目录

概要设计说明书

版本号	更新日期	更新内容说明	更新人	阅读对象
V1.0	2017/4/30	新增文档	XXX	爱旅行项目开发工程师、爱旅行项目测试工程师

引言

截止到目前为止，我们已经完成了爱旅行项目的需求调研和需求分析，接下来我们将根据爱旅行项目的业务，整理并分析项目中的难易点，并对项目的技术进行选型，项目的数据库进行初步设计。参与本项目开发的产品工程师、开发工程师和测试工程师请细阅本文档。如发现设计中的漏洞、不足，请及时与项目经理进行沟通。

注：如多方沟通，需要修改本文档，在在审阅模式下，对文档进行修改。

1 总体设计

1.1 需求规定

爱旅行项目的详细的需求分析文档请参见《爱旅行系统需求分析说明书.doc》

1.2 项目技术选型

由于爱旅行项目的自身的业务复杂性，在技术选型的时候，需要考虑爱旅行项目中的不同应用场景，秉承“专业的事用专业的的技术”的理念，爱旅行项目的技术选型如表 1 所示。

名称	技术	依赖环境
后端开发语言	Java	JDK1.8+
系统缓冲	Redis	Linux+Redis
反向代理	Nginx	Linux
文档管理	Swagger	
搜索引擎	Solr	JDK1.8+ Tomcat
中文分词器		
软件服务器	Tomcat	Ttomcat8+
邮箱认证	JavaMail	JDK1.8+ Tomcat

数据库	MySQL	Linux+MySQL
项目管理	Maven	Maven3.2+
后端项目框架	SSM(spring MVC+Mybatis)	Spring4.1.7+、MyBbatis3.2.2
前端开发语言	HTML5+CSS3+JavaScript	
前端开发框架	React v15.5.4+	Node v6.10.3+、npm3.10.10+、webpack2.6.1+

表-1 爱旅行项目技术选型

1.3 运行环境

在项目开发的过程中，必须考虑在项目开发完成后，项目要部署在什么环境，这其中包括软件环境和硬件环境。

硬件服务器环境：

选项	最低配置要求	其他说明
处理器型号以及内存容量	1G+	
外存容量硬盘	40G+	建议使用磁盘阵列

表-2 爱旅行项目硬件服务器环境

软件服务器环境:

选项	版本要求	其他说明
JDK	1.8	
Tomcat	7+	建议使用 Tomcat7 或者 Tomcat8
MySQL	5.5+	
Nginx（Tengine）		
IE(-m-)	10.0+	需要增加-m-前缀
FrieFox(-moz-)	5.0+	需要增加-moz-前缀
Chrome(-webkit-)	4.0+	需要增加-webkit-前缀

表-3 爱旅行项目软件服务器环境

1.4 系统结构

根据需求分析中的具体业务需求，把爱旅行项目中的系统主要分为 2 个子系统，后端系统和前端系统。

1.4.1 后端系统结构

其中后端系统包括 4 个子系统分别为:

- 主业务系统 biz:包含系统对外公布的大部分接口(如房间、订单、评论、字典、联系人等)。
- 搜索引擎业务系统 search:包含首页、酒店搜索页、热门酒店等查询接口。
- 认证系统 auth:包含用户认证、换取用户凭证等接口。
- 支付系统 trade:包含用户支付相关接口。

前端系统包括两个子系统，分别为:

- 手机端:用户可以在手机端进行酒店的查询预订等相关操作。
- WEB 端:用户可以在浏览器进行酒店的查询预订等相关操作。

爱旅行项目的结构如图 1 所示。

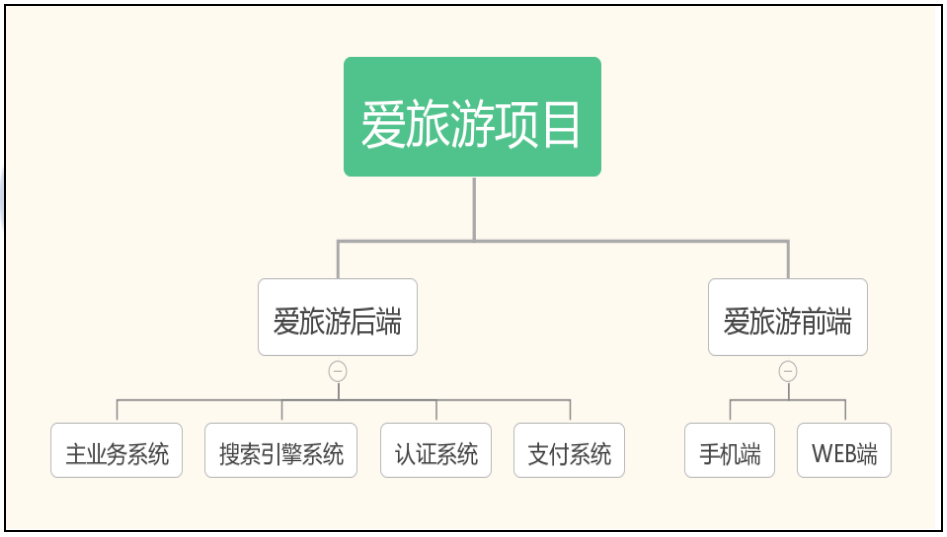

图 1:爱旅行项目结构图

1.4.2 后端系统依赖关系

爱旅行项目的后端程序，统一使用 Maven 集成环境进行统一管理，根据以上项目结构，以及依赖关系，秉承系统耦合性低的原则，将爱旅行项目的模块划分如下:

模块名	内容	依赖	打包格式
itripbeans	pojo:和数据库对应的 JavaBean VO:封装的返回给客		jar

	户端的参数		
itriputils	系统工具类	itripbeans	jar
itripdao	MyBatis 配置文件，Mapper 类	itripbeans、itriputils	jar
itripbiz	service 接口和实现类	itripdao	war
itriptrade	交易模块相关业务	itripdao	war
itripauth	认证模板业务	itripdao	war
itripsearch	搜索模块	itriputils	war

表-4 爱旅行项目依赖关系表

爱旅行项目后台结构如图 2 所示.

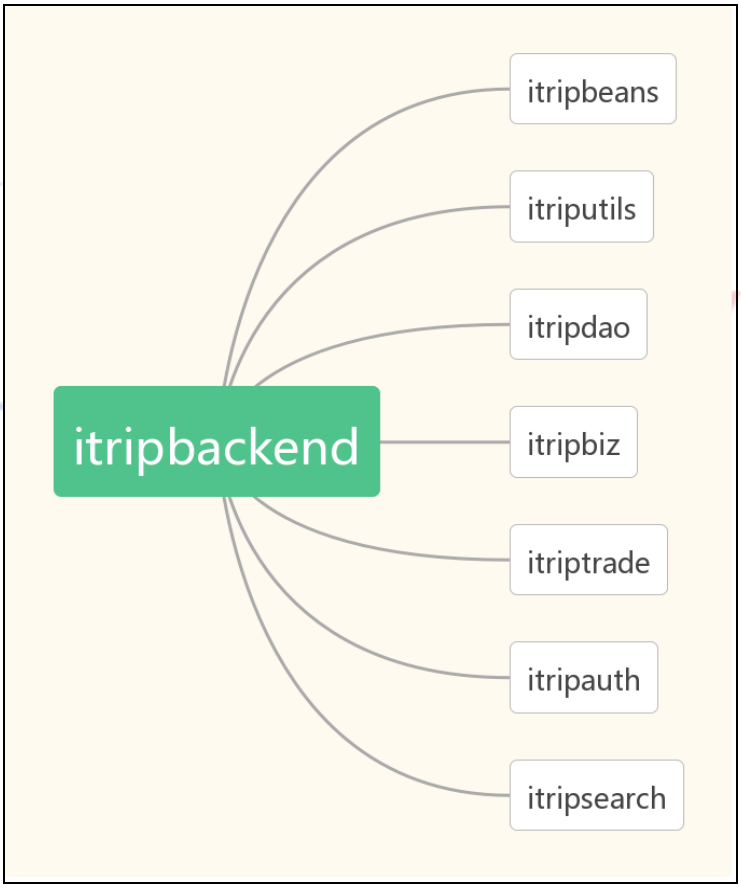

图 2:爱旅行项目依赖关系图

1.4.3 前端系统结构

其中前端 SRC 文件下的系统结构:

- Common ：存储公共的文件（css、js、font、images 文件）

- Components：公共的业务组件和功能组件
- Constants：存放不经常修改的文件，例如：提交或下单请求的地址常量等
- Containers：容器，页面加载需要的容器
- Pages：功能文件地址，包含所有的业务模块的代码
- router: 用于控制页面之间的跳转和页面之间的切换

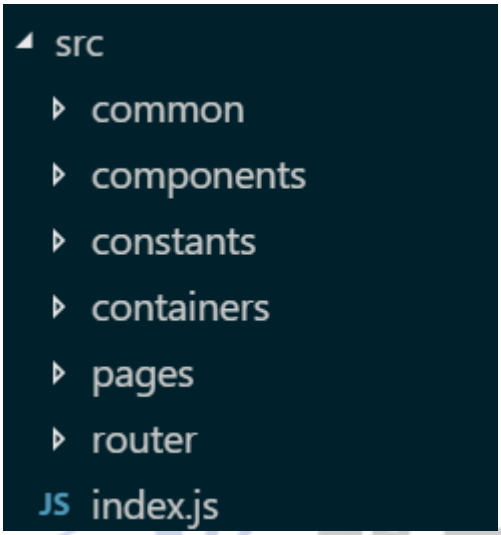

图 3：前端代码结构

1.4.4 前端系统采用技术

采用现在比较流行的 React 技术实现，结合 JSX 和 ES6 语法进行程序的开发和编写，React 实现了虚拟 Dom 的加载方式，提高了界面加载的速度和渲染速度。React 开发时代码可以高度组件化，结合 JSX 和 ES6 语法 可以帮助我们在短时间内快速完成功能的开发，大大的提高了我们的开发效率；

