前端项目如何准确预估个人工时

补充

看来很多小伙伴对这个问题感兴趣，大家不要忽视了压工时这个事。

领导为什么会压工时？

1. 使他的KPI更好看
2. 不清楚做这个东西实际要多长时间
3. 因为第2点的原因，他也无法去争取合理时间
4. 部分人看着下属加班，有种大权在握，言出法随的畅快感

码农为什么不要轻易答应压工时？

- 无形中会打击你的自信心，当自信心全无的时候，要么是职业生涯结束，要么是变成人人都跑来拿捏一手的角色
- 轻易妥协，会让你的说的话，可信度降低。毕竟，别人随便说一下，激一下，你就妥协了，那很容易就让人觉得，你就是随意乱说一个时间
- 这会妨碍你对自己真实能力的认知和评估

被压工时了怎么办？

- 偶尔有少量任务，被压了少量工时，个人认为是可以接受的，毕竟不可能一切都能按规划走
- 大量工作被压工时，那就告知延期风险，你的工作速度应该保持不变，完不成，就让项目延期。如何解决延期问题？那是领导的事情，不是一个小码农应该操心的。

没怎么压工时，但把工作时间延长了？

- 首先，工作该是什么速度，就是什么速度，不要替领导着急着赶进度
- 其次，反馈这有延期风险，建议领导增派人手。（记得先和其他成员通个气）
- 该提加班就提加班，调休或加班工资是你应得的，累了就调休，你是人，不是机器

为什么要给自己留缓冲时间？加缓冲时间是工作不饱和？

- 加缓冲时间不是工作不饱和
- 8小时工作日，你不可能每分每秒都在写代码，谁也做不到。
- 你不可能熟悉每个API，总有要你查资料的时候，而查资料，可能你查了4-5个地方，才总结出正确用法，这需要额外的时间

- 你的工作随时可能被人打断，比如：开会，喝水，同事问你问题等等，这都会耗费你的时间
- 你拉取代码，提交代码，思考实现方式，和业务进一步确认需求细节，和UI沟通交互细节，自测，造mock数据，这都需要时间
- 如果没有缓冲时间，一个任务延期，可能会导致，后续N个任务都延期。
- 即使从项目角度分析，足够的缓冲时间，有利于降低项目延期风险

工作总是被人打断怎么办?

- 比如：开会，比如插入了一个紧急工作任务，这种较长时间的打断，那就将这些被占用的时间，写进工作日志，即时向项目组反馈，要求原本的工作任务加工时或延迟开始
- 被同事问问题。几句话能说清楚的，那不妨和他直说。几句话说不清楚的，那只能等你有空的时候，再给他解答。要优先考虑完成自己的工作任务。

大方的承认自己的不足，能力多大，就做多少事，明确自己的定位

可能有的小伙伴，可能被别人激一下，被人以质疑的语句问一下，后续就被人牵着鼻子走了。有很大因素是因为不敢承认某方面的工作能力暂有欠缺。其实大方的承认即可，有问题，那就暴露问题，如果项目组其他成员会，那就让他来教你，这也属于沟通协作。如果没人会，那说明这是一个需要集思广益的公共问题。

可能有同学觉得自己就是个小码农甚至因为自己是外包，不敢发表自己的想法和见解，其实大可不必，只要你就事论事，有理有据，完全可以大方说出来，你不说出来，你永远只能从自己的角度看这个问题，你无法确认自己是对的还是错的。错了咱改，对了继续保持。既不贬低别人，也不看轻自己，以平常心讨论即可。

明确自己的定位，就是个普通码农，普通干活的，项目延期了，天塌了也是领导想办法解决。自己不会的就反馈，别人不会自己会的，那就友好分享。不会的，不要羞于请教。干不过来了，及时告知领导，让其协调解决。坦坦荡荡，不卑不亢。

前提

1. 此方法是在没有技术阻碍的前提条件下预估，如果有技术障碍，请先解决技术阻碍
2. 此方法需要根据个人实际情况调整
3. 这里以普通的以vue,element-plus,axios为基础技术的管理系统为例
4. 这些都是个人见解，欢迎在评论区提出不同观点
5. 请先以一个普通的CRUD界面，测算自己的基本编码速度

