

说明文档

姓名 徐啸

学号 20164999

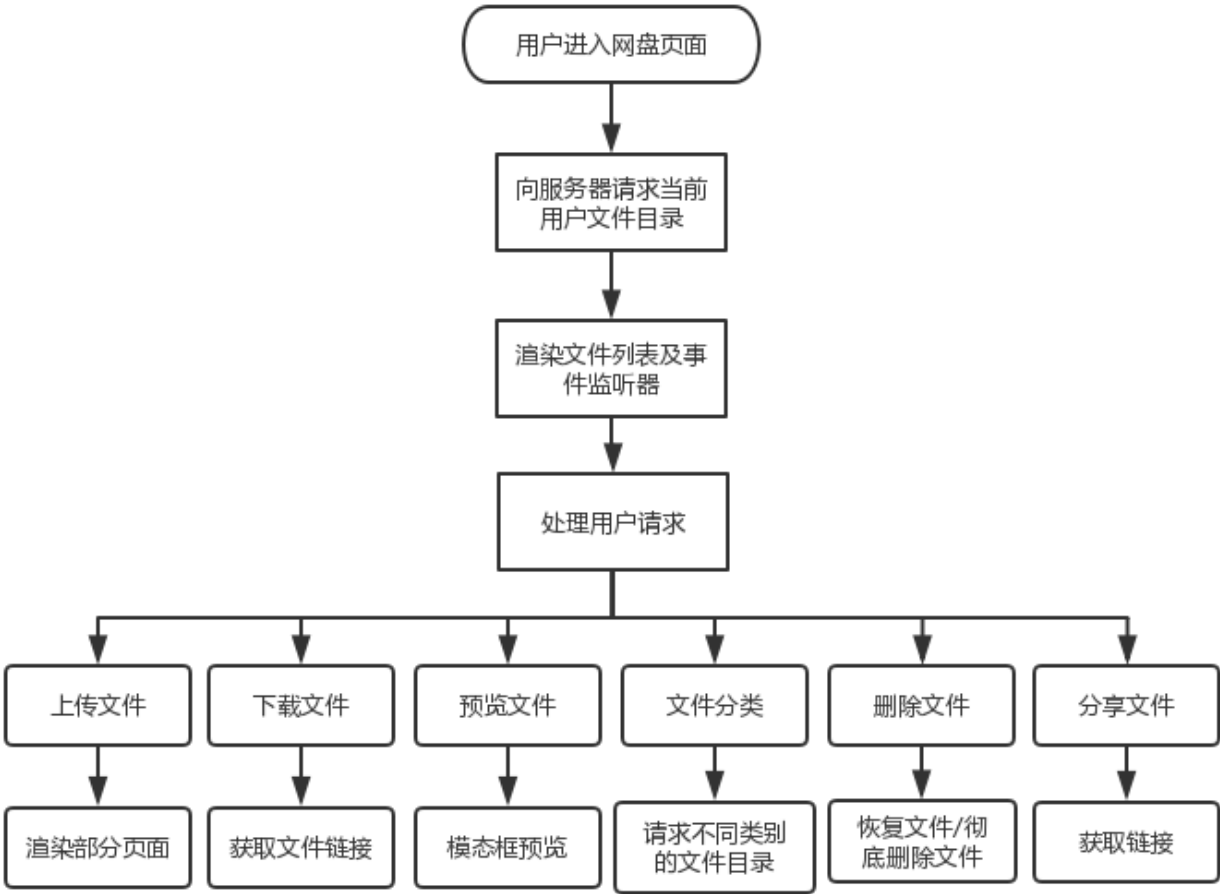
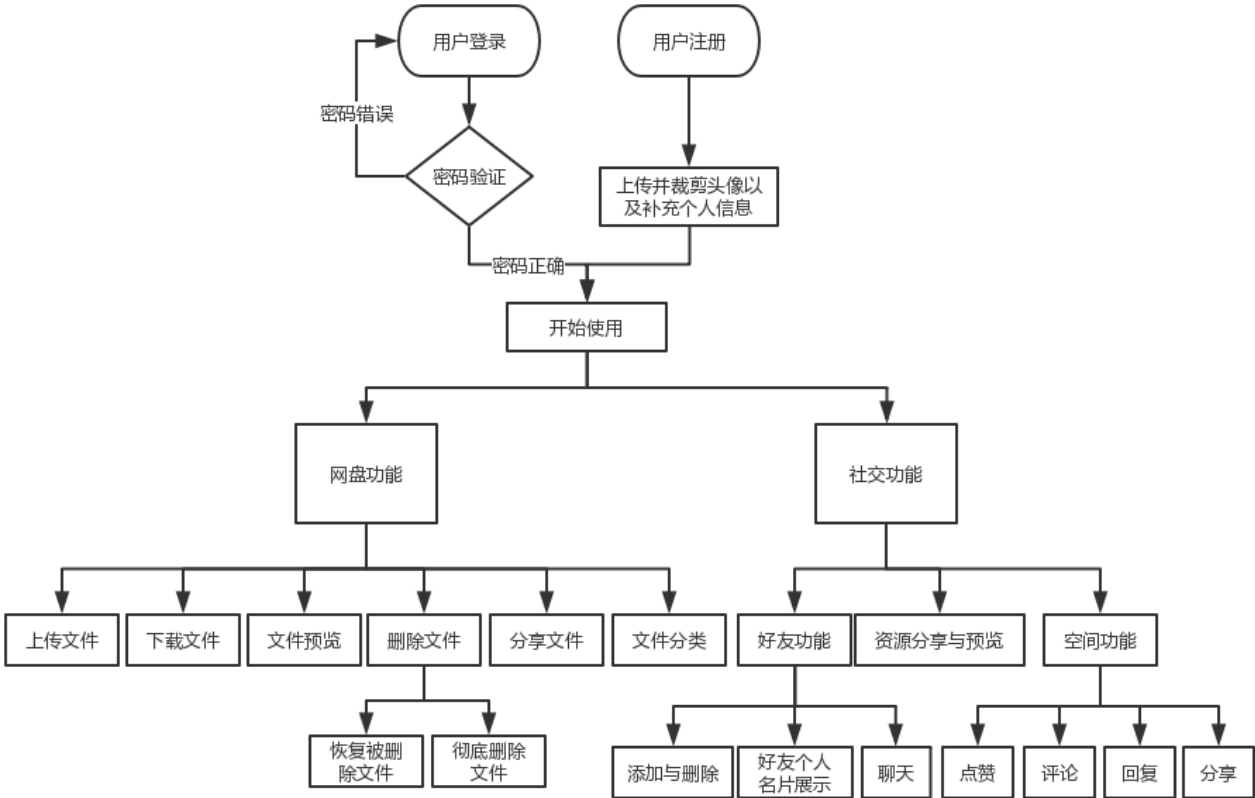
班级 软工1604

