中山大学数据科学与计算机学院本科生实验报告

2019年秋季学期

课程名称	区块链原理与技术	任课教师	郑子彬
年级	17	专业（方向）	软件工程
学号	16327109	姓名	谢昆成
Email	xiekch@qq.com	github项目	Supply-Chain-Finance

项目背景

某车企(宝马)因为其造车技术特别牛,消费者口碑好,所以其在同行业中占据绝对优势地位。因此,在金融机构(银行)对该车企的信用评级将很高,认为他有很大的风险承担的能力。在某次交易中,该车企从轮胎公司购买了一批轮胎,但由于资金暂时短缺向轮胎公司签订了 1000 万的应收账款单据,承诺 1 年后归还轮胎公司 1000 万。这个过程可以拉上金融机构例如银行来对这笔交易作见证,确认这笔交易的真实性。在接下里的几个月里,轮胎公司因为资金短缺需要融资,这个时候它可以凭借跟某车企签订的应收账款单据向金融结构借款,金融机构认可该车企(核心企业)的还款能力,因此愿意借款给轮胎公司。但是,这样的信任关系并不会往下游传递。在某个交易中,轮胎公司从轮毂公司购买了一批轮毂,但由于租金暂时短缺向轮胎公司签订了 500 万的应收账款单据,承诺 1 年后归还轮胎公司 500万。当轮毂公司想利用这个应收账款单据向金融机构借款融资的时候,金融机构因为不认可轮胎公司的还款能力,需要对轮胎公司进行详细的信用分析以评估其还款能力同时验证应收账款单据的真实性,才能决定是否借款给轮毂公司。这个过程将增加很多经济成本,而这个问题主要是由于该车企的信用无法在整个供应链中传递以及交易信息不透明化所导致的。

本项目将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

方案设计

我们需要实现如下功能:

功能一:实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

功能二:实现应收账款的转让上链,轮胎公司从轮毂公司购买一笔轮毂,便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。

功能三:利用应收账款向银行融资上链,供应链上所有可以利用应收账款单据向银行申请融资。

功能四:应收账款支付结算上链,应收账款单据到期时核心企业向下游企业支付相应的欠款。

存储设计

```
    uint receiptsSize;
    mapping(address => Company) public companies;
    mapping(uint => Receipt) public receipts;
```

通过mapping存储公司信息。

在合约内通过mapping将合约id（uint）映射到应收账款。每个合约id唯一。

数据结构

定义结构体应收账款Receipt

```
    struct Receipt {
        address from;
        address to;
        uint amount;
        bool isValid;
    }
```

