如何搭建区块链应用

主讲人: 银链科技CEO 申屠青春

2017.05.19

个人简介

- 金链盟常务副秘书长；
- 深圳金融标准委员会委员；
- 深圳大学博士、高级工程师；
- 5项发明专利授权，获2012年深圳市发明专利奖；
- 15年信息安全从业经验；
- 15年技术管理经验，管理技术团队120人；
- 2012年创办银链科技；
- 2013年开始研究区块链技术
- 2014年推出区块链隐私保护项目，发表20多篇区块链技术文章。多次参加金融论坛并作区块链主题演讲。
- 2016年转向金融行业

一、区块链应用案例

跨境支付

- 阿布扎比国家银行正式宣布推出了实时区块链跨境支付

2017年2月，阿布扎比国家银行（NBAD）已经宣布通过整合瑞波（Ripple）区块链技术成功推出了实时跨境支付。 由此，中东汇款通道上的客户将会因此直接受益。阿布扎比是阿拉伯联合酋长国中最大的酋长国，而NBAD是该国最大的银行。

- 澳新银行与富国银行联合测试实时跨境支付区块链原型，提升银行清算结算效率

澳新银行和富国银行公开表示，区块链技术将会增加流动性和降低跨境支付的风险和成本。该项目创建并测试了一种分布式账本，用于在两家银行之间核对和结算付款。

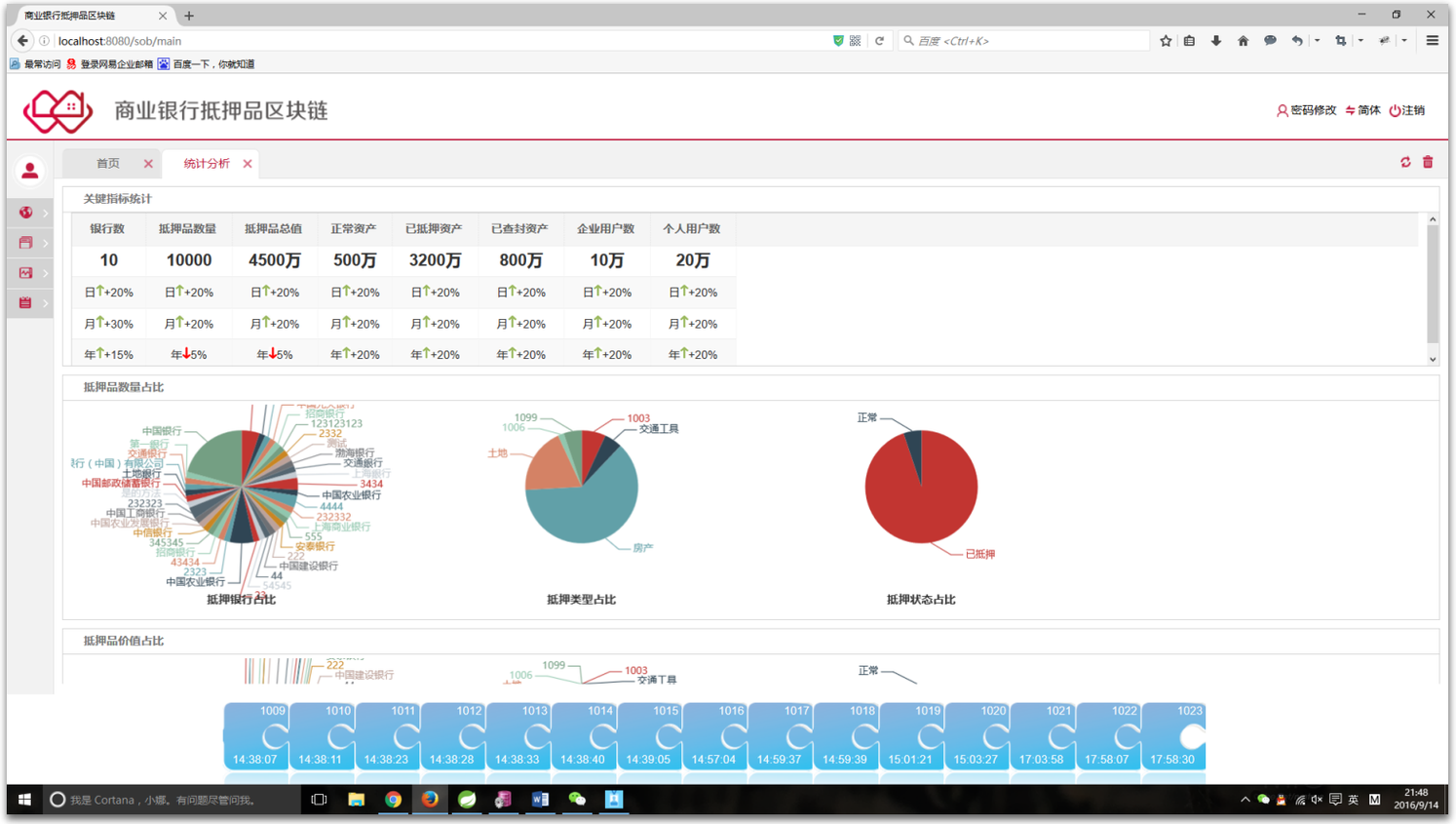
电子票据

- 中国银联建设基于区块链的可信电子凭证系统

中国银联自2011年起开启了电子票据的研究和创新，研究了多种技术架构体系，2016年通过引入先进的区块链技术，建立去中心化的电子票据系统，企业和用户通过任意节点都可以加入银联建立的区块链网络，从而有效降低金融“信用”的建立成本，解决了凭证电子化带来的信任问题，构建了国内首个基于区块链的可信电子凭证系统

