基于区块链的可信共享办公及服务租赁平台

摘 要

共享办公是共享经济的重要组成部分。依托社会资源，为入住者提供低成本、便利化、全要素的服务平台。共享办公市场目前正处于高速发展阶段。同时，中心化共享办公平台也现存有一些问题和隐忧：专业程度不高；盈利模式单一；无法互联互通相互整合资源；没有建立可信的信用评价体系。其根本原因在于单一的中心化平台掌握了数据的读写权利，无法与其他平台建立信任。区块链[1]技术是一个多方共同维护的分布式账本技术，具有公开透明、不可篡改和可追溯的特性，可以在去信任中介的环境下达成各方的互信。区块链被认为是下一代的互联网和价值互联网。本毕业设计将通过区块链分布式存储、点对点共识、不可篡改、可追溯的特点和密码学相关原理、智能合约[2]的相关功能打通共享办公平台之间的壁垒，整合社会资源，并建立可信的共享办公服务体系。本毕业设计研究的方向主要包括四个方面：（1）分析区块链技术如何解决共享办公场景下资源分布的不均衡问题；（2）依靠区块链不可篡改的特性设计和实现一套可信的信用评价体系；（3）设计合理的积分发行、流通与消费体系并引入多元增值服务促进本平台的运转与价值流通；（4）用户交互界面使用 Android app 提供用户交互界面。最后根据实验结果进行总结并展望未来发展方向。

关键词： 共享办公，区块链，共享账本，信用评价体系

A Trusted Shared Office and Service Leasing Platform Based on Blockchain

ABSTRACT

Shared office is an important part of the sharing economy. Relying on social resources, it provides residents with low-cost, convenient and full-fledged service platforms. The shared office market is currently in a period of rapid development. At the same time, there are still some problems and hidden concerns in the centralized and shared office platform: the degree of professionalism is not high; the profit model is single; there is no interconnection and mutual integration of resources; there is no credible credit evaluation system. The fundamental reason is that a single centralized platform has the right to read and write data, and it cannot establish trust with other platforms. Blockchain[1] technology is a distributed ledger technology maintained by multiple parties. It is open, transparent, non-destructive, and traceable. It can achieve mutual trust in the context of trust brokering. Blockchain is considered to be the next generation of Internet and value Internet. This graduation design will open up barriers between shared office platforms and integrate social resources through blockchain distributed storage, peer-to-peer consensus, non-destructive, traceability, cryptographic correlation, and related functions of smart contracts [2]. Establish a trusted shared office service system. The direction of this graduation design research mainly includes four aspects: (1) How to solve the problem of unbalanced distribution of resources under the shared office scene; (2) Design and implement a set of reusable features that rely on the tamperable nature of blockchains. Credit's credit evaluation system; (3) Designing a reasonable point distribution, circulation and consumption system and introducing multiple value-added services to promote the operation and value circulation of the platform; (4) User interaction interface uses Android app to provide user interaction interface. Finally, based on the experimental results, we summarize and look forward to the future development direction.

Key words: shared office, blockchain, shared ledger, credit evaluation system

目 录

装 订 线

1 绪论

1.1 研究的背景和意义

1.1.1共享办公现状及问题

共享办公，指依托社会资源，提供包含工作、网络、交流和资源共享空间在内的各种类型的办公场所，为进驻者提供低成本、方便化、全要素服务的平台，包含具有创业企业孵化能力的众创空间。2015 年以来，我国在“大众创业、万众创新”的号召下，在政策鼓励之下众创空间雨后春笋般出现。

共享办公市场现状调查：

（1）共享办公空间调查

据 Vs SaaS 统计 2017 年全国共有 459 家众创空间。知名众创空间平台包括美国的 WeWork、国内的优客工场以及氪空间等。预计 2020 年将有超过 1200 家国家级共享办公空间。目前我国共享办公空间的发展状态仍然主要处于高速发展阶段。

（2）共享办公现存问题

中心化的共享办公及服务平台存在以下 4 个问题：（1）平台间各为信息孤岛、相互独立，无法进行资源互补互通；（2）平台与平台、平台与用户间数据不透明、信息不对称，单一平台的数据不被其他平台认可；（3）盈利模式单一，单个共享办公平台主要收入为工位租金；（4）因为互不信任其它凭条的数据而无法建立起跨平台间的信用评价体系，不存在信用与价值的传递。中心化共享办公平台存在的问题的根部院系是各平台间相互独立，无法建立全网可信的共享数据账本。传统中心化共享办公平台间的关系可用下图 1.1 表示。

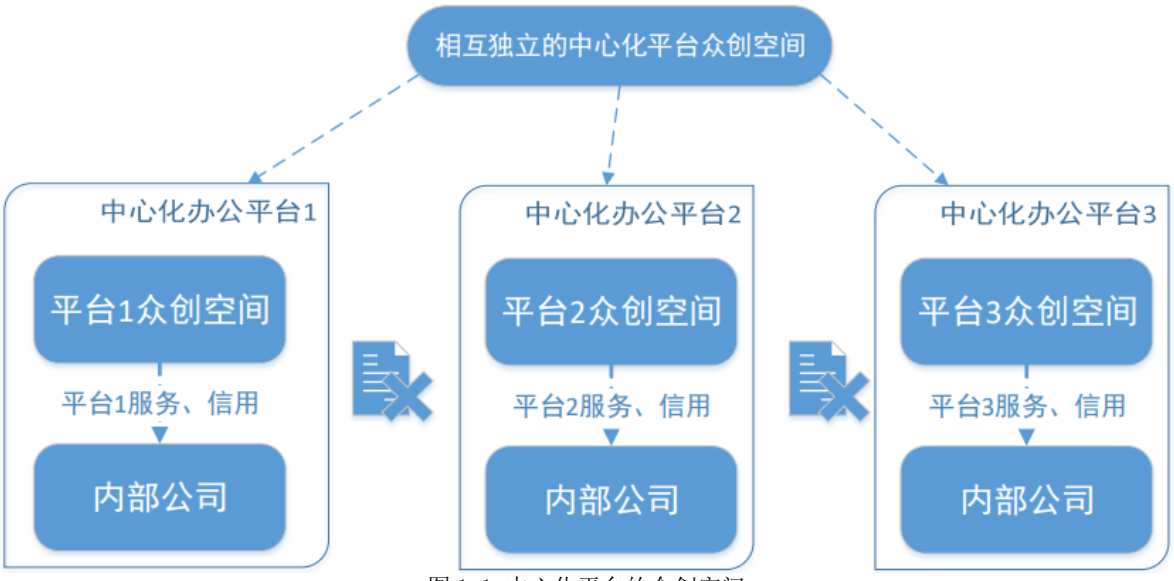

图 1.1 中心化平台的众创空间

（3）共享办公空间实地考察

本毕业设计实地考察了上海周边 A（代号）市 B（代号）新城附近的办公大楼以及部分众创空间的状况。该地区的众创空间的工位及配套资源的完善，某些中小型企业入驻在该新城附近众创空间

1.1.2区块链应用发展现状

区块链是指分布式数据存储、点对点传输、共识机制[3]、密码学技术等一些基础且成熟技术的集合。其特点是一个分布式存储、全网共识、不可篡改、数据公开透明的数据共享账本。

以区块链为底层支持环境，其本质相当于为基于区块链的应用提供操作系统运行上层应用。目前最流行的区块链系统包括比特币[4]区块链、以太坊[5]区块链、超级账本的 Fabric 区块链以及 IOTA 等区块链项目。区块链技术是近 20 多年来计算机技术成果的集合，包括计算机密码学，分布式账本技术(DLT) [6]，p2p 网络[7]，分布式共识机制和智能合约等技术。当今流行的区块链项目有：

（1）比特币(Bitcoin)：区块链项目的诞生的源头，是世界上第一个区块链项目引入了哈希函数[8]及 RIPEMD-160[9]编码等密码学技术。

（2）以太坊(Ethereum)：一个开源的具有图灵完备的智能合约（Smart Contract）的公有区块链平台。是第一个具有可编程合约的区块链平台,未来可能使用更加节省资源的 PoS[10]共识算法。

（3）超级账本(Hyperledger)：该项目打造透明、公开、去中心化的分布式账本项目。其中最受关注的项目为 Fabric， Fabric 专门针对企业级的区块链应用而设计。

（4）IOTA：IOTA 是为物联网（IoT）而设计的一个的新型交易结算和数据转移平台。其社区期望 IOTA 能够成为物联网公开的支柱，能够在所有设备中实现真正的互通。

区块链发展状况：

从诞生到现在，区块链从最初的“加密数字货币”形态的比特币区块链；到具有可编程、图灵完备的“智能合约”编程语言的以太坊区块链，及功能和应用场景更加丰富的区块链应用平台。区块链发展划分为三个阶段：区块链 1.0、区块链 2.0、区块链 3.0。

（1）区块链 1.0：可编程货币

为构建一个可信赖的、自由、无中心、有序的货币交易世界，可编程货币的出现让价值在互联网中直接流通交换成为可能。

（2）区块链 2.0：可编程金融

将“智能合约”加入区块链形成可编程金融。可编程金融已经在包括股票、私募股权等领域有了初步的应用，包括目前交易所尝试用区块链技术实现股权登记、转让等功能。

（3）区块链 3.0：可编程社会

在法律、零售、物联、医疗等领域，区块链可以解决信任问题。不再依靠第三方来建立信用和信息共享，提高整个行业的运行效率和整体水平。

1.2 课题及研究目的及方法

1.2.1研究目的

本项目研究内容是：（1）打通各共享办公平台以及众创空间之间的壁垒；（2）消除平台与用户之间的不信任关系。通过建立一个可信的共享办公及服务租赁平台来实现研究内容，其关键工作如下所示：

（1）构建一个去中心化的分布式网络。本去中心化网络还建立相关的身份控制体系来实现节点之间的权限控制，使得网络处于稳定可控的状态。

（2）建立信用评价体系来量化用户的可信程度。基于用户行为的积分作为量化信用的指标在各平台间传递，各平台都可在分布式共享账本平台进行用户积分的查询。

（3）引入多元增值服务。通过接入与共享办公运营管理培训相关服务和创业法律、融资等相关服务使得信用评价体系建立的信任具有丰富的使用价值。引导平台上的参与角色做出诚信行为。

区块链是计算机科学过去 20 年多成熟技术的结合，包括了计算机 p2p 网络、数据库技术、分布式系统、 分布式共识算法、计算机密码学等技术。其体系架构分为 6 层，从底层数据层到最上层应用层及其相关功能为：

（1）数据层将一段时间内接收到的交易数据封装到带有时间戳的数据区块中，并按时间顺序链接到当前最长的主区块链上，生成最新的区块。

（2）网络层设计特定的传播协议和数据验证机制，使得每个节点都能参与区块数据的校验和记账过程。

（3）共识层保障各节点区块数据的有效性达成共识。

（4）激励层提供激励机制和措施，鼓励节点参与区块链的安全工作。

（5）合约层封装区块链系统的各类脚本代码、算法以及由此生成的更为复杂的智能合约。合约层是实现区块链系统灵活编程和操作数据的基础。

（6）应用层面向用户，提供用户交互界面。

区块链的去中心化是区块链项目的关键要素，其网络图谱结构是一个扁平化的对等网络，不存在中心化的机构。分布式对等网络的每一个计算机节点可以为连接到该节点用户界面提供全网络一致的服务。区块链的网络拓扑结构如下图 1.2 所示。

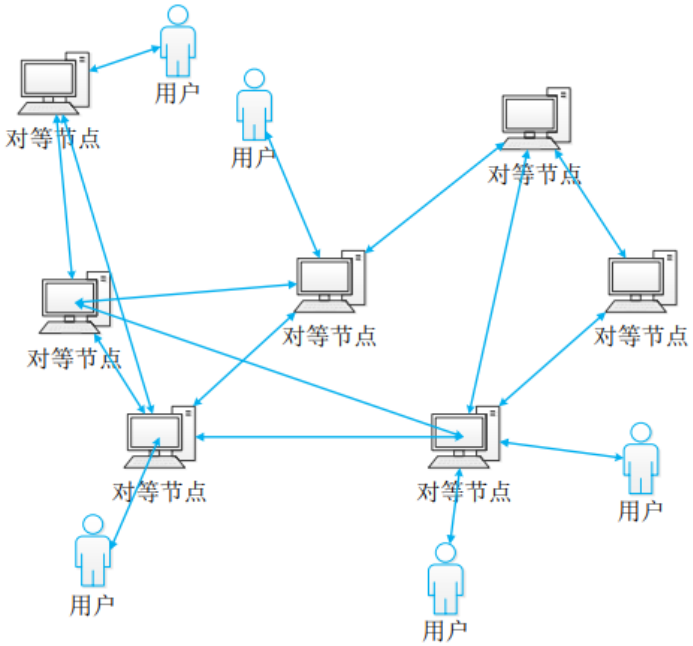

图 1.2 区块链网络拓扑结构

这种不由中心化机构提供服务，而由对等节点经过分布式共识协议提供服务与记账功能的网络所带来的特点是区块链网络对外呈现为一个分布式存储、全网共识、不可篡改、可溯源、数据公开透明的共享账本。

通过区块链的分布式共享账本技术可以解决原有共享办公平台的问题有：

（1）信息孤岛问题。通过分布式记账使得业务上相关联的各方共同持有数据账本，各平台间的数据信息不再孤立。

（2）数据不对称、不透明的问题。通过建立多方共享的数据账本使得单一的机构无法对数据进行隐藏，而是各正常工作的节点提供一致的数据查询服务。

（3）无法传递信任与价值。由于共享数据账本不可篡改、可追溯的特点，使得信任与价值的载体可以在各平台间低成本地传递与运行。

（4）信任需要强大的中央机构背书。区块链去中心化的特点正是由于各方互不信任的原因而采取共同技术数据账本的机制。多方共同记账与对账使得区块链传递的信任无需任何中心化机构背书。

