

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 级	专业 (方向)	软件工程
学号	17343018	姓名	陈瑜祯
电话	17623329058	Email	cccxxmo@outlook.com
开始日期	2019/12/5	完成日期	2019/12/12

一、项目背景

- 在传统供应链金融中，企业的信用只对与该企业直接相关的交易有效，无法向下游传递。以车企为例：

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。在接下来的几个月里，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。

- 在区块链+供应链中，区块链会将供应链上的每一笔交易和应收账款单据上链，并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。区块链也会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。

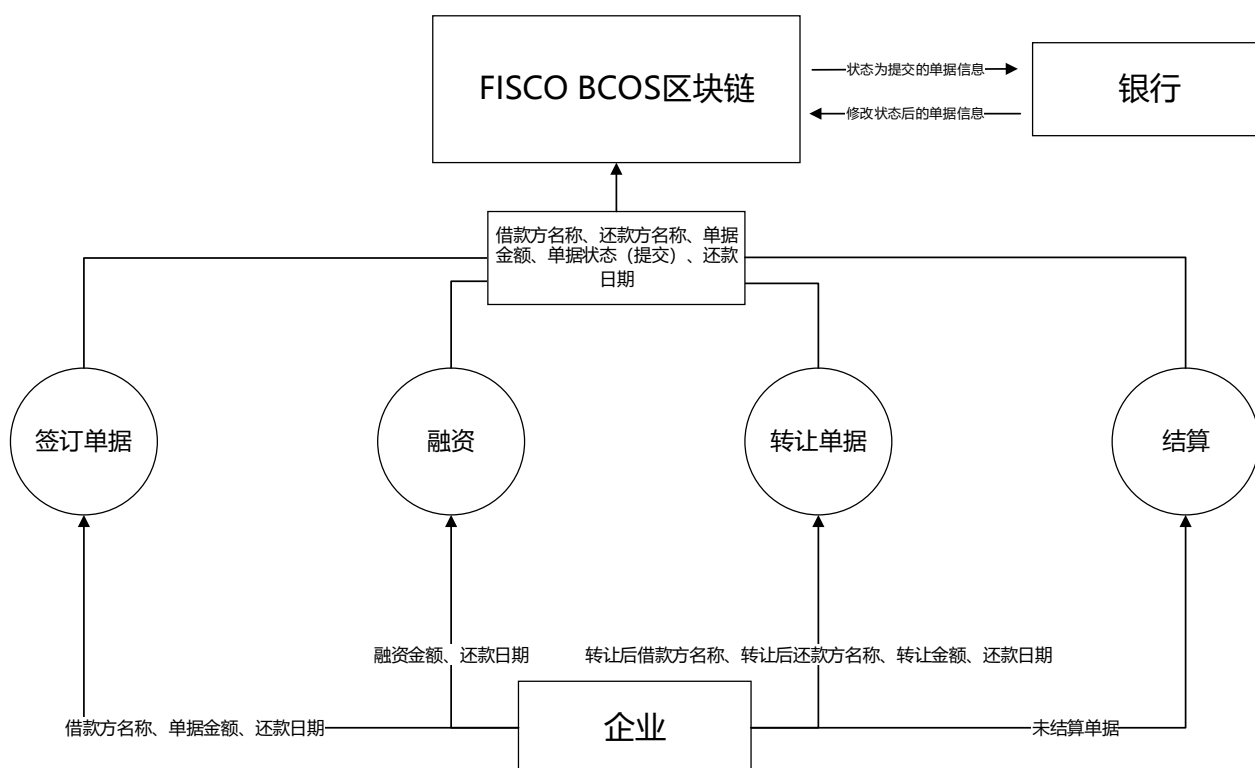
二、 方案设计

- 存储设计

- 采用 FISCO-BCOS 平台提供的 CRUD 接口实现企业信息和收据的存储。

CRUD 接口通过在 Solidity 合约中支持分布式存储预编译合约，可以实现将 Solidity 合约中数据存储在 FISCO BCOS 平台 AMDB 的表结构中，实现合约逻辑与数据的分离。

- 数据流图



- 核心功能介绍

- 企业注册 (Sign Up)

为企业创建一个账户，并生成相应的密钥。

- ◆ 后端代码

```
def on_press_register(self):
    name, password = self.line_name.text(), self.line_pwd.text()
    max_account_len = 240
    if len(name) > max_account_len:
        QMessageBox.warning(self, 'Error', 'The name should be less than 240 characters!')
        sys.exit(1)
    print("starting : {} {}".format(name, password))
```

```

ac = Account.create(password)
print("new address :\t", ac.address)
print("new privkey :\t", encode_hex(ac.key))
print("new pubkey :\t", ac.publickey)

kf = Account.encrypt(ac.privateKey, password)
keyfile = "{}/{}.keystore".format(client_config.account_keyfile_path, name)
print("save to file : {}".format(keyfile))
with open(keyfile, "w") as dump_f:
    json.dump(kf, dump_f)
    dump_f.close()
print(
    "INFO >> Read {} again after new account,address & keys in file:".forma
t(keyfile))
with open(keyfile, "r") as dump_f:
    keytext = json.load(dump_f)
    privkey = Account.decrypt(keytext, password)
    ac2 = Account.from_key(privkey)
    print("address:\t", ac2.address)
    print("privkey:\t", encode_hex(ac2.key))
    print("pubkey :\t", ac2.publickey)
    print("\naccount store in file: {}".format(keyfile))
    dump_f.close()

global client, contract_abi, to_address
args = [name, ac.address, 'Company']
print(name)
receipt = client.sendRawTransactionGetReceipt(to_address,contract_abi,"regist
er",args)
print("receipt:",receipt['output'])

QMessageBox.information(self,'Prompt','Successfully registered!', QMessageBox
.Ok)

