旅游代理系统实验报告

刘朝洋　2018104076
周　芳　2018100439
李姿翰　2018100430

目录

一、实验内容

本次实验的实验目标是建立一个 REST 服务的分布式的旅游代理服务，系统需要包含如下功能：

企业端：企业端可以修改信息。企业包括：航空公司，酒店，租车公司和景点管理公司。

用户端。用户能够注册和登录。登陆后输入需求，包括出行和返程时间，旅游目的地、大概预算，期望的旅游计划等基本信息。系统能够规划并为用户推荐旅游方案。其中旅游方案包括：

[1]. 机票购买。根据时间、出发时间、到达城市、用户偏好等等，推荐航班。

[2]. 酒店预定，根据时间、城市、偏好等等，推荐酒店房间。

[3]. 租车，根据租车地点、车型、手动挡/自动挡，为用户推荐合适的租车公司和车型。

[4]. 景点，根据旅游目的地推荐当地知名景点。

[5]. 导游。根据为用户推荐的景点，为用户推荐擅长讲解这个景点的导游。

二、实验设计

由于本次实验目标是实现 REST 服务的分布式系统，我们小组在 spring boot 框架的基础上，进行了实验的设计。首先考虑到要实现服务商的动态加入和离开，系统的整体架构设计如图 1 所示：

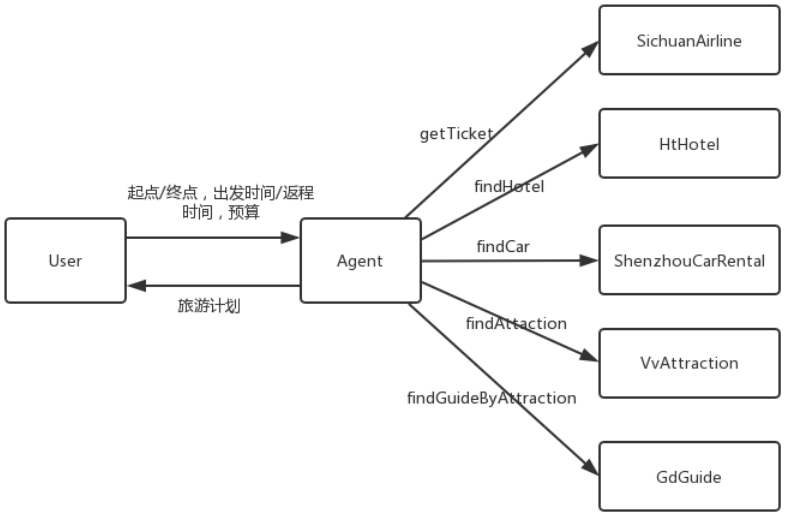

图 1 系统结构

系统主要分为三部分：用户、代理商、底层服务提供者，下面我们将分别讲述每个部分的设计。

2.1 用户层设计

对于用户而言，整个系统是一个“瘦客户-胖服务端”的设计，用户端只需要通过 web 输入对于旅游计划的需求，提交后会获得一个系统规划的旅行计划。其实需求包括：出发时间、返程时间、出发地点、目的地、预算、穷游/尊享游。旅行计划中应当包括往返机票信息、住宿酒店推荐、租车信息推荐、景点推荐、导游推荐、和最终的预算。此外，我们加入了认证系统和权限控制，只有已注册且登录的用户，才能输入需求获得计划。

基于上述条件，我们设计了 user 表，用于保存用户的账户信息，设计了 Userrequest 类用于获得用户的需求。

```
mysql> desc user;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| user_id  | int(11)      | NO   | PRI | NULL    | auto_increment |
| active   | int(11)      | YES  |     | NULL    |                |
| email    | varchar(255) | NO   |     | NULL    |                |
| name     | varchar(255) | NO   |     | NULL    |                |
| password | varchar(255) | NO   |     | NULL    |                |
| userType | varchar(255) | NO   |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
6 rows in set (0.06 sec)
```

图 2 user 表结构

用户由 email 唯一标识，对于用户密码，我们会进行加密处理再存到数据库中。User 表中有一项 userType， 用于标识用户是普通用户还是 admin 账户。

```
public class Userrequest {

    private String aim;

    private String departure_time;

    private String return_time;

    private String depature_point;

    private int cost;

    ……
}
```