商业银行抵押品

- 宝生村镇银行发布商业银行抵押品区块链



2016年6月，宝生村镇银行联合金融区块链合作联盟（深圳）发布商业银行抵押品区块链，并免费提供给金链盟成员单位使用。

贸易金融 /供应链金融

- 法国兴业银行、荷兰银行ING的Easy TradING Connect成功使用区块链技术完成了第一笔大型石油贸易

2017年1月，法国兴业银行、荷兰银行ING的Easy TradING Connect和能源巨头摩科瑞（Mercuria），共同宣布成功使用区块链技术完成了第一笔大型石油贸易，实现了石油贸易数字化的重大突破

银行间清算结算

- 微众银行：银行间联合贷款的清算结算业务

微众银行与华瑞银行联合开发了一套区块链应用系统，用于两家银行微粒贷联合贷款的结算、清算业务，该系统已于2016年9月投入试验运行。

支付和结算

- 瑞银联合世界四家最大银行开发基于区块链的交易结算系统，该项目于2016年8月公布新型数字货币UBS，用于支付结算。

瑞银采用区块链技术开发了一种新型数字货币USC（Utility Settlement Coin），并且与央行发行的主要货币有同等价值，比如欧元或美元，而不同于比特币等去中心化数字货币。

移动数字汇票

- 2017年1月3日浙商银行基于区块链技术的移动数字汇票产品正式上线并完成了首笔交易

区别于传统纸质与电子汇票，移动汇票通过采用区块链技术，将以数字资产的方式进行存储、交易，在区块链系统内流通，不易丢失、无法篡改，具有更强的安全性和不可抵赖性。此外，纸质汇票电子化，有效解决了防伪、流通、遗失等问题。

银行间贸易

- 澳大利亚联邦银行，富国银行和博瑞棉花（Brighann Cotton）完成了第一笔银行间贸易交易

这笔交易结合了区块链技术，智能合约和物联网,这笔交易所涉及的信用证是通过一种存储在私有分布式账本上的数字智能合约来执行的——使用了Skuchain的Brackets系统。

监管与简化流程

- 邮储银行将区块链技术成功应用于真实的生产环境。免去了重复信用校验的过程。

采用超级账本架构(Hyperledger Fabric)将区块链技术成功应用于真实的生产环境。免去了重复信用校验的过程，能将原有业务环节缩短60%至80%，令信用交换更为高效。

监管与简化流程

- 纽约梅隆银行用区块链技术备份银行交易

梅隆银行(BNY) 开发的测试版区块链系统可用于创建银行经纪业务交易的备份记录。这个名为‘BDS 360’ 的新系统，其目的是作为银行现有交易记录系统的一个“备胎”，既当银行第一层交易记录系统变得不可用时，该系统就顶替而上。

互助保险

- 国内首个区块链保险卡单上线于2016年7月29日

阳光保险推出航空意外险，这种产品60元购买一份，可使用20人次，每次可获得高达200万元的航空意外保障，相当于每次花费3元即可获得200万元的保障。

资产证券化

- 纳斯达克（Nasdaq）交易所运营商正在推出一个新的基于区块链的广告合同市场。

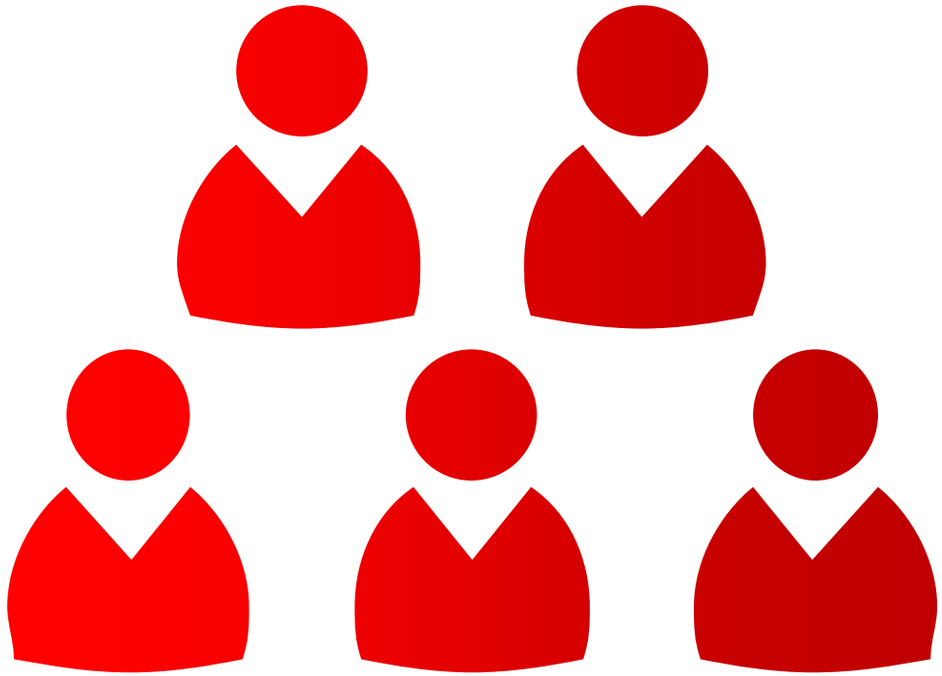
纽约互动广告交易所（NYIAX）已经把交易所上线定位为一种透明的买卖双方进行数字媒体交易的方式，旨在通过将广告合同证券化来简化价值320亿美元的媒体广告市场。该公司表示他们将会利用部署在云端的专用技术，将区块链作为市场交易“核心账本”。Nasdaq正在为这个广告交易所提供所需要的技术。