1.4.4.1 React 组件

React 推出后，出于不同的原因先后出现两种定义 react 组件的方式，殊途同归；具体的两种方式：

1、es5 原生方式 React.createClass 定义的组件
2、es6 形式的 extends React.Component 定义的组件

我们这里是有的是第二种方式即 es6 形式的定义组件，接下来我们就来讲解一下第二种方式：

extends React.Component 定义的组件

React.Component 是以 ES6 的形式来创建 react 的组件的，是 React 目前极为推荐的创建有状态组件的方式，最终会取代 React.createClass 形式；相对于 React.createClass 可以更好实现代码复用。将上面 React.createClass 的

形式改为 React.Component 形式如下：

```
class InputControlES6 extends React.Component {
    constructor(props) {
        super(props);

        // 设置 initial state
        this.state = {
            text: props.initialValue || 'placeholder'
        };

        // ES6 类中函数必须手动绑定
        this.handleChange = this.handleChange.bind(this);
    }

    handleChange(event) {
        this.setState({
            text: event.target.value
        });
    }

    render() {
        return (
            <div>
                Type something:
                <input onChange={this.handleChange}
                value={this.state.text} />
            </div>
        );
    }
}
InputControlES6.propTypes = {
    initialValue: React.PropTypes.string
};
InputControlES6.defaultProps = {
    initialValue: ''
};
```

1.4.4.2 React 路由

路由库 React-Router。它是官方维护的，事实上也是唯一可选的路由库。它通过管理URL，实现组件的切换和状态的变化，开发复杂的应用几乎肯定会用到。

基本用法

React Router 安装命令如下。

```
$ npm install -S react-router
```

使用时，路由器 Router 就是 React 的一个组件。

```
import { Router } from 'react-router';
render(<Router/>, document.getElementById('app'));
```

Router 组件本身只是一个容器，真正的路由要通过 Route 组件定义。

```
import { Router, Route, hashHistory } from 'react-router';
render((
  <Router history={hashHistory}>
    <Route path="/" component={App}/>
  </Router>
), document.getElementById('app'));
```

上面代码中，用户访问根路由/，组件 APP 就会加载到 document.getElementById('app')。

你可能还注意到，Router 组件有一个参数 history，它的值 hashHistory 表示，路由的切换由 URL 的 hash 变化决定，即 URL 的#部分发生变化。举例来说，用户访问http://www.example.com/，实际会看到的是 http://www.example.com/#/。

Route 组件定义了 URL 路径与组件的对应关系。你可以同时使用多个 Route 组件。

```
<Router history={hashHistory}>
  <Route path="/" component={App}/>
  <Route path="/repos" component={Repos}/>
  <Route path="/about" component={About}/>
</Router>
```

上面代码中，用户访问/repos（比如 http://localhost:8080/#/repos）时，加载 Repos 组件；访问/about（http://localhost:8080/#/about）时，加载 About 组件。

嵌套路由

Route 组件还可以嵌套。

```
<Router history={hashHistory}>
  <Route path="/" component={App}>
    <Route path="/repos" component={Repos}/>
    <Route path="/about" component={About}/>
  </Route>
</Router>
```

上面代码中，用户访问/repos 时，会先加载 App 组件，然后在它的内部再加载 Repos 组件。

```
<App>
  <Repos/>
</App>
```

App 组件要写成下面的样子。

```
export default React.createClass({
  render() {
    return <div>
      {this.props.children}
    </div>
  }
})
```

上面代码中，App 组件的 this.props.children 属性就是子组件。

子路由也可以不写在 Router 组件里面，单独传入 Router 组件的 routes 属性。

```
let routes = <Route path="/" component={App}>
  <Route path="/repos" component={Repos}/>
  <Route path="/about" component={About}/>
</Route>;
<Router routes={routes} history={browserHistory}/>
```

Path 属性

Route 组件的 path 属性指定路由的匹配规则。这个属性是可以省略的，这样的话，不管路径是否匹配，总是会加载指定组件。

请看下面的例子。

```
<Route path="inbox" component={Inbox}>
  <Route path="messages/:id" component={Message} />
</Route>
```

上面代码中，当用户访问/inbox/messages/:id 时，会加载下面的组件。

```
<Inbox>
  <Message/>
</Inbox>
```

如果省略外层 Route 的 path 参数，写成下面的样子。

```
<Route component={Inbox}>
  <Route path="inbox/messages/:id" component={Message} />
</Route>
```

现在用户访问/inbox/messages/:id 时，组件加载还是原来的样子。

```
<Inbox>
  <Message/>
</Inbox>
```

上述讲到的三点路由知识，是最基本的，也会是最常用的几点，如果大家有兴趣的话，可以自己去 React 官方网站上自行学习；

这是官方的实例库（https://github.com/reactjs/react-router-tutorial/tree/master/lessons）

1.4.4.3 React 状态

React 把组件看成是一个状态机（State Machines）。通过与用户的交互，实现不同状态，然后渲染 UI，让用户界面和数据保持一致。

React 里，只需更新组件的 state，然后根据新的 state 重新渲染用户界面（不要操作 DOM）。

以下实例中创建了 LikeButton 组件，getInitialState 方法用于定义初始状态，也就是一个对象，这个对象可以通过 this.state 属性读取。当用户点击组件，导致状态变化，this.setState 方法就修改状态值，每次修改以后，自动调用 this.render 方法，再次渲染组件。

案例核心代码如下:

```
var LikeButton = React.createClass({
        getInitialState: function() {
          return {liked: false};
        },
        handleClick: function(event) {
          this.setState({liked: !this.state.liked});
        },
        render: function() {
          var text = this.state.liked ? '喜欢' : '不喜欢';
          return (
            <p onClick={this.handleClick}>
              你<b>{text}</b>我。点我切换状态。
            </p>
          );
        }
      });

      ReactDOM.render(
        <LikeButton />,
        document.getElementById('example')
```

);

state 工作原理

通过调用 setState(data, callback)方法，改变状态，就会触发 React 更新 UI。大部分情况下，我们不需要提供 callback 函数。React 会自动的帮我们更新 UI。

什么样的组件该有 state

大部分的组件应该从 props 属性中获取数据并渲染。但有的时候组件得到相应用户输入，同服务器交互，这些情况下会用到 state。

React 的官方说法是：尽可能的保持你的组件无状态化。为了实现这个目标，得保持你的状态同业务逻辑分离，并减少冗余信息，尽可能保持组件的单一职责。

React 官方推荐的一种模式就是：构建几个无状态的组件用来渲染数据，在这些之上构建一个有状态的组件同用户和服务交互，数据通过 props 传递给无状态的组件。我的理解大概就是这样

state 应该包含什么样的数据

UI 交互会导致改变的数据。

state 不应包含什么样的数据

计算过的数据

组件

从 props 复制的数据

state 应保含最原始的数据，比如说时间，格式化应该交给展现层去做。组件应在 render 方法里控制。

补充：

React 的作者认为，组件应该同关注分离，而不是同模板和展现逻辑分离。结构化标记和生成结构化标记的代码是紧密关联的，此外，展现逻辑一般都很复杂，使用模板语言会使展现变得笨重。

React 解决这个问题的方式就是：直接通过 JavaScript 代码生成 HTML 和组件树，这样的话，你就可以使用 JavaScript 丰富的表达力去构建 UI。为了使这个过程变得更简单，React 创建了类似 HTML 的语法去构建节点树，也就是 JSX 了。

JSX 语法是可选的，也就是说你也可以不使用，直接写 JavaScript 代码。看个对比例子：

```
#JSX 语法
React.render(
    <div className="c-list">content</div>,
    document.getElementById('example')
);

# JavaScript
```

```
React.render(
    React.createElement('div', {className: 'c-list'}, 'content'),
    document.getElementById('example')
);
```

这样简单的例子，我们都能感觉到 JSX 更加的语义化，更别说复杂的组件了。**所以强烈建议使用 JSX，关于 JSX 和 ES6 语法这里不做过多的讲解，详见：《前端-React 技术分析》**。

2 系统数据结构设计

2.1 数据库表设计

根据需求分析，酒店业务主要实体包括：酒店、房型、订单、库存、用户、图片。

实体间的关系如下:

- 酒店:房型=1:N
- 酒店:图片=1:N
- 房型:订单=1:1
- 房型:库存=1:1
- 用户:订单=1:N

各实体之间的关系如下图 4 所示:

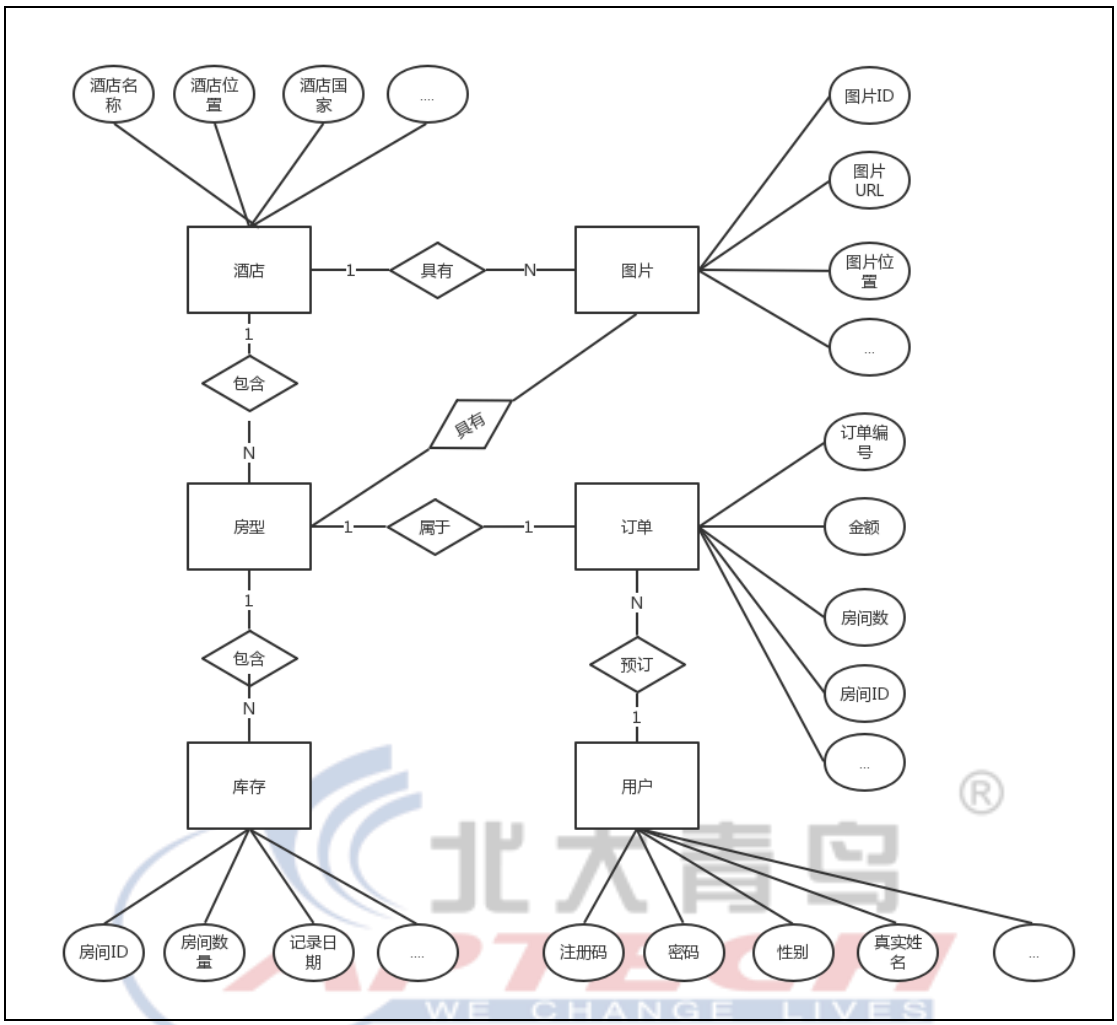

图 4:爱旅行项目 E-R 图

注：图中只列出系统中主要实体，其实系统中包含的实体还包括，城市、国家等。

根据系统的 ER 图，可以设计出爱旅行项目的数据库结构。爱旅行项目的数据库结构如图 4 所示。

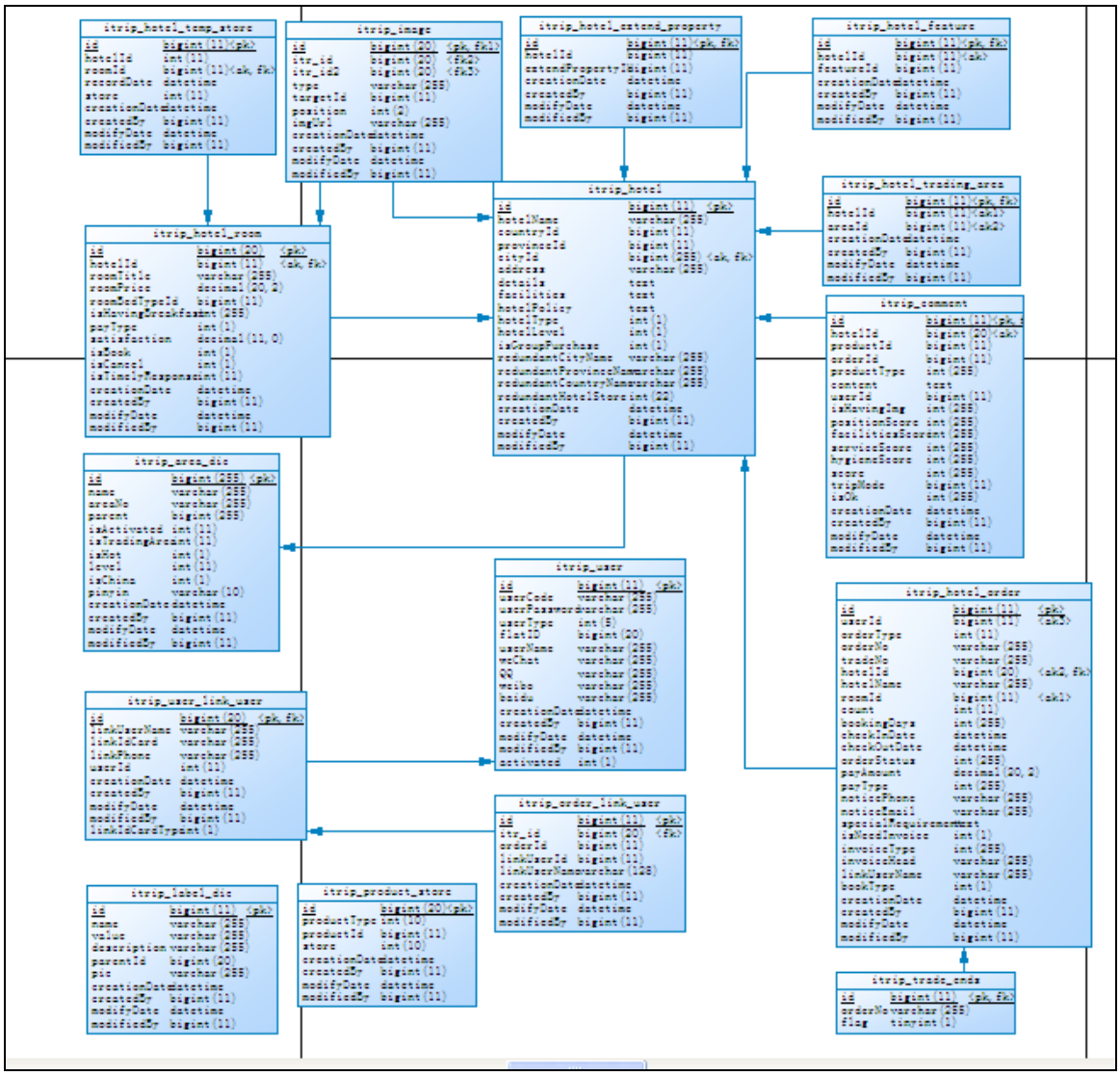

图 5:爱旅行项目数据库表关系图

根据以上实体，详细的爱旅行项目的数据库表如下:

酒店表用来保存酒店信息，为了更好的实现对酒店信息的查询，酒店表中冗余了部分字段。

表名	itrip_hotel（酒店表）	
字段名称	属性	说明
id	bigint	主键
hotelName	varchar	酒店名称
countryId	bigint	国家 ID
provinceId	bigint	省份 ID
cityId	bigint	市 ID
address	varchar	地址
details	text	酒店描述

facilities	text	酒店设施
hotelPolicy	text	酒店政策
hotelType	int	酒店类型
hotelLevel	int	酒店级别
isGroupPurchase	int	是否团购(0:不支持 1:支持)
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-5 酒店表

房间表用来保存房型信息，每一个房型只能属于一个酒店，所以在房型表中保存了酒店的关联字段。

表名	itrip_hotel_room(酒店房间表)	
字段名称	属性	说明
id	bigint	房间 ID
hotelId	bigint	酒店 ID
roomTitle	varchar	房间标题
roomPrice	decimal	房间价格
roomBedTypeId	bigint	房间床型
isHavingBreakfast	int	是否含早餐
payType	int	支付类型
satisfaction	decimal	满意度
isBook	int	是否可预订
isCancel	int	是否可取消
isTimelyResponse	int	是否及时确认
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表名	itrip_user(用户表)	
字段名称	属性	说明
id	bigint	主键
userCode	varchar	用户登录名称
userPassword	varchar	用户登录密码
userType	int	用户类型
flatID	bigint	平台 ID（根据不同登录用户，进行相应存入：自注册用户主键 ID、微信 ID、QQID、微博 ID）
userName	varchar	用户真实姓名
weChat	varchar	微信号
QQ	varchar	QQ 号
weibo	varchar	微博号
baidu	varchar	百度账号
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人
activated	int	是否激活(0:未激活 1:激活)

表-6 房间表

表名	itrip_area_dic(区域字典表)	
字段名称	属性	说明
id	bigint	主键
name	varchar	区域名称
areaNo	varchar	区域编号
parent	bigint	父级区域 ID
isActivated	int	是否激活
isTradingArea	int	是否是商圈
isHot	int	是否是热门城市
level	int	区域级别(0:国家级 1:省级 2:市级 3:县/区)

isChina	int	1:国内 2：国外
pinyin	varchar	区域名称拼音
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-7 区域字典表

表名	itrip_image(图片表)	
字段名称	属性	说明
id	bigint	主键
type	varchar	图片类型(0:酒店图片 1:房间图片 2:评论图片)
targetId	bigint	关联 id
position	int	图片上传顺序位置
imgUrl	varchar	图片地址
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-8 图片表

表名	itrip_label_dic(通用字典表)	
字段名称	属性	说明
id	bigint	主键
name	varchar	通用字典名称
description	varchar	描述
parentId	bigint	父节点 ID
pic	varchar	class
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-9 通用字典表

表名	itrip_hotel_order(订单表)	
字段名称	属性	说明
id	bigint	主键
userId	bigint	用户 Id
orderType	int	订单类型
orderNo	varchar	订单编号
tradeNo	varchar	交易编号
hotelId	bigint	酒店 ID
hotelName	varchar	酒店名称
roomId	bigint	房间 ID
count	int	房间数量
bookingDays	int	预订天数
checkInDate	datetime	入住日期
checkOutDate	datetime	退房日期
orderStatus	int	订单状态
payAmount	decimal	订单金额
payType	int	订单支付类型
noticePhone	varchar	通知手机号
noticeEmail	varchar	通知邮箱
specialRequirement	text	特殊需求
isNeedInvoice	int	是否需要发票
invoiceType	int	发票类型
invoiceHead	varchar	发票抬头
linkUserName	varchar	联系人姓名
bookType	int	预订类型
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-10 订单表