MyCloud

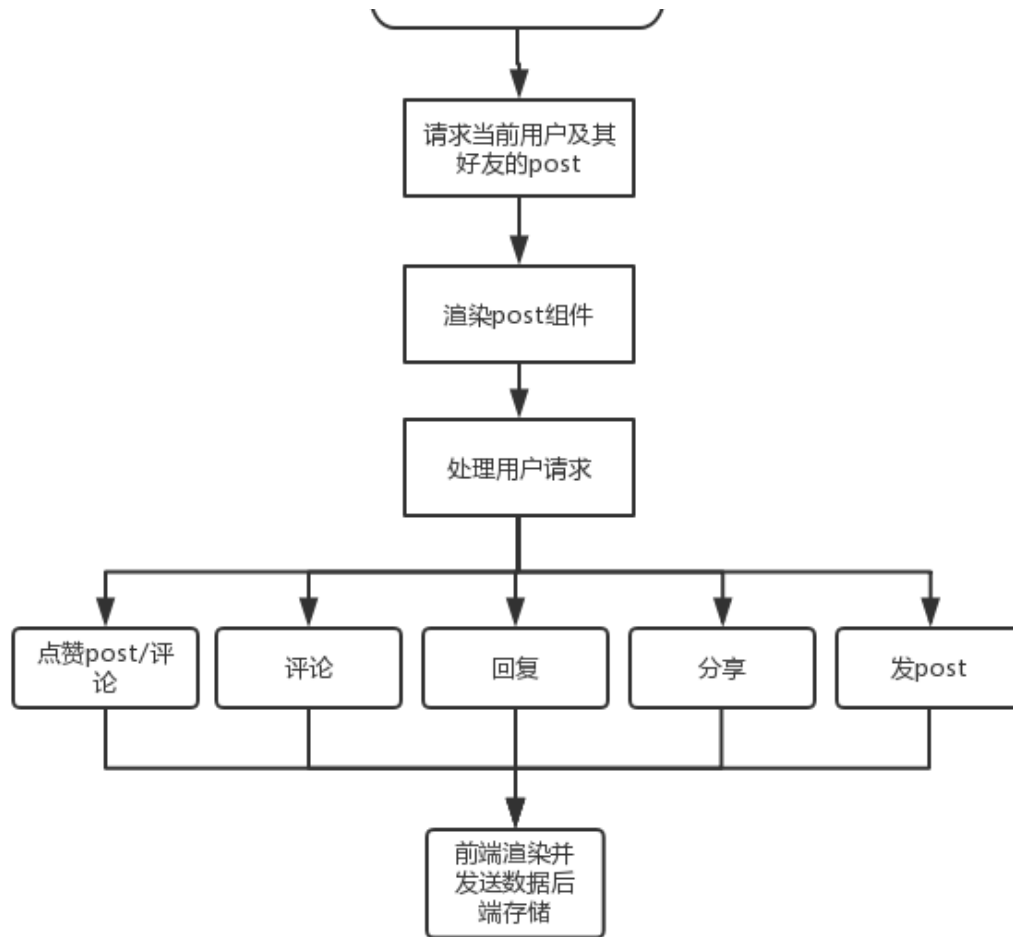
Experiment 1: Web service with database support

项目介绍

- 项目名称：MyCloud
- 项目概述：MyCloud为用户提供了网盘和社交两方面内容
 - 网盘：
 - 提供了文件的上传、下载、分享、重命名等功能；
 - 提供了文件的删除、恢复、彻底删除等功能；
 - 提供了音频、视频两种类型文件的预览功能；
 - 提供了音频、视频、图像、文档四种文件类型的分类查询与操作功能；
 - 提供了验证邮件服务，解决了用户忘记密码的问题。
 - 社交：
 - 提供了用户之间的好友功能与提醒功能；
 - 提供了用户之间的点对点聊天；
 - 提供了表情包、图片、超链接等基本的说说功能；
 - 提供了点赞、评论、回复、分享等功能；
 - 提供了用户之间的网盘资源分享以及更好的预览功能。