与中心化共享办公平台最大的区别在于通过分布式共享账本打通了个平台间的信息与资源壁垒，使得信任与价值得以在各平台间低成本地传递。本设计的区块链网络在共享办公场景下的结构可用下图 1.3 所示。

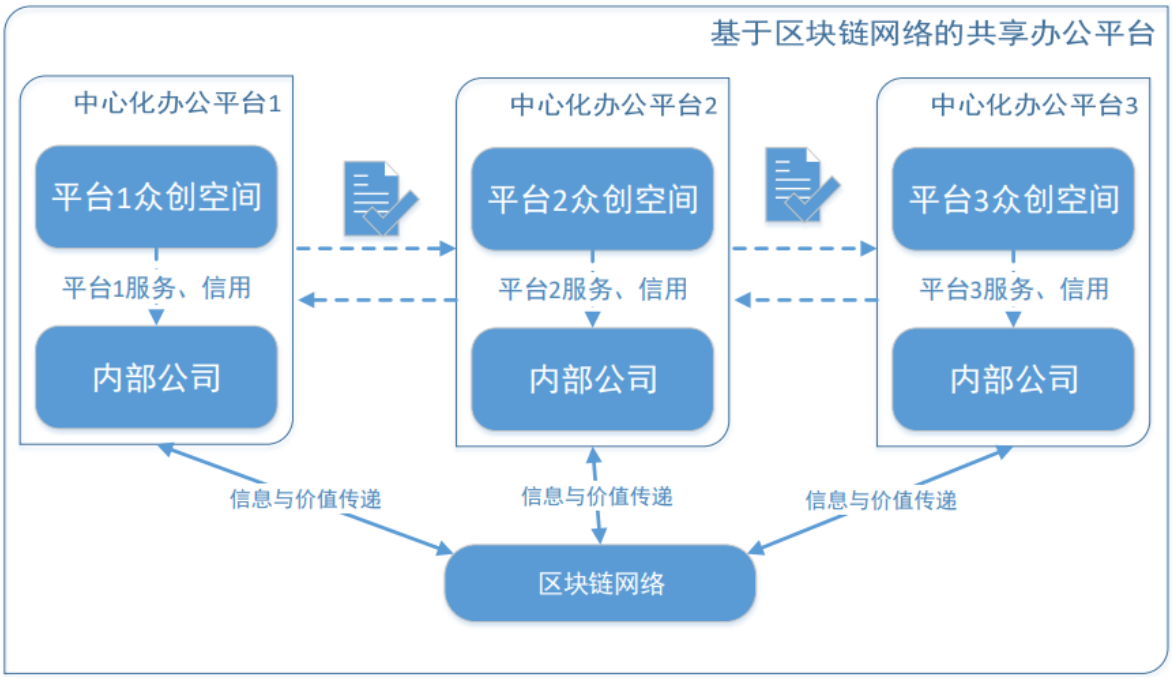

图 1.3 基于区块链共享办公平台

本项目使用的研究方法如下：

1.2.2分布式账本实现共享数据

分布式账本技术（DLT，Distributed ledger Technology）又称为共享账本。本设计通过在不同节点之间达成共识，使得每个节点记录相同的账本数据，从而达成节点之间的互信。

本项中采用背书/共识（Endorsemnet/Consensus）模型。本项目网络中共有三种类型的节点，分别是背书节点（Endorser）、提交节点（Commiter）和排序节点（Orderer）。

对交易进行验证和模拟执行在背书节点中进行，该类型的节点具有验证数字签名和模拟执行智能合约的功能。

通过分布式账本技术，使得网络中每个节点都参与维护分布式账本，共同持有分布式账本数据的备份，使得区块链网络的数据变得公开透明、不可篡改，从而是可信的。

1.2.3通道技术隔离数据

在本项目区块链网络中使用了身份验证机制，在确定了各节点的真实身份之后便可以通过共享的通信通道（Communication channel）来实现交易广播服务的设置。由客户端与对等网络节点共同加入到一个通信通道，往往是为了完成一组商业合作关系而建立，可称为商业通道（business channel）。加入到同一通道的所有对等节点会收到来自排序节点相同的输出数据，并且拥有相同的交易顺序。

1.2.4智能合约实现业务功能

智能合约在区块链网络中约定了各当事人或参与方的协议内容。参与方都对合约的内容达成一致，并遵守合约执行的结果。智能合约部署之后能立即自动生效。本项目中使用的智能合约局域图灵完备性，能完成本设计中整个区块链网络中所有的业务逻辑。

1.2.5并发机制实现性能测试

区块链的交易性能是区块链项目最大的局限于瓶颈，本设计将通过 go 语言的并发访问控制机制和异步进程之间的数据同步机制实现对区块链网络的性能测试，主要是测试区块链网络每秒处理交易的速率（Transactions Per Second，tps）。并通过性能测试的结果提出区块链网络性能优化的方向。

1.3 论文的主要内容

本设计研究的方向主要包括四个方面：（1）分析通过区块链技术如何解决共享办公场景下资源分布的不均衡问题；（2）依靠区块链不可篡改的特性设计和实现一套可信的积分评价体系；（3）设计合理的积分发行、流通与消费体系促进本平台的运转；（4）应用层使用 Android 手机 APP 进行相关功能展示并根据实验结果进行总结并展望未来发展方向。

基于对考察的 A 市 B 新城附近的众创空间资源构建了一个最小的区块链网络系统并进行了场景的假设与模拟，最终在该模拟环境下能使得资源的共享不再局限于某一企业级平台，通过积分建立全网可信的信用评价体系。本文的结构安排如下：

（1）第一章绪论介绍研究的背景、意义。介绍了共享办公平台的市场状况、存在问题及区块链技术的进展。以及通过区块链的方法和技术能解决的传统共享办公平台哪些问题。

（2）第二章分析通过区块链技术解决共享办公现有问题的理论方法指导、设计总体业务逻辑并结合实地考察搭建网络模型。

（3）第三章结合理论设计一套区块底层框架的实现，并用 Android app 提供用户交互页面。

（4）第四章是测试结果展示与分析，分为三部分：区块链底层网络节点启动测试；项目的功能测试；项目的性能测试。

（5）第五章是结论与展望。对项目的不足以及对未来拓展方向的思考。

装 订 线

2 平台设计与理论支持

2.1 平台业务逻辑

鉴于以上信息，本毕业设计核心的开发部分为区块链网络的部署、对区块链网络功能调用的SDK 的开发和部署以及应用程序的开发。在此之前充分调研的市场情况，然后先后确定了平台的功能和基本的业务逻辑。如下表 2.1 所示。

表 2.1 平台功能简述

类型	功能
基本功能	租赁服务、评价体系
附加功能	购物、餐饮（外卖）、论坛、广告发布
增值服务	投融资、借贷、股权登记、法律、财务、培训

业务逻辑总图如下图 2.1 所示。

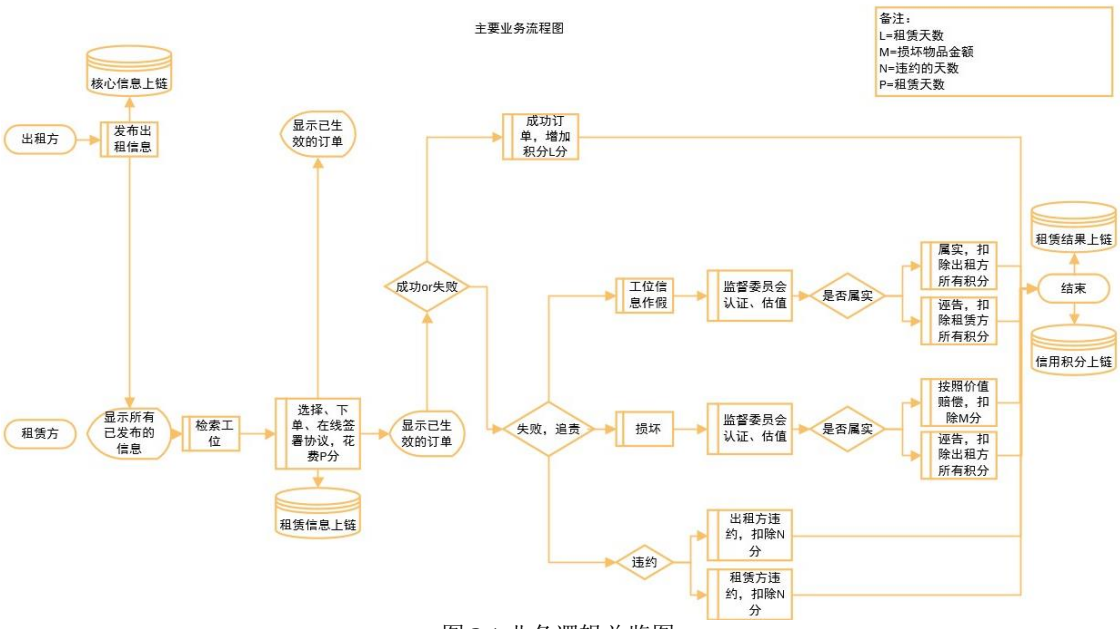

图 2.1 业务逻辑总览图

整个项目的业务逻辑部分大致可分为 4 部分，分别是应用层应用(Application)、软件开发工具包 SDK、智能合约(chaincode)、共享账本(shared Ledger)。其中智能合约月共享账本属于区块链底层基础，而应用 Application 和 SDK 更接近于与用户交互的上层应用。基本的区块链网络不在本项目的业务逻辑的考虑范围。

本项目中将构建以下相关的模块内容，建立完善的区块链底层到应用层的体系架构。

(1) 应用(Application)

应用是用户接触到的可视化界面，提供用户数据采集，提交用户请求与响应用户请求；封装好的软件开发工具包 SDK 提供 API 调用区块链接口，并将访问的返回的数据进行进一步的封装方便用户查看。

(2) 软件开发工具包(SDK)

SDK 一方面接收用户从应用前端发送的 HTTP 请求，另一方面封装用户发送的数据，通过 gRPC(google 主导开发的 RPC[11]远程过程调用)与区块链部分的智能合约进行交互。

(3) 智能合约(smart chaincode)

智能合约封装了经过各方同意的业务逻辑，是区块链中进行业务逻辑的核心机构，可以再线下进行编写，经过各方共识之后安装与实例化到由权限执行合约的节点上。

(4) 共享账本(shared ledger)