表名	itrip_comment(评论表)	
字段名称	属性	说明
id	bigint	主键
hotelId	bigint	酒店 ID
productId	bigint	房间 ID
orderId	bigint	订单 ID
productType	int	订单类型
content	text	评论内容
userId	bigint	评论用户 ID
isHavingImg	int	是否含有图片
positionScore	int	位置评分
facilitiesScore	int	特色评分
serviceScore	int	服务评分
hygieneScore	int	卫生评分
score	int	总平均分
tripMode	bigint	出游类型
isOk	int	是否满意
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-11 评论表

表名	itrip_hotel_extend_property(酒店拓展属性表)	
字段名称	属性	说明
id	bigint	主键 ID
hotelId	bigint	酒店 ID
extendPropertyId	bigint	拓展属性 ID
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间

modifiedBy	bigint	修改人

表-12 酒店拓展属性表

表名	itrip_hotel_feature(酒店特色表)	
字段名称	属性	说明
id	bigint	主键
hotelId	bigint	酒店 ID
featureId	bigint	酒店特色 ID
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-13 酒店特色表

表名	itrip_hotel_temp_store(房间实时库存表)	
字段名称	属性	说明
id	bigint	主键
hotelId	int	酒店 ID
roomId	bigint	房间 ID
recordDate	datetime	记录日期
store	int	库存
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-14 房间实时库存表

表名	itrip_hotel_trading_area (酒店商圈表)	
字段名称	属性	说明
id	bigint	主键
hotelId	bigint	房间 ID
areaId	bigint	区域 ID
creationDate	datetime	创建时间
createdBy	bigint	创建人

modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-15 酒店商圈表

表名	itrip_order_link_user(订单联系人表)	
字段名称	属性	说明
id	bigint	主键
orderId	bigint	订单 ID
linkUserId	bigint	联系人 ID
linkUserName	varchar	联系人姓名
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-16 订单联系人表

表名	itrip_product_store(商品原始库存表)	
字段名称	属性	说明
id	bigint	主键
productType	int	商品类型
productId	bigint	商品 ID
store	int	商品库存
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人

表-17 商品原始库存表

表名	itrip_trade_ends(支付宝交易成功中间表)	
字段名称	属性	说明
id	bigint	主键
orderNo	varchar	订单编号
flag	tinyint	操作状态

表名	itrip_user_link_user(用户联系人表)	
字段名称	属性	说明
id	bigint	主键
linkUserName	varchar	联系人姓名
linkIdCard	varchar	联系人身份证号
linkPhone	varchar	联系电话
userId	int	用户 ID
creationDate	datetime	创建时间
createdBy	bigint	创建人
modifyDate	datetime	修改时间
modifiedBy	bigint	修改人
linkIdCardType	int	联系人证件类型

表-18 用户联系人表

• 2.2 核心表设计

2.2.1 库存及订单表设计

背景：本次项目开发，只考虑酒店的相关业务。因此，本次库存设计，只考虑酒店房间的库存。

库存表的具体设计，请参考《订单业务设计分析说明文档.docx》

2.2.2 用户表设计

用户模块需求分析：本次系统开发，用户模块包含的功能有：

- 注册
  a) 邮箱注册
  b) 手机注册
- 登录
  a) 第三方登录（微信登录、微博登录、QQ 登录）
  b) 自注册用户登录

- Token 有效期维护及置换

根据功能需求，用户表的具体设计，请参考《Token 技术分析.docx》。

2.2.4 图片表设计

图片相关需求分析：爱旅行项目中的图片较多，这其中包括：

- 酒店图片：每个酒店包含 1 张以上图片。
- 酒店房间图片：每个房间最多包含 4 张图片。
- 评论图片：每条评论，最多可以上传 4 张图片。

由于图片本身和多个实体之间都是多对多的关系，因此，本次系统开发中，专门建立了一个图片表进行图片信息维护。图片表的关键字段设计如下：

- type 字段：保存图片类型(0:酒店图片 1:房间图片 2:评论图片)。
- targetId 字段：保存相关联字段 ID。
- position 字段：记录用户图片上传的顺序。
- imgUrl 字段：记录用户上传的原始图片的访问地址。

2.2.5 酒店系列表设计

酒店系列数据需求分析：

- 酒店和房型是 1:N 的关系，一个酒店可以有多个房型。
- 房型和房间是 1:N 的关系，一个房型可以有多个房间。
- 酒店和商圈是 N:N 的关系，一个酒店可以在多个商圈，1 个商圈可以包含多个酒店。
- 酒店和特色是 N:N 的关系，一个酒店可以具有多种特色，一种特色可以属于多个酒店。
- 酒店和图片是 1:N 的关系，一个酒店可以有多张图片。

酒店系列数据库表设计：

- 酒店表:保存酒店信息，酒店描述，介绍等保存成大文本格式。
- 房型表:保存酒店房型信息，包括房间名称，价格等。
- 原始库存表：关联房型表，指定该房型在没有客人入住的情况下的原始可用房间数。
- 酒店商圈关联表：关联酒店表和商圈表，指定酒店在那些商圈。
- 商圈表：保存到数据字典表。
- 酒店特色关联表：关联酒店表和特色表，指定酒店的特色。

- 酒店特色表保存到数据字典表
- 图片表：用于保存酒店的图片。
- 房型实时库存表：由于某一房型的房间的数量是随着用户入住和退房的变化而变化的，所有设计了房间实时库存表用来保存房间的实时库存，并随着用户的操作而变化。

2.2.6 数据字典表设计

系统的运行依赖一些基础数据，这些数据包括

- 地理位置数据：国家信息、省\市\县信息、商圈信息。
- 基础信息数据：床型数据、分类数据、特色数据、酒店配置数据等。

基于以上数据的分类，将数据字典表设计成为两张表。

- 地理位置数据表：保存国家信息、省\市\县信息、商圈信息地理位置数据。
- 基础信息数据表：保存床型数据、分类数据、特色数据、酒店配置数据。

注：数据字典表的设计请参照本文档中的表 7、表 9。

3 接口设计

3.1 接口请求数据格式

3.2 用户接口

爱旅行项目对外提供的接口中，全部采用 Http 请求的调用方式，接口返回的数据内容（json 格式）如下：

选项	说明
errorCode	返回异常码
msg	返回提示信息
success	成功失败标识
data	返回的数据

表-19 通用接口返回数据

注：在以下的接口数据返回中，全部采用以上的格式进行数据返回，不在赘述（详见《项目开发规范.docx》）。

以下列出爱旅行项目中较为关键的接口设计。

3.2.1 搜索酒店接口

需求分析：

酒店搜索的参数如图 6 所示，这其中包括目的地、入住退房日期、关键词、位置、价格、星级、特色等。由于酒店的搜索参数众多，类型复杂，所以本次项目开发，采用搜索引擎来完成酒店的搜索功能。酒店搜索数据流向图如图 7 所示。

图 6:酒店搜索参数

搜索引擎架构中，我们包括三个系统，分别是 biz 主业务系统、solr 搜索引擎系统、search 搜索引擎管理系统。

主业务系统：biz 系统负责调用 search 系统获取酒店数据，并可以向 search 传参。

solr 搜索引擎系统:负责从数据库获取数据，并执行增量更新。

search 搜索引擎管理系统:负责连接 biz 系统和 solr 系统，调用 solr 系统获取数据。

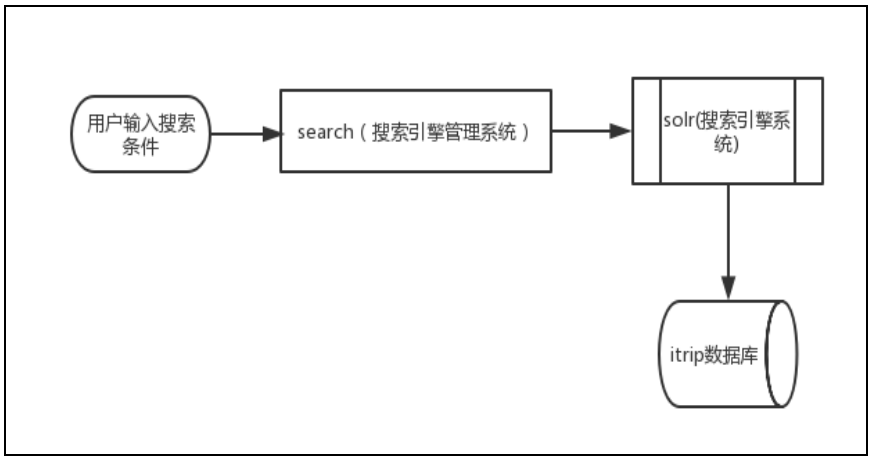

图 7 酒店搜索数据流向图

酒店搜索接口传入参数设计:

```
{
   "ascSort": "string", //升序排列字段
   "checkInDate": "2017-06-19",//入住时间
```

```
  "checkOutDate": "2017-06-19",//退房时间
  "descSort": "string",//倒叙排列字段
  "destination": "string",//目的地
  "featureIds": "string",//酒店特色 IDS
  "hotelLevel": 0, //酒店级别
  "keywords": "string",//关键词
  "maxPrice": 0, //最高价
  "minPrice": 0, //最低价
  "pageNo": 0, //页数
  "pageSize": 0, //页大小
  "tradeAreaIds": "string"//商圈 ids
}
```

酒店搜索接口返回参数设计:

- 其中 data、errorCode、msg、successCode 请参考统一返回数据。
- data 内部为实际返回的数据。

```
{
  "data": {
    "beginPos": 0,
    "curPage": 1,
    "pageCount": 3,
    "pageSize": 2,
    "rows": [
      {
        "address": "北京  东城区  前门东大街 3 号 ， 近正义路。",
        "avgScore": 4.3636,
        "commentCount": 11,
        "extendPropertyNames": "有无线上网,免费停车,提供餐饮",
        "extendPropertyPics": "fa fa-wifi,fa fa-product-hunt,icon-food",
        "featureNames": "东南亚风情,休闲度假,亲子时刻,会议酒店",
```

```
    "hotelLevel": 5,
    "hotelName": "北京首都大酒店",
    "id": 1,
    "imgUrl": "http://192.168.9.44/hotel/8-4567454545412-4567891.jpg",
    "isOkCount": 0,
    "maxPrice": 1677,
    "minPrice": 605,
    "redundantCityName": "北京市",
    "redundantCountryName": "中国",
    "redundantProvinceName": "北京市",
    "tradingAreaNames": "前门"
  },
  {
    "address": "北京 朝阳区 紫玉东路 1 号 ， 近北辰购物中心。",
    "avgScore": 3,
    "commentCount": 2,
    "extendPropertyNames": "免费停车",
    "extendPropertyPics": "fa fa-product-hunt",
    "featureNames": "商务出行,青年旅社,会议酒店,亲子时刻",
    "hotelLevel": 3,
    "hotelName": "北京紫玉度假酒店",
    "id": 3,
    "imgUrl": "http://192.168.9.44/hotel/8-4567454545412-4567893.jpg",
    "isOkCount": 0,
    "maxPrice": 379,
    "minPrice": 275,
    "redundantCityName": "北京市",
    "redundantCountryName": "中国",
    "redundantProvinceName": "北京市",
```

```
            "tradingAreaNames": "亚运村、奥体中心地区"
        }
    ],
    "total": 6
},
"errorCode": "0",
"msg": null,
"success": "true"
}
```


3.2.2 库存验证接口

当用户进行房间筛选或者下单操作的时候，biz 系统会根据用户填写的入住、退房时间，以及其它信息对房间信息进行筛选，或者验证。由于酒店房间的库存不同于普通商品的库存，酒店房间的库存会随着时间的变化而变化。因此在数据库设计时，设计了房间实时库存表用来保存房间的库存信息。下单操作整体的数据流向图如图 8 所示。

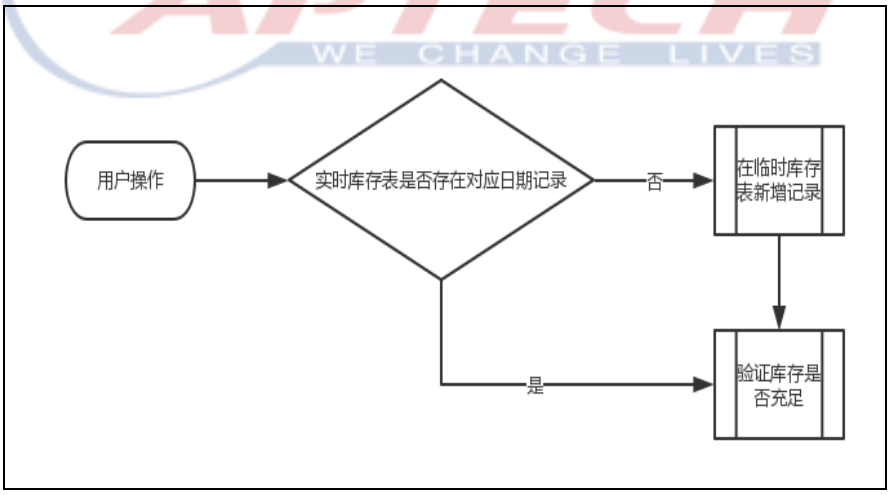

图 8 酒店搜索数据流向图

3.2.3 下单接口

下单流程分解如下，

1. 用户下单
2. 判断商品库存是否充足

3. 商品充足则进行下单操作
4. 在规定时间内（2 小时）等待用户支付，并在这个时间段内为用户锁定库存。
5. 用户支付成功，则实际的扣减库存
6. 用户超时未支付则取消订单，释放库存。
7. 用户支付成功后，系统自动检测当前日期是否大于用户的入住日期，如果大于则将订单状态修改为待评价状态。

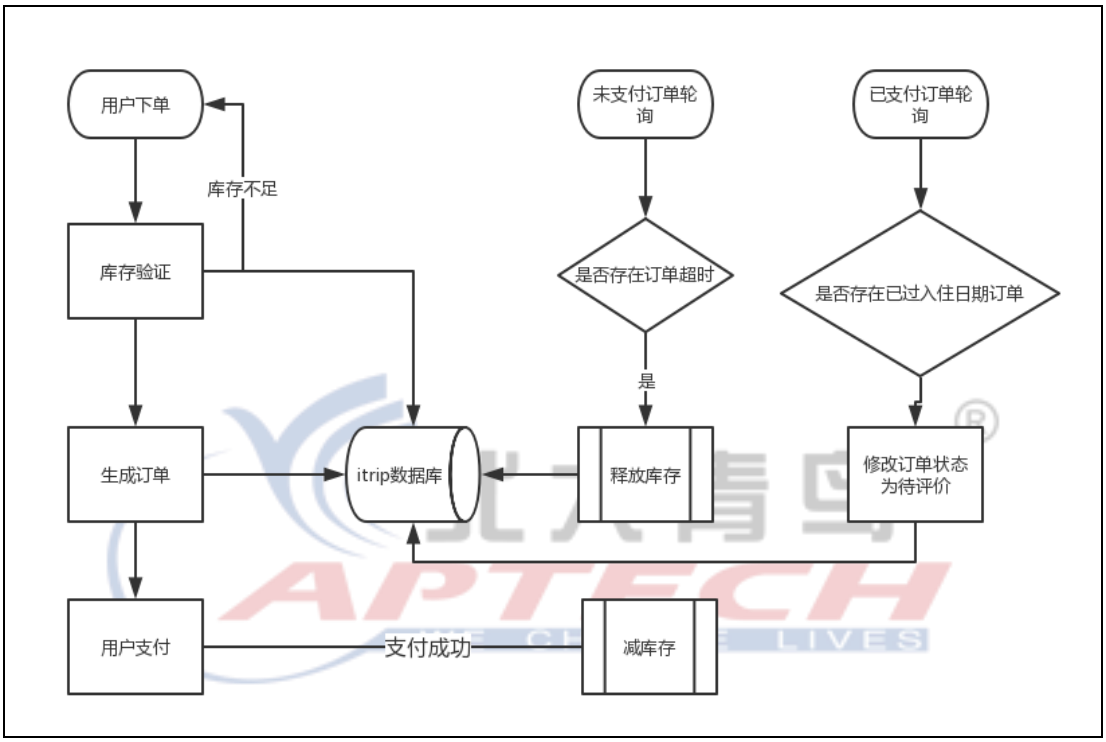

图 9 下单数据流向图

下单接口请求参数如下：

```
{
    "checkInDate": "2017-01-01 12:00:00", //入住日期
    "checkOutDate": "2017-01-03 12:00:00",//退房日期
    "count": 20,//房间数量
    "hotelId": 1,//酒店 id
    "roomId": 1//房间 id
}
```

下单接口返回数据如下：

- 其他参数同统一返回参数。
- data 数据返回具体的订单号

```
{
    "data": {
        "orderNo": "D10000012017060110542 46a81e1",
        "id": "83"
    },
    "errorCode": "0",
    "msg": "生成订单成功",
    "success": "true"
}
```


3.2.4 支付接口

支付宝的支付流程涉及到多个应用，包括 itrip 交易模块、支付宝应用、前端程序、itrip 主业务系统。

支付流程如下

- 用户通过 itrip 网站进行支付宝支付提交。
- 交易模块处理交易数据，将交易数据提交给支付宝平台。
- 支付宝平台进行相关支付操作。
- 支付宝支付成功，调用回调 URL，回调至交易模块。
- 交易模块进行订单状态修改，并将成功记录写入中间交易记录表，并通知主业务系统。
- 主业务系统查询、轮询交易记录表，执行减库存操作。

支付宝支付流程如图 10 所示。

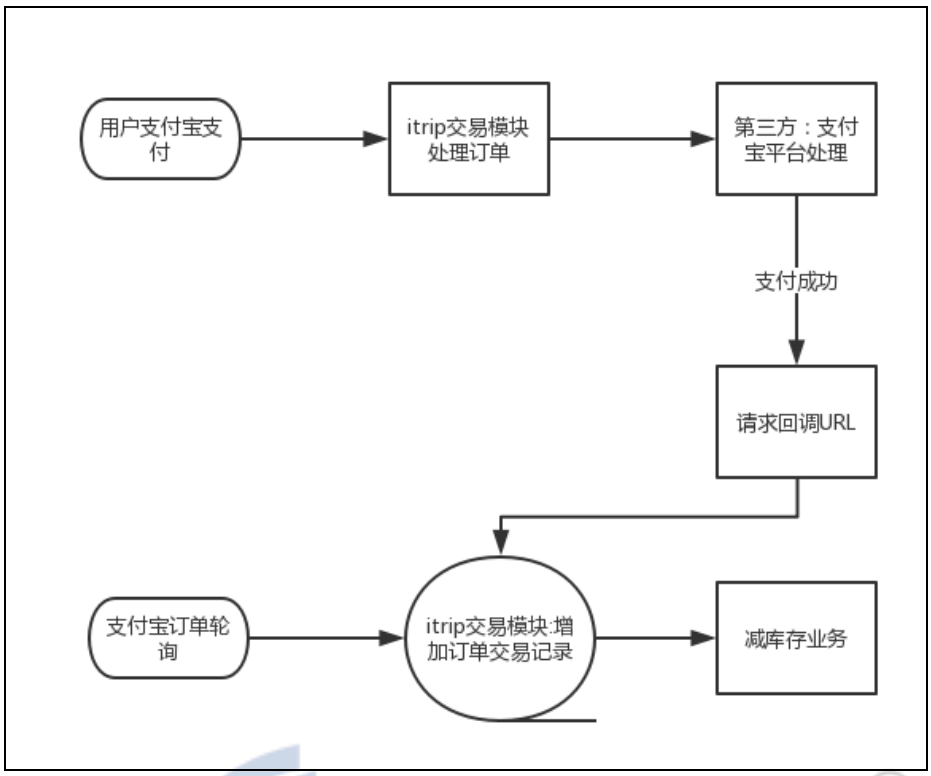

图 10：支付宝支付流程

前端程序通过支付接口使用表单 POST 提交，表单参数分别如下：

WIDout_trade_no	//订单编号
WIDsubject	//订单名称
WIDtotal_amount	//订单金额

支付完成后更新订单交易号、状态并跳转至前端程序。同时，当支付成功后增加后续订单待处理记录，并通知业务系统“订单支付成功后续处理接口”执行减库存操作，以 GET 方式传递参数如下：