```

◆ 链端代码

```

// register for companies
function register(string _name, string _address, string _type) public returns (in
t count)
{
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("company_t");
    Entry entry_to_insert = table.newEntry();
    entry_to_insert.set("dummy", "active");
    entry_to_insert.set("name", _name);
    entry_to_insert.set("address", _address);
    entry_to_insert.set("type", _type);
    emit InsertResult(count);
    count = table.insert("active", entry_to_insert);
    return count;
}

```

```
}
```

■ 企业登录 (Log In)

已注册的企业输入名称和密码进行登录。

◆ 后端代码

```
def validate(self):
    name = self.line_name.text()
    password = self.line_pwd.text()
    if name == "bank" and password == "bank":
        bank_window.show()
        bank_window.set_table_content()
    else:
        keyfile = "{}/{}/.keystore".format(client_config.account_keyfile_path, name)
        #if the account doesn't exists
        if os.path.exists(keyfile) is False:
            QMessageBox.warning(self,
                                "error",
                                "Name {} doesn't exists. Please register first.".format(name),
                                QMessageBox.Yes)
        else:
            print("name : {}, keyfile:{} ,password {} ".format(name, keyfile, password))
            try:
                with open(keyfile, "r") as dump_f:
                    keytext = json.load(dump_f)
                    privkey = Account.decrypt(keytext, password)
                    ac2 = Account.from_key(privkey)
                    print("address:\t", ac2.address)
                    print("privkey:\t", encode_hex(ac2.key))
                    print("pubkey :\t", ac2.publickey)
                    company_window.show()
                    company_window.set_basic_info(name)
            except Exception as e:
                QMessageBox.warning(self,
                                    "error",
                                    ("Failed to load account info for [{}], "
                                     " error info: {}!").format(name, e),
                                    QMessageBox.Yes)
```

■ 签订单据 (Purchase)

企业登陆后, 可与其他企业签订单据。

◆ 后端代码

```
def on_submit_purchase(self):
    _amt = self.line_pur_amt.text()
    _due = self.purchase_date.dateTime().toString("yyyy/MM/dd hh:mm:ss")
    _from = self.line_pur_from.text()
    global client, contract_abi, to_address
    args = [self.company_name, _from, int(_amt), _due]
    info_tuple = client.sendRawTransactionGetReceipt(to_address, contract_abi, "purchase", args)
    print("receipt:", info_tuple['output'])
    res = hex_to_signed(info_tuple['output'])
    if res == -3:
        QMessageBox.warning(self, 'Error', 'Companies must be registered first!', QMessageBox.Ok)
    elif res == 1:
        QMessageBox.information(self, 'Prompt', 'Successfully submitted purchasing request.', QMessageBox.Ok)
```

◆ 链端代码

```
function purchase(string _from, string _to, int amt, string dd) public returns(int)
{
    if(is_registered(_from) == -1 || is_registered(_to) == -1)
    {
        return -3;
    }
    int count = insert(_from, _to, amt, "submitted", dd);
    if(count == 1)
    {
        return 1;
    }
    else
    {
        return -1;
    }
}
```

■ 融资 (Finance)

企业登录后，可向银行申请融资。融资金额不得超过该企业借款总金额与欠款总金额之差。

◆ 后端代码

```
def on_submit_finance(self):
    _amt = int(self.line_fin_amt.text())
```

```

_due = self.finance_date.dateTime().toString("yyyy/MM/dd hh:mm:ss")
if _amt > (self.total_lent - self.total_borrowed):
    QMessageBox.warning(self, 'Error', "You don't have enough capacity to finance. Your capacity is {}".format(str(self.total_lent - self.total_borrowed))), QMessageBox.Ok)
else:
    global client, contract_abi, to_address
    args = [self.company_name, "bank", _amt, _due]
    info_tuple = client.sendRawTransactionGetReceipt(to_address, contract_abi, "finance", args)
    QMessageBox.information(self, 'Prompt', 'Successfully financed.', QMessageBox.Ok)

```

◆ 链端代码

```

//finance from bank
function finance(string _from, string _to, int _amount, string dd) public
{
    insert(_from, _to, _amount, "submitted", dd);
}

```

■ 转让单据 (Transfer)

