PRD 模版

To Gemini：注意，以下开始是我之前写的一篇文章《产品需求文档写作指南》，你可以以此文章为参考，检查写的 prd 是否能满足我的要求。你不需要每篇文章都严格按照我的要求，比较简单的需求可以适当省略一些内容。文章最后我会给你一个 prd 示例。

1.写作背景

需求文档是产品经理吃饭的家伙，讲怎么写需求文档的文章、视频也很多，但大多数内容仅仅局限在一个 prd 的格式是什么。我的理解是，不同情况下，需求文档的格式没有一定之规，但文档内容的标准是能够归纳出来的。本文章试图归纳一下一份好的需求文档的标准是什么。

2020 年左右，我参考了李守中的文章：https://www.douban.com/note/740671393/?_i=6051139rnZDWYs，输出一版：https://zhuanlan.zhihu.com/p/159106958。

2022 年初，我又抽空调整了一版框架，就是下文的 prd 自查工具。在实际应用过程中，仅凭一个框架还是不容易清晰表达我的意图，所以想抽出时间详细写写。

如前所述，好的 prd 标准并没有一定之规，本文仅是从个人角度梳理好的 prd 需要注意哪些要素，以及我是怎么写 prd 的。相比于“怎么写出来”，这篇文章我想着重梳理一下自己“为什么这么写”的理解。

在跟朋友的交流中，发现这个话题属于“高手不屑于写，新手写不明白”的范围，我也试图取个巧偷个鸡。另外也说明两点，其一，prd 写得好不好属于产品经理的基本功之一，但是如果一个产品经理只会写 prd，显然不是一个靠谱的产品经理，我们仍然需要从结果来评价产品经理；其二，我虽然试图梳理这个话题，但并不是说我本人是一个写 prd 多牛逼，这篇文章也是自己的反思和走查表，也希望有缘人多多拍砖，批评指正。

2.需求文档重要吗?

非常重要。

需求文档是研发流程中最重要的凭据。

首先，**需求文档描述了这个需求的原始说明**。它的场景是什么，它要解决什么问题，它用何种方式解决，它期望的解决效果是什么。后置的技术文档、测试用例、代码实现等，均依赖原始的需求文档进行生产。

其次，**需求文档是团队各方沟通的依据**。需求方、设计、研发、测试、运营等每一个环节，都可以借助需求文档同步思路，确认各方所构思的东西是同一个东西，确保需求执行中的所有改动能有效、准确同步给各个相关方。

再次，**需求文档是产品经理直接产出的“产品”**。从经验上来判断，需求文档写得好的产品经理一般基本功不会太差。好的需求文档也是对产品经理的一种评价标准。

最后，虽然大多数需求是在需求文档撰写前就已经规划好的。但是撰写过程并非仅仅把讨论中的内容结构化写在纸上这么简单，**写文档是对需求的深度思考的过程**，写文档也是产品经理自己思维训练的过程。

3.需求文档模块说明

3.1 需求文档的模块是固定的吗?

是也不是。

固定的，为了准确说明一个需求，目标、背景、相关方、前端交互、后端逻辑、埋点方案等各个模块是需要齐全的，任何一块缺少描述都可能会导致需求实现过程中的变形。

不固定的，不同的交付时间要求、不同的团队风格、不同的业务阶段可能对需求的“容忍度”是不一样的。例如上午收到需求，中午评审，明天上线的场景；例如固定产研班子的长期合作关系下，有时后台需求直接用字段表来说明即可，不需要描述后台的交互。

总结成好文档的必要条件是——**好文档能得到团队中各个角色的认可，且不会因为文档问题导致实现变形**。

评价需求文档要回归文档的本质——**需求文档是产品经理撰写的、供团队内不同角色使用的、保证合理需求稳定实现的工具。**

所以当我看到有很多 PRD 写作指南给出一个规范模板的时候，都感觉不太赞同——不同的团队、不同的业务阶段，对 PRD 的格式要求是不同的。

3.2 需求文档需要有哪些模块?

模块	作用	子模块
需求背景	研发不直接实现的、为了便于相关方理解需求的内容	根据需求类型、复杂程度不同，可能会包含以下子模块： – 一句话描述需求 – 需求目标 – 用户诉求 – 业务诉求 – ROI 评估 – 名词解释 – 竞品分析 …… ※ 子模块放在需求背景里面，还是单独拆出作为一级标题，全凭个人喜好
需求概述	在讲解需求细节前，能一眼全览需求全流程、相关流程	根据需求类型、复杂程度不同，可能会包含以下子模块： – **用户端交互流转图**：用线框图 + 箭头 + 条件，直观表示相关页面之间的跳转关系，并标明各个页面的名称 – **后端流程图**：用流程

		图：用流程图或泳道图的形式，表示相关后端逻辑框架；个人习惯总流程图中不展开说明子流程、逆向、异常流程，仅标注出哪些场景下进入子流程，并在需求详述里面描述子流程细节 – **其他全局视角的图表**：产品框架图（模块之间的相互关系、迭代节奏）、功能清单或脑图、用户旅程图等
需求详述	按照从总到分、从主流程到边缘流程、分场景逐一描述各个需求模块	详述基本是 PRD 最重要的部分，下文详细讨论需求详述的标准
埋点方案	涉及到埋点需求的文档，需要说明关注哪些行为、关注哪些数据	一般单独作为一个模块，也有整合到需求详述中的习惯
文档记录	文档推进过程中的	– **变更记录**：变更时

	所有记录	间、变更人、变更内容 – **相关方**：哪些团队、哪些人 – **协作节奏**：谁、什么时间、交付什么 ……

4.需求背景怎么写

4.1 一句话简介

在我看需求文档的时候，往往很关注产品经理能否一句话把这个需求讲清楚。能讲清楚的人，一般对需求的思考也是相对全面的。

一般来说，这一句话分成三个小句，**为什么，怎么做，有什么效果。**

当然约束条件越多，这句话可能相对复杂一点。无论怎么复杂，都尽量清晰、直白地把这个需求表述出来。

这句话最重要的功能在于向不清楚的人讲解需求的时候，他们能最快时间了解到你要做的事情。

既然这个模块是为便于理解服务的，所以最好语言简洁、直白，描述具体，尽可能不用大词、生僻词语。

> **好的例子**（仅从表述角度是好的）：
> 为了降低用户在主动搜索时的决策成本，基于 XX 数据生成推荐标签，期望搜索结果点击率提升 XX%

> **坏的例子：**
> 为了提升用户体验（什么体验？不具体），设计推荐特征模块（难理解），能满足用户快、准、好的核心需求（目标非量化、不说人话）

4.2 需求目标

需求目标用来详细解释一句话简介中的“为什么”部分。

（1）务虚目标

务虚的需求目标可以是基于用户洞察、基于业务需求，但最好量化出来。务虚目标一般用于未上线的产品、数据分析困难的产品。

> **好的例子：**
> 经用户调研（N=2000)得知，影响用户购买决策的 top3 因素有 A（30%）/B（20%）/C（10%），本需求用来强化 A 因素在商品详情页的露出，进而刺激用户的购买意愿。

> **坏的例子：**
> 本需求用多种手段刺激用户的购买意愿，提高购买率。

很多情况下，务虚的目标不容易量化。我回顾了一下自己做的需求，大多数务虚目标都是比较难量化的。我的习惯是保证逻辑论证是严密的、有据可依的，大多时候没有量化的指标。

如果需要量化，可以参考费米预估，大问题拆成小问题，在安全范围中取值。

（2）务实目标

一般来说，已上线产品、较稳定产品，是有一些方法预估出数据指标的。把新需求作为输入动作，来预估输出值的变化。

务实的量化指标是通过落地措施一层一层正向推导出来的，不是根据更宏观的指标逆向拆分出来的。

> **好的例子：**
> 目前完课率 80%。分析未完课用户中，10% 的用户存在 XX 问题。本需求通过优化 XX 问题，预计完课率提升至 85%。

> **坏的例子：**
> 为了达成最终 8000w 的销售目标，还差 2000w。本需求需要贡献其中 1000w 的目标。

产品工作必然会接受很多老板需求或其他团队的需求，这种情况下执行同学很容易不深度思考需求的目标。我很坚持无论需求来源是什么，负责的产品经理应该客观、深刻分析需求目标，并如实写出来。哪怕分析的结果偏差很大，分析过程本身也是有价值的。

在日常工作中，我也观察到一些人不太重视需求目标的思考。务虚的目标如果发生偏离，可能会导致后续产品设计重点跑偏的现象。务实的目标如果发生偏离，可能会本末倒置或者为了指标动作变形。

需求目标可能很简短，但是短短几句话，也容易看出来产品经理对这个需求要解决的问题思考的是不是深入，甚至预估到未来一系列需求变形的可能性大不大。

4.3 用户诉求

用户诉求用来描述这个需求满足了哪些用户诉求。

需要包括两部分，**哪些用户，什么诉求。**

用户诉求模块，需要注意解释**为什么是这些用户，为什么是这些诉求**。一般来说，这里需要有一些数据分析、用研结论或者内容盘点结论的支持。

这回到产品决策的话题，决策应该基于事实做出，而不应该基于二手信息或者凭感觉做出。

> **好的例子：**
> 社区存量用户中，XX% 是 XX 群体。盘点他们发布内容的习惯中，A 类内容占比 XX，B 类内容占比 XX，C 类内容占比 XX。其中 B 类内容的曝光量显著低于其他类，互动率显著高于其他类。结合用户访谈结论，XX% 的用户对 B 类内容有消费需求。
> 需要激励 B 类内容创作者，并合理提高 B 类内容的分发流量。

> **坏的例子：**
> 满足用户消费 B 类内容的需求。

4.4 业务诉求

有时一个需求不完全是因为用户需求发起的，也有可能是纯业务需求。

一般业务诉求的描述没那么严格，直白、可读性高即可。

我习惯把业务需求和用户需求分开写，这样有助于确认几个信息：

（1）在满足业务需求的前提下，对当前用户的体验是否有附加价值

（2）满足了业务需求，是否会对用户体验产生负面影响

（3）业务需求还有没有其他满足途径

4.5 ROI 评估

其实经过上面需求目标、用户诉求、业务诉求的分析之后，稍简单一点的需求就已经完成了 ROI 分析的过程。

如果本需求影响的功能是某个复杂系统的子集，最好在系统角度衡量这个需求的 ROI。

我觉得万变不离其宗，就是下面这个公式：

净利润 = 毛利 – 固定成本 = 收入 – 变动成本 – 固定成本

交易类型的产品好算一些，都能换算成价值；非交易产品可能要归到自己的北极星指标上。

同时，很多需求的 ROI 可能是不明确的，也有可能是明确负的。但出于信仰判断，有时这种需求也有做的价值。关于信仰判断怎么做不展开分析，应该能搜到比较成熟的方法论。

4.6 名词解释

如果这个需求引入了新的专用名词，需要对专用名词的称呼进行限定，对名词进行解释，以免出现本需求及后续需求称呼来回变化、指代不明确的现象。

同时在产品设计时，也一定要保证**每一个名词是用户能理解的，对于同一对象的称呼应该是始终统一的**。

这个模块不是必须的，简单需求、团队内明显约定俗成的称呼一般不需要在文档中单独说明。

哪些名词需要解释的判断标准是，一个第一天入职的同事能否看明白你的文档。

除了简单的名字解释之外，也可以在名词解释部分说明可能有的业务限制，帮助大家对边界情况进行判断。

> **好的例子：**
> 豆邮：豆瓣站内用户与用户之间、平台与用户之间相互发送的消息系统，相关需求均以“豆邮”指代豆瓣站内消息。
> 服务供应商：提供安装服务的第三方公司，一个供应商会服务多个省份，同一个省份也会有多个供应商；当前业务预计会接入 6 家左右的供应商。

