API开发文档

基于Django的家政服务平台项目API开发文档

一.管理后台

登录

方法 `POST` 请求地址: `passport/login/`

描述

> 登陆接口，使用ajax进行提交请求

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`user-name`	`string`	账号名，3位以上，支持邮箱与手机号登录
`user-password`	`string`	密码，6-12位
`email_code`	`string`	邮箱验证码
`phone_code`	`string`	手机验证码

成功返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	返回 `"ok"`

失败返回

参数名	参数类型	描述
`reponse_data`	`HttpResponse`	登录失败提示信息

说明/示例

登录成功后，session保留了登录状态

邮箱验证

方法 `POST` 请求地址: `passport/email_auth/`

描述

> 邮箱验证，向邮箱发送验证码邮件，使用ajax进行提交请求

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`email`	`string`	注册时的邮箱
`operate_type`	`string`	验证码操作类型， `forget` 为找回密码, `register` 为注册

成功返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	发送成功，返回 `"ok"`

失败返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	发送失败提示信息

说明/示例

手机号验证

方法 `POST` 请求地址: `passport/phone_auth/`

描述

> 手机验证，向手机发送验证码短信，使用ajax进行提交请求

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`phone`	`string`	注册时的手机号
`operate_type`	`string`	验证码操作类型， `forget` 为修改密码, `register` 为注册

成功返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	发送成功，返回 `"ok"`

失败返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	发送失败提示信息

说明/示例

注册

方法 `POST` 请求地址: `passport/register/`

描述

> 注册接口，使用ajax进行提交请求

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`data`	`dict`	包含用户注册时所有信息
`user-name`	`string`	账号名，3位以上，支持邮箱与手机号登录
`user-password`	`string`	密码，6-12位
`phone-number`	`string`	手机号码，11位
`user-email`	`string`	邮箱，格式为邮箱格式
`prov`	`int`	省份代码
`city`	`int`	市代码
`county`	`int`	县代码
`address`	`string`	详细地址
`email_code`	`string`	邮箱验证码
`phone_code`	`string`	手机验证码

成功返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	注册成功，返回 `"ok"`

失败返回

参数名	参数类型	描述
`response_data`	`HttpResponse`	注册失败提示信息

说明/示例

二.服务列表

获取服务列表页面

方法 `GET` 请求地址: `services/?search={{key}}&page={{page}}`

描述

> 获取第 `page` 页的服务列表页面，通过关键字 `key` 搜索

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`name`	`string`	服务种类名

成功返回

参数名	参数类型	描述
`services`	`Service[]`	搜索结果 `Service` 对象列表
`services_num`	`int`	服务总个数
`key`	`string`	搜索关键字
`page_num`	`int`	当前的页面数

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回 `"404 Not Found"`

说明/示例

`Service`对象所含属性如下

```
    name = models.CharField('服务名称', max_length=32)
    price = models.DecimalField('服务价格', max_digits=10, decimal_places=2)
    status = models.BooleanField('服务状态')
    img1 = models.CharField('服务图片位置', max_length=255, unique=True, blank=True, null=True)
    img2 = models.CharField('服务图片位置', max_length=255, unique=True, blank=True, null=True)
    intro = models.CharField('服务简介', max_length=255, unique=True, blank=True, null=True)
    sales = models.IntegerField("销量", default=0)
    shop = models.ForeignKey('Shop', on_delete=models.CASCADE)  # 联接Shop表
    sort = models.ForeignKey('Type', on_delete=models.CASCADE)  # 联接Type表
```

获取 商品种类名 使用`services.下标.sort.name`

HTML用`for`循环显示图片

三.店铺

获取服务详情页面

方法`GET` 请求地址:`shop/service/?se_id={{service.id}}`

描述

> 获取服务详情页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`se_id`	`int`	服务id号

