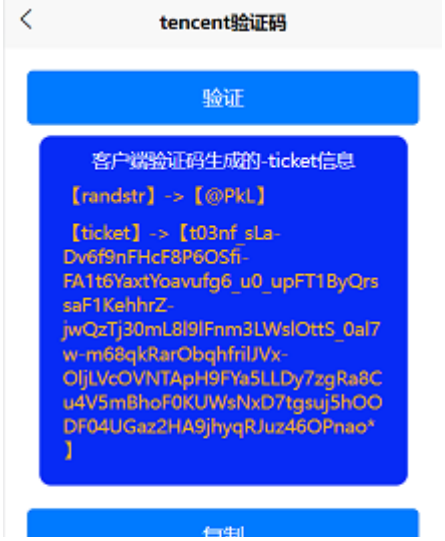
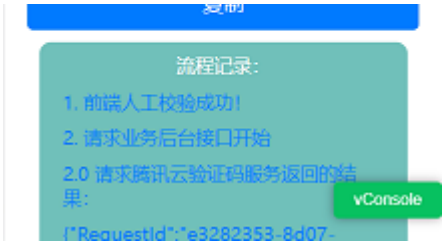
第三方验证码方案

目前实现的验证码demo效果：https://m.zhaoyj.work/msb/#/pages/login/other

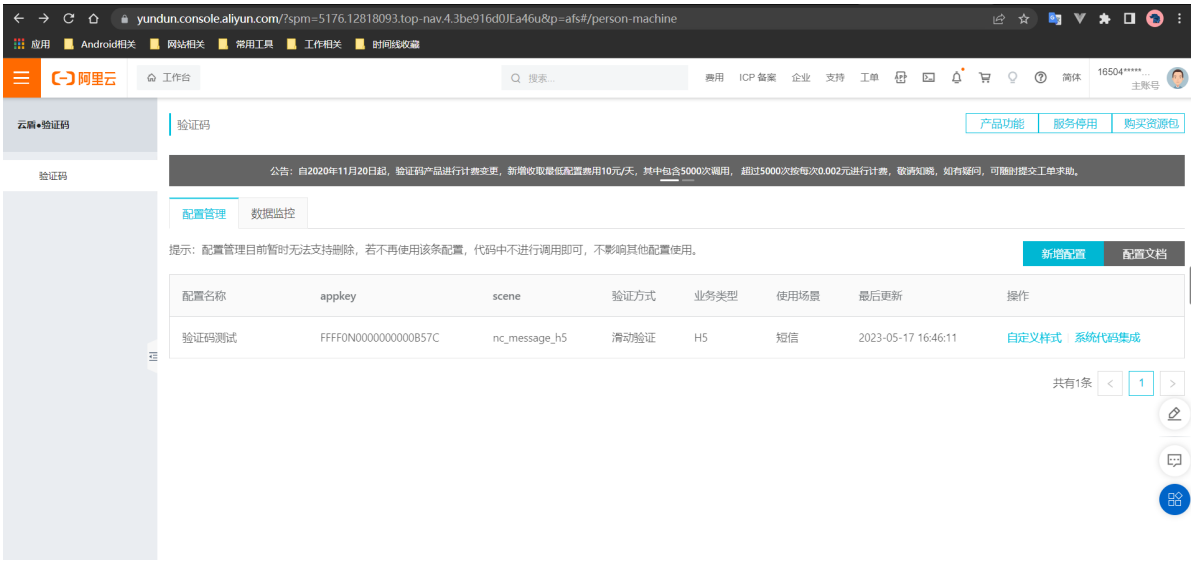

其他测试
打开新的webview测试
调用unicloud函数测试
https-请求接口测试
http-请求接口测试
renderjs
顶象验证码校验
腾讯验证码校验


vConsole
tencent验证码
验证
安全验证
拖动下方滑块完成拼图
ROUTE
vConsole


tencent验证码
验证
客户端验证码生成的-ticket信息
【randstr】->【@PkL】
【ticket】->【t03nf_sLa-Dv6f9nFHcF8P6OSfi-FA1t6YaxtYoavufg6_u0_upFT1ByQrssaF1KehhrZ-jwQzTj30mL8l9lFnm3LWslOttS_0al7w-m68qkRarObqhfrilJVx-OljLVcOVNTApH9FYa5LLDy7zgRa8Cu4V5mBhoF0KUWsNxD7tgsuj5hOODF04UGaz2HA9jhyqRJuz46OPnao*】