企业登陆后，若同时存在借出单据和欠款单据，则可转让单据。转让金额不得超过任一被转让单据的金额。

◆ 后端代码

```

def on_submit_transfer(self):
    global client, contract_abi, to_address
    if self.table_trans_lent.selectionModel().hasSelection() and self.table_trans_bor.selectionModel().hasSelection():
        row_lent = self.table_trans_lent.currentRow()
        row_bor = self.table_trans_bor.currentRow()
        _from = self.table_trans_lent.item(row_lent, 1).text()
        _due = self.table_trans_lent.item(row_lent, 4).text()
        _from_prev_amt = int(self.table_trans_lent.item(row_lent, 2).text())
        _to_prev_amt = int(self.table_trans_bor.item(row_bor, 2).text())
        _to = self.table_trans_bor.item(row_bor, 0).text()
        self.transfer_date.setDateTime(QDateTime.fromString(_due, 'yyyy/MM/dd hh:mm:ss'))
        _amt = int(self.line_trans_amt.text())
        print(_from, _to, _due, _amt)
        args = [_from, self.company_name, _to, _from_prev_amt, _to_prev_amt, _amt, _due]
        if self.table_trans_bor.item(row_bor, 3).text() == "authorized" and self.table_trans_lent.item(row_lent, 3).text() == "authorized":

```

```

        info_tuple = client.sendRawTransactionGetReceipt(to_address, contract
_abi, "transfer", args)
        print("receipt:", info_tuple['output'])
        res = hex_to_signed(info_tuple['output'])
        if res == -3:
            QMessageBox.warning(self, 'Error', 'Companies must be registered fi
rst!', QMessageBox.Ok)
        elif res == -1:
            QMessageBox.warning(self, 'Error', 'Amount must be no
more than the amount of any selected records.', QMessageBox.Ok)
        elif res == 1:
            QMessageBox.information(self, 'Prompt', 'Successfully transferred.'
, QMessageBox.Ok)
        else:
            QMessageBox.warning(self, 'Error', 'Only [Authorized] receipts can be t
ransferred!', QMessageBox.Ok)
        else:
            QMessageBox.warning(self, 'Prompt', 'Failed to transfer. Please click to se
lect records!', QMessageBox.Ok)

```

◆ 链端代码

```

function transfer (string _from, string _to, string _to_to, int _from_prev_amt, in
t _to_prev_amt, int _amount, string dd) public returns(int)
{
    if(is_registered(_from) == -1 || is_registered(_to) == -1)
    {
        return -3;
    }
    if(_amount > _from_prev_amt || _amount > _to_prev_amt)
    {
        return -1;
    }
    update(_from, _to, _from_prev_amt - _amount, "submitted", dd);
    update(_to, _to_to, _to_prev_amt - _amount, "submitted", dd);
    insert(_from, _to_to, _amount, "submitted", dd);
    return 1;
}

```

■ 结算 (Repay)

企业登录后，可对已有的欠款单据进行结算。

◆ 后端代码

```

def on_repay(self):
    global client, contract_abi, to_address
    if self.table_repay.selectionModel().hasSelection():
        row = self.table_repay.currentRow()

```

```

        args = [self.table_repay.item(row, 0).text(), self.table_repay.item(row,
1).text(), \
                int(self.table_repay.item(row, 2).text()),self.table_repay.item(row,
4).text()]\
        print(args)
        if self.table_repay.item(row, 3).text() == "authorized":
            info_tuple = client.sendRawTransactionGetReceipt(to_address, contract
_abi, "repay", args)
            print("receipt:",info_tuple)
            QMessageBox.information(self,'Prompt','Successfully repayed.', QMessa
geBox.Ok)
            self.table_repay.setRowCount(0)
            self.set_table_repay_content(self.company_name)
        else:
            QMessageBox.warning(self,'Error','Only [Authorized] receipts can be r
epayed!', QMessageBox.Ok)
        else:
            QMessageBox.warning(self,'Prompt','Failed to repay. Please click to selec
t a record!', QMessageBox.Ok)

```

◆ 链端代码

```

//repay a receipt
function repay(string _from, string _to, int _amount, string _expiration) public
returns(int)
{
    remove(_from, _to, _amount, _expiration);
}

```

■ 银行确认单据

银行登录后（账号名称和密码均为 bank），可批准或拒绝已提交的单据。

◆ 后端代码

```

def on_authorize(self):
    global client, contract_abi, to_address
    if self.table.selectionModel().hasSelection():
        row = self.table.currentRow()
        args = [self.table.item(row, 0).text(), self.table.item(row, 1).text(), \
                int(self.table.item(row, 2).text()), "authorized",self.table.item(row
, 4).text()]\
        print(args)
        info_tuple = client.sendRawTransactionGetReceipt(to_address, contract_abi
, "update", args)
        print("receipt:",info_tuple)
        QMessageBox.information(self,'Prompt','Successfully authorized!', QMessag
ebox.Ok)
        self.table.setRowCount(0)