Userrequest 类的定义如上所示，Userrequest 不会存储在数据库中，定义这个类是为了便于与前端界面进行数据交互。

2.2 agent 层设计

在 agent 层，为了能够知道当前有哪些服务商可以提供数据，我们定义一张服务表 agentservices 来存储当前可用的服务商的信息。表中每一行是一个服务商的信息，包括:

serviced ： 服务商的 id

serviceType ： 服务类型（eg：airline）

serviceName：服务商名称

serviceIP：服务商的 ip

```
mysql> desc agentservices;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| serviceID   | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| serviceIP   | varchar(255)| NO   |     | NULL    |                |
| serviceName | varchar(255)| NO   |     | NULL    |                |
| serviceType | varchar(255)| NO   |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

图 3 服务商注册表 agentservices 表结构

当 agent 收到用户的需求后，首先应当检查用户的旅游计划是穷游版还是尊享版，决定调用 plan A 还是 plan B 来根据用户需求规划旅游计划。

不论是 plan A 还是 plan B，流程都是一致的：

1. 首先遍历 agentservices 表，找到所有 serviceType 为 airline 的服务商。对于每一个 airline 的服务商，根据 serviceIP 和 serviceName，通过 RestTemplate 访问对应服务商提供的接口，获得最便宜/最舒适的机票信息。比较每个服务商返回的机票信息，保留最便宜/最舒适的机票。
2. 遍历 agentservices 表，找到所有 serviceType 为 hotel 的服务商，对每个服务商，根据目的地城市，找到最便宜/星级最高的酒店，比较所有服务商返回的酒店信息，留下最优的酒店信息。
3. 遍历 agentservices 表，找到所有 serviceType 为 carrental 的服务商，根据指定租车地和还车地，找到价格最低/最高的租车信息，比较所有服务商返回的信息，留下最优的信息。
4. 遍历 agentservices 表，找到所有 serviceType 为 attraction 的服务商，根据目的地，找到星级最高的景点信息，
5. 遍历 agentservices 表，找到所有 serviceType 为 guide 的服务商，对每一个服务商，根据第 4 步中找到的景点的 attractionId，找到评分最高的导游，比较每个服务商提供的信息，找出评分最高的导游，并保留相关信息。

整合上面得到的信息，将详细的机票信息、酒店信息、租车信息、景点信息和导游信息，传递给前端，展示给用户。

2.3 底层服务设计

对于底层的服务而言，需要实现 agent 层指定的接口，并把整个服务打包成 war 包，发布在 tomcat 服务器上，再将 tomcat 服务的 ip，自己的服务类型和名称注册到 agent 层的 agentservices 表中。对于不再提供服务的服务商，应当先在 agentservices 表中删除自己的记录，再从 tomcat 服务器上删除自己的记录。

不论是机票、租车、酒店、景点还是导游信息的服务商，设计时都按照

model-repository-service-controller

的四层结构进行设计，再加上服务的启动类和 config 包（config 中对页面的访问权限进行了控制）。

在这四层结构中，model 定义类，如 airline，类的每一个实例则对应一张机票的信息，并且 model 下的类定义，通过 Entity 将每一个实例与数据库中的每一行对应了起来，这样操作每个实例就等价的对数据中的行进行了增删改查。Repository 对 JpaRepository 进行了扩展，可以比较方便的通过函数名来进行一些基本的数据库操作，不需要自己管理与数据库

的通信。

2.3.1 airline 设计

