# 基于Socket文件传输的横向联邦学习软件仿真平台

一、系统说明

**1.技术背景：**

联邦学习本质上是一种分布式机器学习技术，或机器学习框架。我们把每个参与共同建模的企业称为参与方，根据多参与方之间数据分布的不同，把联邦学习分为三类：横向联邦学习、纵向联邦学习和联邦迁移学习。我们实现的仿真平台主要是对横向联邦学习的一个软件仿真。横向联邦学习的本质是样本的联合，适用于参与者间业态相同但触达客户不同，即特征重叠多，用户重叠少时的场景，比如不同地区的银行间，他们的业务相似（特征相似），但用户不同（样本不同）。

横向联邦学习的主要过程如下：

step1：参与方各自从服务器A下载最新模型；

step2：每个参与方利用本地数据训练模型，加密梯度上传给服务器A，服务器A聚合各用户的梯度更新模型参数；

step3：服务器A返回更新后的模型给各参与方；

step4：各参与方更新各自模型。

在传统的机器学习建模中，通常是把模型训练需要的数据集合到一个数据中心然后再训练模型，之后预测。在横向联邦学习中，可以看作是基于样本的分布式模型训练，分发全部数据到不同的机器，每台机器从服务器下载模型，然后利用本地数据训练模型，之后返回给服务器需要更新的参数；服务器聚合各机器上的返回的参数，更新模型，再把最新的模型反馈到每台机器。

在这个过程中，每台机器下都是相同且完整的模型，且机器之间不交流不依赖，在预测时每台机器也可以独立预测，可以把这个过程看作成基于样本的分布式模型训练。

**2.功能介绍**

根据上面所描述的横向联邦学习的特征和过程我们设计了一种基于参数文件传输的联邦学习软件仿真框架。整个框架由聚合服务器Server，本地训练客户端Client，通信模块FileControl，网页后端服务器WebServer四个模块组成，根据任务需求我们总结了以下各个模块的功能

1. 聚合服务器Server

* 接收本地模型参数
* 模型聚合计算
* 基于验证集测试并记录
* 发送全局模型参数
* 接受Web提交的初始化训练参数
* 保存训练所得模型权重文件
* 结束训练任务后重置服务器，等待下次训练任务

