**爱旅行业务和技术要点**

**项目背景**

爱旅行系统是一个 B2C 的管理平台，本系统的主题是旅游，包含 PC 端和手机端 app 两个版 本。所以其应用范围是爱好旅游或者有旅游意向的人群，系统的主要作用是将旅游商品化管理，通过互联网向用户介绍多种旅游套餐和攻略、景区的特点、出行时各地酒店的预订以及车船机票等的网上下单订购。

**技术架构**

爱旅行架构分为：前端架构和后端架构，将前端和后端完全分离，并采用分布式集群的架构设计，通过定义API，与前端进行数据交互，前端使用 html 进行数据展现，并可加入移动端（Andriod、Ios）实现，直接调用 API 即可。

前端架构部署在Nginx上。

后端架构采用SSM框架，多模块化开发，使用 Maven 进行多项目管理。数据库使用了mysql和 Redis。接口测试工具Swagger和Postman。

1. **根据业务进行模块划分：**

itripbackend 分为以下四个 Web 模块：

1. 主业务模块（itripbiz）：包括酒店业务（Quartz）

2. 搜索模块（itripsearch）：爱旅行项目中搜索酒店功能（solr）

3. 登录认证模块（itripauth）：负责用户身份验证，生成并维护 Token

4. 支付模块（itrptrade）：包括支付宝支付、微信支付

以及为 Web 模块提供服务的三个基础模块：itripdao、itriputils、itripbeans。

每个 web 模块均独立部署在单独的 web 服务器上(tomcat)。

1. **技术讲解**

**Nginx:**

轻量级、高性能的WEB服务器/反向代理服务器。只处理静态内容不处理动态内容，动态内容交给后台Tomcat处理。

1. 反向代理

位于用户与目标服务器之间，代理的是目标服务器，即用户直接访问反向代理服务器就可以获得目标服务器的资源。同时，用户不需要知道目标服务器的地址，也无须在用户端作任何特别的设置。

1. 负载均衡

将请求/数据【均匀】分摊到多个操作单元上执行。均衡的分摊压力。

**Maven:**

1. 拥有资源库，本地仓库/远程仓库，pom文件配置想要的jar，自动下载
2. 构建多模块项目分块开发，方便维护与发布

**Spring-SpringMVC-Mybatis：**

**Spring:**

1. 轻量级开源框架
2. 方便解耦，简化开发
   1. 通过Spring提供的IOC容器，将对象之间的依赖关系交由Spring进行控制，避免硬编码所造成的过度程序耦合
3. AOP编码的支持
4. 通过Spring提供的[AOP](https://baike.baidu.com/item/AOP)功能，方便进行面向切面的编程
5. 声明式事务支持、
6. 通过声明式方式灵活地进行事务的管理，提高开发效率和质量。
7. 方便集成各种优秀框架
   1. Spring可以降低各种框架的使用难度，Spring提供了对各种优秀框架

**Spring-MVC:**

* 1. 多视图共享一个模型，提高代码的可重用性
  2. MVC三个模块(视图、控制器、模型)相互独立，松耦合架构
  3. 但原理复杂

**MyBatis:**

1. MyBatis是最简单的持久化框架，小巧并且简单易学。
2. SQL写在XML里，从程序代码中彻底分离，降低耦合度，便于统一管理和优化，可重用。
3. 支持编写动态SQL语句（XML中使用if等标签）。

**数据库:**

数据库分为:关系型数据库和非关系型数据库。

1. 关系数据库：主要用于存储格式化的数据结构，比如 MySQL、Oracle、SqlServer 等都属于关系型数据库。
2. 非关系型数据库：存储的数据都是按照键值对（Key-Value）的结构进行存储。向外提供特定的 API，而不是提供通用的 SQL 接口，所以效率更加高效，非关系型数据库可以不写任何SQL 语句就能实现数据的存储与查询的数据库。数据都是缓存在内存中，并基于内存操作，性能较高。

**接口测试工具：**

更好的实现前后端分离,支持多种语言。

Postman是一个网页调试与发送网页HTTP请求的工具

Swagger API文档与API定义同步更新(swagger只是生成API接口文档 属于web服务)