二、区块链应用开发案例

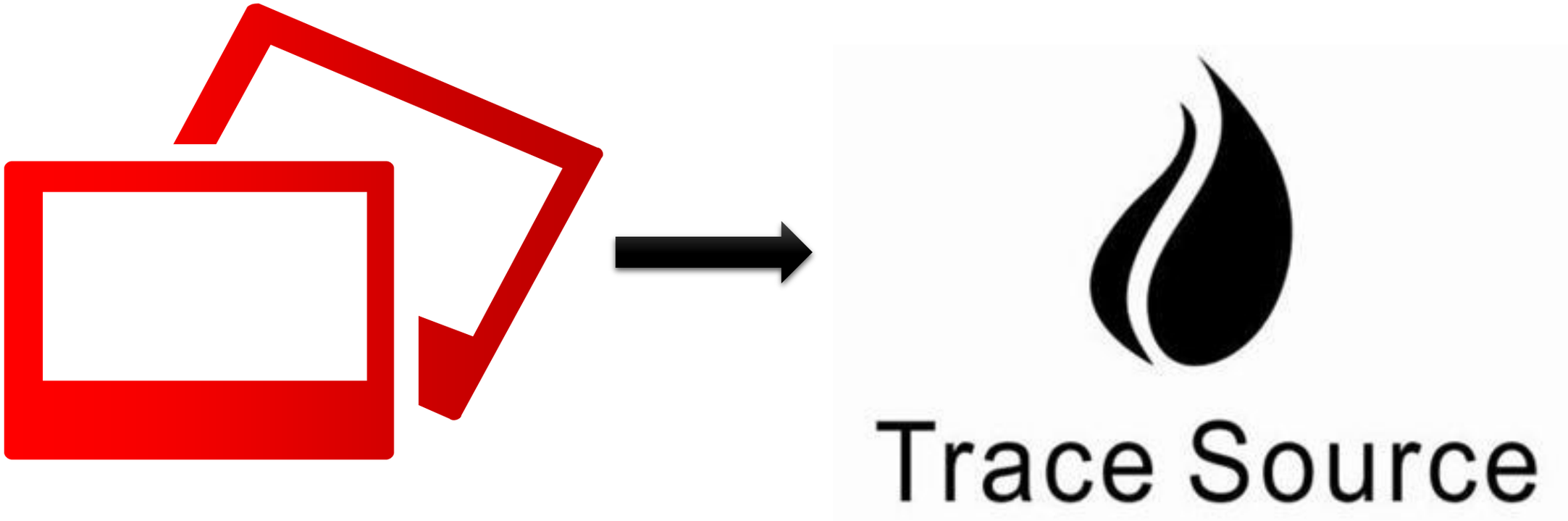

应用场景
区块链
开发工具
开发过程

应用场景的选择

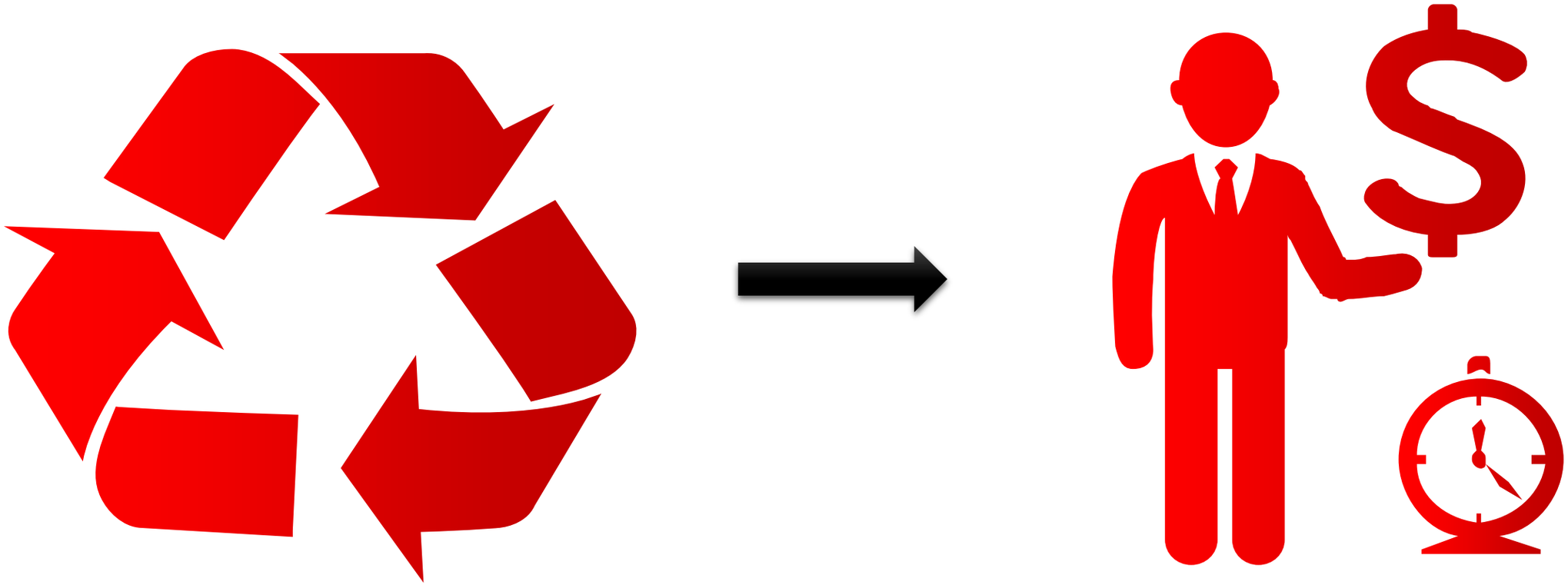
1. 多方协作下账本一致以建立信任