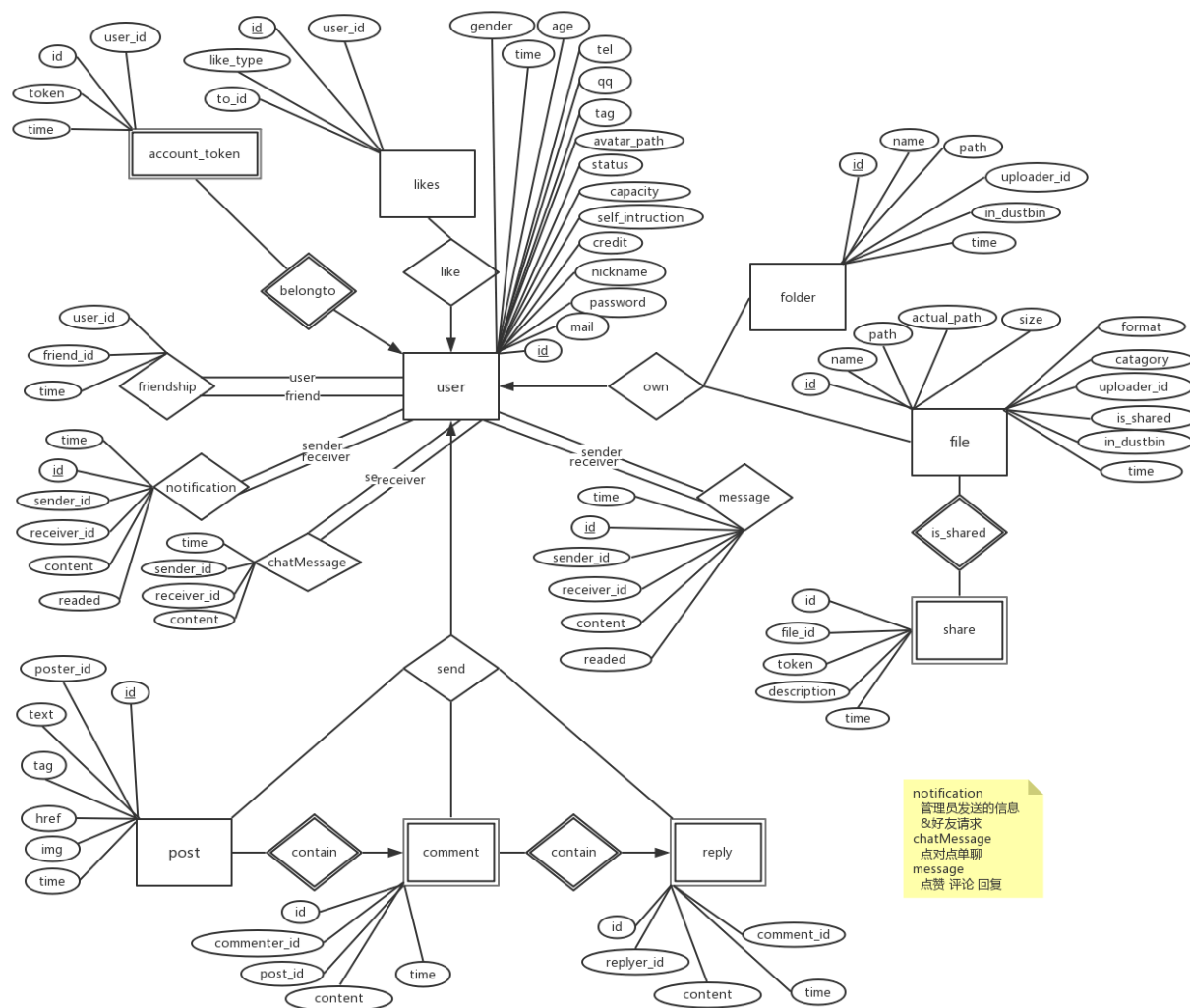
用户进入社交页面



项目架构

- 采用Restful架构，前后端分离，利用NodeJs处理高并发请求
 - 资源与URI：前端通过ajax向统一资源接口下的对应URI发送请求，通过Node提供的API获得对应资源
 - 资源的表述：采用JSON进行规范的前后端通信，前端对后端返回的JSON加以处理并渲染
 - 无状态：前端负责维护应用状态，而服务器维护资源状态。前端与服务器的交互是无状态的，并在每一次请求中包含处理该请求所需的一切信息。服务器不需要在请求间保留应用状态，只有在接受到实际请求的时候，服务器才会关注应用状态
 - Restful的资源与无状态规范，以及URI的规范使得路由控制更加方便且容易理解，JSON的资源表述避免了消息的繁杂冗余，保证了消息的简洁有效
- 前端采用 Bootstrap3 作为前端框架，并使用sb-admin-2作为前端主题，使用 fontawesome作为图标库。
 - 使用了诸如cropper.js、audioplayer.js、video.js、filtertable.js等服务
 - 图床使用https://sm.ms/api/upload这一API
- 后端采用JSP服务器与Node服务器相结合
 - JSP服务器
 - 验证登陆状态实现跳转和部分的静态的后端渲染（头像，昵称）
 - Node服务器
 - 利用node能够有效处理高并发IO，和简便的数据库操作的优点，处理大部分高频的ajax请求
 - 包括文件的增删改查请求的处理，通知的派发和实时聊天通信

- 数据库采用MySQL数据库



项目技术

- Java编写邮件系统，用户登陆验证，MD5加密模块
 - Java mail采用了多线程处理，调用QQ邮箱SMTP服务。JSP当成模板引擎使用，对部分的页面做后端渲染。
 - 生成token: token是一个32位的MD5码，随机生成，且用了辅助函数tokenExist()来保证了token的唯一性。
- 用户头像的上传与裁剪
 - 调用cropper.js，使用其作为截图工具，并通过HTML5的canvas绘制出裁剪结果，并将裁剪结果压缩后转成blob数据通过ajax传给图床，并将图床返回的图片在服务器上的地址再通过ajax传送给后端，保存进数据库，并借助refreshSession.jsp将用户头像数据存入session，以供跳转后的主页面使用。
- 网盘页面
 - 列表渲染: 考虑到原生JS渲染元素过多的table时代码过多且冗余，效率不高，采取了jQuery渲染的方式，分别渲染出文件夹、文件(对应其类别的)图标、文件名称、文件格式、文件大小、文件上传时间。
 - 列表右键菜单: 禁用了鼠标右键的默认事件，获取鼠标坐标并建立右键菜单，调整效果并用switch语句识别功能选择，分别完成交互。
 - 文件预览: 考虑到网盘页面大量的用户交互，而文件分享的展示页面则不需要用户交互所以在两个页面的文件预览功能，分别采用、<video>、<audio>等原生标签预览和

audioplayer.js与video.js两个jQuery插件两种方式预览

- 网盘主体：前端渲染table时将文件的name、path、category的信息送给dblclick事件响应的preview函数，再使用原生标签，完成预览。
- 分享页面部分：通过jsp渲染html时留下的文件信息来判断文件类别，并结合bootstrap的折叠板技术，在折叠板内调用audioplayer预览音频，调用video预览视频，并调整插件效果。
- 文件与好友的搜索
 - 使用filtertable.js，实现了文件与好友的模糊搜索、高亮显示、字典序排序
- 社交页面
 - 页面主体：参考bootstrap框架的demo，将post组件化为post_body、post_line、post_mainMenu、post_comment、post_footer、comment_menu、comment_reply等，再结合折叠板组件完成了textarea的动态效果，并复用之前的页面渲染的组件，在ajax与后端交互存入数据库的同时，前端渲染出post/comment/reply/thumbs-up/share的效果，并将后端发来的组件所需的新id赋给对应的标签。
 - Post发送框：结合正则表达式完成了对文本内的表情包(如：[emoji/1.png])标签的识别，并结合模态框技术与FileReader的相关函数，将图片文件转为base64并显示再将其转为blob数据，发送给图床并获取返回的地址存到数据库中。
- 图标：使用fontawesome图标库，完成所有图标的选择与相关效果调整。
- 图表：使用chart.js，将其组件化供个人中心调用。

项目部署

- 数据库部署
 - /MyCloud/nodeServer/SQLbulider.txt：运行其中的SQL语句完成数据库的建立
- Node服务器搭建
 - /MyCloud/nodeServer/MySQLConnectionConfig.js：修改其中的数据库配置
 - 一般只需修改user与password即可
 - /MyCloud/nodeServer/fileRouter.js：修改其中的绝对路径配置

```
let newPath =
'E:\\college\\Junior\\MyCloudUploads\\'+req.file.filename+fileObj.ext;
// 文件的绝对路径
// 位于文件的103行，仅需修改 E:\\college\\Junior\\ 作为网盘文件实际存储的根
目录，并在根目录下新建文件夹MyCloudUploads
```

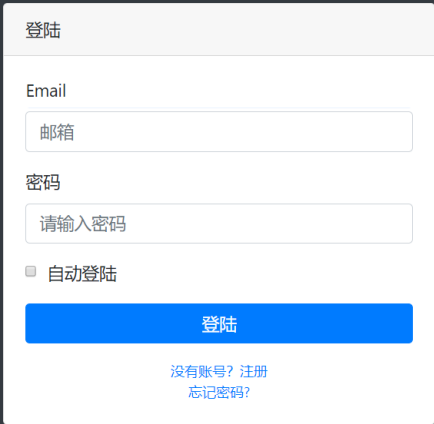
- npm install express cookie-session cookie-parser mysql multer
- Tomcat服务器配置
 - 采用eclipse-ee版本开发，使用该软件打开/MyCloud/TomcatApp文件夹
 - /MyCloud/TomcatApp/src/Config.java：其中的MySQL配置也需要如上修改，Mail与server的配置可以不做修改
 - /MyCloud/TomcatApp/WebContent/js/config.js：其中的文件服务器、Node服务器、JSP服务器以及图床的地址都可以不做修改