rate 表示信用评级。

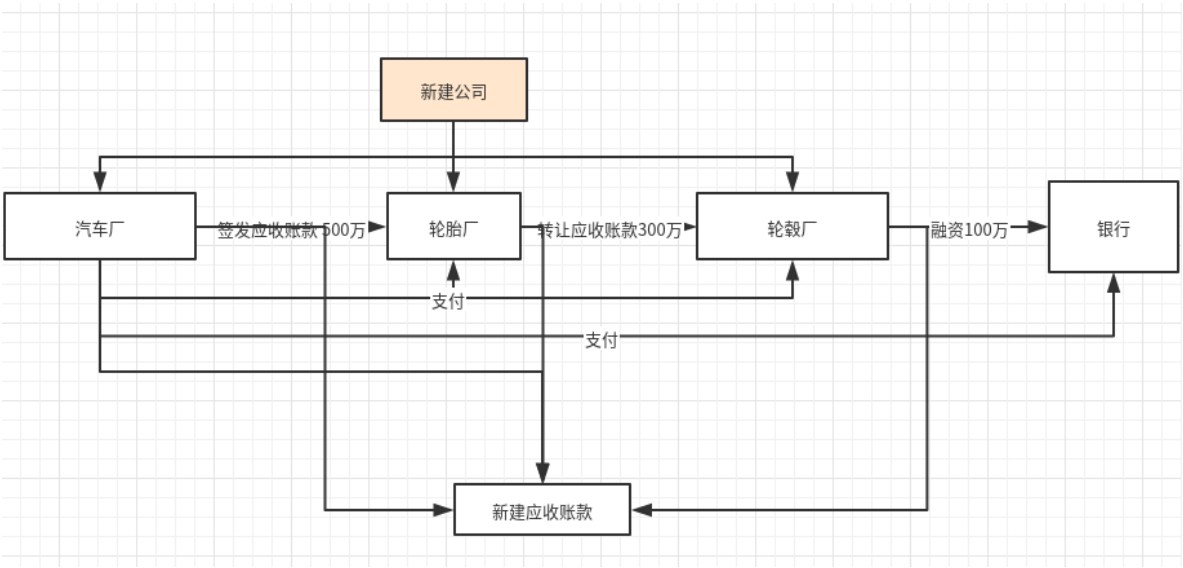
Receipt[] public receipts`存储所有的应收账款。

定义结构体Company

```
        struct Company {
        string name;
        uint balance;
        uint rate;
        uint toPay;
        uint toReceive;
        bool isBank;
        bool isValid;
    }
```

balance 为结余，rate为信用评级，toPay 表示应支付，toReceive表示应收款。

数据流图



每次签发应收账款、转让应收账款、融资和支付都会新建一笔应收账款。从而实现应收账款可拆分、可溯源。

核心功能介绍

- 注册登陆

本项目是web应用，用户界面友好，操作方便简单。打开服务器并访问便可操作相应功能，通过对账户的管理实现权限控制和认证。

- 新建企业

```
    function newCompany(address com, string memory name, uint
balance, uint rate) public{
        if(companies[com].isValid == true){
            emit resultEvent(0,0);
        }
        companies[com] = Company(name, balance, rate, 0, 0,
false, true);
        emit resultEvent(1,0);
    }
```

- 实现采购商品—签发应收账款 交易上链。例如车企从轮胎公司购买一批轮胎并签订应收账款单据。

创建deal的调用者到`to`的应收账款。返回应收账款的id。

```
        // send `to` a reciept after a deal
    function deal(address from, address to, uint amount) public
returns (uint id) {
        if(companies[to].isValid == false){
            emit resultEvent(0,0);
        }

        if(companies[from].isValid == false){
            emit resultEvent(0,0);
        }
        companies[from].toPay += amount;
        companies[to].toReceive += amount;
        receiptsSize += 1;
        id = receiptsSize;
        receipts[id] = Receipt(from, to, amount, true);
        emit resultEvent(1,id);
    }
```

- 实现应收账款的转让上链，轮胎公司从轮毂公司购买一笔轮毂，便将于车企的应收账款单据部分转让给轮毂公司。轮毂公司可以利用这个新的单据去融资或者要求车企到期时归还钱款。

通过receiptId查出合法的应收账款。将原应收账款的amount减去转让的amount，创建新的应收账款，from为原应收账款的from。这样就将上层企业的应收账款分割开来，实现了信用传递。

```
        // transfer account receivable from `from` to `to` with
the `receiptId`
    function transfer(address from, address to, uint receiptId,
uint amount) public returns (uint id) {
        if(companies[to].isValid == false){
            emit resultEvent(0,0);
        }

        if(companies[from].isValid == false){
            emit resultEvent(0,0);
        }

        if(receiptId > receiptsSize){
            emit resultEvent(0,0);
        }

        if(receipts[receiptId].to != from){
            emit resultEvent(0,0);
        }

        if(receipts[receiptId].amount < amount){
            emit resultEvent(0,0);
        }
        address upper = receipts[receiptId].from;
        receipts[receiptId].amount -= amount;

        companies[from].toReceive -= amount;
        companies[to].toReceive += amount;
        receiptsSize += 1;
        id = receiptsSize;
        receipts[id] = Receipt(upper, to, amount, true);
        emit resultEvent(1,id);
    }
```

