风控行为验证码实现

背景

当前应用内对人机的验证能力不足，为确保对应用内的机器人脚本等恶意用户进行检测并做到对正常用户的打扰和影响最低，需开发对于人机验证的相关能力

需求目的：

1. 实现多种图片类人机验证方式的建设；
2. 实现对不同人机验证下的数据进行埋点监控，为后续对异常用户分析做数据支撑；
3. 实现对机器操作的异常用户实现打断异常的操作行为；

行为验证码

支持的验证码类型：数字计算验证码、滑块还原验证码、滑动还原验证码、文字点选验证码、语序点选验证码、旋转还原验证码、图片选择验证码

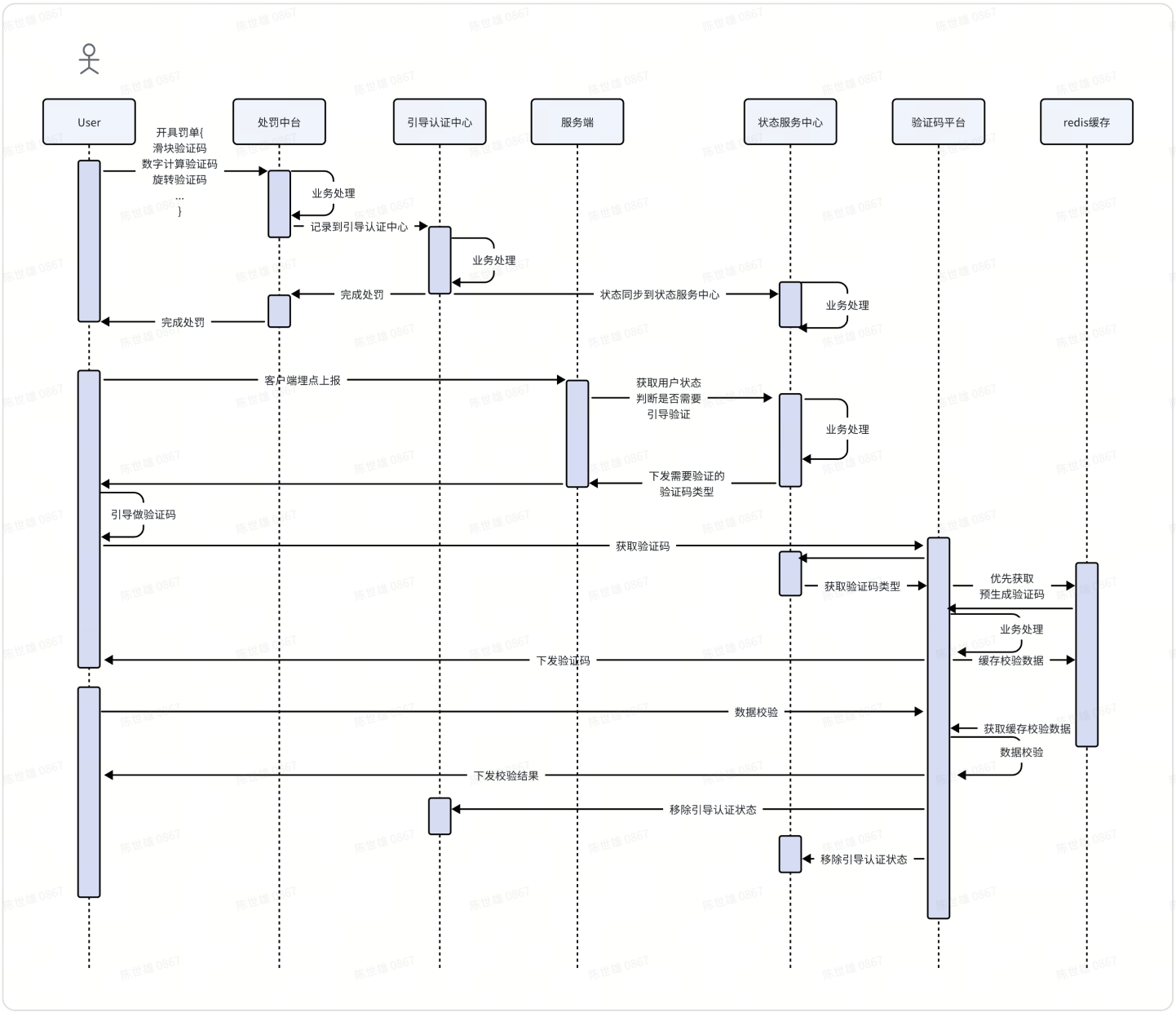
核心功能点：

1.自定义资源库（背景图、文字库、滑块形状等）

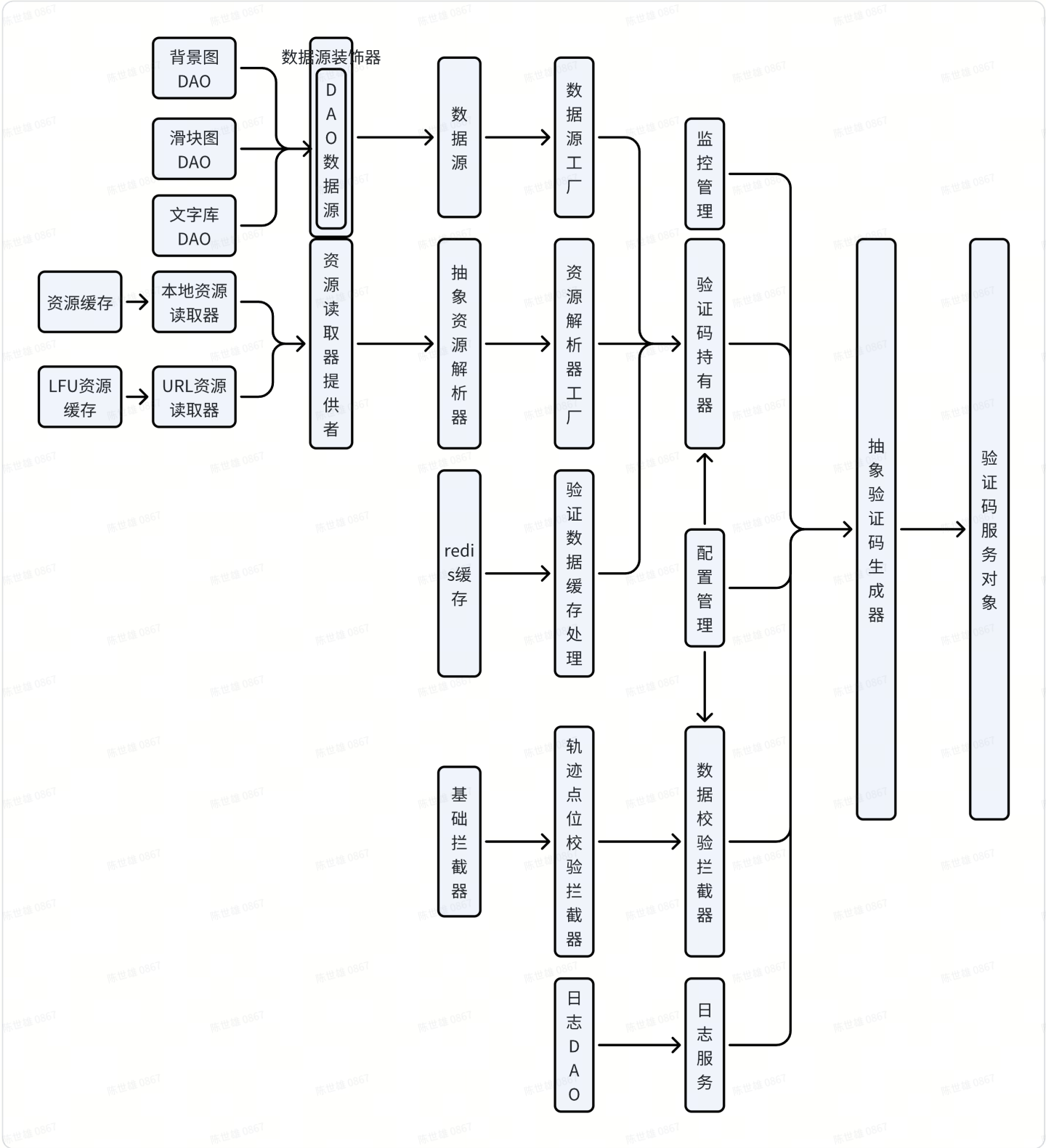
2.验证码生成策略可控

3.自己增加校验规则

验证码流程


User
处罚中台
引导认证中心
服务端
状态服务中心
验证码平台
redis缓存
开具罚单{
滑块验证码
数字计算验证码
旋转验证码
...
}
业务处理
记录到引导认证中心
业务处理
完成处罚
状态同步到状态服务中心
业务处理
完成处罚
客户端埋点上报
获取用户状态
判断是否需要
引导验证
业务处理
下发需要验证的
验证码类型
引导做验证码
获取验证码
获取验证码类型
优先获取
预生成验证码
业务处理
下发验证码
缓存校验数据
数据校验
获取缓存校验数据
数据校验
下发校验结果
移除引导认证状态
移除引导认证状态

验证码架构



验证码服务对象：

用来处理获取验证码的一些业务处理，比如先去状态服务中心获取需要下发的验证码类型等。

抽象的验证码生成器：

抽象类，主要用来定义验证码生成流程，以及校验流程有以下两种实现。

