TakeMeAway接口文档

@LocateViewController

· 简述：确定用户位置。主要分为手动输入地址搜索、通过定位功能定位、显示历史位置等功能。

· “饿了么”的手动输入地址搜索功能是需要网络交互的，这里应该向服务器发起查询请求，并获取返回的地址。这些地址提供外卖服务，记载在服务器的数据库中。提供外卖服务地址搜索接口。

1 接口设计：

HTTP Method: GET

请求参数：地址名*locationName*或一部分（全部或部分，字符串形式）

返回结果：搜索到的地址数组（*JSON*格式，地址内容包括地址名、经纬度）

返回结果示例：

[

*{“name” : “*华南师范大学西门*”, “latitude” : “100.0”, “longitude” : “100.0”},*

*{“name” : “*华南师范大学正门*”, “latitude” : “100.0”, “longitude” : “100.0”}*

]

说明：这里的地址是本公司提供外卖服务的站点

字段说明：

name	地址名
latitude	纬度
longitude	经度

@StoresTableViewController

· 简介：根据用户的当前位置搜索出附近的所有商家。

· 根据地址获取附近商家列表

2 接口设计：

HTTP Method: GET

请求参数：经度*longitude*，纬度*latitude*

返回结果：商家列表（*JSON*格式）

返回结果示例：

[

{“id” : “0001”, “logoURL” : “http://…”, “name” : “牛魔王”, “score” : “4.0”, “detail” : “100m / 100元起送 / 20分钟”, “state” : “1”},

{“id” : “0002”, “logoURL” : “http://…”, “name” : “麻辣烫”, “score” : “5.0”, “detail” : “200m / 200元起送 / 30分钟”, “state” : “0”}

]

说明：

（1）*detail*必须按照“空格 斜杠 空格”的形式分隔开来；*detail*中的三项如果有其中一项没有记录，就返回“未知”，不能有任何一项留空；也绝不要添加多余的东西；*detail*中不能出现额外的”/”字符。目的是客户端分割字符串。

（2）*state*表示商家的状态，*0*表示不营业（不接受任何外卖订单），*1*表示营业中。

字段说明：

id	商家的标识字符串
logoURL	商标存放的网址
name	商家名字
score	商家评分
detail	商家的详细信息（距离当前位置多少米，多少前才送，平均送达时间）
state	商家的状态，是否营业

@ShoppingViewController

· 简介：显示商店的美食列表和提供订购功能。

· 根据商家标识信息获取其美食列表

3 接口设计：

HTTP Method:GET

请求参数：商家的标识信息（即商家的*id*）

返回结果：商家的美食列表（*JSON*格式）

返回结果示例：

[

[“粥”, “粉”, “面”, “饭”],

{“id” : “0001”, “name” : “叉烧粉” , “sales” : “100”, “price” : “10.0”, “score” : “4.0”, “state” : “1”, “FoodType” : “粥”},

{“id” : “0002”, “name” : “炒饭” , “sales” : “200”, “price” : “15.0”, “score” : “5.0” , “state” : “0” “FoodType” : “粉”}

]

注意：

（1）*sales, price, score*三个*value*不能带有任何文字，*sales*为月销量

（2）数组中的首项是食物类别，剩余项为食物列表

字段说明：

id	美食的标识字符串
name	美食名称
sales	月销量
price	价格
state	当前状态，是否已经售完
FoodType	美食的类型，由商家自定，例如白粥属于粥类

@ConfirmOrderViewController

· 简介：该页面用于确认订购信息。

· 生成并确认订单

4 接口设计：

HTTP Method:POST

请求参数：商家的*ID*，未确认的美食列表的*json*字符串，订餐客户的信息

未确认美食列表的*json*字符串示例：

[

{“FoodID” : “001”, “OrderQuantity” : “1”},

{“FoodID” : “002”, “OrderQuantity” : “2”}

]

订餐客户信息的*json*字符串

{

“username” : “abc”,

“userType” : “1”

*“address” : “*华南师范大学*”, //*客户的地址

*“tel” : “10086”, //*客户的电话

*“arriveTime” : “*立即送出*” //*送达时间

*“comments” : “*不加葱*” //*备注

*“payment” : “*货到付款*” //*支付方式

}

请求参数示例：*storeID=0001&unconfirmShoppingList=JSONString&clientInfo=JSONString2*

返回结果：生成订单成功后，返回该订单的部分信息

返回结果示例：

{“orderID” : “0001”, “orderTime” : “2014-01-01 10:00:00”, ~~*“arriveTime” : “2014-01-01 12:00:00”,*~~ *“storeID” : “0004”, “state” : “0”}*