```



```

        self.set_table_content()
    else:
        QMessageBox.warning(self, 'Prompt', 'Failed to authorize. Please click to select a record!', QMessageBox.Ok)

def on_reject(self):
    global client, contract_abi, to_address
    if self.table.selectionModel().hasSelection():
        row = self.table.currentRow()
        args = [self.table.item(row, 0).text(), self.table.item(row, 1).text(), \
                int(self.table.item(row, 2).text()), self.table.item(row, 4).text()]
        print(args)
        info_tuple = client.sendRawTransactionGetReceipt(to_address, contract_abi, "remove", args)
        print("receipt:", info_tuple)
        QMessageBox.information(self, 'Prompt', 'Successfully rejected!', QMessageBox.Ok)

        self.table.setRowCount(0)
        self.set_table_content()
    else:
        QMessageBox.warning(self, 'Prompt', 'Failed to reject. Please click to select a record!', QMessageBox.Ok)

```

◆ 链端代码

```

//update records
function update(string _from, string _to, int amt, string sta, string dd) public returns(int)
{
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("receipt_t");
    Entry entry = table.newEntry();
    entry.set("dummy", "active");
    entry.set("from", _from);
    entry.set("to", _to);
    entry.set("amount", amt);
    entry.set("status", sta);
    entry.set("due_date", dd);
    Condition condition = table.newCondition();
    condition.EQ("from", _from);
    condition.EQ("to", _to);
    int count = table.update("active", entry, condition);
    emit UpdateResult(count);
    return count;
}

//remove records
function remove(string _from, string _to, int amt, string dd) public returns(int)
{
    TableFactory tf = TableFactory(0x1001);
    Table table = tf.openTable("receipt_t");

```

```
Condition condition = table.newCondition();
condition.EQ("from", _from);
condition.EQ("to", _to);
condition.EQ("amount", amt);
condition.EQ("due_date", dd);

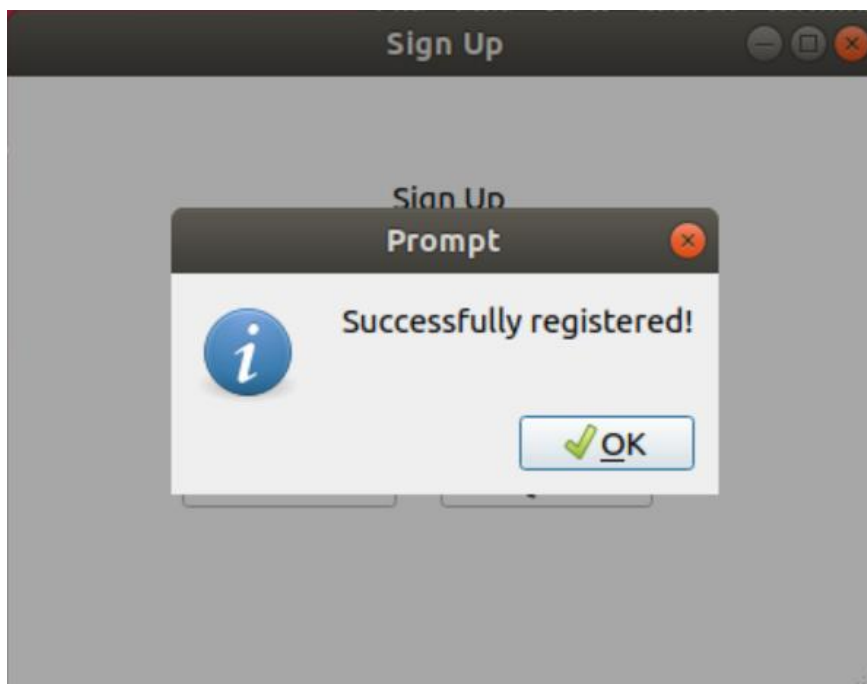
int count = table.remove("active", condition);
emit RemoveResult(count);

return count;
}
```

三、 功能测试

- **企业注册 (Sign Up)**

注册一个名称为 test, 密码为 123 的账户。



GUI 截图 (注册成功)

```
starting : test 123
new address : 0x0F94866E17a3075E519678f622C7Cf7B903B8023
new privkey : 0x82b0ccbb080b7fd33eeefc661a061ec2642da20374ce6a
51532fe2c8f8e3abb4
new pubkey : 0x0893b273635fe448c78585a152b6b960637a0a6034cce4
57d914526a3ca35dd3a85d99486304b21c10998d2ed70a21eedfac0519cf1ca63
dfd2e98acab734e79
save to file : [bin/accounts/test.keystore]
INFO >> Read [bin/accounts/test.keystore] again after new account
,address & keys in file:
address: 0x0F94866E17a3075E519678f622C7Cf7B903B8023
privkey: 0x82b0ccbb080b7fd33eeefc661a061ec2642da20374ce6a
51532fe2c8f8e3abb4
pubkey : 0x0893b273635fe448c78585a152b6b960637a0a6034cce4
57d914526a3ca35dd3a85d99486304b21c10998d2ed70a21eedfac0519cf1ca63
dfd2e98acab734e79

account store in file: [bin/accounts/test.keystore]
test
receipt: 0x0000000000000000000000000000000000000000000000000000000000000000
0000000001
```

命令行截图

- 企业登录 (Log In)

登录 test 账户。

The screenshot shows a window titled "Company" with a dark header bar containing standard window controls. Below the header is a row of six buttons: "Info", "Transfer", "Purchase", "Finance", "Repay", and "Quit". The "Info" button is selected. The main area displays account statistics: "Total borrowed" and "Total lent", both with input fields showing "0". Below these are two tables. The first table is titled "borrowed" and has columns "From", "To", and "Amount". It is currently empty. The second table is titled "lent" and also has columns "From", "To", and "Amount", and is also empty. Both tables have scrollbars at the bottom.