顶象的集成比较简单效果也一般，腾讯云的目前看着比较完善，阿里云的配置及文档比较简单(后台没有提供nodejs的sdk，没有接入测试)，百度的没有发现有相关功能。



一、参考网站

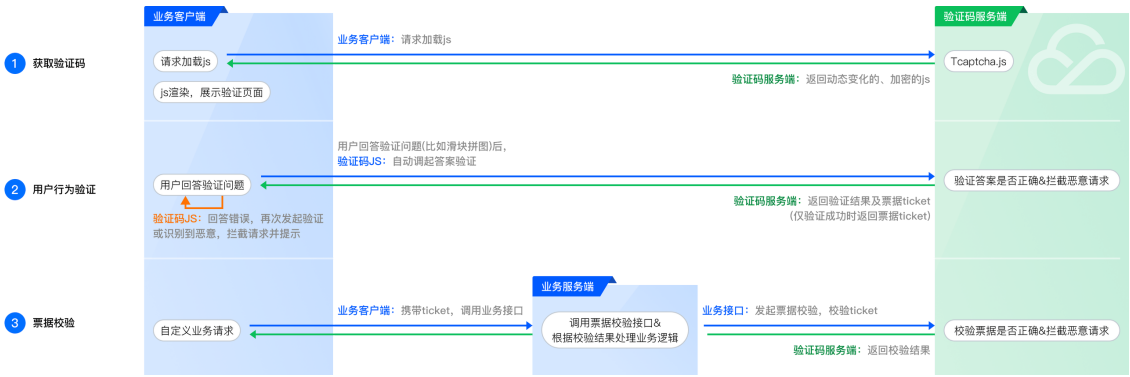
[验证码平台哪家强，六大验证码使用评测-2017]

[产品防护：5种常见的短信验证码防刷策略-2017]

[验证码哪家好？主流验证码全面对比-2022]