1. 本地训练客户端Client

* 接受全局模型参数
* 更新本地模型参数
* 发送本地模型参数
* 接收初始化训练参数

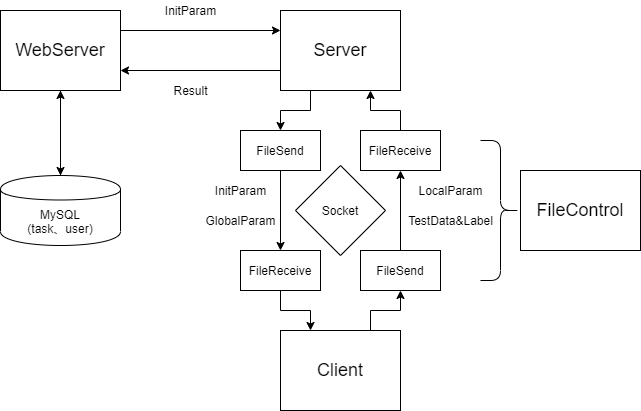
1. 通信模块FileControl

* 建立聚合服务器Server与本地训练客户端Client之间的连接
* 对参数文件进行发送和接收
* 对使用过的参数文件及时清除

1. 网页后端服务器WebServer

* 提供用户的注册和登录
* 接收训练任务的配置参数
* 发送训练任务配置参数给聚合服务器Server
* 将训练任务信息保存到MySQL数据库中

**3.模块设计图**

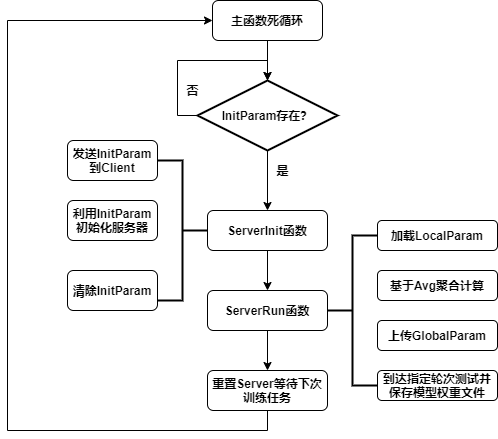


二、详细设计说明

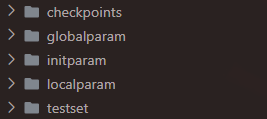
**1.聚合服务器Server**

1.1流程设计

通过分析上述模块的设计思路，我们将聚合服务器Server的工作过程设计如下图



1.2文件系统



checkpoints：保存训练结果

globalparam：保存全局参数

initparam：保存初始化参数

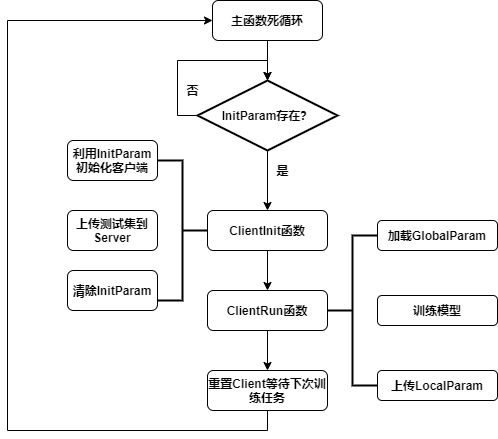
localparam：保存局部参数

testset：保存测试集

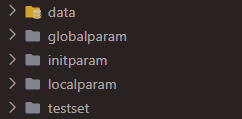
**2.本地训练客户端Client**

2.1流程设计

Client端的实现我们是一种基于伪分布式的实现，即在一台物理机上通过python对象模拟出多个客户端，然后串行的执行训练任务，这样可以简化很多的文件同步问题和端口配置问题。当然，我们也预留了一些函数接口，便于之后改经，实现对Client的拆分。



2.2文件系统



data：保存数据集

globalparam：保存全局参数

initparam：保存初始化参数

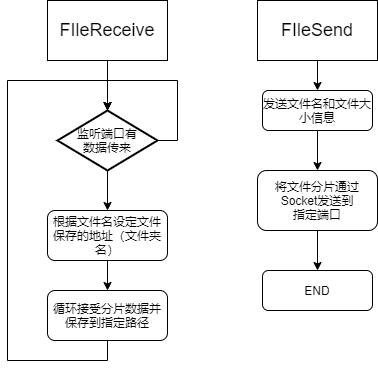
localparam：保存局部参数

testset：保存测试集

**3.通信模块FileControl**

3.1流程设计

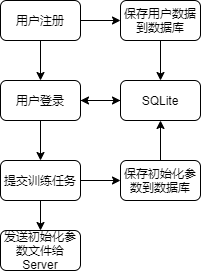
我们设计的文件控制模块主要负责参数文件的发送和接收功能，底层依靠的是Socket协议，恰好Python内嵌有Socket编程相关的API。FileReceive需要单独的一个终端来运行，在循环中持续对某个端口监听。而FileSend是被Client和Server中的main程序调用使用的，只在需要的时候才会执行。

