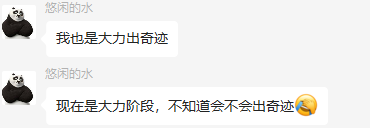
PS：**基于RxDrag项目简单学习后的反馈，自身非前端开发部分反馈可能存在不当的地方，仅做参考。**

看到这个文档，第一个反应就是感动，非常感谢有这么热心的朋友帮助自己，这是前进的最大动力，争取早日做好产品，回馈社区。

# 1.初衷：希望做一点有价值的事情，但不知道有没有用



反馈：有利于我们提高效率、降低成本、改善生活的事情都是有价值的，都值得我们投入，做到极致奇迹就会会发生。

就低代码平台的定位而言：可降低使用者门槛、提高开发效率，方向上至少是一件对的事情。

是的，我也始终相信，要做对的事情，做对别人有用的事情，做创造价值的事情。我本人并没有到完全财务自由的阶段，所以做事情还是看收益的。不过，我相信，只要做出的东西，能让足够多的人受益，那么我自己的收益只是副产品。

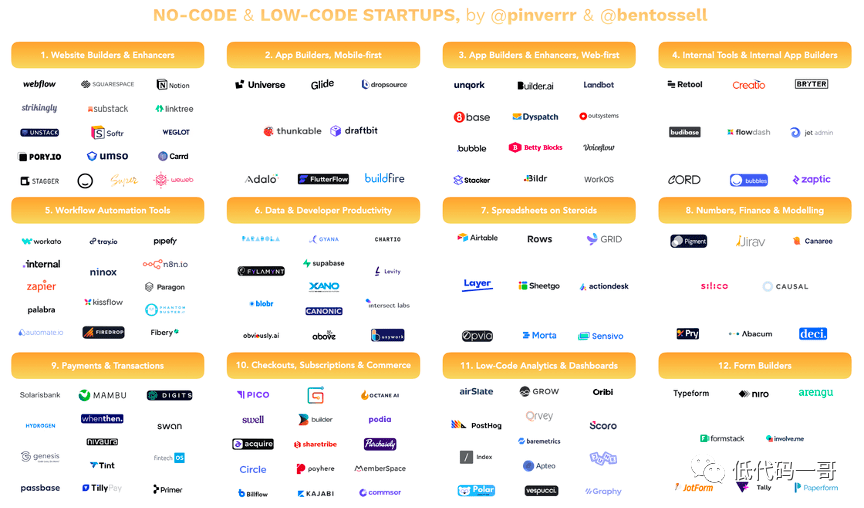
就是说，先给他人创造价值，然后再考虑自己的收益。

另外，自己是一个业余程序员，没有什么市场资源，又热爱这个行当，做开源项目是相对比较容易的切入点。但是，做好要比闭源难。

路，选了，是一定要走下去的。

# 2.哪些因素会影响奇迹的到来？

* 产品价值：能够为客户/用户带来的价值，低代码平台其实是近几年的热点，产品需求和价值是明确的
* 产品竞争：竞争态势上来看，其实不是很妙，国内外的低代码平台(或者号称做低代码的)其实很多了



* 产品定位：其实，通用的低代码平台很难，比如能通过低代码开发出抖音吗？微信？因此找一个合适的场景做好定位，或许是成功的关键
* 用户体验：功能层面，苹果、微信的功能其实竞品都有，但是用户体验是不一样的，良好的用户体验是竞争激烈产品，成功的关键因素之一(建议:团队引入UED？)
* 商业模式和应用推广：良好的商业模式是产品可持续发展的关键，应用推广可以让更多的人受到产品带来的价值，同时积累其品牌。应用中的用户反馈同样是产品改进的驱动力。

低代码，目前确实是一片红海。中国人这么多，做什么没有竞争呢？所以，我不怕竞争，也崇尚段永平说的“敢为天下后”。

敢为天下后的底气，是用心做事的发心。愿意观察市场痛点，愿意俯下身子去满足用户的需求，愿意把用户体验做到极致，不急功近利，愿意从又苦又累的工作开始，慢慢积累口碑与市场及渠道资源。

不怕失败，不管跌倒多少次，都能爬起来。只要目标清晰，总能做出适合用户的东西。

一个人的力量是有限，做成一件事，需要汇集多方面的力量。我相信随产品的不断完善与成熟，会有擅长商业运作又志同道合的朋友来谈合作的。

产品的定位，绝对不能是通用的。后面会找一个垂直领域，切入并深耕。目前比较看好的领域是，工厂mes之类的应用。这类应用的特点是，不算复杂，但是对订制要求极高，正好是低代码发挥效力的地方。

