信息是公开的，但账户信息等信息高度加密，只有在被数据持有者授权后才能访问，从而保障了数据的安全和个人隐私。[[[2]](#footnote-1)]

共识机制解决的是所有记账节点之间如何去认证一个记录的有效性的问题，这既是认证的手段，也是防止篡改的方法。区块链的共识机制结合了“少数服从多数”和“人人平等”的特点，区块链的“少数服从多数”结合了节点数量、计算能力等特征量，而“人人平等”是指当节点满足条件时，所有节点都有权优先提出共识结果，然后交予其他节点进行判断并最后有机会成为最终的结果的特点。

智能合约是区块链应用的基础，它可以自动化地执行一些预先定义好的规则和条约。合约的参与方都需要遵循智能合约上的条约，因为已经预先定义好要遵循的规则，所以智能合约大大提高了交易的效率，也进一步保证了交易的准确性和不可篡改性。

1. **联盟链和公有链的异同**

公有链的一大特点是各个节点可以自由地加入和退出，并参加链上数据的读写，而联盟链通常有与之对应的实体机构组织，要经过授权后才能加入或退出。

鉴于公有链极高的自由度，其网络中不存在任何中心化的服务端节点。公有链的访问门槛很低，因为不需要通过机构的授权，所以只要拥有一台能联网的计算机，用户就能加入到公有链中。每个用户在公有链上的行为都是透明可以查看的，想要篡改在公有链上的数据是难以实现的。同时个节点之间存在匿名性的特点，无需通过彼此信任就可以进行操作，很大程度上保护了用户的隐私。

联盟链与公有链的不同之处在于联盟链是有多个机构共同参与管理和维护的区块链，每个机构拥有一个或多个节点，更适合组织机构之间的交易和结算。联盟链只针对某个群体的成员和有限的第三方，内部指定多个预选的节点为记账人，每个块的生成由所有的预选节点共同决定，其他接入点可以参与交易，但不参与记账的过程，其他第三方可以通过该区块链开放的API进行限定的查询。为了获得更好的性能，联盟链对于共识或验证节点的配置和网络环境有一定要求。准入门槛的提高使交易性能更容易提高，避免了由于参与者性能的参差而产生的问题。[[[3]](#footnote-2)]

相较于公有链，联盟链在效率和灵活性上更有优势，首先是在联盟链上的交易只需要被几个受信的高算力节点验证就可以了，避免了全网验证的额外消耗。然后节点的连接性好，故障可以通过人工干预来修复，并允许使用共识算法减少区块时间，从而加快交易的速度。最后在某些特殊情况下，共同管理联盟链的机构可以根据需求修改该链的规则。

公有链和联盟链的不同主要体现在它们的去中心化程度上，公有链的去中心化程度最高，其不受第三方机构的控制，允许所有用户读取链上的记录、参与交易和竞争新区块的记账权。而联盟链是介于公有链和私有链之间的区块链，可以实现“部分去中心化”，具有成本较低和效率较高的特点，适用于不同实体之间的交易、结算等B2B交易。

1. **信任链的建立**

区块链的密码学基础是建立在哈希算法、加解密算法、消息认证与数字签名、数字证书、PKI体系和Merkle树结构上的。

哈希算法应用在明文的传输中，通过给定的明文，哈希算法可以在有限的时间和资源的情况下将不同的明文编译成唯一的哈希码。哈希码其中一个特点是要从哈希码逆推出明文是极其困难的，这说明哈希算法在很大程度上保证了明文的安全性。

加解密算法主要分为对称加密算法和非对称加密算法两种，非对称加密的特点是对数据的加密和解密采用了两个不同的密钥，分别是公钥和私钥。在区块链的应用中，用户的公私钥生成都是由非对称加密算法实现的。非对称加密增强了点对点式交易的安全性，相比与对称加密算法中交易一方私钥泄露而带来的风险，非对称加密允许双方在不需要预先同步密钥的情况下实现点对点的交易。

