

项目开发

学习目标

1. 了解项目背景和用户诉求
2. 了解项目需求和区块链系统组成
3. 利用第三方云平台完成区块链系统服务创建
4. 独立完成个人信息认证
5. 独立完成房屋信息认证
6. 独立完成个人征信认证
7. 独立完成电子合同和租金记录管理

租房网区块链系统

场景分析

近年来，租房这个相对偏传统的行业，在快速发展中已经成为不少大城的“痛”，虚假房源泛滥、黑中介横行、租客和房东之间缺乏信任、行业交易效率低下等问题一直存在。

用户诉求

构建一个更加可靠的互联网租房系统，从根本上解决价值交换与转移中存在的欺诈和寻租等现象。租房者和出租人信息不对称等问题。



项目背景

中国银行与蚂蚁金服合作，通过区块链技术助力“雄安”租房领域

4月20日，蚂蚁雄安数字技术有限公司（下称“蚂蚁雄安”）与中国银行雄安分行在雄安新区签署战略合作协议据了解，未来双方将基于在区块链技术打造的雄安住房租赁相关领域开展深度合作，为各方提供相关金融服务支持，以此助力新区在住房租赁领域快速发展。

项目分析

系统会部分代替中介工作，通过房管局节点校验房屋信息，通过公安局节点验证个人信息，通过征信中心验证个人信用。

当租房时，租户能够了解到房屋的一些信息，比如：是否是可租赁房屋，是否是房东直租，房屋是否存在抵押情况等。

房东可以通过平台了解到租户的个人信息，是否有违法记录，个人征信信息是否良好等。

当出现合租时，房东和现有租户，都可以了解到新租户的个人信息，同时新租户也可以了解到对等的信息。

个人认证信息、房产信息、征信信息都会分布在不同的节点上，现阶段由各个部门单独管理，我们在需要认证时发送认证请求即可。

租赁合同、租金、违约金等不可篡改数据会上链。各个节点均可记录这些信息。

系统组成（联盟链）

Orderer Service：2个节点，共识策略使用kafka

Organizations：3个组织（公安局、房管局、征信中心），每个组织两个节点

channel：

1. 三个组织独立的channel
2. 三个组织的联合channel

利用华为云平台完成上述系统的创建。

个人认证

学习目标：

1. 独立完成个人信息认证（chaincode）编写
2. 独立完成后端个人信息认证功能
3. 独立完成前端个人信息认证功能
4. 独立完成个人信息上传（chaincode）编写
5. 独立完成chaincode更新
6. 独立完成后端信息上传功能
7. 独立完成前端信息上传功能

认证chaincode编写

个人认证信息会发送到相关的政府部门节点，通过查询区块链上的数据得到用户的认证信息结果。回复内容包括两项认证结果：姓名与证件是否匹配的结果和是否有不良记录的结果（考虑到个人隐私，不会返回不良记录内容）。为了能够测试通过，我们当前仅仅返回测试结果。

```

package main

import (
    "github.com/hyperledger/fabric/core/chaincode/shim"
    "github.com/hyperledger/fabric/protos/peer"
    "fmt"
    "strings"
)

// 个人认证
// 发送姓名和身份证信息，回复：是否通过认证，是否有不良个人记录

type Auth struct {
}

func (this *Auth) Init(stub shim.ChaincodeStubInterface) peer.Response {
    return shim.Success(nil)
}

func (this *Auth) Invoke(stub shim.ChaincodeStubInterface) peer.Response {
    function, args := stub.GetFunctionAndParameters()

    if function == "check" {
        return this.check(stub, args)
    }

    return shim.Error("Invalid Smart Contract function name.")
}

func (this *Auth) check(stub shim.ChaincodeStubInterface, args []string) peer.Response {
    name := args[0]
    id := args[1]

    data, err := stub.GetState(id) // 通过身份证号查询出姓名和是否有不良记录信息。数据结构
    name:true(false)
    if err != nil {
        return shim.Error(fmt.Sprintf("Error getting data %s", err.Error()))
    }

    var result string = ""
    if data != nil {
        var str string = string(data[:]) // []byte结果转换成字符串
        split := strings.Split(str, ":") // 依据":"对结果进行划分

        if split[0] == id {
            result = "true"
        } else {
            result = "false"
        }
        result = result + ":" + split[1]
        return shim.Success([]byte(result))
    }
    // TODO 由于没有记录数据,此处我们使用模拟数据