- 利用应收账款向银行融资上链，供应链上所有可以利用应收账款单据向银行申请融资。

银行判断应收账款的from的信用评级。如果大于一个等级，则同意融资。

```
    // finace from bank
    function financing(address to, address bank, uint receiptId,
uint amount)public returns(bool success, uint id) {
```

```
        if(companies[to].isValid == false){
            emit resultEvent(0,0);
        }

        if(companies[bank].isValid == false){
            emit resultEvent(0,0);
        }

        if(receiptId > receiptsSize){
            emit resultEvent(0,0);
        }

        if(receipts[receiptId].to != to){
            emit resultEvent(0,0);
        }

        if(receipts[receiptId].amount < amount){
            emit resultEvent(0,0);
        }
        // address to = msg.sender;
        address from = receipts[receiptId].from;
        companies[to].toReceive -= amount;
        companies[bank].toReceive += amount;

        receipts[receiptId].amount -= amount;
        companies[bank].balance -= amount;
        companies[to].balance += amount;
        receiptsSize += 1;
        id = receiptsSize;
        // a new receipt to bank
        receipts[id] = Receipt(from, bank, amount, true);
        success = true;
        emit resultEvent(1,id);
    }
```

- 应收账款支付结算上链，应收账款单据到期时核心企业向下游企业支付相应的欠款。

核心企业调用pay，向下游企业或者银行支付欠款。将原有应收账款的amount置0.

```
        // pay a receipt
    function pay(address from, uint receiptId)public{
        address to = receipts[receiptId].to;
        require(from == receipts[receiptId].from,"");
```

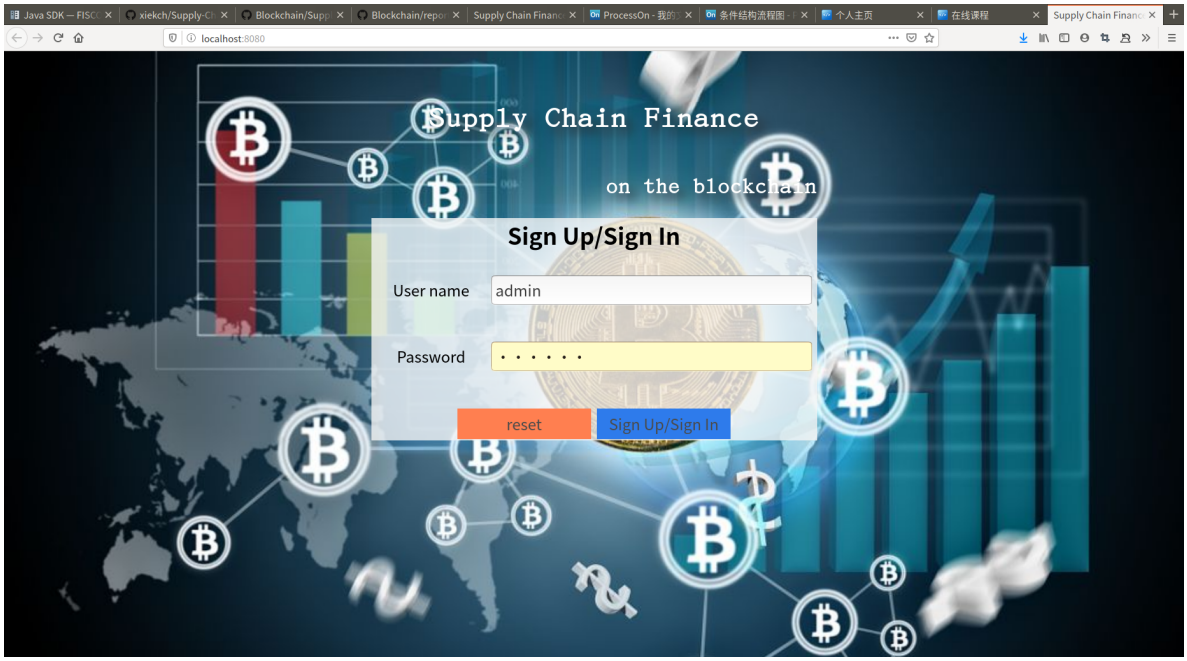
```
        require(companies[from].balance >=
receipts[receiptId].amount,"");
        if(companies[to].isValid == false){
            emit resultEvent(0,0);
        }

        if(companies[from].isValid == false){
            emit resultEvent(0,0);
        }

        if(receiptId > receiptsSize){
            emit resultEvent(0,0);
        }
        companies[from].balance -= receipts[receiptId].amount;
        companies[to].balance += receipts[receiptId].amount;
        receipts[receiptId].amount = 0;
        companies[from].toPay -= receipts[receiptId].amount;
        companies[to].toReceive -= receipts[receiptId].amount;
        emit resultEvent(1,0);
    }
```