PKI体系解决了证书生命是周期相关的认证和管理问题，它建立在公私钥基础上，实现了安全可靠的信息传递和身份确认。

Merkle树是一种典型的二叉树结构，记录了大量的哈希数据。Merkle树结构的特点允许存储在树上的数据进行快速的比较和定位修改。

消息认证基于哈希算法和对称加密，可以用于对消息完整性的保护。利用提前共享的密钥和哈希算法对消息进行加密处理可以获得HMAC值，而HMAC值持有方可以借此证明自己拥有共享的对称密钥。而数字签名基于非对称加密，既可以用于证实数字内容的完整性，也可以确定消息的来源。值得一提的是数字证书可以证明公钥的归属，并确保内容被篡改的时候会被探测出来。在主流操作系统和浏览器中，往往会存在一些预置的权威机构的证书，被这些机构认证过的中间层机构和后续机构都会被验证为合法，这样从预置的根证书到中间层的证书，再到最底下的实体证书，构成了一条完整的证书信任链，当信任链上任一证书被认定为不可靠时，所有依赖它的后续证书都会失去保障。举个例子：客户端得到服务端返回的证书，通过读取得到服务端证书的发布机构。然后客户端去操作系统查找这个发布机构的的证书，如果不是根证书就继续递归下去，直到拿到根证书。接下来用根证书的公钥去解密验证上一层证书的合法性，再拿上一层证书的公钥去验证更上层证书的合法性，进行递归回溯。最后验证服务器端的证书是可信任的。这样的一整个过程就是信任链的形成。

1. **Gas与智能合约**

智能合约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。[[[4]](#footnote-3)]智能合约允许在没有第三方的情况下进行可信交易，它的特点是提供有别于传统合同方法的安全性以及减少了与合同相关的交易成本。

由于智能合约成本低的特点，所以需要一种衡量智能合约执行消耗的单位，而Gas就是智能合约中计算交易费的单位，当不同的指令被执行时，不同数量的Gas就会被消耗。

Gas的重要性在于他确保了提交给网络的交易支付了适当的费用，并尽可能地保证无意义无价值的工作不会被执行。每笔交易都会有其允许的最大消耗Gas数量，如果合约的执行过程中达到了Gas的限制，那合约就会被停止执行。

我在编写音链项目的智能合约时便遇到了到达Gas限制的情况，由于Solidity是一种简洁高效的编程语言，它对代码的效率有很高的要求。我在初步编写合约时由于自己糟糕的编程习惯，经常遇到Gas不足的问题。经过研究和查询后我发现将合约作为数据存储是一种不好的代码实现，所以取而代之我使用了映射来提高代码的工作效率。

Gas的存在对于区块链有着很大的意义，它保证了区块链的高效运作和资源的合理分配，虽然它增加了智能合约的编写难度，但也有助于程序员改良自己的编码习惯，提高代码的简洁性和效率。

1. **分布式存储与区块链**

分布式存储系统，是将数据[分散存储](https://baike.baidu.com/item/%E5%88%86%E6%95%A3%E5%AD%98%E5%82%A8" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/_blank)在多台独立的设备上。传统的[网络存储](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%AD%98%E5%82%A8" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/_blank)系统采用集中的[存储服务器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/_blank)存放所有数据，存储服务器成为系统性能的瓶颈，也是可靠性和安全性的焦点，不能满足大规模存储应用的需要。分布式网络存储系统采用可扩展的[系统结构](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E7%BB%93%E6%9E%84" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/_blank)，利用多台存储服务器分担存储负荷，利用位置服务器定位存储信息，它不但提高了系统的可靠性、可用性和存取效率，还易于扩展。[[[5]](#footnote-4)]

分布式存储对于区块链最大的意义是使篡改数据的代价变得空前巨大，进而保证了数据的安全性。由于存储数据的节点和互相之间并不认识对方，这使数据的保密性进一步地提高。分布式存储系统中的多台服务器通过网络进行连接。但是我们无法保证网络是一直通畅的，分布式系统需要具有一定的容错性来处理网络故障带来的问题。一个令人满意的情况是，当一个网络因为故障而分解为多个部分的时候，分布式存储系统仍然能够工作。[[[6]](#footnote-5)]

分布式的特点允许交易在发生时同时引入多家机构的验证和确认，省去了事后同步的操作，保证了交易记录的一致性和真实性。而联盟链在区块链的基础上引入了准入机制、监管节点、身份认证、去代币等安全保障，进一步保证了各方在进行金融交易等操作时，信息能够得到保障。

