链滴

手把手教你用 Vue 搭建带预览的「上传图片」管理后台

作者：HiJiangChuan

原文链接：https://ld246.com/article/1647827074231

来源网站：链滴

本文首发：《Vue 搭建带预览的「上传图片」管理后台》

手把手教你开发带预览的 Vue 图片上传组件，即图片上传管理后台。只要你跟本教程一步一步走，终能很好的理解整个前后端传图的工程逻辑。前端我们使用 Vue + Axios + Multipart 来搭建前端上图片应用，后端我们使用 Node.js + Express + Multer 来搭建后端上传图片的后端处理。

本教程还会教给大家如何写一个可限制上传图片大小，有进度条，可报错，可显示图片列表，可下载上传图片的图片管理后台。

最后完成的上传图片工具后台如下图，跟随本教学习，你也可以搭出来。

如果你对前端不是很熟悉，不想写前端，推荐使用卡拉云搭建后台管理系统，卡拉云是新一代低代码发工具，不用懂任何前端技术，仅靠鼠标拖拽，即可快速搭建包括「图片上传」在内的任何后台管理具。立即试用卡拉云 1 分钟搭建「图片上传」工具。详情见本文文末。

Vue + Node.js「图片上传」前后端项目结构

Vue 前端部分

- UploadFilesService.js：这个脚本调用通过 Axios 保存文件和获取文件的方法
- UploadFiles.vue：这个组件包含所有图片上传相关的信息和操作
- App.vue：把我们的组件导入到 Vue 起始页
- index.html：用于导入 Bootstrap
- http-common.js：配置并初始化 Axios
- vue.config.js：配置 APP 端口

Node.js 后端部分

- resources/static/assets/uploads：用于存储上传的文件
- middleware/upload.js：初始化 Multer 引擎并定义中间件
- file.controller.js：配置 Rest API
- routes/index.js：路由，定义前端请求后端如何执行
- server.js：运行Node.js Express 应用

前端部分 - Vue + Vue 图片上传组件 + Axios + Multipart

配置 Vue 环境

使用 npm 安装 Vue 脚手架 vue-cli

```
npm install -g @vue/cli
```

然后我们创建一个 Vue 项目 kalacloud-vue-upload-image

```
vue create kalacloud-vue-upload-image
```

安装完成后，cd 进入 kalacloud-vue-upload-image 目录，接下来所有操作都在这个目录之下。

安装 Axios：

```
npm install axios
```

我们先跑一下 Vue ， 这是 vue 的默认状态

```
npm run serve
```

我们可以看到浏览器里 Vue 已经在 localhost:8080 跑起来了。

扩展阅读：《Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器》

导入 Bootstrap 到项目中

打开 index.html 把以下代码添加到<head> 中：

文件位置：public/index.html

```
<!DOCTYPE html>
<html lang="en"><head>
    ...
    <link type="text/css" rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap
4.4.1/css/bootstrap.min.css" /></head>
  ...
</html>
```

初始化 Axios HTTP 客户端

在 src 文件夹下，创建 http-common.js 文件，如下所示：

文件位置：src/http-common.js

```
import axios from "axios";
export default axios.create({
  baseURL: "http://localhost:8080",
  headers: {
    "Content-type": "application/json"
  }
```

```
});
```

请记住更改baseURL，它取决于您的服务器配置的 REST API 网址。

扩展阅读：《12 款最棒 Vue 开源 UI 库测评 - 特别针对国内使用场景推荐》

创建「上传图片」功能

我们来写一个 JS 脚本，这个脚本调用 Axios 发送 HTTP API 请求，与后端服务器通讯。

这个脚本包含 2 个功能

- upload(file): POST 数据到后端，再加一个上传进度的回调，可以放个上传进度条。
- getFiles(): 用于获取服务器图片上传夹中的文件列表

文件位置：src/services/UploadFilesService.js

```
import http from "../http-common";
class UploadFilesService {
  upload(file, onUploadProgress) {
    let formData = new FormData();
    formData.append("file", file);
    return http.post("/upload", formData, {
      headers: {
        "Content-Type": "multipart/form-data"
      },
      onUploadProgress
    });
  }
  getFiles() {
    return http.get("/files");
  }
}
export default new UploadFilesService();
```

- 首先导入我们刚刚写好的 Axios HTTP 配置文件 http-common.js
- FormData 是一种可将数据编译成键值对的数据结构
- Axios的进度条事件， onUploadProgress 是用来监测上传进度，显示进度信息
- 最后我们调用 Axios 提供的 post()&get() 来向后端 API 发送 POST & GET 请求

扩展阅读：《顶级好用的 8 款 Vue 弹窗组件测评与推荐》

创建一个 Vue 图片上传组件

让我们创建一个带有进度条、卡片、按钮和消息的图像上传 UI。

首先我们创建一个 Vue 组件模板并导入UploadFilesService：

文件位置：src/components/UploadFiles.vue

接下来，我们来写一个 Vue 图片上传组件，这个组件要包含上传图片的所有基本功能，比如 上传按

、进度条、图片预览、提示信息、基本 UI 等。

首先，创建一个 Vue 组件模版（UploadFiles.vue）然后把刚刚写好的配置文件（UploadFilesService js）导入进去。

文件位置：src/components/UploadFiles.vue

```
<template>
</template>
<script>
import UploadService from "../services/UploadFilesService";
export default {
  name: "upload-image",
  data() {
    return {
    };
  },
  methods: {

  }
};
</script>
```