orderNo	//订单编号

3.2.5 图片上传接口

由于爱旅行项目的图片众多，因此爱旅行项目在项目管理上，将爱旅行项目中的图片单独进行管理。

图片的上传流程拆解开来，如下：

- 用户上传图片
- 主业务系统执行图片上传

- 图片上传成功，主业务系统返回图片 URL
- 用户将 URL 绑定到其它业务并执行相关操作

图片上传的流程如图 11 所示。

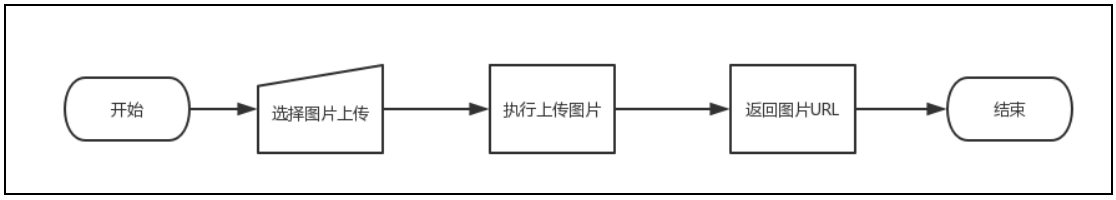

图 11

注意：

1> 所有上传的原始图片文件均存放于服务器指定路径下：

- 评论图片：/data1/uploadimg/comment
- 酒店图片：/data1/uploadimg/hotel
- 酒店房间图片：/data1/uploadimg/hotelroom

2> 为了确保文件名的唯一性，必须对上传文件进行重命名。命名规则：用户 id+当前系统时间+随机数

图片上传接口的返回数据如下：

- 其他参数同统一返回参数。
- data 数据返回上传图片的 URL。

```
{
    "data": [
        "http://192.168.9.44/comment/8-1497864513314-2026748.jpg",
        "http://192.168.9.44/comment/8-1497864513347-2002457.jpg"
    ],
    "errorCode": "0",
    "msg": "文件上传成功",
    "success": "true"
}
```

3.2.6 图片下载接口

图片下载主要涉及到评论图片、酒店图片、酒店房型图片的展现。下载流程较为简单，

此处不再赘述。

注意：由于爱旅行平台涉及图片展现较多，故在基于系统性能、页面响应速度以及用户体验的考虑，该系统中展现的所有图片均不使用原图，采用 Tengine + Lua + GraphicsMagick 实现图片的自动裁剪/缩放，即：按需裁剪，上传初始化时只有原始图，并且可以有效的省去切图的工作量。比如：访问原始图(8-3566787986543-7654678.jpg)的 URL 为：http://192.168.9.44/comment/8-3566787986543-7654678.jpg （如图 12 所示）若根据需求进行缩放（300x300），前端只需访问：http://192.168.9.44/comment/8-3566787986543-7654678.jpg_300x300.jpg 即可（如图 13 所示）。关于如何实现图片自动裁剪/缩放，详见《使用 Tengine + Lua + GraphicsMagick 实现图片自动裁剪/缩放.docx》。

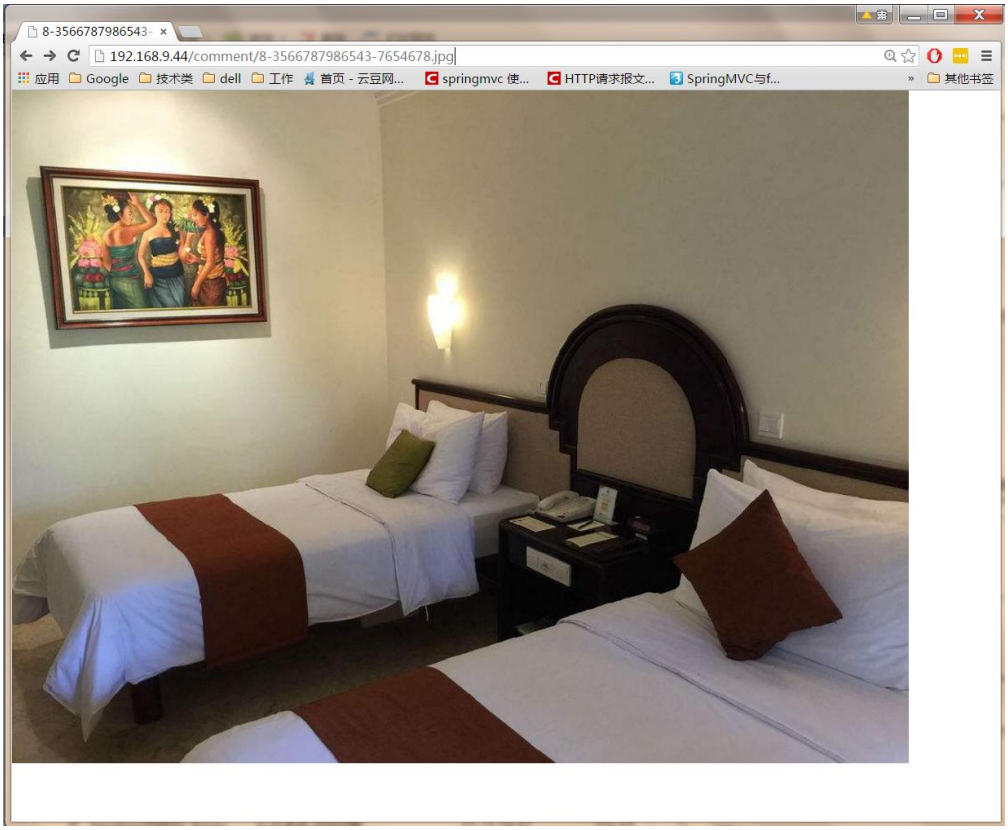

图 12

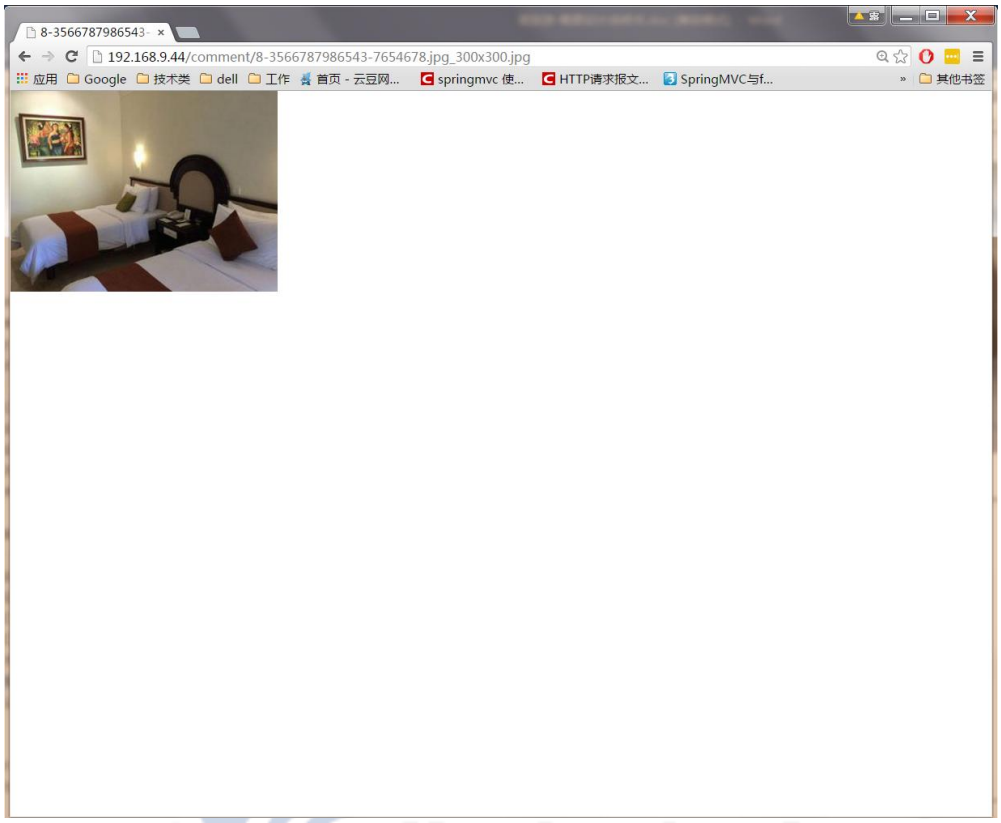

图 13

获取指定图片接口包括：

- 根据 targetId（酒店 id）查询酒店图片(type=0)
- 根据 targetId（酒店房型 id）查询酒店房型图片(type=1)
- 根据 targetId（评论 id）查询评论图片(type=2)