> **坏的例子：**
> 豆邮：相关需求里面，有时写“豆邮”，有时写“消息”，有时写“站内信”。
> 服务供应商：文档中服务商、供应商、安装公司等名词混用。

4.7 竞品分析

竞品分析是一个复杂的话题，展开聊充分，可能需要巨长篇幅的文章。

我从观察到的，容易流于表面的竞品分析入手，提炼竞品分析的几个核心注意事项有：

- 带着问题
- 把点连成面
- 考虑不同的视角

带着问题进行竞品分析是必要条件。

我们不知道靶子，自然不知道向何处射箭。市面上能搜到很多竞品分析的文章往往是**为了分析而分析**，先比较行业数据、产品数据(a)，再列一张脑图(b)，列出来某个 app 的前端功能点，然后截几张截图，说说自己对截图中功能点的看法(c)，最后列出一堆建议(d)。但实际上，工作实践中我们几乎不会进行这种格式的竞品分析，这种竞品分析也很难得出我们期望的洞察和结论。

没有目标的竞品分析难以得出洞察的原因就是不聚焦。

a.行业和产品数据是大面数据，要对这个数据进行调优，可能依靠行业竞争格局、行业背景发展、业务的供给和消费变化。相比之下，产品功能的改动几乎是最末尾的动作。

b.产品架构图，相比 C 端功能点枚举，我们也需要考虑到后端和策略的部分，枚举出来全集之后，再按照场景归纳成几条产品线，例如抖音 app 为了扶持创作者做了哪些前后台的功能。一条产品线中的产品功能点并非在物理上挨得很近，比如上面抖音的例子，既要考虑到信息流的流量扶持策略，也要关注到创作者中心里面的功能点。如果仅仅按照功能点的页面结构对 C 端功能进行梳理，结构化的深度是有限的，也不容易获得洞察。

c.没有对业务线的认知，仅凭几张截图其实很难评价某个页面设计的好坏。再加上页面的迭代周期、当前目标等多种因素不同，同一个样式的页面，在 A 产品内可能就是个巧妙的解决方案，在 B 产品内可能就没那么优雅。

d.缺乏上面的洞察，最终给产品提一堆建议，也大概率会流于表面，起不到太多业务价值。

上面说的是反面的例子，那么怎么带着问题进行竞品分析呢?

如果是工作中的场景，问题是显而易见的，基本就是我们工作中要设计的需求对应的问题。

如果是出于兴趣的竞品分析，可以按照**“哪些用户”+“什么场景下”+“遇到什么问题”+“产品是怎么解决的”**这个公式提炼自己的问题。问题越具体，越容易得出有价值的洞察。

一些好的竞品分析问题：

- 新品类的消费受众小，优秀创作者少，抖音从产品层面是怎么扶持新品类的新创作者的?
- 小红书一篇爆款 UGC 文章从发布到互动量爆炸，会经历哪些前后台的流程，会归纳成哪几个阶段?
- 热播剧结束之后，腾讯视频是怎么吸引用户继续使用本 app 的?

把点连成面，指的就是结构化。

竞品分析最基础的工作是枚举信息量，但单纯的信息量几乎起不了什么业务价值。我们知道苹果发展史，但是我们做不出来苹果。比如我梳理了小红书 app 所有页面的名字和类型，但拿这些信息能做什么，其实挺迷茫的。

把信息量通过某种方式结构化梳理出来，这就形成了知识。如果你有小红书 app 所有页面的名字，你能否按照用户和场景不同，把他们分类一下？如果某一类的新功能点比较多，是不是就表示这可能是小红书的核心模块或者正在发力的模块？

把知识再抽象、结构化，提炼出来他们共同的特性，这就成了方法论。假如我们发现小红书围绕图文、视频两类内容，虽然功能点不一样，但是某些功能组合及运营套路相近，就有可能归纳出来他们对不同类内容的扶持方式（当然这里就不能仅局限在产品分析了，需要运营视角）。这就是小红书内容方法论。

信息量是难以单独产生价值的，产生价值的是方法论。通过对竞品的归纳分析，总结竞品方法论，并放到本产品的场景中进行思考，进而找到可以借鉴的点。

考虑不同视角。

第一个维度前面提到了，产品的前后端视角。

有很多流量策略、分发逻辑、推荐策略、冷启策略等，并不一定直观呈现在 C 端产品上，只聚焦某个页面长啥样，分析肯定是不全面的。

另一个角度，通过对业务具体场景的结构化抽象，形成的通用的、中台性质的功能，也是很有分析学习的必要的。

第二个维度，产品、运营、财务、品牌的视角。

拿内容产品来说，除了用户价值以外，上面提到的这些因素都是会影响产品设计的维度。

例如最右，信息流的样式设计这种产品问题，跟最右内容极度依赖 pgc 内容采买是互为因果、息息相关的。

再比如很多免费工具产品，刻意强调它们广告太多了，在我看来是有失偏颇的竞品分析结论，因为免费工具 app 的变现手段极为有限，总不能让他们活不下去吧。

第三个维度，时间视角。

当前是产品刚出现的 mvp 阶段，还是追求用户量增长的阶段，还是追求利润率的阶段？每个阶段的打法和重点是不一样的。有些竞品分析里面会提到 XX 产品如果有 XXX 功能就好了，可能确实是加分项，但那不一定是该产品当前最生死攸关的问题。

最后说说竞品分析这个模块的格式。

一般是结论先行，结论就是我们看竞品时，心里带着的那个问题的答案，根据竞品的实践抽象出来的方法论。

结论过后是直接你结论的论据。可能竞品分析的时候，我们收集了 100 条信息，此处只要展示出来足够能说明你结论的那几条即可。

所以竞品分析不一定篇幅很长，竞品分析不需要面面俱到，只要能解答你心中的那个问题即可。

5.需求概述怎么写?

5.1 需求概述是什么?

需求概述是写在需求详细描述前，用最概括的方式让大家看到需求全景的模块。

为什么要写需求概述呢?

对于简单的需求，按照有逻辑的顺序依次说明功能即可，但是对于有一定复杂度的需求、涉及到前后端多端交互的需求、用户流程较长的需求，如果没有需求概述，大家在看需求详情的时候不容易理解每个模块之间的关系是怎么样的。