```
mysql> desc airlines;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| airlineInfoid | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| airlineid     | varchar(255)| NO   |     | NULL    |                |
| airlinePrice  | float       | NO   |     | NULL    |                |
| arrivalTime   | varchar(255)| NO   |     | NULL    |                |
| departure     | varchar(255)| NO   |     | NULL    |                |
| departureTime | varchar(255)| YES  |     | NULL    |                |
| destination   | varchar(255)| NO   |     | NULL    |                |
| duration      | float       | NO   |     | NULL    |                |
| remainNum     | int(11)     | NO   |     | NULL    |                |
| status        | int(11)     | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
10 rows in set (0.27 sec)
```

图 4 airline 表结构

针对航空公司，我们设计 airline 类，与之对应的数据库表是 airlines。Airline 有 10 个成员变量 airlinePrice，duration，status，分别表示记录号，航班号，航班出发地，航班目的地，航班出发时间，航班到达时间，航班价格，航行时间，航班状态。

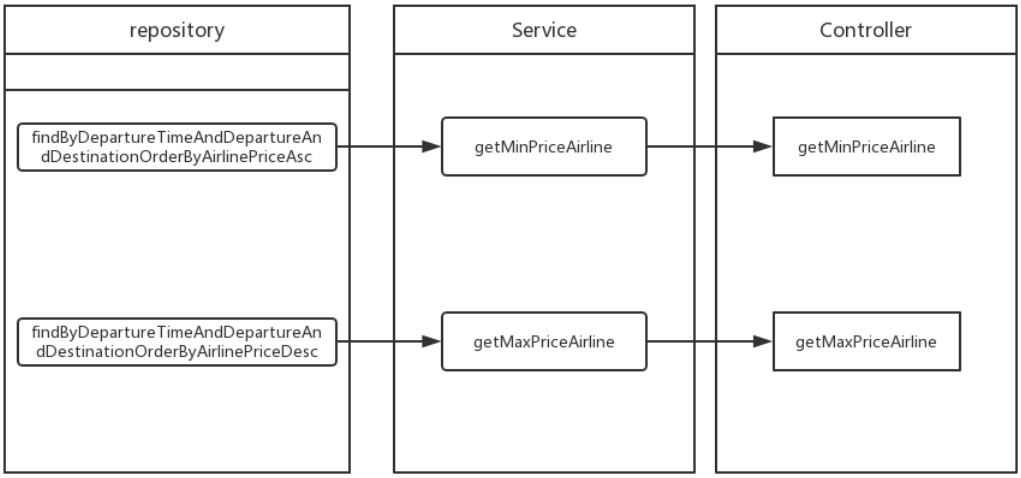

图 5 airline service 中的接口对应关系

2.3.2 hotel 设计

```
mysql> desc hotels;
+----------------+-------------+------+-----+---------+----------------+
| Field          | Type        | Null | Key | Default | Extra          |
+----------------+-------------+------+-----+---------+----------------+
| autoId         | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| hotelAddress   | varchar(255)| NO   |     | NULL    |                |
| appintmentTime | varchar(255)| NO   |     | NULL    |                |
| hotelCity      | varchar(255)| NO   |     | NULL    |                |
| hotelComment   | int(11)     | NO   |     | NULL    |                |
| hotelPrice     | bigint(20)  | NO   |     | NULL    |                |
| hotelRemaining | int(11)     | NO   |     | NULL    |                |
| hotelStar      | int(11)     | NO   |     | NULL    |                |
| hotelState     | varchar(255)| NO   |     | NULL    |                |
| hotelId        | int(11)     | NO   |     | NULL    |                |
| hotelName      | varchar(255)| NO   |     | NULL    |                |
| hotelPhone     | varchar(255)| NO   |     | NULL    |                |
+----------------+-------------+------+-----+---------+----------------+
12 rows in set (0.05 sec)
```

图 6 hotels 表结构

针对酒店，我们设计了 hotel 类，与之对应的数据库表名为 hotels。hotels 有 12 个成员变量 autoId, hotelId, hotelName, hotelPhone, hotelState, hotelCity, appintmentTime, hotelRemaining，hotelStar，hotelComment，hotelPrice，分别表示了酒店的自增 id，id，酒店名，联系电话，所在省份，所在城市，预定时间，剩余数量，星级，评分和日租金

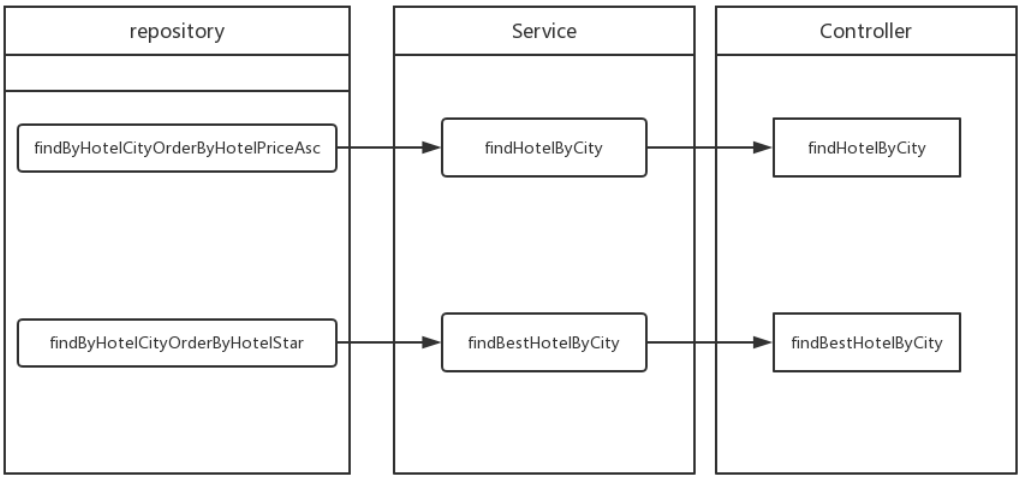

图 7 hotel service 中的接口对应关系

2.3.3 carRental 设计

