**寻TA--Restful网络接口设计**

**一、基本设置**

**1. 配置**

**Npm：**基于nodejs的包管理工具，用于安装项目过程中需要的依赖：

body-parse在处理程序之前分析中间件中的传入请求体，在req.body属性下可用。可以通过body-parser 对象创建中间件，当接收到客户端请求时所有的中间件都会给req.body 添加属性，请求体为空，则解析为空（或者出现错误）。

**Events Node：**大多数核心 API 都采用惯用的异步事件驱动架构，其中某些类

的对象（触发器）会周期性地触发命名事件来调用函数对象（监听器）。

所有能触发事件的对象都是 EventEmitter 类的实例。

**Express：**基于nodejs的web开发框架，主要作为服务器端使用。基本操作流程：引入模块——创建服务器——返回文件——监听端口。

**Mongoose：**用于和mongod服务器交互，以实现数据的增删改查。

**Multer:** 用于处理multipart/form-data,这种MIME类型主要用于上传文件，是基于busboy编写的最大化效率工具

**2. 定义**

let upload = multer({ dest: 'uploads/' })，用于将上传过来的图片用multer中间件进行处理;

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true}));使用bodyParser解析传输过来的json数据，采用URL编码;

const Ima="images/"设置图片存取路径，用于存放用户的图片

mongoose.connect('mongodb://localhost:27017/test',{useNewUrlParser:true,useUnifiedTopology:true});连接本地的mongod服务器以操作数据库的数据

**二、接口规范**

**1. 概要接口设计**

GET部分：通过不同的uri来判断需要返回的数据。主要有：

/up 登录和注册

/upload 用于上传用户的图片

/search 根据用户名来搜索图片的path并返回url

/follow 用于关注好友和取消关注

/friendlist 用于根据用户名来返回好友列表

**2. 详细接口设计**

POST部分：通过get的uri来判断所需要进行的服务：

/up 由请求体中的成员type来判断进行注册还是登录。

登录：根据用户名和密码来查找mongodb数据库，找到了则返回访问图片用的url，找不到则返回“0”;

注册：查询是否已被注册。若已被注册则返回“0”，并在服务器显示注册错误。若未被注册则保存用户名、用户密码、图片路径;

/upload 传照片是单独的一步。通过中间件multer保存图片到本地，再将图片的路径保存到数据库中;

/search 根据用户名在数据库中查找图片路径，此处是为了查找而不是登录所用，因此不需要密码，返回的也是图片的路径;

/follow 根据请求体的type来决定关注动作还是取消关注动作。关注动作则将用户名和相应的好友的用户名存入数据库，取消关注动作则删除与用户名和好友用户名匹配的数据库数据;

/friendlist 根据用户名查询数据库中与该用户名匹配的行，成功则返回由好友用户名组成的字符串列表，失败则返回“0”。