然后在这里定义 data() 变量

```
// src/components/UploadFiles.vue

export default {
  name: "upload-files",
  data() {
    return {
      selectedFiles: undefined,
      progressInfos: [],
      message: "",
      fileInfos: [],
    };
  },
};
```

接下来，我们定义 methods 让 selectImage() 从 ref="file" 中获取选定的文件。

```
// src/components/UploadFiles.vue

export default {
  name: "upload-image",
  ...
  methods: {
    selectImage() {
      this.currentImage = this.$refs.file.files.item(0);
      this.previewImage = URL.createObjectURL(this.currentImage);
      this.progress = 0;
      this.message = "";
    },
  }
};
```

我们读取 currentImage 作为 image 的输入参数。用 URL.createObjectURL() 静态方法来获取图像览的 URL 存入 previewImage，方法会创建一个 [DOMString](https://developer.mozilla.org/zh-N/docs/Web/API/DOMString) ，其中包含一个表示参数中提供的对象的URL。这个 URL 的生命周和创建它的窗口中的 [document](https://developer.mozilla.org/zh-CN/docs/Web/API/Documen) 绑定。

最后，我们还定义了 upload() 上传图片的方法，如下所示：

```
// src/components/UploadFiles.vue

export default {
  name: "upload-image",
  ...
  methods: {
    ...
    upload() {
      this.progress = 0;
      UploadService.upload(this.currentImage, (event) => {
        this.progress = Math.round((100 * event.loaded) / event.total);
      })
        .then((response) => {
          this.message = response.data.message;
          return UploadService.getFiles();
        })
        .then((images) => {
          this.imageInfos = images.data;
        })
        .catch((err) => {
          this.progress = 0;
          this.message = "Could not upload the image! " + err;
          this.currentImage = undefined;
        });
    },
  }
};
```

图片的上传进度根据 event.loaded 和 event.total 来计算。如果图片上传成功，我们调用 UploadSer ice.getFiles() 来获取图片信息并把结果分赋值给 imageInfos

imageInfos 是一个 {name, url} 对象数组。接着我们把它加到 mounted() 中，让它可以执行。

```
export default {
  name: "upload-image",
  ...
  mounted() {
    UploadService.getFiles().then(response => {
      this.imageInfos= response.data;
    });
  }
};
```

现在我们实现上传图片 UI 的 HTML 模板。将以下内容添加到<template>：

接下来，我们来写上传图片的 UI HTML 模版。请将以下代码添加到 <template> 之间。

```
// src/components/UploadFiles.vue

<template>
  <div>
    <div class="row">
      <div class="col-8">
        <label class="btn btn-default p-0">
          <input
            type="file"
            accept="image/*"
            ref="file"
            @change="selectImage"
          />
        </label>
      </div>
      <div class="col-4">
        <button
          class="btn btn-success btn-sm float-right"
          :disabled="!currentImage"
          @click="upload"
        >
          上传
        </button>
      </div>
    </div>
    <div v-if="currentImage" class="progress">
      <div
        class="progress-bar progress-bar-info"
        role="progressbar"
        :aria-valuenow="progress"
        aria-valuemin="0"
        aria-valuemax="100"
        :style="{ width: progress + '%' }"
      >
        {{ progress }}%
      </div>
    </div>
    <div v-if="previewImage">
      <div>
        <img class="preview my-3" :src="previewImage" alt="" />
      </div>
    </div>
    <div v-if="message" class="alert alert-secondary" role="alert">
      {{ message }}
    </div>
    <div class="card mt-3">
      <div class="card-header">图片列表</div>
      <ul class="list-group list-group-flush">
        <li
          class="list-group-item"
          v-for="(image, index) in imageInfos"
          :key="index"
        >
          <a :href="image.url">{{ image.name }}</a>
```

```
      </li>
    </ul>
   </div>
  </div>
</template>
```

在上面的代码中，我们使用 Bootstrap 进度条，这里不展开讲了，更多细节可查看 Bootstrap 文档。

在 App.vue 中添加「上传图片」组件

打开 App.vue ,在代码中导入 UploadFiles 组件。

```
<template>
  <div id="app">
    <div class="container" style="width:600px">
      <div class="my-4">
        <h2>使用 Vue 搭建图片上传组件 Demo</h2>
        <h5><a href="http://kalacloud.com">卡拉云</a> - 低代码开发工具，1 秒搭建上传图片组
工具</h5>
        <a>使用卡拉云无需懂任何前端技术，仅需拖拽即可搭建属于你的后台管理系统</a>

      </div>
      <upload-image></upload-image>
    </div>
  </div>
</template>
<script>
import UploadImage from "./components/UploadImage";
export default {
  name: "App",
  components: {
    UploadImage
  }
};
</script>
```

扩展阅读：《ECharts X 轴标签过长导致文字重叠的 4 种解决方案》

给 Vue 图片上传服务配置访问端口

文件位置：src/vue.config.js

```
module.exports = {
  devServer: {
    port: 8081
  }
}
```