**Quartz：**

Quartz任务调度框架(在某一个时间点执行什么操作)，三个核心概念：任务、触发器、调度器。任务：需要执行的具体工作(具体业务流程)，触发器：在特定时间触发任务的执行(时间)，调度器：任务实际的执行者,负责粘合任务和触发器。

触发器分为两种SimpleTrigger(固定时间间隔的任务调度，需要设置一些属性)和cronTrigger(定义Cron表达式)。

**SOLR：**

Solr 提供了全文搜索，高亮显示，精确搜索，完善的功能管理界面，增量更新并且支持多种输出格式（包括 XML/XSLT 和 JSON 格式）。solr是一个java搜索引擎服务器。

对比于mysql全文索引和模糊查询来说solr是将数据查询之后保存在了内存中然后我们在内存中查询，我们频繁去数据库查询会加重mysql的负担。并且全文索引在一些版本中不支持中文，使用时在很多场景也不适合。Like模糊查询在少量文本推荐使用但是它并不支持分词效果。

**TOKEN：**

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。Token存储在redis中在一些用户管理方面也比较方便。移动端一般要求永久保存用户登录信息，除非用户注销或重新登录，PC端不需长期保存用户登录信息，redis可以方便处理token的过期和保存。

**登录认证模块(ed-login)**

1. 发送注册码(邮箱注册/手机注册)

输入邮箱/手机号码，判断邮箱/手机号码输入格式是否正确，生成相对应的激活码，发送至邮箱/手机号码，将激活码保存在redis中(在规定时间内实现注册码发送(验证码验证功能)。

1. 用户注册(邮箱注册/手机注册)

输入邮箱/手机号码、密码，验证码,判断验证码是否正确，证手机号码/邮箱输入格式是否正确，密码使用MD5加密，生成用户编号，用户注册成功。

1. 修改密码

输入电话号码、验证码、密码、用户编号,判断验证码是否正确,判断手机格式，密码使用MD5加密，进行修改密码操作。

1. 登录验证

输入邮箱/手机号码和密码，判断邮箱/手机号码输入格式是否正确，验证是否存在用户，生成TOKEN值保存在redis(可规定保存时间)。返回token和token使用时间给客户端。

1. 用户注销

传输TOKEN值，判断TOKEN是否有效给出提示。

1. 第三方登录(QQ)

注册成为开发者(获取APPID和APPKEY)，创建应用，通过审核并获取接口权限。(可下载官网demo)

**讲师模块(**ed-instructor**)**

* 讲师管理

1. 讲师申请

传入讲师信息(申请审核信息)，新增讲师审核表，传入后台进行讲师审核。

* 课程管理

1. 查询相关讲师的课程信息

传入讲师编号和审核状态(9=待审核 10=审核通过 11=审核失败)

1. 新增讲师课程信息

传入课程信息(申请审核信息)，新增课程审核表，传入后台进行课程审核。

* 收益管理

1. 查询讲师详细信息

传入讲师编号查询讲师详细信息。

**支付模块(**ed-payment**)**

1. 微信支付(支付成功修改支付状态，讲师可提金额增加)

微信支付平台提供了几种收款方式(项目选用公众号支付，采用的是微信官方提供的‘模式2’)。输入订单号(根据订单号查询信息)获取订单的详细信息生成微信支付二维码(qrcode.js)。支付成功编写微信异步通知接口(实现业务操作)。

1. 支付宝支付(支付成功修改支付状态，讲师可提金额增加)

利用支付宝官方提供的页面进行测试，根据编号查询订单信息，点击” 浏览器支付” ，进入支付宝支付操作，并接收支付宝接口需要的参数。设置支付宝异步通知信息(实现业务操作)，设置同步地址。

**主业务模块(ed-biz)**

1. 查询用户基本信息。
2. 根据用户编号和订单状态，查询用户订单信息
3. 学习记录

查询用户支付成功的订单信息，由订单信息查询相关的课程详细信息。

1. 课程收藏

根据课程编号和课程名称新增课程收藏信息。

1. 消息推送

**订单模块(ed-order)**

1. 新增订单信息(普通订单/抢购订单)