功能测试

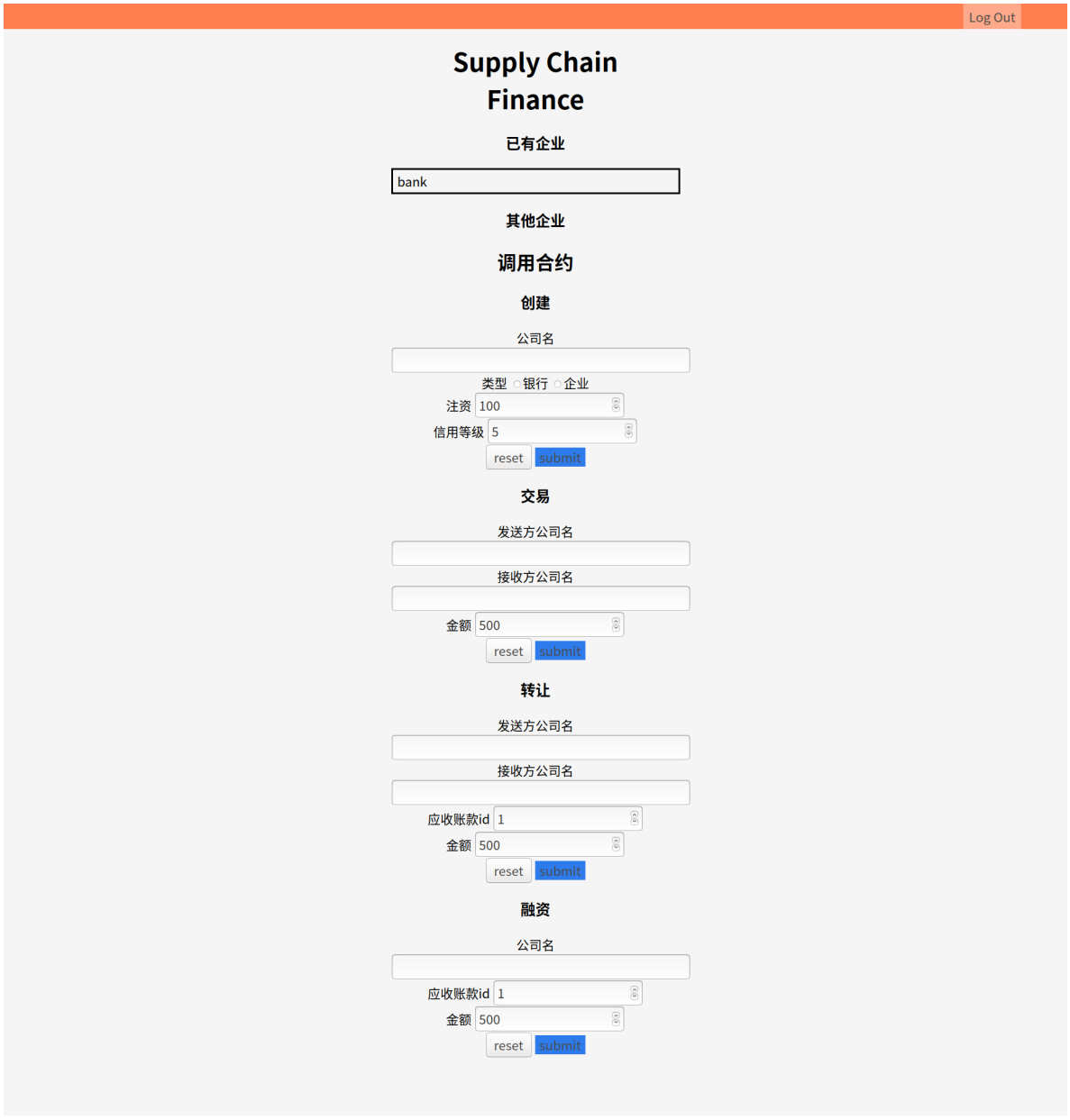
功能测试视频

界面展示

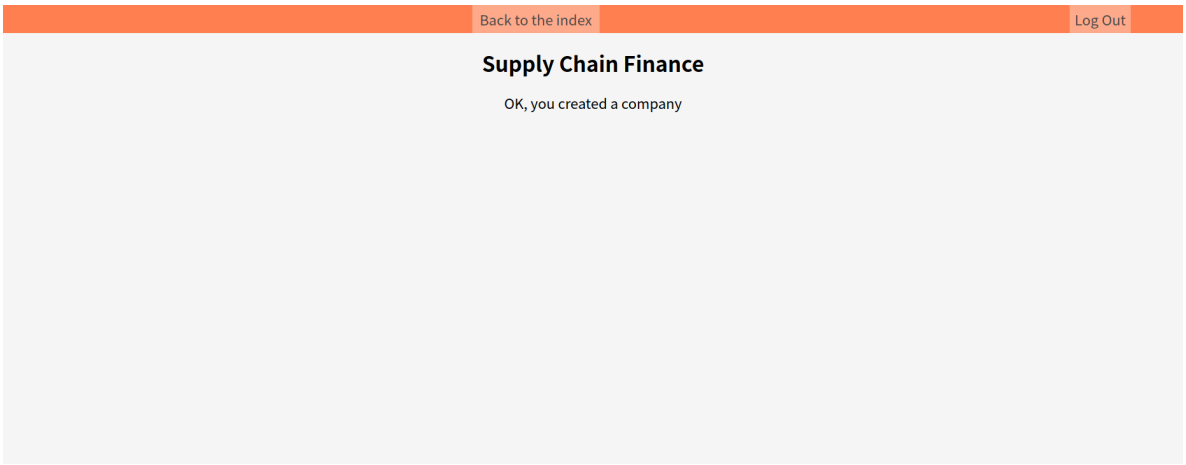
登陆注册页面，美观简洁


Supply Chain Finance
on the blockchain
Sign Up/Sign In
User name
admin
Password
reset
Sign Up/Sign In

主页，操作方便


Log Out
Supply Chain Finance
已有企业
bank
其他企业
调用合约
创建
公司名
类型 银行 企业
注资 100
信用等级 5
reset
submit
交易
发送方公司名
接收方公司名
金额 500
reset
submit
转让
发送方公司名
接收方公司名
应收账款id 1
金额 500
reset
submit
融资
公司名
应收账款id 1
金额 500
reset
submit

调用合约结果反馈页面，清晰明了



源代码说明

本项目是一个web应用。基于java 的Springboot 开发，gradle构建工具。

根目录：

```
nodes/        链端
server/       服务器
constracts/   合约solidity源代码
assets/       markdown文档使用的静态资源
report/       报告文档
.gitignore
build_chain.sh  建链脚本
LICENSE       MIT协议
readme.md
```

合约

对合约的介绍见上核心功能介绍中关于合约的部分。

Springboot

构建springboot项目，可在Spring initializer中创建。

springboot采用mvc架构，目录结构如下：

```
src/main/java/com/xiekch/server/  存放源码
    autoconfigure/            SDK配置
    constants/                使用的常数
    controller/               前端控制器，负责页面访问控制
```

```
    domain/                  模型model
    service/                 业务类代码
    solidity/                合约编译成的java类
    ServerApplication.java   主类
src/main/resources/          资源
    static/                  存放静态文件，比如css、js、image
    templetes/               存放静态模板页面html
    application.yml          区块链节点配置文件
    ca.crt                   区块链节点证书
    sdk.crt                  区块链节点证书
    sdk.key                  区块链节点密钥
```