```

```
return shim.Success([]byte("true:true"))
//return shim.Success([]byte("false:false"))
}
```

后端个人信息认证

```
func (this *AuthController) Check() {
    name := this.GetString("name")
    id := this.GetString("id")
    if name == "" || id == "" {
        handleResponse(this.Ctx.ResponseWriter, 400, "Request parameter name(or id) can't be empty")
        return
    }
    beego.Info(name + ":@" + id)
    args := [][]byte{[]byte(name), []byte(id)}

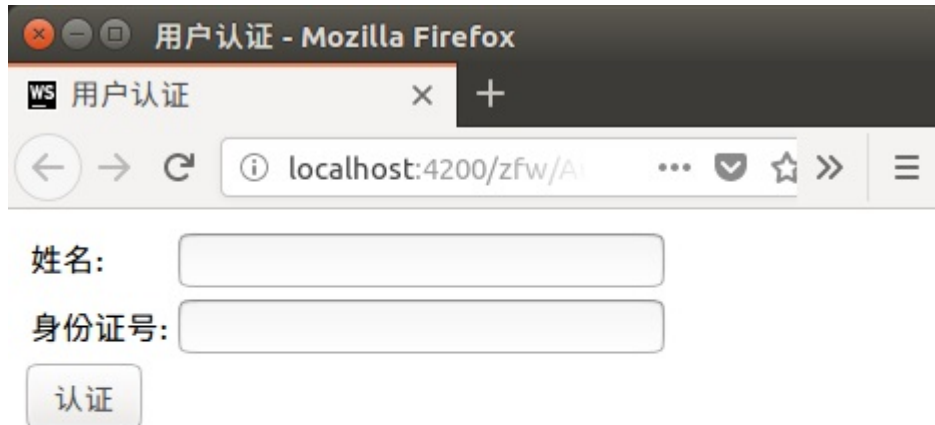
    var (
        channelId      = beego.AppConfig.String("channel_id_gaj")
        chainCodeName = beego.AppConfig.String("chaincode_id_auth")
    )
    ccs, err := models.Initialize(channelId, chainCodeName, userId)
    if err != nil {
        handleResponse1(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
    defer ccs.Close()

    response, err := ccs.ChaincodeQuery("check", args)
    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
    handleResponse(this.Ctx.ResponseWriter, 200, response)
}
```

测试

- 1、证书下载及解压
- 2、配置文件下载及解压
- 3、Host文件修改
- 4、app.conf配置文件修改（参照test项目，完成key和value的同时修改）
- 5、通过浏览器模拟get请求

前端个人信息认证



The screenshot shows a Mozilla Firefox browser window titled "用户认证 - Mozilla Firefox". The address bar displays "localhost:4200/zfw/A". The page content includes a form with two input fields: "姓名:" (Name) and "身份证号:" (ID Number). Below these fields is a button labeled "认证" (Authenticate).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>用户认证</title>
  <script src="vue.js"></script>
  <script src="vue-resource.js"></script>
</head>
<body>
<div id="app">
  <table>
    <tr>
      <td>姓名:</td>
      <td><input type="text" v-model="name"></td>
    </tr>
    <tr>
      <td>身份证号:</td>
      <td><input type="text" v-model="id"><br></td>
    </tr>
  </table>
  <button v-on:click="check">认证</button>
</div>
</body>
<script>
  var vm = new Vue({
    el: '#app',
    data:{
      name:"",
      id:""
    },
    methods:{
      check:function () {
```

```

        var url="http://localhost:8080/auth"
        var params={"name":this.name,"id":this.id}
        this.$http.get(
            url,
            {params:params},// get请求传递参数固定格式
            {emulateJSON: true}// 跨域访问
        ).then(
            function (response) {
                console.log(response)
            },
            function (response) {
                console.log(response)
            }
        )
    }
}
})
</script>
</html>

```

上传chaincode编写