****

**4.网页后端服务器WebServer**

4.1流程设计

根据用户相关、提交任务、数据库存储这三个核心功能，我们设计了以下流程



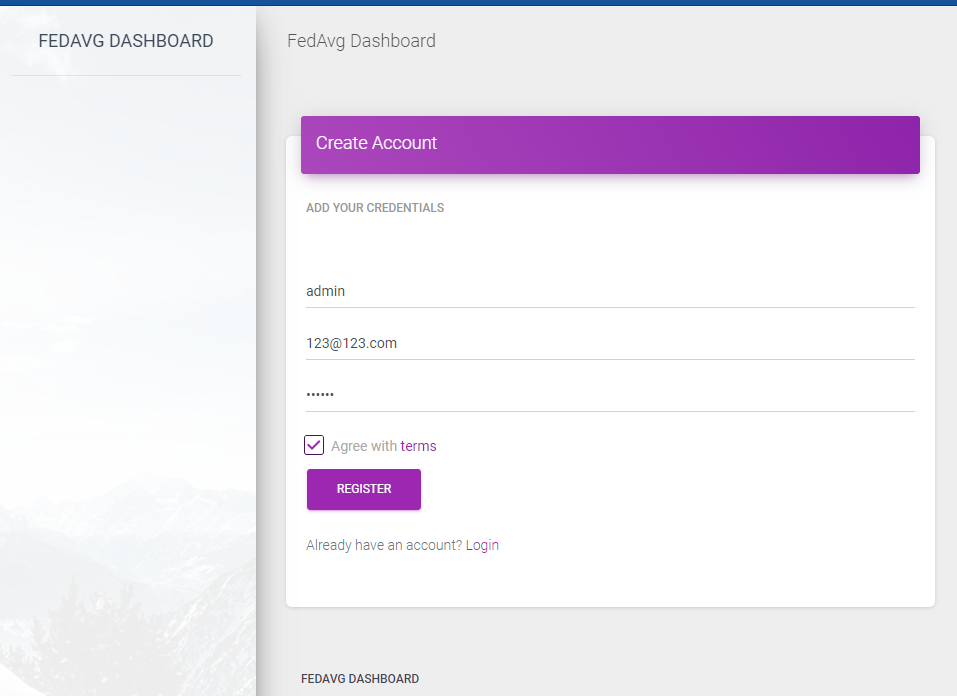
三、结果说明

**1.WebServer功能验证**

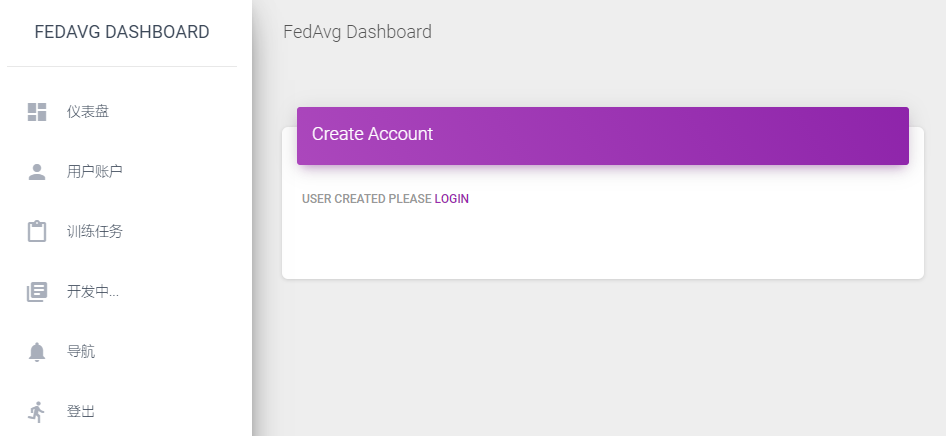
1.1用户注册

1.1.1网页

注册页面

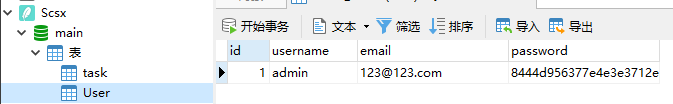


成功注册反馈



1.1.2数据库

密码经过hash加密，保护用户安全



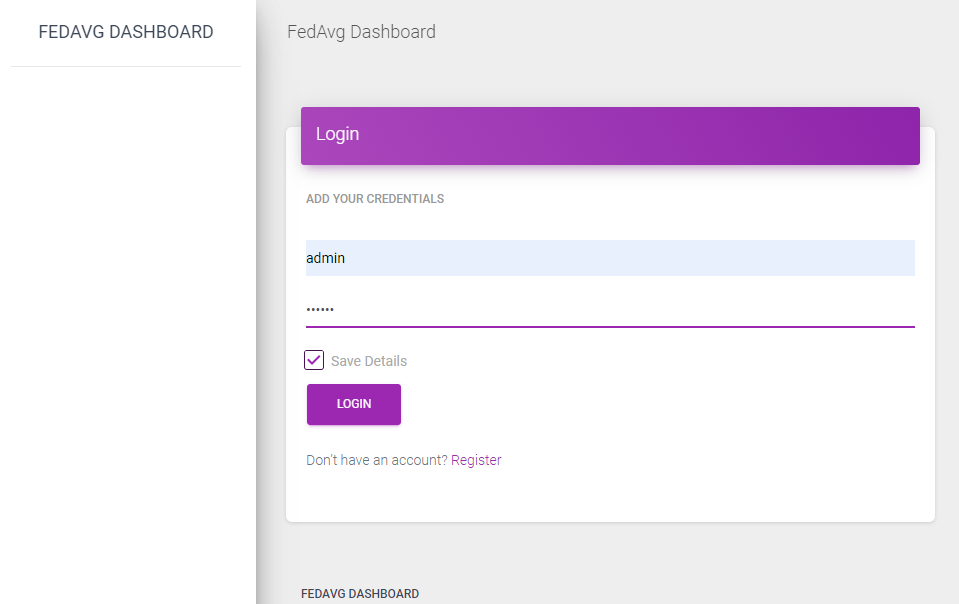
1.1.3服务器日志



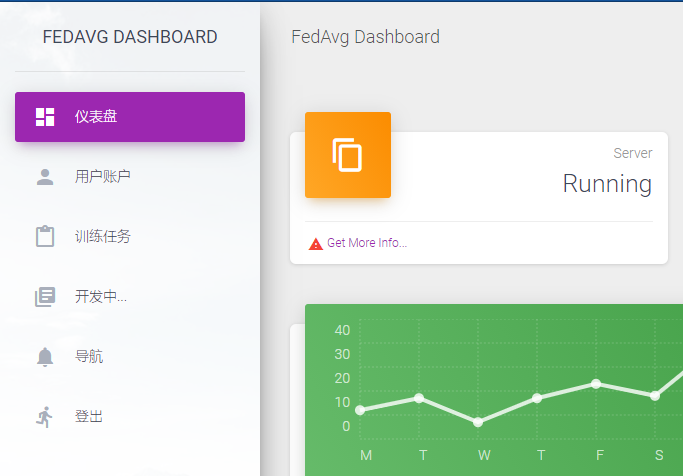
1.2用户登录