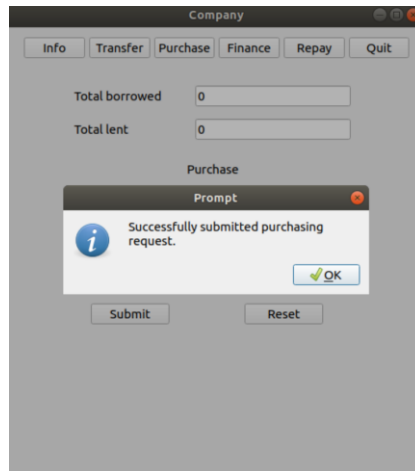
GUI 截图 (进入账户信息页)

```
name : test, keyfile:bin/accounts/test.keystore ,password 123
address:      0x0F94866E17a3075E519678f622C7Cf7B903B8023
privkey:      0x82b0ccbb080b7fd33eeefc661a061ec2642da20374ce6a
51532fe2c8f8e3abb4
pubkey :      0x0893b273635fe448c78585a152b6b960637a0a6034cce4
57d914526a3ca35dd3a85d99486304b21c10998d2ed70a21eedfac0519cf1ca63
dfd2e98acab734e79
receipt: ((), (), (), (), ())
receipt: ((), (), (), (), ())
receipt: ((), (), (), (), ())
receipt: ((), (), (), (), ())
receipt: ((), (), (), (), ())
```

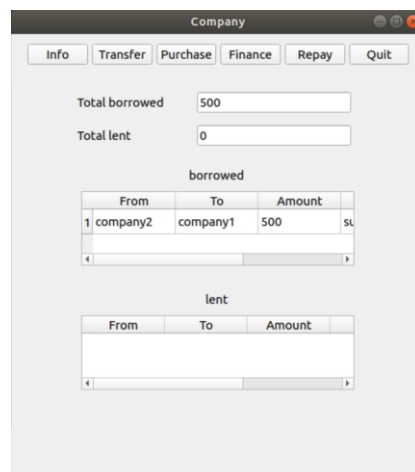
命令行截图

- 签订单据 (Purchase)

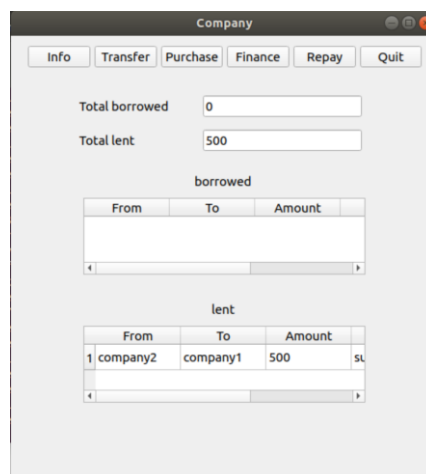
company1 向 company2 签订购买 500 元物品的单据。



GUI 截图 (签订成功)



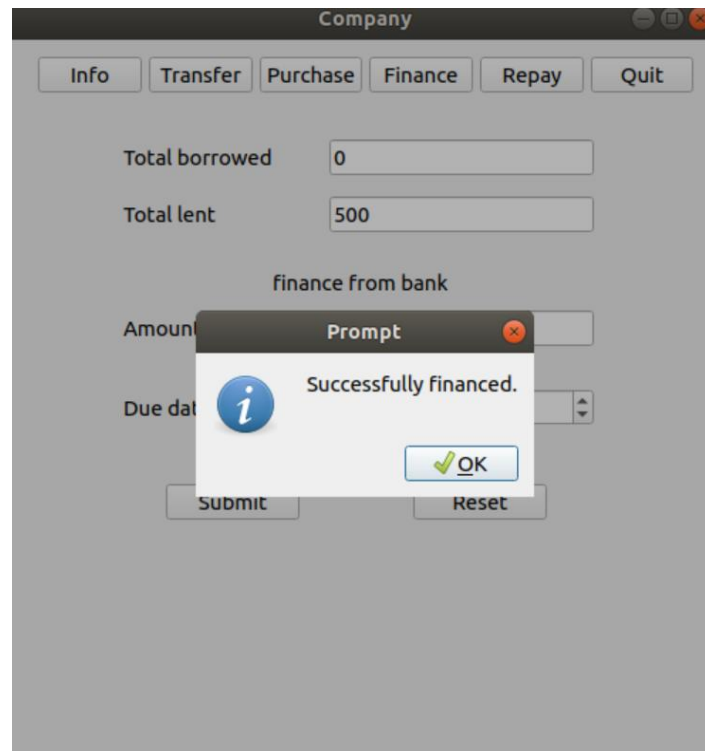
GUI 截图 (签订成功后查看 company1 的账户信息页)