- 文件服务器配置
 - 项目采用Apache服务器，为部署方便，可使用`python3 -m http.server 80` 直接启动HTTP服务
 - 如只安装了Python3，可以直接使用`python -m http.server 80`启动HTTP服务
 - 以`E:\\college\\Junior\\` 作为网盘文件实际存储的根目录为例，其下已新建MyCloudUploads文件夹，在该根目录下启动HTTP服务

```
E:\college\Junior> python -m http.server 80
```

项目使用

- 登录与注册



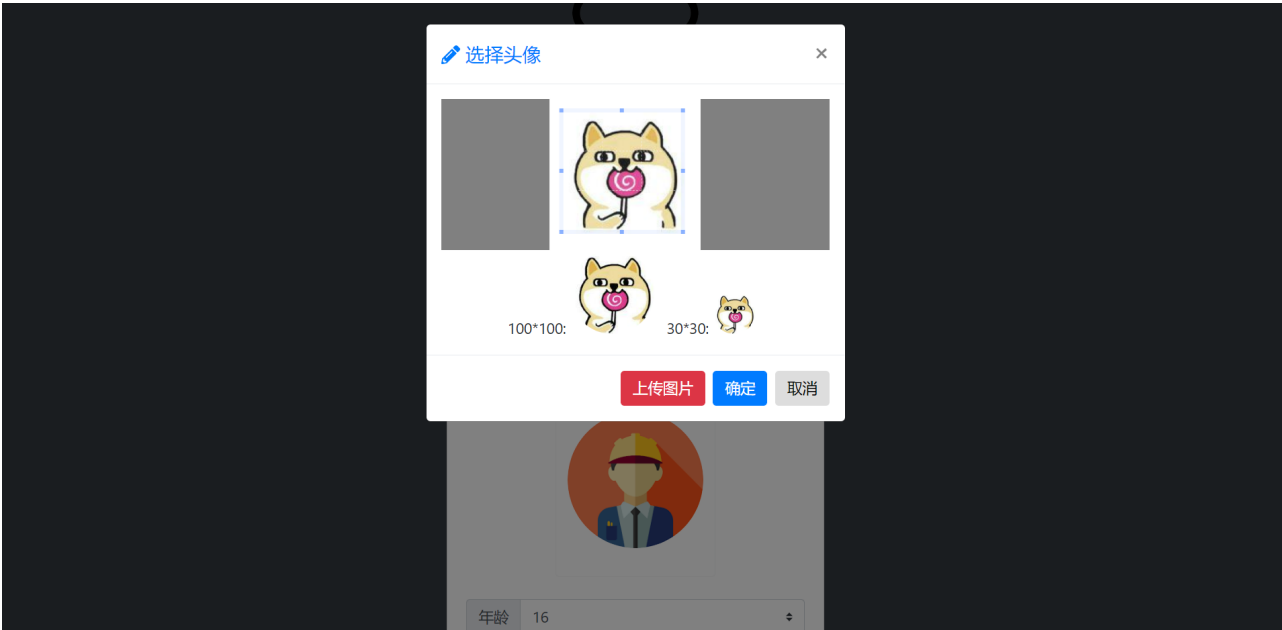
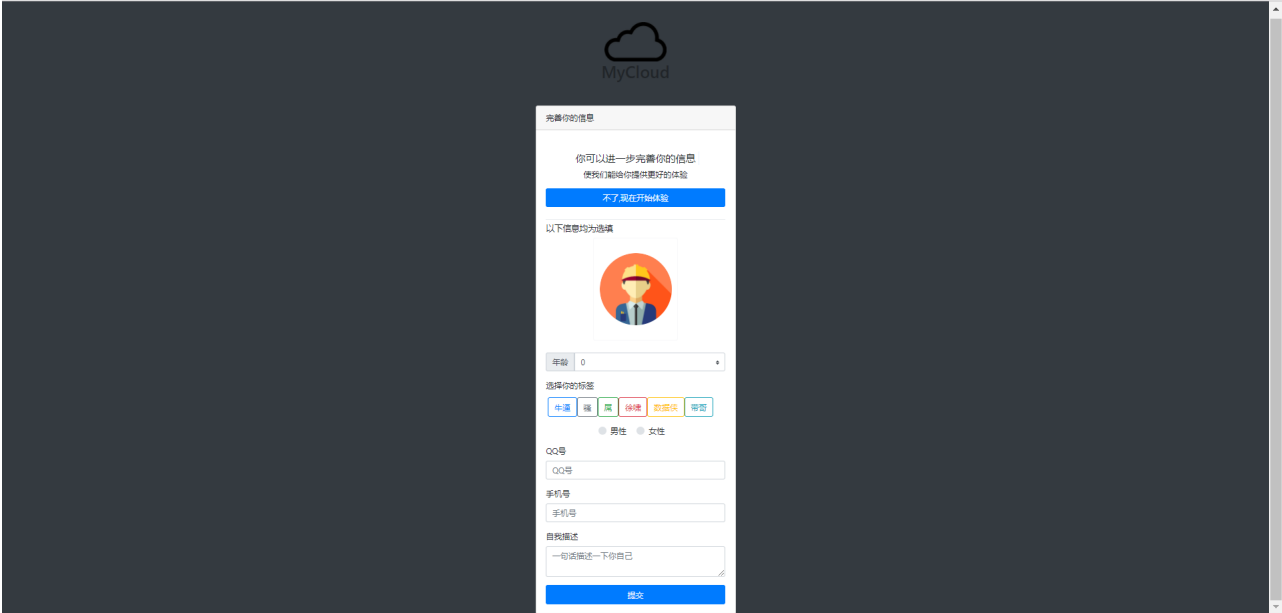
The login form is titled "MyCloud" with a cloud icon. It contains a "登陆" (Login) section with the following fields and options:

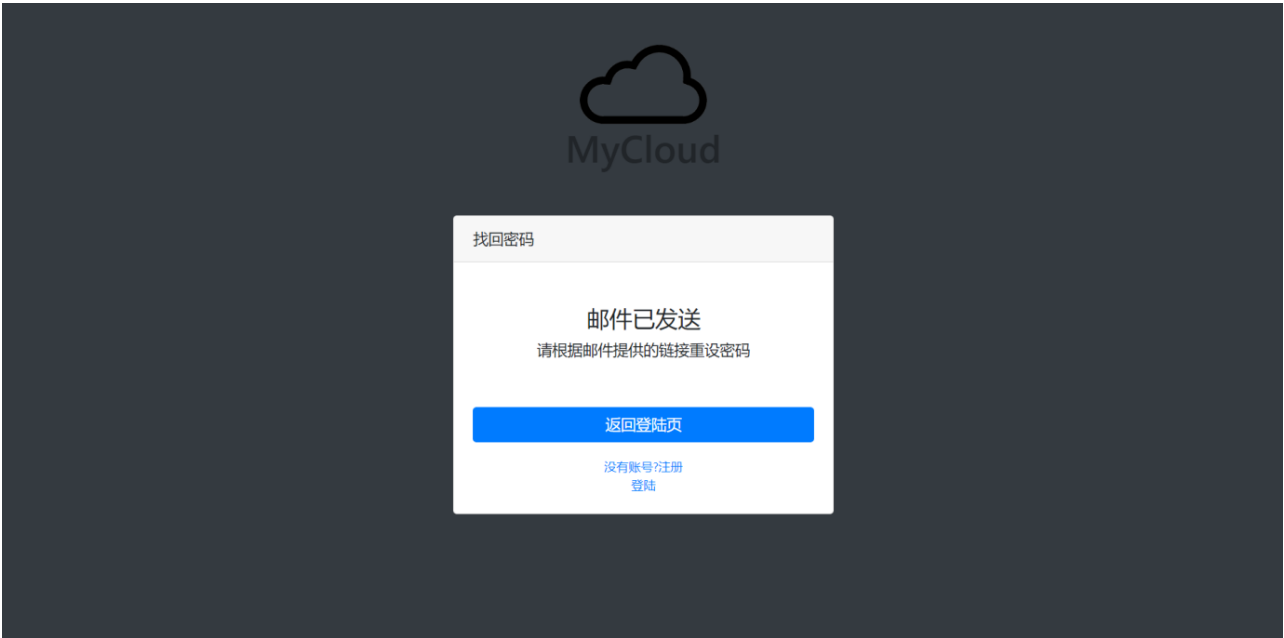
- Email: 邮箱 (Email)
- 密码: 请输入密码 (Password)
- ☐ 自动登陆 (Remember me)
- 登陆 (Login button)
- 没有账号? 注册 (No account? Register)
- 忘记密码? (Forgot password?)



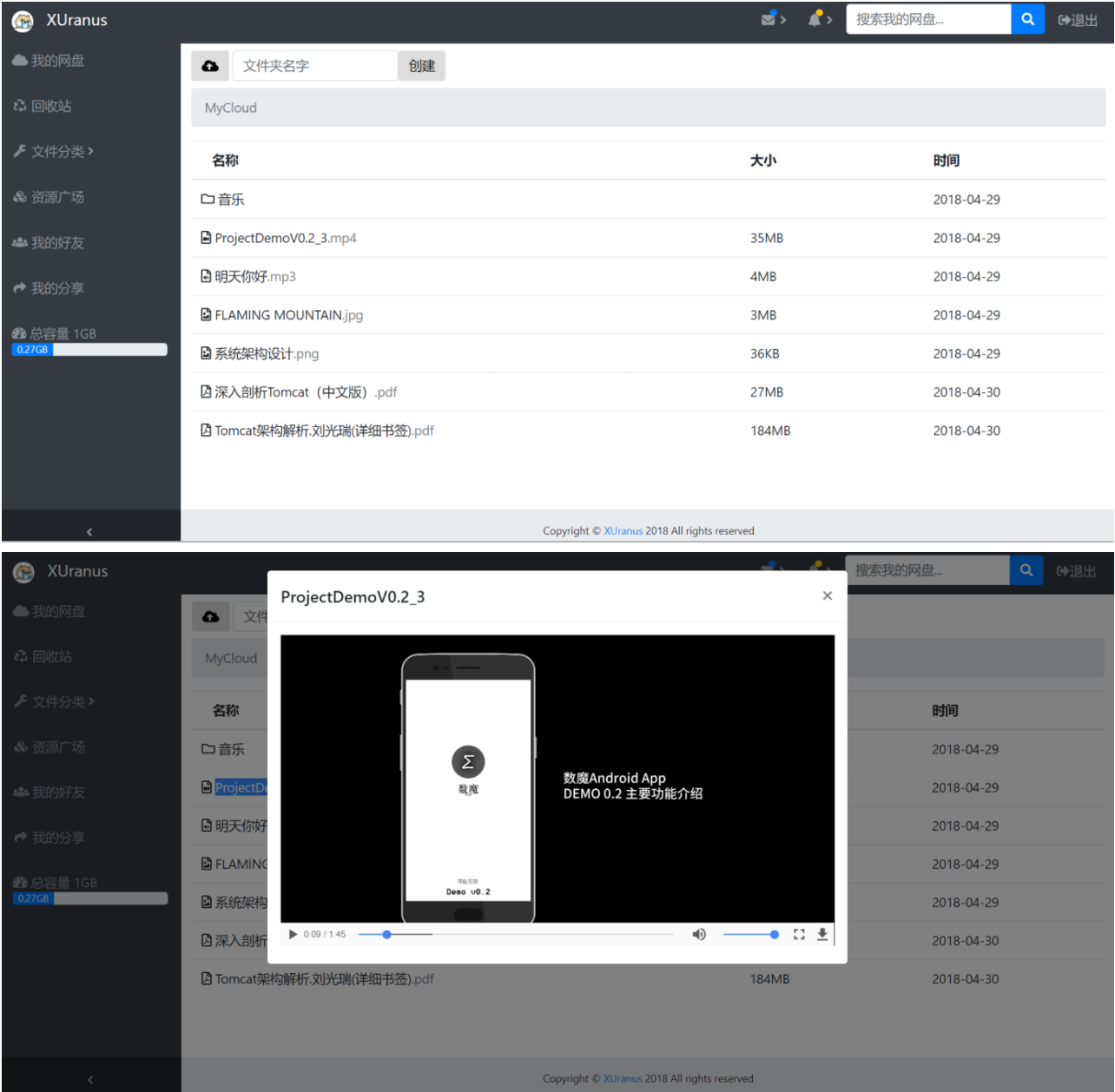
The registration form is titled "注册MyCloud账号" (Register MyCloud account). It contains the following fields and options:

- 邮箱: 输入你的邮箱... (Email: Enter your email...)
- 昵称: 昵称最多20个字 (Nickname: Nickname up to 20 characters)
- 密码: 输入密码 (Password: Enter password)
- 确认密码: 再输入一次密码 (Confirm password: Enter password again)
- 注册 (Register button)
- 已有账号? 登陆 (Already have an account? Login)
- 忘记密码? (Forgot password?)





