华南理工大学

课程报告

课程名称：	企业软件项目实训
学生姓名：	张光云
学生学号：	**201630666318**
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、实训概述

此次实训是一次关于区块链的实训。在实训中，讲授了区块链的基础知识，深入讲解了以太坊和联盟链的知识，最后使用微众银行推出的 fisco-bcos 完成一个关于联盟链相关的区块链项目。

此次实训是由华南理工大学软件学院与微众银行共同举办，课程中由许可老师负责校内和平时学生人员的管理工作，每周或每两周由微众银行老师进行一天的授课，最终共同验收学生完成的项目。

在我们的项目小组中，我们完成了一个关于音乐版权管理和版权授权平台的项目（首页如图 1；），该项目通过音乐人上传音乐来获取该音乐的版权，并进行版权的管理，而其他用户则可以进行音乐的使用权的申请，从而获得使用音乐版权的资格。

图 1

在该项目中，我的角色的产品经理，这和我本身的职业规划路线相吻合，因此能够在需求调研和产品设计上发挥更多的力量。我的工作首先是寻找各个方向，提出了药品追溯、音乐版权追溯、学生信息追溯、学生论文追溯等方向，最后在与组员的讨论之下，确定了音乐版权追溯的方向。接着进行市场调研、产品分析、产业链调研等等工作，分析产品的可实现性的高低。针对现有的问题，提出了相应的解决功能，由此进行产品的设计，列出产品的功能点后进行产品原型的设计，与组内成员进行协商妥协，然后完成了整个项目工作。

二、区块链技术原理

讲解区块链，首先就要讲解什么是区块链以及区块链的历史。[1]

区块链是分布式数据存储、点对点传输机制、共识机制、加密算法等计算机技术的新型应用模式，是一种只能不断增长的记录列表、使用加密技术进行链接和保护的分布式账本。

从区块链技术的历史来看，区块链的产生不是一蹴而就，也不是单纯由一个人提出来的。区块链的思想最开始诞生在网络中的数字货币，密码学家 David Chaum 提出了 e-Cash 的概念，并创建了首个匿名化的数字加密货币，但该数字货币依旧是依靠一个中心化的机构，因此最后还是失败了。随后， HashCahs 技术、POW 机制、时间戳协议等等技术的发明与提出为比特币的出现打下了基础。2008 年，一个名叫中本聪的人提出了一个完全去中心化的电子现金系统。随后，比特币作为一种数字货币逐渐开始大火，而背后的区块链技术也逐渐被人们发现并提炼出来，越来越多的人参与到区块链中去。

从数据结构上来看区块链技术，该技术是一块一块的，不同的块由既定的技术链接在一起，在块中如数据包一样，除了本身包含的数据，就是与块相关的数据信息。（如图 2；）

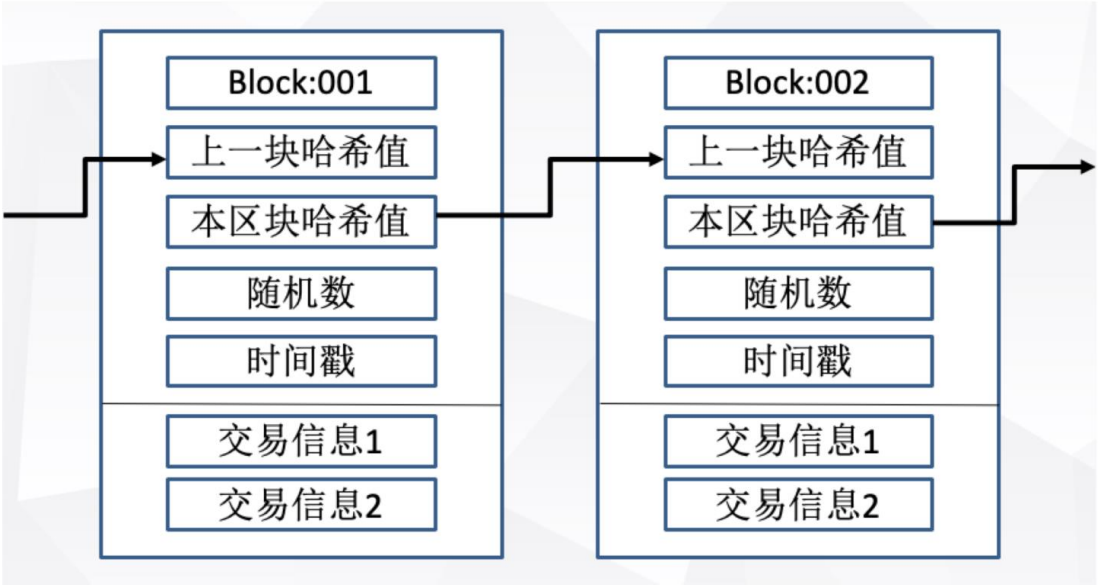


图 2

区块链的特征有去中心化、防篡改、匿名性、开放性、自治性等特点。去中心化总的来说，由于使用分布式核算和存储，体系不存在中心化的硬件或管理机构，任意节点的权利和义务都是均等的，系统中的数据块由整个系统中具有维护功能的节点来共同维护。防篡改，一旦信息经过验证并添加至区块链，就会永久存储起来，除非能够同时控制住系统中超过 51%的节点，否则单个节点中对数据库的修改是无效的，因此区块链的数据稳定性和可靠性极高。匿名性指的是，由于节点之间的交换遵循固定的算法，数据交互是无需信任的，因此交易对手无需通过公开的身份让对方对自己产生信任，对信用的积累非常由帮助。开放性则指的是，整个系统是开放的，除了交易各方的私有信息被加密外，区块链的数据对

所有人公开，任何人都可以通过公开的接口查询区块链数据和开发相关应用，因此整个系统信息高度透明。自治性，区块链采用给予协商一致的规范和协议使得整个系统中所有阶段能够在去信任的环境自由安全地交换数据，使得对人的信息改成了对机器的信任，任何人为的干预都不起作用。

在区块链中几种核心的基础技术有 hash 算法、加密解密算法、消息认证码与数字签名、数字证书、Merkel 树结构等。

Hash 算法能够将任意长度的二进制明文串映射为短的二进制串，好的 hash 算法能够做到正向快速计算，逆向基本不可能在有限的时间内推出明文，输入敏感，两个不同明文之间产生的 hash 值的冲突是可以避免的。

加密算法分为两类，分别是对称加密和非对称加密。对称加密的加密解密的密钥相同，计算效率高；非对称加密算法加密解密的密钥不同，但是无需提前共享密钥。非对称加密对于数字签名和密钥协商的常见具有较好的优势。两种模式混合使用可形成混合加密机制。

消息认证码是基于对称加密算法、可以用户对消息完整性进行保护的算法；数字签名是基于非对称加密算法，既可以用户于证实某数字内容的完整性，又可以确认来源。

数字证书机制是可以证明某个公钥是某个实体的，并且确保一旦内容被篡改就能被探测出来，从而实现对用户公钥的安全分发。

Merkel 树结构又叫哈希树（如图 3；），树最下面的树叶节点包含存储数据或其 hasd 值，非叶子节点都是它的两个孩子节点内容的 hash 值。推广到多叉树时，非叶子节点的内容为它所有孩子节点的内容的 hash 值，逐层记录 hash 值，当底层数据有任何改变时，都会传递到父节点，从而发现数据的改变。