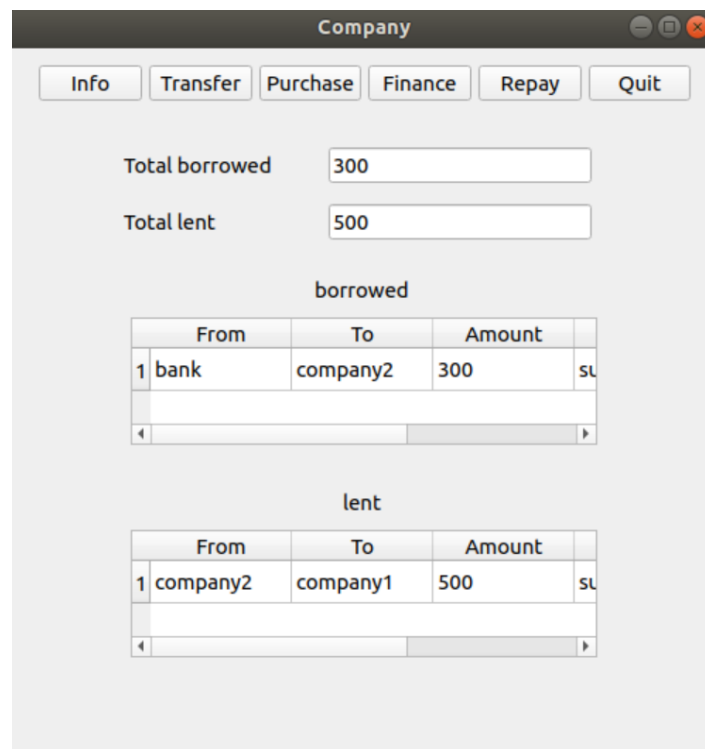
GUI 截图 (签订成功后查看 company2 的账户信息页)

- 融资 (Finance)

company2 向银行融资 300 元。



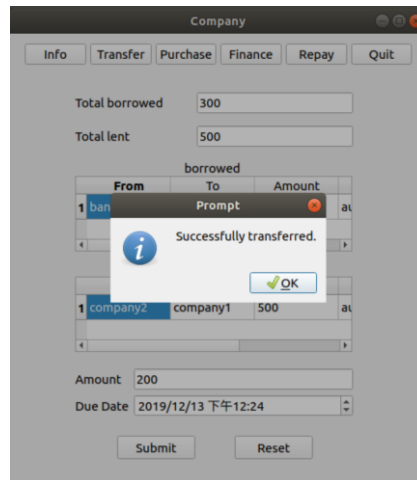
GUI 截图 (融资成功)



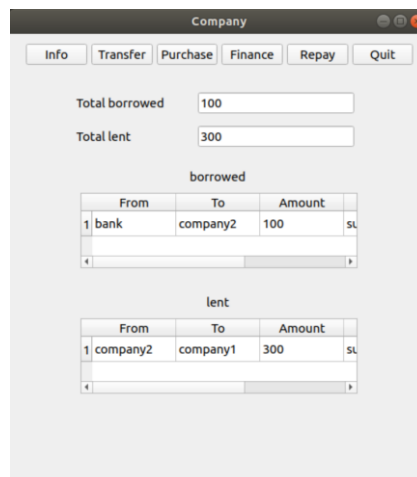
GUI 截图 (融资成功后查看 company2 的账户信息页)

- **转让单据 (Transfer)**

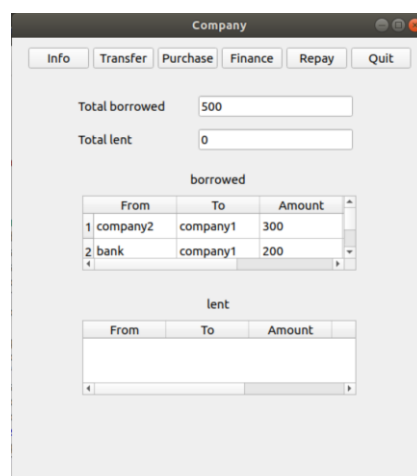
company2 将融资欠款中的 200 元转让给 company1。



GUI 截图 (转让成功)



GUI 截图 (转让成功后查看 company2 的账户信息页)



GUI 截图 (转让成功后查看 company1 的账户信息页)

- 结算 (Repay)

company1 结算与银行的单据。

Company

Info Transfer Purchase Finance Repay Quit

Total borrowed 200

Total lent 0

borrowed

	From	To	Amount
1	bank	company1	200

lent

	From	To	Amount
--	------	----	--------

GUI 截图 (结算成功前查看 company1 的账户信息页)

Company

Info Transfer Purchase Finance Repay Quit

Total borrowed 200

Total lent 0

Click on a record to select.

1 comp

From To Amount

1 bank company1 200

lent

From To Amount

Prompt

Successfully repayed.