```

func (this *Auth) add(stub shim.ChaincodeStubInterface, args [] string) peer.Response {
    // 以身份证号为key进行信息存储
    // id:name:record
    if len(args) != 3 {
        return shim.Error("Incorrect number of arguments. Expecting 3")
    }
    id := args[0]
    name := args[1]
    record := args[2]

    err := stub.PutState(id, []byte(name+":"+record))
    if err != nil {
        shim.Error(fmt.Sprintf("error in commit account info %s", err.Error()))
    }

    return shim.Success(nil)
}

```

chaincode更新

后端上传

CSV文件操作

写操作

```
func main() {
    file, _ := os.OpenFile("test.csv", os.O_WRONLY|os.O_CREATE, os.ModePerm)
    w := csv.NewWriter(file)
    w.Write([]string{"123", "234234", "345345", "234234"})
    w.Write([]string{"123", "234234", "345345", "234234"})
    w.Flush()

    file.Close()
}
```

读操作

```
func main() {
    file, _ := os.Open("test.csv")
    r := csv.NewReader(file)

    for {
        strs, err := r.Read()

        if err == io.EOF{
            break
        }
        if err != nil{
            beego.Error(err)
        }

        for _, str := range strs {
            fmt.Print(str, "\t")
        }
        fmt.Println()
    }
}
```

CSV文件上传

获取上传文件，保存到本地，通过任务管理文件内容写入区块链。

保存文件到服务器：

```
//上传文件
func (this *AuthController) RecordAuth() {
    beego.Debug("receive file")
    //file, 这是一个key值, 对应上传Form中的name信息
    f, h, e := this.GetFile("file")
    if e != nil {
```

```

        // 如果出现异常了,f内容为nil
        handleResponse(this.Ctx.ResponseWriter, 400, e.Error())
        return
    }
    //关闭上传的文件, 不然的话会出现临时文件不能清除的情况
    defer f.Close()

    //得到文件的名称
    fileName := h.FileName
    fmt.Println("文件名称:")
    fmt.Println(fileName)

    //保存文件到指定的位置
    //static/uploadfile,这个是文件的地址, 第一个static前面不要有/
    err := this.SaveToFile("file", path.Join("static/uploadfile", fileName))

    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }

    handleResponse(this.Ctx.ResponseWriter, 200, "ok")
}

```

开启异步任务：

```

//上传文件
func (this *AuthController) RecordAuth() {
    beego.Debug("receive file")
    //file, 这是一个key值, 对应上传Form中的name信息
    f, h, e := this.GetFile("file")
    if e != nil {
        // 如果出现异常了,f内容为nil
        handleResponse(this.Ctx.ResponseWriter, 400, e.Error())
        return
    }
    //关闭上传的文件, 不然的话会出现临时文件不能清除的情况
    defer f.Close()

    //得到文件的名称
    fileName := h.FileName
    fmt.Println("文件名称:")
    fmt.Println(fileName)

    //保存文件到指定的位置
    //static/uploadfile,这个是文件的地址, 第一个static前面不要有/
    err := this.SaveToFile("file", path.Join("static/uploadfile", fileName))

    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
}

```



```

// 开启任务
// 在当前时间的5秒钟后开启
t := time.Now().Add(5 * time.Second)
second := t.Second()
minute := t.Minute()
hour := t.Hour()
spec := fmt.Sprintf("%d %d %d * * *", second, minute, hour)

tk := toolbox.NewTask("myTask", spec,
    func() error {
        beego.Info("start task")
        defer toolbox.StopTask()
        return myTask(fileName)
    })

toolbox.AddTask("myTask", tk)
toolbox.StartTask()

handleResponse(this.Ctx.ResponseWriter, 200, "ok")
}

```

myTask内容：

```

func myTask(fileName string) error{
    rf, _ := os.Open(path.Join("static/uploadfile", fileName))
    r := csv.NewReader(rf)

    for {
        str, err := r.Read()

        if err == io.EOF {
            break
        }
        if err != nil {
            beego.Error(err)
        }

        for _, str := range str {
            fmt.Print(str, "\t")
        }
        fmt.Println()
    }
    return nil
}

```

记录信息到区块：