对于不同的需求类型，需求概述的角度是由不同的：

流程复杂 C 端需求，可以用一张图表示清楚各个 C 端界面的流转情况，并给每个界面标号，方便概述和详述对照看。

模块复杂的 C 端界面，可以用一张图详细说明当前页面的所有模块，以及每个模块可能出现的样式，相当于是描述了全集，这样详述的时候就可以按照场景或者逻辑详细描述了。

复杂后端需求，可以用一张图表示后端各个模块之间的交互情况，并且对于有 C 端界面的环节进行标记，方便前后端一起理解

对已有功能的修改需求，需要把现有的完整流程画出来，然后标识出本次修改了哪个部分的流程。

根据需求复杂度的不同，需求概述一般是上面四种情况的任意组合形式。

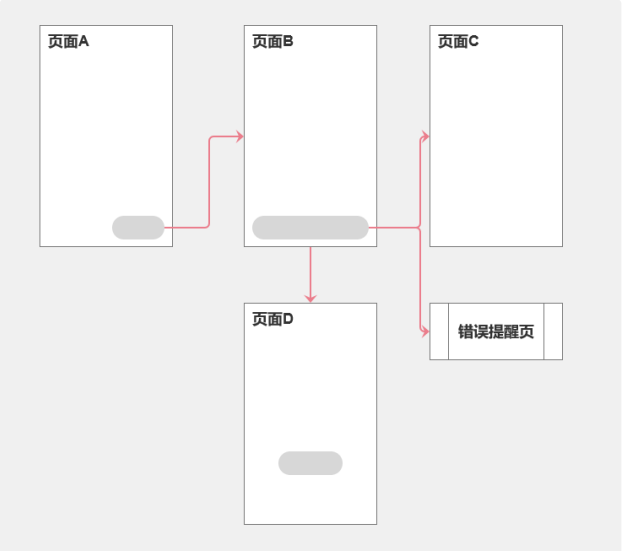
5.2 怎么一图流讲清楚前端交互

第一种情况是前端的复杂流程。

指的是包含 N 个页面，页面与页面之间有相对复杂的跳转逻辑。

我的习惯是用一张图画出来 C 端的主流程，然后用文字或流程模块标识出边界情况。

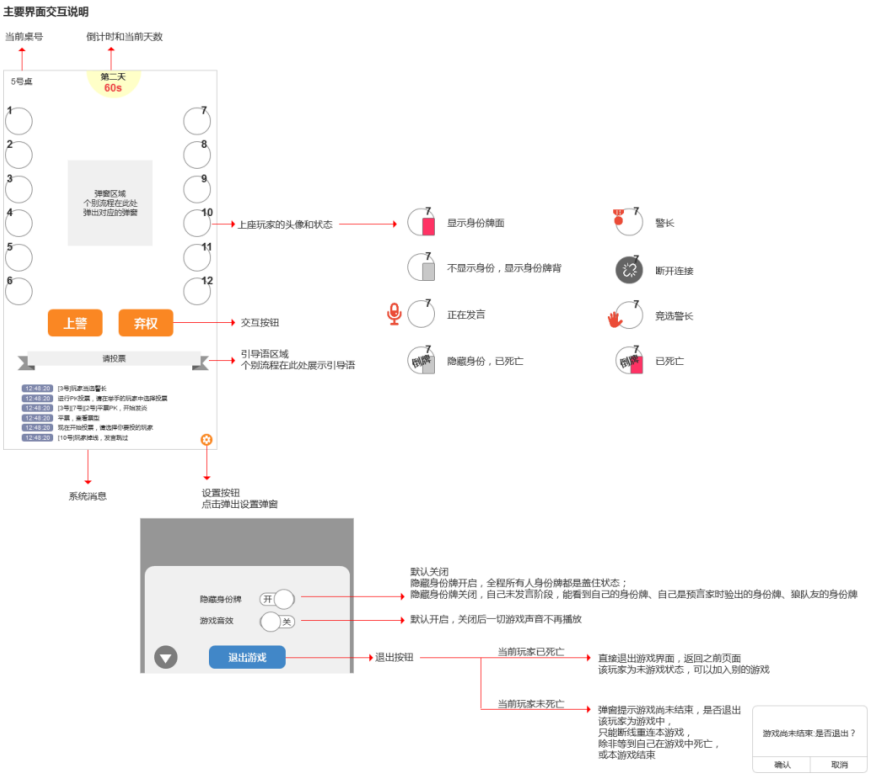
例如：



第二种情况是前端的复杂界面。

指的是单页面，但是单页面的元素比较多，每个元素在不同的场景下会有不同的样式。

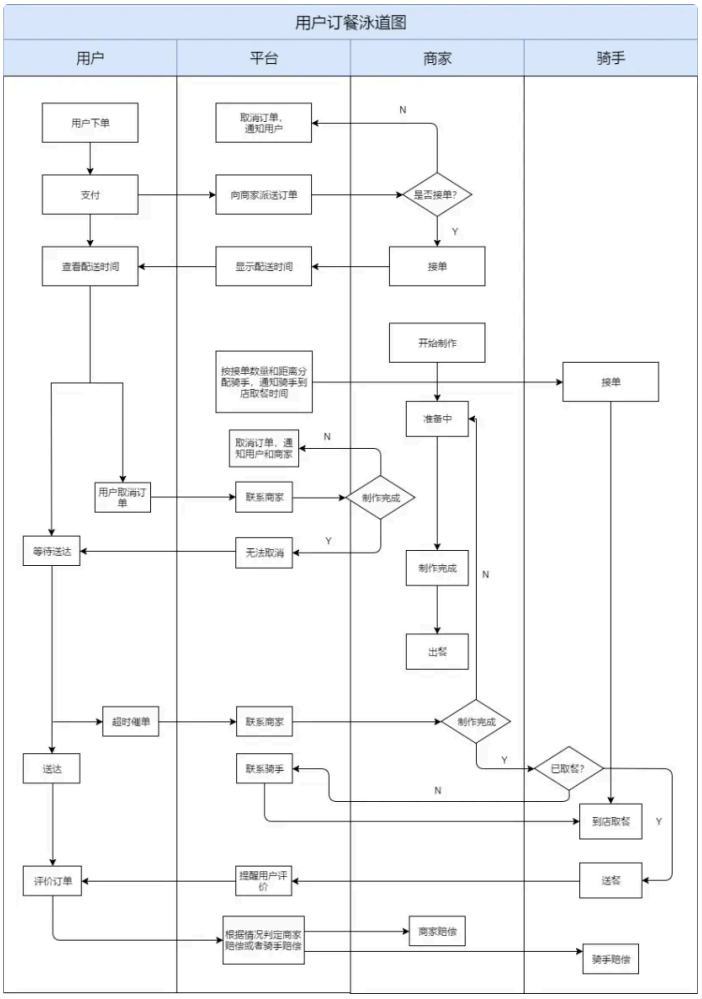
例如：



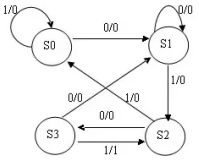
5.3 怎么一图流讲清楚后端交互

一般说明后端交互会有几种形式：

第一种是说明**不同模块之间的交互关系**，可以用泳道图的方式来描述。



第二种是**某个属性的复杂变化**，一般用状态机来描述。

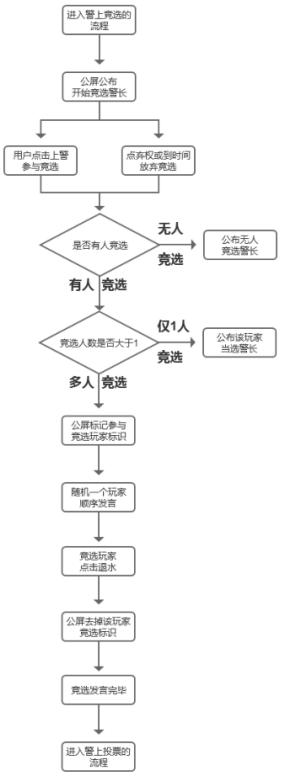


第三种是**业务流程在不同情况下的走向**，用流程图来描述。

流程图有一个起点，一个终点，详细画法可以参考：

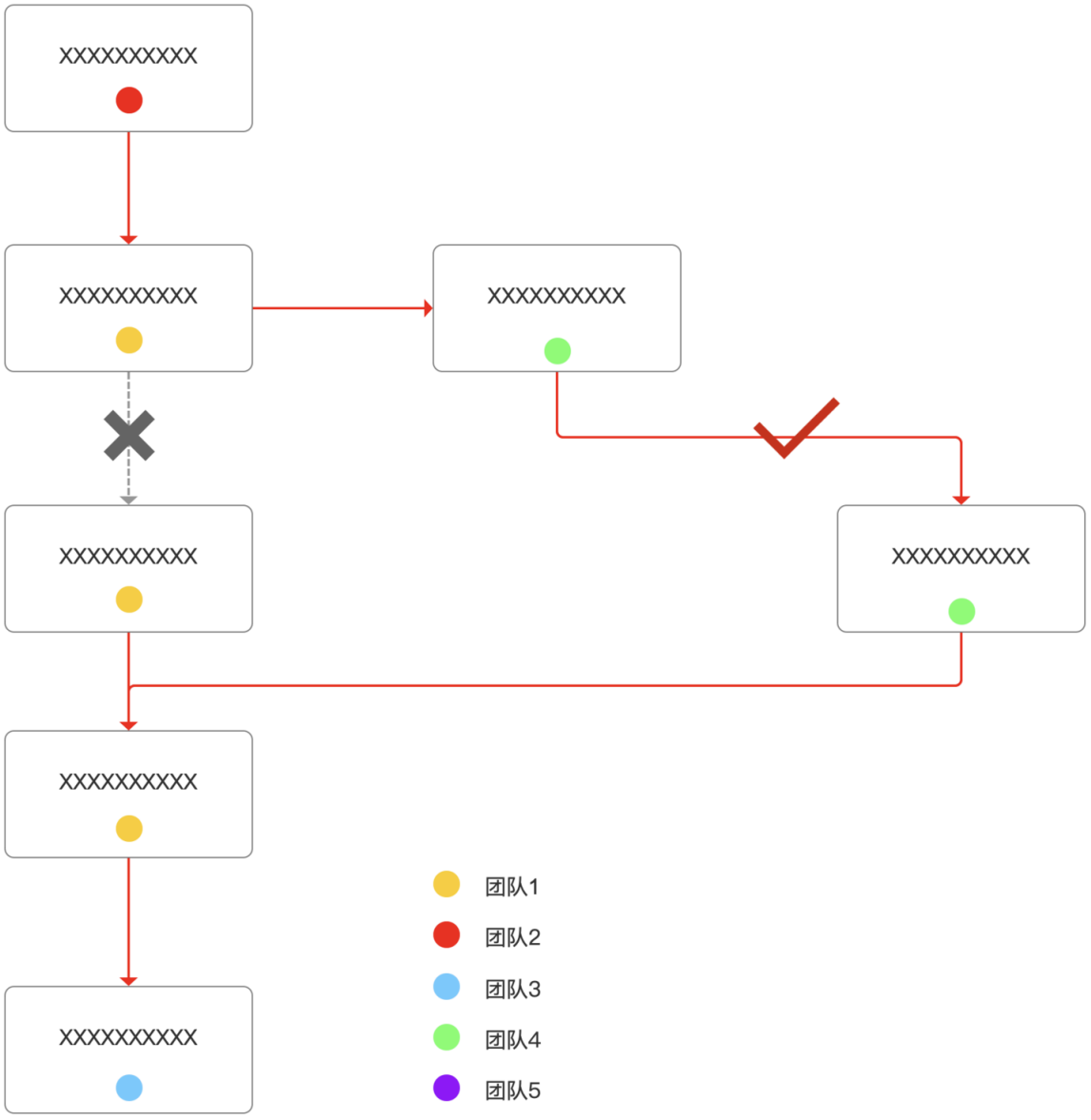
https://www.cnblogs.com/lukelook/p/11217031.html

实际的需求文档中，其实不一定遵循标准画法，大家容易理解即可。



5.4 还有哪些需要说明的?

如果需求涉及的参与方比较多，可以在文档中说明相关参与方、负责的模块、交互顺序、交付物、时间点等信息，用于让大家了解整体项目节奏。



需求模块	相关方	描述	负责人	时间点
A	a	XXX	aa	什么时间点前，完成什么交付物
B	b	XXX	bb	
C	c	XXX	cc	

6.需求详述怎么写?

6.1 需求详述的作用是什么?

还用说嘛?

需求详述就是需求文档的本体，是需求的说明书。

关注需求详述的人有谁呢?

- **研发和测试团队**。所以需求详述除了要按照用户视角描述需求以外，也要尽可能的按照研发视角展示产品经理对这个需求前后端结构化的抽象思考。同时，为了保证抽象化的语言是容易理解、没有偏差的，我一般还会把具体的一些边界用例写在星号后面，来具体展示实际场景中，我们设计的抽象逻辑是怎么应用的。
- **设计团队**。所以除了需求的功能实现要讲清楚以外，对设计有明确依赖的，要写明设计意图或设计倾向。例如：此处建议使用滑动动效，但具体效果以 UE 设计稿为准。对于看到的好的设计、好的交互，如果产品经理能讲明白为什么要这样设计，也可以把具体的例子放到文档里。
- **合作伙伴或后来人**。需求除了实现的时候会看以外，可能你的继任者、你的合作伙伴也会翻你的文档。为了保证理解门槛足够低，自己的需求文档尽量形成固定的写作风格，并且保证自己文档中的各种名词、代号是统一的。

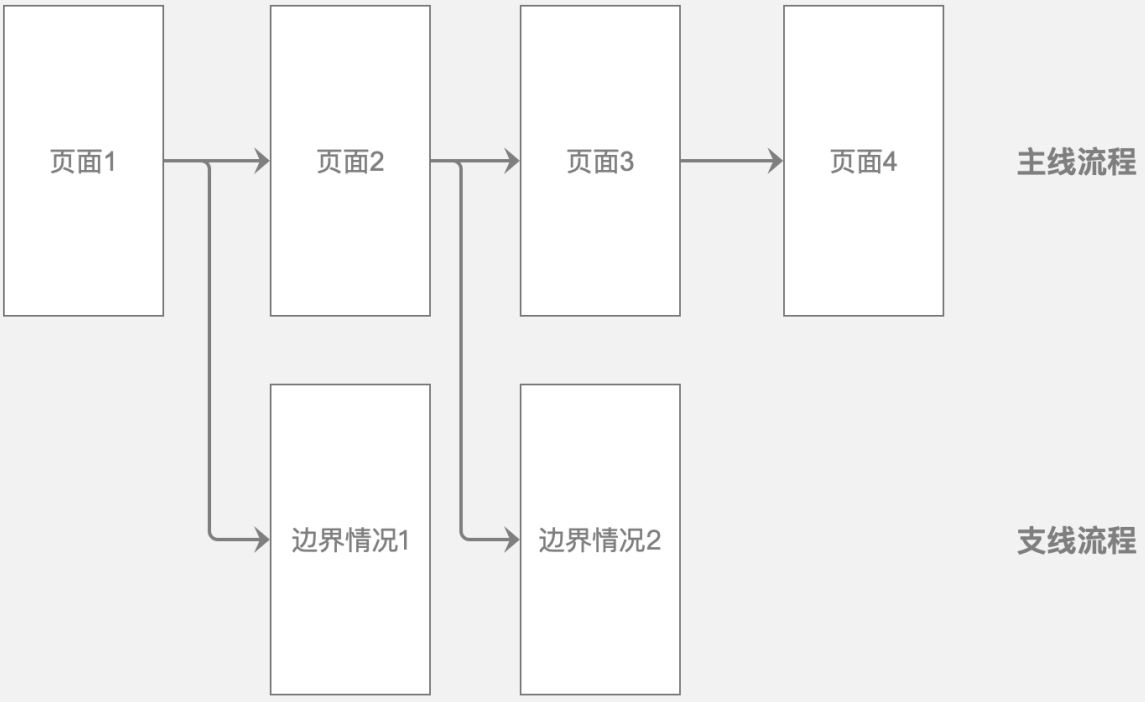
6.2 按照什么样的顺序讲需求模块?