• 网盘



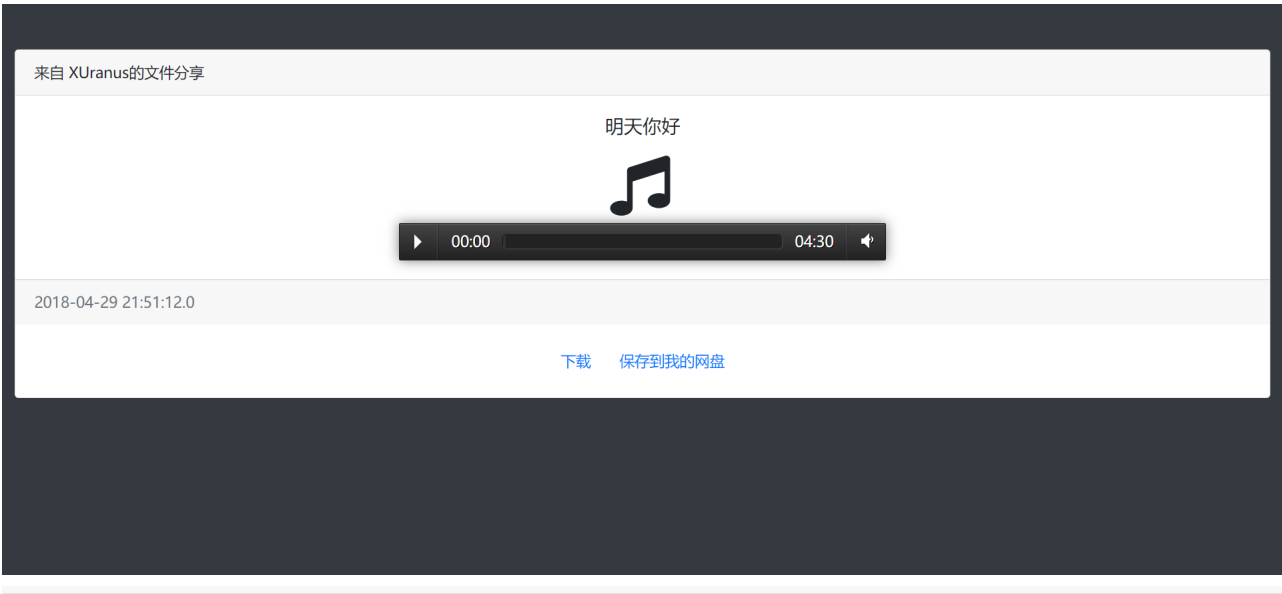
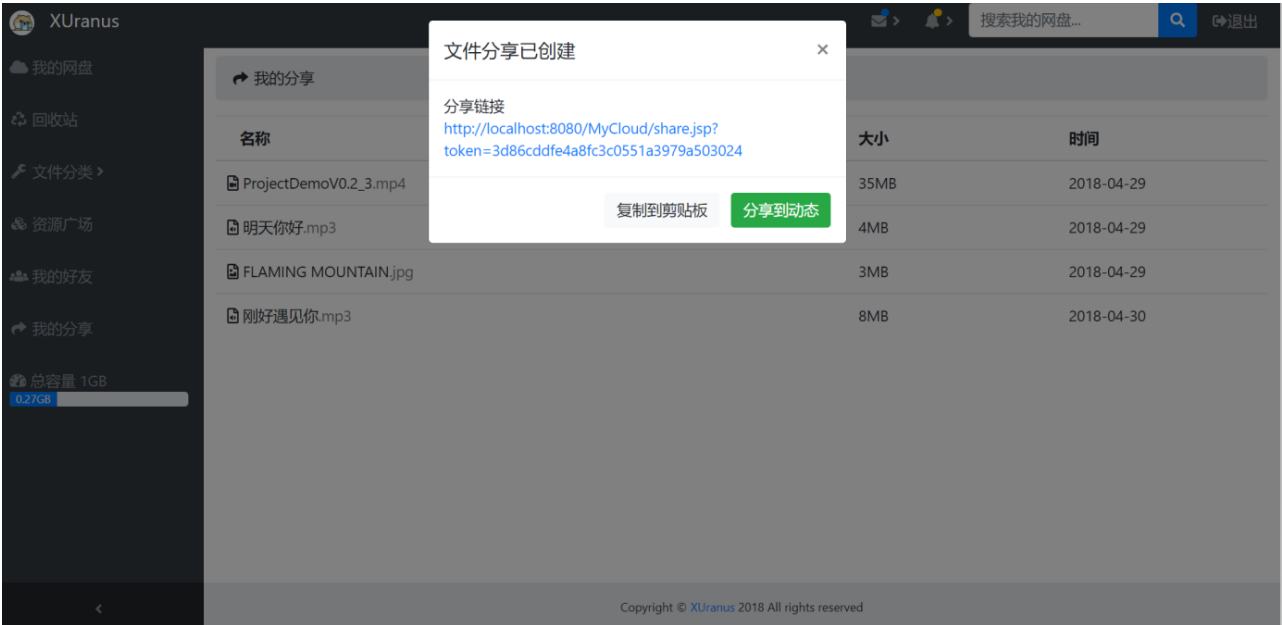
×