为啥评估会不准确

自我评估时

领导给你评估时

功能	领导认为的	领导忘记的	领导认为的时间	实际时间
加个字段	加个显示字段而已，实际只要3分钟吧	码农要找到对应代码，查看代码上下文，或许还涉及样式的修改，后端接口可能还没有这个字段, 还要自测	20分钟	2小时
做个纯列表页面	前端只要把后端的字段显示出来就好了吧，肯定会很快	可能没有直接可用的组件，即使有，前端可能需要查组件文档，看具体用法, 还得处理loading状态，空状态，然后还得查看后端接口文档，看哪些字段需要额外处理，最后还得自测，甚至可能在真正对接前，需要自己造mock接口	2小时	8小时
		前端需要理解业务逻辑，需要做数据校验，对于类似下拉数据，图片上传，可能还要和后端沟通，数据从哪里取，分别表示什么意		

		思，怎么上		
编辑/新增界面	就写个表单，前端把数据提交给后端就完事了	传图片，提交数据后，成功后要怎么做，以及失败的异常处理，用户填了一半数据之后，刷新了界面，应该如何处理，后端接口没出来前，需要自己mock接口，用来自测	4小时	3天
一个响应式界面	就一个普通界面应该不至于有什么难度吧	忽略了这是一个响应式界面，前端需要与UI设计师沟通，确认在不同情况，界面如何响应，以及思考如何实现，如果业务数据还会对界面的响应式产生影响，那还得进一步深入分析	8小时	3天
实现多语言功能	多语言，不就是用编码代替原本的文字嘛，根本不需要额外的时间处理吧	前端需要考虑多语言数据从哪里来，多语言切换之后对原本布局的影响，多语言切换之后，表单错误提示的处理方式	不给额外时间	3-4天
		前端可能需		

		要进行数据		
做个3/4环	直接使用图表插件，调下API就出来了	转换，需要查看图表插件的文档，图表插件可能没有现成的，需要通过搜索引擎找类似的示例，然后模仿实现，甚至图表插件根本无法实现这种效果，需要用其他技术实现	3小时	4天
前期一个连续的类似界面	上一个界面和这个类似，把上个界面的代码复制过来，改改字段和接口，应该能很快完成	很多界面看着一样，但实际业务逻辑差别很大，只是界面表现形式类似，有些字段是动态显示/隐藏的，有些可以固定写，表单字段的验证逻辑，可能也不一样。并且上一个界面的代码都还没写，还没测试，这里还有很多不确定因素，直接复制还可能导致，同一个错误，在多个界面同时出现	2-3小时	前一个界面花了多久，这个界面可能还是花了差不多的时间
仿照xx官网的效果，做个静态界面	好多网站都是这个效果，这应该是烂大街的	某个效果可能是知识盲区，需要查	2天	1周，甚至可能做不了

静态界面	是拉入街的效果吧	资料		
参考公司内部其他系统界面，实现类似界面	现成的东西，这系统也上线好久了，应该把代码复制过来，稍微改改就OK了吧	当前这个人从未接触过这个系统，对这个系统一点都不了解，了解需要时间，可能另外的项目有自己的框架，和当前系统的框架不同，无法直接使用，另外一个项目无法直接给项目代码给你，只能让人给你讲解，但讲解人没时间或不是随时都有时间，或就是随意讲讲，另一个项目的这个界面，可能是经过多人集思广益，多轮讨论与重构才最终得到了这个效果	5小时	3-5天
		组件可能没有，有组件可能某些业务逻辑在这个低代码平台难以实现，需要咨询低代码平台的提供方，但低代码提供方，几个人需要服务几十个		

用低代码平台实现个界面	就是拖拖组件的事情，代码都不用写，应该很快	服务几千个人，无法优先给你解答，即使解答了，可能给出的方案不通用(或者他们还需要继续问他们内部团队的人)，遇到下个类似的情况，原来的解决方案又无效了。难以调试或无法调试，前端原本的知识储备，在低代码平台，仅剩原始的js语法有效	2天	3周