运行 Vue 前端部分

到这里，我们已经把 Vue 图片上传的前端部分写完，运行起来看看效果吧。

在 kalacloud-vue-multiple-files-upload 根目录跑一下：

```
npm run serve
```

在浏览器打开 http://localhost:8081/ 可以看到前端部分已经完美显示。

图片选择器、上传按钮、图片列表都已经可以显示出来了，但还无法上传。这是因为后端部分还没有起来，接下来，我带领大家手把手搭建图片上传的后端部分。

Vue 前端「图片上传」源码

你可以在我的 github 上下载到完整的 Vue 图片上传 Demo。

当然你也可以不用这么费劲搭建前端做图片上传功能，直接使用卡拉云，无需懂前后端，简单拖拽即生成一套属于你自己的后台管理工具。

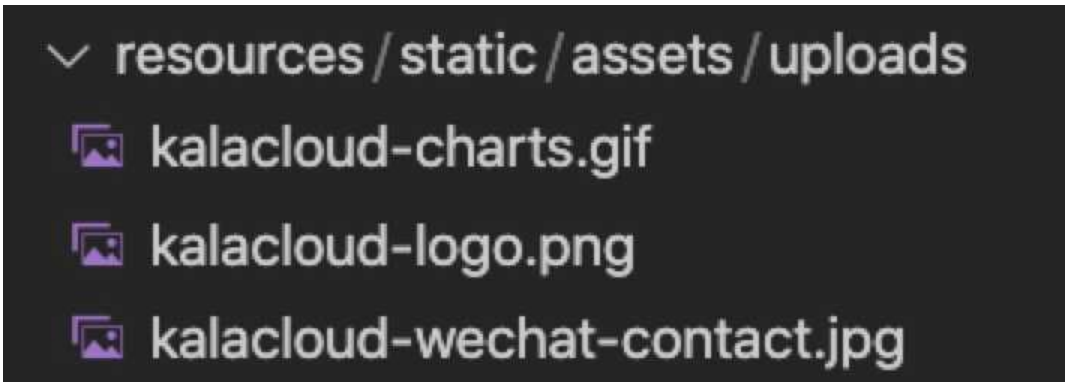
□ 后端部分 - 图片上传 Node.js + Express + Multer

前文我们介绍了如何使用 Vue 搭建图片上传管理工具的前端部分，接下来我教大家使用 Node.js + Express + Multer 来搭建一套图片上传的后端 Rest API，提供给 Vue 前端使用，从而实现 Vue 选择件 + Axios 调用后端 API HTTP 通讯，最后把图片上传到服务器指定目录。

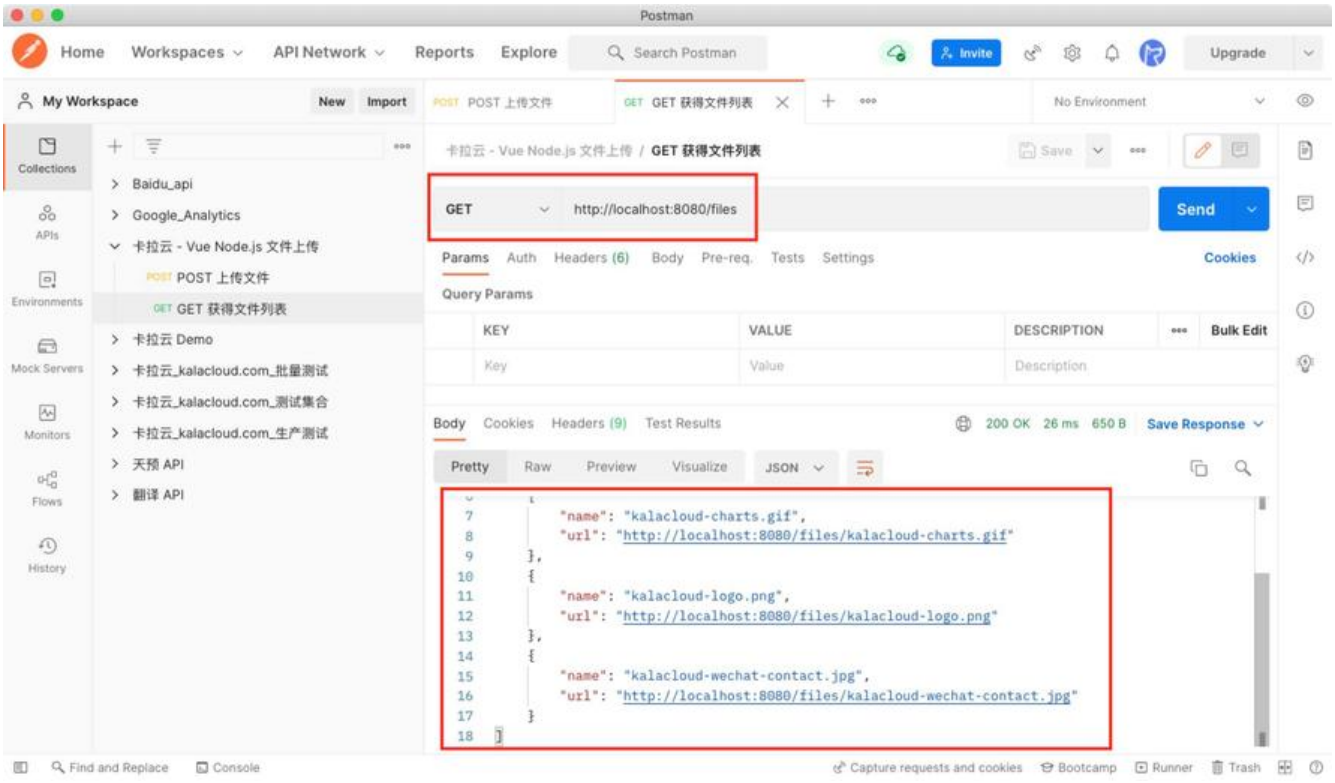
接下来，大家一起跟随本教程创建一套 Node.js 图片上传 Rest API，它的功能包括：