二、验证码校验流程

- 腾讯验证码调用时序图



- 顶象验证码校验流程

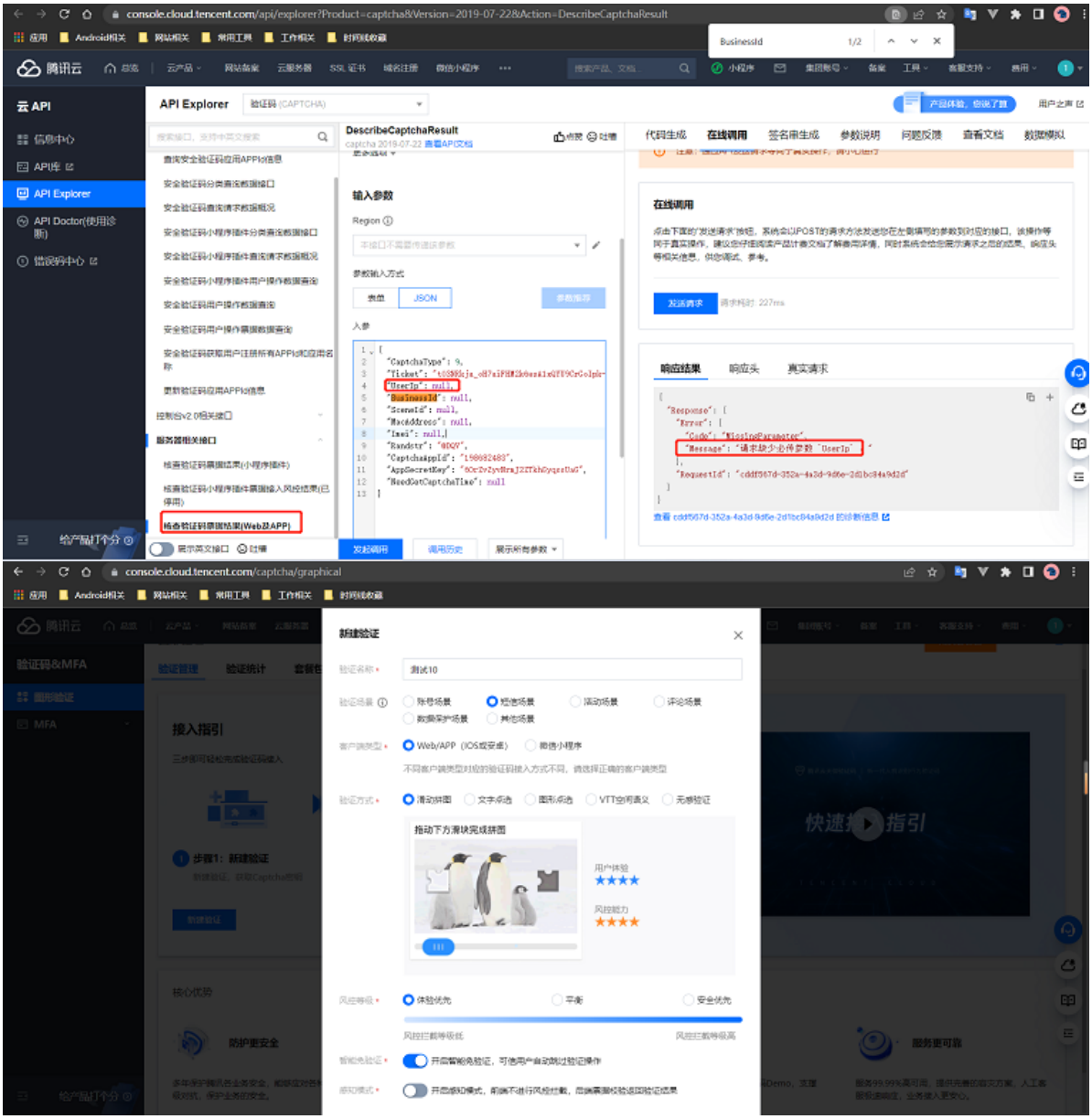
简单说来：
1.【验证码客户端组件】实现的人工校验，校验通过生成一个token
2.用户请求自己的业务接口时使用token等信息先调用【验证码后台服务接口】判断是否是客户端生成的
3.校验通过再执行自己的相关业务逻辑

三、代码实现流程

- 顶象技术开发文档：
  https://www.dingxiang-inc.com/docs/detail/captcha#doc-h3-20
- 腾讯技术开发文档：
  https://cloud.tencent.com/document/product/1110/36841