总原则

- 不要duang的一下，对整个界面/模块进行评估，应该对行列，表单项，逻辑点，进行评估，然后将总的时间加起来，就是这个界面的预估工时
- 要至少多估20%的时间，一个是因为你很难持续性的投入(比如：有人突然问你问题，上厕所，喝水，或有事请假)
- 请将一天的工作时间最多算6.5小时(因为你可能需要开会，可能被其他事情打断，可能有时不在状态，同时也算是给自己留点思考时间)
- 尽量不要在过了一遍需求之后，立马评估工时(不要被项目经理或业务的节奏带偏)，而是要自己再思考一遍需求，想想大概的实现逻辑，重难点等等，尽量不要当天给出工时评估
- 如果是给别人评估工时，那尽可能给别人多评点工时
- 工期紧的时候，加人有必要，但1个人干7天的活，7个人未必能1天干完
- 有公共组件和没有公共组件完成同样的功能，所需要的时间可能天差地别, 因此，请确保先完成公共组件的开发
- 请先将业务逻辑理顺，把工作进行拆分，直至自己能正确预估的范围内

前端有哪些地方需要耗费工时

- 思考实现方式
- 静态UI界面还原与响应式
- 业务逻辑实现
- 动态UI交互
- 后端接口mock
- 后端接口对接
- 自测

前端项目应该分成几步实现

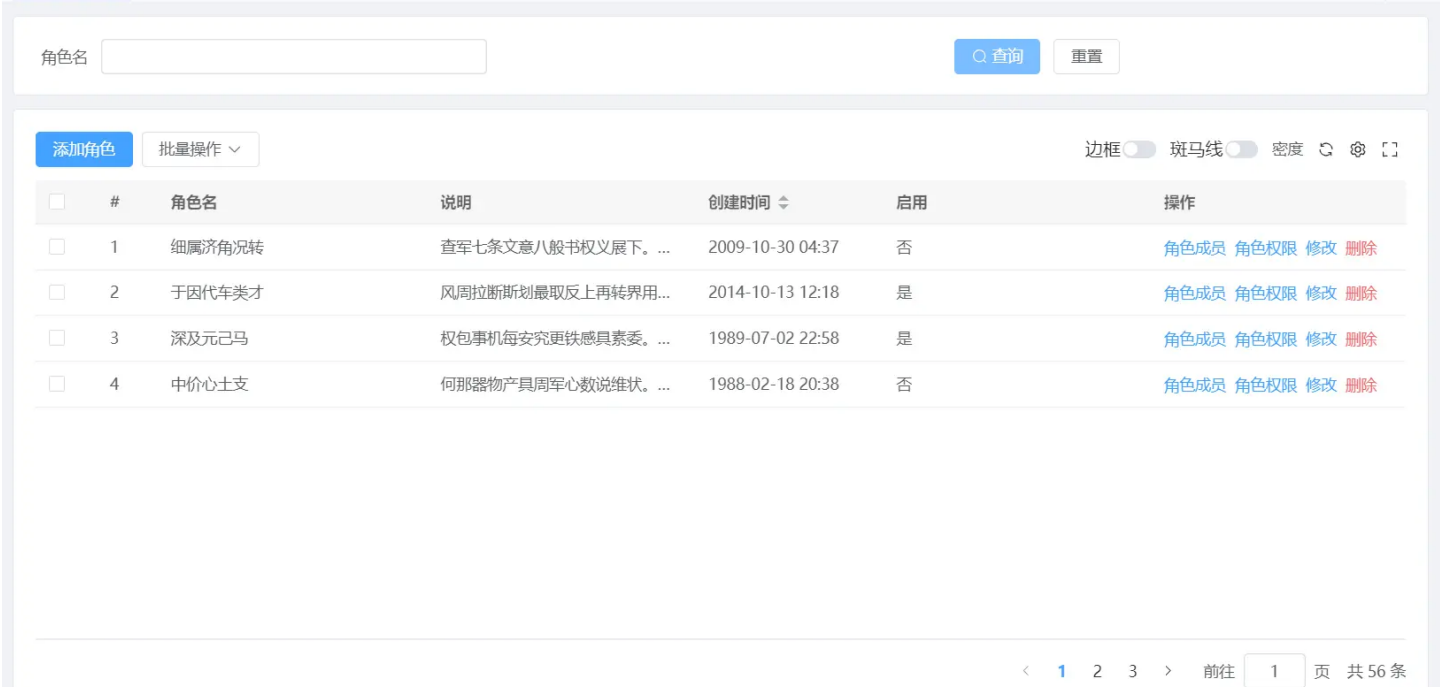
1. 整体项目搭建以及规范与约束确认
2. 整体页面结构，路由结构搭建
3. 统一UI风格，以及公共组件实现
4. 具体的界面实现