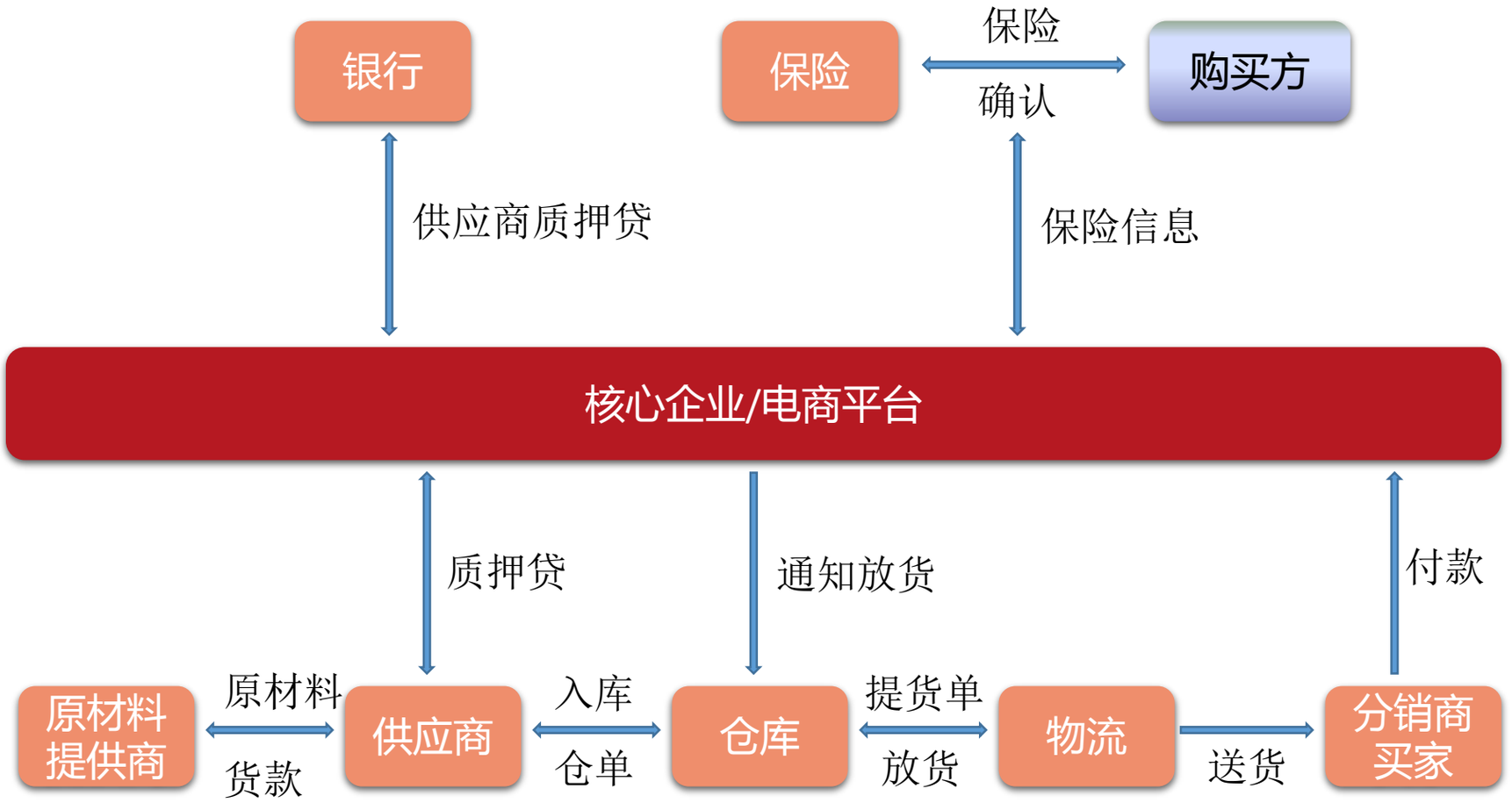
2. 单据电子化方便流转和溯源



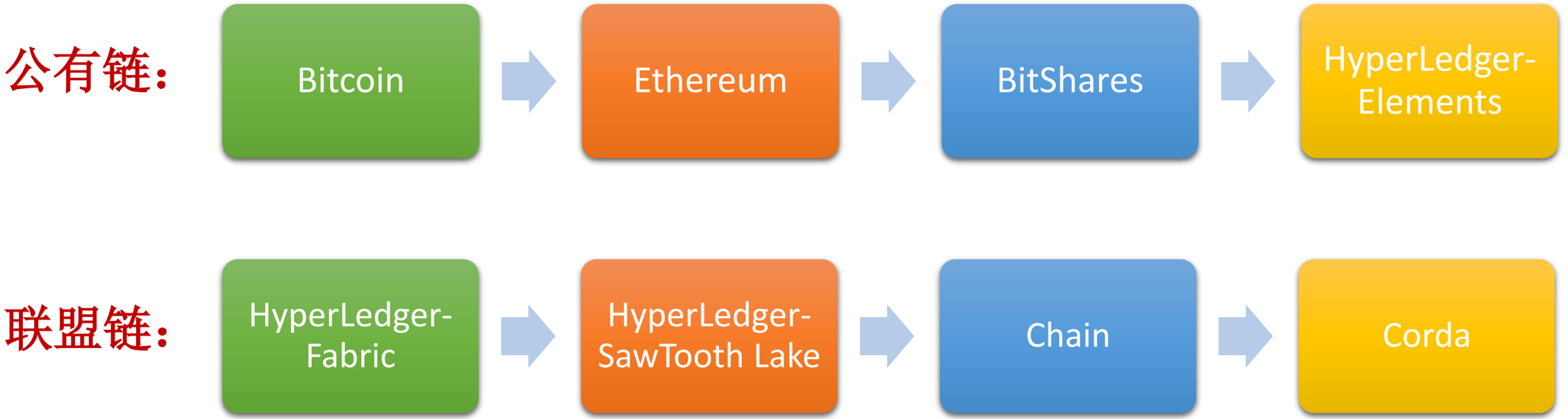
3.流程优化节省人力和时间