- 将 Vue 前端选中的图片上传到服务器的静态文件夹中
- 限制图片上传大小，最大 2MB
- GET 服务器中存储文件的 URL ，可用于下载
- GET 文件信息列表（文件名 + URL）

这是存储所有图片上传的静态文件夹：



如果我们 GET 文件列表，Node.js Rest API 会返回：



GET /files ，API 返回 文件名 + URL

我们构建的 Node.js Rest API 包含这三个功能：

- POST /upload 上传一个文件
- GET /files 获取文件列表（文件名+URL）
- GET /files/[filename] 下载指定文件

扩展阅读：《如何使 Vue ECharts 柱状图中，每个柱子颜色各不同（随机或指定颜色）》

配置 Node.js 开发环境

在根目录新建 Node.js 的后端文件夹 kalacloud-express-file-upload，接下来所有操作都在这个文夹中进行。

```
mkdir kalacloud-express-file-upload
cd kalacloud-express-file-upload
```

在 kalacloud-express-file-upload 文件夹根目录安装 Express、Multer、CORS 这三个模块：

```
npm install express multer cors
```

package.json 文件：

```
{
  "name": "kalacloud-express-file-upload",
  "version": "1.0.0",
  "description": "Node.js Express Rest APIs to Upload/Download Files",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "node js",
    "upload",
    "download",
    "file",
    "multipart",
    "rest api",
    "express"
  ],
  "author": "bezkoder",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "multer": "^1.4.2"
  }
}
```

扩展阅读：《7 种最棒的 Vue Loading 加载动画组件测评与推荐 - 穷尽市面上所有加载动画效果（Ve loader）类型》

配置图片上传中间件 Multer

我们使用 Multer 中间件来处理多图片上传，更多 Multer 细节请阅读它的开发文档

文件位置：src/middleware/upload.js

```
const util = require("util");
const multer = require("multer");
const maxSize = 2 * 1024 * 1024;
let storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, __basedir + "/resources/static/assets/uploads/");
  },
  filename: (req, file, cb) => {
    console.log(file.originalname);
    cb(null, file.originalname);
  },
});
let uploadFile = multer({
```

```
  storage: storage,
  limits: { fileSize: maxSize },
}).single("file");
let uploadFileMiddleware = util.promisify(uploadFile);
module.exports = uploadFileMiddleware;
```

上面的代码我们做了这么几件事：

- 导入 multer 模块。
- 配置 multer 为磁盘存储引擎。
- destination：指向用于存储图片上传的文件夹。
- filename：图片上传上传后的文件名。

使用 Multer 限制文件大小

我们可以使用 Multer API 来限制图片上传大小，添加 limits: { fileSize: maxSize } 以限制文件大小。

```
let storage = multer.diskStorage(...);
const maxSize = 2 * 1024 * 1024;
let uploadFile = multer({
  storage: storage,
  limits: { fileSize: maxSize }
}).single("file");
```

扩展阅读：《Vue ECharts 饼状图颜色设置教程 - 4 种方式设置饼图颜色》

创建图片上传 / 下载控制器

在 controller 文件夹中创建 file.controller.js

图片上传：我们使用 upload() 函数

- 使用中间件功能图片上传
- 图片上传错误信息（在 Multer 中间件函数中）
- 返回信息

下载文件：

- 使用 getListFiles() 读取服务器图片上传夹中的所有文件，包含文件名和 URL
- 使用 download() 接收文件名作为输入参数，然后使用 Express res.downloa() 以附件形式传输 RL（目录+文件名）

文件位置：src/controller/file.controller.js

```
const uploadFile = require("../middleware/upload");
const upload = async (req, res) => {
  try {
    await uploadFile(req, res);
    if (req.file == undefined) {
      return res.status(400).send({ message: "请选择要上传的文件" });
```

```
    }
    res.status(200).send({
      message: "图片上传成功: " + req.file.originalname,
    });
  } catch (err) {
    res.status(500).send({
      message: `无法图片上传: ${req.file.originalname}. ${err}`,
    });
  }
};
const getListFiles = (req, res) => {
  const directoryPath = __basedir + "/resources/static/assets/uploads/";
  fs.readdir(directoryPath, function (err, files) {
    if (err) {
      res.status(500).send({
        message: "没有找到文件。",
      });
    }
    let fileInfos = [];
    files.forEach((file) => {
      fileInfos.push({
        name: file,
        url: baseUrl + file,
      });
    });
    res.status(200).send(fileInfos);
  });
};
const download = (req, res) => {
  const fileName = req.params.name;
  const directoryPath = __basedir + "/resources/static/assets/uploads/";
  res.download(directoryPath + fileName, fileName, (err) => {
    if (err) {
      res.status(500).send({
        message: "无法获取文件。" + err,
      });
    }
  });
};
module.exports = {
  upload,
  getListFiles,
  download,
};
```

- 我们首先调用中间件函数 uploadFile()
- 如果 HTTP 请求不包含文件，返回 400 错误信息
- 如果出现获取错误，返回 500 错误信息

如果用户图片上传大小超限的文件应该怎么处理?