由只能合约处理约定的业务逻辑，经过共识协议最终记录在了分布式的上共享账本。

参与到平台中的用户包括：工位租赁者、工位发布者与多元增值服务发布者。主要的业务逻辑包括：

注册，通过应用前端将用户数据存储到区块链账本，如图 2.2 所示。

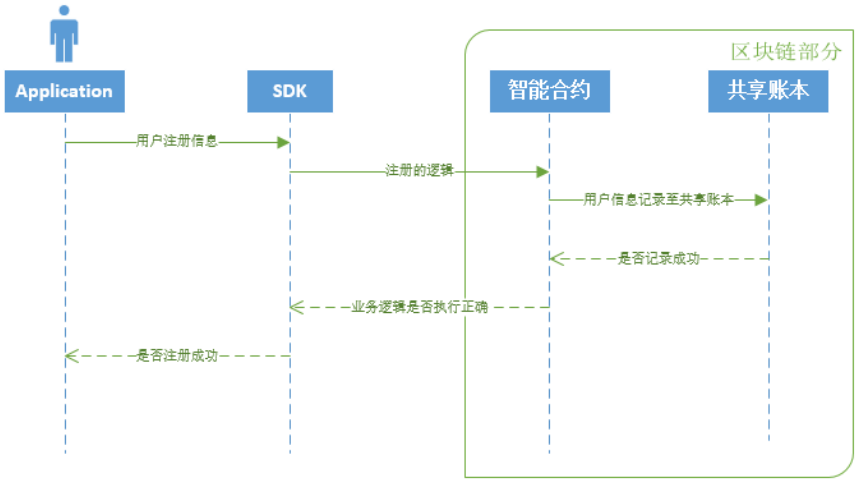

图 2.2 注册

登录，通过用户名和密码验证登录，如图 2.3 所示。

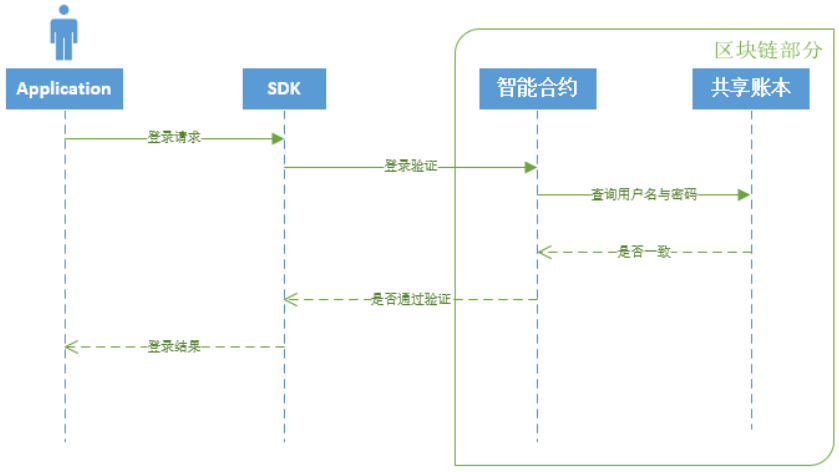

图 2.3 登录

众创空间管理者发布工位/服务获取积分，服务支付方式可为人民币或积分，如图 2.4 所示。

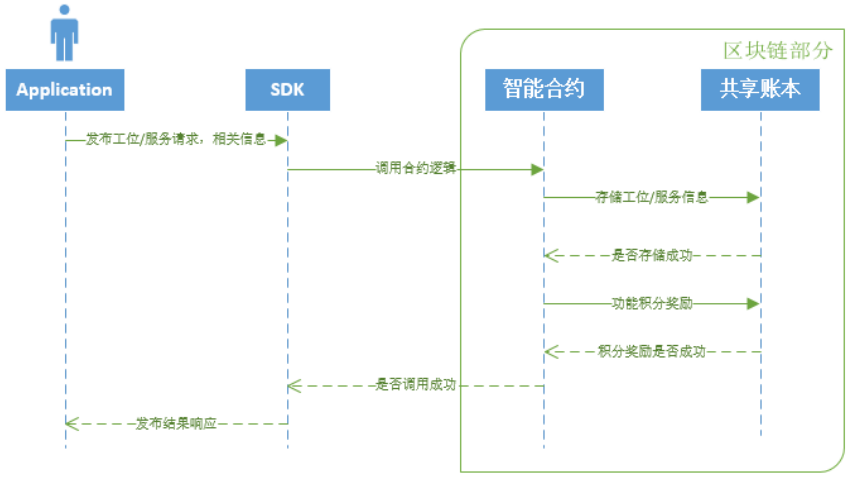

图 2.4 租赁者发布服务获取积分

租赁者租赁工位/服务，支付方式为发布者指定的支付方式为人民币或积分，如图 2.5 所示。

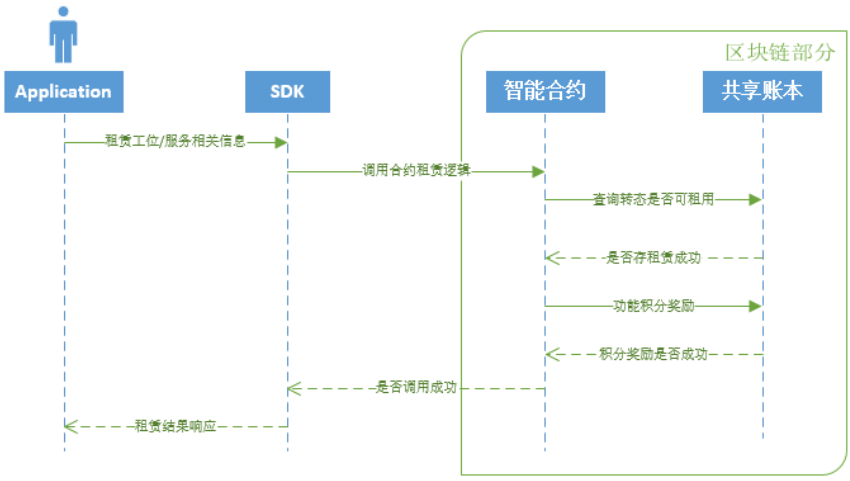

图 2.5 租赁工位/服务

租赁者对工位/服务的评价，如图 2.6 所示。

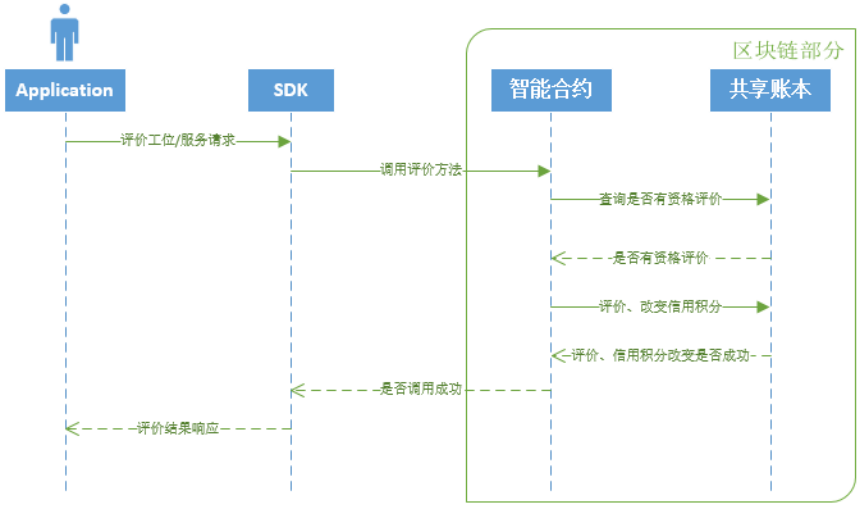

图 2.6 工位/服务评价

众创空间及多元增值服务提供商对租赁者使用服务的评价如下图 2.7 所示。

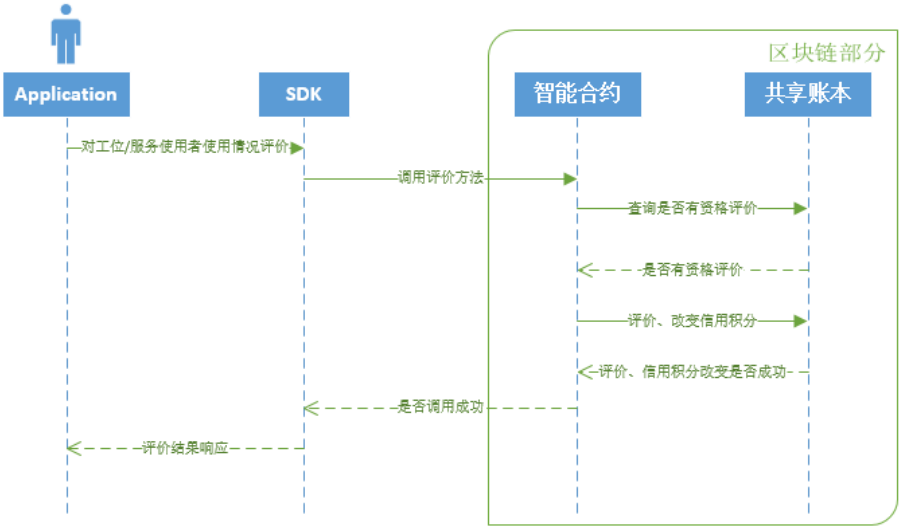

图 2.7 对使用者的评价

项目的核心业务逻辑是构建信用评价体系，信用评价体系的建立基于用户行为产生的信用积分。通过区块链分布式存储与点对点共识机制形成不可篡改的信用评价体系。基于这个积分体系可以在平台打造的生态内进行丰富的资源分配权限调整。使得信用积分高者具有更多的优惠与便利，更能接触到更多高级的服务需求。如创业公司的法律、人才招聘资源、公司运营培训以及投

融资等稀缺资源的需求。而共享办公空间和多元服务提供商也能够因为提供高质量地服务吸引更多的用户，从而使得自身的服务更具有价值

2.2 实现方法指导

针对以上研究内容和工作，对应的研究方法如下：

2.2.1建立 p2p 网络

建立 p2p 网络的过程包括了路由发现、节点连接与数据传播三部分。

（1）路由发现

常用的路由发现功能可通过：（1）配置文件预先对需要连接的节点的 IP 地址与连接端口号 port 写入配置文件；（2）通过命令行工具在节点启动之后进行配置；（3）部署种子节点服务器存储可连接节点的路由表。

本毕业设计使用的方法是事先在配置文件中写入需要连接的节点的 IP 地址与端口号。通过配置在文件中读取连接的 IP 地址和端口。即通过事先配置路由表的方法实现了节点发现的功能。这符合个企业布置的节点在要建立商业合作时事先告知其他合作伙伴的场景。而命令行工具的使用则可以进行动态添加节点加入到 p2p 网络。

（2）节点连接

节点连接使用 gRPC 协议。

RPC（Remote Procedure Call，远程过程调用）通过网络从远程计算机程序上请求服务。该协议基于输协议 TCP 或 UDP，在本设计中使用 TCP 协议为通信程序之间携带信息数据。

RPC 采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。

由于在 p2p 对等网络中每个节点既是服务器又是客户端，所以每个节点既有 RPC 服务器的功能又有 RPC 客户端的功能。

gRPC 是由 google 公司开发的 RPC 系统协议。

本项目中使用了 Go 语言实现的 gRPC 协议来进行节点间的相互访问。使用 protocol buffers 作为接口定义语言，来描述服务接口和有效载荷消息结构。protocol buffers 作为接口定义语言比 xml 或 json 作为接口定义语言其描述文件只需 xml 或 json 原来的 1/10 至 1/3，解析速度是原来的 20 倍至 100 倍。Protocol buffers 定义的数据和结构放在一个.proto 文件之中。

如下图 2.8 所示定义了一个 rpc 服务。

```
// 定义了一个可与客户端双向访问的rpc服务
service Events {
    rpc Chat(stream SignedEvent) returns (stream Event) {}
}
```

图 2.8 rpc 服务的定义

在本设计中的每个对等节点都开放了两个端口提供 gRPC 服务，分别处理交易提案信息和交易确认信息。对等节点开放的节点为 7051 和 7053 端口。

（3）数据传播

本项目中采用基于 Gossip 协议实现 P2P 对等网络节点之间的数据分发。本项目中的 Gossip 协议模块负责连接节点后，实现从单个源节点到所有节点的数据分发。

利用 P2P 实现数据广播最简单的实现方法是洪泛（Flooding）。洪泛在节点接收到数据包后直接转发给所有的邻居节点，指导所有节点都接收到了数据包，或者数据包的跳数（Hop Count）超过一定的限制。

Gossip 协议与洪泛的广播策略不同点在于，节点在接收到数据包之后不是直接转发给邻居节点。在本项目中采用的是随机选择 3 个节点进行转发。

数据传播使用 Gossip 协议和洪泛法的优缺点对比如下表 2.2 所示：

P2P 数据分发方法	优点	缺点
洪泛	节点覆盖率高、冗余度好网络健壮。	低效，可能会出现广播风暴。
Gossip	转发效率高、扩展性好、适应性强、优雅降级。	极端情况下某些节点很难收到广播消息。

表 2.2 数据传播协议比较

Gossip 协议在本设计中主要完成的功能和如下所示：

（1）不需要所有节点都连接到排序服务获取区块数据的情况下，网络中所有的节点还有相同的数据信息。

（2）系统网络正常启动运行之后，对于新加入到网络中的节点可以从对等节点或排序服务的节点中获取到区块数据信息。

（3）能使得未同步最新区块数据的节点能够获取到缺失的区块信息。

（4）维护和管理成员节点的信息，跟踪哪些节点是可提供服务的，哪些节点是有故障的，并能根据各节点状态重新调整网络拓扑。

（5）同步数据的速度要快，能保证大量的数据可以在节点之间传输。

2.2.2网络的共识机制

分布式系统需采用共识协议使得 p2p 分布式网络节点对外呈现一致的功能服务。同时要确保区块里的交易有效且顺序一致。本项目中采用了 solo 单程序和 Kafka 消息队列完成排序。本项目中节点类型可分为三种：排序服务节点（Orderer）负责对交易进行排序；背书节点（Endorser）负责对对客户端发送的请求提案进行验证与背书；确认节点（Commiter）负责对区块进行确认，即记录到本地的账本中背书节点兼有确认节点的功能。

共识机制由三个几点完成：

（1）客户端向背书节点提交提案进行签名背书后将数据返回到客户端。

（2）客户端将背书后的交易提交给排序服务统一进行交易的排序，生成新的区块。

（3）排序服务将区块广播给确认节点验证交易后写入本地账本。即确认节点将从排序服务生成的交易集合同步到本地账本。

交易的流程可用下图 3.9 表示，完成了上述共识机制的三步内容，顺序按时间从上而下发生。

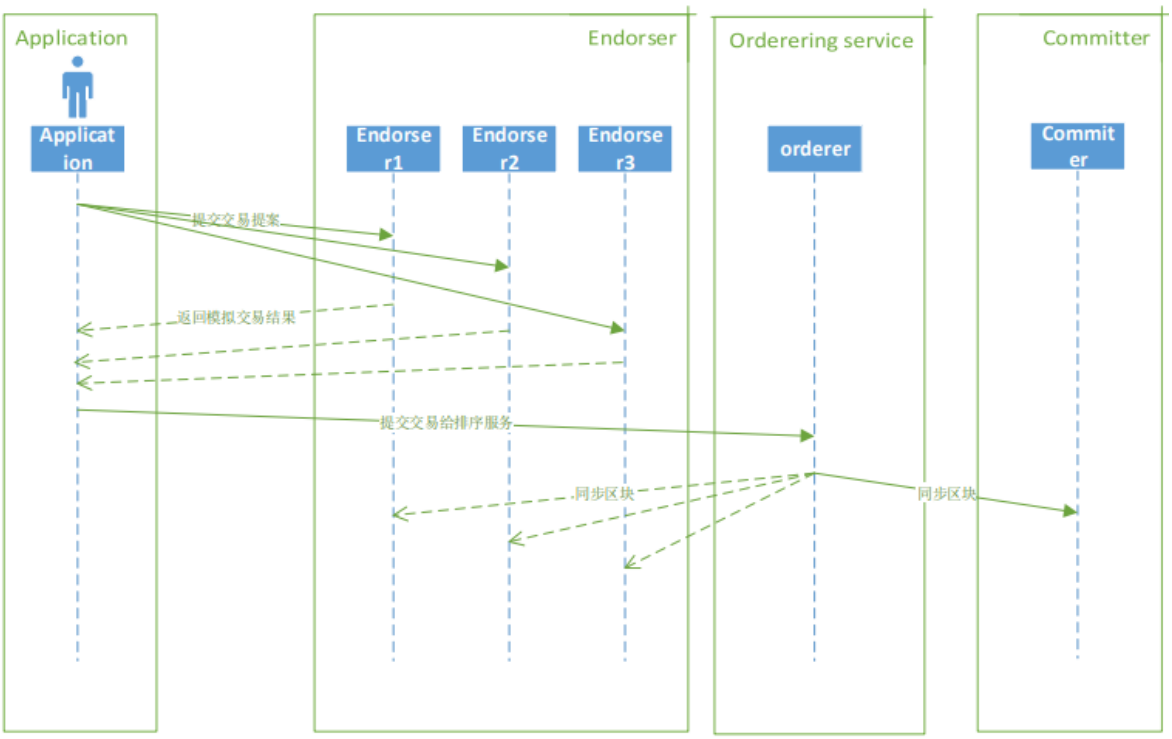

图 3.9 交易流程

其中排序服务可以是单程序 solo 或 Kafka 消息队列。

当使用单程序 solo 的时候排序服务只有一个排序节点，而使用 Kafka 消息队列的时候排序服务由节点的集群组成。在模拟测试环境下一般使用单程序 solo。

2.2.3智能合约实现业务逻辑

智能合约是可自动执行的脚本的集合，具有图灵完备性，能完成相关的业务逻辑。

本项目中使用智能合约来完成信用评价体系的构建和多元增值服务的引入以此来打造闭环生态空间：

（1）信用评价体系的构建

平台内设置有两种积分：功能积分与信用积分。能使得功能积分增加的操作有：发布工位、预定工位、发布普通多元增值服务、购买多元增值服务、参与对工位或服务的评价；会使得减少积分的操作有：使用普通增值服务；能使得信用积分增加的操作有：按照智能合约的约定执行合同、发布的工位或服务获得好评；会使得信用积分减少的操作有：未按照职能合约的约定执行合同、发布的服务被基于差评。