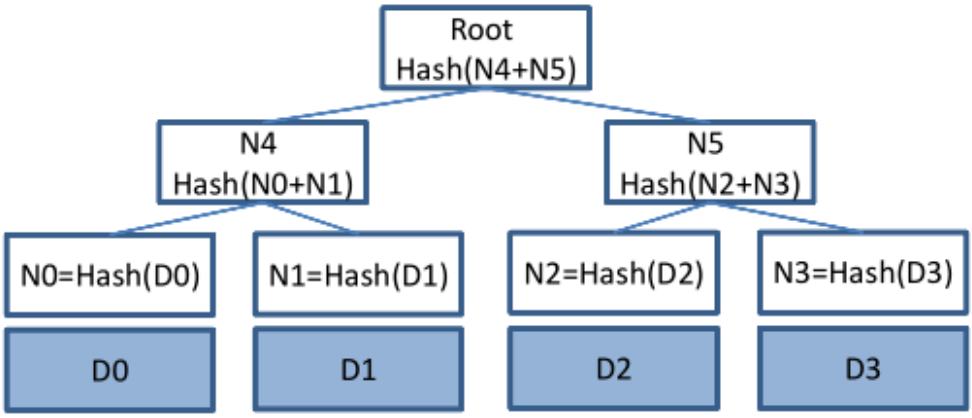


图 3

三、联盟链原理以及与公链的区别

在本次实训中，我们学习的是金链盟的联盟链 FISCO-BCOS。该联盟是由深圳市金融科技协会、微众银行、深圳通等二十余家金融机构和科技企业共同发起的非营利组织。

FISCO-BCOS 是针对于分布式商业提出的区块链技术的延展，能够在组织的的情况下实施多方参与，需要多方进行共同协作，在信息纷杂散乱、缺乏公信力的情况下，通过联盟链，进行数据校验、多方验证、无法篡改，最终实现身份确认、资产确权、司法认可等目标。

下图是联盟链的总体架构。在 FISCO-BCOS 2.0 的开源代码中，为了更加契合分布式商业，提出了群组架构、并行计算、分布式存储等方面的新特性。

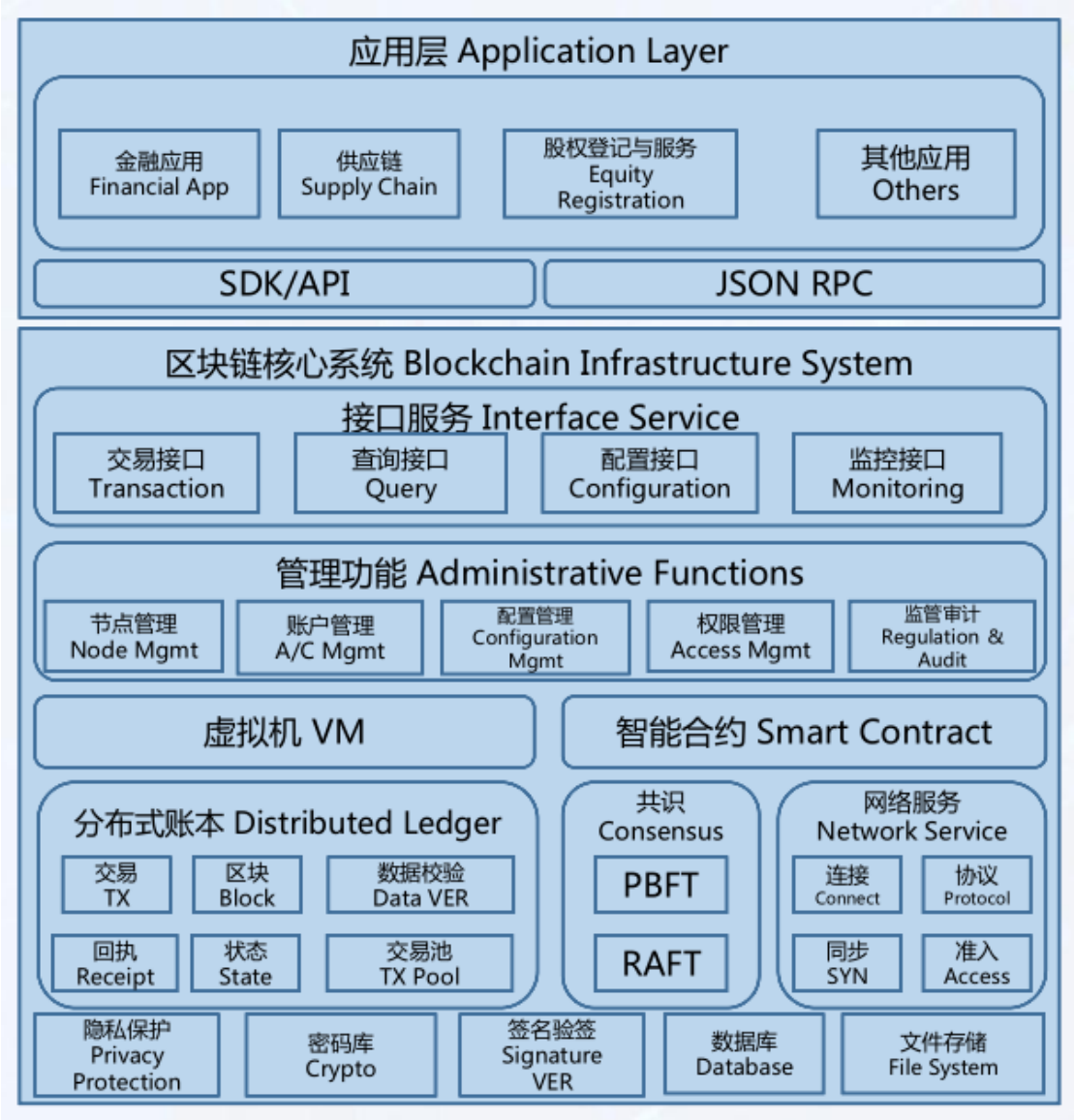


图 4

多群组架构也就是说，在多个节点组成的区块链中，部分节点通过配置，能够形成独立的账本，而不与其他节点互通，账本内的节点进行独立的共识，存储独立的状态，只需要修改配置就能够建立账本，无需额外的运营资源。

并行计算模型能够自动识别交易互斥的资源，构建出 DAG，规划一个可并行的执行路径，在最佳的情况下，能够完全发挥 CPU 的性能优势。

分布式存储突破了单机磁盘容量的限制，支持横向扩展，分布式存储定义了标准的数据访问 CRUD 接口，适配多种数据存储系统，区块链节点既可以直接访问存储，也可以通过专用的数据代理访问存储。

总的来说，联盟链使用了节点准入机制，参与区块链网络的节点都需要经过

准入控制，采用了 SSL 通信机制，基于 PKI 的 CA 认证机制；也使用了证书的机制，通过证书，能够识别传输的数据是否被修改，同时也能够通过证书验证传输信息人的身份，在 FISCO-BCOS 中使用了根证书、机构证书、节点证书三级证书机制；在合约上，使用的是以太坊的 solidity 语言进行开发，solidity 是图灵完备的高级语言，支持循环、函数调用等用法，具有大量普通高级语言的特性，能够很好地支持语言开发。

从技术特点来看，公链具有去中心化、防篡改、匿名性、开放性、自治性等特点，而联盟链则是针对于分布式商业产生的。[2]

公有链是完全去中心化的，链上可以加入任何验证机，每台验证机拥有同样的权利，参与全网的交易验证以及收益分配，主要特点是人人平等。它的优势是公开、透明、不可篡改，因为高度去中心化，所以试图篡改交易数据几乎不可能。但是，它具有低 TPS 的缺陷，交易的速度缓慢，吞吐量低。

而联盟链则是有限的去中心化，它的验证机受到某一个机构的限制，节点只有通过授权才能加入，一旦加入，节点拥有相同的权益。联盟链具有低成本允许和维护，交易只需要被几个受信的高算力节点验证就可以了，而无需全网验证，同时它也具有高交易速度、良好的延展性，更加灵活。

图 5

四、智能合约

智能合约是在区块链上达成协议的一种方式，智能合约并没有超越传统合约的功能，但它能最小化达成协议所需的信任。智能合约不仅是一个可以自动执行的程序，它自己就是一个区块链参与者。它对接收到的交易进行回应，它可以接收和储存价值，也可以向外发送信息和价值。[3]