Browse

类型: image/jpeg

关闭

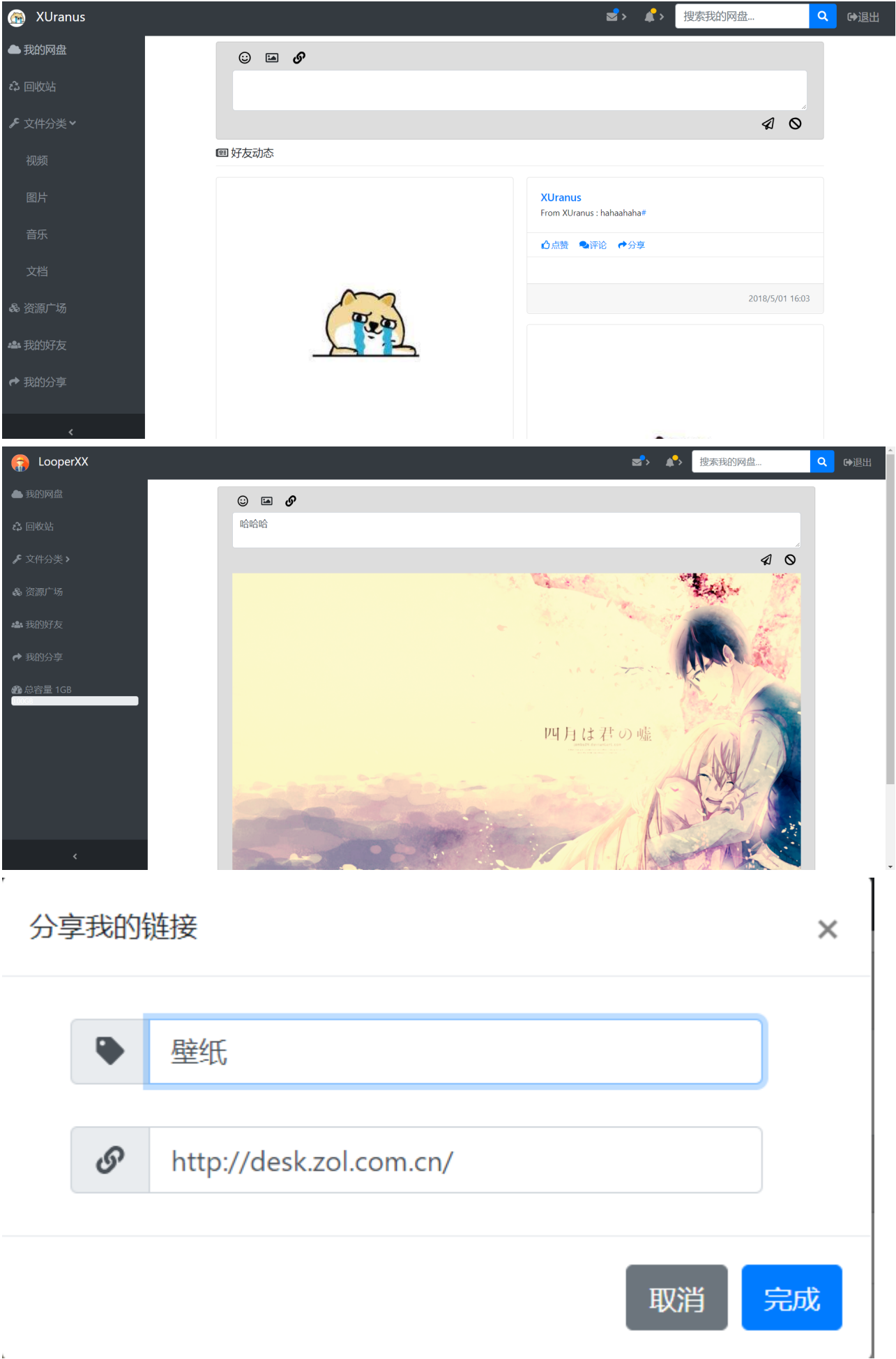


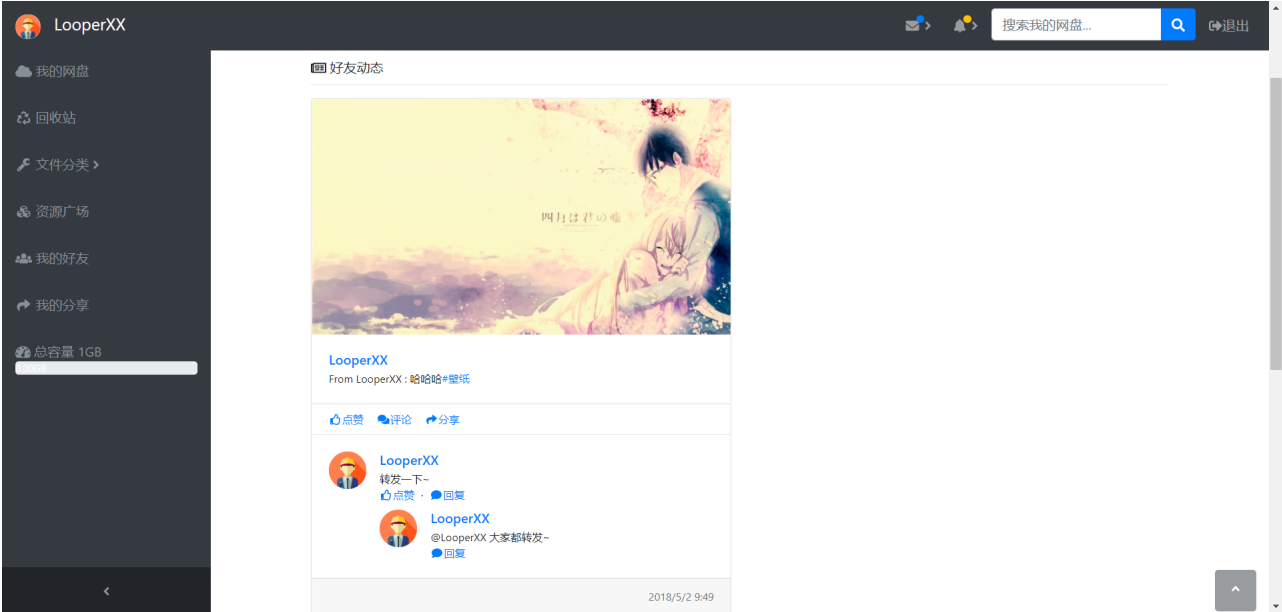
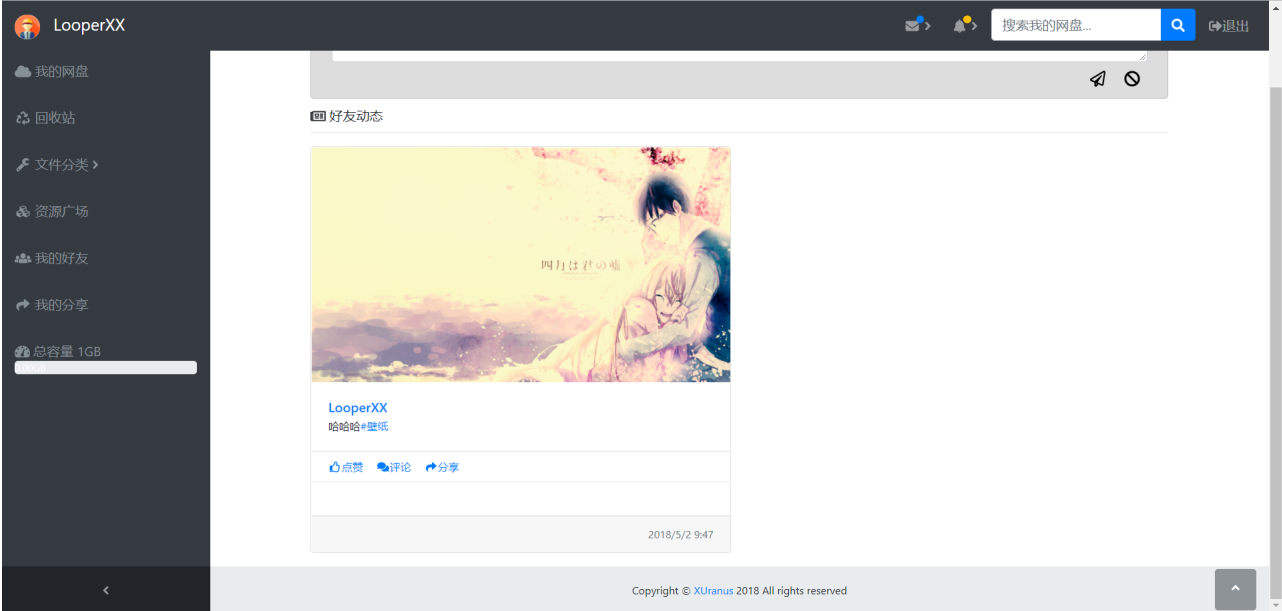