（2）多元增值服务的引入

普通的增值服务可使用功能积分进行消费，包括：在平台上发起任务、购买平台引入商家的各类服务如酒店预订、外卖、电影票和外卖等等。

高级的增值服务需要信用积分达到一定的高度才能租用或购买，包括：企业间合作对接、高级合作如软件开发的帮助、金融知识、法律知识、运营管理的培训、投融资等。

（3）闭环生态空间的建立

通过租用需求、租用交易、信用评价体系三者之间构建一个良性循环的健康价值生态。其内环生态位：租用需求对租用交易具有引流作用；租用交易的体验促进用户进行评价；用户评价作为反馈能促进服务质量的提升，服务质量的提升将进一步改善内环生态的租用需求。从而构建了一个良性的内环生态，在未来将开放接口与其他的区块链平台进一步对接，构建外环生态。整个项目的生态环境可用下图 2.10 表示。

装 订 线

图 3.

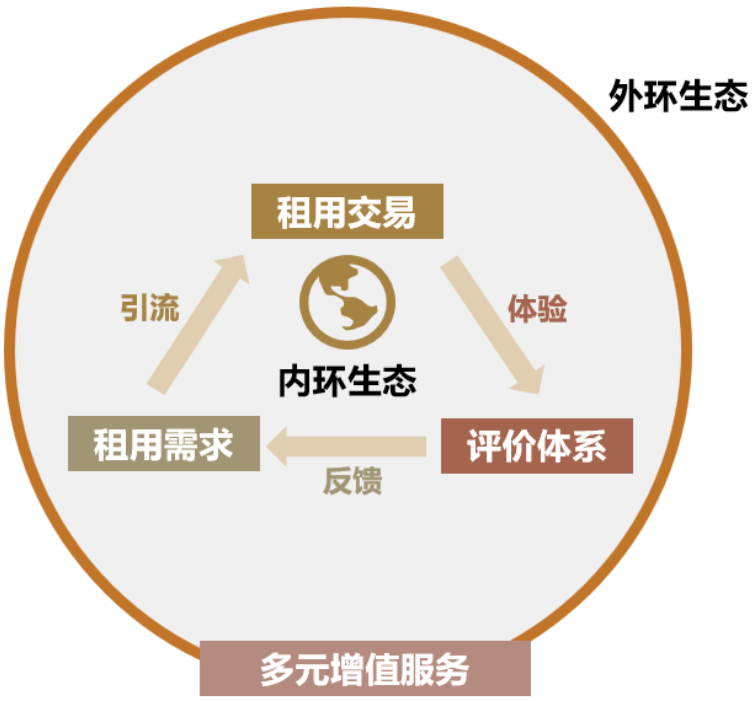

图 2. 10 项目的生态闭环

本项目智能合约是图灵完备的开发语言，能实现项目需要在区块链上完成各方约定的业务逻辑。

2.2.4构建中间层服务器

使得客户端 app 能方便地与区块链进行交互，避免与多个节点连接的中间环节是构建中间层服务器的必要原因。本项中开发的 SDK（软件开发工具包）提供 gRPC 协议连接的 API，包含了网络中各节点的身份认证、成员管理服务、对交易的处理、对区块的查询和对交易的查询等功能。

SDK 提供 API 方便 Android 前端发出的 HTTP 的 POST 类型请求对区块链网络进行交互。SDK 服务器本质是一个 HTTP 服务器，响应 Android 前端发出的 HTTP 的 POST 类型的请求。本毕业设计中采用的是基于 Node.js 编程语言实现 SDK。

使用 Express 框架来快速构建 SDK 服务器。该框架的实现基于 MVC(Model View Controller) 设计模式。

SDK 提供用于访问区块链的 API 主要包括功能下：

(1) 创建通道

(2) 请求对等节点 peer 加入通道

(3) 在对等节点 peer 中安装 chaincodes

(4) 在通道中实例化用户链码 chaincode

(5) 通过调用 chaincode 来执行交易

(6) 查询账本的交易或区块

具体的 SDK 的 API 部分调用方式如下表 2.3 所示。

表 2.3 SDK 提供 API 的接口

API 功能	API 名称及参数列表	API 参数类型	返回值
创建应用通道	createChannel(req)	req 为 channelRequest 类型结构体	创建通道请求是否被 orderer 界节点接受
加入应用通道	joinChannel(req, timeout)	req 为 joinChannelRequest 类型，timeout 为表示超时的数值，单位为毫秒	请求成功的目标节点数组
安装用户链码	installChaincode(req, timeout)	req 为 ChaincodeInstallRequest 类型对象，timeout 为表示超时的数值，单位毫秒	ProposalResponseObject 对象
实例化用户链码提案	sendInstantiateProposal(req, timeout)	req 为 ChaincodeInstan -tiateUpgradeRequest 对象，timeout 为表示超时的数值，单位毫秒	ProposalResponseObject 对象
实例化用户链码交易	sendTransaction(req)	req 为 TransactionRequest 类型对象	HTTP 状态码，200 表示成功

SDK 与区块链的交互通过 gRPC 远程过程调用，访问区块链中 orderer 排序服务节点开放的 7050 端口。

2.2.5交互界面

智能手机的普及和移动互联网时代的到来，以及众创空间的办公人员经常出差在外不在室内电脑前面。本项目采用基于 Android 的 app 做为前端。Android 前端保留扫描二维码的方便之后进一步扩展 App 的功能。

Android 应用对于区块链网络是一个应用前端，Android 内部可分为两部分：提供用户交互并采集用户输入数据的前端，以及封装 SDK 提供的 API 格式和用户数据并发送到区块链网络的后台服务。

（1）前端功能

布局采用 XML。Java 可绑定 XML 对象的属性来获取用户输入或者动态设置 XML 的属性以改变 UI。

Android 前端使用一个 activity 表示一个用户交互界面，用不同的 activity 表示不同的用户界面，在 activity 间灵活地切换来控制用户界面。用户交互界面包含的 activity 及相关的用户输入属性如表 2.4 所示。

表 2.4 用户交互窗口界面 activity

activity 名称	需要输入的属性	实现功能
MainActivity	无	首页
SigninActivity	用户名、密码	注册
LoginActivity	用户名、密码	登录

续表 4.4

activity 名称	需要输入的属性	实现功能
MyInfoActivity	无	用户信息
PactActivity	编号、类型、名称、价格、描述、地点、容量	发布租赁合同的历史信息
RentInfoActivity	用户名	租赁历史记录
TransferPointActivity	转账方、接收方、额度	转送积分
CrditInfoActivity	用户名	查看信用积分
ToTopActivity	发布编号、使用积分额度	使用积分使发布的服务排名靠前
ShopActivity	无	消费积分的便利商店
StationInfoActivity	服务编号	工位信息

（2）内部交互接口

Android 后端使用 c++实现数据结构化封装。与 Android 前端进行 socket 通信，开放端口为 50000，把那个与 SDK 服务器进行 HTTP 通信，发送端口为 4000。

Android 后端使用有限状态机机制来处理不同的用户端请求，对应状态及处理函数的功能由表 2.5 表示。

表 2.5 操作码与处理函数

操作码	定义常量	处理函数功能
CLIENT_USER_SIGNUP	1	注册
CLIENT_USER_SIGNIN	2	登录
CLIENT_USER_CHGPWD	3	修改密码
CLIENT_USER_REVISE	4	修改个人信息
CLIENT_USER_QUERY	5	查询个人信息
CLIENT_USER_TRANSFER	7	积分转账
CLIENT_STATION_APPLY	8	申请工位
CLIENT_STATION_REVOKE	9	撤销工位
CLIENT_STATION_RENT	10	租用工位
CLIENT_STATION_RETURN	11	返还工位
CLIENT_STATION_QUERY	12	查询工位
CLINET_ STATION _POINT	13	使用积分申请工位
CLINET_ STATION _TOP	14	使用积分使服务排名靠前

Android 窗口界面（Activity）生命周期介绍

在日常应用中 Activity 是与用户交互的接口，它提供了一个用户完成相关操作的窗口。在开发中创建 Activity 后，通过调用 setContentView(View)方法来给该 Activity 指定一个布局界面，该界面就是提供给用户交互的接口。Android 系统中是通过堆栈的方式来管理 Activity 的，而 Activity 自身则是通过生命周期的方法来管理的 Activity 的创建与销毁，Activity 的形态如下：

（1）Active/Running:

Activity 处于活动状态，此时 Activity 处于栈顶，是可见状态，可与用户进行交互。

（2）Paused：

当 Activity 失去焦点时，或被一个新的非全屏的 Activity，或被一个透明的 Activity 放置在栈顶时，Activity 就转化为 Paused 状态。但我们需要明白，此时 Activity 只是失去了与用户交互的能力，其所有的状态信息及其成员变量都还存在，只有在系统内存紧张的情况下，才有可能被系统回收掉。

（3）Stopped：

当一个 Activity 被另一个 Activity 完全覆盖时，被覆盖的 Activity 就会进入 Stopped 状态，此时它不再可见，但是跟 Paused 状态一样保持着其所有状态信息及其成员变量。

（4）Killed：

当 Activity 被系统回收掉时，Activity 就处于 Killed 状态。

Activity 会在以上四种形态中相互切换，至于如何切换，这因用户的操作不同而异。基于了解了 Activity 的 4 种形态后，Activity 的生命周期如下图 2.11 所示。

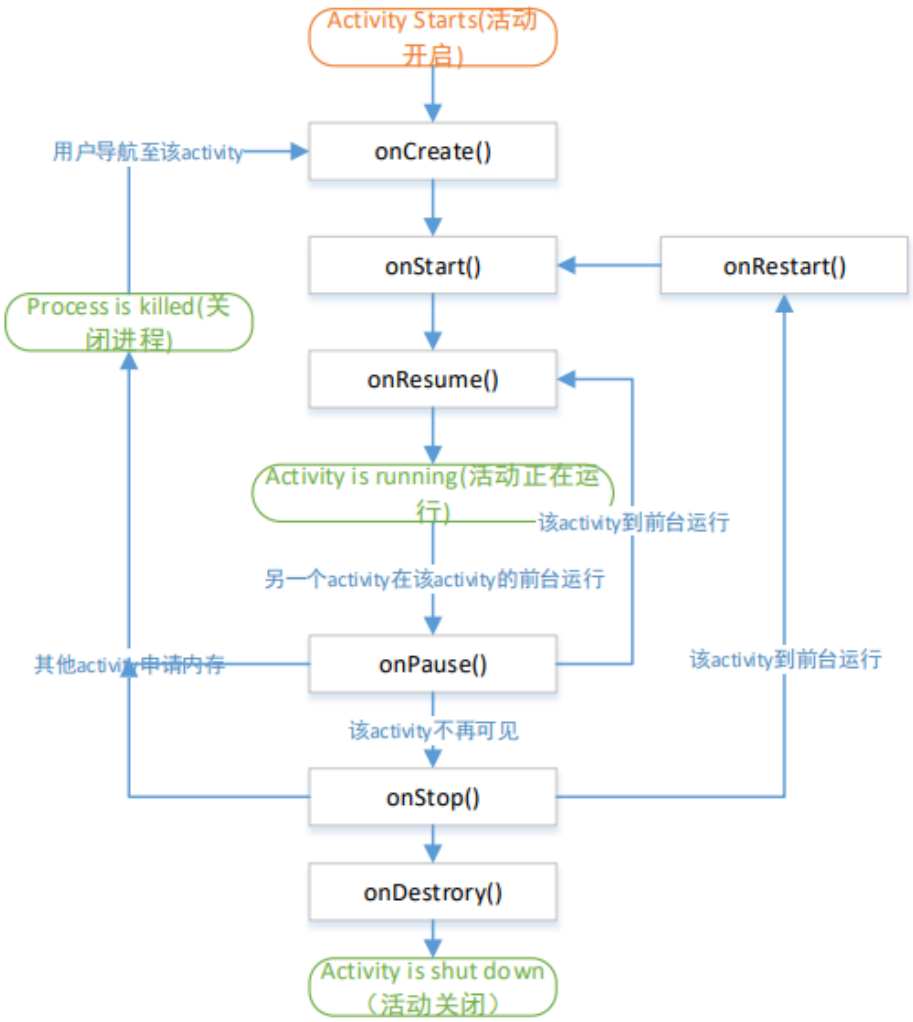

图 2.11 Activity 生命周期

2.3 模拟场景设计

联盟区块链超级账本 Fabric 是为企业级应用打造的区块链底层服务平台。超级账本 Fabric 目前流行的可商业应用的版本为 1.0 版本和 1.1 版本，本毕业设计选用了 1.0 版本。其版本的优势如下：