扩展阅读：《vue.draggable 入门指南 - 手把手教你开发任务看板》

使用Multer 处理文件大小超限错误

我们可以通过 catch() 来检查文件超限错误（LIMIT_FILE_SIZE）

文件位置：src/controller/file.controller.js

```
 const upload = async (req, res) => {
  try {
    await uploadFile(req, res);
    ...
  } catch (err) {
    if (err.code == "LIMIT_FILE_SIZE") {
      return res.status(500).send({
        message: "文件大小不能超过 2MB",
      });
    }
    res.status(500).send({
      message: `不能图片上传: ${req.file.originalname}. ${err}`,
    });
  }
};
```

扩展阅读：《最好用的 6 款 Vue 拖拽组件库推荐》

设置后端 Rest API 图片上传的路径

当 Vue 前端通过 Axios 发送 HTTP 请求时，我们需要通过路由来确定服务器应该如何响应

我们来设置三种常用到的图片上传所需功能

- POST　/upload：upload()
- GET　/files：getListFiles()
- GET　/files/[fileName]：download()

我们在 routes 文件夹中创建 index.js 文件：

文件位置：src/routes/index.js

```
const express = require("express");
const router = express.Router();
const controller = require("../controller/file.controller");
let routes = (app) => {
  router.post("/upload", controller.upload);
  router.get("/files", controller.getListFiles);
  router.get("/files/:name", controller.download);
  app.use(router);
};
module.exports = routes;
```

上面这段代码调用了我们前文写的控制器 file.controller.js

扩展阅读：《Vue 实现 PDF 文件在线预览 - 手把手教你写 Vue PDF 预览功能》

创建 Express 服务

最后一步，创建 Express 服务，在根目录新建一个 server.js 文件

文件位置：kalacloud-express-file-upload/server.js

```
const cors = require("cors");
const express = require("express");
const app = express();
global.__basedir = __dirname;
var corsOptions = {
  origin: "http://localhost:8081"
};
app.use(cors(corsOptions));
const initRoutes = require("./src/routes");
app.use(express.urlencoded({ extended: true }));
initRoutes(app);
let port = 8080;
app.listen(port, () => {
  console.log(`Running at localhost:${port}`);
});
```

导入 express 和 cors 模块：

- 创建 Express 应用，用于构建 Rest API，然后添加 cors中间件。
- 设置 http://localhost:8081 为 origin，这里允许前端传入