与比特币脚本相比，以太坊智能合约最大的不同点是：它可以支持图灵完备

的开发语言，其允许开发者在智能合约内执行任意功能，对资产进行精确状态，访问整个区块链的状态。

图 6

本质上，智能合约的交易是从一个账户发往另一个账户的消息如果交易发送的目的地是外部账户，那么这个交易的目的就是转账，如果交易发送的目的地是合约账户，那么这个交易的目的就是调用智能合约。收到这类交易后，底层会取出目的地合约账户的代码（code），以交易的数据（data）做为智能合约调用的参数，初始化一个 EVM 并执行，如果交易发送的目的地是空地址，那么这个交易的目的就是创建智能合约。收到这类交易后，底层会根据发送者的地址和发送者的 nonce，计算哈希值，作为一个新的合约账户的地址。同时，EVM 会执行交易的数据（data），将执行的结果，做为合约代码保存到新合约账户的代码中。

Solidity 是运行在 EVM 上，面向合约的高级编程语言，它的语法受 C++、Python 和 JavaScript 的影响 Solidity 使用静态类型，支持继承、库和用户自定义类型等特性。

任何事物都不可能是完美的。就像其他的系统协议一样。智能合约也不是完美的。智能合约的优点和缺点，包含了使用效率和监管机制。具体表现在：

在处理文档时效率会更高是智能合约中的优点。它不需要人参与，只要满足合约代码要求即可，实现了全自动化流程。优点是节省了时间、降低了成本、让交易更准确，且无法被篡改。智能合约不受任何第三方机构的干扰，这进一步实现了去中心化。人为的错误、无法完全实施、法律状态的不确定性，这些都会很大程度地影响到智能合约，缺乏有效的监管机制，人为的错误以及完全实施难度很大。

五、音链项目介绍

随着区块链技术应用的推广，该技术在越来越多的领域得到了应用，特别是在企业服务方向的应用，给越来越多的企业和政府部门带来了一定程度效率上的提高，数据不可篡改和去中心化的特性帮助企业之间建立更加完备的信任机制，减少了在谈判和不信任的情况下的花销。

在调研中，我们发现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。

尽管如此，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。

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。

在该项目中，我们的愿景就是希望能够通过区块链技术，为企业、音乐人、其他音乐使用用户群体提供更加便捷且有效的音乐版权管理的方式，一方面是版权归属的管理，另一方面是授权的管理。在版权管理上，我们希望能够匹配当前的管理方式，让版权管理的组织能够加入到我们的区块链中；在企业的版权管理上，为更多的中小型的版权商、版权企业提供更加便捷的管理方式，能够快速地版权自我的版权，并在电商式的平台上，让自己的版权得到良好的推广。

六、需求分析

1. 市场分析[4]

从艾瑞咨询的音乐行业市场报告来看，中国的音乐市场在未来五年间，仍然有庞大的增长空间，而当前音乐市场的总收入达到了近百亿元的规模。

图 7

从 2017 和 2019 年艾瑞咨询整理的中国数字音乐产业链来看，可以将音乐产业粗略划分为生产者、消费者，另外，还有音乐的分发方，版权管理协会。

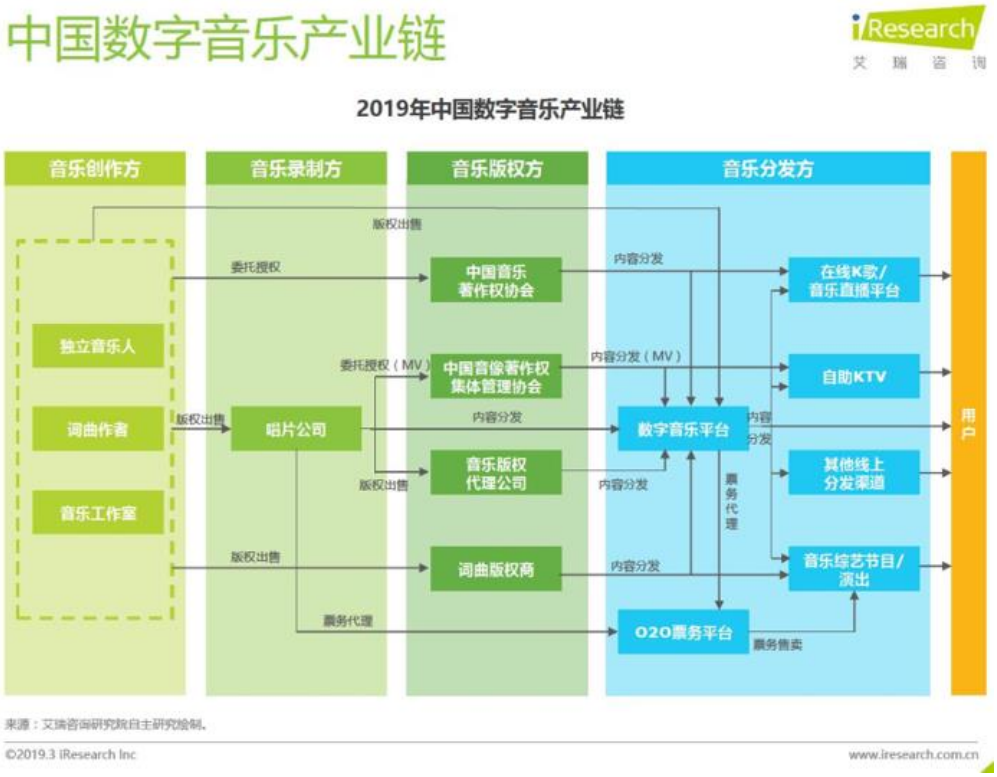


图 8

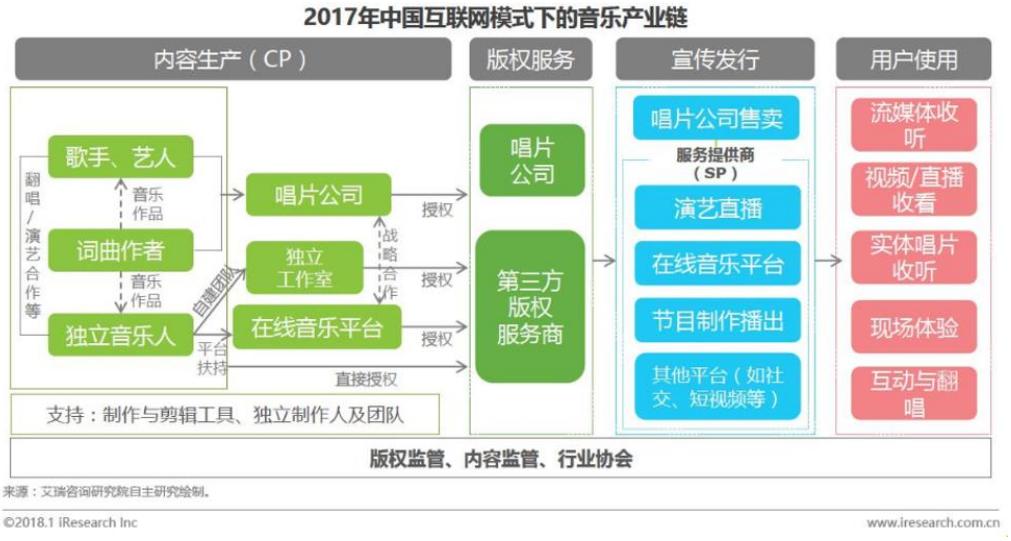


图 9

2. 国内音乐授权平台分析

平台：AGM 平台

业务：数字音乐素材授权、音乐定制、授权查询、音乐人合作

平台：100Audio

业务：数字音乐素材授权、音乐人合作

平台：V.Fine Music 平台

业务：数字音乐素材授权、音乐人合作、音乐定制、授权查询

特点：形式均为音乐商城、音乐库规模达、模式成熟

商业模式：专注于版权音乐授权的电商平台

主要商品：数字音乐素材

特点：授权快捷；使用范围、时间、价格、数量统一，减少双方谈判沟通的成本，避免劣币驱逐良币；提供快速寻找合适的音乐素材，提供定制功能。

从 V.Fine Musci 和 100Audio 两家公司给音乐人的收益来看，40%——70%不等的收益都是由平台来划定的，且不同的平台和不同类型的音乐人之间的收益可能会存在较大的差异。