成功返回

参数名	参数类型	描述
`service`	`Service`	`Service`对象
`order_list`	`Order[]`	已评论该商品的订单列表

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回`"404 Not Found"`

说明/示例

`User`对象所含属性如下

```
class User(AbstractUser):
    #继承了AbstractUser的username,password,email
    phone = models.CharField(verbose_name='用户电话', max_length=32, unique=True, blank=True, null=True)
    province = models.IntegerField(verbose_name='省份', blank=True, null=True)
    city = models.IntegerField(verbose_name='城市', blank=True, null=True)
    district = models.IntegerField(verbose_name='区县', blank=True, null=True)
    details = models.CharField(verbose_name='详细地址', max_length=255, blank=True, null=True)
    mod_date = models.DateTimeField(verbose_name='Last modified', null=True, auto_now=True)

```

`Order`对象所含属性如下

```python
class Order(models.Model):
    service = models.ForeignKey('Service', null=True, blank=True, on_delete=models.SET_NULL)  # 联接Service表
    user = models.ForeignKey(User, null=True, blank=True, on_delete=models.SET_NULL)  # 联接User表
    create_time = models.DateTimeField('订单创建时间')
    start_time = models.DateTimeField('订单开始时间')
    end_time = models.DateTimeField('订单结束时间')
    pay_status = models.BooleanField('订单支付状态')
    comment = models.CharField(verbose_name='评价', max_length=255, blank=True, null=True)
    star = models.IntegerField(verbose_name='星级', blank=True, null=True)#1~5的整数
```

例:

- 根据`order`获取用户名`order.user.username`
- 根据`order`获取评论`order.comment`

获取订单支付页面

方法`GET`请求地址:`shop/service/pay/?se_id={{service.id}}&from_cart={{from_cart}}`

描述

> 获取订单支付页面，当未登录时会被重定向至登录页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`se_id`	`int`	服务id号
`from_cart`	`boolean`	是否通过购物车购买

成功返回

参数名	参数类型	描述
`services`	`Service[]`	`Service`对象列表
`user`	`User`	买家对象
`total_cost`	`double`	总金额

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回`"404 Not Found"`

说明/示例

四.账户信息

获取买家用户页面

方法`GET`请求地址:`/account/users/user_info_manage/`

描述

> 获取买家用户页面，当用户未登录时被重定向至登录页面，当用户身份为商家时被重定向至`account/vendors/vendor_info_manage/`

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述

成功返回

参数名	参数类型	描述
`user`	`User`	`User` 对象
`order_list`	`Order[]`	该用户的全部订单列表

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回 `"404 Not Found"`

修改买家个人信息页面

方法 `POST` 请求地址: `/account/users/user_info_manage/`

描述

> 提交修改内容，并刷新页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`username`	`string`	修改后的用户名
`email`	`string`	修改后的邮箱
`phone`	`string`	修改后的手机号
`prov`	`int`	修改后的省份代码
`city`	`int`	修改后的市代码
`county`	`int`	修改后的县代码
`addr`	`string`	修改后的详细地址

成功返回

参数名	参数类型	描述
`user`	`User`	修改后的`User`对象
`order_list`	`Order[]`	该用户的全部订单列表

失败返回

参数名	参数类型	描述
`user`	`User`	修改前的`User`对象
`order_list`	`Order[]`	该用户的全部订单列表
`msg`	`string`	返回无法修改原因

修改商家/买家密码页面

方法`POST`请求地址:`/account/change_psw/`

描述

> 用Ajax提交修改的密码

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`old_password`	`string`	原密码
`new_password`	`string`	新密码

成功返回

参数名	参数类型	描述
`user`	`User`	`User` 对象
`order_list`	`Order[]`	该用户的全部订单列表
`return_msg`	`HttpResponse`	返回 修改密码成功

失败返回

参数名	参数类型	描述
`return_msg`	`HttpResponse`	返回修改密码失败的原因: 原密码错误

获取买家购物车页面

方法 `POST` 请求地址: `/account/users/shop_cart/`

描述

> 获取购物车页面，当用户未登录时跳转到登录页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述
`service_id`	`int`	被删除的服务 `id` 号

成功返回

参数名	参数类型	描述
`user`	`User`	`User`对象
`services`	`Service[]`	购物车中的全部服务
`services_num`	`int`	购物车中的全部数量
`total_cost`	`double`	总金额

