1.5 SOCIAL 模块开发

SOCIAL 功能概述

1. 交友模块
    - 获取推荐列表
    - 喜欢 / 超级喜欢 / 不喜欢
    - 反悔 (每天允许返回 3 次)
    - 查看喜欢过我的人
2. 好友模块
    - 查看好友列表
    - 查看好友信息

开发中的难点

1. 滑动需有大量用户，如何初始化大量用户以供测试?
2. 推荐算法
3. 如何从推荐列表中去除已经滑过的用户
4. 滑动操作，如何避免重复滑动同一人
5. 如果双方互相喜欢，需如何处理
6. 好友关系如何记录，数据库表结构如何设计?
7. 反悔接口
    1. "反悔"都应该执行哪些操作
    2. 每日只允许"反悔" 3 次应如何处理
    3. 后期运营时，如何方便的修改反悔次数
8. 内部很深的逻辑错误如何比较方便的将错误码返回给最外层接口

关系分析

1. 滑动者与被滑动者
    - 一个人可以滑动很多人
    - 一个人可以被多人滑动

    - 结论: 同表之内构建起来的逻辑上的多对多关系
2. 用户与好友
    - 一个用户由多个好友
    - 一个用户也可以被多人加为好友
    - 结论: 同表之内构建起来的逻辑上的多对多关系, Friend 表实际上就是一个关系表

模型设计参考

1. Swiped (划过的记录)

Field	Description
uid	用户自身 id
sid	被滑的陌生人 id
mark	滑动类型
time	滑动的时间

2. Friend (匹配到的好友)

Field	Description
uid1	好友 ID
uid2	好友 ID

类方法与静态方法

- `method`
    - 通过实例调用
    - 可以引用类内部的 **任何属性和方法**
- `classmethod`
    - 无需实例化
    - 可以调用类属性和类方法
    - 无法取到普通的成员属性和方法
- `staticmethod`
    - 无需实例化
    - **无法**取到类内部的任何属性和方法, 完全独立的一个方法

利用 Q 对象进行复杂查询

```
from django.db.models import Q

# AND
Model.objects.filter(Q(x=1) & Q(y=2))

# OR
Model.objects.filter(Q(x=1) | Q(y=2))

# NOT
Model.objects.filter(~Q(name='kitty'))
```

设计文档、业务逻辑

系统架构

1. Django + sqlite / mysql
2. 典型的 Web application
    1. NginX -> Django -> DB

需求分析

1. User
    1. 获取短信验证码：1个接口
    2. 验证短信验证码，登录：1个接口
        1. 如果无此用户，直接创建
        2. 同时完成登录和验证
    3. 获取资料：1个接口
    4. 更改资料：1个接口
    5. 上传头像：1个接口

数据模型

1. User
    1. 手机号：
        1. phonenum : varchar(20)
            1. +8613501234567
            2. 135-0123-4567
    2. 用户名
        1. nickname: varchar(50)

3. 性别
    1. gender / sex: varchar(20)
        1. male
        2. female
        3. unknown
        4. others
4. 生日
    1. birth_year: smallint(双字节)
    2. birth_month: tinyint(单字节)
    3. birth_day: tinyint(单字节)
5. 头像
    1. avatar: (URI: Uniform Resource Identifier): varchar(4096)
6. 居住地
    1. addr / location: varchar(1024)
    2. 规范：北京：beijing、bj、peking，朝阳，永泰庄

2. Profile
    1. 目标城市 varchar (10)
        1. 城市的区
        2. 定位：经纬度
            1. 从ip地址
        3. 用地图插件
        4. 用前端获取经纬度：app、html5、小程序
        5. 多个接口接收用户的经纬度位置
    2. min_distance
    3. max_distance
    4. min_age
    5. max_age
    6. dating_sex
    7. vibration
    8. only_match
    9. auto_play
    10. user_id

API接口定义

1. User
    1. 获取验证码：
        1. 描述：

2. 方法：GET (POST / PUT / DELETE)
3. 路径 / 路由 / URL : http://www.some.com/user/submit/phone
4. 参数：
    1. phonenum=13501234567
5. 返回值：

```
{
  "code": 0,
  "data": {
    "status": "ok"
  }
}
```

6. code范围：
    1. 0 正常
    2. 1000 服务器内部错误
    3. 2000 客户端错误
    4. 2001 北京号码不能注册

2. 校验验证码：
    1. 描述：
    2. 方法：POST
    3. 路由：http://www.some.com/user/submit/vcode
    4. 参数：
        1. phonenum=13512345678&vcode=2345
    5. response：
        1. { "code": 0, "data": { "status": "ok" } }
        2. { "code": 2001, "data": { "status": "code error" } }
3. 获取资料：
    1. 描述：登录后使用
    2. 方法：GET
    3. URL：api/user/get/profile
    4. 参数:
        1. 不需要提交其他信息
    5. Profile 的 json 数据：
        1. 用户个人资料
        2. 具体哪些字段?
4. 修改资料：
    1. 描述：登录后修改资料
    2. 方法：POST
    3. URL：api/user/set/profile

4. 请求：
    1. key=value
5. response：
    1. 我们修改的结果：成功还是失败

5. 上传头像：
    1. 描述：登录后使用
    2. 方法：POST
    3. URL：api/user/upload/avatar
    4. request：
        1. file
    5. response：
        1. ok

2. Social接口
    1. 获取推荐列表
    2. 喜欢、超级喜欢、不喜欢
    3. 反悔
    4. 查看喜欢过我的人

RESTFUL 风格对比

1. 混乱风格：project 1：
    1. GET http://www.some.com/user/1/edit
    2. GET http://www.some.com/user/1
2. 混乱风格：project 2:
    1. POST http://www.some.com/edit/user/1
    2. GET http://www.some.com/get/user/1
    3. GET http://www.some.com/user/1/edit
3. RESTful：
    1. GET http://www.some.com/user?id=1
    2. POST http://www.some.com/user
    3. PUT http://www.some.com/user
        1. body: gender=male

业务描述、业务流程、表结构

- 请介绍你做过的业务
- 请描述你的业务核心流程

- 请介绍你的业务逻辑结构
- 请画出核心表结构
- 针对你的业务，如果增加以下功能，该如何设计？
    - 电商：多件商品捆绑优惠促销
    - 游戏：多人对战
    - 社交：好友排行