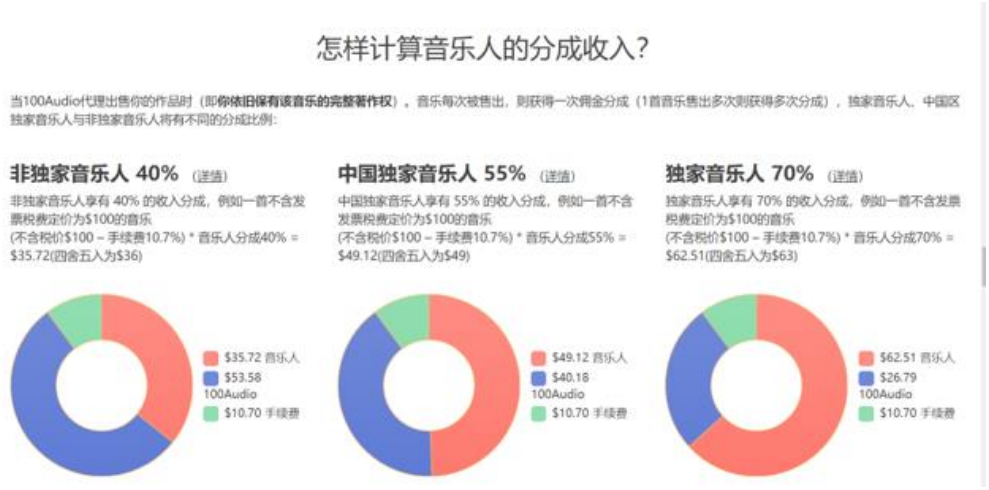


图 10

来源：100Audio 官网

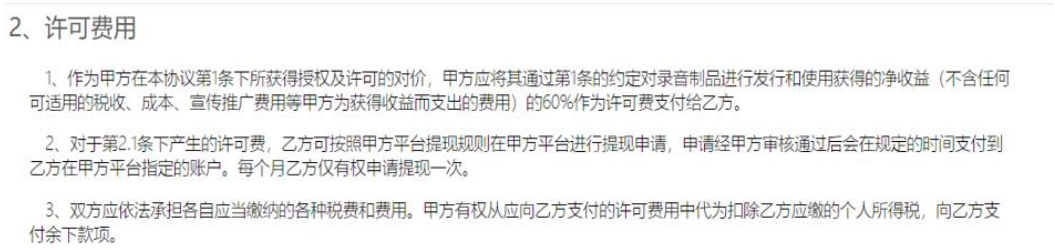


图 11

来源：V.Fine.Music 官网

3. 中国音乐著作权协会分析

中国音乐著作权协会是国内唯一的音乐著作权集体管理组织，针对海量音乐使用者出现的侵权情况，登记并管理音乐作品及其版权资料，提供著作许可服务并收取费用，提供法律支持。

协会中，拥有版权的人可以进行：版权注册、版权交易、作品修改、作品注销等功能。但是从版权注册的流程来看，申请版权注册需要 45+30+邮费接近 100 元的费用，版权人的作品分配授权费用时需要至少两个月的时间才能够到账。

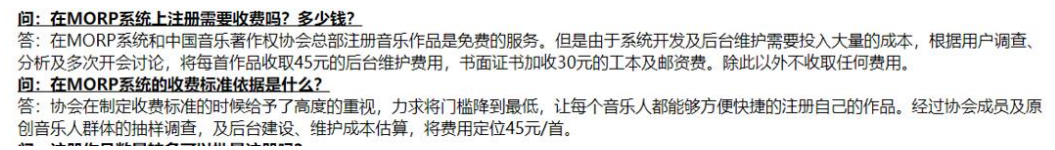


图 12

图 13

来源：中国音乐著作权协会官网

对于音乐版权的使用者来说，协会为这类用户提供音乐的复制权、表演权、广播权、信息网络传播权的申请使用。

但是其中的问题有：需提前明确使用的歌曲及其信息；通过协会报备、审核；办理许可、出具发票；收费标准由协会统一商定。这整个过程都会需要通过音乐协会才能进行，且整个流程的周期较长。