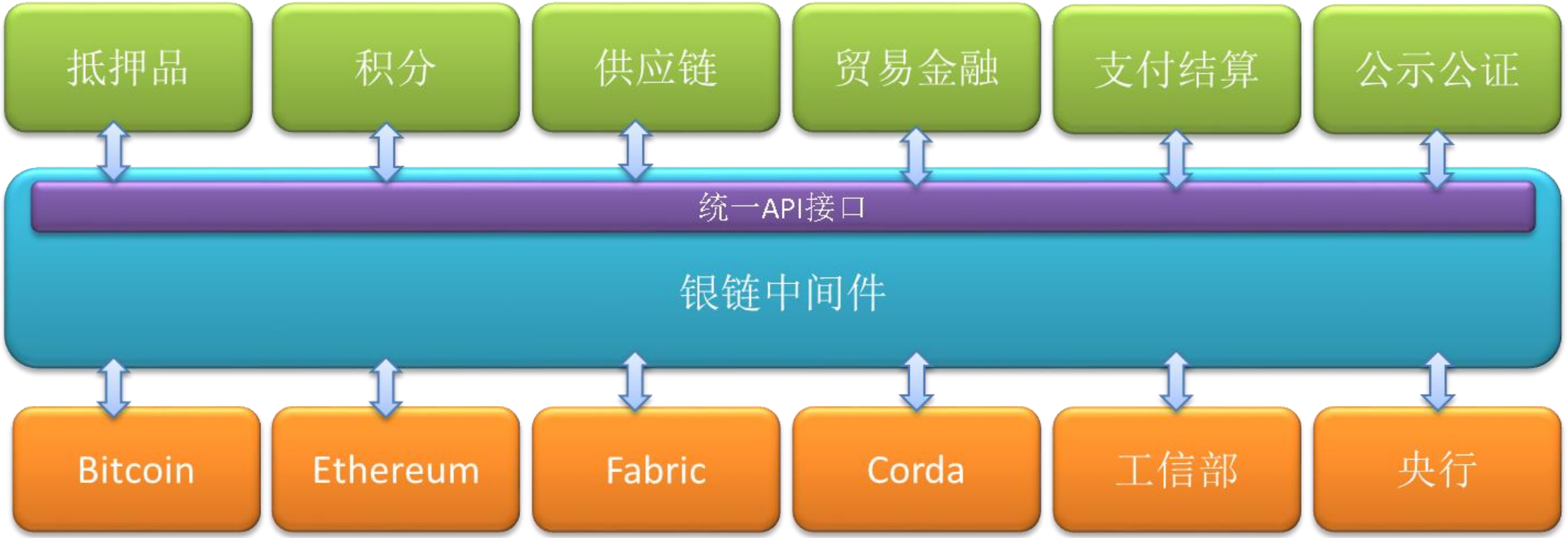
仓单质押融资-业务流程


银行
保险
保险
确认
购买方
供应商质押贷
保险信息
核心企业/电商平台
质押贷
通知放货
付款
原材料
提供商
原材料
货款
供应商
入库
仓单
仓库
提货单
放货
物流
送货
分销商
买家