**六、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。

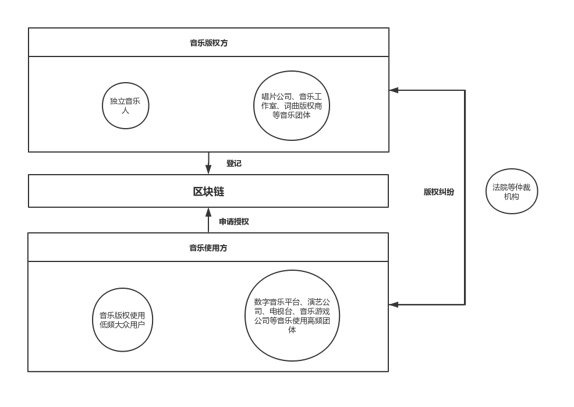


图1 角色联系

* + 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。



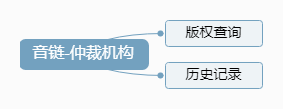


图2 用户具有功能

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

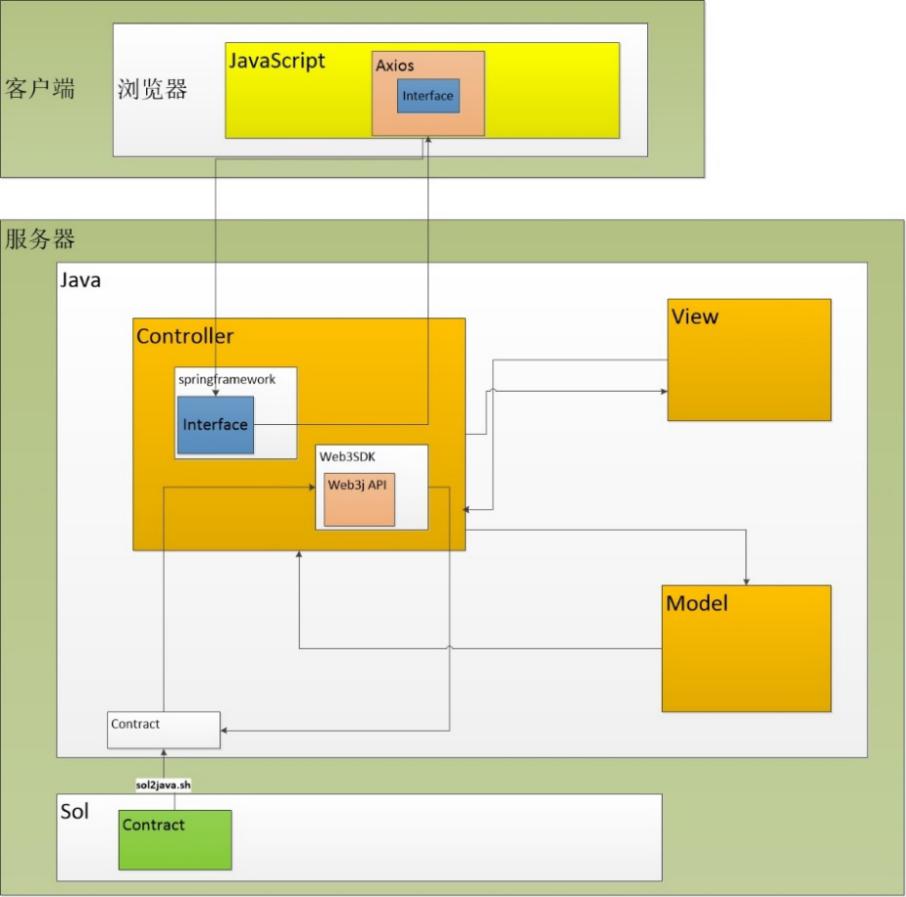
****

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：

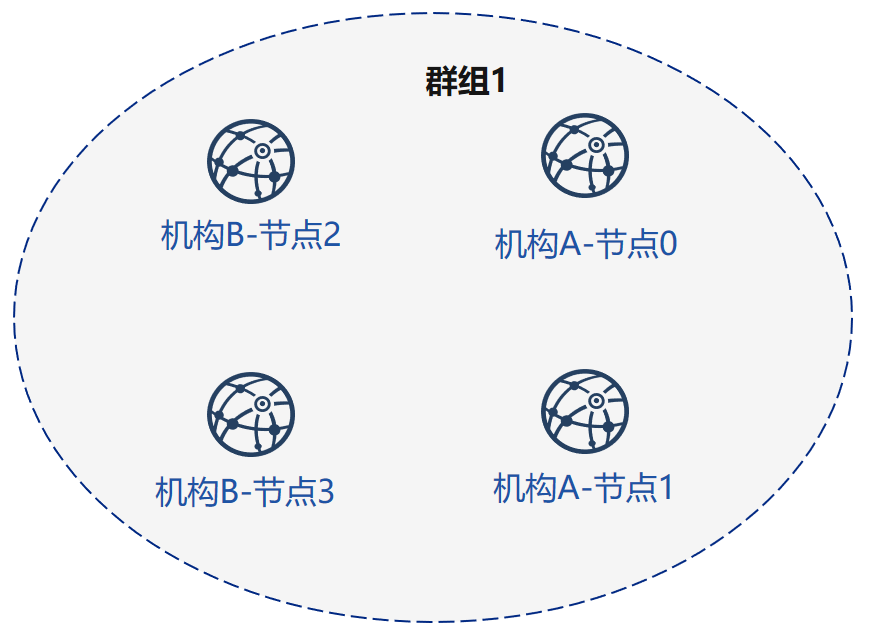


图4 组网模式

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

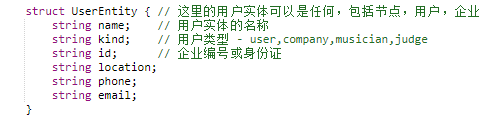


图6 用户参数

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

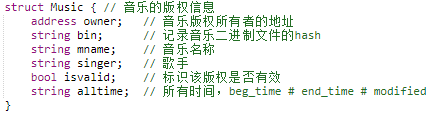


图7 音乐参数

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

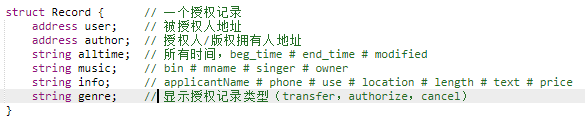


图8 授权记录参数

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

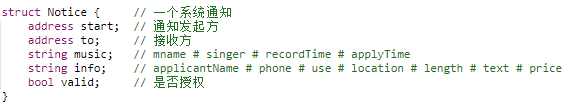


图9 系统通知参数

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

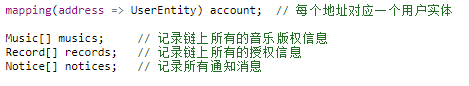


图10 其他参数

* 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

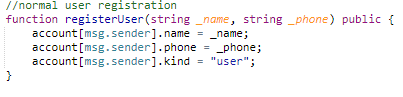


图11 用户注册函数

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. **区块链目前的困难**

虽然目前区块链逐渐成为互联网当中的重要基础设施，但区块链目前还面临着不小的挑战。