4. 侵权数据统计

《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。

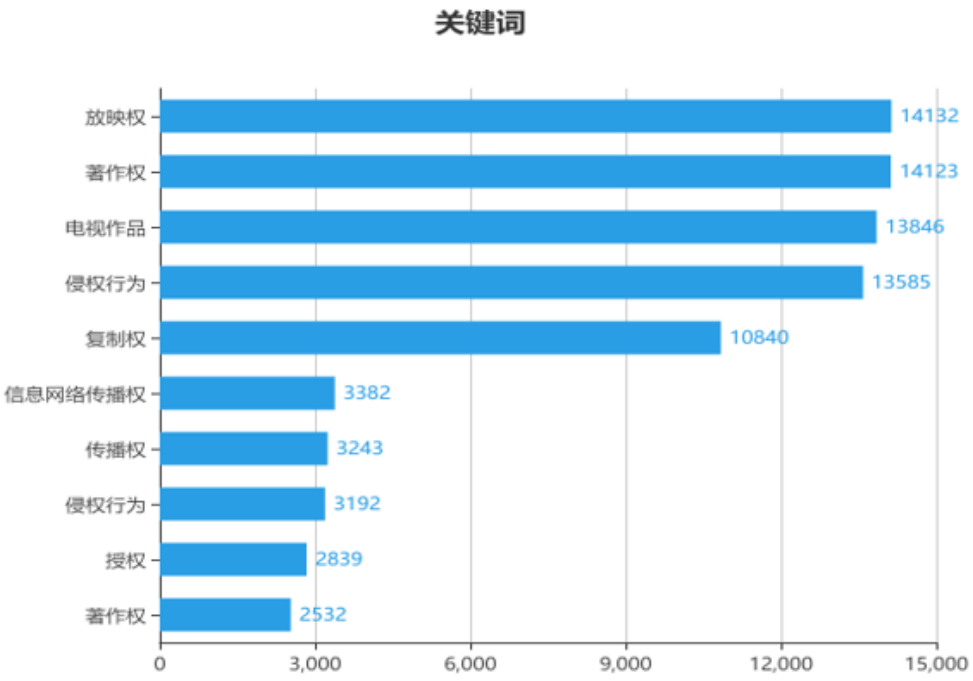


图 14

时间：2015-2019 年
来源：无讼案例
网址：https://www.itslaw.com/bj

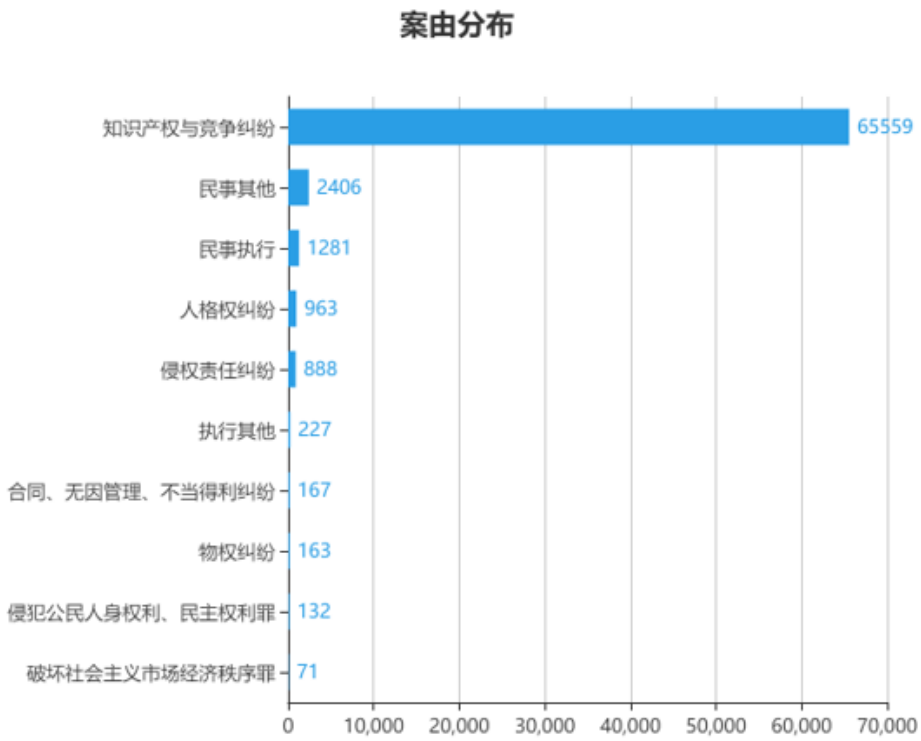


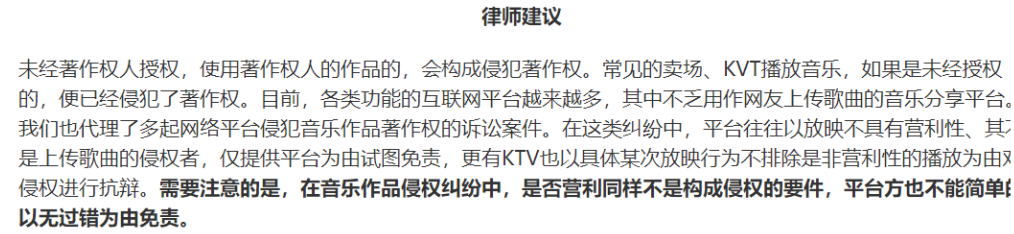
图 15

时间：2015-2019 年
来源：无讼案例
网址：https://www.itslaw.com/bj

律师建议：从律师的建议来看，即便是非营利性质的使用音乐仍然可能造成侵权；在进行音乐授权时，需要对授权人的授权资格进行足够的验证，防止授权过程中出错。[7]

对于其他使用他人音乐作品的行为，使用人还是应当注意事先取得授权，合法使用。由于音乐作品是词、曲的结合，有些还包括MV，因此一首音乐作品的授权涉及词、曲、MV 的多项授权。一般情况下，包括词、曲、MV 在内的音乐作品完整著作权在音乐公司、音著协，向音乐公司、音著协就可以取得完整的授权。当然，由于很多音乐作品的信息网络传播权被网易云音乐、腾讯音乐等移动音乐平台享有，音著协可能仅享有音乐作品的复制、发行、改编等信息网络传播权以外的著作权，因此，**在取得授权时，要留意需取得的授权内容，授权人是否享有该等权利。**

图 16


律师建议

未经著作权人授权，使用著作权人的作品的，会构成侵犯著作权。常见的卖场、KVT播放音乐，如果是未经授权的，便已经侵犯了著作权。目前，各类功能的互联网平台越来越多，其中不乏用作网友上传歌曲的音乐分享平台。我们也代理了多起网络平台侵犯音乐作品著作权的诉讼案件。在这类纠纷中，平台往往以放映不具有营利性、其是上传歌曲的侵权者，仅提供平台为由试图免责，更有KTV也以具体某次放映行为不排除是非营利性的播放为由对侵权进行抗辩。**需要注意的是，在音乐作品侵权纠纷中，是否营利同样不是构成侵权的要件，平台方也不能简单以无过错为由免责。**

图 17

来源：星瀚律师-新浪微博

网址：http://blog.sina.com.cn/s/blog_7813a0670102xc7u.html

5. 音乐人生存现状

数据来源：中国传媒大学音乐与录音艺术学院：2018 年音乐人生存现况与版权认知抓鬼看调研研究报告（节选）[5]

该报告数据主要来源于问卷调查和访谈，共计回收问卷 406 份，调查对象包括词曲作者、唱作人、歌手、编曲制作人、录音师等；采访音乐人 112 位，有效访问 104 位。

下图为音乐人在数字音乐平台上传音乐的频率，由此来看，有 36%的音乐人每年上传音乐的频率在 6 首以上，剩余音乐人的上传音乐的频率并不高。

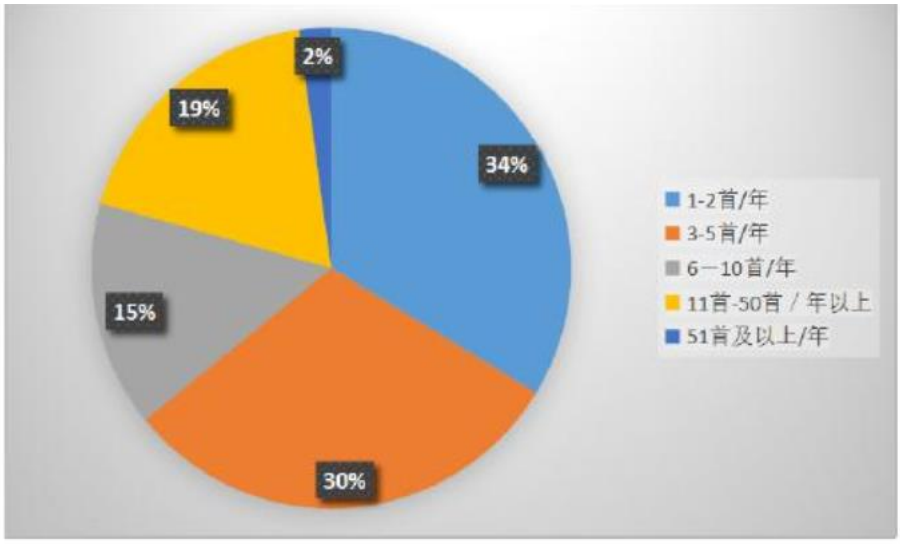


图 4.2.3 音乐人数字音乐平台上传频率

图 18

下图为音乐人税前收入的分布状况，其中有 60.49%的音乐人的收入有 8000 元以上相对较高的收入。但是仍然有近 40%的音乐人处于较低的收入水平，甚至无法通过音乐正常维持自己的生活水平。此外，报告中指出，相对于欧美国家的音乐人的收入水平，我国音乐人的收入水平与他们相比有较大的差距。

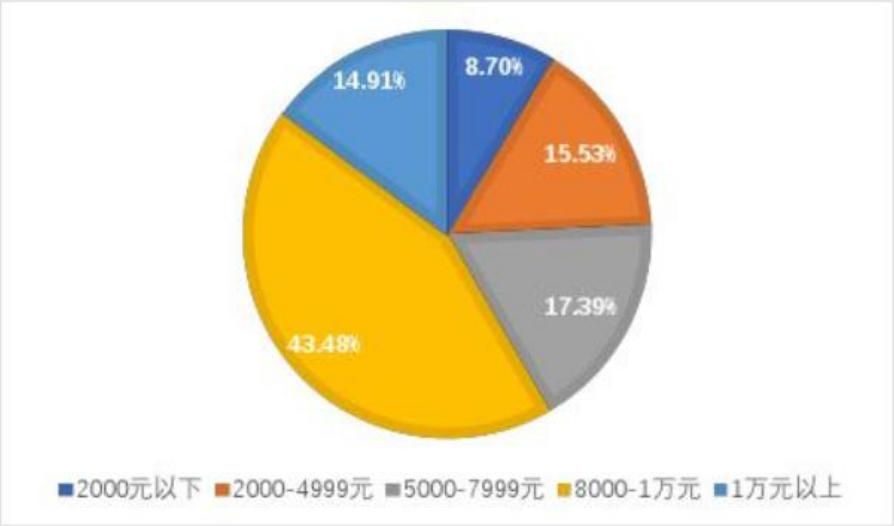


图 3.1.2 税前月收入

图 19

下图为近五年来，中国政府开始整顿音乐版权以后，音乐人的收入增长，相对来看，较大比例的音乐人的收入有了不同程度的增长。但是，这相对与国内的物质增长水平和欧美国家的收入而言，均不够理想。