区块链的选择

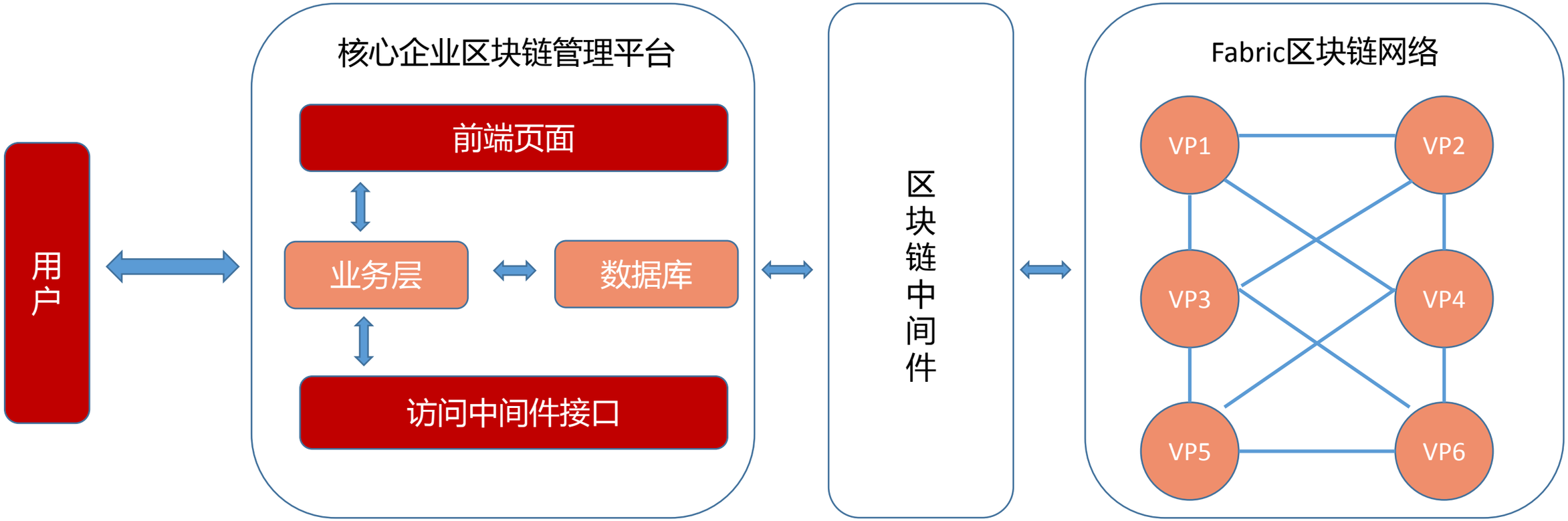

公有链：
Bitcoin
Ethereum
BitShares
HyperLedger-Elements
联盟链：
HyperLedger-Fabric
HyperLedger-SawTooth Lake
Chain
Corda

开发工具的选择-区块链中间件BMWare



系统架构

系统架构


用户
核心企业区块链管理平台
前端页面
业务层
数据库
访问中间件接口
区块链中间件
Fabric区块链网络
VP1
VP2
VP3
VP4
VP5
VP6

搭建Fabric节点

安装节点程序

- 准备Ubuntu14.04系统虚拟机
- 安装Docker（curl -sSL https://get.daocloud.io/docker | sh）
- 安装Rocksdb4.1（https://github.com/facebook/rocksdb/releases）
- 安装Go1.6.2
- 设置Gopath环境变量，GOPATH=“/opt/gopath”
- 下载Fabric代码到Gopath目录（https://github.com/hyperledger/fabric）
- 编译代码 make peer

启动程序

- 在虚拟机窗口1启动CA程序，./membersrvc
- 在虚拟机窗口2启动peer程序，./peer node start

编写智能合约

定义智能合约

```
// SimpleChaincode example simple Chaincode implementation
type SimpleChaincode struct {
}
```

实现智能合约3个接口，分别是：初始化、调用、查询

```
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error)
func (t *SimpleChaincode) Invoke(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error)
func (t *SimpleChaincode) Query(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error)
```

接口参数说明：

- stub，智能合约实例，用它能调用Fabric提供的对外智合约接口
- function，智能合约中的自定义函数
- args，传给自定义函数的参数

举例：

```
func (t *RegisterChaincode) Invoke(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error) {
    if function == "issueAsset" {
        return t.issueAsset(stub, args)
    }
func (t *RegisterChaincode) issueAsset(stub shim.ChaincodeStubInterface, args []string) ([]byte, error)
```

