# 爱旅行项目—Token 技术分析

爱旅行项目架构设计采用前后端分离，并采用分布式的架构设计（具体可参看《爱旅行系统架构设计及部署策略分析文档》），通过定义接口 API，与前端进行数据交互，前端使用 html 进行数据展现。若加入移动端（Andriod、Ios）实现，可直接使用 API 接口即可。

由于该项目进行前后端分离，Session 就没有意义了，并且移动端开发也是无法使用 Session 的。那么需要使用 Token 进行 session 的管理，通过搭建一个认证（Auth）系统负责用户身份验证，并进行整个系统 Token 的维护与管理。具体的设计方案如下：

## 1.1 用户表设计

该系统除了用户自注册还涉及到第三方的接入，若用户使用微信登录（第三方登录）成功后，则需要进行账号合并，进行数据同步。具体业务流程如下：

对于第一次登录系统的用户（不进行注册，直接通过微信登录进入），那么系统需要给他生成一个临时账号（userCode）和临时密码（userPassword），并且用户表需要记录微信 ID（微信接口返回），以便于该微信用户下次登录系统时继续使用微信登录而不绑定注册（手机或者邮箱）。并且登录成功之后，需要把当前用户信息放入到 Token 中进行统一管理。那么用户表设计必须包含如下字段：

1. id：主键 ID
2. userType：用户类型（标识：0 自注册用户 1 微信登录 2 QQ 登录 3 微博登录）
3. userCode：（若是第三方登录，系统将自动生成唯一账号、密码；自注册用户则为邮箱或者手机号）
4. userPassword：用户密码
5. flatId：（<自注册用户>用户表主键 ID、<第三方登录用户>微信 ID、QQID、微博 ID，注：该字段是微信账号数字化的唯一标识，Token 使用），

那么当第二次使用第三方登录（没有进行账号绑定），此时校验用户身份，需要使用微信号（weChat）和微信 ID（flatID）联合校验（为了避免不同平台返回的平台 ID 出现一致，不唯一的情况）

所以不管第三方登录还是自注册用户登录（通过手机号、邮箱），Token 里面放的数据结构内容需要一致，并且在 Auth 系统中需要实现自有平台（爱旅行）的 Token 维护以及第

三方（比如：微信、QQ 等）的 Token 维护。

补充说明：1 期项目暂不实现第三方登录（如：微信登录）。

## 1.2 Token 的数据结构及内容

Token 的数据结构为 Key-Value，具体内容如下：

1. Key：token，其设计原则：必须保证在整个系统中唯一存在

   根据不同客户端(PC、移动)，为了便于统一管理和维护，token 生成算法设计如下：

   PC 端：token:PC-USERCODE[加密]-USERID-CREATIONDATE-RONDEM[6 位]

   移动端：token: MOBILE-USERCODE[加密]-USERID-CREATIONDATE-RONDEM[6 位]

2. Value：存储登录用户的信息数据（数据格式为 json），具体内容包括：

   1> id：用户表主键 ID

   2> userCode：登录账号

   3> userName：用户昵称

   4> userPassword：登录密码

   5> userType：用户类型

   6> flatId：平台 ID

   7> activated：是否激活（0：否，1：是）

## 1.3 Token 有效期的维护

基于系统的安全性考虑，需要设置 Token 的有效期，并且为了维护 Token 的有效期，须把 Token 放入到 Redis 里进行维护管理。对于不同客户端（PC 端、移动端）的 Token 所设置的有效期策略不同。

### 1.3.1 PC 端

Token 的有效期为 2 个小时，若 2 个小时内没有进行 Token 置换的话，就会自动在 Redis 里清除该 Token，那么当该用户再次发送请求时，则会提示：Token 失效，请重登录。此处应注意：前端须自行管理 Token 的生命周期，原因是 Token 存在 cookie 里，web 的安全性较差。

### 1.3.2 移动端

Token 永不失效，修改密码后须更换 Token。

注意：由于移动端的 Token 一般不需要过期，只有当在 PC 页面进行个人密码修改后，移动端才会退出重登录，或者当在移动端修改密码操作，用户也不需要退出重登录，直接在 Redis 中更新该 Token 中用户修改的新密码即可。

补充说明：爱旅行项目的一期仅实现为在 Auth 系统里生成 Token 时，根据参数不同（PC/MOBILE）来生成唯一的 Token，存入 Redis 时，设置不同的有效期即可。二期新增修改个人密码功能后，再进一步实现。