默认验证码生成器：获取验证码的流程中增加了频空，以及校验的后置处理。

1.频控检测等前置处理

2.获取数据源

3.获取解析器

4.从预先生成的验证码队列中获取验证码源数据

5.没有预先生成的数据则使用数据源获取数据源信息

6.用解析器实时生成验证码源数据

7.根据验证码数据源缓存需要校验的数据

8.根据验证码源数据转换成前端需要的数据

9.记录获取日志

10.记录到gfa监控

验证码预生成器：主要用到验证码生成流程，预先生成验证码放入缓存队列中。

1.获取队列信息判断队列大小是否小于1千

2.获取数据源

2.获取解析器

3.使用数据源获取数据源信息

4.用解析器实时生成验证码源数据

5.将验证码源数据放入队列中

验证码持有器：

数据源：获取背景图数据源信息等。

验证码解析器：加载图片资源，切割图片生成所需要的验证码图片资源，将数据转换成base64信息。

数据缓存器：缓存实现，目前是redis，解耦开来，后续可改造成各种缓存实现。

数据校验拦截器：

责任链模式，除了数据校验外会有一些人机操作的判断，通过上传的点位数据进行判断，后续可增加不同拦截器来进行不同的校验。

日志服务：

记录验证码下发日志以及校验结果，校验值对比信息等。

监控管理：

gfa监控数据，目前监控下发验证码量、下发兜底验证码量、校验成功量、校验失败量。

配置管理：

一些配置项，容错值、频控值等。

新增验证码类型

1. **定义类型**

   cn.taqu.risk.punish.service.enums.CaptchaTypeEnum增加验证码类型

2. **创建数据源（service）**

   实现cn.taqu.risk.punish.service.captcha.CaptchaImageResource

   <T *extends* ImageCaptchaResourceInfo> T getImageResource(String imageType, Integer appCode, Integer cloned);

```
1  @Override
2  @SuppressWarnings("unchecked")
3  public <T extends ImageCaptchaResourceInfo> T getImageResource(String
   imageType, Integer appCode, Integer cloned) {
4      ImageCaptchaResourceInfo resourceInfo = new ImageCaptchaResourceInfo();
5      int size = 1;
6      if
   (TextClickCaptchaImageResourceAdapter.TEXT_CLICK_TYPE.equals(imageType)) {
7          //避免一条数据只录入一个字的情况
8          size = 10;
9      }
10     List<CaptchaResourceInfo> captchaResourceInfos =
   captchaResourceInfoDao.randomGetResourceInfoByType(imageType, appCode,
   appCode, size);
11     resourceInfo.setBackgroundImage(captchaResourceInfos.get(0).getResInfo());
12     return (T) resourceInfo;
13 }
```