```

func myTask(fileName string) error{
    // 区块写操作

```

```

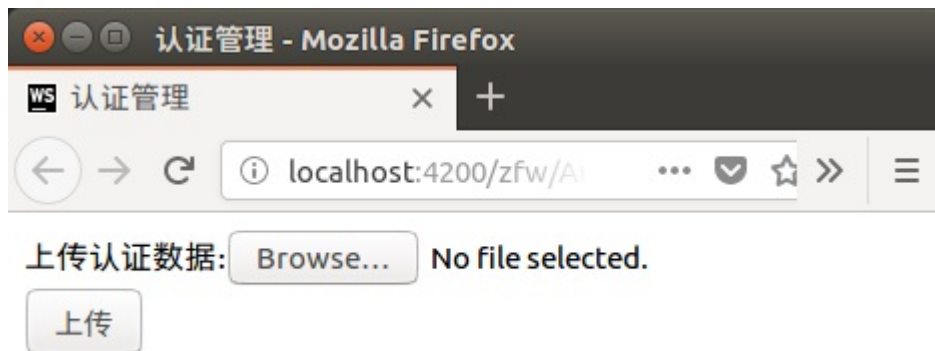
var (
    channelId      = beego.AppConfig.String("channel_id_gaj")
    chainCodeId    = beego.AppConfig.String("chaincode_id_auth")
)
ccs, err := models.Initialize(channelId, chainCodeId, userId)
if err != nil {
    beego.Error(err)
    return err
}
defer ccs.Close()
rf, err := os.Open(path.Join("static/uploadfile", fileName))
if err != nil {
    beego.Error(err)
    return err
}
rf.Close()
r := csv.NewReader(rf)

var line = 0
var lines []string
for {
    line += 1
    lineStr := strconv.Itoa(line)
    str, err := r.Read()

    if err == io.EOF {
        break
    }
    if err != nil {
        lines = append(lines, lineStr)
        continue
    }
    if len(str) != 3 {
        // 读取到的信息长度错误
        lines = append(lines, lineStr)
        continue
    }
    var args [][]byte
    for _, data := range str {
        args = append(args, []byte(data))
    }
    _, e := ccs.ChaincodeSet("add", args)
    if e != nil {
        lines = append(lines, lineStr)
    }
}
if len(lines) > 0 {
    beego.Error("Error lines:", strings.Join(lines, ","))
}
return nil
}

```

前端上传



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>认证管理</title>
  <script src="vue.js"></script>
  <script src="vue-resource.js"></script>
</head>
<body>
<div id="app">
  上传认证数据:<input type="file" @change="onUpload"><br>
  <button v-on:click="upload">上传</button>
</div>
</body>
<script>
  var vm = new Vue({
    el: '#app',
    data:{
      formData:new FormData()
    },
    methods:{
      onUpload(e){
        this.formData.append('file', e.target.files[0])
        this.formData.append('type', 'csv')
      },
      upload:function () {
        var url ="http://localhost:8080/auth"
        this.$http.post(url,this.formData).then(
          function (response) {
            alert("true")
          },
          function (response) {
            alert("false")
          }
        )
      }
    }
  })
})
```

```
</script>
</html>
```

房东认证

学习目标：

1. 独立完成个房东息认证 (chaincode) 编写
2. 独立完成后端房东信息认证功能
3. 独立完成前端房东信息认证功能
4. 独立完成房东信息上传 (chaincode) 编写
5. 独立完成chaincode更新
6. 独立完成后端信息上传功能
7. 独立完成前端信息上传功能