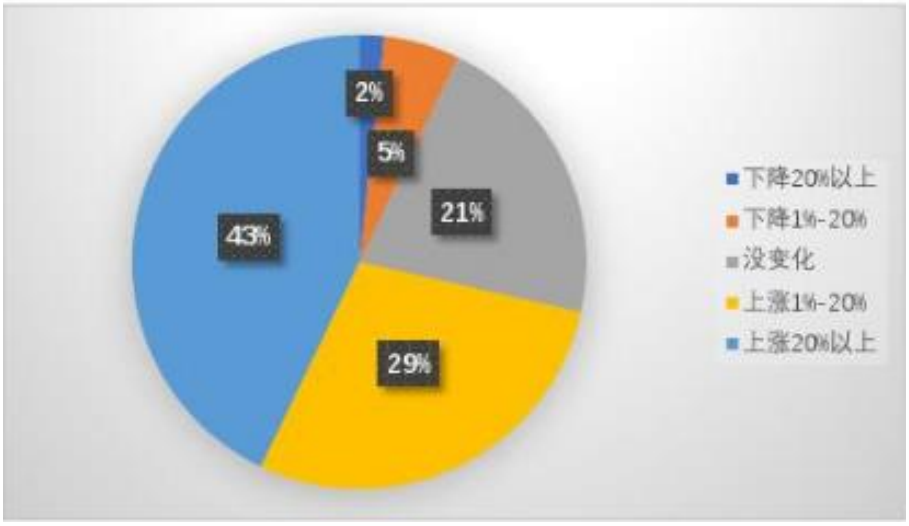


图 20

下图是调查对象的收入来源的比例图，音乐人收入来源是多样的，但是其中，音乐人的版权收入比例仅为 6%，报告中指出，这个数字相对于欧美国家仍然有较大的差距，有很大的增长空间。

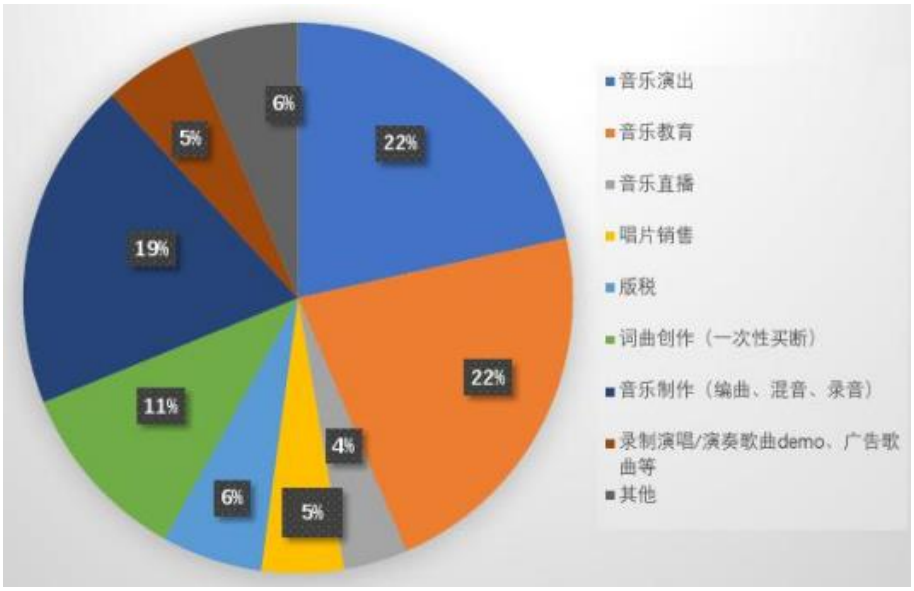


图 21

下图为音乐人的权益受到侵犯时维权途径的选择倾向，维权的形式主要由团体企业或公众来帮忙维权，而通过法律手段亲自维权的比例仅为13%，通过律师维权的比例也仅为 19%，更有 11%的音乐人放弃对侵权行为的维权。

