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.

前端项目02

上次回顾

项目，对自己的要求，做成什么样

1. 数据量想大
2. 网络，电脑的性能想的差一些
3. 用户想的傻一些

前端架构师的身份，如何看待项目

从工程师=》前端管理的过程，别的知识体系

1. 项目人员的规划
2. 需求问题（项目管理）
3. 项目的可维护性 （未来）
    1. 代码的实现是开心的现在
4. 小项目的leader之后，如何进一步

1. 每一条路，对知识和能力的要求i仕不一样的

就从这个项目开始展开

1. 登录注册
2. 用户中心
3. 文章管理
4. 关注点赞（多对多的关系设计）
5. 评论

大部分的项目，都可以这个几个需求展开扩展

做项目之前

做项目的代码

做项目之后

1. 制定规范(规矩)
2. 开源（开发效率，影响力）
3. code review
4. 工具（开发工具）

代码之后

1. 人效（每个维度综合考虑）
    1. 代码量（吐槽），可以用gitlab的api，来统计
    2. 每个人都要触碰到自己的极限
        1. 比如轮岗（每个人尽可能能维护两个模块+）
        2. 比如内部系统奖励机制，发挥内部主管能动性
            1. 日志监控，主动领任务
        3. 。。。。。
2. 基础建设
    1. 项目长期维护的必备设置，美好的未来
    2. 命令行工具 cli
    3. 组件（代码规范，单元测试）
    4. 埋点（数据采集）
        1. 性能监控 浏览器或者node，统计的性能参数，发给我们服务器
            1. perfomance
            2. lighthourse (宏观的)
        2. 错误监控
            1. sentry
            2. fundebug也挺有意思的，复现功能
            3. 原理也不难，window.onerror， try catch

主动上报
4. promise报错
3. 用户行为日志监控
    1. ga
    2. 百度统计
    3. growingio
5. 构建发布
    1. gitlab
    2. github(action)
    3. 自动化发布，钉钉推送消息
3. 计划 （向上管理）
    1. 团队的资源
    2. 团队的成长
    3. 目的是通过某个项目，优化自己的知识体系

同一个项目，同一个需求，不同的前端，做出来就是不一样

对团队和自己的要求

技术级别，有点像学历，

1. P5
    1. 独立完成 经验丰富，给我需求，奥利给
2. P6

    1. 要求担当
    2. 前沿研究，踩坑
3. P7
    1. 领域专家
    2. 体系化知识

技术路线，P几只是其中一条路，winter老师，玉伯，张云龙，尤大，张xinxu。。。有很多

1. 玉伯，体验科技 前端技术和管理深挖
2. winter P8之后，再搞教育
3. 张云龙 fis的作者 走的是全栈CTO的路线
4. 鑫旭，css 死磕
5. 尤大 独立开发者

前端的路线又很多，每条路要求的知识都不一样

1. 大公司前端架构师
    1. 技术原理，vue nuxt，egg等等
    2. 计算机基础
        1. 算法，网络，操作系统，数据库....
        2. 从算法开始 ， 算法训练营欢迎大家

    3. 网络协议跟上
3. 前端的前沿技术
4. 有没有兴趣读完vue源码

2. 大公司技术管理
    1. 多关注人
    2. 管理，听起来高大上，干起来很辛苦
    3. 人员的招聘规划，进度的管理，资源的协调。。。懂技术，懂人
    4. 同时，你的技术还不能落下，否则地下的人对你没有技术尊重，队伍不好带啊
    5. 工作的内容会发生变化
        1. 把我们的小开社区对外发布维护，你需要几个人
        2. 这几个人，怎么招 跟hr聊
        3. 招到之后，怎么建设团队
            1. 提升积极性
            2. 提升成员技术能力
        4. 流程有效执行
            1. 绩效 KPI，还是okr
    5. 绩效差的哥们，和好的哥们，怎么去跟他沟通

6. 怎么留人和开人。。。。。。有很多类似这种任务

3. 小公司技术总监
    1. 稍微弱化前端，强化后端或者全栈
    2. 强化产品能力
    3. 强化管理能力
    4. 接触的更广，更泛泛，更需要综合实力
4. 独立开发者， 独立做项目
    1. 公司就你一个人，做一个产品来养活自己
    2. 全栈！！！
    3. 营销
    4. 产品
    5. 用户增长....
    6. 什么都干
    7. 极可能多用开源和成型产品
5. 等等

摸鱼：上班偷懒

钓鱼：真的钓鱼

做项目之后的反思

加班

很多公司都有加班，长期996的结果，并不能提升你的效率。只会让你爱上摸鱼

1. 学习的重要性大于加班
2. 短期996可以接受，常年996 还是算了

我不是科班出身的，我有半年时间，学习时间只学算法

以后学不动了会不会被淘汰啊

1. 只学皮毛，一定会被淘汰
2. 学核心的知识，核心知识4，永不过时
    1. 算法
    2. 编译原理
        1. ast codegen核心理念
    3. 武林高手要学习内力，而不是十八般兵器
    4. 任何一个知识点，都有一本经典的书，死磕下来，你就超过了大部分同行
    5. 任何一个之前的技能，都不可能简单的获得

前端的路很宽广的，是技术圈最懂用户体验的，最有可能独立做出项目的

年龄限制，只针对增删改查程序员

读书

1. 无论是你想赚钱，提升编程，职位晋升，都需要看书
    1. 就算你想找对象，我也推荐一本书《魔鬼约会学》，如何聊天
    2. 技术层面的书，豆瓣8分以上，都值得看
    3. 我早上看书，无论几点睡觉，6点~7点起床，打两把游戏，看一个小时书
    4. 开课吧的装修，就i是图书馆，公司也可以看书
2. 读书其实是有快感的
    1. 薪资的提升《算法》
    2. 逼格比较高 ，适合装逼

项目的思考，职业生涯的思考