认证chaincode编写

房屋认证信息会发送到相关的政府部门节点，通过查询区块链上的数据得到房屋的认证信息结果。回复内容包括两项认证结果：房产证与身份证的匹配结果和是否可以出租。为了能够测试通过，我们当前仅仅返回测试结果。

```
func (this *House) check(stub shim.ChaincodeStubInterface, args []string) peer.Response {
    // 按顺序获取房产证编号信息和房东身份证信息
    houseId := args[0]
    id := args[1]

    // 依据房产证信编号获取房屋所有者信息
    data, err := stub.GetState(houseId) // 返回结果"id:true(false)"
    if err != nil {
        return shim.Error(fmt.Sprintf("Error getting data %s", err.Error()))
    }

    var result string = ""
    if data != nil {
        var str string = string(data[:]) // []byte结果转换成字符串
        split := strings.Split(str, ":") // 依据":"对结果进行划分

        if split[0] == id {
            result = "true"
        } else {
            result = "false"
        }
        result = result + ":" + split[1]
        return shim.Success([]byte(result))
    }
    // TODO 由于没有记录数据,此处我们使用模拟数据
    return shim.Success([]byte("true:true"))
    //return shim.Success([]byte("false:false"))
}
```

后端房屋信息认证

```

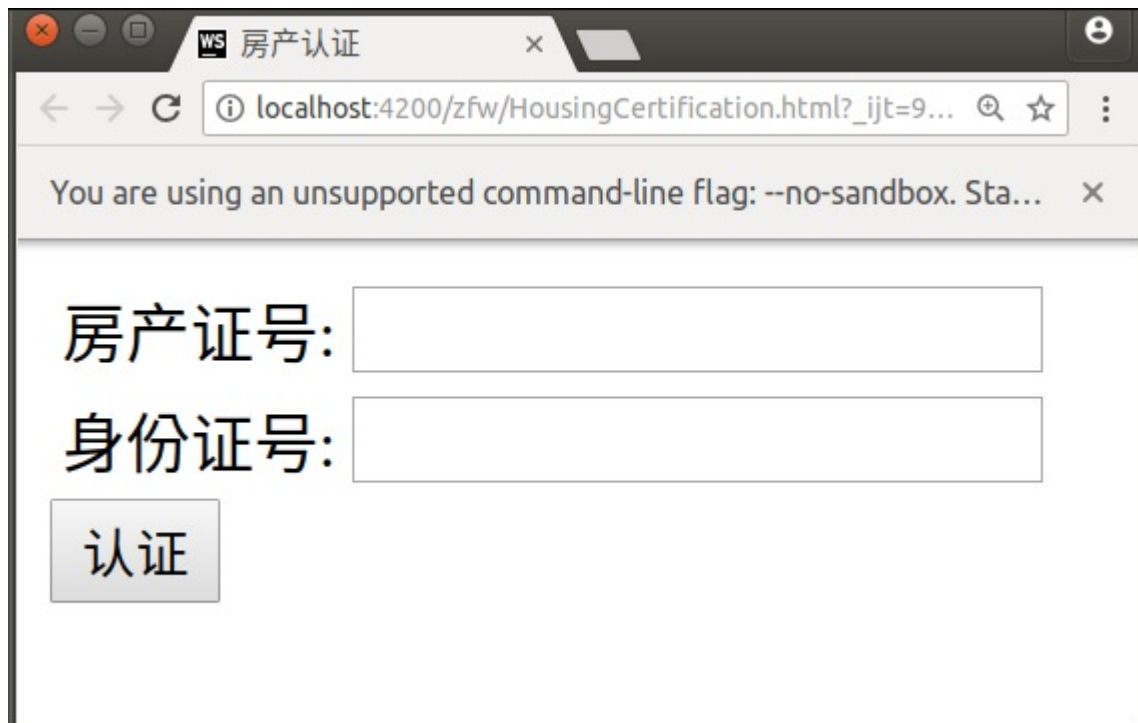
func (this *CertificationController) Check() {
    houseId := this.GetString("houseId")
    id := this.GetString("id")
    if houseId == "" || id == "" {
        handleResponse(this.Ctx.ResponseWriter, 400, "Request parameter houseId(or id) can't be empty")
        return
    }
    beego.Info(houseId + "::" + id)
    args := [][]byte{[]byte(houseId), []byte(id)}

    var (
        channelId      = beego.AppConfig.String("channel_id_fgj")
        chainCodeName = beego.AppConfig.String("chaincode_id_house")
    )
    ccs, err := models.Initialize(channelId, chainCodeName, userId)
    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
    defer ccs.Close()

    response, err := ccs.ChaincodeQuery("check", args)
    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
    handleResponse(this.Ctx.ResponseWriter, 200, response)
}

