# 督办仪表板 Python 后端 API 文档

## 概述

这是一个基于 Flask 的 Python 后端 API 服务，用于支持督办仪表板的各种功能，包括短信服务、数据处理和任务管理。

## 项目结构

督办/  
├── app.py # Flask 主应用  
├── run.py # 启动脚本  
├── config.py # 配置文件  
├── requirements.txt # Python 依赖  
├── .env.example # 环境变量示例  
├── api/ # API 模块  
│ ├── \_\_init\_\_.py  
│ ├── sms\_service.py # 短信服务 API  
│ ├── data\_service.py # 数据处理 API  
│ └── task\_service.py # 任务管理 API  
├── utils/ # 工具模块  
│ ├── \_\_init\_\_.py  
│ ├── sms\_client.py # 阿里云短信客户端  
│ ├── data\_processor.py # 数据处理工具  
│ └── validators.py # 数据验证工具  
├── uploads/ # 文件上传目录  
└── logs/ # 日志目录

## 安装和配置

### 1. 安装依赖

pip install -r requirements.txt

### 2. 环境配置

复制 .env.example 为 .env 并配置：

cp .env.example .env

编辑 .env 文件：

# 阿里云短信服务配置  
ALIBABA\_CLOUD\_ACCESS\_KEY\_ID=your\_access\_key\_id  
ALIBABA\_CLOUD\_ACCESS\_KEY\_SECRET=your\_access\_key\_secret  
  
# Flask 应用配置  
FLASK\_ENV=development  
FLASK\_DEBUG=True  
FLASK\_PORT=3000  
  
# 其他配置...

### 3. 启动服务

# 方式1: 使用启动脚本  
python run.py  
  
# 方式2: 直接运行应用  
python app.py  
  
# 方式3: 使用 Flask 命令  
flask run --host=0.0.0.0 --port=3000

## API 接口文档

### 基础信息

* **基础URL**: http://localhost:3000
* **内容类型**: application/json
* **字符编码**: UTF-8

### 1. 短信服务 API (/api/sms)

#### 1.1 发送单条短信

**接口**: POST /api/sms/send

**请求体**:

{  
 "phoneNumber": "13800000000",  
 "signName": "测试签名",  
 "templateCode": "SMS\_123456",  
 "templateParam": {"code": "1234"}  
}

**响应**:

{  
 "success": true,  
 "message": "短信发送成功",  
 "data": {  
 "request\_id": "xxx",  
 "biz\_id": "xxx",  
 "phone\_number": "13800000000"  
 }  
}

#### 1.2 批量发送短信

**接口**: POST /api/sms/batch-send

**请求体**:

{  
 "phoneNumbers": ["13800000000", "13800000001"],  
 "signName": "测试签名",  
 "templateCode": "SMS\_123456",  
 "templateParam": {"code": "1234"}  
}

#### 1.3 查询发送状态

**接口**: GET /api/sms/query/{biz\_id}?phoneNumber=13800000000

#### 1.4 测试连接

**接口**: GET /api/sms/test

### 2. 数据处理 API (/api/data)

#### 2.1 上传数据文件

**接口**: POST /api/data/upload

**请求**: multipart/form-data - file: 数据文件 (CSV, Excel)

**响应**:

{  
 "success": true,  
 "message": "文件处理成功",  
 "data": {  
 "columns": ["列1", "列2", "列3"],  
 "records": [...],  
 "total\_count": 100,  
 "statistics": {...}  
 }  
}

#### 2.2 过滤数据

**接口**: POST /api/data/filter

**请求体**:

{  
 "data": [...],  
 "filters": {  
 "status": "completed|incomplete|urgent",  
 "department": "部门名称",  
 "type": "任务类型"  
 }  
}

#### 2.3 导出数据

**接口**: POST /api/data/export

**请求体**:

{  
 "data": [...],  
 "format": "json|csv|excel",  
 "filename": "export\_data"  
}

#### 2.4 获取统计信息

**接口**: POST /api/data/statistics

### 3. 任务管理 API (/api/task)

#### 3.1 获取任务列表

**接口**: GET /api/task/list

**查询参数**: - status: 任务状态 (completed|incomplete|urgent|all) - department: 部门名称 - type: 任务类型 - page: 页码 (默认1) - limit: 每页数量 (默认20)

#### 3.2 创建任务

**接口**: POST /api/task/create

**请求体**:

{  
 "title": "任务标题",  
 "description": "任务描述",  
 "department": "负责部门",  
 "type": "任务类型",  
 "deadline": "2024-12-31",  
 "priority": "high|medium|low",  
 "assignee": "负责人",  
 "phone": "联系电话"  
}

#### 3.3 获取任务详情

**接口**: GET /api/task/{task\_id}

#### 3.4 更新任务

**接口**: PUT /api/task/{task\_id}

#### 3.5 删除任务

**接口**: DELETE /api/task/{task\_id}

#### 3.6 获取紧急任务

**接口**: GET /api/task/urgent

### 4. 系统接口

#### 4.1 健康检查

**接口**: GET /api/health

#### 4.2 API 信息

**接口**: GET /api/info

## 错误处理

所有 API 接口都遵循统一的错误响应格式：

{  
 "success": false,  
 "message": "错误描述",  
 "error": "详细错误信息"  
}

常见 HTTP 状态码： - 200: 成功 - 201: 创建成功 - 400: 请求参数错误 - 401: 未授权 - 403: 禁止访问 - 404: 资源不存在 - 405: 方法不允许 - 413: 请求实体过大 - 500: 服务器内部错误

## 日志记录

应用会自动记录以下信息： - 请求日志：记录所有 API 请求 - 错误日志：记录所有异常和错误 - 业务日志：记录重要的业务操作

日志文件位置：logs/app.log

## 安全特性

1. **CORS 支持**: 配置跨域访问控制
2. **请求验证**: 严格的参数验证和数据校验
3. **文件安全**: 文件上传安全检查
4. **错误处理**: 统一的错误处理和日志记录
5. **安全头**: 自动添加安全响应头

## 性能优化

1. **异步处理**: 支持并发请求处理
2. **文件缓存**: 上传文件的缓存机制
3. **数据分页**: 大数据集的分页处理
4. **连接池**: 数据库连接池优化

## 部署建议

### 开发环境

python run.py

### 生产环境

# 使用 Gunicorn  
pip install gunicorn  
gunicorn -w 4 -b 0.0.0.0:3000 app:app  
  
# 使用 uWSGI  
pip install uwsgi  
uwsgi --http :3000 --module app:app --processes 4

## 监控和维护

1. **健康检查**: 使用 /api/health 端点监控服务状态
2. **日志监控**: 定期检查 logs/app.log 文件
3. **性能监控**: 监控 API 响应时间和错误率
4. **资源监控**: 监控服务器 CPU、内存和磁盘使用情况