智能合约交互

部署智能合约

POST host:port/chaincode

```
{
  "jsonrpc": "2.0",
  "method": "deploy",
  "params": {
    "type": "GOLANG",
    "chaincodeID":{
        "path":"github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02"
    },
    "ctorMsg": {
        "function":"init",
        "args":["a", "100", "b", "200"]
    }
  },
  "id": "1"
}
```

调用智能合约

POST host:port/chaincode

```
{
  "jsonrpc": "2.0",
  "method": "invoke",
  "params": {
    "type": "GOLANG",
    "chaincodeID":{
      "name":"52b0d803fc395b5e34d8d4a7cd69fb6aa00099b8fabed83504ac1c5d61a425
              aca5b3ad3bf96643ea4fdaac132c417c37b00f88fa800de7ece387d008a76d3586"
    },
    "ctorMsg": {
        "function":"invoke",
        "args":["a", "b", "100"]
    }
  },
  "id": "3"
}
```

查询智能合约

POST host:port/chaincode

```
{
  "jsonrpc": "2.0",
  "method": "query",
  "params": {
    "type": "GOLANG",
    "chaincodeID":{
      "name":"52b0d803fc395b5e34d8d4a7cd69fb6aa00099b8fabed83504ac1c5d61a425
              aca5b3ad3bf96643ea4fdaac132c417c37b00f88fa800de7ece387d008a76d3586"
    },
    "ctorMsg": {
        "function":"query",
        "args":["a"]
    },
    "secureContext": "lukas"
  },
  "id": "5"
}
```

部署结果

```
{
        "jsonrpc": "2.0",
        "result": {
            "status": "OK",
            "message": "52b0d803fc395b5e34d8d4a7cd69fb6aa00099b8fabed83504ac1c5d61a425
                        aca5b3ad3bf96643ea4fdaac132c417c37b00f88fa800de7ece387d008a76d3586"
        },
        "id": 1
}
```

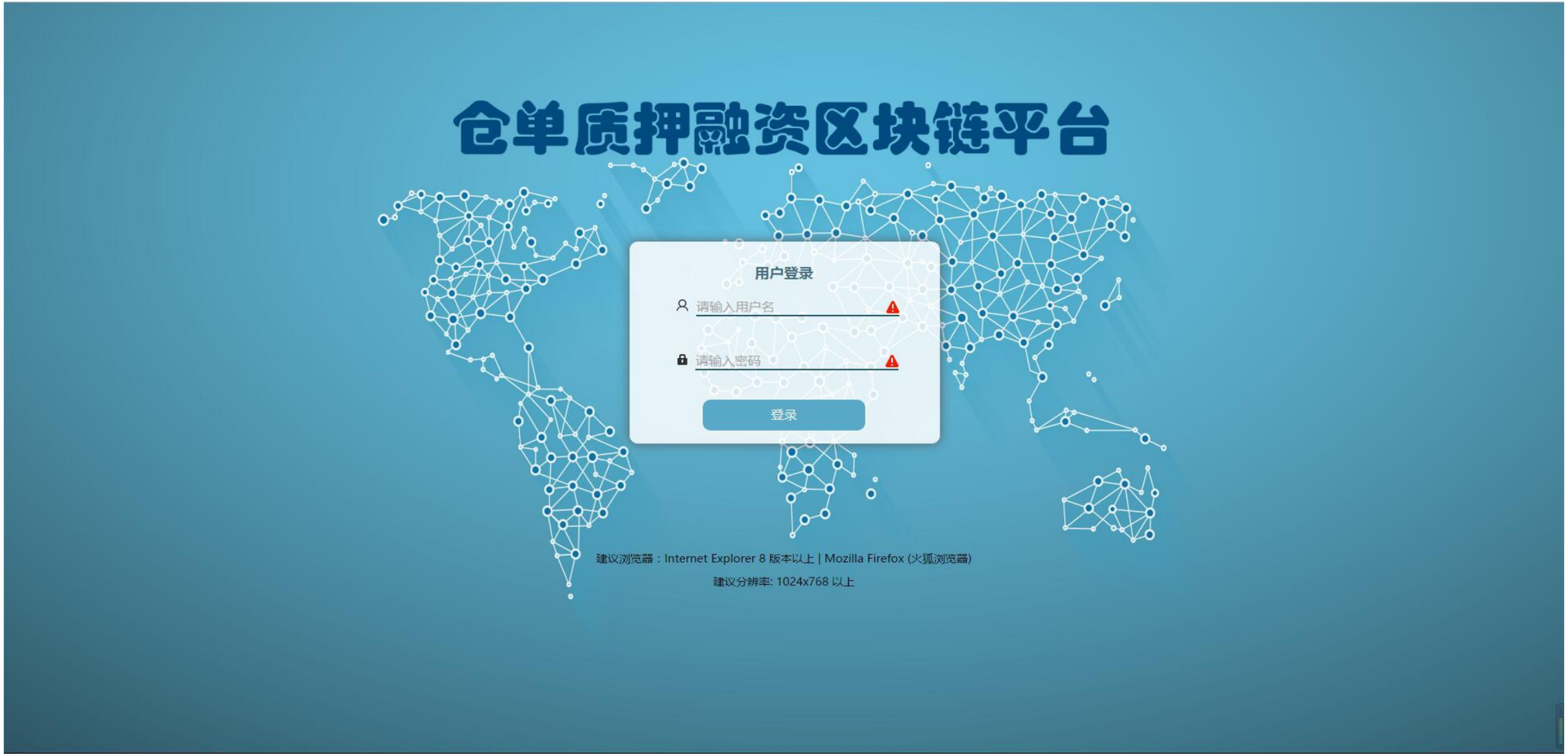
调用结果

```
{
    "jsonrpc": "2.0",
    "result": {
        "status": "OK",
        "message": "5a4540e5-902b-422d-a6ab-e70ab36a2e6d"
    },
    "id": 3
}
```