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回`"404 Not Found"`

说明/示例

删除购物车中选中服务

方法`GET`请求地址:`/account/users/shop_cart/`

描述

> 获取购物车页面，当用户未登录时跳转到登录页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述

成功返回

参数名	参数类型	描述
`user`	`User`	`User`对象
`services`	`Service[]`	购物车中的全部服务
`services_num`	`int`	购物车中的全部数量
`total_cost`	`double`	总金额

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回`"404 Not Found"`

说明/示例

获取商家用户页面

方法`GET`请求地址:`/account/vendors/vendor_info_manage/`

描述

> 获取商家用户页面，当用户未登录时被重定向至登录页面

请求头

参数名	参数类型	描述
`content-type`	`string`	HTML

请求参数

参数名	参数类型	描述

成功返回

参数名	参数类型	描述
`user`	`User`	`User` 对象
`order_list`	`Order[]`	该商家的全部订单列表
`shop`	`Shop`	该商家的店铺

失败返回

参数名	参数类型	描述
	`HttpResponse`	返回 `"404 Not Found"`

说明/示例

`Shop` 类所含属性如下

```
class Shop(models.Model):
    name = models.CharField('店铺名称', max_length=32, unique=True)
    create_time = models.DateTimeField('店铺创建时间')
    status = models.BooleanField('店铺状态')
    star = models.IntegerField(verbose_name='店铺星级', blank=True, null=True)
```

五.算法

推荐算法

描述

> 分析用户行为并返回推荐的服务

输入参数

参数名	必选	类型	说明
`user`	是	`User`	被推荐的用户对象
`services`	是	`Service[]`	所有服务的列表
`services_cart`	是	`Service[]`	被推荐的用户购物车中的商品
`orders`	是	`Order[]`	被推荐的用户买过的商品

返回参数

参数名	必选	类型	说明
`services`	是	`Service[]`	推荐给用户的服务顺序

说明/示例

`User` 对象所含属性如下

```
class User(AbstractUser):
    #继承了AbstractUser的username,password,email
    phone = models.CharField(verbose_name='用户电话', max_length=32, unique=True, blank=True, null=True)
    province = models.IntegerField(verbose_name='省份', blank=True, null=True)
    city = models.IntegerField(verbose_name='城市', blank=True, null=True)
    district = models.IntegerField(verbose_name='区县', blank=True, null=True)
    details = models.CharField(verbose_name='详细地址', max_length=255, blank=True, null=True)
    mod_date = models.DateTimeField(verbose_name='Last modified', null=True, auto_now=True)

```

`Order` 对象所含属性如下

```python
class Order(models.Model):
    service = models.ForeignKey('Service', null=True, blank=True, on_delete=models.SET_NULL)  # 联接Service表
    user = models.ForeignKey(User, null=True, blank=True, on_delete=models.SET_NULL)  # 联接User表
    create_time = models.DateTimeField('订单创建时间')
    start_time = models.DateTimeField('订单开始时间')
    end_time = models.DateTimeField('订单结束时间')
    pay_status = models.BooleanField('订单支付状态')
    comment = models.CharField(verbose_name='评价', max_length=255, blank=True, null=True)
    star = models.IntegerField(verbose_name='星级', blank=True, null=True)#1~5的整数
```

`Service`对象所含属性如下

```python
class Service(models.Model):
    name = models.CharField('服务名称', max_length=32)
    price = models.DecimalField('服务价格', max_digits=10, decimal_places=2)
    status = models.BooleanField('服务状态')
    img1 = models.CharField('服务图片位置', max_length=255, unique=True, blank=True, null=True)
    img2 = models.CharField('服务图片位置', max_length=255, unique=True, blank=True, null=True)
    intro = models.CharField('服务简介', max_length=255, unique=True, blank=True, null=True)
    sales = models.IntegerField("销量", default=0)
    shop = models.ForeignKey('Shop', on_delete=models.CASCADE)  # 联接Shop表
    sort = models.ForeignKey('Type', on_delete=models.CASCADE)  # 联接Type表
```

数据库查询操作举例：

查询`Service`的`name`

```python
se_name=Service.objects.filter(name='xxx')[0]
```

以`price`给`Service`排序

```python
se_order_by_price = Service.objects.order_by("price")
```

六.第三方接口