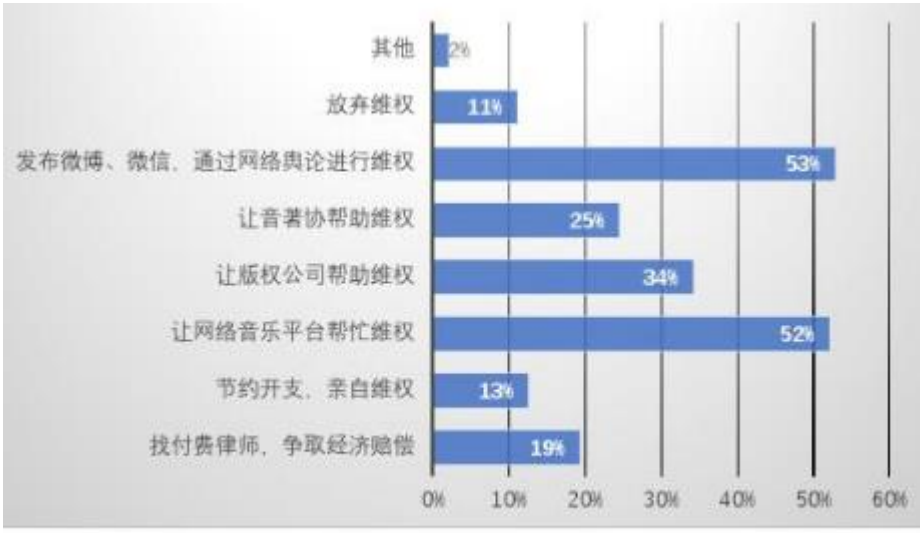


图 5.2.2 音乐人维权途径的选择倾向

图 22

下图为音乐人放弃维权的原因，一方面是音乐人本身缺乏足够的维权意识，主动放弃维权，另一方面是维权对音乐人不够友好，需要音乐人花费较多的时间进行维权。

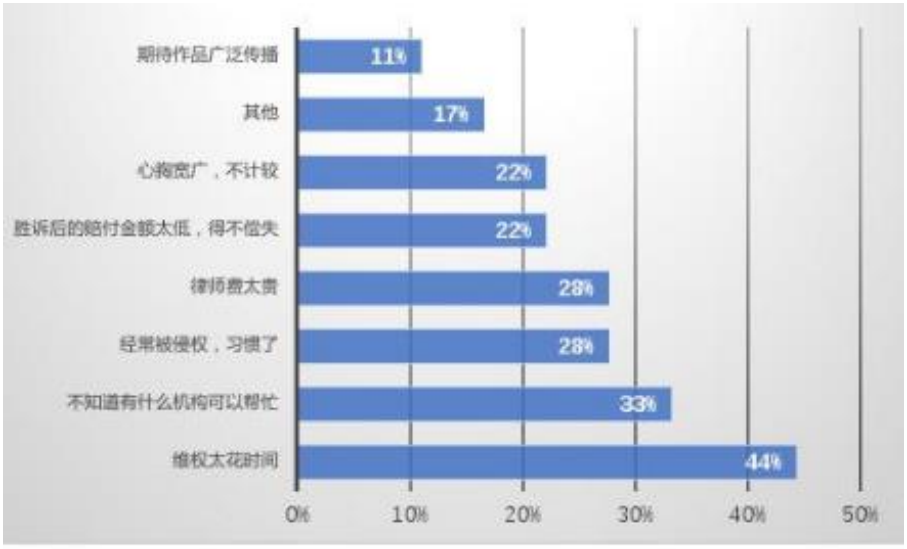


图 5.2.3.1 音乐人放弃维权的原因

图 23

6. 现状总结

音乐授权平台

- 缺少词曲版权登记功能
- 对音乐人收益抽成高
- 价格固定，缺少市场调节

中国音乐著作权协会

- 中心化平台，工作效率难以得到最大的提升
- 音乐使用价格由协会指定，缺少市场调节

侵权行为：

- 尽管国家在 2015 年以来加大对版权的保护力度，但是仍然存在大量的侵权现象

音乐人生存现状：

- 音乐人的收入增长较慢，收入较低，存在较大的增长空间
- 音乐人收入来源中，版权收入占比低
- 音乐人维权难以亲自依靠法律武器进行维权，维权的消耗大

七、产品设计

a) 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。

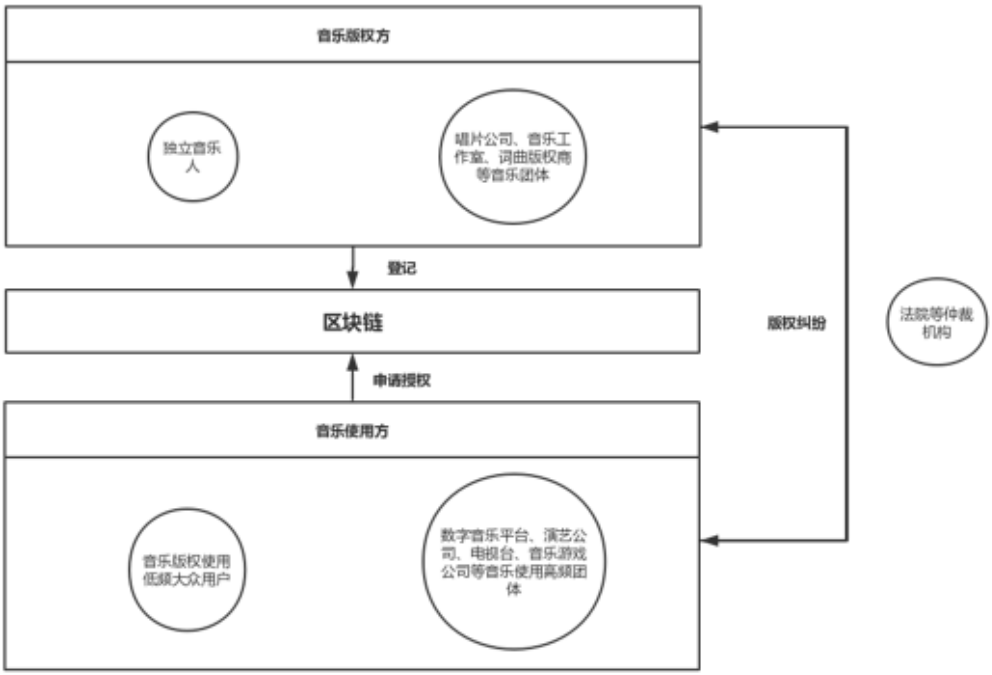


图 24

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。

2. 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。

3. 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。

4. 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。

5. 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。

b) 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；

二、 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 25

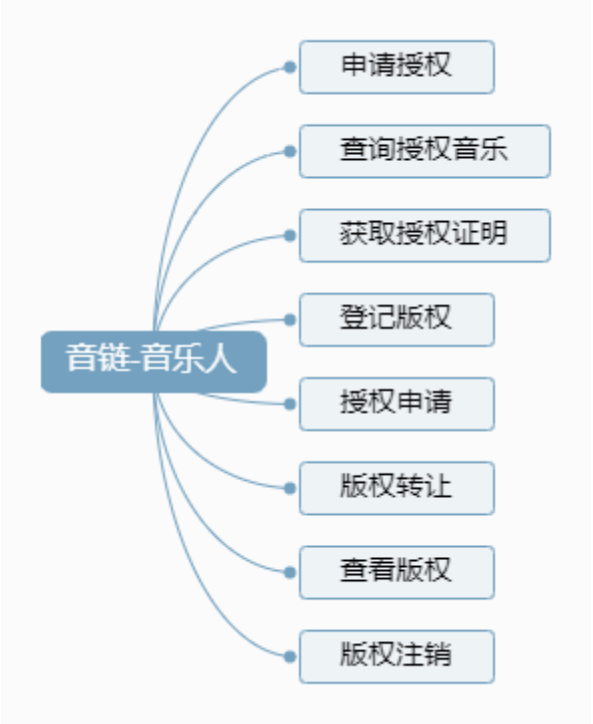

音链-音乐人
申请授权
查询授权音乐
获取授权证明
登记版权
授权申请
版权转让
查看版权
版权注销

图 26


音链-企业
申请授权
查询授权音乐
获取授权证明
登记版权
授权申请
版权转让
查看版权
版权注销
版权共享

图 27

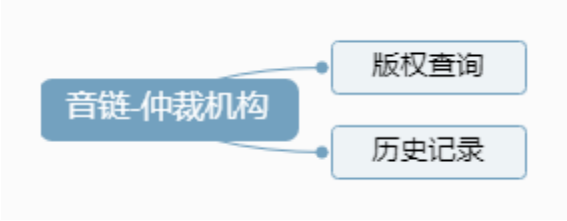


图 28

功能性需求分析：

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。	

非功能性需求：

a) 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；

b) 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；

八、项目成果

原型：
https://org.modao.cc/app/90e8640e1a31563900cfed27bdc1bb13#screen=s229711DEE91561967147540

项目地址为：https://github.com/fisco-bcos-group1/front-end

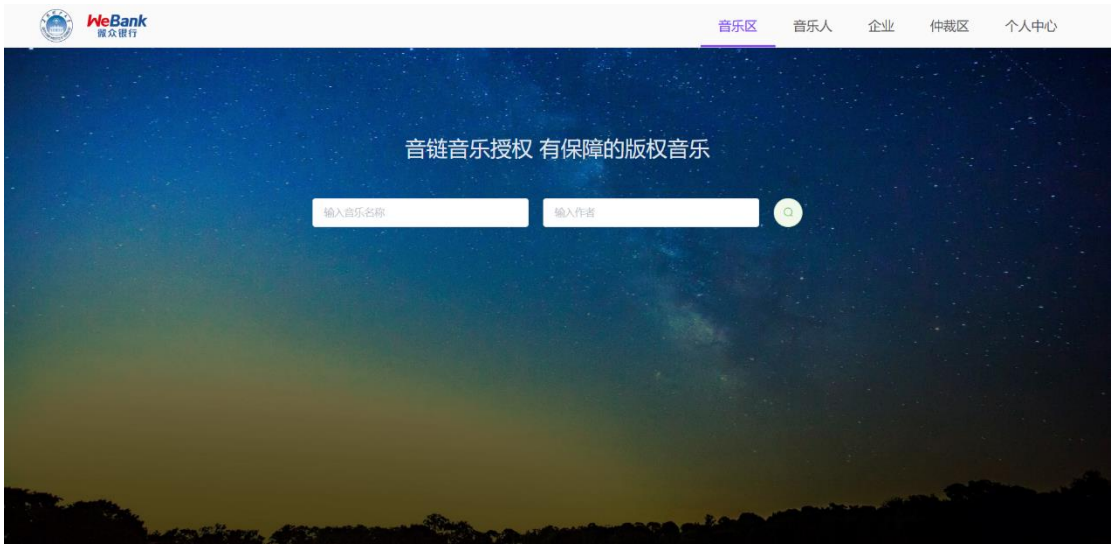

WeBank
微众银行
音乐区
音乐人
企业
仲裁区
个人中心
音链音乐授权 有保障的版权音乐
输入音乐名称
输入作者

图 29


WeBank
微众银行
音乐区
音乐人
企业
仲裁区
个人中心
查询音乐版权信息
输入音乐名称
输入作者
机构登录
机构注册
版权信息
历史记录
机构信息

图 30


WeBank
微众银行
音乐区
音乐人
企业
仲裁区
个人中心
用户信息
授权订单
版权音乐
登记版权
版权转让
授权申请
姓名
123
身份证号码
123123123
所在地
1231231
联系电话
1231231
电子邮箱
12312312

图 31


用户信息
授权订单
版权音乐
登记版权
版权转让
授权申请

图 32

图 33

九、系统设计

1. 基本架构设计

1.1 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

```
struct UserEntity { // 这里的用户实体可以是任何，包括节点，用户，企业
    string name;    // 用户实体的名称
    string kind;    // 用户类型 - user,company,musician,judge
    string id;      // 企业编号或身份证
    string location;
    string phone;
    string email;
}
```