```
mysql> desc car_rental;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| carId        | int(11)      | NO   | PRI | NULL    | auto_increment |
| carBrand     | varchar(255) | NO   |     | NULL    |                |
| carPlate     | varchar(255) | NO   |     | NULL    |                |
| carPrice     | bigint(20)   | NO   |     | NULL    |                |
| carRemaining | int(11)      | NO   |     | NULL    |                |
| cartype      | int(11)      | NO   |     | NULL    |                |
| carName      | varchar(255) | NO   |     | NULL    |                |
| rentalLoc    | varchar(255) | NO   |     | NULL    |                |
| returnLoc    | varchar(255) | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
9 rows in set (0.24 sec)
```

图 8 carRental 表结构

针对租车公司，我们设计了 CarRental 类，与之对应的数据库表名为 car_rental。car_rental 有 9 个成员变量 carId，carPlate，carBrand，carName，cartype，rentalLoc，returnLoc，carRemaining，carPrice，分别表示了车的 id，车牌，品牌，型号，自动挡/手动挡，租车地点，还车地点，剩余数量和日租金 。

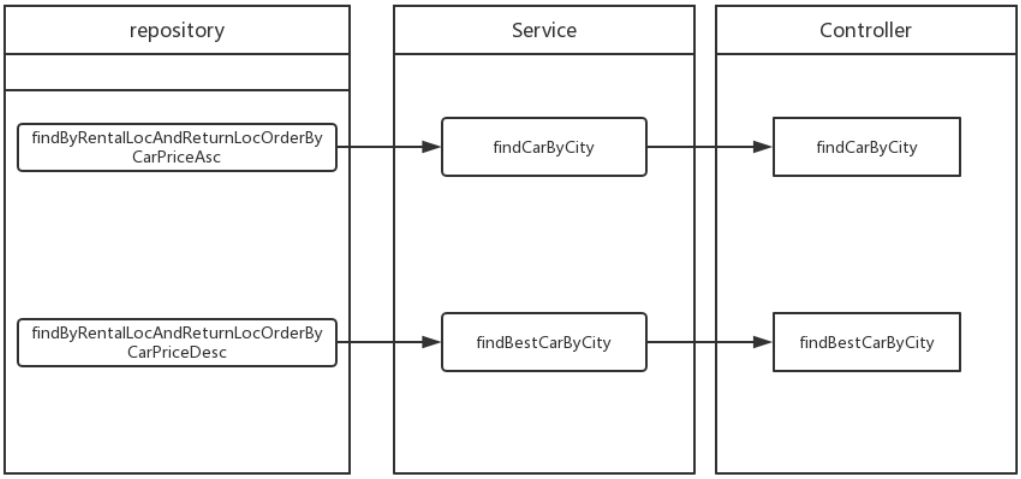

图 9 carRental service 中的接口对应关系

2.3.4 attraction 设计