1.2.1网页

登录页面



成功登录进入index

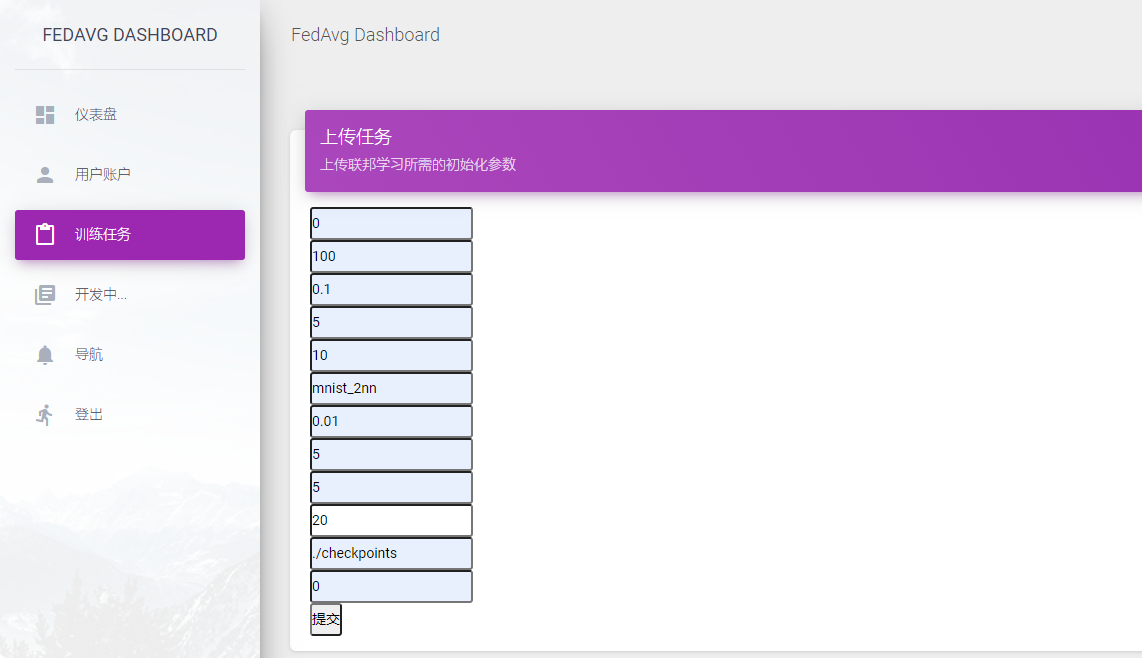


1.2.2服务器日志

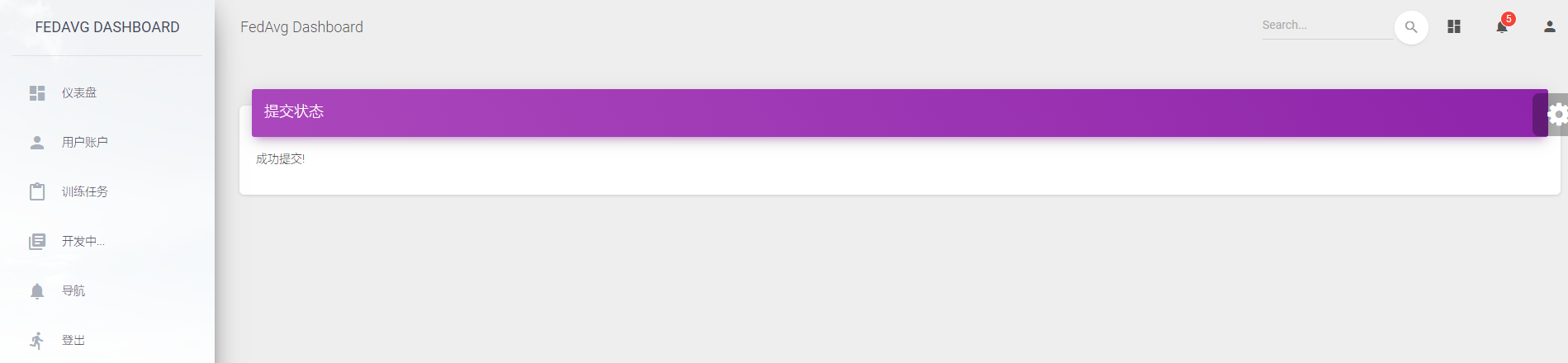


1.3提交训练任务

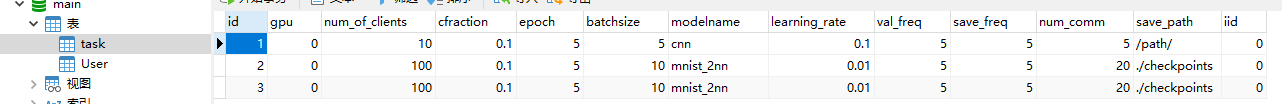
1.3.1提交页面



1.3.2成功提交反馈



1.3.3数据库



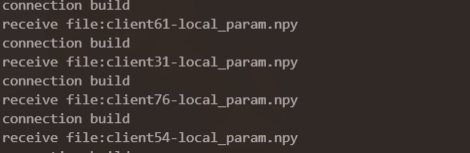
**2.Server功能验证**

2.1接收参数

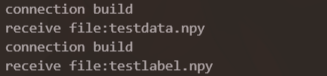
2.1.1初始化参数initparam.json



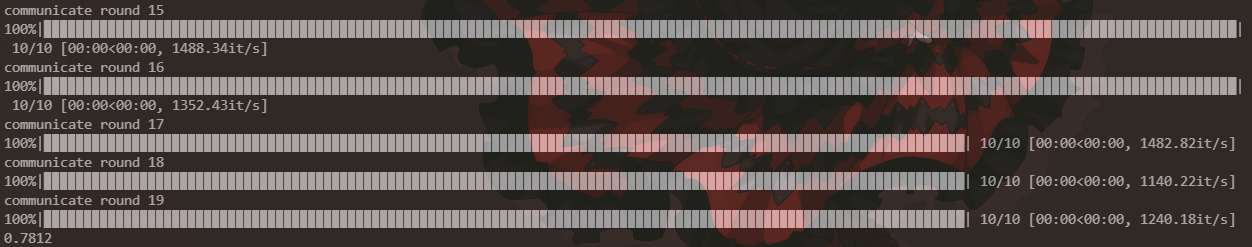