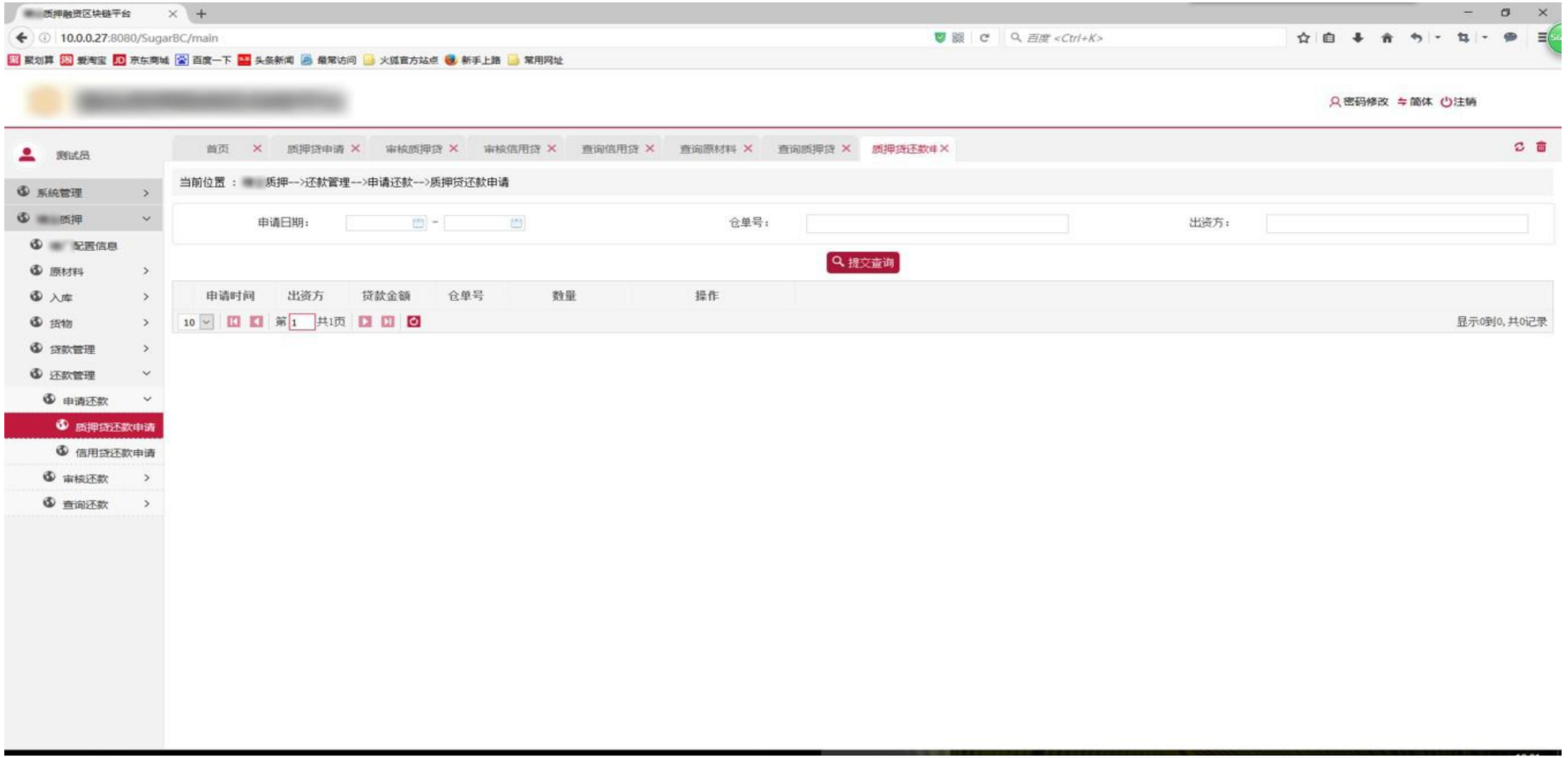
查询结果

```
{
    "jsonrpc": "2.0",
    "result": {
        "status": "OK",
        "message": "90"
    },
    "id": 5
}
```

实现前端界面(JAVA SDK)



实现前端界面(JAVA SDK)



欢迎交流!