参考金字塔原理，需求模块的讲解顺序，可以参考下面几个顺序：

按照**总分关系，递进讲解**，一般适用于逻辑结构复杂，一个场景对应 N 个下游的需求，例如淘宝首页的设计

按照**用户旅程，顺序讲解**，一般是用于用户流程较长，但是有明确的主线和支线之分的需求，例如从商详页下单到成功支付的需求

按照用户旅程



先按照用户旅程，逐一说明主流程的各个页面：页面1、页面2、页面3、页面4

再按照先后顺序，逐一说明支线流程：边界情况1、边界情况2

按照**模块结构，逐个讲解**，一般是流程不长、模块与模块之间相对独立的需求，例如商品上下架的前后台需求

按照模块关系

按照模块之间的关系，逐一说明各个模块
如上图的例子，可以先说明C端页面的展示逻辑，
再说明商品的展示策略配置，
再说明商品基础信息的管理后台
也可以反向说明。

根据不同的需求类型，选择最容易理解的讲解顺序。

6.3 怎么把一个需求模块讲清楚?

单个需求模块的讲解也需求遵循**金字塔原理，从主到次，从总到分**。

一般我的习惯是这么讲解——

模块概述：一句话讲明白这个模块是什么

出现条件/进入条件/上级页面：说明什么情况下，这个模块会出现

描述逐个元素：

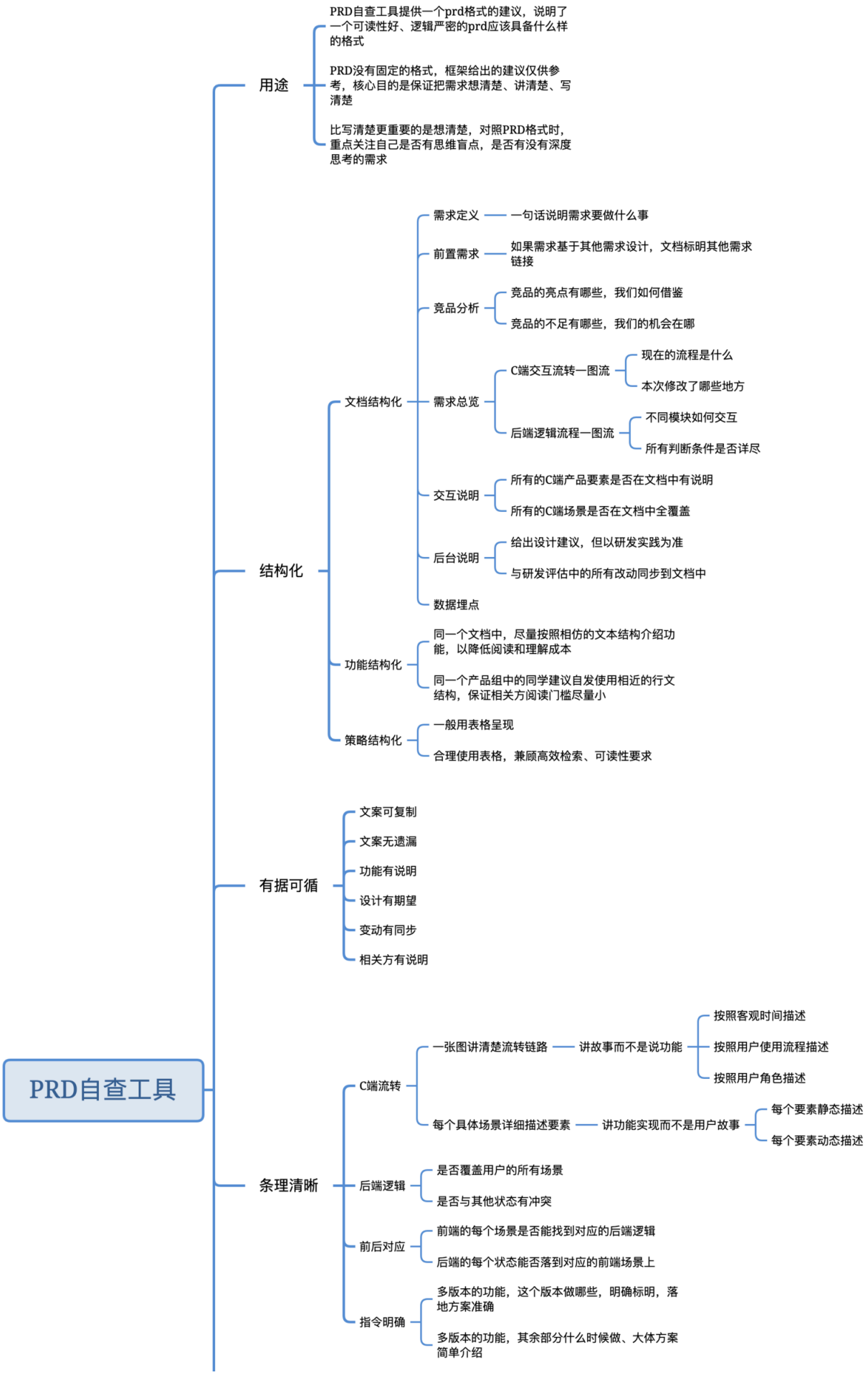
- 文案是什么
- 类型是仅展示、可交互还是输入组件
- 交互效果是什么
- 下级页面是什么
- 对应的后端交互是什么

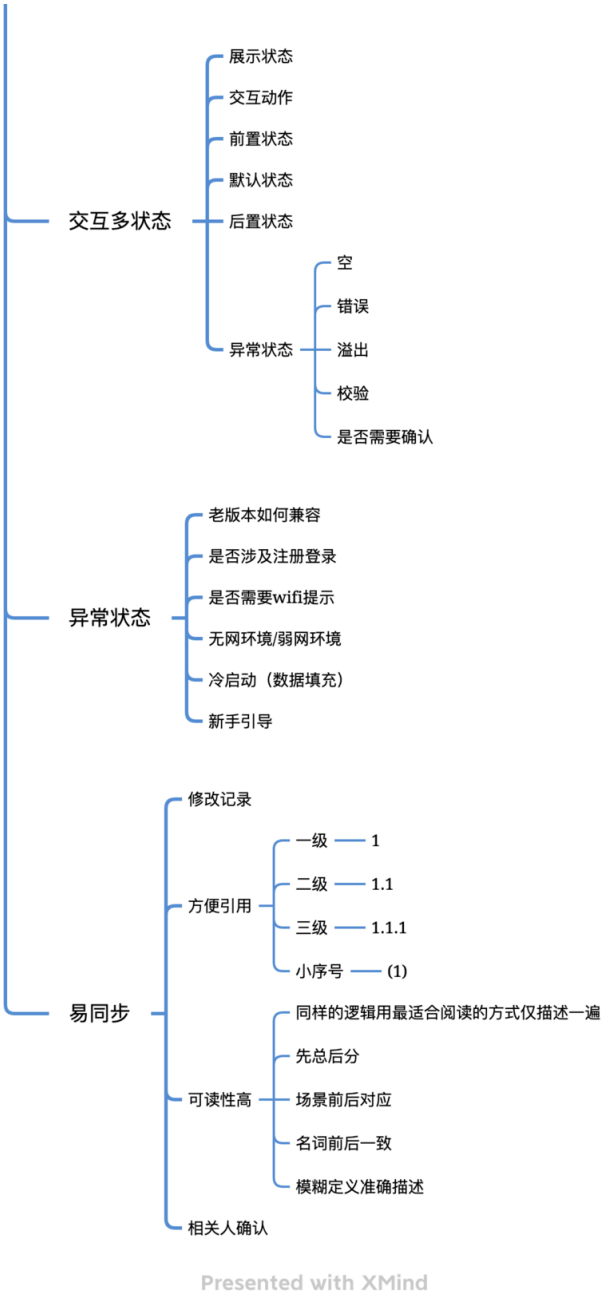
特殊情况：这个模块会遇到的边界情况有哪些，需要怎么处理

> 举个例子，说明淘宝的确认购买按钮：

6.4 还有漏的吗?

为了防止需求描述有遗漏，我整理了一个 prd 自查工具，自己写文档的时候用来对照查看自己有没有遗漏的地方。


PRD自查工具
用途
PRD自查工具提供一个prd格式的建议，说明了一个可读性好、逻辑严密的prd应该具备什么样的格式
PRD没有固定的格式，框架给出的建议仅供参考，核心目的是保证把需求想清楚、讲清楚、写清楚
比写清楚更重要的是想清楚，对照PRD格式时，重点关注自己是否有思维盲点，是否有没有深度思考的需求
结构化
文档结构化
需求定义
一句话说明需求要做什么事
前置需求
如果需求基于其他需求设计，文档标明其他需求链接
竞品分析
竞品的亮点有哪些，我们如何借鉴
竞品的不足有哪些，我们的机会在哪
需求总览
C端交互流转一图流
现在的流程是什么
本次修改了哪些地方
后端逻辑流程一图流
不同模块如何交互
所有判断条件是否详尽
交互说明
所有的C端产品要素是否在文档中有说明
所有的C端场景是否在文档中全覆盖
后台说明
给出设计建议，但以研发实践为准
与研发评估中的所有改动同步到文档中
数据埋点
功能结构化
同一个文档中，尽量按照相仿的文本结构介绍功能，以降低阅读和理解成本
同一个产品组中的同学建议自发使用相近的行文结构，保证相关方阅读门槛尽量小
策略结构化
一般用表格呈现
合理使用表格，兼顾高效检索、可读性要求
有据可循
文案可复制
文案无遗漏
功能有说明
设计有期望
变动有同步
相关方有说明
条理清晰
C端流转
一张图讲清楚流转链路
讲故事而不是说功能
按照客观时间描述
按照用户使用流程描述
按照用户角色描述
每个具体场景详细描述要素
讲功能实现而不是用户故事
每个要素静态描述
每个要素动态描述
后端逻辑
是否覆盖用户的所有场景
是否与其他状态有冲突
前后对应
前端的每个场景是否能找到对应的后端逻辑
后端的每个状态能否落到对应的前端场景上
指令明确
多版本的功能，这个版本做哪些，明确标明，落地方案准确
多版本的功能，其余部分什么时候做、大体方案简单介绍



7.prd 有没有固定的写作格式?

我是有自己的写作习惯、写作格式的，但总感觉“写作格式”太死板了，我更喜欢称其为“结构严谨”。

结构严谨有三个作用：

第一个作用，如果你的 prd 是结构化的，形成肌肉记忆之后，每次撰写文档时你不需要花太多精力在文档格式的排版上，可以**释放出来精力到需求合理性、方案合理性的思考**上。让你的手更快，效率更高。

第二个作用，显而易见的可读性更高，**沟通成本更低**。适当结构化的内容大概率比未结构化的内容更容易理解、记忆。

第三个作用，**形成个人风格**。结构严谨、风格一致的文档会成为你的职业标签，有助于你形成专业的标签。

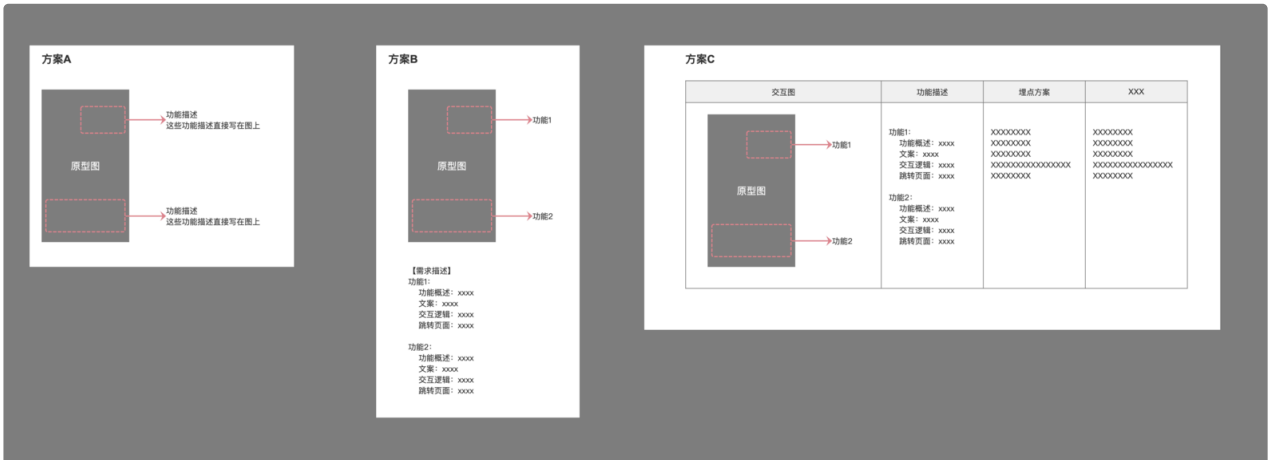
怎么写一个结构严谨的文档呢?