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

```
struct Music { // 音乐的版权信息
    address owner;   // 音乐版权所有者的地址
    string bin;      // 记录音乐二进制文件的hash
    string mname;    // 音乐名称
    string singer;   // 歌手
    bool isvalid;    // 标识该版权是否有效
    string alltime;  // 所有时间，beg_time # end_time # modified
}
```

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

```
struct Record {      // 一个授权记录
    address user;    // 被授权人地址
    address author;  // 授权人/版权拥有人地址
    string alltime;  // 所有时间，beg_time # end_time # modified
    string music;    // bin # mname # singer # owner
    string info;     // applicantName # phone # use # location # length # text # price
    string genre;    // 显示授权记录类型（transfer，authorize，cancel）
}
```

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

```
struct Notice {     // 一个系统通知
    address start;  // 通知发起方
    address to;     // 接收方
    string music;   // mname # singer # recordTime # applyTime
    string info;    // applicantName # phone # use # location # length # text # price
    bool valid;     // 是否授权
}
```

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

```
mapping(address => UserEntity) account;  // 每个地址对应一个用户实体

Music[] musics;     // 记录链上所有的音乐版权信息
Record[] records;   // 记录链上所有的授权信息
Notice[] notices;   // 记录所有通知消息
```

1.2 基本注册功能设计

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

```
//normal user registration
function registerUser(string _name, string _phone) public {
    account[msg.sender].name = _name;
    account[msg.sender].phone = _phone;
    account[msg.sender].kind = "user";
}
```

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

```
//company registration
function registerCompany(string _name, string _id, string _location, string _phone, string _email) public {
    account[msg.sender].name = _name;
    account[msg.sender].phone = _phone;
    account[msg.sender].kind = "company";
    account[msg.sender].id = _id;
    account[msg.sender].email = _email;
    account[msg.sender].location = _location;
}

//musician registration
function registerMusician(string _name, string _id, string _location, string _phone, string _email) public {
    account[msg.sender].name = _name;
    account[msg.sender].phone = _phone;
    account[msg.sender].kind = "musician";
    account[msg.sender].id = _id;
    account[msg.sender].email = _email;
    account[msg.sender].location = _location;
}

//judge registration
function registerJudge(string _name, string _id, string _location, string _phone, string _email) public {
    account[msg.sender].name = _name;
    account[msg.sender].phone = _phone;
    account[msg.sender].kind = "judge";
    account[msg.sender].id = _id;
    account[msg.sender].email = _email;
    account[msg.sender].location = _location;
}
```

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

```
//music registration
function registerMusic(string _bin, string _mname, string _alltime) public {
    musics.push(Music(msg.sender, _bin, _mname, account[msg.sender].name, true, _alltime));
}
```

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

```
//notice registration
function registerNotice(address _to, string _music, string _info) public {
    address _from = msg.sender;
    notices.push(Notice(_from, _to, _music, _info, false));
}
```

2. 功能实现函数设计

2.1 主要功能函数设计

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

```
function transferMusic(address _to, string _binhash, string _alltime) public {
    address owner = msg.sender;
    uint8 num = 0;
    for (num = 0; num < musics.length; num++){
        if(owner == musics[num].owner && keccak256(musics[num].bin) == keccak256(_binhash)){
            musics[num].owner = _to;
            musics[num].alltime = _alltime;
            records.push(Record(_to, msg.sender, _alltime,getM(_to,msg.sender), getI(_to,msg.sender),"transfer"));
        }
    }
}
```

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

```
function authorizeMusic(address _to, string _binhash, string _alltime, string _music, string _info) public {
    address owner = msg.sender;
    uint8 num = 0;
    //string gen = "authorize";
    for(num = 0; num < musics.length; num++){
        if(keccak256(musics[num].bin) == keccak256(_binhash) && musics[num].owner == msg.sender){
            records.push(Record(_to, msg.sender, _alltime, _music, _info,"authorize"));
        }
    }
}
```

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

```
function cancelMusic(string _binhash, string _alltime, string m, string i) public {
    uint8 num = 0;
    address owner = msg.sender;
    //string gen = "cancel";
    for(num = 0; num < musics.length; num++){
        if(musics[num].owner == owner && keccak256(musics[num].bin) == keccak256(_binhash) && musics[num].isvalid == true){
            musics[num].isvalid = false;
            musics[num].alltime = _alltime;
            records.push(Record(msg.sender, msg.sender, _alltime, m, i,"cancel"));
        }
    }
}
```

2.2 辅助函数设计

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。

```
function getMusic(uint numb) public view returns(address,string,string,string, bool,string){
    return (musics[numb].owner, musics[numb].bin, musics[numb].mname, musics[numb].singer, musics[numb].isvalid, musics[numb].alltime);
}

function getMusicNumber() public view returns(uint8[]) {
    uint8[] num2;
    for(uint8 i = 0; i < musics.length; i++){
        if (musics[i].isvalid && musics[i].owner == msg.sender) {
            num2.push(i);
        }
    }
    return num2;
}

function searchMusic(string _mname,string _singer) public view returns (address,string,string,string,string) {
    for (uint8 i = 0; i < musics.length; i++) {
        if (musics[i].isvalid && (keccak256(musics[i].mname) == keccak256(_mname)) && (keccak256(musics[i].singer) == keccak256(_singer))){
            return (musics[i].owner, musics[i].bin, musics[i].mname, musics[i].singer, musics[i].alltime);
        }
    }
}

function getRecord(uint numbe) public view returns(address,address,string,string,string,string){
    return (records[numbe].user,records[numbe].author,records[numbe].alltime,records[numbe].music,records[numbe].info,records[numbe].genre);
}

function getRecordNumber (address _user,address _author) public view returns(uint8[]) {
    uint8 num1 = 0;
    uint8[] num3;
    uint8 count = 0;
    for(num1 = 0;num1 < records.length; num1++){
        if((keccak256(records[num1].user) == keccak256(_user)) && (keccak256(records[num1].author) == keccak256(_author))){
            //return main info
            num3[count] = num1;
            count++;
        }
    }
    return num3;
}

function getUser() public view returns (string,string,string,string,string,string) {
    UserEntity storage user = account[msg.sender];
    return (user.name, user.kind, user.id, user.location, user.phone, user.email);
}

function getUserByAddress(address _user) public view returns (string,string,string,string,string,string) {
    UserEntity storage user = account[_user];
    return (user.name, user.kind, user.id, user.location, user.phone, user.email);
}
```

```
function getNoticeNumberByStart() public view returns (uint8[]) {
    uint8[] storage idxs;
    for (uint8 i = 0; i < notices.length; i++){
        if (notices[i].start == msg.sender){
            idxs.push(i);
        }
    }
    return idxs;
}

function getNoticeNumberByTo() public view returns (uint8[]) {
    uint8[] storage idxs;
    for (uint8 i = 0; i < notices.length; i++){
        if (notices[i].to == msg.sender){
            idxs.push(i);
        }
    }
    return idxs;
}

function getNotice(uint8 _idx) public view returns (address, address, string, string, bool) {
    return (notices[_idx].start, notices[_idx].to, notices[_idx].music, notices[_idx].info, notices[_idx].valid);
}

function consumeNotice(uint8 _idx) public {
    notices[_idx].valid = true;
}

function getHistory(string _music)public view returns (uint8[]){
    uint8 num1 = 0;
    uint8[] num3;
    uint8 count = 0;
    for(num1 = 0;num1 < records.length; num1++){
        if(keccak256(records[num1].music) == keccak256(_music)){
            //return main info
            num3[count] = num1;
            count++;
        }
    }
    return num3;
}
```


[1]区块链基础 PDF-微众银行

[2]《贝尔链：公链私链联盟链，谁才是地道的区块链？》

[3]智能合约 PDF-微众银行

[4]《商业复习：2019 中国数字音乐产业研究报告》

[5]中国传媒大学音乐与录音艺术学院：2018 年音乐人生存现况与版权认知抓鬼看调研研究报告

[6]无讼案例搜索数据库

[7]星瀚律师-新浪微博