注意：

（*1*）*state = 0/1/2/3*，*0*表示下单成功，*1*表示商家已确认，*2*表示完成订单，*3*表示订单已取消

（*2*）统一时间的显示格式为：*yyyy-MM-dd hh:mm:ss*

字段说明：

FoodID	美食的标识字符串
OrderQuantity	该美食的订购数量
username	客户的用户名（如果是游客，则为由服务器生成的唯一标识字符串）
userType	客户的类型（游客，普通类型等）
address	客户的地址
tel	客户的电话
arriveTime	客户希望什么时候开始送外卖
comments	订单的备注
payment	付款方式，暂时只有货到付款一种
storeID	商家的标识字符串
unconfirmShoppingList	当前订单中的美食列表
clientInfo	客户的信息
orderiD	订单的标识字符串
orderTime	下单的时间
state	当前订单的状态

5 接口设计

HTTP Method:GET

请求参数：空

返回结果：游客的唯一标识符*anonymousUserID*，客户端在发起某些请求（如下单确认，查询历史订单内容等）时，需要使用该值，首先查询本地的数据库或文件中是否存有该值，如果有就不用调用本接口，如果没有就调用本接口，由服务器生成一个唯一的游客字符串并作为结果返回。

@FoodDetailViewController

· 简介：显示美食的细节

· 根据美食的ID从网络获取其完整的信息，包括获取评论和完整的美食信息

6 接口设计：

HTTP Method:GET

请求参数：唯一标识美食的*ID*

返回结果：评论列表和美食的详细信息

返回结果示例：

{

“detailPhotoURL” : “http://...”,

“comments” : [

{“username” : “abc”, “userType” : “1”, “commentTime” : “2014-1-1 12:00:00”, “content” : “好吃”, “score” : “5.0”},

{“username” : “def”, “userType” : “2”, “commentTime” : “2014-2-2 13:00:00”, “content” : “赞”, “score” : “4.8”}

]

}

detailPhotoURL	美食照片存放的网址
comments	用户的评论
username	用户名
userType	用户的类型
commentTime	评论时间
content	评论的内容
score	对该美食的评分

@StoreDetailViewController

· 简介：显示商家的细节

· 根据商家的ID从网络获取其完整的信息，包括获取店面图片和联系方式等

7 接口设计：

HTTP Method:GET

请求参数：唯一标识商家的*ID*

返回结果：商家的店面图片和联系方式

返回结果示例：

{

*“detailPhotoURL” : “http://...”, //*商家店面，具体情况，

*“location” : “*华南师范大学西门外津津美食*”,*

“tel” : “020-6666666”,

*“intro” : “*这是一家美食店*” //*商家的简介

~~,“email” : “abc@abc.com”~~

}

字段说明：

detailPhotoURL	商家店面或店内的照片存放的网址
location	商家所处的地理位置
tel	商家的联系电话
intro	商家的简介

@OrdersTableViewController

· 简介：显示用户的所有订单。如果是非登录用户，直接从数据库中提取订单信息；如果是登录用户，从网络中获取订单信息。

· 从网络获取登录用户的历史订单信息

8 接口设计：

HTTP Method:POST

请求参数：唯一标识用户的字符串，用户类型（如果是已经登录的用户，就用用户名加具体的用户类型。如果是游客，就用接口*5*生成的标识字符串，此时用户类型为*0*）

返回结果：历史订单信息的*JSON*数据

返回结果示例：

[

*{“orderID” : “20140101120000”, “orderTime” : “2014-01-01 12:00:00”, “storeID” : “0001”, “storeLogoURL” : “http://...”, “storeName” : “*津津美食*”, “arriveTime” : “2014-01-01 12:30:00”, “state” : “0”},*

*{“orderID” : “20140404120000”, “orderTime” : “2014-04-04 12:00:00”, “storeID” : “0001”, “storeLogoURL” : “http://...”, “storeName” : “*牛魔王*”,“arriveTime” : “2014-04-04 12:40:00”, “state” : “1”}*

]

字段说明：

orderID	订单的唯一标识字符串
orderTime	订单的下单时间
storeID	商家的唯一标识字符串
storeLogoURL	商家*logo*存放的网址
storeName	商家的名字
state	当前订单的状态

