# 文档三：详细设计文档

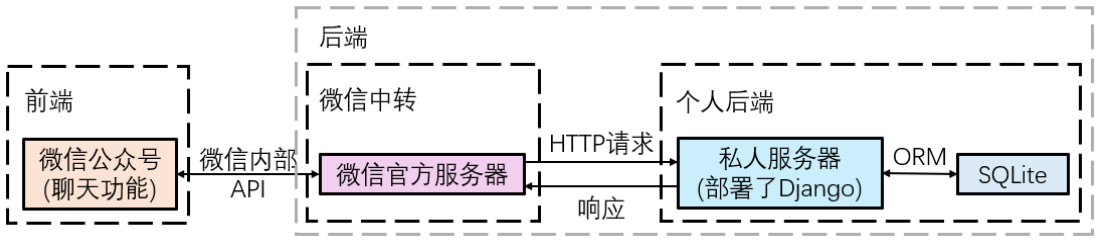
21301114 俞贤皓

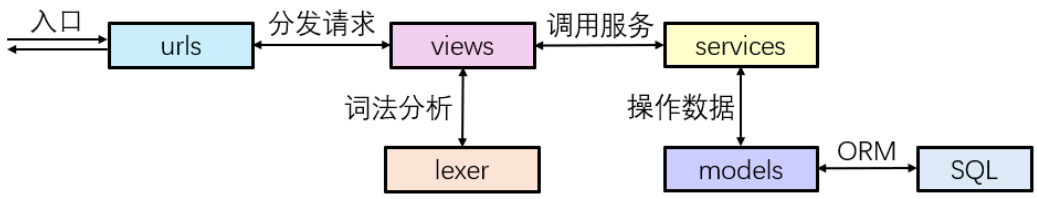
## 1. 系统方案设计

### 1.1 技术路线

- 前端：微信公众号，聊天
  - 部署高效，不需要额外申请并开发小程序。
- 后端：Django + Python
  - Django套件完善，开发高效，可以在短时间内开发出一个功能完善且可用的后端。
- 数据库：SQLite
  - 部署高效，不需要额外配置一个数据库服务。
  - 支持数百万级的数据记录，通常可以支持100人以上的用户，适合快速开发一个可用的软件。
  - 由于Django的ORM特性，所以就算以后需要切换为其他数据库，在代码层面也只需要改动几行。当用户量增加时，也可以兼容未来数据库的迭代。
- 命令行接口形式的交互
  - 因为在微信公众号中，用户会采用聊天的形式进行交互，所以我们可以将微信公众号看作一个Shell，并采用命令行接口的形式和系统交互。
  - 因此，我们在后端需要词法分析等工具来解析用户的命令。

### 1.2 系统架构图

- 外部系统架构
  - 

  - 前端：处理用户输入。
  - 微信中转：安全性校验、用户鉴权。
  - 个人后端：逻辑处理、响应请求、数据库交互等。
- 系统内部架构
  - 

- urls：分发不同URL请求至对应函数
- views：不同请求的入口函数
- lexer：词法分析，解析用户命令，并调用对应的服务
  - 因为采用命令行接口的形式进行交互，所以需要这个模块
- services：对models的封装，提供了所需的API
  - **与作业有关的主要API设计，均在 services模块 中实现**
- models：定义了数据库的表结构和一些基础accessor
- SQL：数据库

# 2. API设计

## 2.1 对内API设计

### 2.1.1 标准方法

- get_offer( filter ) -> Offer
  - 查询匹配Filter的第一个Offer
- list_offers( filter ) -> List[Offer]
  - 查询匹配Filter的Offer集合
- list_offers_with_page( filter, page ) -> (List[Offer], page, page_sum)
  - 分页，查询匹配Filter、并且位于对应页数的Offer集合
  - 返回值为Offer集合、页数、总页数
- create_offer( info ) -> int
  - 根据对应信息创建一个新的Offer
- update_offer( filter, info ) -> int
  - 根据Filter和对应信息，更新第一个匹配的Offer
- delete_offer( filter ) -> int
  - 根据Filter，删除第一个匹配的Offer
- replace_offer( filter, info ) -> int
  - 根据Filter和对应信息，整体替换第一个匹配的Offer

### 2.2.2 批量修改方法

- create_offers( List[info] ) -> int
  - 根据对应信息创建多个Offer
- update_offers( filter, info ) -> int
  - 根据Filter，更新所有匹配的Offer
- delete_offers( filter ) -> int
  - 根据Filter，删除所有匹配的Offer
- replace_offers( filter, info ) -> int

    - 根据Filter和对应信息，替换所有匹配的Offer

### 2.2.3 权限控制

- 此处的用户仅用于记录状态，例如爬虫用户、作弊用户等，不记录其他信息。
- get_user( username ) -> int
    - 查询一个用户
- create_user( username ) -> int
    - 创建一个用户
- get_user_state( username ) -> int
    - 该方法会返回用户状态
- check_user_state( username, command, update = False )
    - 该方法会返回，用户是否有权限执行本命令
    - 本接口应该在每次用户发出请求时进行调用，获取用户状态，以决定用户是否可以执行对应命令。
    - 为了便于使用，调用本接口时，**若启用了update参数，并将update参数设置为True**，则本接口开始会自动监控用户行为。若用户行为出现异常，该API会对用户进行封禁。
        - 为了遵循单一职责原则，默认不启用本功能
- reset_user_state( username ) -> int
    - 重置对应User的状态
- set_user_state( username , state ) -> int
    - 将User的状态设置为对应状态
- USER_STATE
    - User状态枚举值

```
USER_STATE_DEFAULT = 0b0000
USER_STATE_BANNED  = 0b0001
USER_STATE_SPIDER  = 0b0010
```

## 2.2 使用手册

- 本系统使用命令行接口的形式进行交互
- `help`：打印系统使用手册
    - 命令格式：`help`
- `commit`：提交一份Offer
    - 命令格式：`commit arg1 arg2 arg3 arg4`
        - `arg1`：Offer公司
        - `arg2`：Offer公司所在城市
        - `arg3`：Offer岗位

        - `arg4`：Offer薪资，必须为整数
    - 若命令格式正确，则会提示：`收到喵~`
    - 若命令格式错误，则会提示：`这个命令看不懂喵~~`
    - 示例

      ```
      commit 未来科技有限公司 上海 Unity3D客户端开发 114514
      ```

- `group-commit`：提交多份Offer
    - 命令格式：`group-commit arg0-1 arg0-2 arg0-3 arg0-4 [arg1-1 arg1-2 arg1-3 arg1-4 [arg2-1 ...]]`
- `query`：查询多个Offer
    - 命令格式：`query [--company com] [--city city] [--position pos] [--page page] [--sort-new] [--sort-salary]`
        - `--company`，若该选项存在，则会根据 **公司** 对Offer进行筛选
        - `--city`，若该选项存在，则会根据 **城市** 对Offer进行筛选
        - `--position`，若该选项存在，则会根据 **岗位** 对Offer进行筛选
        - `--page`，若该选项存在，则会根据 **页面编号** 进行分页。
            - 若该选项不存在，则页面编号默认为 1
        - `--sort-new`，若该选项存在，则按 **时间顺序**，从新到旧显示Offer。
        - `--sort-salary`，若该选项存在，则按 **薪资顺序**，从高到低显示Offer。
            - `--sort-salary` 会覆盖 `--sort-new`
    - 示例

      ```
      query
      query --page 5
      query --page 2 --company 未来科技有限公司
      query --sort-salary --sort-new --page 1 --city 南京 --company 谷雅丰帝国
      ```