第一个是**模块层面的结构化**，就是本文之前讲的那些。形成肌肉记忆后，你每次写文档就会不自觉反思所有模块是否齐全，少了很多错漏风险。

第二个是**文本格式层面的结构化**。但这没有一定之规。我的习惯是大的格式保持一致，小的格式就事论事。

大的格式例如本文，不同层级的标题严格按照 1.一级标题 1.1 二级标题 1.1.1 三级标题来划分，并习惯性地处理好首行缩进。

小的格式展开聊聊，比如一个功能的描述，我见过的有三种类型：



方案	描述
A	以交互图为主，把功能描述直接标注在交互图上
B	开局一张图，标注功能位置 下方用文本详细描述功能逻辑
C	列一个大表格，第一列是交互图，按照模块划分，后面每一列放对应的说明

这三种方案有对错之分吗？三种方案我都用过，我认为在不同的场景下，对三个方案的评价是不一样的。

既然 prd 是我们产品经理自己的产品，我们要分析一下这个产品的用户是谁，使用场景是什么，有哪些需求。

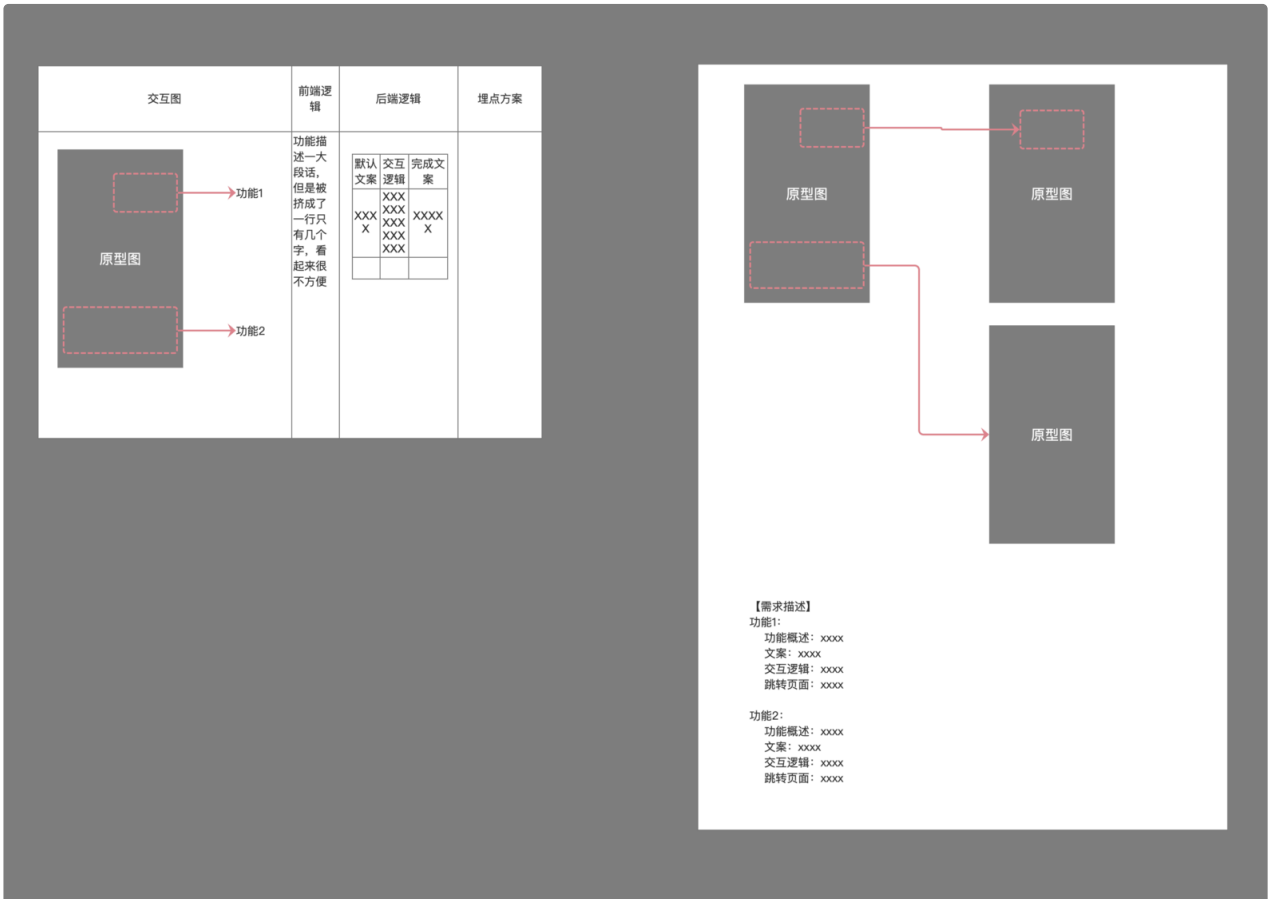
> 核心用户：研发、测试、设计师
> 场景：需求评审前默读、需求评审、实现需求过程中
> 需求：逻辑清晰、表意明确、阅读效率高

对于方案 A，如果你的需求本身工作量非常小，逻辑很简单，或者工期特别特别紧，合作团队配合特别特别紧密，实际上可以采用 A 方案，如果一些异常情况、边界情况已经提前定义好了，例如整个 app 都有一个统一的断网提醒，你可能就不需要详细描述这个页面加载的时候断网了，用什么容器、什么文案、什么时长显示断网提醒。尽管如此，我其实不太推荐 A，能详细描述的，还是要保证每个要素都描述过。

对于方案 B 和 C，我认为主要的差别在于可读性上。C 显然结构化更强，每个模块都是分开描述的。比如一列写前端交互，一列写后端逻辑，一列写文案，一列写埋点。但是我自己的经验是，**很多需求场景下，用方案 C 来写，对可读性反而是减分的**。

第一个场景是功能逻辑比较复杂，比如你这个页面有 10 个按钮，每个按钮有 5 种加载状态，每种加载状态又有对应的完成状态。对于这些按钮状态和文案的描述显然用表格来呈现最清晰。但由于已经有一个大表格来区分模块了，所以需要大表格里面容纳一个小表格，甚至由于屏幕宽度的关系，小表格的一列会变得非常窄，一行只有两三个字，可读性非常差。

第二个场景是跨页面的、跨功能的交互，如果用方案 B 来描述，可能上面要配一张两个页面的交互流程图，下面综合描述两个页面的功能耦合关系。这种需求把每个页面分成一行，用方案 C 的方式来描述，也不太恰当。



总结就是，作为产品经理，你得了解你的团队（他们是你的用户），你要想想具体每个场景里用户们有什么需求，然后选择合适的结构形式。

8.怎么使用需求文档

8.1 评审前沟通

需求评审是做什么的？快速拉齐共识，评估需求的可实现性，确认需求有价值、可实现后，进入需求排期的阶段。

需求评审的常见问题也是对应的：**参与者对需求价值达不成共识、需求的实现方案有问题、排期时/撰写技术方案时/撰写测试用例时/实现时才发现之前没暴露的问题**。

好的需求文档应该有助于解决上面提到的问题。

针对需求价值的问题，这回到了上文对需求背景的讨论里。为什么我要花挺大篇幅讨论需求背景，因为对于有些人而言，其实只关注自己做的“活儿”是什么，不关注需求价值，但总有一些优秀的同学既要知道做什么，又要知道为什么做。所以为了满足各种用户的需求，需求背景不宜写的太长、太啰嗦，但又要把需求价值清晰地展示出来。

描述清楚需求价值，评审前就可以拿着文档找几个关键节点的人评估一下，包括产品线的上级、合作伙伴、研发等角色里面的 leader。

针对需求实现方案的问题，一方面是让自己评估方案是否可实现、实现难度有多大，第二方面是方便提前找人评估可实现性。结合需求自查表，如果你对需求有充分的全览，大概率会对需求复杂度有一定的理解。除了需求本身的结构清晰导致阅读成本小以外，需求目录等外部因素的结构清晰，也方便跟你合作的研发降低工作中的各种摩擦力。

另外，如果你的需求涉及安全、法务、风控等层面，需求文档写好了，正式评审之前，也应当跟相关同学进行确认，不要把问题留到需求评审会当场。

8.2 评审

评审时是你和你的需求文档的高光时刻，需求文档写的扎实不扎实，在评审时很容易分辨出来。

坏的需求文档，满屋子人眉头紧皱，沉默寡言，耗费大量的精力用来理解你要做什么事。

好的需求文档，当然有一遍过的情况，更多的情况是大家七嘴八舌，对实现方案进行讨论。

产品经理并不需要技术出身，技术实现角度考虑不周到几乎是必然事件，但结构清晰的需求文档能快速让参与的人理解你要做的需求是要解决什么问题（技术角度是不是还有更好的手段？）、你期望的实现效果是什么（技术角度能支持吗？）。

除此之外，也要考虑到需求评审过程的高效，核心思路还是**把研发团队当成自己的用户，从用户需求角度出发。**

比如我评审的时候，会把同一个文档打开到两个页面中，在屏幕上左右分屏展示，左边展示交互图的时候，右边可以展示文字描述的功能逻辑，方便大家对照查看。

比如我在把交互图从 axure 复制到文档中的时候，会给每个图片添加一个不透明的黑色或白色背景，有些文档工具点开大图后，会自动把透明背景处理成其他颜色或者棋盘格，会造成阅读障碍。

比如我在文档中添加一个图片之后，会手动调整图片缩略图的大小，保证不用点开大图，基本也能在一屏把图上的内容看清楚。

这些细节就不一一展开，团队风格不同，你要“贴心”地进行的细节操作也不同。

8.3 评审后

评审结束后，产品经理的工作就结束了吗？显然不是。

我了解有一些公司的流程会在评审结束后的某个时间点，把需求文档锁定，要求需求文档中不能修改。但我自己亲身经历的每一个团队，在需求评审后直到需求上线前，需求文档都会是动态变化的状态。

一种变化是**由于实现方案导致的需求变更**，比如做一个后台需求，翻页器可以调节每页显示多少行数据，之前需求文档中定义的是 20/50/100，研发实现的时候发现有个现成的组件可以复用，但那个组件是 10/50/100，这个时候一般就会把文档改成研发实现的方案。

第二种变化是**实现前难以预估的实现效果**，例如我自己做过的一个 web 端产品，之前设计期望的是网页背景有一个炫酷的流光动画，到了实现的时候发现实在搞不定性能问题，这个时候就会修改方案。

第三种变化是**与运营动作相关的变化**。比如这个功能需要导入 100 条数据才能跑起来，但运营节奏调整，前期只能给 20 条数据，对应的产品侧的变化也是需要体现在文档中的。

第四种是**推进节奏的标记同步**。比如产研以外的人，什么时间点交付什么材料（数据、信息、物料等），如果对实现或测试有影响，这些进展最好记录到需求文档中，帮助跟需求有关的所有人及时查看。并且就算已经写到了文档中，相关进展也必须在共同的群里或当面沟通同步下。

文档的修改要遵循团队共识的流程，是否有变更流程，没有变更流程的话也要在变更前后跟相关人及时沟通。

文档修改的痕迹要便于查看。我的一个习惯是每次修改都会用不一样的颜色把修改文字标出来，并且再文档头部的修改记录里标清楚本次修改是什么颜色的部分。我的另外一个习惯是如果文档工具本身支持锚点的话，我会在修改点的前一行增加一个锚点，并且在修改记录里逐条添加超链接关联到对应的锚点上。

8.4 上线后

上线之后，需求文档就算寿终正寝了吗?