2.2.2局部参数localparam.npy



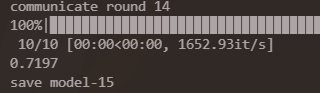
2.2.3 测试集testdata.npy、testlabel.npy

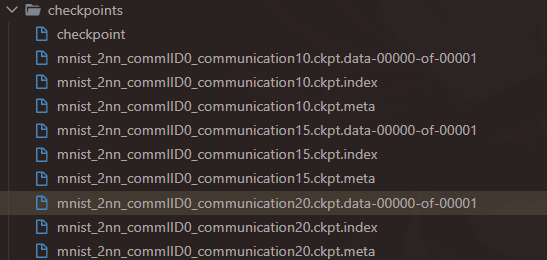


2.3聚合计算并基于测试集完成测试



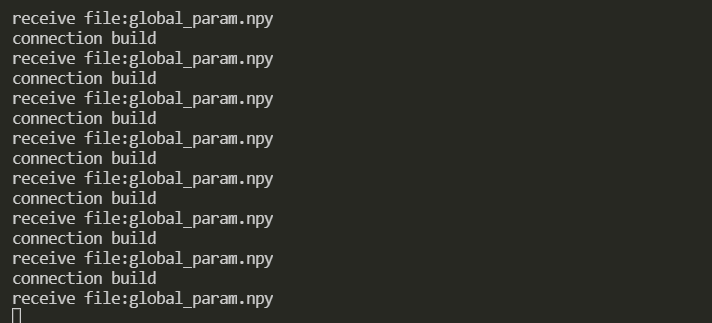
2.4 保存训练模型ckpt



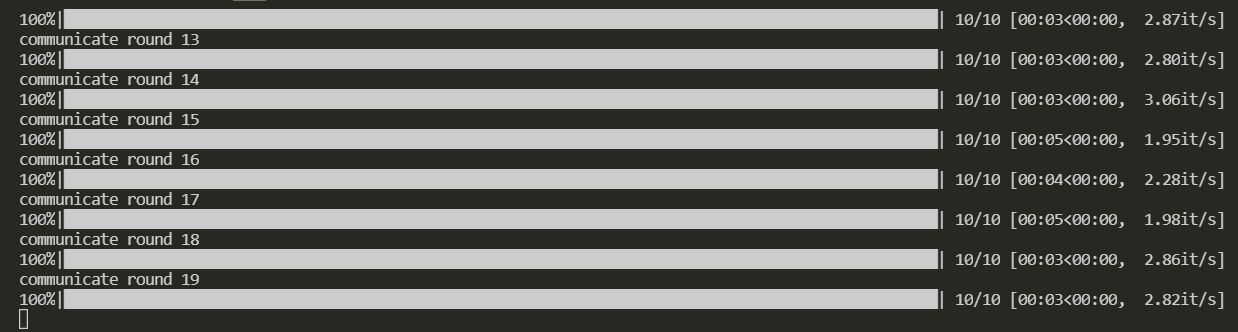


**3.Client功能验证**

3.1接收参数

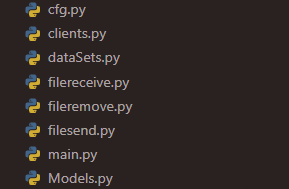


3.2完成本地训练



四、代码说明

**1.聚合服务器Server**



cfg：配置文件，设置socket的发送地址和端口

main：主程序

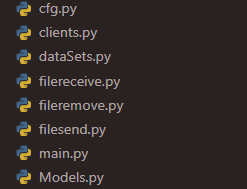
filereceive&filesend：文件发送模块

dataSets；数据集相关

Models：模型定义

其他模块未使用到

**2.本地训练客户端Client**

****

cfg：配置文件，设置socket的发送地址和端口

main：主程序

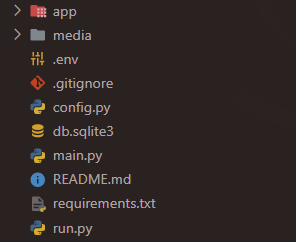
filereceive&filesend：文件发送模块

dataSets；数据集相关

Models：模型定义

client：客户端类的实现

**3.WebServer**



app：前后端代码实现

config：配置文件

db：sqlite数据库文件

main：主程序

run：后端启动入口