```
mysql> desc attractions;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| attractionId      | int(11)      | NO   | PRI | NULL    | auto_increment |
| attractionAddress | varchar(255) | NO   |     | NULL    |                |
| attractionCity    | varchar(255) | NO   |     | NULL    |                |
| attractionComment | varchar(255) | NO   |     | NULL    |                |
| attractionPrice   | bigint(20)   | NO   |     | NULL    |                |
| attractionStar    | int(11)      | NO   |     | NULL    |                |
| attractionState   | varchar(255) | NO   |     | NULL    |                |
| attractionName    | varchar(255) | NO   |     | NULL    |                |
| attractionPhone   | varchar(255) | NO   |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
9 rows in set (0.11 sec)
```

图 10 attraction 表结构

针对景点，我们设计了 attraction 类，与之对应的数据库表名为 attractions。attractions 有 9 个成员变量 attractionId，attractionName，attractionPhone，attractionState，attractionCity，attractionAddress，attractionStar，attractionComment，attractionPrice 分别表示了景点的 id，景点名，联系电话，所在省份，所在城市，星级，评分和票价。

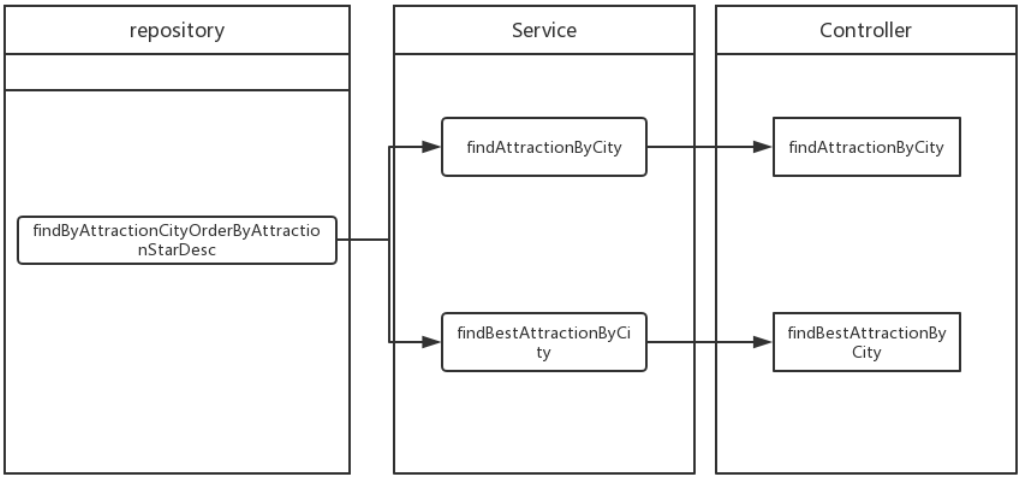

图 11 attraction service 中的接口对应关系

2.3.5 guide 设计