大部分情况是的。

第一个情况是有一些优秀的团队会把上线后的复盘结论添加到文档上，或者文档的子目录上，这些都是团队的高价值知识沉淀。

第二个情况是涉及到新人入职、老人交接，阅读过去浩如烟海、格式各异的文档显然是一个极为痛苦的体验。如果工作量可以允许的情况下，一般某个项目进展到一个程度，我都会自己或让某个同学根据过往文档梳理出来一篇合辑，只写明需求的核心流程、后台的使用方法等，类似于一个需求的使用说明书。如果有跨部门的合作伙伴来咨询需求，可以直接用这个文档来进行沟通。同时，新人入职后，第一个任务是梳理出来这个文档的话，也容易帮他快速了解过往需求。

9.结尾

这篇文章我从没想到会写成这样。

写这篇文章的时候，我把它当成一份需求文档，先用 xmind 搭好框架，在文档工具里面写出来目录，提醒自己每几天写一段。

没想到，断断续续写了两个多月，有些环节也写的又臭又长。

写到一半的时候，我意识到这种写法并不遵循“精益创业”的做法，于是把每个章节都拆成片段，发到小红书上，想看看即时反馈。

果不其然，最干货的地方，是反响最强烈的，需求概述和需求详述两部分图文并茂、直接讲怎么做的部分，两天内收获了几百个赞。但其他模块我自言自语了许多“为什么”的，反响平平。

看来我还要修炼啊。

另外很讽刺的是，完稿日前一天我们组的需求还在评审会上别提了一堆问题，尽管我已经审核并通过了，还是有不少考虑不周到、设计有疏漏、提前未确认的点。

漫漫的修炼之路啊。

这篇文章我会持续开放评论，希望各路朋友评论批注，哪里讲的不对，哪里讲的不够细，哪里讲的太啰嗦，请不吝指正。

需求文档是产品经理吃饭的家伙事，我也希望能不断迭代这项技能。

To Gemini：注意，以下开始是 PRD 示例，这是我自己写 PRD 常用的格式，你可以参考我的格式、措辞、详略程度，以带格式的文本形式输出给我。如果其中需要有原型图，如果我已经上传给你原型图了，你只需要告诉我在哪里插入哪张图、对应的图片名称是什么即可；如果我没有上传给你原型图，你只需要描述页面里有哪些元素即可，不需要作图。以下例文开始：

1.**需求背景**

本文档描述 D2 项目相关云平台的基础前端框架，云平台及衍生 web 后台功能尽可能遵守本文档的前端框架，以减少重复设计和开发工作量。

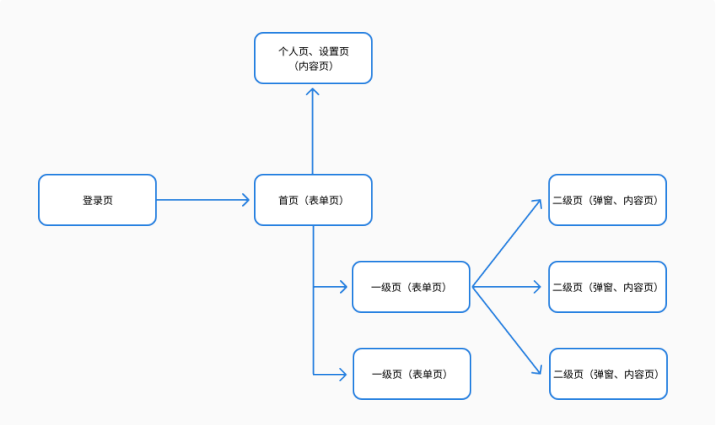
2.**前端框架参考**

D2 相关云平台尽量采用某个统一的前端设计框架，推荐字节的 arco.design：https://arco.design/

后续产品经理在定义后台功能时，需要翻阅 arco 的设计指南，确保自己的设计能够通过已有组件搭建实现；设计师尽量遵守 arco 提供的设计规范，如果需要出设计稿，可以更侧重布局、间距等定义，尽量使用 arco 的通用组件。

本文档基于 arco 撰写，如果更换为其他前端框架，会评审沟通后修改文档。

3.**云平台页面结构**



上图为简单示例，为了便于说明，云平台相关的页面分为四大类，下文将逐个说明每一类的通用交互原则。

- 登录页：登录、登出、修改密码等功能的承载页面，原则上未登录状态下访问云平台仅可访问登录页，未登录则无法触达含有具体内容的其他页面
- 表单页：首页或其他一级功能的承载页面，一般包含顶栏、侧边栏、筛选器和表单
- 内容页：查看、修改表单中详细内容的承载页面，一般包含顶栏、内容区
- 弹窗：查看、修改表单中详细内容的承载容器，非独立页面

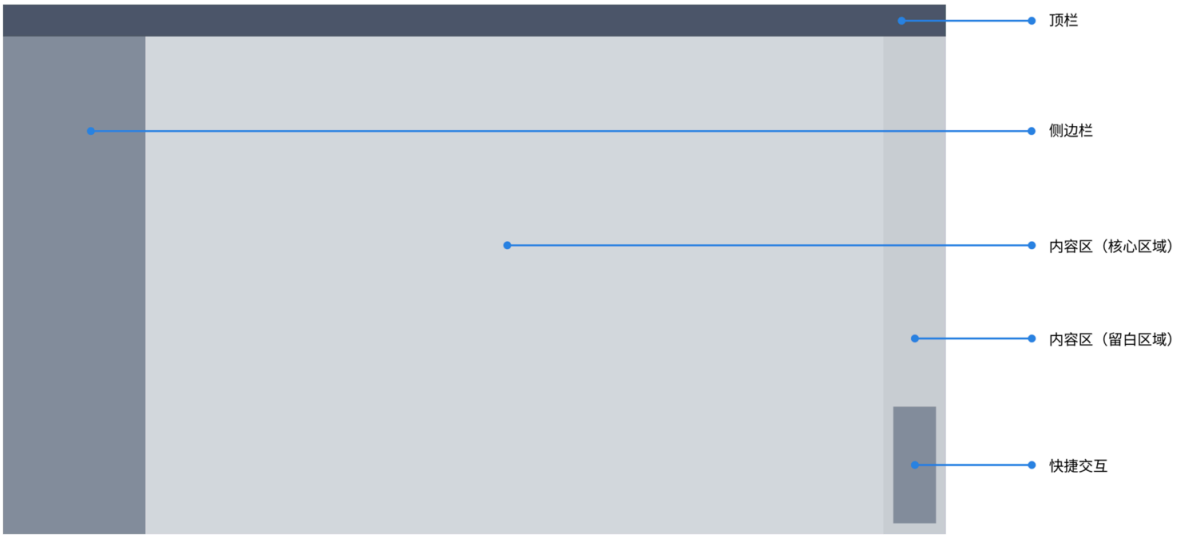
4.**交互原则**

4.1 登录页

相关页面：登录页、找回密码、修改密码……

功能描述：接入滴滴 IAM，包含权限系统、SSO 登录等功能

4.2 表单页

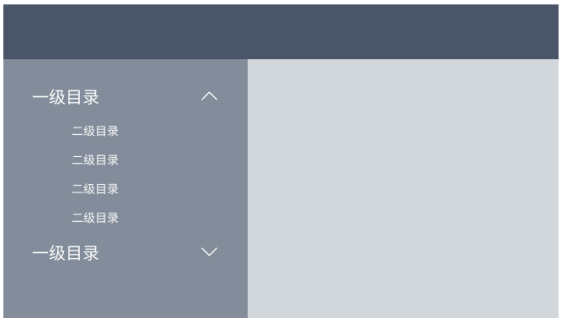


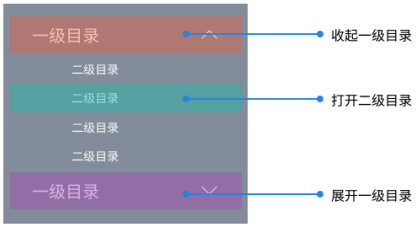
相关页面：

- 云平台的各一级页面

页面布局：

- **顶栏**：吸顶、固定高度的导航栏，用于承载 logo、导航、个人中心等功能。
- **侧边栏**：常驻页面左侧，固定宽度，上下通栏。用于承载一二级 tab 菜单。





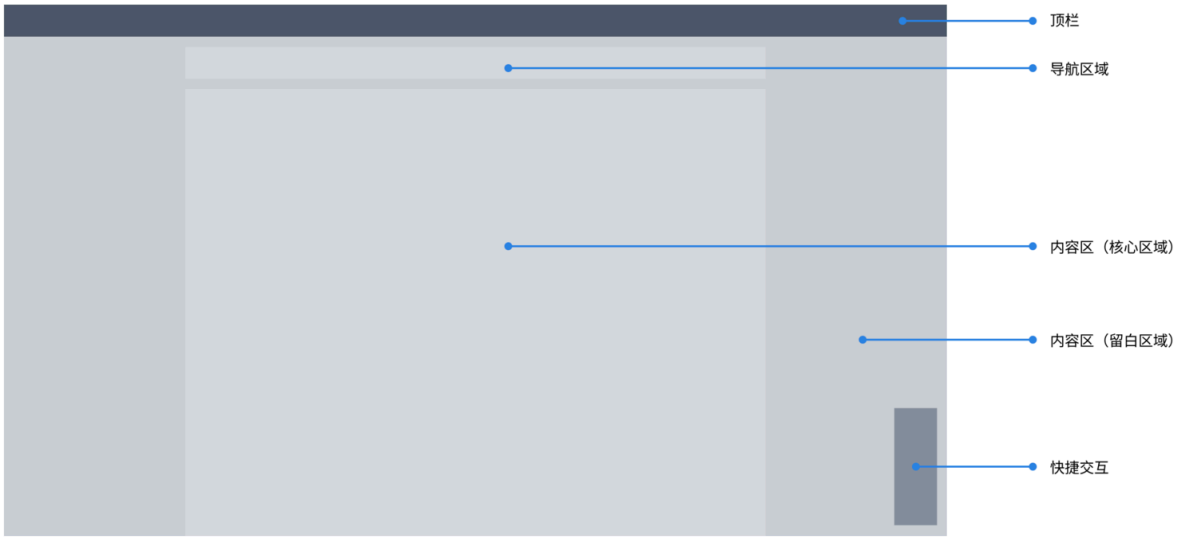
- **一级目录**：默认为收起状态，点击交互热区展开/收起下属的二级目录
- **二级目录**：点击交互热区，在当前页切换显示对应的表单页
- **内容区**：用来承载表单、复杂交互的区域
- **内容区（核心区域）**：布局上尽量将表单、筛选器、按钮等交互布置在核心区域
- **内容区（留白区域）**：因为有的页面可能会有常驻右下角的快捷操作浮窗，因此内容区域右侧建议留白，防止右下角内容与浮窗重叠