使用腾讯云验证码过程

后台配置相关：



一、vue前端代码集成

```
//页面按钮触发验证码功能
<button type="primary" @click="verifyYzm" style="margin-top: 20px">验证</button>


export function loadScript(url) {
  return new Promise((resolve, reject) => {
    const script = document.createElement('script')

    script.onload = () => resolve("load success")

    script.onerror = () => reject(new Error(`Load script from ${url} failed`))

    script.src = url
    const head =
        document.head || document.getElementsByTagName('head')[0]
    ;(document.body || head).appendChild(script)
  })
}
//加载TCaptcha.js
mounted() {
```

```
  if (typeof TencentCaptcha == 'undefined')
    loadScript("https://ssl.captcha.qq.com/TCaptcha.js", 'tencent')
        .then(res => {
          this.loadFinish = true
          console.log("loadScript=success", res)
        })
        .catch(err => {
          console.error("loadScript=error", err)
        })
},
methods: {
  verifyYzm() {
      this.logList.length = 0
      this.ticketInfo.length = 0
      let _this = this
      if (!this.loadFinish) uni.showToast({title: "js文件加载失败！"})

      // 定义回调函数
      function callback(res) {
        // 第一个参数传入回调结果，结果如下：
        // ret         Int        验证结果，0：验证成功。2：用户主动关闭验证码。
        // ticket      String     验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
        // CaptchaAppId       String     验证码应用ID。
        // bizState    Any        自定义透传参数。
        // randstr     String     本次验证的随机串，后续票据校验时需传递该参数。
        console.log('callback:0000000000', res);

        // res（用户主动关闭验证码）= {ret: 2, ticket: null}
        // res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
        // res（请求验证码发生错误，验证码自动返回terror_前缀的容灾票据） = {ret: 0,
ticket: "String", randstr: "String",  errorCode: Number, errorMessage: "String"}
        // 此处代码仅为验证结果的展示示例，真实业务接入，建议基于ticket和errorCode情况做不同
的业务处理
        if (res.ret === 0) {
          // 复制结果至剪切板
          var str = '【randstr】->【' + res.randstr + '】     【ticket】->【' +
res.ticket + '】';
          console.log("callback:11111", str)
          _this.ticketInfo.push('【randstr】->【' + res.randstr + '】')
          _this.ticketInfo.push('【ticket】->【' + res.ticket + '】')
          // _this.requestProcess('1' + res.ticket) //todo 后端校验不通过情况
          _this.logList.push("1. 前端人工校验成功！")
          _this.requestProcess(res.ticket)
        }
      }

      // 定义验证码js加载错误处理函数
      function loadErrorCallback() {
        var appid = '您的CaptchaAppId';
        // 生成容灾票据或自行做其它处理
        var ticket = 'terror_1001_' + appid + '_' + Math.floor(new
Date().getTime() / 1000);
        callback({
          ret: 0,
          randstr: '@' + Math.random().toString(36).substr(2),
```

```
          ticket: ticket,
          errorCode: 1001,
          errorMessage: 'jsload_error'
        });
      }

      try {
        var captcha = new TencentCaptcha('198682483', callback, {});
        // 调用方法，显示验证码
        captcha.show();
      } catch (error) {
        // 加载异常，调用验证码js加载错误处理函数
        loadErrorCallback();
      }
  },
}
```

二、nodejs后端代码集成

```
// /tencent/index.js
const tencentcloud = require("tencentcloud-sdk-nodejs-captcha");
async function verifyTencentToken(token, ip_info) {
  return new Promise((resolve, reject) => {
    const CaptchaClient = tencentcloud.captcha.v20190722.Client;
    // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，此处还需注意密钥对的保密
    // 代码泄露可能会导致 SecretId 和 SecretKey 泄露，并威胁账号下所有资源的安全性。以下代码示例仅供参考，建议采用更安全的方式来使用密钥，请参见：https://cloud.tencent.com/document/product/1278/85305
    // 密钥可前往官网控制台 https://console.cloud.tencent.com/cam/capi 进行获取
    const clientConfig = {
      credential: {
        secretId: "xxx",
        secretKey: "xxx",
      },
      region: "",
      profile: {
        httpProfile: {
          endpoint: "captcha.tencentcloudapi.com",
        },
      },
    };
    // 实例化要请求产品的client对象,clientProfile是可选的
    const client = new CaptchaClient(clientConfig);
    const params = {
      "CaptchaType": 9,
      "Ticket": token,
      "UserIp": ip_info ? ip_info : "192.168.43.222",
      "Randstr": "@bEq",
      "CaptchaAppId": xxx,
      "AppSecretKey": "xxx"
    };
    console.log("verifyTencentToken=========params==========", params)
    client.DescribeCaptchaResult(params).then(
        (data) => {
```

```
          console.log("success", data);
          return resolve(data)
        },
        (err) => {
          console.error("error", err);
          return reject(err)
        }
    );
  })
}
module.exports = {
  verifyTencentToken
}

// index.js
const tencentClient = require("./tencent/index");
// 调用腾讯二维码校验模块功能
tencentClient.verifyTencentToken(ticket, ip_info)
```