OK

GUI 截图 (结算成功)

Company

Info Transfer Purchase Finance Repay Quit

Total borrowed 0

Total lent 0

borrowed

	From	To	Amount
--	------	----	--------

lent

	From	To	Amount
--	------	----	--------

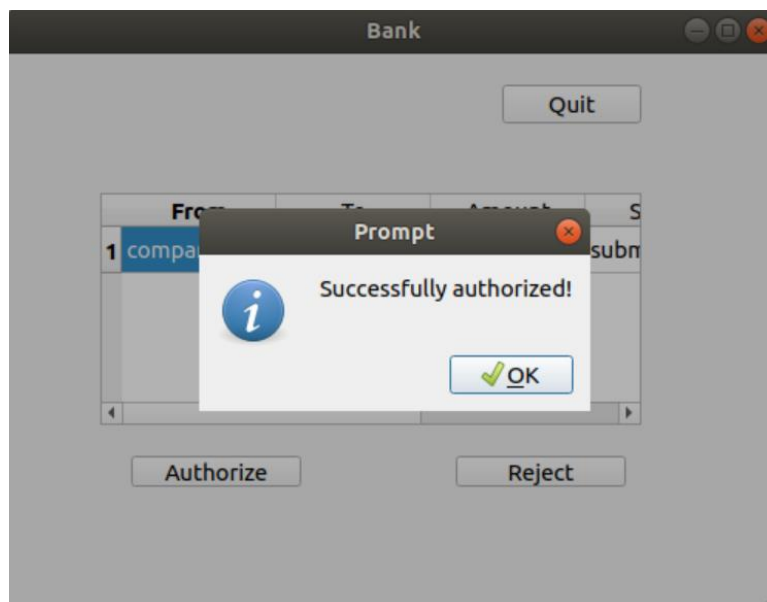
GUI 截图 (结算成功后查看 company1 的账户信息页)

- 银行确认单据

银行登陆后（账户名称和密码均为 bank）批准 company2 向银行融资 300 元。

borrowed				
	To	Amount	Status	
1	company2	300	submitted	20

GUI 截图（银行处理单据前查看 company2 的账户信息页）



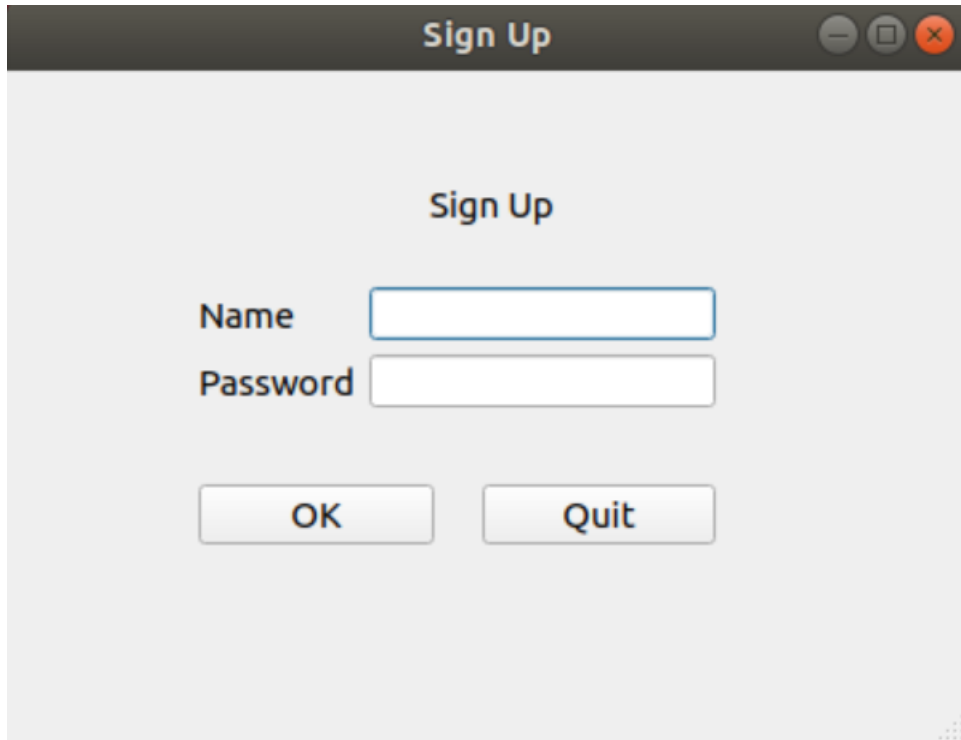
GUI 截图（银行批准单据）

borrowed				
	To	Amount	Status	
1	company2	300	authorized	20

GUI 截图（银行批准单据后查看 company2 的账户信息页）

四、 界面展示

- 企业注册 (Sign Up)



A screenshot of a 'Sign Up' dialog box. The title bar is dark gray with the text 'Sign Up' and standard window control buttons (minimize, maximize, close). The main area is light gray. At the top center is the text 'Sign Up'. Below it are two input fields: 'Name' and 'Password'. At the bottom are two buttons: 'OK' and 'Quit'.