- **快捷交互**：如需可在页面右下角布置最高层的常驻弹窗，一般会承载高频交互

适配策略：

- **顶栏**：固定高度吸顶
- **侧边栏**：固定宽度常驻左侧，可以扩展为浏览器宽度过小时，折叠为缩略显示态
- **快捷交互**：固定尺寸常驻右下角
- **内容区**：根据浏览器大小自适应变化，其中显示和交互元素有最小宽度和最大宽度，高度固定。
- 浏览器宽度小于最小宽度时，底部出现滚动条；
- 浏览器宽度大于最大宽度时，元素中央居中、左右扩展留白；
- 浏览器高度过小时，显示右侧滚动条；
- 浏览器高度过大时，底部扩展留白。

4.3 内容页



相关页面：

- 由表单页列表点击“查看”或“编辑”进入的详情页。
- 用于新建复杂内容的页面（例如：新建策略、新建活动）。

页面布局：

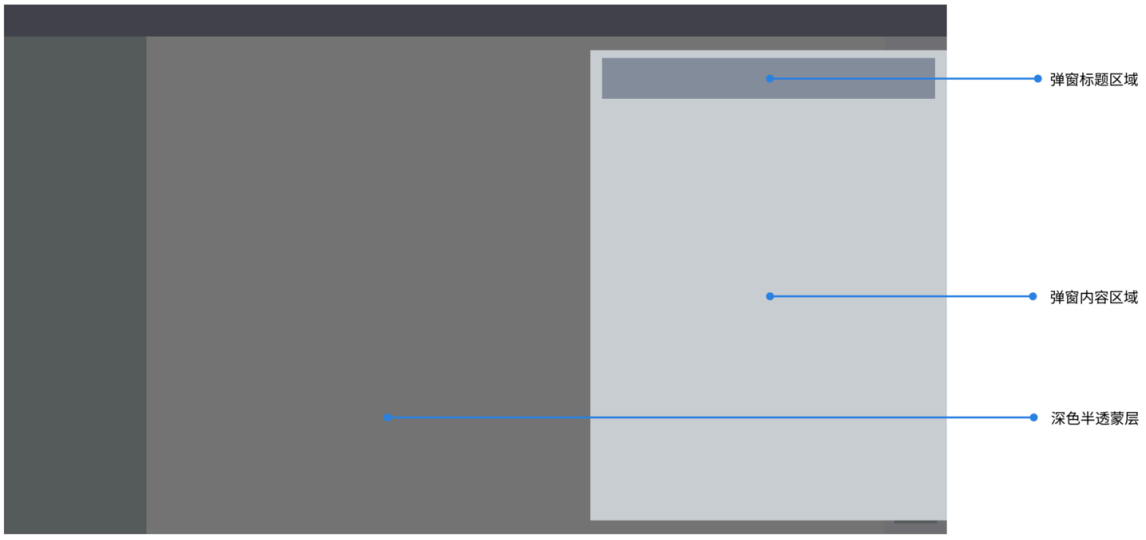
- **顶栏**：同 4.2 表单页–顶栏。吸顶、固定高度的导航栏，用于承载 logo、导航、个人中心等功能。
- **导航区域**：位于顶栏下方，一般用于承载面包屑（Breadcrumb）导航、返回按钮、保存或取消按钮、页内 Tab 切换。
- **面包屑**：标明当前页面在层级结构中的位置，并支持返回上一级。
- **页内 Tab**：用于在同一页面内切换不同的内容模块。
- **内容区**：承载详情、表单、数据看板等核心交互的区域。
- **内容区（核心区域）**：布局上尽量将详情内容、数据图表、操作按钮等交互布置在核心区域。

- **内容区（留白区域）**：因为有的页面可能会有常驻右下角的快捷操作浮窗，因此内容区域右侧建议留白，防止右下角内容与浮窗重叠。
- **快捷交互**：同 4.2 表单页–快捷交互。如需可在页面右下角布置最高层的常驻弹窗，一般会承载高频交互。

适配策略：

- **顶栏**：固定高度吸顶。
- **导航区域**：固定在顶栏下方，宽度随内容区变化。当内容区出现横向滚动条时，导航区域随之滚动。
- **快捷交互**：固定尺寸常驻右下角。
- **内容区**：根据浏览器大小自适应变化，其中显示和交互元素有最小宽度和最大宽度，高度自适应。
- 浏览器宽度小于最小宽度时，底部出现滚动条；
- 浏览器宽度大于最大宽度时，元素中央居中、左右扩展留白；
- 浏览器高度过小时，内容区显示右侧滚动条（顶栏和导航区域固定）；
- 浏览器高度过大时，底部扩展留白。

4.4 弹窗



相关页面：

- 作为承载容器，在当前页面（表单页或内容页）上弹出，用于执行快速查看、编辑、新建或确认等操作。
- 使用弹窗可避免不必要的页面跳转，保持用户当前的操作上下文。

页面布局：

- **深色半透明蒙层**：用于覆盖除弹窗外的整个页面可视区域，将视觉焦点集中在弹窗上。点击蒙层区域无交互，防止误触。
- **弹窗标题区域**：位于弹窗顶部，固定高度。一般包含弹窗标题和关闭按钮（“x”）。
- **弹窗内容区域**：位于标题区域下方，用于承载表单、详情信息、确认文案等核心内容。

适配策略：

- **蒙层**：铺满整个浏览器可视窗口。
- **弹窗**：
- **位置**：在浏览器可视窗口中保持垂直居中，从右侧展开。
- **尺寸**：宽度一般为固定值（或根据内容在一定范围内自适应），高度根据内容自适应，但必须设定最大高度。
- **滚动**：当弹窗内容区域的高度超过设定的最大高度时，内容区域内部出现滚动条，而标题区域和底部的操作按钮区域（如有）保持固定。

5.**页面层级**

5.1 层级说明

为了方便维护和管理，把 D2 所有自研云平台相关功能聚合在同一个域名下，通过权限系统控制展示、访问权限。每个层级的页面说明如下：

- 顶栏导航：按照业务线区分，列入「OTA 平台」、「远程诊断」、「车态数据」等
- 侧边栏
- 一级目录：按照某个业务线下的核心模块区分，例如「OTA 平台」下可能会有的一级目录有「配置管理」、「车辆管理」等
- 二级目录：某个模块内的具体功能，例如「OTA 平台」–「车辆管理」–「车型基础数据」/「车辆基础数据」等

5.2 目录规范

云平台的目录层级维护在以下表格「云平台规范文档」中：
https://cooper.didichuxing.com/docs2/sheet/2205750851938?sheetId=cU5cY

为了便于沟通，后续 PRD 及原型图中，不必要体现完整的目录层级，只需要显示文档中相关的目录层级即可，完整的目录层级、顺序、名称以表格为准

5.3 工作流

后续如果需要新增或调整目录层级，请按照以下流程推进：

（1）具体功能的产品与张高志沟通，确认需求

（2）由张高志在目录规范文档中，新增或修改

（3）具体功能 PRD 按照目录规范中定义的页面层级撰写，进入 PRD 评审流程

具体功能的 PRD 中如果需要声明后台目录层级，可以引用云平台目录规范文档。

1.需求背景

本文档描述云平台的基础前端框架，云平台及衍生 web 后台功能尽可能遵守本文档的前端框架，以减少重复设计和开发工作量。

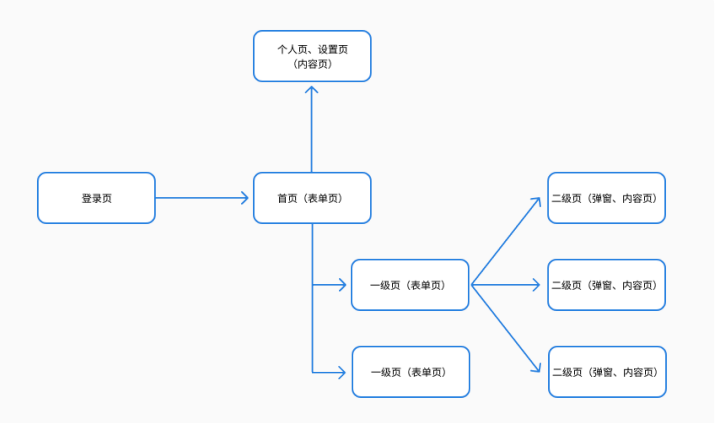
2.前端框架参考

云平台尽量采用某个统一的前端设计框架，推荐字节的 arco.design：https://arco.design/

后续产品经理在定义后台功能时，需要翻阅 arco 的设计指南，确保自己的设计能够通过已有组件搭建实现；设计师尽量遵守 arco 提供的设计规范，如果需要出设计稿，可以更侧重布局、间距等定义，尽量使用 arco 的通用组件。

本文档基于 arco 撰写，如果更换为其他前端框架，会评审沟通后修改文档。

3.云平台页面结构



上图为简单示例，为了便于说明，云平台相关的页面分为四大类，下文将逐个说明每一类的通用交互原则。

- 登录页：登录、登出、修改密码等功能的承载页面，原则上未登录状态下访问云平台仅可访问登录页，未登录则无法触达含有具体内容的其他页面
- 表单页：首页或其他一级功能的承载页面，一般包含顶栏、侧边栏、筛选器和表单
- 内容页：查看、修改表单中详细内容的承载页面，一般包含顶栏、内容区
- 弹窗：查看、修改表单中详细内容的承载容器，非独立页面

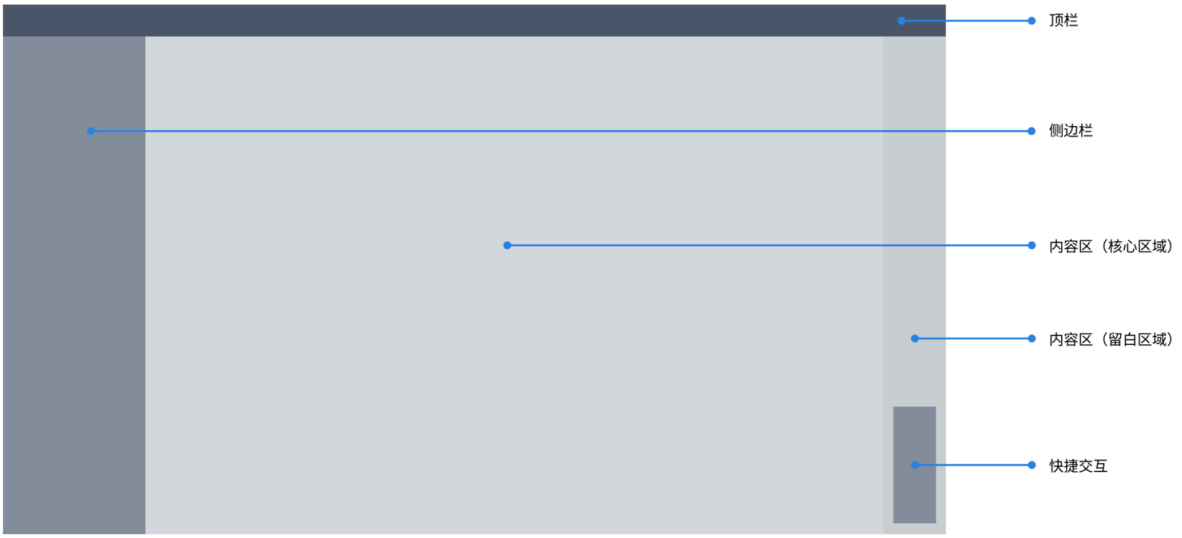
4.交互原则

4.1 登录页

相关页面：登录页、找回密码、修改密码……

功能描述：接入集团 IAM，包含权限系统、SSO 登录等功能

4.2 表单页

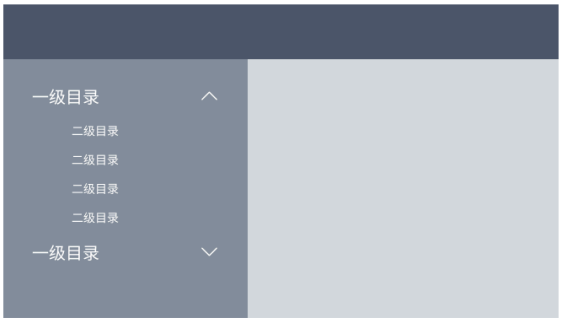


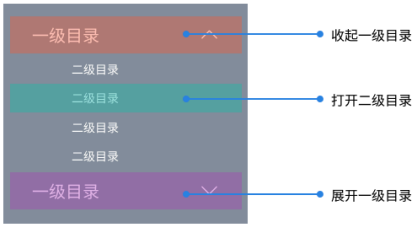
相关页面：

- 云平台的各一级页面

页面布局：

- **顶栏**：吸顶、固定高度的导航栏，用于承载 logo、导航、个人中心等功能。
- **侧边栏**：常驻页面左侧，固定宽度，上下通栏。用于承载一二级 tab 菜单。