```

前端房屋信息认证



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>房产认证</title>

  <script src="vue1.js"></script>
  <script src="vue-resource.js"></script>
</head>
<body>

<div id="app">

  <table>
    <tr>
      <td>房产证号:</td>
      <td><input type="text" v-model="houseId"></td>
    </tr>
    <tr>
      <td>身份证号:</td>
      <td><input type="text" v-model="id"></td>
    </tr>
  </table>

  <button v-on:click="check">认证</button>

</div>

</body>
```

```

<script>
  var vm = new Vue({
    el: '#app',
    data: {
      houseId: "",
      id: ""
    },
    methods: {
      check: function () {
        var url = "http://localhost:8080/house"
        var params = {"houseId": this.houseId, "id": this.id}
        this.$http.get(
          url,
          {params: params},
          {emulateJSON: true} // 跨域访问
        ).then(
          function (response) {
            console.log(response)
          },
          function (response) {
            console.log(response)
          }
        )
      }
    }
  })
</script>
</html>

```

上传chaincode编写

```

func (this *House) add(stub shim.ChaincodeStubInterface, args [] string) peer.Response {
    // 记录房产信息,以房产证编号为key
    // houseId:id:record
    if len(args) != 3 {
        return shim.Error("Incorrect number of arguments. Expecting 3")
    }
    houseId := args[0]
    id := args[1]
    record := args[2]

    err := stub.PutState(houseId, []byte(id+"-"+record))
    if err != nil {
        shim.Error(fmt.Sprintf("error in commit account info %s", err.Error()))
    }

    return shim.Success(nil)
}

```

chaincode更新

后端上传

```
func (this *CertificationController) RecordAuth() {
    beego.Debug("receive file")

    //file, 这是一个key值, 对应上传Form中的name信息
    file, _, e := this.GetFile("file")
    if e != nil {
        // 如果出现异常了, f内容为nil
        handleResponse(this.Ctx.ResponseWriter, 400, e.Error())
        return
    }
    defer file.Close()

    var (
        channelId    = beego.AppConfig.String("channel_id_fgj")
        chainCodeId  = beego.AppConfig.String("chaincode_id_house")
    )
    // 写操作
    ccs, err := models.Initialize(channelId, chainCodeId, userId)
    if err != nil {
        handleResponse(this.Ctx.ResponseWriter, 500, err.Error())
        return
    }
    defer ccs.Close()

    reader := csv.NewReader(file)
    var isTransactionFailed bool
    var failedTransactionMessage string = "Transaction failed : "
    for {
        str, err := reader.Read()
        if err == io.EOF {
            // 文件结尾
            break
        }
        if err != nil {
            beego.Error(err)
        }
        if len(str) != 3 {
            // 读取到的信息长度错误
            beego.Error(str, "write error")
            continue
        }
        var args [][]byte
        for _, data := range str {
            args = append(args, []byte(data))
            // fmt.Print(data, "\t")
        }
        // fmt.Println()
        _, e := ccs.ChaincodeSet("add", args)
        if e != nil {
            isTransactionFailed = true
        }
    }
}
```



```

        failedTransactionMessage = failedTransactionMessage + strs[0] + " "
        beego.Error(strs, "write error:", e.Error())
    }
}

if isTransactionFailed {
    handleResponse(this.Ctx.ResponseWriter, 400, failedTransactionMessage)
}
handleResponse(this.Ctx.ResponseWriter, 200, "ok")
}

```

前端上传

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>房产管理</title>
    <script src="vue.js"></script>
    <script src="vue-resource.js"></script>
</head>
<body>
<div id="app">
    上传房产数据:<input type="file" @change="onUpload"><br>
    <button v-on:click="upload">上传</button>

</div>
</body>
<script>
    var vm = new Vue({
        el: '#app',
        data:{
            formData:new FormData()
        },
        methods:{
            onUpload(e){
                this.formData.append('file', e.target.files[0])
                this.formData.append('type', 'csv')
            },
            upload:function () {
                var url ="http://localhost:8080/house"
                this.$http.post(url,this.formData).then(
                    function (response) {
                        console.log(response)
                    },
                    function (response) {
                        console.log(response)
                    }
                )
            }
        }
    })
})