（1）参与记录数据的节点加入需要身份认证与审核，使得这种区块链网络较为稳定，减少了记账时共识的时间和消耗的电力资源。

（2）灵活的访问控制权限，通过多通道的建立可以选择与联盟中的商业合作伙伴进行数据共享，而与并无合作关系的节点不进行数据共享。

（3）各模块之间完全解耦，可针对区块链底层中的不同模块部分进行相应的开发和设计，并通过相应的接口进行交互。

本项目中针对超级账本 Fabric1.0 具体模块的设计如下：

（1）数据层

超级账本 Fabric 1.0 数据层封装了区块链数据的存储方式与数据结构。区块的存储方式：基于文件系统的存储于基于非关系型的数据库(NoSQL)的存储方式，使用的数据库为 LevelDB 和 CouchDB。为保证交易速度 Fabric 区块链将一批（多笔）交易打包成一个区块链一次性记录到 Peer 节点的分布式账本，交易的数量及区块最大生成时间可在配置文件中进行设置。

为了方便对每个节点数据进行验证，数据存储的方式采用了“块-链”式的结构。区块的区块头存储着前一区块头的哈希值，方便其他对等的 Peer 节点验证某一 Peer 存储的数据是否被篡改。交易被存放在区块体当，并默克尔树(Merkel tree)的形式组织，用来验证区块的完整性。

（2）网络层

基于 p2p 网络的去中心化的分布式应用系统。参与实体需经过授权与验证才可加入到区块链网络，网络较为稳定且部分节点的自由退出不会影响区块链系统的稳定性与安全性。

本毕业设计基于联盟链超级账本 Fabric 1.0。可通过成员管理服务提供商模块来进行身份验证的网络，而证书颁发机构(Certificate Authority,CA)基于公钥基础设施 PKI(Public Key Infrastructure)建立网络成员实体的身份认证系统。组成的网络较为稳定，同时也减少了作出现恶节点的可能性和数量。以此保证区块链应用网络的安全。在网络启动之前 CA 服务器需要给网络成员实体颁发证书以确定身份，其流程为：

（1）客户端（Fabric-CA client）向 CA 服务器发送请求获得身份证书

（2）CA 服务器（通常为顶级 CA 的安全会设置中间层的 CA）将客户端的信息存在 CA 服务器（单个或集群）的数据库中（如 Mysql）。

（3）客户端获得证书与注册身份后可通过 SDK 访问区块链网络，即 Peer 组成的 p2p 网络。

过程如图 2.12 所示。

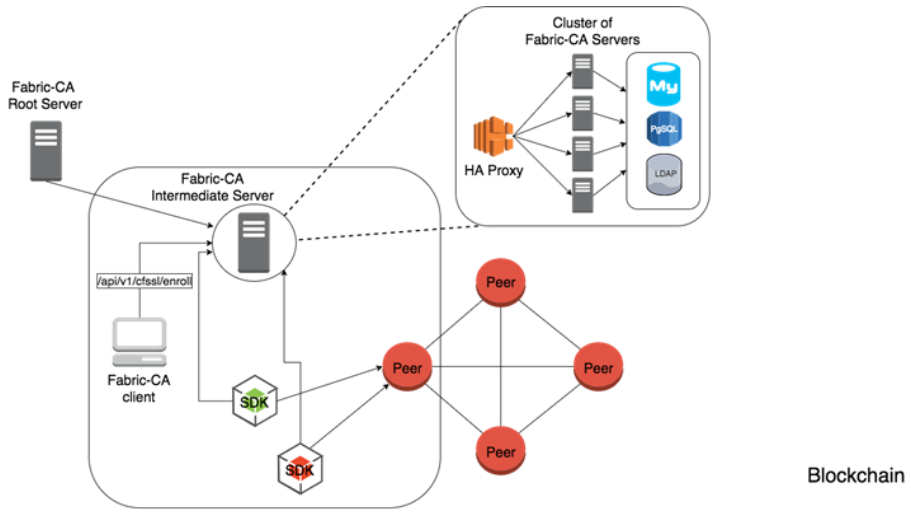

图 2.12 证书申请流程

本毕业设计基于在 A 市 B 新城附近的 A 大厦、B 大厦实地考察了其内部 A 众创空间和 B 众创空间。

对等节点为 4 个：A 众创空间、B 众创空间以及附近其他两个监管组织：A 管委会和 B 城区控股集团有限公司总计 4 个节点。基于这 4 个节点建立分为两个组织（organization）和 1 个提供排序服务功能的测试环境， 2 个证书签发机构在完成网络各实体的证书签发之后可暂时脱离网络。选定组织 1 中的第一个节点 A 管委会与组织外的网络进行通信，第二个节点表示由 A 众创空间部署的一个节点；组织 2 中的第一个节点 B 城区控股集团负责与组织外进行网络通信，B 众创空间部署的为组织内第二个节点。实排序服务节点由本平台部署运维。组织间节点数据可进行相互通信，各节点间的数据公开透明。

结合本项目前期在 A 市 B 新城的分析，网络的拓扑图如下图 2.13 所示。

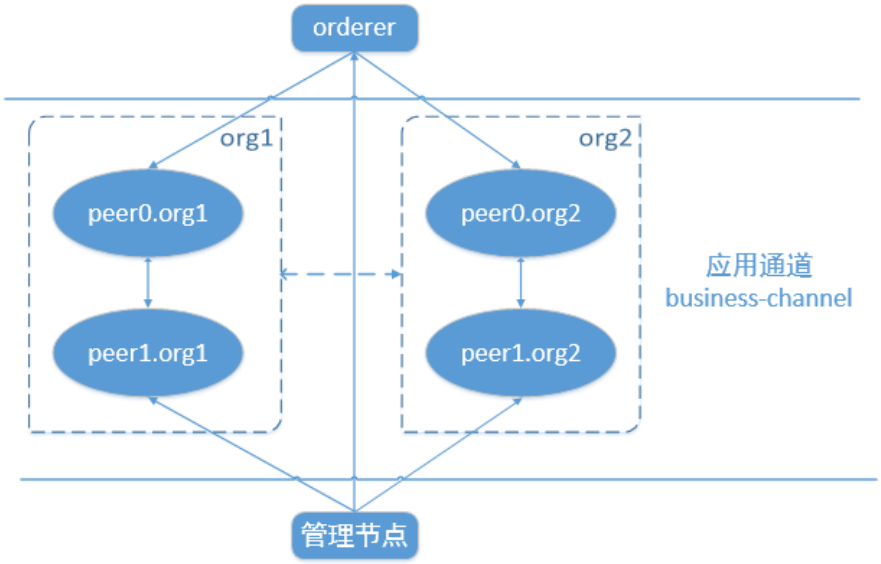

图 2.13 A 市 B 新城测试环境网络拓扑结构

本设计的区块链部分使用 IP 为 202.120.167.93 的主机搭建，其各个节点的的部署方及域名如下表 2.6 所示：

表 2.6 对应节点部署详情

部署方	开放端口	节点标识	节点名称	所属组织	提供服务
本平台	7050	orderer	orderer.example.com	Orderer	交易共识排序
A 管委会	7051、7053	peer0	peer0.org1.example.com	Org1	提案背书 交易确认
A 众创空间	7056、7058	pee1	peer1.org1.example.com	Org1	提案背书 交易确认
B 城区控股集团	8051、8053	peer2	peer0.org2.example.com	Org2	提案背书 交易确认
B 众创空间	8056、8058	peer3	peer1.org2.example.com	Org2	提案背书 交易确认

（3）共识层

高效地达成共识是分布式的区块链网络的重要组成部分。分布式网络不可避免地会出现网络分区现象，在 FLP[14]不可能定理指导下 CAP 原理是分布式系统在一致性 (Consistence)、可用性（Availability）、分区容忍性（Partition tolerance）上的取舍，本项目牺牲一致性来提高可用性，但数据账本最终满足一致性。

分布式网络系统需要考虑宕机与拜占庭将军问题[15]。共识机制选用 solo 单程序用于测试环境，而使用 Kafka 消息队列用于生成环境。本测试环境使用 solo 单程序共识机制，能容忍对等节点的宕机与拜占庭节点[15]的行为。常用算法有 BFT[15]类算法，如 PBFT[16]算法可容忍 33%的拜占庭节点。本设计中排序服务节在生成环境中使用 Kafka 消息队列集群来达到容忍排序服务中节点宕机与拜占庭节点。

（4）激励层

本平台为了在区块链网络中构建可信的信用评价，引入了传统互联网平台的积分机制，通过引入多元增值服务使得积分增值。

使用了两种积分：功能积分与诚信积分。积分的发行、流通、使用环节包括：

1. 积分的发行：基于用户活动包括发布工位/服务、预定/服务工位产生功能积分；基于用户的诚信行为被评为好评产生诚信积分。

2. 积分的流通：功能积分可转赠给平台内部用户，诚信积分不可转赠。

3. 积分的使用：功能积分可用于预定工位、部分多元增值服务的支付手段；诚信积分可用于获取高级的多元增值服务，如法律援助、投资融资机会等。

用户积分作为基于用户行为的信用评价体系。用户既包括租赁者、提供工位租赁的众创空间、平台自身以及后续加入到区块链网络的多元服务提供商。

多元增值服务可使用积分兑换。可兑换的多元增值服务包括：

(1) 排名优化：在有提示使用的积分优化排名的情况下使得服务处在显眼的位置。

(2) 创业投融资：入驻的企业及投资人可以通过积分优先进行对接。

(3) 培训服务：在法律、财务和运营培训等方面的优先资源可使用积分进行兑换。

(4) 本地生活服务：与日常生活先关的权益如外卖、电影票和租车等服务可使用积分兑换。

用户可将产生的积分在用户之间流通和消费。积分的价值取决于入驻商家提供的服务的质量与数量。一段时间后需要对产生积分的难度进行调整，以控制在平台上积分的数量，保证其稀缺性，从而保证其流通价值。

（5）合约层

本项目中的智能合约(smart contract)有如下特性：

各参与方对合约的内容达成一致，遵守合约执行的结果。也是计算机可读可执行的代码。

智能合约是完全自我执行和自我强制：自我执行是指合约能自动生效；自我强制是指合约的参与方能根据最优的结果自行决定是否参与或者终止和关联方的联系。该过程不需要第三方干预。

智能合约实例化之后是可独立运行的应用程序，运行在基于 Docker 的沙箱环境中。

智能合约生命周期包括智能合约打包(package)、安装(install)、实例化(instantiate)和升级(upgrade)。在不删除实例化合约情况下可以停止和启动智能合约。还可以通过调用(invoke)和查询(query)方法来调用执行链码中的业务逻辑，完成合约调用。合约的生命周期，如图 2.7 所示：

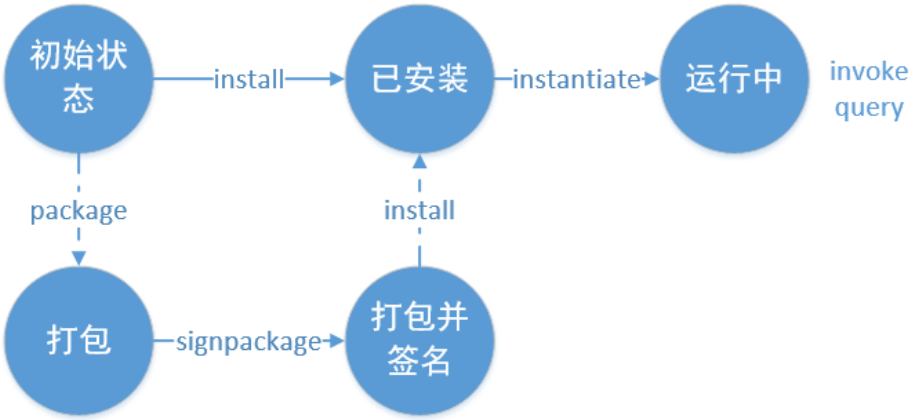

图 2.7 链码生命周期

（6）应用层

应用前端发往 SDK 的数据应采用密文传输,以防止数据被监听或篡改。

在本设计中采用基于 JWT(JSON Web Token)的 token 认证机制。JWT 的声明被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，该 token 也可直接被用于认证，可被加密。

JWT 由三部分组成：头部、载荷与签名。

token 可通过 HS256 等算法对传输数据进行加密。

3 架构及实现

3.1 测试环境与模块关联

（1） 测试环境

本设计为避免项目各部分过度耦合采用了模块化的开发思路。每一部分的服务单独开发，然后提供接口供其他服务进行调用。

本项目使用到了云服务器一台，是 Linux 内核的 centos 7 版本，以及 windows 10 操作系统用于开发 Android 移动应用程序，并且在 windows 10 操作系统中安装 Android 虚拟机用于 Android 移动应用程序的调试。

使用到的编程语言为：Node.js、Golang、C++、Java 与 XML。

使用的工具主要有：docker 和 docker-compose。这两个工具用于打包应用及依赖包到一个可移植的容器中。

（2）模块关联

本设计的项目搭建需要实现区块链组网、智能合约开发、SDK 接口实现与 Android 应用前端的实现。基于 Fabric 1.0 区块链而对区块链底层网络重新构建。在此基础之上开发基于 Android app 访问 SDK 提供的 API 与区块链网络进行交互。在 SDK 与区块链网络中则使用安全传输层协议(Transport Layer Security,TLS)确保在应用程序 Android 前端与 SDK 服务端的数据保密服务。通过基于椭圆曲线密码学(Elliptic curve cryptography，ECC)[12]的数字签名可以验证交易的合法性。