对合约调用的处理主要由`src/main/java/com/xiekch/server/service/ContractsService.java`类处理。

controller把前端的请求发送给service，由各service进行处理。MVC架构用一种业务逻辑、数据、界面显示分离的方法组织代码，实现了M和V的代码分离。

配置SDK

对SDK的配置主要在`application.yml`文件和`src/main/java/com/xiekch/server/autoconfigure/`下的配置类。

FISCO BCOS作为联盟链，其SDK连接区块链节点需要通过证书(ca.crt、sdk.crt)和私钥(sdk.key)进行双向认证。因此需要将节点所在目录`nodes/${ip}/sdk`下的`ca.crt`、`sdk.crt`和`sdk.key`文件拷贝到项目的资源目录，供SDK与节点建立连接时使用。

本Spring Boot项目中关于`application.yml`的配置如下所示。

```
encrypt-type: # 0：普通， 1：国密
 encrypt-type: 0

group-channel-connections-config:
  caCert: classpath:ca.crt
  sslCert: classpath:sdk.crt
  sslKey: classpath:sdk.key
  all-channel-connections:
  - group-id: 1  # 群组ID
    connections-str:
                    - 127.0.0.1:20200  # 节点，
listen_ip:channel_listen_port
                    - 127.0.0.1:20201
                    - 127.0.0.1:20202
```

```
                    - 127.0.0.1:20203

channel-service:
  group-id: 1 # sdk实际连接的群组
  agency-name: agency # 机构名称
```

详细说明:

- encryptType: 国密算法开关(默认为0)
  - 0: 不使用国密算法发交易
  - 1: 使用国密算法发交易(开启国密功能，需要连接的区块链节点是国密节点，搭建国密版FISCO BCOS区块链参考这里)
- groupChannelConnectionsConfig:
  - 配置待连接的群组，可以配置一个或多个群组，每个群组需要配置群组ID
  - 每个群组可以配置一个或多个节点，设置群组节点的配置文件**config.ini**中`[rpc]`部分的`listen_ip`和`channel_listen_port`。
  - `caCert`用于配置链ca证书路径
  - `sslCert`用于配置SDK所使用的证书路径
  - `sslKey`用于配置SDK所使用的证书对应的私钥路径

建链

本项目已经包含了链端，且配置完毕。

如果欲自己建链，则将nodes/删除，可使用根目录下的`build_chain.sh`搭建。例如执行下面的指令，生成一条单群组4节点的FISCO链。请确保机器的`30300~30303，20200~20203，8545~8548`端口没有被占用。

```
bash build_chain.sh -l "127.0.0.1:4" -p 30300,20200,8545
```

命令执行成功会输出`All completed`。如果执行出错，请检查`nodes/build.log`文件中的错误信息。建链成功后需按配置SDK配置。

具体说明参考单群组FISCO BCOS联盟链的搭建。

心得体会

区块链是分布式的记账账本。具有防伪造、防篡改、可追溯的技术特性，有利于解决制造业中的设备管理、数据共享、多方信任协作、安全保障等问题，对于提升工业生产效率、降低成本，提升供应链协同水平和效率，以及促进管理创新和业务创新具有重要作用。

通过这次大作业，深入体验了基于FISCO-BCOS 的开发过程。个人感觉区块链项目开放与普通项目的不同是将对数据库的访问改成于区块链合约的访问调用。此外深刻认识区块链在供应链金融这个实际场景中的应用和价值。

供应链金融将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

实验中遇到不少坑，主要是与合约配置和SDK调用相关。花了许多时间去debug。

感觉SDK的文档不够清晰，很多函数不知道如何使用。

完成的加分项

- 友好高效的用户界面
- 源代码及相关说明

参考资料

- Java SDK
- Spring Boot Starter