## 1.4 AS 系统(Auth System)设计方案

该系统主要负责登录用户身份的验证，登录成功后生成唯一的 Token，并将 Token 存入到 Redis 里进行维护管理，以及 Token 的置换（Reload）等。具体的设计流程如图 1 所示：

北大青鸟 APTECH WE CHANGE LIVES

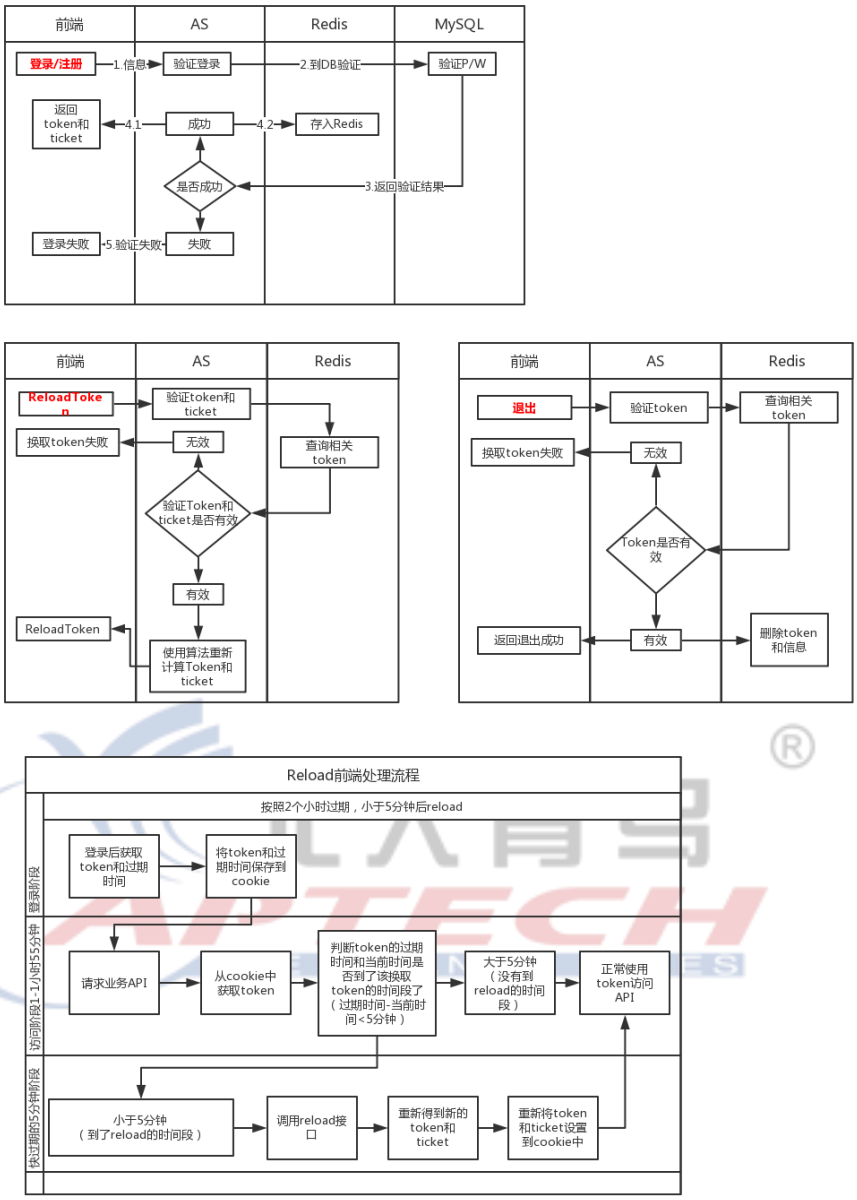


图 1 爱旅行-Token 设计流程

### 1.4.1 生成 Token

当用户进行系统登录时，Auth 系统会进行用户名和密码的校验，验证成功后生成 Token，存入到 Redis 中，如图 2 所示：

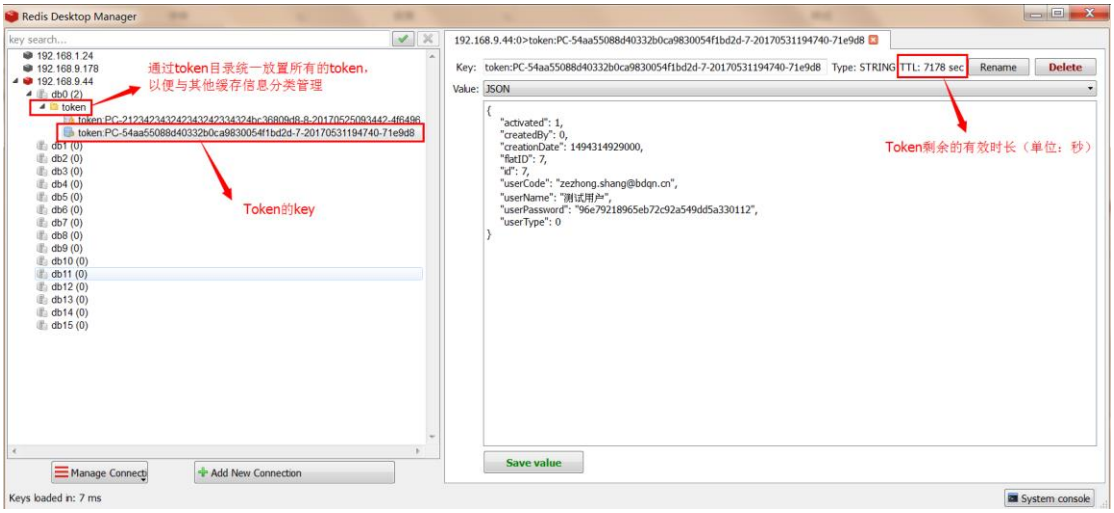


图 2　Redis 中统一管理 Token

同时将该 Token 的 key 返回给前端，前端调用的所有 API 都必须传递该 Token，并规定 Token 统一放在请求头（header）里，后端需要获取当前用户信息时，就可以直接从 header 中获取即可。

为了保证绝对的安全，正常业务下的 API 返回给前端的 Token 信息如下：

```
{
 "status":"200",
 "data":{
    "token":"token:PC-3066014fa0b10792e4a762-23-20170531133947-4f6496",
    "ticket":"yhe736dyfhfyw",
    "expTime":"6474783",
    "genTime":"7565746"
},
 "errorcode":"0",
 "msg":{}
}
```

通常情况下返回给前端的 Token 数据只有四项内容：token(key)、ticket、Token 的生成时间、Token 的失效时间。为了安全考虑，不会包含当前用户的任何信息，每次后端需要获取 Token 信息时，应从 header 中获取到 token(key)和 ticket，通过它们两个去 Redis 中进行 k-v 匹配获取相应的 Token 信息，即：当前用户的相关信息。