四、各平台价格介绍

- 1.腾讯云：天御-验证码服务

次数包	有效期	配置费用
3万次	1年	300.00元
8万次	1年	600.00元
20万次	1年	1,000.00元
100万次	1年	2,000.00元
200万次	1年	2,700.00元
500万次	1年	5,700.00元
1000万次	1年	9,600.00元
5000万次	1年	44,000.00元
1亿次	1年	78,000.00元
5亿次	1年	322,000.00元
10亿次	1年	500,000.00元

- 2.阿里-验证码服务包

次数包	有效期	配置费用
200万次	1年	3,000.00元
500万次	1年	6,000.00元
1000万次	1年	10,000.00元

次数包	有效期	配置费用
5000万次	1年	45,000.00元
10000万次	1年	80,000.00元
50000万次	1年	325,000.00元

- 3.顶象验证码价格

功能	备注解释	基础版	标准版A	标准版B	高级版A	高级版B	企业版
价格	按年付服务费	5800元/年	9800元/年	19800元/年	39800元/年	66800元/年	96000元/年
验证场景	智能场景（通用）/账号类场景/活动类场景/UGC类场景/辅助验证类场景	5类场景	5类场景	5类场景	5类场景	5类场景	5类场景
验证偏好	用户体验优先/平衡/安全优先	3种偏好	3种偏好	3种偏好	3种偏好	3种偏好	3种偏好
验证模式	智能无感模式/强校验模式	2种模式	2种模式	2种模式	2种模式	2种模式	2种模式
二次验证开关	可以选择是否开启或在关闭二次验证	✓	✓	✓	✓	✓	✓
基础验证包	滑动拼图验证/文字点选验证	✓	✓	✓	✓	✓	✓
拓展验证包	图标点选验证/语序点选/刮刮卡验证/空间语义验证/乱序拼图验证/旋转验证	X	图标点选/空间语义	任意3种	任意5种	任意7种	✓（全部）
验证魔方	可手动配置不同验证方式组合，并直观显示当前验证应用下验证判断流程	X	✓	✓	✓	✓	✓
常规平台支持	支持Web/H5、Android4.0+、iOS8.0+	✓	✓	✓	✓	✓	✓
小程序支持	支持微信小程序中的顶象验证码插件	X	✓	✓	✓	✓	✓
验证并发量	每秒支持的最大并发量	100QPS	200QPS	500QPS	1000QPS	1500QPS	2000QPS
每日验证量	每日支持的最大验证累计量	10000次	20000次	50000次	100000次	300000次	50万次
可使用验证应用数	每个验证应用对应一个可接入使用的 APPID	5个	8个	10个	15个	20个	50个
统计数据更新频率	统计数据的更新频率	实时更新	实时更新	实时更新	实时更新	实时更新	实时更新
验证模型更新频率	验证模型的更新频率	每月更新	每月更新	每月更新	每月更新	每周更新	每周更新
支持HTTPS	支持HTTPS加密访问	✓	✓	✓	✓	✓	✓
基础统计数据	1、今日请求量： 验证码总请求次数	✓	✓	✓	✓	✓	✓
	2、今日验证量： 实际提交验证的请求次数						
	3、今日验证成功量： 验证成功的请求次数						
	4、今日验证拦截量： 验证失败的请求次数						
	5、验证趋势图： 支持展示今天/近7天/近15天/近30天/自定义时间区间内的请求量、验证量、验证成功量、验证拦截量的统计趋势						
操作环境检测	1、异常设备识别，包括UA不一致、分辨率不一致、篡改颜色深度、屏幕可高度异常、模拟器特征等超过百维的环境风险维度	✓	✓	✓	✓	✓	✓
	2、IP风险的检测						
轨迹模型检测	1、异常轨迹耗时检测	✓	✓	✓	✓	✓	✓
	2、轨迹重放检测						
	3、动态异常聚类轨迹检测						
	4、直线轨迹、匀速轨迹、异常波动轨迹、异常变速轨迹、异常随机轨迹、贝塞尔曲线拟合等主流针对模拟轨迹的检测手段						
	5、轨迹模型风险分评估，通过海量的轨迹黑白样本数据进行风险评估						