1，2点应该由项目组长完成 3点应该由项目组长以及技术较强的组员共同完成

常见的公共组件工时

组件	工时
查询按钮	60 分钟
提交按钮	60 分钟
confirm按钮	60 分钟
下拉按钮	60 分钟
分页表格	360 分钟
JSON配置分页表格	240 分钟
动态表单	360 分钟
JSON动态表单	360 分钟
模态框	90 分钟
抽屉组件	90 分钟
select组件	90 分钟
tree组件	120 分钟
cascade组件	90 分钟
日期选择组件	60 分钟
日期范围选择组件	120 分钟
axios封装	360 分钟
卡片组件	60 分钟
面包屑组件	60 分钟

列表页拆分与编码工时预估



首先做总体拆分，分成3大部分

1. 头部的搜索表单

每个表单项30分钟左右，每个功能按钮40分钟左右

因此这里是1个表单项（30分钟），2个功能按钮(80分钟)，总计110分钟

1. 中间的工具栏

P.S. 这里没算右侧工具条，只算了左侧功能按钮

因为是列表页，`添加角色`这个按钮，只考虑是个简单按钮加个点击事件，至于点击按钮之后的角色添加界面的工时不放在列表页评估，而是在添加角色界面单独评估，因此`添加角色`按钮算30分钟

批量操作按钮，应该使用公共组件的下拉按钮组件，以及与分页表格组件配合实现，因此算40-60分钟

因此这里整体应该总计在70分钟内

1. 主体的分页表格

应该使用公共组件的分页表格组件实现

- 普通列（直接显示字段值的列，和简单转换的列）每列算20分钟
- 操作列按每个操作按钮另算
- 复杂转换列按40-60分钟算
- 排序列按40-60分钟算
- 分页表格组件调用30分钟

从界面看，这里有6列，checkbox列和序号列，是分页表格组件实现的，无需再算工时，除操作列和创建时间外，其他都属于普通列算20分钟每列，创建时间列算40分钟，因此总共100分钟

操作列角色成员，角色权限和修改，都需要打开一个抽屉界面（抽屉界面里的东西另算，不算在列表页中），删除需要调后端接口以及确认，因此

- `角色成员按钮` : 20分钟
- `角色权限按钮` : 20分钟
- `修改按钮` : 20分钟
- `删除按钮` : 30分钟

总计: 100 + 20*3 + 30 = 190分钟

因此整个列表页工时

列表页需要mock 1个接口，列表接口，算20分钟

110 + 70 + 190 + 20 = 390 分钟 = 6.5小时

再在390分钟的基础上再多加20% = 390*1.2 = 468 分钟 = 7.8 小时

P.S.

1. 添加角色/角色成员/角色权限这是独立界面，需要单独计算时间。计算方式也与上面的类似
2. 没有单独计算自测时间，个人认为理想情况应该对1个界面，加2-3小时自测时间
3. 没有计算联调时间，联调时间应该另算
4. 没有计算UI还原时间，对于复杂UI界面或UI还原度要求高的界面，应该单独计算UI还原时间
5. 对于复杂的业务逻辑，可以将业务逻辑拆解为一条条的业务逻辑项，每个业务逻辑项以40分钟左右每条作为参考实现时间
6. 没有考虑思考时间，对于复杂的业务逻辑，或者没做过的界面形态，或者复杂的界面形态等，必须将思考时间计算进来，或者说，在已经基本想明白怎么去实现的基础上，再去评估工时

被误解的敏捷开发模式

错误的敏捷开发

- 敏捷开发就是强调一个快字
- 敏捷开发就是不断的压榨工时

- 敏捷开发就是不停的加班

正确的敏捷开发

- 测试在项目之初就介入，编写完测试用例之后，共享给开发，方便开发自测
- 将一个完整的项目进行合理拆分，拆分为若干独立小迭代
- 每个小迭代完成之后，进行提测以及收集用户试用反馈，尽早反馈，以及尽早发现问题
- 在小迭代提测期间，应该让开发略作修整（改bug或修整）和总结(总结共性问题，避免下阶段，再重复出现这些共性问题)，而非让开发立马进入下阶段开发，否则容易造成，开发一边赶下阶段需求，一边赶上阶段bug
- 个人认为敏捷开发，重点在于敏捷，灵巧好掉头，分阶段交付，及早发现问题，拥抱需求变化。而非简单的抽着鞭子让程序员加班赶工996或007