# 环境

## 1.1 Andrioid环境

Android版本>21也就是5.1版本

## 1.2 java 环境

JDK>=1.8

## 1.3 gradle 添加依赖

gradle方式:

Step 1. Add the JitPack repository to your build file  
  
gradle  
Add it in your root build.gradle at the end of repositories:  
  
 allprojects {  
 repositories {  
 ...  
 maven { url 'https://www.jitpack.io' }  
 }  
 }  
Step 2. Add the dependency  
  
 dependencies {  
 implementation ‘com.github.TokenTM:tks\_android\_aar:0.4.2\_test’

implementation ‘com.github.TokenTM:tks\_android:0.3.4\_test’  
 }

最新版本请参考github

## 1.4 文档与Demo

demo地址:<https://github.com/TokenTM/tks_android> [点击查看](https://github.com/TokenTM/tks_android)

详细java文档 <https://github.com/TokenTM/tks_android_aar/blob/master/doc/htmldoc.zip>

# 2. 架构说明

统一使用XXF框架,点击->[文档地址](https://github.com/NBXXF/xxf_android)

client对象：TokenTmClient

使用方式:

TokenTmClient.getService(RepoService.class)  
 .xxxFunction()

.subscribe();

已经提供的RepoService

## 2.1. RepoService 服务

//did服务

DIDService;

//私有数据存储

StoreService;

//认证服务

CertService;

//文件服务

FileService;

//身份密码服务

IdentityService;

//上链服务

ChainService;

//物权转移记录服务

CommodityService

//node 节点服务

NodeSrvice;

//node 在线加解密服务

NodeEncryptSrvice;

...更多参考sdk源码

## 2.2. 具体使用方式

TokenTmClient.getService(DIDService.class)...

TokenTmClient.getService(StoreService.class)...

......

# 3. 专业术语

## 3.1 DID是什么？

Did 是区块链唯一身份标识符,类似与中心化系统的userId

## 3.2 身份密码是什么？

身份密码是did 数据安全管理的密码,决定上链操作和私有数据的密钥获取

## 3.3 实名认证是什么？

实名认证是鉴别用户真伪的一种形式,例如购买手机号需要携带身份证

## 3.4 私有数据是什么？

私有数据是用户的财产,确保隐私性,不同于中心化系统保存,中心化系统可能存在泄漏问题（程序员可以看,管理员可以看,数据不一定安全...）,私有数据密钥由用户管理,哪些数据可以保存到私有数据服务里面呢？比如:用户的身份证号码？用户的浏览记录？用户的收藏？...

# 4. 身份密码UI模组

身份份密码UI模组,是基于tokentm 样式UI和IdentityPwdService实现的管理身份密码的界面,如果不喜欢这种界面设计,可以自行设计,然后调用IdentityPwdService实现核心逻辑;

注意:身份密码是RepoService的基础,相关服务调用都需要身份密码,确保数据的安全性,所以在调用这些服务之前一定要先设置身份密码,否则将会报错,推荐在中心化系统注册之后立即设置身份密码,当然也可以在具体业务之前设置身份密码;

## 4.1. 设置身份密码



ComponentUtils.launchUserIdentityConfirmActivity(

LoginOrRegisterActivity.this,

binding.loginPhoneEt.getText().toString(),

new Consumer<BindUDID>() {

@Override

public void accept(BindUDID bindUDID) throws Exception {

//TODO 中心化系统和userId进行绑定

DemoSp.getInstance().login(binding.loginPhoneEt.getText().toString(), bindUDID.getDid());