鉴于目前的区块链技术发展成熟度，很多的技术方案仍然需要验证，很多的区块链技术受限于交易的实时性和高频交易处理能力的问题。加上屡屡曝出的区块链应用安全问题，使得人们对于目前的区块链应用还是抱有很大的疑问。

另一个问题是区块链的应用场景问题。毫无疑问，区块链具有其他系统无可比拟的优势，但区块链应用的落地并不如它的技术优势般乐观。目前很大部分的区块链应用都聚集在一定的领域中，而另外的一些区块链应用则是让人们质疑其放在区块链上运行的意义所在。加上相关监管条例的缺失，也使人们对区块链技术保持着观望态度而不敢参与到其中。

1. **个人感想**

本次的企业实训让我对区块链有了全新的认识，虽然我之前阅读过有关比特币的书籍，我对区块链的具体实现还是一窍不通。在实训的过程中我也拓展了我对区块链引用的想法：目前很多的版权问题都可以由区块链应用处理。从底层的智能合约编写到数据对接，我通过实操增长了自己的知识和经验，我在这次的企业实训收获匪浅。

**参考文献**

[1]迪瓦特&#;德什潘德, 凯瑟琳&#;斯图尔特, 路易斯&#;列皮特,等. 理解分布式账本技术/区块链——挑战、机遇和未来标准[J]. 信息安全与通信保密, 2017(12):22-31.

[2]朱岩, 甘国华, 邓迪,等. 区块链关键技术中的安全性研究[J]. 信息安全研究, 2016, 2(12):1090-1097

[3]高志豪. 公有链和联盟链的道法术器[J]. 金卡工程, 2017(3):35-39.

[4]Thomas Bocek. Digital Marketplaces Unleashed. Springer-Verlag GmbH. 2017-09-15: 169-184. ISBN 978-3-662-49274-1.

[5][分布式存储系统的实现.](https://baike.baidu.com/reference/6608875/cb64bVp96ISttDSxt4w_Ss54wHur8hpMiEbhZeK7dM6o5oRRLHSZbRJ3gVrVeT7Utp_bVk41OmuehjsiHBrey3A4kls2B7CLGuwo" \t "https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F/_blank)比特网．2010-11-29[引用日期2014-04-13]

[6]陈敏，张东，张引，亓开元编著,大数据浪潮 大数据整体解决方案及关键技术探索=BIG DATA INSPIRATION,华中科技大学出版社,2015.10,29-30

1. [↑](#footnote-ref-0)
2. [↑](#footnote-ref-1)
3. [↑](#footnote-ref-2)
4. [↑](#footnote-ref-3)
5. [↑](#footnote-ref-4)
6. [↑](#footnote-ref-5)