项目

直播+训练营的模式

增删改查

这个项目，尽可能把每个需求，都做成亮点（站在架构师的角度，来设计项目）

1. 纠正视角，不从开发工程师的角度来看，增删改查的代码，只是一小部分工作
2. 一个项目需要什么
    1. 文档
    2. 版本控制(git) 规范 （分支，message） （gitlab，github）
        1. git分支管理 master test dev
    3. 质量 （代码质量，eslint，jest、 jira 。。bug管理）
    4. 开发流程 （敏捷开发）

5. 写代码（代码设计（分模块，分任务），代码实现，联调）
6. 发布部署（自动化部署） mvp版本发步，给产品虐待 后续考虑的工作和任务
7. 维护，功能开发4和5持续执行
8. 开发效率（组件化，发npm包 考虑私有npm服务）
9. 权限， 监控，统计，报错收集 量化我们的产品性能
10. 上面提高开发效率的内容，考虑固化沉淀为系统，这就是前端团队的基础建设

3. 一个项目怎么做才算亮点
    1. 每个需求 ，都可以做成你的亮点，只要你有心
        1. 数据量想的贼大
        2. 网络情况不稳定
        3. 用户体验（把用户想成傻）
    2. 文件上传
        1. inp ut type=file , axios.post ,node接受文件 存起来，over 最多加一个上传进度条
        2. 粘贴，拖拽
        3. 文件2个G的视频，网速100K还不稳定，

1. 文件切片，分片上传
2. 断点续传（上传之前，后端告知已经存在的切片）
3. file.slice() 就可以做文件切片了

4. 如何让后端只知道你是哪个文件 如何确定文件的唯一性，用文件名肯定不靠谱
    1. Md5 2各的G的文件 大概计算md5 要15秒左右的时间
    2. 怎么解决卡顿问题
        1. webworker （会额外加载js，）
        2. 思考一下，学习的框架源码，怎么处理任务量大这个场景
            1. 时间切片来计算，利用浏览器空闲时间计算
            2. requestIdleCallback 你也可以自己模拟，React就是自己模拟的，利用event-loop的机制就可以模拟
        3. 抽样哈希
            1. 抽取特征值
            2. 每个切片都是1M， 第一个切片和最后一个切片全部的数据
                1. 中间的切片 取前中后2各子杰，拼在

一起

2. 文件多大， 抽样值都在3M以内

3. 布隆过滤器

4. 两个文件hash一样，可能文件不一样，hash不一样，文件一定不一样

5. file.slice 不会造成卡顿，浏览器并没有新建内存区间来存储

3. 计算hash卡顿解决了，比如又100各切片

    1. 如果直接promise.all上传，浏览器发起100各tcp网络请求，虽然浏览器又并发限制，只会又6各传递数据，同时建立这么多请求i，会让浏览器卡顿

    2. 控制并发数 比如控制在4，异步任务的并发数控制 ，使用队列就可以了 这个功能，本身就是头条经常用的笔试题

    3. 还可以做报错重试

        1. 异步任务通过一个队列 任务报错，出列，再塞进去
        2. 同一个任务报错3次，或者2次，统一终止整个上传任务，提示用户报错，重试 用对象{task1:1}

4. 根据网速确定切片大小

    1. 先穿一个切片，看看返回的时间
    2. 怎么流畅的判定呢
        1. TCP的慢启动逻辑就可以，很流畅
        2. 先丢一个小区块， 判断返回时间，如果比较短 *2 如果超市/2
        3. 2这个系数，可以用一些数据公式 变得平缓一些
5. 以上，下次面试官胆敢在问你文件上传，你还说不出亮点吗
6. 文件扩展名，怎么判断用户上传的是符合要求的文件呢
    1. 如果我们要求只能上传png图片
    2. 每个文件都有固定的头信息，二进制的文件流 固定位数的值，确定一个文件类型，通过文件内容判断，而不是简单的后缀名
    3. 图片你的宽高，也在二进制里
    4. 依然在我的掌控之中
5. 基本上上面大家的提的问题，到此为止，基本是我考虑到所有的点
    1. 你们考虑了需求，但是没考虑解决方案
1. 底气来自于平时的思考和准备

二进制写起来比较麻烦， 4各二进制一起，变成16进制好现实

1. 表格渲染，列表渲染
    1. 数据量大，虚拟列表 分页，虚拟列表

需求

1. 登录注册 jwt
2. 个人中心 图片上传
3. 文章发布
    1. 简单的定制一下markdown编辑器
4. 文章的列表 考虑虚拟列表
5. 用户关注，文章点赞，评论
    1. 用户一对多，多对多的关系设计
6. webrtc
7.

技术选型

1. 技术选型没有对错，只有合不合适
2. VUE REACT

1. 团队现状
2. 上手难度
3. 技术生态

element VS ivew VS;;....

1. 组件数
2. npm下载
3. 团队人数
4. 某个组件
5. 按需加载
6. 配合的admin框架
7. 。。。。。

开发规范

1. eslint
    1. 老项目，可以考虑增量eslint： lint-staged
2. git分支 dev=》test=》master
3. git 钩子
    1. precommit之前，跑eslint
4. git log规范
    1. git commit -m'日志规范
5. Npm script工作流
6. 目录规范nuxt+eggjs 这俩自己的规范，我们用就可以了
7. 统计
    1. 百度统计
    2. GA
    3. growingio
8. 报错
    1. sentry
9. 代码部署
    1. github action或者gitlab 简单的自动化
    2. push触发任务，跑测试，发布部署，部署结果通

知钉钉

10. AXIOS配置等等

以上所有，训练营每行都会敲代码

做需求的时候，用插件仕没问题的，但是，想进步，就要看源码

我们开发项目，用vue，想进步，不能只会用，而要看源码

站在一个稍微高级一点的视角，一个项目到底需要那些东西

目的是为了站在一个架构师的角度

代码能力只是其中一部分(整体把控)

我们会做一个项目 训练营的方式来敲，今天算是一个启动

需求

1.