扩展阅读：《最好用的 12 款 Vue Timepicker 时间日期选择器测评推荐》

运行后端并测试

首先，在 kalacloud-express-file-upload 根目录执行

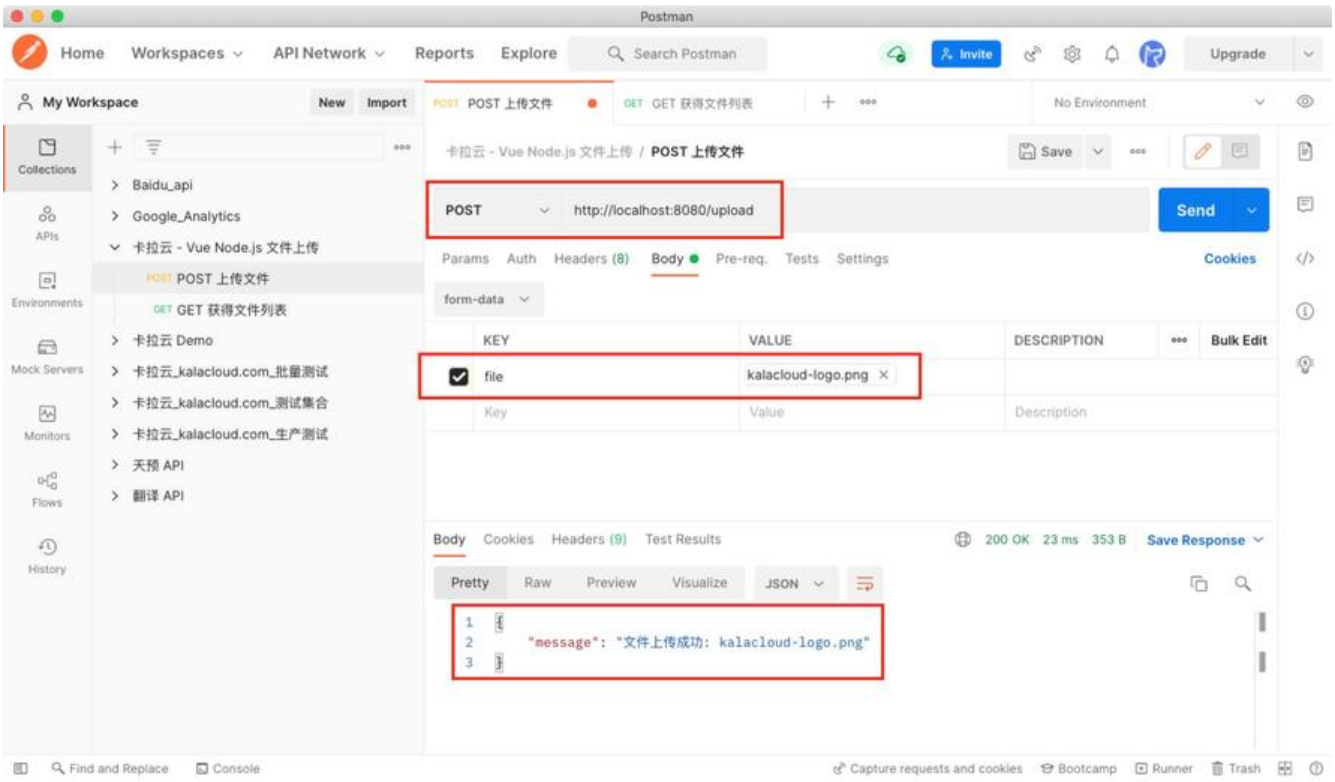
```
node server.js
```

把后端服务启动起来。然后我们使用 Postman 来发送 HTTP 请求，看看后端是否运行正常。

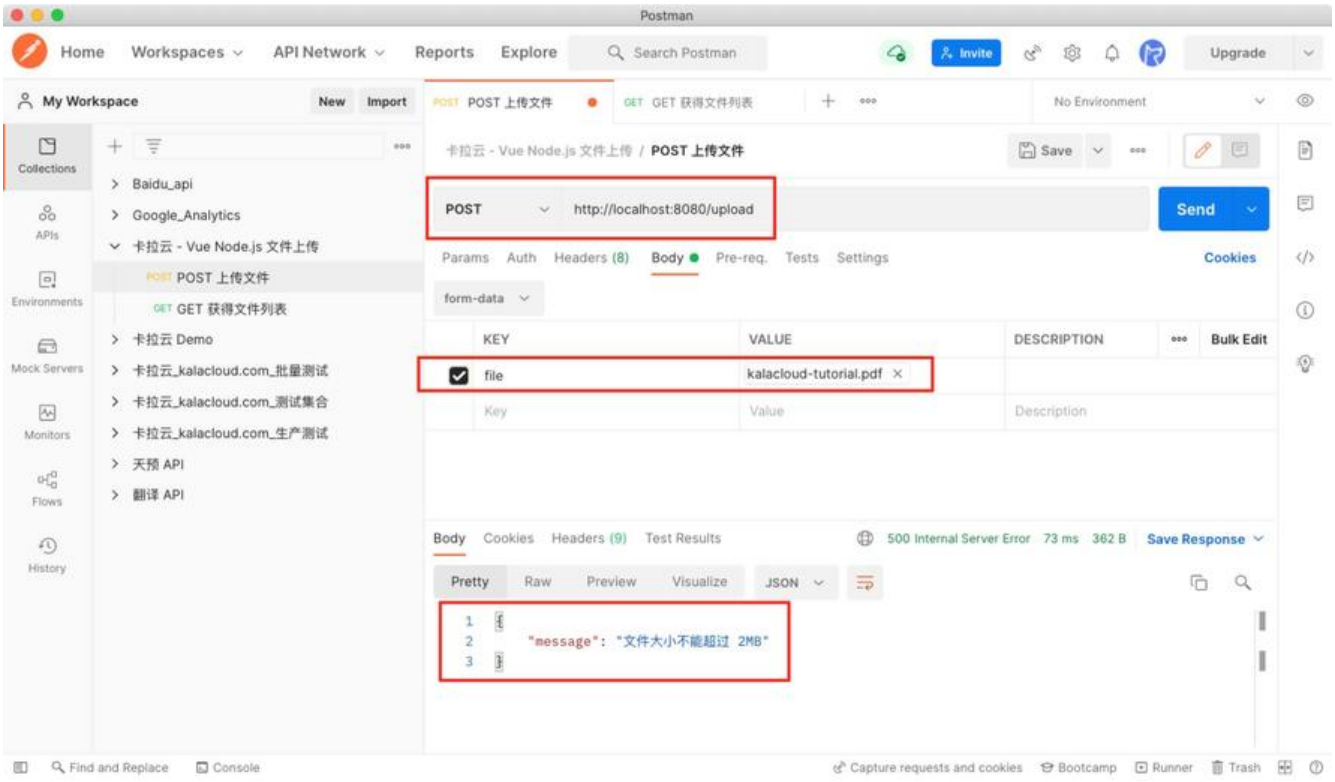
```
 kalacloud-express-file-upload node server.js
Running at localhost:8080
```