此处我们系统设计简单化处理，去掉 ticket。但是 Token 的生成时间和失效时间必须保留，并传递给前端，这是由于仅有失效时间还不足以让前端去判断是否应该进行 Token 的置换，还需要 Token 的生成时间（即：Redis 服务器时间），这也是由于爱旅行项目采用分布式架构，可能会存在服务器时间同步问题。故：前端需要拿到这两个时间进行判断。简单的理解就是：若前端服务器时间若比后端服务器时间晚两个小时，那么前端拿到 Token 就会过期。

## 1.4.2 Token 置换

Token 置换规则定义：前端获取 Token 的 1.5 时后可进行 Token 置换，若在最后的半个小时内，客户端发出请求，则会进行 Token 置换，拿到重新生成的 Token（包括：token(key)、生成时间、失效时间），若客户端在最后的半个小时内没有发送任何请求，那么两个小时后自动过期，即：该 Token 自动从 Redis 里清除，用户须重新登录。

需要注意事项：

1> 不论是最后半个小时的置换时间还是 Token 的 2 个小时有效期，都是根据系统的业务需求所设计的策略方案。

2> 为了防止客户端恶意的进行 Token 置换，需要保证生成 Token 后的 1 个小时内不允许置换。

3> 需要保证客户端传递有效的 Token 进行置换。

4> 为了解决页面的并发问题，在进行置换 Token 时，生成新 Token，但是旧 Token 不能立即失效，应设置为置换后的时间延长 2 分钟。比如：当加载在某个页面（比如：酒店详情页，如图 3 所示）时，该页面中至少有 5 个模块同时进行异步请求（①房型列表、②酒店介绍、③酒店设施、④酒店政策、⑤用户点评），这样必然会存在严重的并发问题，那么在此处，前端必须保证只有一个请求（比如：请求②）去进行 Token 的置换，即：置换 Token 的请求放置只能在页面中被调用一次，那么此时在 Token 的置换期间，该页面的其他请求（请求①、请求③、请求④、请求⑤）必定还是拿着老 Token 进行数据交互，故老 Token 需要继续保持有效，即：它的失效时间必须延长 2 分钟。若非如此，就会出现页面因为 Token 失效而无法完成置换。