```
mysql> desc guides;
+--------------------+--------------+------+-----+---------+----------------+
| Field              | Type         | Null | Key | Default | Extra          |
+--------------------+--------------+------+-----+---------+----------------+
| autoId             | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| guideAge           | int(11)      | NO   |     | NULL    |                |
| guideAttractionId  | bigint(20)   | NO   |     | NULL    |                |
| guideCity          | varchar(255) | NO   |     | NULL    |                |
| guideComment       | int(11)      | NO   |     | NULL    |                |
| guideGender        | bit(1)       | NO   |     | NULL    |                |
| guideId            | bigint(20)   | NO   |     | NULL    |                |
| guideName          | varchar(255) | NO   |     | NULL    |                |
| guidePhone         | varchar(255) | NO   |     | NULL    |                |
| guidePrice         | int(11)      | NO   |     | NULL    |                |
| guideState         | varchar(255) | NO   |     | NULL    |                |
+--------------------+--------------+------+-----+---------+----------------+
11 rows in set (0.12 sec)
```

图 12 guide 表结构

针对导游，我们设计了 guide 类，与之对应的数据库表名为 guides。guides 有 11 个成员变量 autoId, guideId, guideGender, guideAge, guideName, guidePhone, guideState, guideCity，guideAttractionId，guideComment，guidePrice 分别表示了导游的自增 id，id，性别，年龄，姓名，电话，所在省份，所在城市，擅长的景点 id，评分和价格。

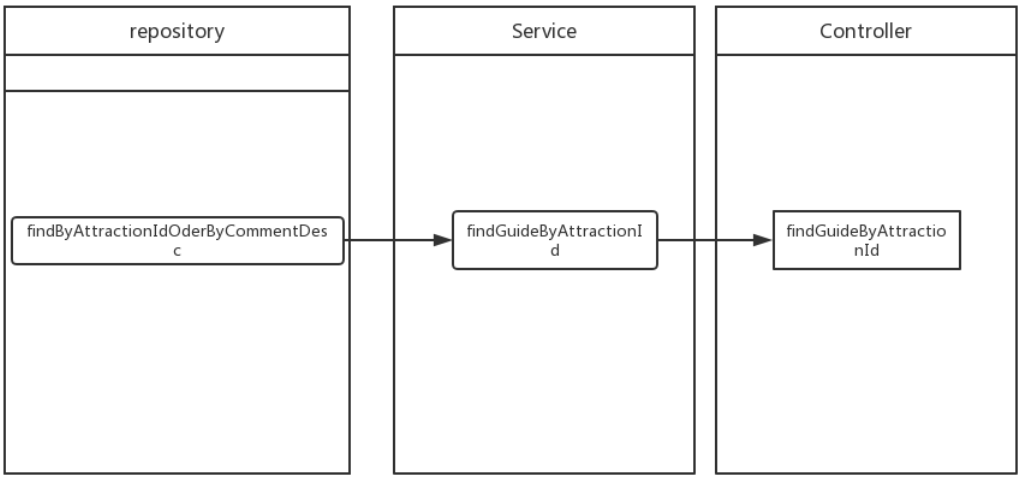

图 13 guide service 中的接口对应关系

三、实验亮点

在本次实验中，除了作业规定的内容之外，我们小组做了一些额外的功能，比如考虑到了系统的安全性。

在 Web 应用开发中，安全一直是非常重要的一个方面。安全虽然属于应用的非功能性需求，但是应该在应用开发的初期就考虑进来。如果在应用开发的后期才考虑安全的问题，

就可能陷入一个两难的境地：一方面，应用存在严重的安全漏洞，无法满足用户的要求，并可能造成用户的隐私数据被攻击者窃取；另一方面，应用的基本架构已经确定，要修复安全漏洞，可能需要对系统的架构做出比较重大的调整，因而需要更多的开发时间，影响应用的发布进程。因此，从应用开发的第一天就应该把安全相关的因素考虑进来，并在整个应用的开发过程中。

Spring Security 基于 Spring 框架，提供了一套 Web 应用安全性的完整解决方案。一般来说，Web 应用的安全性包括用户认证（Authentication）和用户授权（Authorization）两个部分。用户认证指的是验证某个用户是否为系统中的合法主体，也就是说用户能否访问该系统。用户认证一般要求用户提供用户名和密码。系统通过校验用户名和密码来完成认证过程。用户授权指的是验证某个用户是否有权限执行某个操作。在一个系统中，不同用户所具有的权限是不同的。比如对一个文件来说，有的用户只能进行读取，而有的用户可以进行修改。一般来说，系统会为不同的用户分配不同的角色，而每个角色则对应一系列的权限。

1. 用户密码加密

对于用户的密码，由于是高敏数据，我们采用了 spring Security 提供的密码加密方案——注册用户时，使用 SHA-256+随机盐+密钥把用户输入的密码进行 hash 处理，得到密码的 hash 值，然后将其存入数据库中。用户登录时，密码匹配阶段并没有进行密码解密（因为密码经过 Hash 处理，是不可逆的），而是使用相同的算法把用户输入的密码进行 hash 处理，得到密码的 hash 值，然后将其与从数据库中查询到的密码 hash 值进行比较。如果两者相同，说明用户输入的密码正确。

2. 用户权限管理

Spring security 提供了一个用户权限管理的框架，可以很容易对用户的权限进行管理。Spring Security 提供了针对具体 URL 的权限控制的设置接口，可以比较容易地针对项目需要对相应 URL 和用户进行权限控制。

3. 多种旅游计划规划

在用户输入需求时，用户可以指定希望获得穷游版的旅游计划还是舒适度更高的尊享版旅游计划

四、小组分工

刘朝洋：底层服务设计、报告撰写
周芳：agent 层、报告撰写
李姿翰：前端、数据库设计、ppt 制作、