# 3.项目反馈和建议

3.1.关于产品定位上

从项目介绍来看，平台的野心还是很大的，对软件类产品做了尽可能的抽象，甚至低代码平台自身的管理应用也是通过配置和导入来实现。

平台自举这个功能，确实有些画蛇添足，并不是因为野心大，是因为领域经验不多，没有合适的实践机会。

通过导入配置，自举平台，有个非常大的缺点就是，dsl有少许修改，就会导致系统崩溃。这个部分已经在修改的路上了。

但越是通用，通常来说复杂程度越高，相对来说学习成本和使用门槛就越高了。或许可以找到一些规模适中的场景做相对垂直的优化会更好一些。

是的，一定要找垂直领域进入。

对于OA、CRM、WMS等可以实现的低代码平台市面上相对较多，如果我推荐一个的话，可是试试目前大火的chatgpt集成？人工智能管理？物联网？

chatgpt的话，还能蹭一波热度，做一些推广，哈哈哈

chatgpt后面会集成，等再成熟一下。毕竟现在还不是推广的时候，产品没有完成，过度的推广也没有什么意义。

从3月初写完那篇前端文章，到3月底，各个平台转发，v站发了几个帖子。一个月的时间，Github大约涨了1.8k的星。

这是对产品推广的初步尝试，后面短时间不再推广了，专心做好产品。相信，只要产品足够好，不愁推广的。

3.2低代码平台的管理平台建议初始化到项目安装，演示中也通过配置和导入，固然可以证明平台的强大，但实际应用场景中其实没有必要，用户的目的并不是想要配置出这个平台。

确实没有必要，上面说了，正在改。

3.3做好外部数据的连通，我们做软件平台其实底层就是处理各种各样的数据，然后再上层设计界面来进行人机交互，如果数据仅仅在独立系统中，可以实现的功能就大打折扣了，其实在平台中看到有接口模块是蛮欣喜的，但是发现接口模块是自身生成的接口，并不是和外接通讯的接口。

近期后端会做一个很大的重构，基于微服务架构。第三方接口，可以以微服务的形式集成进平台，并且平台提供生成微服务脚手架功能。

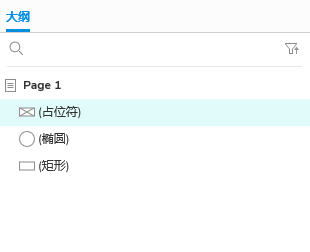
初步想法有这些微服务：

1. entify ，领域模型服务，基于UML类图。一这个为中心，基于领域模型，做一套前后端通用的类型系统，用于各种校验以及脚本、代码的提示等。
2. 脚本服务，通过在线js脚本，扩展系统。
3. 服务编排（逻辑流），通过可视化的形式，把各种微服务编排成新的接口。
4. 订阅服务，订阅的方式发布各种事件
5. 定时任务服务
6. 工作流服务

所有微服务对外提供graphql接口，基于fedration 做API网关。最终发布的接口形式：

API网关+订阅服务。

3.4页面配置建议增加组件列表和选择器，当页面布局拖入太多组件后，会给编辑和管理带来困难，尤其还有一些组件编辑状态是透明的，可参考PPT/Axure增加组件选择工具，类似下面的东西



这个我记下了，后面有机会就加上，类似功能欢迎发issue，免得我忘了。

另外，工具条有辅助线跟辅助空隙，可以帮助选中组件。

3.5允许用户自定义模板和组件，这有助于用户在自身关心的业务或领域，积累自己的模板，进一步提高效率

这个功能在formily版本就已经实现了，后面会移植到平台新版中来。

3.6当用户可以自定义组件和模板之后，我们的组件就可以化繁为简了，就像产品常使用的Axure，其实用简单的方框、文字、按钮、动态面板就可以实现大多数的功能了。将组件精简，但组件配置和可控性更丰富，或许有利于降低用户门槛和学习成本

Formily版中，用户可以自定义组件，并通过插件的形式热插入组件，也可以自定义组件模板并保存，组件模板可以导入、导出。这个功能后面移植过来，会进一步优化用户体验。

3.7基于UML的模型设计很好，我在做客户数字化项目的时候，往往都是先分析业务架构，再根据业务设计数据架构(例如哪些数据来自接口、哪些数据来自配置、哪些数据来自交互)，最后再设计界面UI。看到模型设计中有流程设计、模型设计还是很好的。当然在数据处理过程中，还是有很多流程化的东西，尽早将流程引擎接入还是很好的

是的，上面说了，这方面工作正在做，欢迎继续提宝贵意见。

3.8目标用户引申的功能，

如果目标用户是软件开发团队，我们的平台帮助他们提高定制化交付的效率，那么配置出来的应用应该具备应用导出和私有化部署的能力

这个是重点考虑的，后面会增加静态部署功能，尽量做成可以导出静态部署的代码。

如果我们的目标用户是，业务企业的IT部门，用于企业新的应用需求配置或流程自动化，那么与其他业务系统的接口就很重要了，比如钉钉上的氚云。

这个没有相关经验，目前没考虑太多。

最后，再次感谢您的宝贵意见跟建议。只要有热心的朋友在，就算有再多困难，前行的路上也不会孤单！