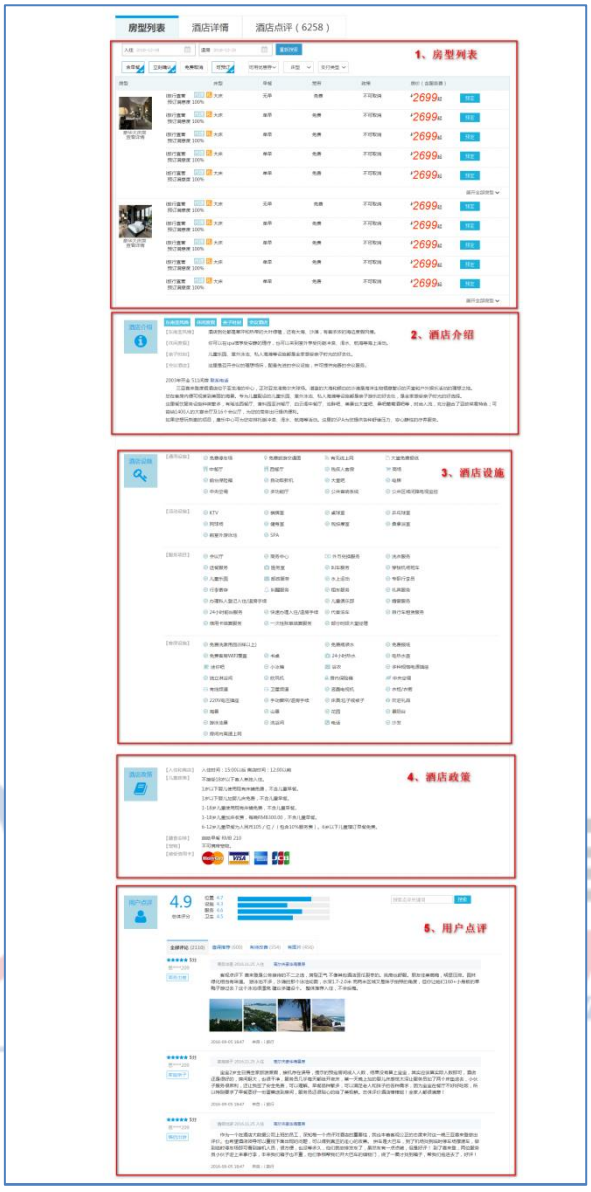


图 3 酒店详情页-多模块异步请求

## 1.5 爱旅行平台的各个业务系统对 Token 的使用

作为爱旅行项目的业务系统，则不需要对 Token 关心过多，只需根据业务需求，从 header 中获取 token，然后在 Redis 中查询相对应的 value 值即可，便可获取到当前用户的相关信息，具体实现如下：

在 Controller 的处理方法中通过 request.getHeader("token")来获取 token 字符串，为了方便进行 Token 的验证，提供统一的 ValidationToken.java，该工具类主要负责通过传入的 token(key) 去 Redis 里进行 value 的查找（ItripUser currentUser = validationToken.getCurrentUser(token);），若找到相应的 value，则返回 currentUser（当前用

户），若无，则返回 null。

## 1.6 前/后端具体实现

### 1.6.1 后端

Auth 系统需要提供 API 如下：

1> 生成 Token

该接口返回的数据内容包括：Token 的 key（注：需要对敏感信息进行加密处理）、Token 的生成时间、Token 的失效时间（注：过期时间减去生成时间一定是两个小时）

2> Token 置换

该接口返回新 Token。实现过程中需要注意如下几点：

a) 生成 Token 后的 1 个小时内不允许置换（注：主要是为了防止客户端恶意的进行 Token 置换）

b) 由于需要保证客户端传递的置换 Token 为真实存在并有效的，故需要在该 API 方法内首先判断 Token 是否有效。

c) 在进行置换 Token，生成新 Token，旧 Token 不能立即失效，应设置为置换后的时间延长 2 分钟。

### 1.6.2 前端

1> 登录成功后，接收 Token 放入 cookie 中，请求的时候从 cookie 中取出放入到 header 里，如下：

```
$.ajax({
   headers:{
       Accept:"application/json;charset=utf-8",
       Content-Type:"application/json;charset=utf-8",
       //从 cookie 中获取
       token:"token:PC-3066014fa0b10792e4a762-23-20170531133947-4f6496"
   },
       type： "post",
       .....
})
```

2> 负责服务器时间同步（根据 API 返回的 Token 生成时间、失效时间进行同步）

3> 置换 Token 需要同步处理，即：保证只有一个请求置换 Token

北大青鸟 APTECH WE CHANGE LIVES