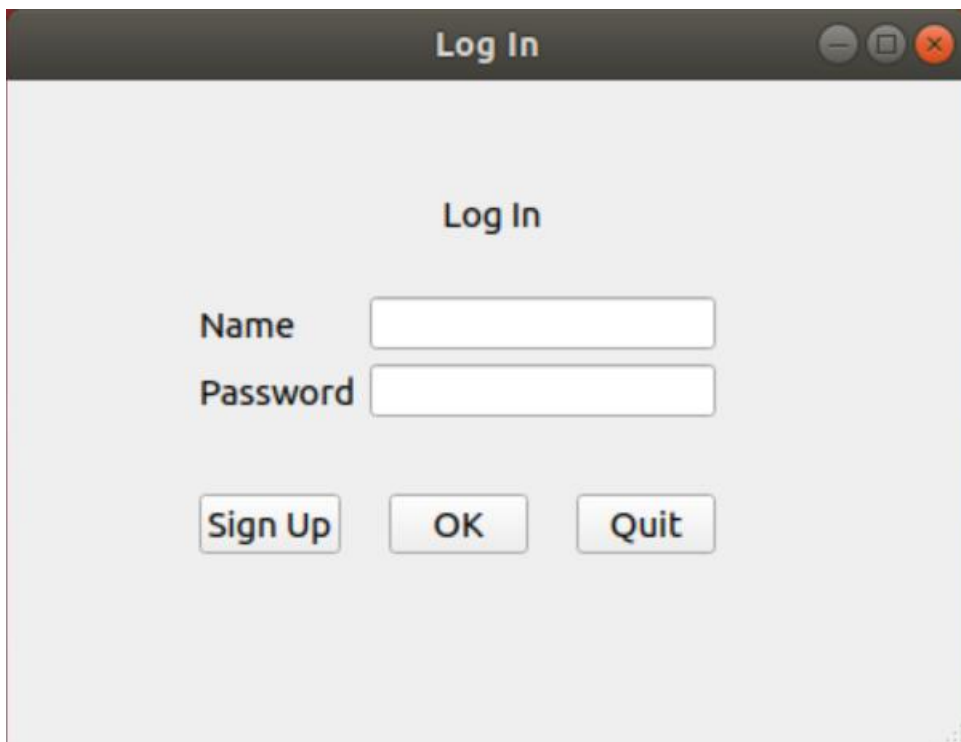
Sign Up

Name

Password

OK Quit

- 企业登录 (Log In)



A screenshot of a 'Log In' dialog box. The title bar is dark gray with the text 'Log In' and standard window control buttons (minimize, maximize, close). The main area is light gray. At the top center is the text 'Log In'. Below it are two input fields: 'Name' and 'Password'. At the bottom are three buttons: 'Sign Up', 'OK', and 'Quit'.

Log In

Name

Password

Sign Up OK Quit

- 企业信息 (Info)

Company

Info

Transfer

Purchase

Finance

Repay

Quit

Total borrowed

0

Total lent

0

borrowed

From	To	Amount	

lent

From	To	Amount	

- 转让单据 (Transfer)

Company

Info

Transfer

Purchase

Finance

Repay

Quit

Total borrowed

0

Total lent

0

borrowed

From	To	Amount	

lent

From	To	Amount	

Amount

Due Date

2019/12/13 下午1:50

Submit

Reset

- 签订单据 (Purchase)

Company

Info

Transfer

Purchase

Finance

Repay

Quit

Total borrowed

0

Total lent

0

Purchase

From

Amount

Due Date

2019/12/13 下午1:50

Submit

Reset

- 融资 (Finance)

Company

Info

Transfer

Purchase

Finance

Repay

Quit

Total borrowed

0

Total lent

0

finance from bank

Amount

Due date

2019/12/13 下午1:50

Submit

Reset

- 结算 (Repay)

Company

—

□

×

Info

Transfer

Purchase

Finance

Repay

Quit

Total borrowed

0

Total lent

0

Click on a record to select.

From	To	Amount	

◀

▶

OK

- 银行确认单据

Bank

Quit

From	To	Amount	Sta
------	----	--------	-----

Authorize

Reject

五、 心得体会

完成本次大作业后，我对区块链的原理有了更深的理解，对前端、后端和链端的开发工具的使用和开发过程也更加熟悉，对其各自的功能和它们之间的数据交互也有了更直观的认识。

由于网上关于 FISCO BCOS 的资源和项目有限，加上自己的经验不足，我在开始对如何处理后端以及其与链端的通信毫无头绪，不得已参考了其他同学的完成方式后才有了些许眉目，尤其是关于数据库的处理和后端与链端之间的通信。因此代码的部分逻辑与结构可能会与其他同学有些相似，在此提前说明。

这次大作业也使我更清晰地认识到了区块链去中心化、可追溯、防篡改的特性在生活中的应用场景的丰富。这次实现的供应链管理的模式同样可以推广到其他领域，比如学历的认证与管理、食品原料的交易与溯源等。联想如今区块链应用的落地情况，可以看出其还有更大的发展空间和发展潜力。

六、 加分项

- 设计了友好高效的用户界面。
 - 在显示借出单据和欠款单据时，修改了表中“from”和“to”的显示顺序，更符合人阅读表格时的思维习惯。
 - 在转让单据时，用户只需点击选择相应的借出单据和欠款单据即可，不需要重新输入单据相关的信息。
 - 当输入不合要求时会弹出窗口提醒用户，便于用户作出相应的更改。

【附】github 地址: <https://github.com/cccxmo/BlockchainFinalProject>