3. **创建资源适配器**

   实现cn.taqu.risk.punish.service.captcha.CaptchaImageResource

   目的：各种验证码需要的资源类型不同，有些只需要背景资源、有些需要背景、文字和滑块资源，一个资源接口只加载一种数据资源，所以需要将资源转换成需要的资源类型和格式。

```
1  @Override
2  @SuppressWarnings("unchecked")
3  public <T extends ImageCaptchaResourceInfo> T getImageResource(String
   imageType, Integer appCode, Integer cloned) {
4      TextClickCaptchaResourceInfo resourceInfo = new
   TextClickCaptchaResourceInfo();
```

```
    ImageCaptchaResourceInfo backgroundResource = this.backgroundResource.getImageResource(BACKGROUND_IMAGE_TYPE, appCode, cloned);
    ImageCaptchaResourceInfo wordImage = this.backgroundResource.getImageResource(TEXT_CLICK_TYPE, appCode, cloned);
    Set<String> wordSet = new HashSet<>(8);
    //不重复随机
    List<String> strings = Arrays.stream(wordImage.getBackgroundImage().split("")).distinct().collect(Collectors.toList());
    //字数限制
    int textClickCount = Math.min(Integer.parseInt(behaviorCaptchaConfig.getTextClickCount()), MAX_COUNT);
    textClickCount = Math.max(MIN_COUNT, textClickCount);
    if (strings.size() < textClickCount) {
        throw new BusinessException("请联系管理员配置验证码资源[" + imageType + "]");
    }
    while (wordSet.size() < textClickCount) {
        int nextInt = ThreadLocalRandom.current().nextInt(0, strings.size());
        wordSet.add(strings.get(nextInt));
    }
    resourceInfo.setResourceType(UrlResourceProvider.RESOURCE_TYPE);
    resourceInfo.setType(imageType);
    resourceInfo.setBackgroundImage(backgroundResource.getBackgroundImage());
    resourceInfo.setContent(String.join("", wordSet));
    return (T) resourceInfo;
}
```

4. 资源适配器放入spring容器管理

需要将适配器和资源服务关联起来，如果是直接读取本地资源的话，就不需要数据源。

```
@Bean("wordOrderClickCaptchaImage")
public WordOrderClickCaptchaImageResourceAdapter createWordOrderClickCaptchaImageResourceAdapter(@Qualifier("captchaResourceService") CaptchaImageResource captchaImageResource) {
    return new WordOrderClickCaptchaImageResourceAdapter(captchaImageResource, captchaImageResource);
}
```

5. 资源适配器放入数据源工厂里

修改数据源工厂
cn.taqu.risk.punish.service.captcha.factory.CaptchaImageResourceBeanFactory

优化点：将资源适配器以属性的形式注入到解析器里，不以工厂的形式获取。

6. **创建验证码解析器**

继承cn.taqu.risk.punish.service.captcha.AbstractCaptchaImageParser

```
1
2  /**
3   * 生成验证码源数据
4   */
5  @Override
6  public CaptchaInfo doGenerateCaptchaInfo(TextClickCaptchaResourceInfo
   resource) {
7      return null;
8  }
9
10 /**
11  * 获取验证码校验数据，会缓存在redis中
12  * @param resource 数据源
13  */
14 @Override
15 public Map<String, Object> getCaptchaCacheData(TextClickCaptchaResourceInfo
   resource) {
16     return null;
17 }
18
19 /**
20  * 获取验证码预生成队列key
21  */
22 @Override
23 public String getPreBuildCaptchaCacheKey() {
24     return null;
25 }
26
27 /**
28  * 获取验证码校验数据key
29  */
30 @Override
31 public String getCaptchaValidCacheKey(Integer appCode, Integer cloned,
   String uuid) {
32     return null;
33 }
34
35 /**
```

```
36   * 包装数据，将不需要下发的数据剔除掉，只下发需要的数据
37   * @param captchaInfo验证码源数据
38   */
39  @Override
40  public CaptchaInfoVO getGenerateResultData(TextClickCaptchaInfo captchaInfo)
     {
41      return null;
42  }
43
44  /**
45   * 校验数据
46   * @param captchaCache 缓存的校验数据
47   * @param dto 前端上传的校验数据
48   */
49  @Override
50  public boolean validCaptchaData(Map<String, Object> captchaCache,
    BehaviorCaptchaValidDTO dto) {
51      return false;
52  }
```

7. 验证码解析器放入工厂里

修改cn.taqu.risk.punish.service.captcha.factory.CaptchaImageParserBeanFactory

优化点：将简单工厂抽象出来，每种验证码类型有自己的工厂，提供自定义能力。