各模块间的数据交互方式使用了 socket 通信、HTTP 的 POST 方法通信和 gRPC 通信，为了保证数据不可篡改性与隐私使用到了数字签名、TLS 等与网络数据安全传输相关技术,确保区块链网络中各模块的安全通信[16]。各模块间的数据加密、安全验证与数据交互机制可用下图 3.1 表示。

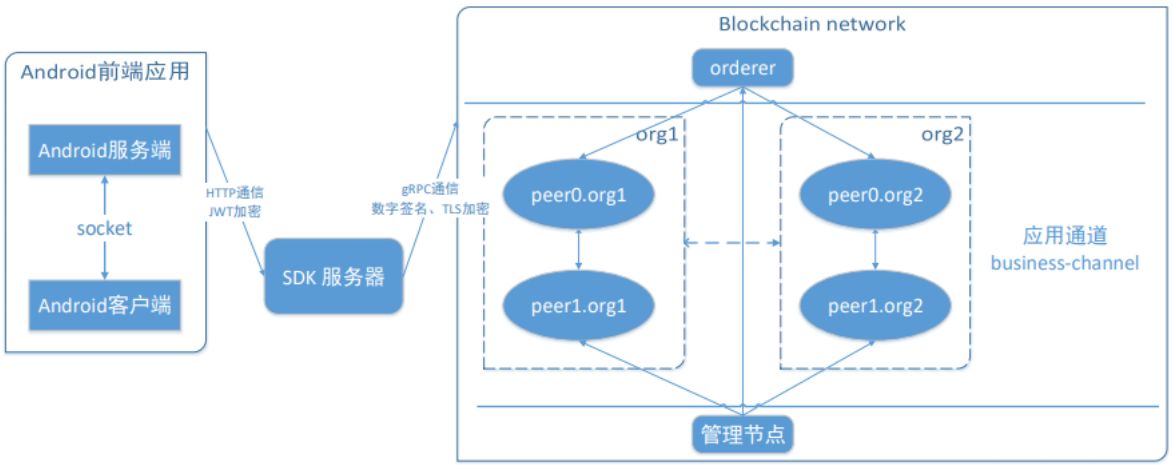

图 3.1 项目模块间数据交互

3.2 底层实现

3.2.1网络结构设计与配置

（1）网络拓扑配置

本设计使用 docker 容器镜像技术来模拟多个节点进行区块链的组网。使用隔离的容器与容器之间与裸机映射的端口实现数据互访，所需的镜像及相关的含义如下表 3.1 所示。

表 3.1 节点依赖的镜像

镜像名称	镜像版本	父镜像	镜像功能
hyperledger/fabric-baseos	x86_64-0.3.1	ubuntu:xenial	ubuntu 基础镜像，用于生成其他镜像，包括 orderer、ca、peer 以及 go 链码容器
hyperledger/fabric-ccenv	x86_64-1.0.0	hyperledger/fabric-baseosimages	支持 go 语言链码容器的基础镜像
hyperledger/fabric-orderer	x86_64-1.0.0	hyperledger/fabric-baseos	orderer 节点，安装了 orderer 相关文件
hyperledger/fabric-ca	x86_64-1.0.0	hyperledger/fabric-baseos	ca 镜像，安装了 ca 相关文件
hyperledger/fabric-peer	x86_64-1.0.0	hyperledger/fabric-baseos	peer 节点镜像，安装了 peer 相关文件

编写配置文件，排序服务节点为例，ca 证书颁发机构节点和组织 1 中的对等节点 peer 等配置分别如下图 3.2、3.3、3.4 所示，属性的解释如下表 3.2 所示。

```
orderer.example.com:
  container_name: orderer.example.com
  image: hyperledger/fabric-orderer:x86_64-1.0.0
  environment:
    - ORDERER_GENERAL_LOGLEVEL=debug
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_GENESISMETHOD=file
    - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/genesis.block
    - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
    - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/crypto/orderer/msp
    - ORDERER_GENERAL_TLS_ENABLED=true
    - ORDERER_GENERAL_TLS_PRIVATEKEY=/etc/hyperledger/crypto/orderer/tls/server.key
    - ORDERER_GENERAL_TLS_CERTIFICATE=/etc/hyperledger/crypto/orderer/tls/server.crt
    - ORDERER_GENERAL_TLS_ROOTCAS=[/etc/hyperledger/crypto/orderer/tls/ca.crt, /etc/hyperledger/crypto/peerOrg1/tls/ca.crt, /etc/hyperledger/crypto/peerOrg2/tls/ca.crt]
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderers
  command: orderer
  ports:
    - 7050:7050
  privileged: true
  volumes:
      - ./channel:/etc/hyperledger/configtx
      - ./channel/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/:/etc/hyperledger/crypto/orderer
      - ./channel/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/:/etc/hyperledger/crypto/peerOrg1
      - ./channel/crypto-config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/:/etc/hyperledger/crypto/peerOrg2
```

图 3.2 Orderer 排序节点启动配置

```
ca.org1.example.com:
  image: hyperledger/fabric-ca:x86_64-1.0.0
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem
    - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/f1f68f3fc993173736b766364391e60a43180b938b17b818883bc9123e2bc93f_sk
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem
    - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-server-config/f1f68f3fc993173736b766364391e60a43180b938b17b818883bc9123e2bc93f_sk
  ports:
    - "7054:7054"
  privileged: true
  command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
  volumes:
    - ./channel/crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config
  container_name: ca_peerOrg1
```

图 3.3 ca 节点配置

```
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  extends:
    file:   base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
  ports:
    - 7051:7051
    - 7053:7053
  privileged: true
  volumes:
      - ./channel/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/:/etc/hyperledger/crypto/peer
  depends_on:
    - orderer.example.com
```

图 3.4 组织 1 中的 peer0 节点与 pee1 节点

以 Orderer 排序服务配置文件为例说明指定一个域名为 orderer.example.com 的节点配置信息，指定了 docker 容器启动的名称、依赖的镜像名称、容器启动后环境配置、工作空间、访问命令、容器与裸机间的端口映射关系、容器启动后的访问权限等，各项配置内容的解释如表 4.2 所示：

表 3.2 Orderer 节点容器启动配置

属性名称	值	作用
domain_name	orderer.example.com	识别节点所属域
container_name	orderer.example.com	节点唯一表示符
image	hyperledger/fabric-orderer:x86_64-1.0.0	依赖的 docker 镜像名称及版本号
enviroment	ORDERER_GENERAL_LOGLEVE ORDERER_GENERAL_LISTENADDRESS ORDERER_GENERAL_LOCALMSPID ORDERER_GENERAL_TLS_ENABLED	设置日志级别、服务监听地址、本地 MSP 服务 Id、是否使用 TLS 等设置
working_dir	/opt/gopath/src/github.com/hyperledger/fabric/orderers	设置工作空间，文件存储路径等
command	orderer	启动容器是运行的命令
ports:	7050:7050	容器与主机端口之间的映射关系，左边代表主机端口，邮编代表容器端口
privileged	true	是否以管理员权限运行容器
volumes	- ./channel:/etc/hyperledger/configtx -……	挂载卷：将裸机上的文件目录添加容器的目录

（2）数字证书配置与生成

在启动 Fabric 网络前需要提前网络实体成员配置文件，主要包括 MSP（成员服务提供商）相关的文件，用于鉴定网络成员的身份，TLS 安全传输相关文件，用于节点间通信时数据加密。

配置文件中定义了两种服务类型(OrdererOrgs 和 PeerOrgs)的相关组织。每个组织中又可以定义多个节点(Spec)及注册用户(User)。由此网络拓扑结构中的三个域：example.com、org1.example.com 和 org2.example.com 可产生相关的身份证书文件、用于签名的私钥和确保数据加密传输的 TLS 配置文件。其相关配置信息如下图 3.5 和图 3.6 所示：

```
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com

    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer
```

图 3.5 OrdererOrgs 域配置

```
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Org1
  # ---------------------------------------------------------------------------
  - Name: Org1
    Domain: org1.example.com
    # ---------------------------------------------------------------------------
    # "CA"
    # ---------------------------------------------------------------------------
    CA:
       Hostname: ca # implicitly ca.org1.example.com

    # ---------------------------------------------------------------------------
    Template:
      Count: 2
      # Start: 5
      # Hostname: {{.Prefix}}{{.Index}} # default
      SANS:
        - "localhost"
    # ---------------------------------------------------------------------------
    # "Users"
    # ---------------------------------------------------------------------------
    # Count: The number of user accounts _in addition_ to Admin
    # ---------------------------------------------------------------------------
    Users:
      Count: 1
```

图 3.6 PeerOrgs 域配置

其 MSP 文件与 TLS 文件相关配置及主要功能如下表 3.3 所示。

表 3.3 网络拓扑配置

配置属性	存放实体	依赖配置文件	主要功能
MSP	Peer、Orderer、客户端 App	cryptogen.yaml	证书文件和签名私钥，用来管理网络实体成员省份信息
TLS	Peer、Orderer、客户端 App	crypto-config.yaml	网络启用 TLS 时使用 TLS 证书

（3）创世区块生成与通道配置及生成

Orderer 排序服务节点在启动时需要提前生成创世区块作为区块链通道的初始配置。创世区块可用 configtxgen 工具生成，读取的配置文件的为 configtx.yaml。

在创世区块中需要指定交易排序的类型、打包区块存储的交易数量、存储的交易的大小限制等内容。具体配置信息如下表 3.4 所示。

表 3.4 创世区块的配置

配置属性	数值	含义
OrdererType	Solo	排序方法，测试环境使用单节点排序
Addresses	orderer.example.com:7050	排序服务监听地址
BatchTimeout:	2s	排序服务打包区块的最大时间间隔
MaxMessageCount	10	排序服务打包区块时容纳最大的交易数量
PreferredMaxBytes	512KB	一个区块最大的容量

装

（4）生成应用通道的配置

应用通道可指定组织之间的通信与数据共享的权限，在本设计中使用了排序服务组织，以及基于排序服务构建的两个对等节点组织的应用通道，所以在原本相对独立的连个组织 Org1 与 Org2 之间可以使用共同记录同一笔交易和共享数据。主要配置信息如下表 3.5 所示。

订

表 3.5 应用通道的配置信息

配置属性	数值	含义
Orderer	OrdererOrg	提供排序服务的组织，为 orderer 单节点(基于 Solo)或 orderer 集群(基于 Kafka)
Consortiums	SampleConsortium	组织之间的联盟，本设计中为 Org1 与 Org2
Application	- *Org1 - *Org2	应用通道包含的，Org1 与 Org2
AnchorPeers	- &Org1: - Host: peer0.org1.example.com Port: 7051 - &Org2: - Host: peer0.org2.example.com Port: 7053	两个组织设置的锚节点，分别代表组织中的节点集群与联盟（consortiums）中的其他组织进行通信

线

3.2.2SDK 服务器

SDK 服务器采用 Node.js 语言开发。启动成功后与区块链网络中的节点采用 gRPC 协议连接。开发 HTTP POST 方法的接口与 Android 前端连接。展示效果如下图 4.所示开启了 SDK 服务器监听 Android 前端发送消息：

SDK 开启 4000 端口通过 HTTP 的 POST 方法监听网络服务，接收的消息格式为：

http post 请求的消息头需要的内容为：

（1）权限访问控制，用于拦截无权限的客户端伪造的交易提案，设置“authorization”的属性为“Bearer”+“jwt 验证消息”

（2）设置发送数据内容格式，设置属性“content-type”的值为“application/json”。

消息体为 json 格式，如下图 3.7 所示。

```
{
    "username":"用户名",
    "orgname":"组织名",
    "channelName":"通道名称",
    "peers": ["对等节点名称"],
    "chaincodeName":"合约名称",
    "functionName":"调用方法",
    "args":["参数1","参数2","参数3","参数n"]
}
```

图 3.7 发送消息体格式

返回的数据类型如下图 3.8 所示。

```
{"success":"bool类型true或false","message":"返回消息，可为字符串或Json对象格式"}
```

图 3.8 返回消息类型

3.2.3智能合约实现

本设计使用了三个用户链码来处理不同的业务逻辑，分别是：

register：用于处理与用户相关的信息，实现登录、注册、身份信息修改、账户余额、功能积分、信用积分变动等功能。

publicService：发布工位/服务、申请工位/服务相关,同时在内置方法中提供接口调 register 合中的方法来修改用户信息，如账户金额变化、积分变化等。

comment：评论工位/服务质量相关，同时可调用 register 合约修改用户信息；也可调用 publicSrvice 合约修改工位/服务的状态。

结构化对象

在合约中将使用的数据结构化为抽象为三类对象，分别是用户、工位/服务、工位/服务相关评论。

用户模型结构说明如表 3.6 所示。

表 3.6 用户模型及其说明

字段名称	数据类型	序列化 json 名称	描述
ID	字符串	id	用户唯一标识符
Password	字符串	password	用户密码
Role	字符串	role	用户角色，“1”代表普通租赁众创空间的用户，“2”代表可以租赁工位的众创空间和孵化器企业
Balance	浮点型	balance	用户账户余额
CrdPoint	整型	crePoint	信用积分，基于用户信用行为产生的积分

续表 3.6

字段名称	数据类型	序列化 json 名称	描述
FunPoint	整型	funPoint	功能积分

工位/服务模型结构说明如表 3.7 所示。

表 3.7 工位/服务模型及其说明

字段名称	数据类型	序列化 json 名称	描述
ID	字符串	id	工位/服务唯一标识符
OwnerID	字符串	ownerId	所有者标识符
Type	字符串	type	类型，"1"工位，"2"服务
Name	字符串	name	工位/服务名称
Function	字符串	function	功能
Money	浮点型	money	价格
FunPoint	整型	funPoint	需要花费的功能积分
Description	字符串	description	描述
State	字符串	state	状态，"0"表示可租用，"1"表示被租用
Top	布尔型	top	是否设置特权支付积分，false 表示非
TopPoint	整型	topPoint	设置特权支付的积分，至少为 1
UserID	字符串	userId	当前租用者标识符
Comments	结构数组	comment	租用者评论集合