接口统一返回数据如下：

- 其他参数同统一返回参数。
- data 数据返回指定图片的 URL。


```
{
  "data": [
    {
      "imgUrl": "http://192.168.9.44/comment/8-6754744453467-4566559.jpg",
      "position": 1
    },
    {
      "imgUrl": "http://192.168.9.44/comment/8-6754744453467-4566558.jpg",
```

```
        "position": 2
      },
      {
        "imgUrl": "http://192.168.9.44/comment/8-6754744453467-4566510.jpg",
        "position": 3
      }
  ],
  "errorCode": "0",
  "msg": "获取评论图片成功",
  "success": "true"
}
```


3.2.7 用户认证接口

用户认证系统（itripauth）将提供登录、注册、激活、注销、token 置换等接口，在该子系统中主要提供用户授权访问及会话保持服务，并满足多种平台下的客户端访问、多服务器端系统间的跨域会话共享以降低认证系统与其它系统间的耦合、更方便地加入第三方认证的支持。具体接口如表-18 所示。

接口	功能	API 调用	请求参数	返回数据

用户登录	根据用户名、密码进行统一认证	/auth/api/dologin	(POST FORM)
			name=test
			password=test
				{
				"data": {
				"expTime":long,
				"genTime":long,
				"token":"string"
				},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				登录成功“success”为 true 否则 false

用户注销	删除当前用户会话信息，客户端需在 header 中发送 token	/auth/api/logout	(GET HEADER) token=token:PC-21ec64d6e9cae0917ea4b54bc36809d8-8-20170601100121-699dfc	{ "data": {}, "errorCode": "string", "msg": "string", "success": "string"

				} 注销成功“success”返回 true 否则 false

用户名验证	验证是否已存在该用户名称	/auth/api/ckuser	（GET QUERY）
			name=test@bdqn.cn
				{
				"data": {},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				用户名可用“success”返回 true 否则 false

使用邮箱注册	输入合法邮箱后，首先保存用户信息，然后生成激活码并发送至邮箱，完成邮箱有效性验证	/auth/api/register	（POST JSON）
			{
			"userCode": "string",
			"userName": "string",
			"userPassword": "string"
			}
				{
				"data": {},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				注册成功“success”返回 true 否则 false，此时账户处于未激活状态

邮箱注册用户激活	用户使用接收到的激活码在指定时间内激活注册信息	/auth/api/activate	（PUT QUERY）
			user=test@bdqn.cn
			code=018f9a8b2381839ee6f40ab2207c0cfe
				{
				"data": {},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				激活成功“success”返回 true 否则 false

使用手机注册	输入合法手机号后，首先保存用户信息，然后调用第三方 SMS 平台发送校验码，完成手机号有效性验证	/auth/api/registerbyphone	（POST JSON）
			{
			"userCode": "string",
			"userName": "string",
			"userPassword": "string"
			}
				{
				"data": {},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				注册成功“success”返回 true 否则 false，此时账户处于未激活状态

手机注册用户短信验证	用户使用接收到的短信验证码在指定时间内输入，由服务器完成校验后激活用户注册信息	/auth/api/validatephone	（PUT QUERY）
			user=13811223344
			code=8790
				{
				"data": {},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				验证成功“success”返

				回 true 否则 false

Token 置换	提供客户端置换 token 操作以刷新会话有效期，服务器需要获取客户端 header 中的 token 串	/auth/api/retoken	（POST HEADER）
			token=token:PC-21ec64d6e9cae0917ea4b54bc36809d8-8-20170601100121-699dfc
				{
				"data": {
				"expTime":14990,
				"genTime":14880,
				"token":"新 token"
				},
				"errorCode": "string",
				"msg": "string",
				"success": "string"
				}
				置换成功“success”返回 true 否则 false

表-20 用户认证子系统接口列表

用户登录、注销、Token 置换流程详见：

docs\关键技术分析类文档\token 流程.png

docs\关键技术分析类文档\Token 技术分析.docx

邮箱与手机注册流程详见：

docs\关键技术分析类文档\用户注册开发指南.docx

3.2.8 订单系列其他接口

除了上述的下单接口，订单接口还提供一下功能接口。

接口	功能	API 调用	传入参数	返回参数

根据订单 ID 查看个人订单详情	根据传入的订单 ID，返回订单相关的个人信息	/api/hotelorder/getpersonalorderinfo/{orderId}	5	{
				"data": {
				"bookType": 0,
				"creationDate": "2017-05-19",
				"id": 5,
				"noticePhone": "13899999999",
				"orderNo": "D10000012017051910140695595 9",
				"orderProcess": {
				"1": "订单提交",
				"2": "订单支付",
				"3": "订单取消"

				}, "orderStatus": 1, "payAmount": 7693, "payType": null, "processNode": "3", "roomPayType": 3 }, "errorCode": "0", "msg": "获取个人订单信息成功", "success": "true" }

根据条件查询个人订单列表，并分页显示	根据搜索条件，查询用户的个人订单列表	/api/hotelorder/getpersonalorderlist	{
			"orderStatus": 3,
			"orderType": 1,
			"pageNo": 1,
			"pageSize": 20,
			}
				{
				"data": {
				"beginPos": 0,
				"curPage": 1,
				"pageCount": 1,
				"pageSize": 20,
				"rows": [
				{
				"checkInDate": "2017-05-19",
				"creationDate": "2017-05-19",
				"hotelId": 1,
				"hotelName": " 北京首都大酒店",
				"id": 4,
				"linkUserName": " 周周",
				"orderNo": "D10000012017051910040 18d5ef8",
				"orderStatus": 3,
				"orderType": 1,
				"payAmount": 605
				}
				],
				"total": 1
				},
				"errorCode": "0",
				"msg": "获取个人订单列表成功",
				"success": "true"
				}

根据订单 ID 获取订单信	根据 id 返回订单的详细	api/hotelorder/queryOrderById/{orderId}		{ "data": {

息	信息			"bookType": 0, "bookingDays": 3, "checkInDate": "2017-05-23", "checkOutDate": "2017-05-25", "count": 2, "hotelId": 1, "hotelName": "北京首都大酒店", "id": 11, "invoiceHead": "北京前门信息公司", "invoiceType": 1, "isNeedInvoice": 1, "itripOrderLinkUserList": [], "linkUserName": "王平", "noticeEmail": "zhouzhou@163.com", "noticePhone": "13899999999", "orderNo": "D10000012017051910362 54c7031", "orderType": 1, "payType": null, "roomId": 2, "specialRequirement": "" }, "errorCode": "0", "msg": "获取订单成功", "success": "true" }

表-21 订单系列接口列表

3.2.9 房型系列接口

爱旅行主业务系统中，房型模块，提供的接口功能如下，包括查询房间列表、查询房间信息、以及查询房间图片的接口。

注意：对房型进行筛选的过程中，请注意房型库存的判断。

接口	API 调用	传入参数	返回参数
根据酒店房型ID查	/api/hotelroom/ge	房型 ID	{

询酒店房型图片	timg/{targetId}		"data": {
			"bookType": 0,
			"bookingDays": 3,
			"checkInDate": "2017-05-23",
			"checkOutDate": "2017-05-25",
			"count": 2,
			"hotelId": 1,
			"hotelName": "北京首都大酒店",
			"id": 11,
			"invoiceHead": "北京前门信息公司",
			"invoiceType": 1,
			"isNeedInvoice": 1,
			"itripOrderLinkUserList": [],
			"linkUserName": "王平",
			"noticeEmail": "zhouzhou@163.com",
			"noticePhone": "13899999999",
			"orderNo":
			"D10000012017051910362 54c7031",
			"orderType": 1,
			"payType": null,
			"roomId": 2,
			"specialRequirement": ""
			},
			"errorCode": "0",
			"msg": "获取订单成功",
			"success": "true"
			}

查询酒店床型列表	/api/hotelroom/queryhotelroombed	无	{ "data": [{ "description": "", "id": 2, "name": "大床", "pic": "" }, { "description": "", "id": 3, "name": "双床", "pic": "" }, { "description": "", "id": 4,

			"name": "多床",
			"pic": ""
			},
			{
			"description": "",
			"id": 5,
			"name": "单人床",
			"pic": ""
			}
			],
			"errorCode": "0",
			"msg": "获取成功",
			"success": "true"
			}

查询酒店房间列表	/api/hotelroom/queryhotelroombyhotel	{
		"endDate": "2017-06-22",
		"hotelId": 1,
		"isBook": 1,
		"isCancel": 1,
		"isHavingBreakfast": 1,
		"isTimelyResponse": 1,
		"payType": 1,
		"startDate": "2017-06-22"
		}
			{
			"data": [
			[
			{
			"hotelId": 1,
			"id": 3,
			"isBook": 1,
			"isCancel": 1,
			"isHavingBreakfast": 1,
			"isTimelyResponse": 1,
			"payType": 3,
			"roomBedTypeId": 4,
			"roomPrice": 1099,
			"roomTitle": "特惠家庭套房",
			"satisfaction": 4
			}
			],
			[
			{
			"hotelId": 1,
			"id": 4,
			"isBook": 1,
			"isCancel": 1,
			"isHavingBreakfast": 1,
			"isTimelyResponse": 1,
			"payType": 3,
			"roomBedTypeId": 4,
			"roomPrice": 1381,
			"roomTitle": "标准套房",
			"satisfaction": 4

			}
			],
			[
			{
			"hotelId": 1,
			"id": 5,
			"isBook": 1,
			"isCancel": 1,
			"isHavingBreakfast": 1,
			"isTimelyResponse": 1,
			"payType": 3,
			"roomBedTypeId": 4,
			"roomPrice": 1677,
			"roomTitle": "行政套房",
			"satisfaction": 4
			}
			]
			],
			"errorCode": "0",
			"msg": "获取成功",
			"success": "true"
			}