接着我们使用 Postman 来测试一下，我们刚刚搭建的后端服务器是否能正常运行

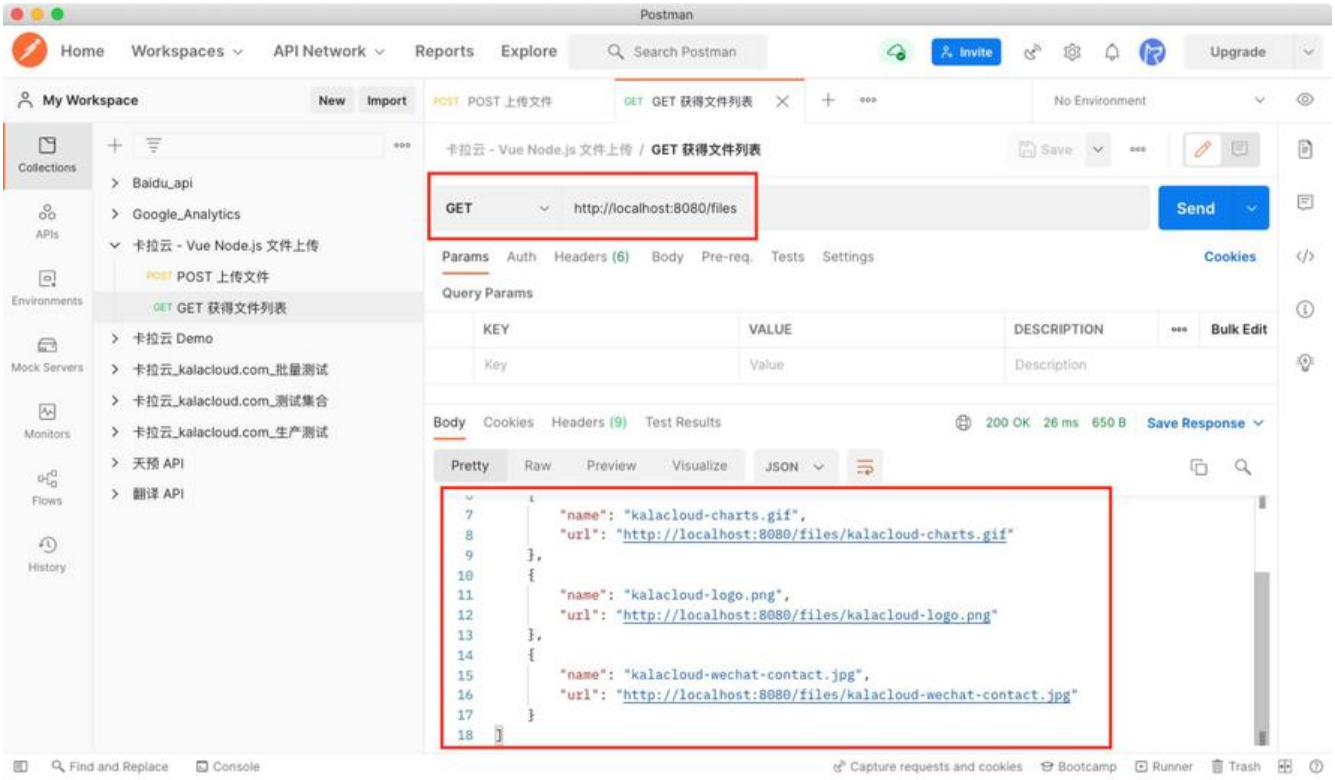
向后端服务器发 POST 请求图片上传



上传大于最大限制 (2MB) 的文件，500 报错。



GET 检索文件信息列表：



我们可以使用返回的文件 URL 下载这些文件，例如： http://localhost:8080/files/kalacloud-logo.png

扩展阅读：《最好用的 7 个 Vue Tree select 树形组件》

Vue + Node.js 图片上传前后端一起运行

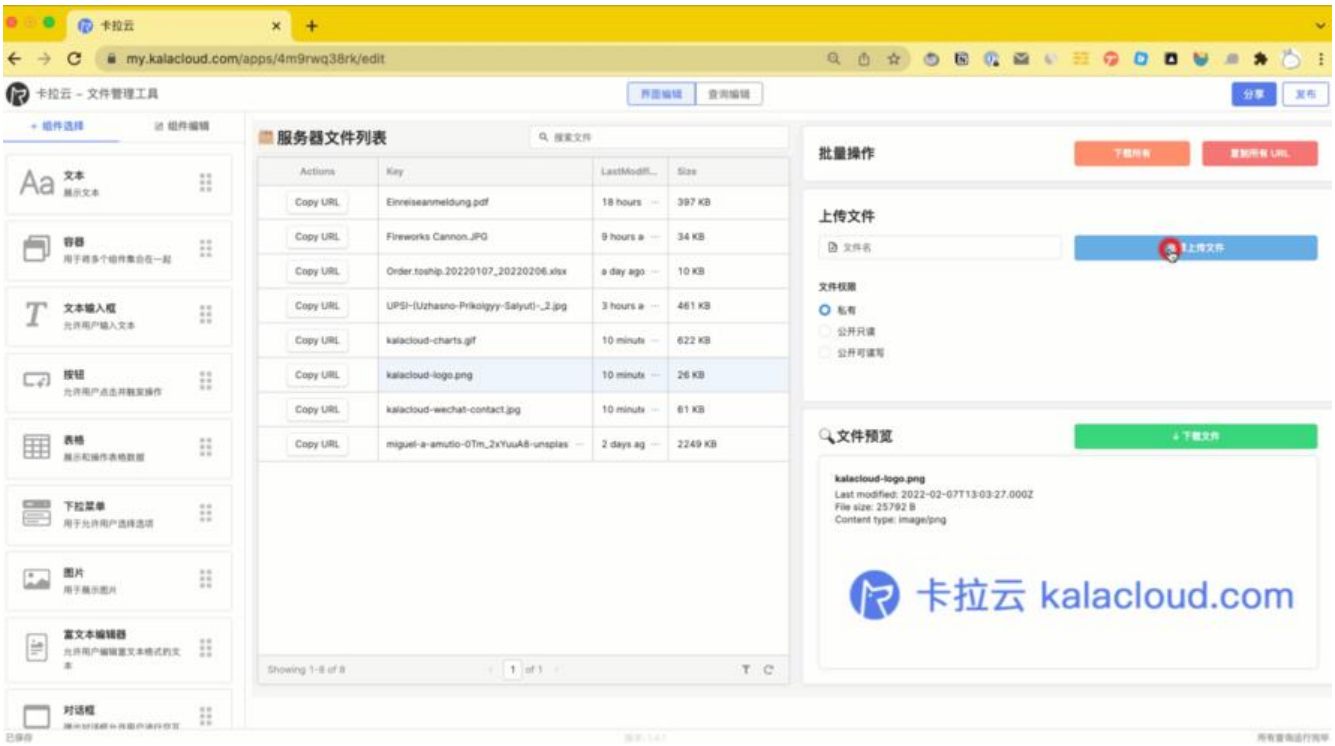
在 kalacloud-vue-multiple-files-upload 文件夹根目录运行前端 Vue

```
npm run serve
```