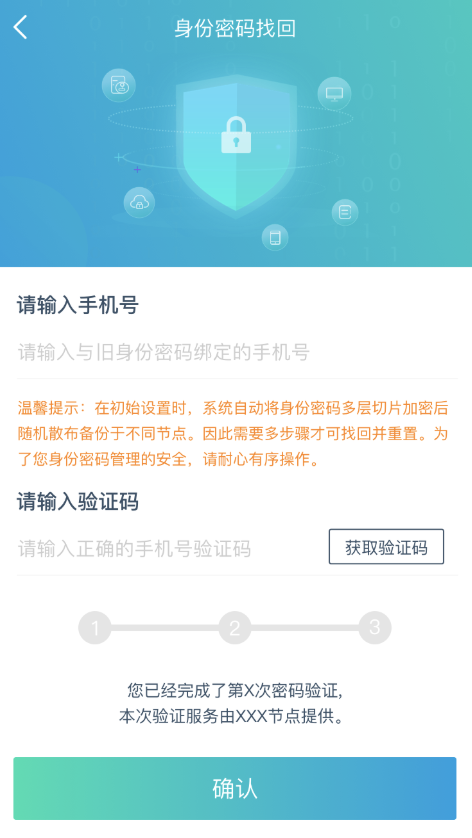
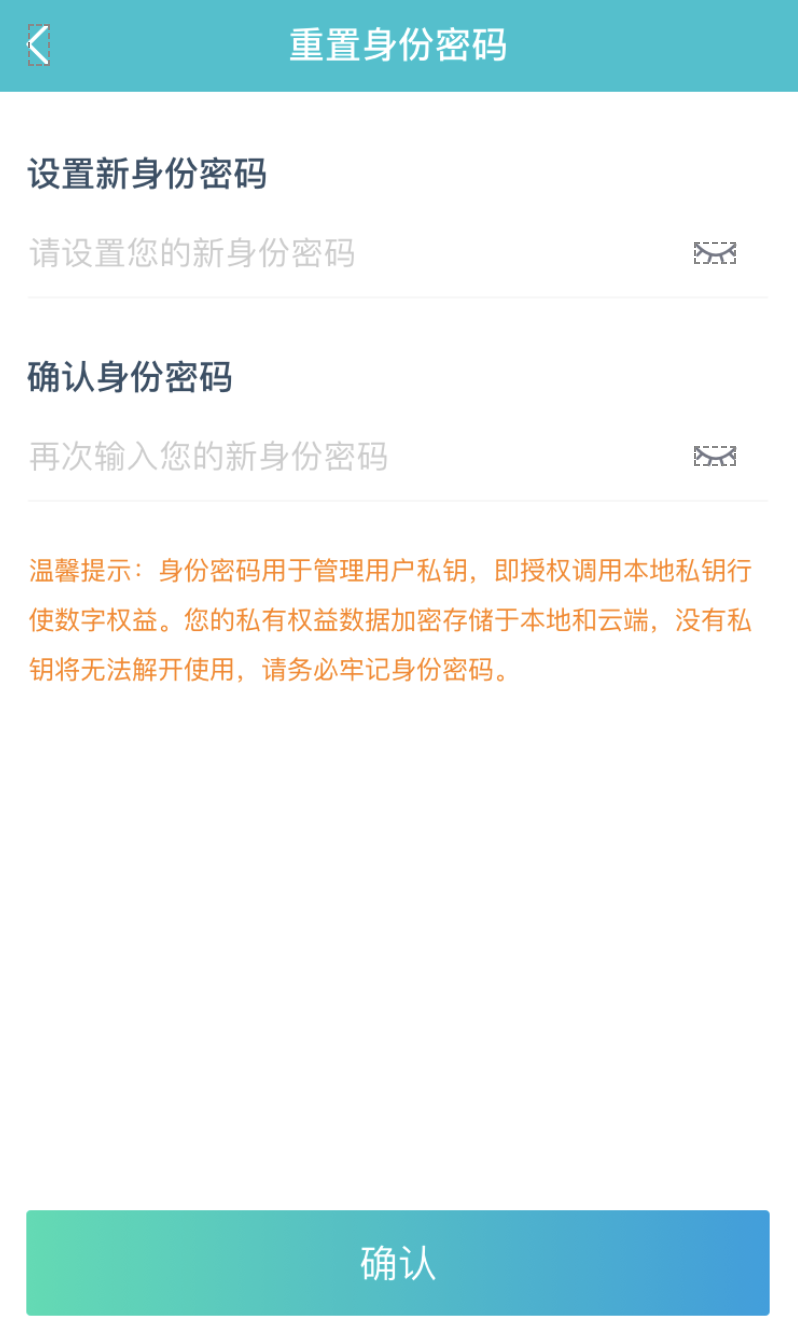
startActivity(new Intent(LoginOrRegisterActivity.this, MainActivity.class));

finish();

}

});

## 4.2 重置身份密码

ComponentUtils.launchForgotIdentityPwd(getActivity(), did, new Consumer<Boolean>() {

@Override

public void accept(Boolean aBoolean) throws Exception {

if (aBoolean) {

ToastUtils.showToast("修改成功");

}

}

});

## 4.3 修改身份密码



ComponentUtils.launchChangeIdentityPwd(getActivity(), did, new Consumer<Boolean>() {

@Override

public void accept(Boolean aBoolean) throws Exception {

if (aBoolean) {

ToastUtils.showToast("修改成功");