用户评论模型说明如表 3.8 所示。

表 3.8 用户评论模型及其说明

字段名称	数据类型	序列化 json 名称	描述
CusHash	字符串	cusHash	用户名称标识符
Grade	整型	gevel	评分，从 1 到 5。1 表示最低评分，5 表示最高评分
Comment	字符串	comment	评论留言

合约中存储着处理相关业务逻辑的方法，用于处理 Android 应用发送的请求。register 合约、publicService 合约和 comment 合约的包含的方法及说明见表 3.9、表 3.10 和表 3.11。

表 3.9 register 合约的方法列表

方法名称	传入参数个数及说明	实现功能
regist	6 个。ID：用户唯一标识符；Password：用户密码；Role：用户角色；Balance：账户余额；CrdPoint：信用积分；FunPoint：功能积分。	注册用户及初始化用户账户。
login	2 个。ID：用户唯一标识符；Password：用户密码。	验证用户身份。
changePwd	3 个。ID：用户唯一标识符；OldPassword：旧用户密码；newPassword：新用户密码。	修改用户密码。
update	4 个。ID：用户唯一标识符；Balance：账户余额；CrdPoint：信用积分；FunPoint：功能积分。	更新用户账户，包括余额、功能积分、信用积分。

续表 3.9

方法名称	传入参数个数及说明	实现功能
delete	1 个。ID：用户唯一标识符。	删除用户状态。
query	1 个。ID：用户唯一标识符。	查询用户最新状态。
getHistoryForKey	1 个。ID：用户唯一标识符。	获取用户历史状态。
transfer	3 个。ID1：支付方 ID；ID2：接收方 ID；FunPoint：转账额度。	功能积分转赠。

表 3.10 publicService 合约的方法列表

方法名称	传入参数个数及说明	实现功能
addService	8 个。ID：工位/服务唯一标识符；OwnerID：发布者标识符；Type：类型；Name：服务名称；Money：价格；描述；所在地点；工位容量。	创建工位/服务
addServiceByFunPoint	8 个。ID：工位/服务唯一标识符；OwnerID：发布者标识符；Type：类型；Name：服务名称；FunPoint：需花费的功能积分；描述；所在地点；工位容量。	创建工位/服务，租赁者需使用功能积分进行租赁
appService	ID：工位/服务唯一标识符；userID：租赁者唯一标识符。	申请工位/服务，支付方式为账户余额
appServiceByFunPoint	ID：工位/服务唯一标识符；userID：租赁者唯一标识符。	申请工位/服务，支付方式为功能积分
topService	ID：工位/服务唯一标识符；TopPoint：功能积分	发布者花费一定的功能积分使得自己发布的服务/工位排名靠前
resetService	ID：工位/服务唯一标识符；state：新的状态	设置工/服务的状态
addComment	userID：租赁者唯一标识符；serviceID：工位/服务的唯一标识符；grade：租赁者给予的评分	租赁者增加对工位/服务的评论
query	ID：工位/服唯一标识符	查询工位/服务最新状态
deleteService	ID：工位/服唯一标识符	删除工位/服务的状态
getHistoryForKey	ID：工位/服唯一标识符	查询工位/服务的历史状态
addComment	userID：评论者唯一标识符；serviceID：工位/服务唯一标识符	增加评论
query	userID：评论者唯一标识符；serviceID：工位/服务唯一标识符	查询评论
deleteComment	userID：评论者唯一标识符；serviceID：工位/服务唯一标识符	删除评论

3.2.4交互界面实现

（1）页面排版布局

通过 XML 语言布局各窗口页面，其中主窗口为“MainActivity”，是项目启动时的入口，也是项目的主页，代码结构如图 3.11 所示。

```
android:name=".MyApp"
android:allowBackup="true"
android:theme="@style/myTheme" >
<activity
    android:name=".MainActivity"
    android:label="主页" >
    <intent-filter>…
    </intent-filter>
</activity>
```

图 3.11 Android 主页声明

副窗口页面有 12 个页面，完成注册、登录、发布租赁合同、个人主页、查看积分、查看租赁信息、搜索工位、租赁工位、工位或服务评价、查看评级记录、查看普通增值服务和查看高级增值服务。部分代码实现如下图 3.12 所示。

```
<activity
    android:name=".SigninActivity"
    android:label="注册"
    android:theme="@style/myTheme" >
</activity>
<activity
    android:name=".Signin2Activity"
    android:label="注册"
    android:theme="@style/myTheme" >
</activity>
<activity
    android:name=".PactActivity"
    android:label="合同"
    android:theme="@style/myTheme" >
</activity>
<activity
    android:name=".MineInfoActivity"
    android:label="我的"
    android:theme="@style/myTheme" >
</activity>
<activity
    android:name=".CreditInfoActivity"
    android:label="信用积分"
    android:theme="@style/myTheme" >
</activity>
<activity
    android:name=".RentInfoActivity"
    android:label="租赁信息"
    android:theme="@style/myTheme" >
</activity>
```

图 3.12 其他窗口页面声明

（2）逻辑处理

使用有限转态机的机制来实现逻辑控制，部分代码如下图 3.13 所示。

```
case CLIENT_USER_QUERY: //查询个人信息
    bc_user_query(input, output);
    break;

/* 工位操作 */
case CLIENT_STATION_APPLY://申请工位
    bc_station_apply(input, output);
    break;

case CLIENT_STATION_REVOKE://撤销工位
    bc_station_revoke(input, output);
    break;

case CLIENT_STATION_RENT://租用工位
    bc_station_rent(input, output);
    break;

case CLIENT_STATION_RETURN://返还工位
    bc_station_return(input, output);
    if (!strcmp(output["result"].asString().c_str(), "success")) //成功返还工位
        bc_trade_stop(input["user_no"].asInt64(), input["trade_no"].asInt64(), true);
    break;

case CLIENT_STATION_QUERY:
    bc_station_query(input, output);
    break;//查询工位

case CLINET_TRADE_QUERY://查询交易
    bc_trade_query(input, output);
    break;

default: break;
```

图 3.13 Android 访问控制处理

（3）发送消息

设置 HTTP POST 请求报头如图 3.14 所示。

```
//HTTP报文头
struct curl_slist* headers = NULL;
headers = curl_slist_append(headers, "authorization: Bearer");
headers = curl_slist_append(headers, "content-type: application/json; charset=utf-8");

//HTTP数据
Json::Value JsonValue;
std::string data;
JsonValue["username"]    = "Jim";
JsonValue["orgname"]     = "org1";
JsonValue["channelName"] = "mychannel";
JsonValue["peers"].append("localhost:7051");
JsonValue["chaincodeName"] = chaincode;
JsonValue["functionName"] = function;
JsonValue["args"] = *args;
// 将数据转为json格式的字符串
data = JsonValue.toStyledString();
```

图 3.14 设置请求报头

向 SDK 服务器发送请求 HTTP POST 请求如图 3.15 所示。

```
if (curl) {
    curl_easy_setopt(curl, CURLOPT_URL, url);                // 设置url地址
    curl_easy_setopt(curl, CURLOPT_POST, 1);                 // post请求
    curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);     // 设置HTTP头
    curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data.c_str());// 设置post内容
    curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, func);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, res);
    curl_easy_setopt(curl, CURLOPT_TIMEOUT, 300);

    curl_res = curl_easy_perform(curl); // 建立发送请求
    if (curl_res != CURLE_OK) // 请求返回结果判断
```

图 3.15 发送 HTTP POST 类型请求

3.2.5性能测试实现

（1）并发机制

使用 go 语言里的并发机制。能使得某个函数独立于其他函数运行。函数创建为 goroutine 时，将被视为一个独立的工作单元。这个单元会被调度到可用的逻辑处理器上执行。goroutine 与操作系统线程之前的映射关系为：

m:n

其中 m>1，n>=1，可由一个操作系统线程映射到多个 goroutine。

并发访问 go 语言实现的关键部分如下图，每隔 0.002 秒发送一个交易请求，即每秒 500 次，go 语言实现如图 3.16 所示。

```
go func() {
    for {
        select {
        // 每0.002秒发送一次请求
        case <-time.After(20000000):
            // Post()为http post类型的请求发送函数，
            go Post()
        }
    }
}()
```

图 3.16 并发请求发送

通过程序设置运行时间来统计成功访问的交易次数和每秒处理交易的平均值，如图 3.17 所示。

```
// 设置程序运行时间
time.Sleep(100 * time.Second)
```

图 3.17 设置程序运行时间

（2）通道共享数据

go 语言中可以使用通道（channel）来发送和接收异步访问的数据。提供了异步进程间同步数据的机制。下图 3.18 中对同步的数据进行声明。

```
// 声明两个共享数据通道，用来统计发送请求和请求成功执行次数，有100缓存空间
var quit chan int = make(chan int, 100)
var send chan int = make(chan int, 100)
```

图 3.18 使用通道同步数据

在通道发送端，在 http 请求获得发送成功的返回值时将一个信号发送进通道，如下图 3.19 所示。

```
// 将整型数据0作为成功访问的信号塞入通道
quit <- 0
```

图 3.19 将异步数据发送至通道

在通道接收端，接收到一个从通道接收端发送过来的数据时设置一个全局变量的值加 1 以统计发送请求次数和交易成功执行的次数，如下图 3.20 所示。

```
go func() {
        for {
                <-send
                j++
                fmt.Printf("发送交易次数：%d 次\n", j)
        }
}()
go func() {
        for {
                <-quit
                i++
                fmt.Printf("交易成功执行次数：%d 次\n", i)
        }
}()
```

图 3.20 发送交易及交易成功次数统计

4 结论与分析

4.1 底层网络部署分析

（1）网络启动。在服务器裸机中模拟区块链网络采用的 docker 容器技术，基于基础镜像可以生成虚拟节点，节点开放虚拟端口与端口主机映射。从而组成分布式去中心化的区块链网络，如图 4.9 所示，分别为容器唯一标识符（CONTAINER ID）、使用的镜像名称和映射端口（IMAGE PORTS）、启动命令名称（COMMAND NAMES）、创建时间（CREATED） 和容器状态（STATUS）如下图 4.1 所示。

```
[root@localhost pbOffice_BC]# docker ps
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STA
TUS                 PORTS                                      NAMES
bf242425dab1        hyperledger/fabric-peer:x86_64-1.0.0   "peer node start"        32 seconds ago      Up
30 seconds          0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp   peer0.org1.example.com
5fe2bf08fc99        hyperledger/fabric-peer:x86_64-1.0.0   "peer node start"        32 seconds ago      Up
30 seconds          0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp   peer0.org2.example.com
992c387a9b94        hyperledger/fabric-peer:x86_64-1.0.0   "peer node start"        32 seconds ago      Up
30 seconds          0.0.0.0:7056->7051/tcp, 0.0.0.0:7058->7053/tcp   peer1.org1.example.com
9918474dfa05        hyperledger/fabric-peer:x86_64-1.0.0   "peer node start"        32 seconds ago      Up
31 seconds          0.0.0.0:8056->7051/tcp, 0.0.0.0:8058->7053/tcp   peer1.org2.example.com
010ebe4f7c97        hyperledger/fabric-ca:x86_64-1.0.0     "sh -c 'fabric-ca-ser"   33 seconds ago      Up
32 seconds          0.0.0.0:8054->7054/tcp                     ca_peerOrg2
a16a30609f0c        hyperledger/fabric-orderer:x86_64-1.0.0   "orderer"                33 seconds ago      Up
32 seconds          0.0.0.0:7050->7050/tcp                     orderer.example.com
a1949b33f5d0        hyperledger/fabric-ca:x86_64-1.0.0     "sh -c 'fabric-ca-ser"   33 seconds ago      Up
32 seconds          0.0.0.0:7054->7054/tcp                     ca_peerOrg1
```

图 4.1 模拟区块链网络节点的启动状态

结果分析：网络中一共有 7 个节点，全部启动正常。具体包括依赖于“fabric-peer”镜像的四个 peer 对等节点；2 个依赖于“fabric-ca”镜像的 ca 证书颁发节点和 1 个依赖于“fabric-orderer”镜像的 orderer 排序节点。以后再生产环境可用增加对等节点 peer 和 orderer 排序节点以增强区块链网络的去中心化程度。

（2）SDK 服务器启动。该服务器通过 gRPC 协议与区块链网络中的节点连接。SDK 服务启动时开放 4000 端口处理用户交互界面 Android app 发送的请求。也可通过 shell 脚本发送测试命令与 SDK 服务器进行交互，完成区块链中的用户注册、创建通道、加入通道、安装链码、实例化链码与调用链码等命令。SDK 服务器启动后显示如下图 4.2 所示，开放 4000 端口监听 HTTP POST 类型请求：

```
[2018-06-18 11:42:52.563] [INFO] SampleWebApp - ********** SERVER STARTED **********
[2018-06-18 11:42:52.563] [INFO] SampleWebApp - **********  http://localhost:4000  **********
```

图 4.2 开启 SDK 服务器

4.2 功能测试分析

4.2.1智能合约

请求发送的 shell 脚本进行智能合约实例化如下图，智能合约安装成功显示状态如图 4.3 所示，3 个合约 register、publicService 和 comment 安装成功，发送的数据格式为符合 json 格式的字符串，返回数据也是 json 格式字符串。

```
curl -s -X POST \
  http://localhost:4001/channels/chaincodes \
  -H "authorization: Bearer" \
  -H "content-type: application/json" \
  -d '{
  "channelName":"mychannel",
  "username":"Jim",
  "orgname":"org1",
  "peers": ["localhost:7051","localhost:8051"],
  "chaincodeName":"register",
  "chaincodeVersion":"v0",
  "functionName":"init",
  "args":[]
}'
echo
echo
```