```

```
</script>
</html>
```

征信认证

学习目标：

独立完成征信认证的功能

个人征信信息包含的信息量过大，我们仅仅通过征信平台给出征信的几个评价级别。

（一）正常：

借款人能够履行合同，不存在任何影响借款人按时足额偿还贷款的因素。金融机构对借款人的还款能力有充分的把握，贷款损失概率为0

（二）关注：

尽管借款人目前有能力偿还贷款本息，但存在一些可能对还款产生不利影响的因素，如果这些因素继续下去，借款人的偿还能力受到影响，贷款损失的概率不会超过5%

（三）次级：

借款人的还款能力出现明显问题，开始有逾期，且完全依靠其正常营业收入无法足额偿还贷款本息，需要通过处理借款人的部分资产甚至执行抵押担保来还款，贷款损失率达到了30%-50%

（四）可疑：

借款人无法足额偿还贷款本息，即使执行抵押或担保，也肯定造成较大损失。金融机构为了挽回资金的损失，会动用清收队伍，为借款人的资产进行重组、合并、兼并、抵押物处理等方式来保证资金的回笼情况，即使损失和浪费了人力成本，也要讨回应得的债务，贷款损失的概率高达50%-75%

（五）损失：

借款人已无法偿还本息，无论采取什么样的措施和程序，贷款注定要失去，或者收回极少部分的资金，从金融机构的角度来看，没有必要保留在资产的账目上，所以很多时候银行、担保公司为了避免出现不良贷款，最好的方式是走法律程序，把坏账转移注销。其贷款的损失率占到75%-100%

发送个人身份证号，可以查询个人的征信级别信息。区块链记录的是身份证号与评价级别。

合同与租金

三个组织会共同组成一个channel，用于记录租房合同信息、租金交易数据等。出现纠纷时所有的历史数据都可以在区块中查询。

合同上链

在终端生成的合同图片信息，为防止篡改，可以将图片进行摘要处理，然后将图片的摘要信息记录到区块链上，当需要取证时，只需要检验合同图片的摘要信息与区块中记录的信息是否相同即可。

数据存储

- 1、每一个订单都会对应一份合同，当租户和房东签名的操作完成后会生成一张合同的图片，我们会将图片做摘要处理（SHA256），然后将订单编号与合同图片的摘要信息组成一个键值对存储在区块链中。
- 2、合同的原始图片信息可以存储在独立的服务器上，也可以放到第三方的云平台上，比如：华为云的对象存储服务OBS。

图片上传

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>租赁合同</title>
  <script src="vue.js"></script>
  <script src="vue-resource.js"></script>
</head>
<body>
<div id="app">
  上传合同图片:<input type="file" @change="onUpload" accept="image/jpeg,application/pdf"><br>
  <button v-on:click="upload">上传</button>
</div>
</body>
<script>
  var vm = new Vue({
    el: '#app',
    data: {
      formData: new FormData()
    },
    methods: {
      onUpload(e) {
        this.formData.append('file', e.target.files[0])
      },
      upload: function () {
        var url = "http://localhost:8080/contract"
        this.$http.post(url, this.formData,
          {
            headers: {'Content-Type': 'multipart/form-data'}
          }
        ).then(
          function (response) {
            console.log(response)
          },
          function (response) {
            console.log(response)
          }
        )
      }
    }
  })
</script>
</html>
```

摘要处理

记录区块

交易数据上链

在支付完成后，应用会将交易的记录信息写入区块。订单编号为键，总金额、房租、押金等作为值记录到区块中。

数据上传

记录区块