初始化加载抢购活动的库存信息，点击购买按钮生成订单信息。传入TOKEN、购买课程、秒杀价格、秒杀ID。通过TOKEN值判断用户是否登录，首先通过课程ID查询课程信息，再查询出讲师信息，算出讲师与平台的收入，通过秒杀价格与秒杀ID判断是秒杀订单还是普通订单，生成订单信息。

1. 刷新订单支付情况

每二十分钟执行一次(两小时内没有支付订单状态改为取消状态)。如果是秒杀订单订单状态改为取消状态释放库存。

**搜索模块(**ed-search**)**

1. 查询课程信息(查询全部，带条件，可分页)

**系统管理模块(**ed-system**)**

1. 动态权限认证
2. 广告管理
3. 首页管理(底部导航 ， 文章管理 ， 友情链接 ， 头部导航)
4. 学员接口管理
5. 平台管理
6. 订单信息管理
7. 讲师提现利润
8. 讲师管理

**接口实例**

1. **简要描述**

根据条件判断临时库存表的库存(输入一段时间内的最低库存量)

1. **请求URL**

http://localhost:8080/biz/orderApi/getpreorderinfo

1. **请求方式**

POST

1. **返回格式**

JSON

1. **参数**

|  |  |  |  |
| --- | --- | --- | --- |
| 参数名 | 必选(是/否) | 类型 | 说明 |
| 酒店ID | 是 | Long | 接收数字类型 |
| 房间ID | 是 | Long | 接收数字类型 |
| 入住时间 | 是 | String | 接收日期时间 |
| 退房时间 | 是 | String | 接收日期时间 |
| 预定房间数量 | 是 | Integer | 接收数字类型 |
| 用户认值TOKEN | 是 | String | 用户登录  自动生成 |

1. **输入示例**

**示例1：**

{

"checkInDate": "2020-2-27",

"checkOutDate": "2020-3-2",

"count": 1,

"hotelId": 1,

"roomId": 4

}

TOKEN(参数)

token:PC-1b4258396dc2a89a6b1baec892cb5ee7-73-be1d09

**示例2：**

{

"checkInDate": "2020-2-27",

"checkOutDate": "2020-3-2",

"count": 1,

"hotelId": 1,

"roomId": 4

}

TOKEN(参数)

token:PC-1b4258396dc2a819a6b1baec892cb5ee7-73-be1d09

**示例3：**

{

"checkInDate": "2020-2-27",

"checkOutDate": "2020-3-2",

"count": 21,

"hotelId": 1,

"roomId": 4

}

TOKEN(参数)

token:PC-1b4258396dc2a819a6b1baec892cb5ee7-73-be1d09

1. **返回示例**

**示例1：**

{

"success": "true",

"errorCode": "0",

"msg": "获取成功",

"data": {

"store": 20

}

}

**示例2：**

{

"success": "false",

"errorCode": "30000",

"msg": "请先登录用户",

"data": null

}

**示例3：**

{

"success": "true",

"errorCode": "0",

"msg": "酒店房间库存不足",

"data": null

}

1. **返回状态**

200(成功状态)

**LINUX命令**

本次项目中所用命令

查看jdk: java -version

用户切换: su 用户名(输入密码终端不显示)

解压：tar -zxvf jdk名字

<**dependency**>  
 <**groupId**>org.mybatis.spring.boot</**groupId**>  
 <**artifactId**>mybatis-spring-boot-starter</**artifactId**>  
 <**version**>1.3.2</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-security</**artifactId**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-data-redis</**artifactId**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>redis.clients</**groupId**>  
 <**artifactId**>jedis</**artifactId**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>com.alibaba</**groupId**>  
 <**artifactId**>druid</**artifactId**>  
 <**version**>1.1.10</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-cache</**artifactId**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger2</**artifactId**>  
 <**version**>2.9.2</**version**>  
</**dependency**>  
  
<**dependency**>  
 <**groupId**>io.springfox</**groupId**>  
 <**artifactId**>springfox-swagger-ui</**artifactId**>  
 <**version**>2.9.2</**version**>  
</**dependency**>  
  
*<!--普通校验-->*<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-validation</**artifactId**>  
</**dependency**>