@OrderDetailViewController

· 简介：显示订单的详细信息

· 根据订单的ID刷新订单状态和时间。

9 接口设计：

HTTP Method:POST

请求参数：唯一标识订单的*ID*

返回结果：订单的状态和状态更新时间

返回结果示例：

[

{“state” : “0”, “time” : “2014-01-01 12:00:00”} // 下单成功和下单成功

{“state” : “1”, “time” : “2014-01-01 12:05:00”} // 商家确认和确认时间

{“state” : “2”, “time” : “2014-01-01 12:20:00”} // 订单完成和完成时间

或*{“state” : “3”, “time” : “2014-01-01 12:10:00”}* // 取消订单和取消时间

]

注：订单状态是有序的：下单成功 —— 商家确认 —— 订单完成，或下单成功 —— 商家确认 —— 订单取消。

· 取消订单功能

10 接口设计：

HTTP Method:POST

请求参数：唯一标识订单的*ID*

返回结果：*HTTP StatusCode*，即*HTTP*状态码，*200*表示操作成功，*4xx*或*5xx*表示操作失败。

@MenuDetailViewController

· 简介：显示订单的美食列表

· 根据订单ID发起网络请求，获取该订单的美食列表

11 接口设计：

HTTP Method:GET

请求参数：唯一标识订单的*ID*

返回结果：该订单对应的美食列表（*JSON*列表）

返回结果示例：

[

{“id” : “0001”, “name” : “叉烧粉” , “sales” : “100”, “price” : “10.0”, “score” : “4.0”, “state” : “1”, “FoodType” : “粥”},

{“id” : “0002”, “name” : “炒饭” , “sales” : “200”, “price” : “15.0”, “score” : “5.0” , “state” : “0” “FoodType” : “粉”}

]

注意：*sales, price, score*三个*value*不能带有任何文字

@LoginViewController

· 简介：登录功能

· 非第三方用户登录功能

12 接口设计：

HTTP Method:POST

请求参数：用户名，密码
返回结果：登录后的用户信息
返回结果示例：
{“username” : “abc”, “usertype” : “0”, “score” : “200”, “thumbnailURL” : “http://…”, “backgroundURL” : “http://...”}
注：*usertype*中*0*表示游客，*1*表示普通用户，*2*表示*qq*用户，*3*表示微信用户，*4*表示微博用户
字段说明：

username	用户名
userType	用户的类型
score	用户的积分
thumbnailURL	用户头像存放的网址
backgroundURL	用户头像背景存放的网址

@RegisterViewController

· 简介：注册功能

· 注册功能

13 接口设计：
HTTP Method:POST
请求参数：用户名，密码，邮箱
返回结果：*HTTP StatusCode*
注：直接注册的用户类型都是普通用户

@SecuritySettingsViewController

· 简介：更改用户密码

· 修改用户密码

14 接口设计：
HTTP Method:POST
请求参数：用户名，用户类型，当前密码，新密码
请求参数示例：*username=abc&usertype=0&curPass=1234&newPass=123456*
返回结果：*HTTP Status Code*
字段说明：

curPass	当前密码
newPass	新的密码

· 退出登录

15 接口设计：

HTTP Method:POST

请求参数：用户名，用户类型

请求参数示例：*username=abc&usertype=0*

返回结果：*HTTP Status Code*

@FavouritesViewController

· 简介：显示个人收藏

· 根据用户名和用户类型获取美食收藏数据

16 接口设计：

HTTP Method:POST

请求参数：用户名，用户类型

返回结果：美食收藏列表

返回结果示例：

[

{“id” : “0001”, “name” : “叉烧粉” , “sales” : “100”, “price” : “10.0”, “score” : “4.0”, “detailPhotoURL” : “http://…”, “storeID” : “0001”},

{“id” : “0002”, “name” : “炒饭” , “sales” : “200”, “price” : “15.0”, “score” : “5.0” , “detailPhotoURL” : “http://…”, “storeID” : “0002”}

]

· 根据用户名和用户类型获取商家收藏数据

17 接口设计：

HTTP Method:POST

请求参数：用户名，用户类型

返回结果：商家收藏列表

返回结果示例：

[

{“id” : “0001”, “logoURL” : “http://…”, “name” : “牛魔王”, “score” : “4.0”, “detail” : “100m / 100元起送 / 20分钟”},

{“id” : “0002”, “logoURL” : “http://…”, “name” : “麻辣烫”, “score” : “5.0”, “detail” : “200m / 200元起送 / 30分钟”}

]