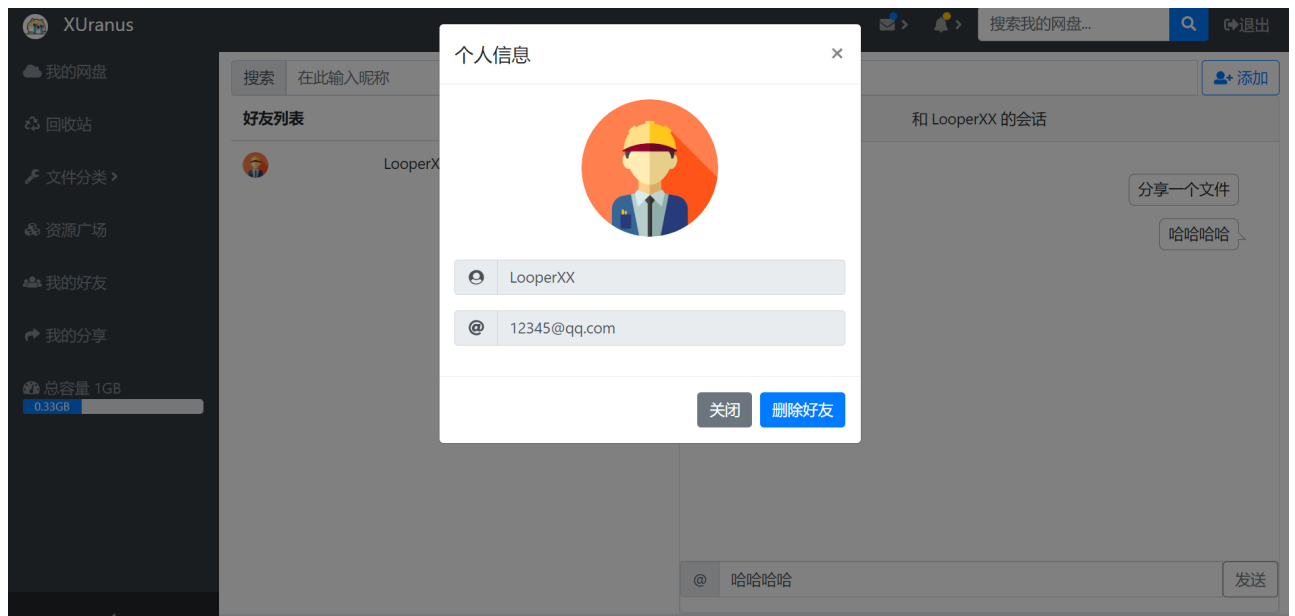
ProjectDemoV0.2_3



• 社交







Face2u

Experiment 2: Web service without database support

项目介绍

- 项目名称: Face2u
- 项目概述: Face2u针对摄像头与本机视频的两种数据流, 提供以下服务
 - 人脸检测
 - 人脸关键点识别
 - 人脸关键点绘制
 - 情绪识别
 - 人脸属性识别
 - 性别、年龄与人种
 - 颜值评分
 - 肤质评分
 - 人脸美颜

项目架构

- 采用Restful架构, 前后端分离
 - 资源与URI: 前端通过ajax向统一资源接口下的对应URI发送请求, 从而获取对应资源 (每一个URI代表一种资源)
 - 资源的表述: 采用JSON进行规范的前后端通信, 前端对后端返回的JSON加以处理并使用Vue.js渲染
 - 无状态: 前端负责维护应用状态, 而服务器维护资源状态。前端与服务器的交互是无状态的, 并在每一次请求中包含处理该请求所需的一切信息。服务器不需要在请求间保留应用状态, 只有在接收到实际请求的时候, 服务器才会关注应用状态
 - Restful的资源与无状态规范, 以及URI的规范使得路由控制更加方便且容易理解, JSON的资源表述避免了消息的繁杂冗余, 保证了消息的简洁有效
 - 结构清晰、符合标准、易于理解、扩展方便
- 前端采用基于Material Design的主流前端响应式框架Materialize与JavaScriptMVVM库Vue.js, 使用Google发布的Material Icons作为图标库
 - 使用了诸如Echarts.js、Face-api.js、Vue.js等JS库, 并调用\$Face^{++}\$人工智能开放平台提供的API
 - 借助\$Face^{++}\$的人脸识别API中的Detect API与Analyze API, 调用其人脸检测、人脸属性、情绪识别、颜值评分、人脸美颜和肤质评估等服务
- 后端采用轻量级 Web应用框架Flask
 - 采用Flask的扩展Flask-RESTful, 快速构建REST APIs的支持, 提供了构建在Flask可拔插视图之上的Resourceful Routing
 - 只需在路由后定义方法, 处理请求并分配资源, 即可完成前后端交互
 - Flask服务器接受前端的资源请求, 访问\$Face^{++}\$的API并将其返回值处理成前端可用的JSON字符串, 作为其所需资源发送给前端

项目技术

- 采用Vue.js
 - 以数据驱动和组件化的思想构建的JavaScript MVVM库
 - 通过ViewModel完成Model与View之间的交互, 实现数据的双向绑定
 - 从View侧看, ViewModel中的DOM Listeners工具会帮我们监测页面上DOM元素的变化, 如果有变化, 则更改Model中的数据
 - 从Model侧看, 当我们更新Model中的数据时, Data Bindings工具会帮我们更新页面中的DOM元素

- 前端收到服务器发来的资源后可以快速渲染，呈现结果
- 采用Face-api.js
 - 使用最新的Face-api.js开源库，基于tensorflow.js实现浏览器内的人脸检测与人脸关键点的抓取，无须配置任何环境。
 - 采用其提供的Tiny Face Detector人脸检测模型。该模型是对Tiny Yolo V2的简化版本，用深度可分离的卷积代替Yolo的常规卷积，实现了完美的实时性。
 - 采用其提供的68个人脸关键点的检测器
 - 针对本地视频与摄像头视频的两种视频形式，在原视频之上覆盖新的Canvas
 - 将人脸检测结果以蓝色Bounding Box的形式，绘制在Canvas上，标识出识别到的人脸
 - 将人脸关键点的检测结果，按照68个人脸关键点的物理意义连接绘制成为绿色的人脸轮廓图
 - 实时调整Input Size与人脸检测的scoreThreshold
- 采用\$Face^{++}\$API\$
 - 将截屏获得的人脸图片转为Base64编码并发送给服务器，服务器将其组装成所需JSON发送给API
 - 从而完成面部情绪识别，年龄、性别与人种检测，颜值评价，肤质评价以及美颜效果
 - 方便用户根据面部关键点与面部分析数据
 - 调整自己的面部表情，增强自身的演讲、面试、辩论时刻的表情控制能力
 - 了解自己的颜值、肤质的情况
 - 获得简单的美颜效果
- 采用Echart.js
 - 将情绪分析结果以柱状图/折线图的方式可视化呈现
 - 解析服务器发来的JSON，组装为ECharts所需的options，并重新渲染

项目部署

```
pip install flask
pip install flask-restful
# Flask-RESTful requires Python version 2.7, 3.4, 3.5, 3.6 or 3.7
```

- PyCharm下运行app.py即可

项目使用

- 由于本项目面向视频流的数据提供服务，故录制了使用视频Face2u_使用视频.mp4，视频流程如下
 - 主页
 - 人脸识别_本地视频
 - 人脸检测
 - 人脸关键点检测与绘制
 - 情绪识别
 - 人脸识别_摄像头
 - 人脸检测
 - 人脸关键点检测与绘制
 - 情绪识别
 - 美颜 颜值与肤质评分
 - 对摄像头进行截屏
 - 调整美颜指数并获得美颜后的图片

- 情绪识别
- 人脸属性识别
 - 性别、年龄与人种
- 颜值评分
- 肤质评分
- 人脸美颜