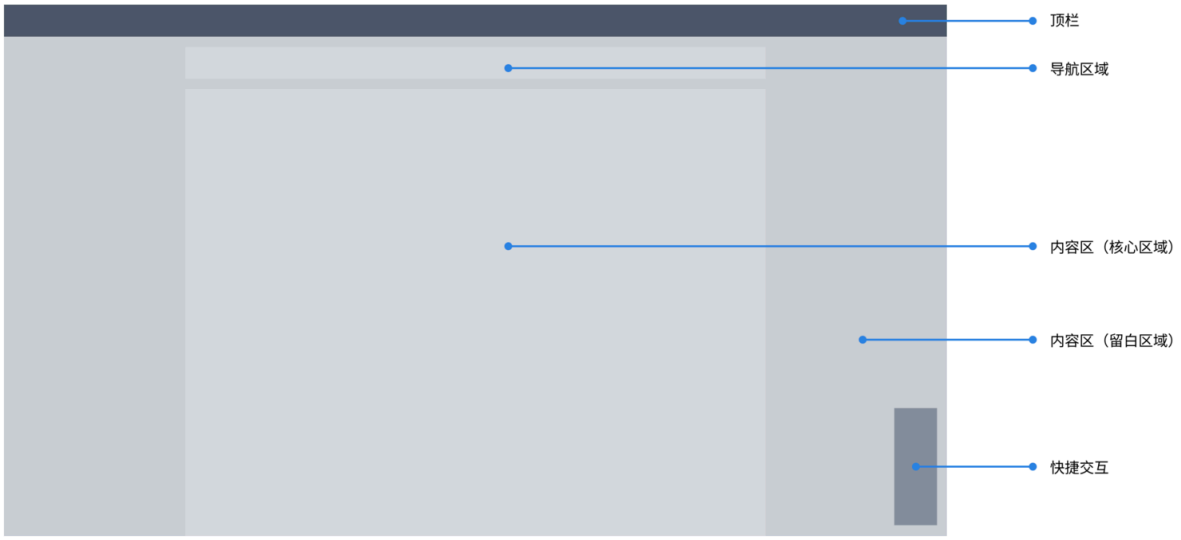
- **一级目录**：默认为收起状态，点击交互热区展开/收起下属的二级目录
- **二级目录**：点击交互热区，在当前页切换显示对应的表单页
- **内容区**：用来承载表单、复杂交互的区域
  - **内容区（核心区域）**：布局上尽量将表单、筛选器、按钮等交互布置在核心区域
  - **内容区（留白区域）**：因为有的页面可能会有常驻右下角的快捷操作浮窗，因此内容区域右侧建议留白，防止右下角内容与浮窗重叠

- **快捷交互**：如需可在页面右下角布置最高层的常驻弹窗，一般会承载高频交互

适配策略：

- **顶栏**：固定高度吸顶
- **侧边栏**：固定宽度常驻左侧，可以扩展为浏览器宽度过小时，折叠为缩略显示态
- **快捷交互**：固定尺寸常驻右下角
- **内容区**：根据浏览器大小自适应变化，其中显示和交互元素有最小宽度和最大宽度，高度固定。
  - 浏览器宽度小于最小宽度时，底部出现滚动条；
  - 浏览器宽度大于最大宽度时，元素中央居中、左右扩展留白；
  - 浏览器高度过小时，显示右侧滚动条；
  - 浏览器高度过大时，底部扩展留白。

4.3 内容页



相关页面：

- 由表单页列表点击“查看”或“编辑”进入的详情页。
- 用于新建复杂内容的页面（例如：新建策略、新建活动）。

页面布局：

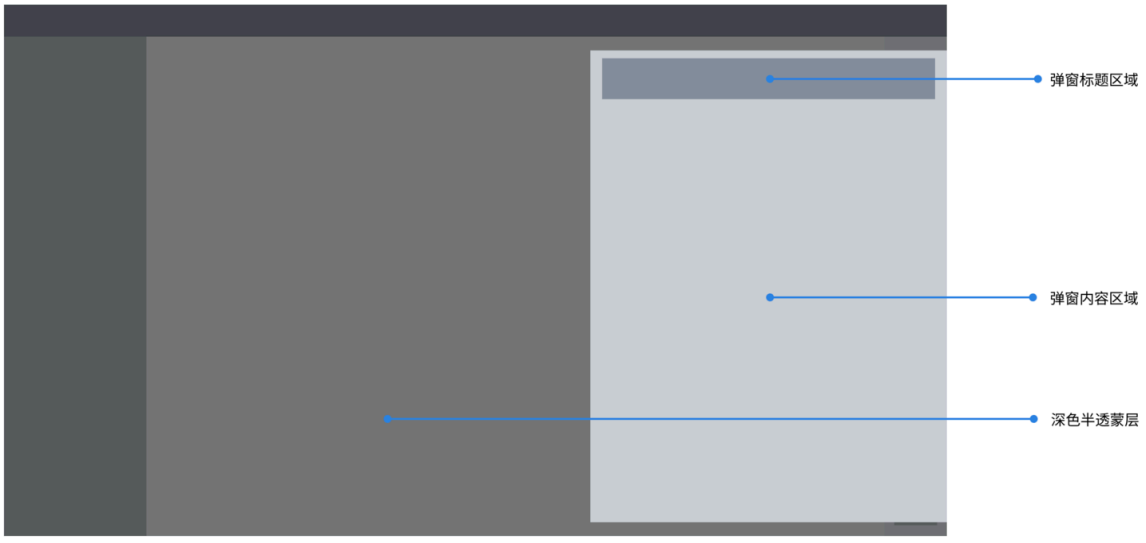
- **顶栏**：同 4.2 表单页–顶栏。吸顶、固定高度的导航栏，用于承载 logo、导航、个人中心等功能。
- **导航区域**：位于顶栏下方，一般用于承载面包屑（Breadcrumb）导航、返回按钮、保存或取消按钮、页内 Tab 切换。
  - **面包屑**：标明当前页面在层级结构中的位置，并支持返回上一级。
  - **页内 Tab**：用于在同一页面内切换不同的内容模块。
- **内容区**：承载详情、表单、数据看板等核心交互的区域。

    - **内容区（核心区域）**：布局上尽量将详情内容、数据图表、操作按钮等交互布置在核心区域。
    - **内容区（留白区域）**：因为有的页面可能会有常驻右下角的快捷操作浮窗，因此内容区域右侧建议留白，防止右下角内容与浮窗重叠。
- **快捷交互**：同 4.2 表单页–快捷交互。如需可在页面右下角布置最高层的常驻弹窗，一般会承载高频交互。

适配策略：

- **顶栏**：固定高度吸顶。
- **导航区域**：固定在顶栏下方，宽度随内容区变化。当内容区出现横向滚动条时，导航区域随之滚动。
- **快捷交互**：固定尺寸常驻右下角。
- **内容区**：根据浏览器大小自适应变化，其中显示和交互元素有最小宽度和最大宽度，高度自适应。
    - 浏览器宽度小于最小宽度时，底部出现滚动条；
    - 浏览器宽度大于最大宽度时，元素中央居中、左右扩展留白；
    - 浏览器高度过小时，内容区显示右侧滚动条（顶栏和导航区域固定）；
    - 浏览器高度过大时，底部扩展留白。

4.4 弹窗



相关页面：

- 作为承载容器，在当前页面（表单页或内容页）上弹出，用于执行快速查看、编辑、新建或确认等操作。
- 使用弹窗可避免不必要的页面跳转，保持用户当前的操作上下文。

页面布局：

- **深色半透明蒙层**：用于覆盖除弹窗外的整个页面可视区域，将视觉焦点集中在弹窗上。点击蒙层区域无交互，防止误触。

- **弹窗标题区域**：位于弹窗顶部，固定高度。一般包含弹窗标题和关闭按钮（“x”）。
- **弹窗内容区域**：位于标题区域下方，用于承载表单、详情信息、确认文案等核心内容。

适配策略：

- **蒙层**：铺满整个浏览器可视窗口。
- **弹窗**：
  - **位置**：在浏览器可视窗口中保持垂直居中，从右侧展开。
  - **尺寸**：宽度一般为固定值（或根据内容在一定范围内自适应），高度根据内容自适应，但必须设定最大高度。
  - **滚动**：当弹窗内容区域的高度超过设定的最大高度时，内容区域内部出现滚动条，而标题区域和底部的操作按钮区域（如有）保持固定。

5.页面层级

5.1 层级说明

为了方便维护和管理，把所有自研云平台相关功能聚合在同一个域名下，通过权限系统控制展示、访问权限。每个层级的页面说明如下：

- 顶栏导航：按照业务线区分，例如「OTA 平台」、「远程诊断」、「车态数据」等
- 侧边栏
  - 一级目录：按照某个业务线下的核心模块区分，例如「OTA 平台」下可能会有的一级目录有「配置管理」、「车辆管理」等
  - 二级目录：某个模块内的具体功能，例如「OTA 平台」–「车辆管理」–「车型基础数据」/「车辆基础数据」等

5.2 目录规范

云平台的目录层级维护在以下表格「云平台规范文档」中：「文档链接」

为了便于沟通，后续 PRD 及原型图中，不必要体现完整的目录层级，只需要显示文档中相关的目录层级即可，完整的目录层级、顺序、名称以表格为准

5.3 工作流

后续如果需要新增或调整目录层级，请按照以下流程推进：

（1）具体功能的产品与 XXX 沟通，确认需求

（2）由 XXX 在目录规范文档中，新增或修改

（3）具体功能 PRD 按照目录规范中定义的页面层级撰写，进入 PRD 评审流程

具体功能的 PRD 中如果需要声明后台目录层级，可以引用云平台目录规范文档。

功能名称：确认购买按钮

功能位置：用户在商详页、XX 页、XX 页，点击购买按钮后，展开商品信息选择弹窗，弹窗底部为确认购买按钮

按钮文案：确认

按钮交互：可点击，点击后进行商品信息校验流程

流程通过 – 进入订单信息确认页

流程未通过 – 停留在本页面，并根据 XXX 文档的定义，使用 Toast 类型 1，提示错误信息

特殊说明：例如用户在点击确认购买按钮的时候，对应商品已经在后台操作下架，此时需要停留在本页面并且取字段 XXX，用 toast 类型 1 提示“对不起，该商品已下架”