图 4.3 发送请求体格式

智能合约的功能测试，完成用户登录、注册、发布工位/服务、租用工位/服务和评价等功能，使用 shell 脚本发送 http post 请求。如下图 4.4 所示。

```
注册用户a1002 ////////////////////////////////
传入的6个参数的含义与注册a1001相同，其中第三个参数为2，表示公司或个人
注册用户a1002返回结果为操作状态和交易ID
{"success":true,"message":"e1d60dce9b772cb51da5cbbcbdba69b4d6bf754249ac842def81eeb35c9bee01"}

查询用户a1002 ////////////////////////////////
args传入的1个参数为：用户ID |
返回结果为查询结果为用户a1002的注册信息
{"success":true,"message":{"id":"a1002","password":"123456","role":"2","balance":1000,"crePoin
t":1000,"funcPoint":1000}}

用户a1001发布服务b1001(owner is a1001) ////////////////////////////////
args传入的8个参数为：服务/工位ID | 发布者ID | 类型（1为工位，2为其他服务） | 类型2（服务名称）
 | 价格（float64） |描述字段 | 地点 | 工位容量
{"success":true,"message":"e41f3a0bf45f86220f57e5c6d6e04402fe12392b870409690a972a76de8044e1"}

查询服务b1001 ////////////////////////////////
args传入的1参数为：服务/工位ID
{"success":true,"message":{"id":"b1001","ownerId":"a1001","type":"1","name":"日租办公室","func
tion":"","money":500,"funPoint":0,"description":"面积大，位置好，设备齐全。","palce":"兆润领域
商务广场11楼","capacity":8,"state":"0","top":false,"topPoint":0,"userId":"","comment":null}}

用户a1002申请服务b1001 ////////////////////////////////
args传入的2个参数为：服务/工位ID | 申请者ID
{"success":true,"message":"e945e1c5fe2ad1c6d486e67cfd65f40cdc4e5c014c0f280081eaf2c63b28b31a"}

查询服务b1001 ////////////////////////////////
args传入的1个参数为：服务/工位ID
{"success":true,"message":{"id":"b1001","ownerId":"a1001","type":"1","name":"日租办公室","func
tion":"","money":500,"funPoint":0,"description":"面积大，位置好，设备齐全。","palce":"兆润领域
商务广场11楼","capacity":8,"state":"1","top":false,"topPoint":0,"userId":"","comment":null}}

用户a1002评价服务b001（属于a1001） 评论使得功能积分＋5//////////////
args传入的4个参数为：使用者ID | 服务ID | 评级（整型1~5） | 评论
{"success":true,"message":"45fcc6f43fa869295627a8e4dc17ab9a8fffbb443587ea9ab071efc7469692f1"}
```

图 4.4 项目逻辑功能测试

结果分析：能够完成注册、登录、发布工位/服务、预定工位/服务、评价的功能。未来可基于特定业务场景设计完成其他功能的智能合约。

4.2.2交互界面

Android app 是用户与本毕业设计项目交互的接口，具体页面包括首页、登录注册、发布工位、查看积分变化、多元增值服务、评价等相关功能分别如下图 4.5、4.6、和 4.7 所示。

图 4.5 首页、登录、注册与注册成功状态页面

图 4.6 用户中心、工位发布、积分变化记录与多元增值服务页面

图 4.7 租用工位、评价、评价成功与评价记录页面

测试结果分析：初步完成页面需要的各项功能逻辑，能够与区块链网络交互数据并且在页面间流畅地切换。未来可根据特定业务场景设计特色的前端页面。

4.3 性能测试分析

使用的硬件水平：Linux 内核 centos 7 操作系统，2CPU，4core（每个 CPU 两个 core）。

测试结果增加新的状态，注册新用户性能测试结果与分析,利用测试脚本发送的 http post 类型请求，测试函数分别为增加状态的 regist 函数和修改状态值得 update 函数，返回值值情况如图 4.8 所示。

```
发送交易次数：49991 次
POST请求:创建成功 {"success":true,"message":"8441ccca8a362e813f214123c67ee46b30c5ba77e08c32cff
efeed6d5255328d"}
交易成功执行次数：21124 次
POST请求:创建成功 {"success":false,"message":"Failed to order the transaction. Error code: und
efined"}
发送交易次数：49992 次
发送交易次数：49993 次
POST请求:创建成功 {"success":true,"message":"b7f4a4783a3979fc5614bdc00c3b0875cf6a1b143e501dabf
b9b44fa7498f65b"}
交易成功执行次数：21125 次
发送交易次数：49994 次
POST请求:创建成功 {"success":false,"message":"Failed to order the transaction. Error code: und
efined"}
POST请求:创建成功 {"success":true,"message":"8ce3b95d2d8e5701c4d821072b766be32db18a6e63a24475c
2470ad3a7014741"}
交易成功执行次数：21126 次
发送交易次数：49995 次
发送交易次数：49996 次
POST请求:创建成功 {"success":true,"message":"70138ccce70f62725932ee816e60d2e385a44595815d1c22c
d8f6710f8647e82"}
交易成功执行次数：21127 次
发送交易次数：49997 次
POST请求:创建成功 {"success":true,"message":"c3fd3da3c96a5ef9c76e208e85c4ffdee5e6530e9fc6394e9
963daa37e9a1064"}
交易成功执行次数：21128 次
POST请求:创建成功 {"success":false,"message":"Failed to order the transaction. Error code: und
efined"}
```

图 4.8 性能测试结果

regist 函数和 update 函数统计情况如下表 4.1 所示。

表 4.1 性能测试结果统计

测试函数	时长	间隔	发送数据	成功处理数据	成功率	每秒成功处理交易数
regist	100s	0.002s	49997 条	21128 条	42.25%	211.13tps
update	100s	0.002s	49997 条	22367 条	44.74&	223.67tps

结果分析：测试结果为 200tps~300tps 间。区块链网络的交易速率与中心化系统相比性能方面低很多，原因在于中心化系统交易无需经过各分布式节点的共识，而只需要在后台完成逻辑后将数据写入中心化数据库即可。

而在区块网络中各分布式节点要达成共识以保证数据的同步。客户端发送的数据传达到区块链网络之后大致需要几个流程：

（1）背书节点（Endorser）检查节点身份进行交易提案背书，模拟测试结果。

（2）背书节点将经过背书的交易提案转发给排序节点（Orderer），排序节点将模拟的交易统一排序打包成区块。

（3）对等节点（Peer）将从排序节点获得同步的数据区块，记录在本地的账本中。

其中对获得足够多数背书节点的背书、排序节点统一将交易排序和各对等节点同步数据区块是相比较于中心化系统需要额外消耗的时间。

下一步需要提高交易速率可从硬件设备选用处理能力更强的计算机、提高交易签名算法速度、提高共识算法的速度等方面入手。

5 结论和展望

5.1 工作总结

本设计结合区块链与共享经济的概念，设计了基于区块链技术构建信用评价体系的共吸纳过办公及服务租赁平台。该平台基于区块链的分布式存储与共识、点对点传输、密码学原理与实现了不可篡改、可追溯的去中心化的信任机制。基于这样的一套信任机制构建可信的信用评价体系，引入多元增值服务使得积分更有价值。相比于传统的中心化租赁平台可能出现的信任危机，如中心化平台篡改或删除数据使得自己朝着有利于平台自身的方向发展。本设计的区块链底层架构基于联盟链超级账本 Fabric 1.0 版本，是一个可用于项目测试和商业化应用的区块链底层技术架构，基于此构建基于信用评价体系的共享办公平台

本项目与区块链行业的项目处境相似，正处于应用项目的探索期。区块链底层提供的性能也需要提高才能大规模投入到实际应用场景。区块链项目存在的问题包括：

(1) 区块链项目的交易速度较慢，主要是共识与同步机制的速度花费的时间较长。

(2) 联盟区块链去中心程度需要网络规模保证，但考虑到共识的效率节点不能太多。一般在 20 到 100 个共识节点之间比较合适。

5.2 展望

本项目是基于办公场景的区块链应用项目，可以在共享办公平台及线下众创空间的结合新的技术。以下是本项目可以改进的地方：

(1)引入数字地图服务商，方便外地出行工作者搜寻附近的众创空间的工位。

(2)结合人脸识别技术引入全新的身份认证机制，快速登录。设置分布式服务器，使用类似于 IPFS 星际文件系统用于进行点对点的分布式文件存储。分布式存储人脸照片，人脸特征值存储于分布式区块链系统形成一个不可篡改的数据，用于数据验证。

(3)结合物联网，通过在工位或众创空间内部安装传感器网络检测办公人员的动向、行为、使用工位的时间等，并设定一定的触发事件将关键事件节点发生的数据自动上传到区块链执行合约，完成自动付费等功能。

(4)通过闪电网络来提高系统交易的性能，在一段时间内将交易的执行结果在区块链之外的服务器进行，而在区块链上只存储最终的交易结果。

未来的工作可以针对以上问题展开深入研究。

参考文献

[1] 邵奇峰, 金澈清, 张召, 钱卫宁, 周傲英. 区块链技术:架构及进展[J]. 计算机学报, 2018(5): 0254-4164

[2] Kosba A, Miller A, Shi E, et al. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts[C]//Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, 2016: 839-858.

[3] 袁勇, 王飞跃. 区块链技术发展现状与展望[J]. 自动化学报, 2016, 42(4):481-494.

[4] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. Consulted, 2008.

[5] Matevž Pustišek, Andrej Kos. Approaches to Front-End IoT Application Development for the Ethereum Blockchain[J]. Procedia Computer Science, 2018, 129:410-419.

[6] Collomb A, Sok K. Blockchain/Distributed Ledger Technology (DLT): What Impact on the Financial Sector?[J]. Communications & Strategies, 2016 (103): 93.

[7] Parameswaran M, Susarla A, Whinston A B. P2P networking: an information sharing alternative[J]. Computer, 2001, 34(7): 31-38.

[8] Rogaway P, Shrimpton T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance[C]//International workshop on fast software encryption. Springer, Berlin, Heidelberg, 2004: 371-388.

[9] Preneel B, Bosselaers A, Dobbertin H. The cryptographic hash function RIPEMD-160[J]. Cryptobytes, 1997.

[10] Daniel Larimer. Transactions as Proof-of-Stake[J]. 28, November, 2013

[11] Seymour K, Nakada H, Matsuoka S, et al. Overview of GridRPC: A remote procedure call API for grid computing[C]//International Workshop on Grid Computing. Springer, Berlin, Heidelberg, 2002: 274-278.

[12] Group S F E C. SEC1. Elliptic Curve Cryptography[J]. 2000.

[13] 谢辉, 王健. 区块链技术及其应用研究[J]. 信息网络安全, 2016(9):192-195.

[14] Fischer M J, Lynch N A, Paterson M S. Impossibility of distributed consensus with one faulty process[J]. Acm Tocs, 1985, 32(2):374-382.

[15] Lamport L, Shostak R, Pease M. The Byzantine Generals Problem[M]. ACM, 1982.

[16] Castro M, Liskov B. Practical Byzantine fault tolerance[C]// ACM, 1999:173-186.

装 订 线

谢 辞

大学四年的时间转瞬即逝，到即将从母校毕业的那一刻才醒悟过来不到 1 个月就要离开母校了。回想在大学的 4 年，有时候自己做过的选择并不是那么的合理。但也在选择中一步步确立了自己的发展方向和未来努力的目标。大学是一个思想自由、文化繁荣之地；大学是一个名师辈出、装载无限知识与力量源泉之地，很幸运当初选择了同济大学，母校选择了我，让我在这里开心同步不算虚度光阴地度过了 4 年的时光。回顾这四年，太多人给予了我帮助。首先，是我的父母，他们一直以来都非常坚定地支持我们我的选择，在我身处异乡的时候给了我非常大的精神鼓励。

其次，是我的本科导师马小峰老师。我还记得马老师第一次在 F 楼 409 实验室给我们这一届的当时大二的本科生介绍区块链技术的原理与应用场景的时候。从那时起便对区块链技术充满好奇，开始渐渐探索其中的奥秘。而到现在 2018 年区块链项目正在各行业不断地进行尝试与探索，并有越来越多的应用项目落地，也让我深深佩服马老师长远的目光。师恩如灯，给与学生光明与希望而不求索取。马老师丰富的专业知识、深入行业的实践探索精神、丰富的交叉学科知识、乐观开朗的态度、指引学生同时又充分给予其自由与想象空间的精神都让我深深地佩服。在大学认识马老师的这三年对我未来的为人处世态度、做事原则潜移默化的影响更让我深深地感激。

然后，是 F 楼 409 实验室的学长学姐们。他们给与了我区块链技术理论和实践操作上的指导，让我少走了很多的弯路才让我更快的进步并对区块链技术有了深层次的理解。

感谢自动化专业的任课老师以及大学以来各位教育过我的老师们，每堂课上我都学得不少知识。无论以后进入哪个行业，我都会怀着老师们传授的知识穷则独善其身，达则兼济天下。

感谢大学四年来的同窗，尤其是为同学服务，以身作则的班干部们；感谢我大学以来认识的 6 位舍友，给予我欢乐，与我交流知识，思想碰撞。

完成这篇毕业设计之后，我的大学生涯即将告一段落，新的征程已经开始。在我以后的日子一定克己复礼，继承同济人的优良作风，不忘母校的照顾与陪伴，铭记“同心同德同舟楫，济人济事济天下”的校训及其深刻含义。

装 订 线