表-22 房型系列接口列表

3.2.10 评论系列接口

除了上述为评论添加图片的功能，评论模块还需要提供新增评论、查询评论列表、查询出游类型、查询酒店评分等接口。具体接口功能如

接口	API 调用	传入参数	返回数据

根据评论类型查询评论列表，并分页显示	/api/comment/getcommentlist	{
		"hotelId": 1,
		"isHavingImg": 1,
		"isOk": 1,
		"pageNo": 1,
		"pageSize": 2
		}
			{
			"data": {
			"beginPos": 0,
			"curPage": 1,
			"pageCount": 3,
			"pageSize": 2,
			"rows": [
			{
			"checkInDate": "2017-05-20",
			"content": "酒店环境很好 服务特别棒 尤其是前台美女张夏女士服务更是没得说 推荐入住 一进房间就给人一种亲切的感觉 有回到家的

			温馨 酒店早餐也是棒棒的 什么小吃都有 非常划算 以后也会毫不犹豫的选择桔子酒店 总之非常好的",
			"creationDate": "2017-05-22",
			"hotelLevel": null,
			"id": 5,
			"isHavingImg": 1,
			"roomTitle": "标准套房",
			"score": 5,
			"tripModeName": "大床",
			"userCode": "yao.liu2015@bdqn.cn"
			},
			{
			"checkInDate": "2017-05-20",
			"content": "老牌酒店 地段优越 服务到位 管理良好 设施较旧但齐全 房间较小 网络优质 价格偏高 近期第二次入住 总体满意，推荐入住。",
			"creationDate": "2017-05-22",
			"hotelLevel": null,
			"id": 6,
			"isHavingImg": 1,
			"roomTitle": "行政套房",
			"score": 5,
			"tripModeName": "大床",
			"userCode": "yao.liu2015@bdqn.cn"
			}
			],
			"total": 6
			},
			"errorCode": "0",
			"msg": null,
			"success": "true"
			}

根据酒店 id 查询评论数量（全部评论、值得推荐、有待改善、有图片）	/api/comment/getcount/{hotelId}	酒店 id	{
			"data": {
			"improve": 3,
			"isok": 8,
			"havingimg": 7,
			"allcomment": 11
			},

			"errorCode": "0", "msg": "获取酒店各类评论数成功", "success": "true" }

获取酒店的总体评分、位置评分、设施评分、服务评分、卫生评分	/api/comment/gethotelscore/{hotelId}	酒店 ID	{
			"data": {
			"avgFacilitiesScore": 4,
			"avgHygieneScore": 4.1,
			"avgPositionScore": 4.1,
			"avgScore": 4.4,
			"avgServiceScore": 2.9
			},
			"errorCode": "0",
			"msg": "获取评分成功",
			"success": "true"
			}

查询出游类型列表	/api/comment/gettraveltype	无	{ "data": [{ "description": "", "id": 108, "name": "商务出差", "pic": "" }, { "description": "", "id": 109, "name": "朋友出游", "pic": "" }, { "description": "", "id": 110, "name": "情侣出游", "pic": "" }, { "description": "", "id": 111, "name": "家庭亲子", "pic": "" }, { "description": "",

			"id": 112,
			"name": "独自旅行",
			"pic": ""
			},
			{
			"description": "",
			"id": 113,
			"name": "代人预定",
			"pic": ""
			},
			{
			"description": "",
			"id": 114,
			"name": "其他",
			"pic": ""
			}
			],
			"errorCode": "0",
			"msg": "获取旅游类型列表成功",
			"success": "true"
			}

表-23 评论系列接口列表

3.3 内部接口

系统中的内部接口如下：

- 搜索引擎数据查询接口：搜索引擎 solr 需对搜索引擎管理系统 search 提供数据查询接口。
- 订单支付成功接口：前端需要为交易模块提供支付成功展示接口，当支付宝通知支付成功后交易模块需要重定向到该接口。
- 数据库接口：数据库需要对主业务系统、用户认证系统、交易系统、搜索引擎系统提供 JDBC 接口。

3.4 前端系统接口相关内容

前后端数据交互，使用 fetch 函数请求后端数据接口，对 fetch 函数进行封装，暴露两个接口 getRequest(url,param)和 postRequest(url,param)两个请求接口，请求接口的参数分别是：url 请求的地址；param。请求访问后台的参数信息；其他页面只需要加载这个封装好的函

数就可以调用后端的数据信息；请求报错使用自身的 catch 方法捕获异常，并提示异常信息也可根据具体的场景调用不同的提示信息，如果需要验证登录系统则重新跳转到登录页面进行登录方可进行后续操作。

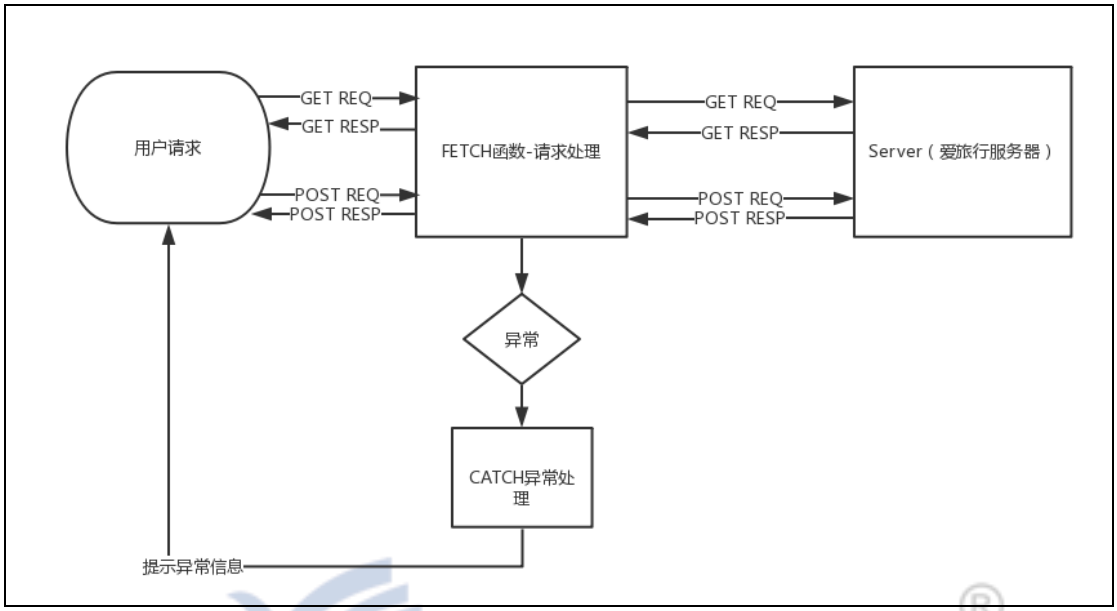

图 13 前端数据处理流程

4 代码规范

具体代码开发规范请参考《项目开发规范.docx》。

5 系统异常处理设计

5.1 异常的抛出与捕获

为了增强系统的稳定性和安全性系统中秉承约定大于俗成的规则，规定：

- 统一在 service 层抛出异常，service 不做异常的捕获。
- 统一在 controller 层处理异常，如果系统异常需要给客户端返回相应的提示。
- 所有异常返回结果都必须包含异常码，异常内容。

5.2 事务控制

系统中的事务控制，统一采用 spring 事务管理机制，对于一个方法中的多次数据库操作，

需要为该方法增加事务控制，确保数据的正确性。

需要事务控制的方法如下(包括但不仅限于以下内容):

- 新增订单
- 新增评论
- 用户注册
- 新增联系人
- 删除联系人

注意：

- 用户在 DAO 层和 Service 层不需要处理异常，只需要把异常抛出。在 Controller 层直接接收异常，并进行异常的相关处理。
- 所有的异常不能抛给客户端，需要在数据返回给客户端之前全部解决。
- 事务统一在 Service 进行控制，需要进行事务控制的方法，请在方法前加 itriptx 前缀。
- 如在代码开发中发现其它需要事务控制的方法，请与项目经理商榷后，补存该文档。

5.3 异常码定义

本系统中异常业务，异常码定义如下:

- 酒店业务：1 开头（10000）
- 评论业务：10001 ——10100
- 酒店详情业务：10101 ——10200
- 订单业务：10201 ——10400
- 搜索业务：2 开头（20000）
- 用户认证业务：3 开头（30000）
- 用户支付业务：4 开头（40000）

6 补充需求设计

根据移动端提出的补充需求，特进行相应设计说明：

1. 需求：移动端 Token 永不失效，修改密码后须更换 Token

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。

对应该实际需求，分别针对 PC 端、移动端的 Token 设计策略为：

1> Auth 系统生成 Token 时，根据参数不同（PC/MOBILE）来生成唯一的 Token，Token 生成算法：

- PC：PC-USERCODE(须加密)-USERID-CREATIONDATE-RONDEM[6 位]
- Android：ANDROID-USERCODE(须加密)-USERID-CREATIONDATE-RONDEM[6 位]
- iOS：IOS-USERCODE(须加密)-USERID-CREATIONDATE-RONDEM[6 位]

2> Token 存入 Redis 时，设置不同的有效期

- PC:2 个小时失效，前端自行管理 Token 的生命周期（原因：Token 存在 cookie 里，web 的安全性较差）
- PC:2 个小时失效，前端自行管理 Token 的生命周期（原因：Token 存在 cookie 里，web 的安全性较差）
- 移动端（Android、iOS）：永不失效，修改密码后更换 Token

注：二期新增修改个人密码功能后，再进一步实现移动端更换 Token。

2. 需求：移动端分页显示时，相对于 PC 端，页面容量需要进行控制

   所有具有分页功能的 API 均须提供 pageSize 参数，前端可自由设置 pageSize

3. 需求：移动端页面的图片展示，需要进行显示大小的设置

   采用 Tengine + Lua + GraphicsMagick 实现自动化图片裁剪，通过 lua 脚本控制可用参数【8x8、80x80、800x800….】与原图自动进行等比压缩即可。比如：

   http://192.168.9.44/comment/8-3566787986543-7654678.jpg【原图】

   http://192.168.9.44/comment/8-3566787986543-7654678.jpg_80x80.jpg【自动压缩为 80x80】