}

}

});

## 4.4 身份密码验证



ComponentUtils.showIdentityPwdDialog(

DeliverGoodsActivity.this, did,

new BiConsumer<DialogInterface, String>() {

@Override

public void accept(DialogInterface dialogInterface, String identityPwd) throws Exception {

dialogInterface.dismiss();

//TODO 下一步

}

});

# 5. 认证UI模组

认证UI模组,是基于tokentm 样式UI和CertService实现的用户实名认证和公司法人认证的界面,如果不喜欢这种界面设计,可以自行设计,然后调用CertService实现核心逻辑;

## 5.1. 实名认证



ComponentUtils.launchUserCertActivity(

getActivity()

, new UserCertByIDCardParams.Builder(did)

.setUserName("")

.setUserIDCard("")

.build()

, new Consumer<ChainResult>() {

@Override

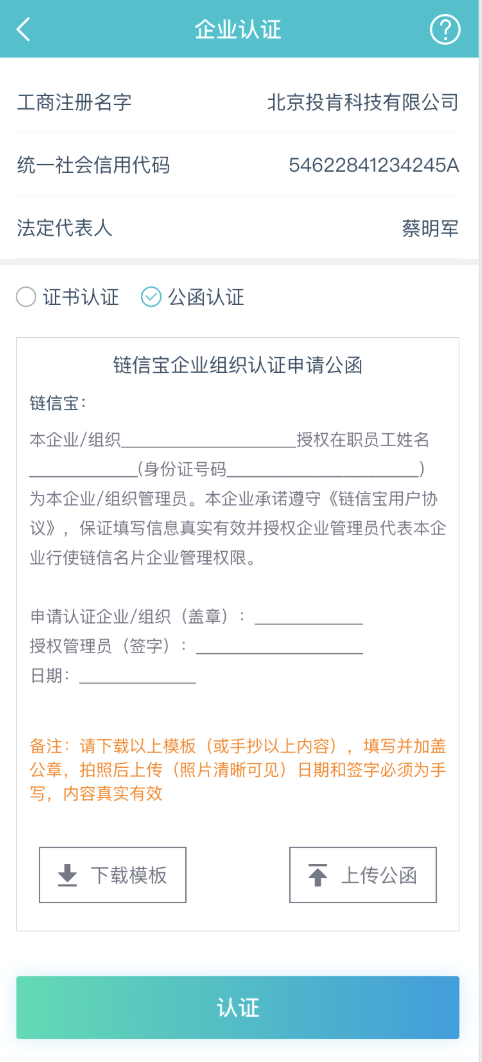
public void accept(ChainResult chainResult) throws Exception {

//中心化自己保存认证信息

}

});

## 5.2. 公司认证

注意: 公司认证前一定要先完成个人身份认证！

ComponentUtils.launchCompanyCertActivity(getActivity()

, new CompanyCertParams.Builder(did, companyName).build()

, new Consumer<ChainCertResult<CompanyCertInfoStoreItem>>() {

@Override

public void accept(ChainCertResult<CompanyCertInfoStoreItem> chainResult) throws Exception {

//存储企业认证信息

}

});

## 5.3. 链信认证(查看自己)



ComponentUtils.launchChainCertificationActivity(getActivity(),

uTxHash, cTxHash, uDid, cDid, new Consumer<CertificationResultParams>() {

@Override

public void accept(CertificationResultParams certificationResultParams) throws Exception {

//中心化去存储认证信息

}

});

**5.4 链信认证(查看别人)**



ComponentUtils.launchChainCertificationOther(getActivity()

, identityTxHash

, companyTxHash

, identityDid

, companyDid);

**5.5 展示去实名认证弹窗**



ComponentUtils.showIdentityCertificationDialogNotForce(getActivity()

, new UserCertByIDCardParams.Builder(did).build()

, new Consumer<ChainResult>() {

@Override

public void accept(ChainResult chainResult) throws Exception {

//保存认证消息

}

});

**5.6 展示去实名并且企业认证弹窗**



ComponentUtils.showIdentityAndCompanyCertificationDialogNotForce(getActivity()

, new UserCertByIDCardParams.Builder(did).build()

, new CompanyCertParams.Builder(did, "").build()

, new Consumer<CertificationResultParams>() {

@Override

public void accept(CertificationResultParams certificationResultParams) throws Exception {

//保存认证信息

}

});

# 6. 物权转移记录



*ComponentUtils.launchGoodsTransferRecordsActivity(getActivity(), goodsId);*