在 kalacloud-express-file-upload 文件夹根目录运行后端 Node.js

```
node server.js
```

然后打开浏览器输入前端访问网址：



到这里整个前后端「上传图片」管理工具就搭建完成了。

扩展阅读：《Vue Router 手把手教你搭 Vue3 路由》

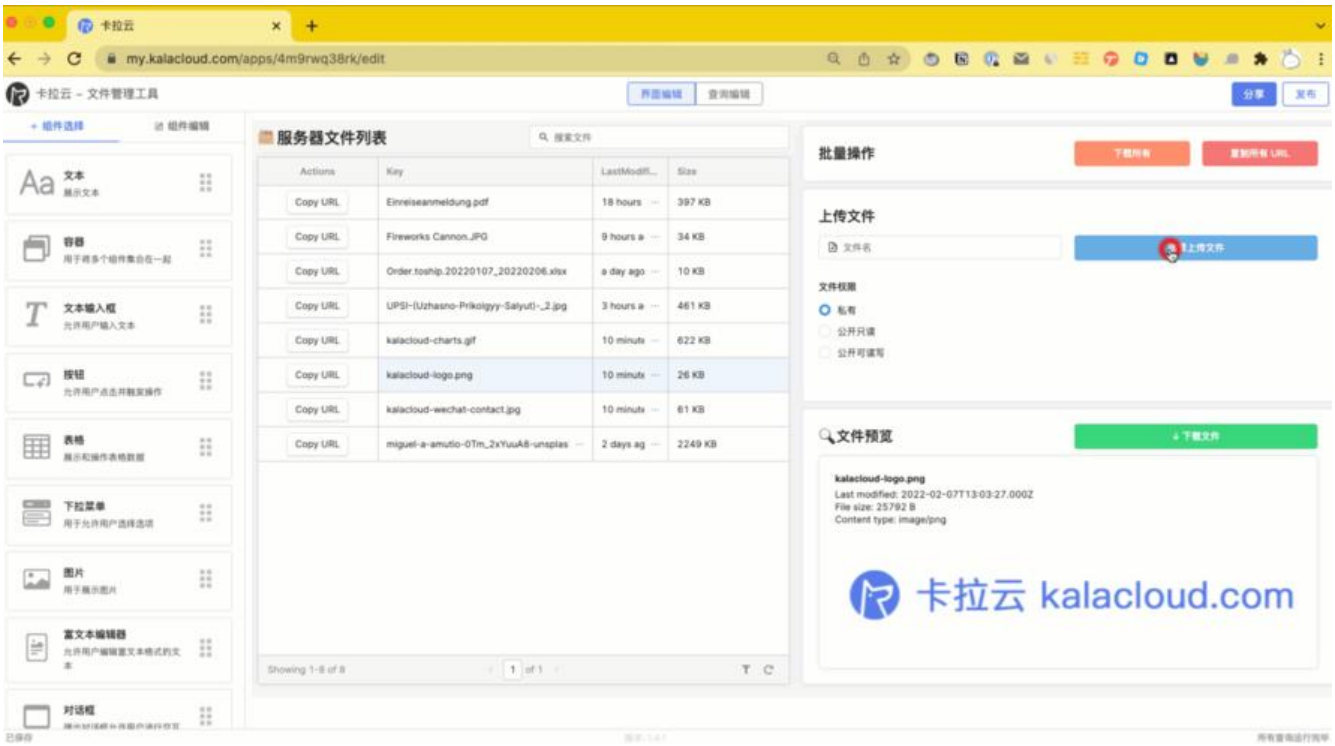
Node.js 后端「上传图片」源码

你可以在我的 github 上下载到完整的 Node.js 后端「图片上传」源码。

如果你还没搞懂，也不用着急，直接使用卡拉云，无需懂任何前后端技术，仅需简单的鼠标拖拽即可速生成包括「图片上传」管理在内的任何后台管理工具。立即试用卡拉云。

「上传图片」前后端搭建总结

本教程手把手教大家搭建 Vue 前端 + Node.js 后端 的「上传图片」管理工具，如果你一步步跟着走一定已经把 Demo 跑起来了。但如果你会使用最新的低代码开发工具「卡拉云」，完全不需要这么琐，仅需 1 分钟，就能搭建起属于自己的「上传图片」管理工具。



注册开通卡拉云，从侧边工具栏直接拖拽组件到页面，生成上传组件和文件管理工具。1 分钟搞定「片上传」管理工具。

扩展阅读：

- 最好用的 7 款 Vue admin 后台管理框架测评
- 顶级好用的 5 款 Vue table 表格组件测评与推荐
- 最好用的 5 款 React 富文本编辑器
- Axios 教程：Vue + Axios 安装及实战 - 手把手教你搭建加密币实时价格看板
- 最好的 6 个免费天气 API 接口对比测评
- 如何在 Vue 中导出数据至 Excel 表格