低代码设计目标及核心设计

接上一篇《低代码的现状与目标分析》，本篇主要细化并阐述一下zPaaS低代码开发平台的设计目标以及核心设计思想。

1. 选择一个赛道

在软件/信息化行业中，并不是所有的细分的场景都适合应用低代码，根据低代码的定义和特性，存在较多重复场景且个性化需求（非业务需求）能够控制在一定范围内的细分场景，会更适合低代码平台的应用和发挥。

根据个人多年的行业经验，企业级应用开发场景是一个比较适合应用低代码的场景：

- **存在较多可抽象的重复场景**，如增删改查类功能场景、流程审批类功能场景、统计报表类功能场景、可视化图表类功能场景、系统接口类功能场景等，这些场景至少可以覆盖一个企业级应用6、70%功能的开发场景。
- 一个系统/一个企业的**UI/UE相对统一**，可以通过扩展主题及布局的方式满足企业/系统的个性化需求，避免目前常规前端低代码平台，逐个功能适配的做法。
- **个性化需求能够控制在一定范围内**，企业的个性化需求以业务需求为主，非业务个性化需求往往会集中在少数的核心业务功能中，对于少数个性化需求较高的功能，可以采用定制化开发结合的方式来解决。
- **前后端的技术底座相对统一**，国内企业级应用的技术底座以JAVA+Spring为主，有利于低代码开发和定制化开发的无缝融合。

因此，zPaaS低代码开发平台选择的赛道是企业级应用的低/零代码的开发场景。

2. 确定设计目标

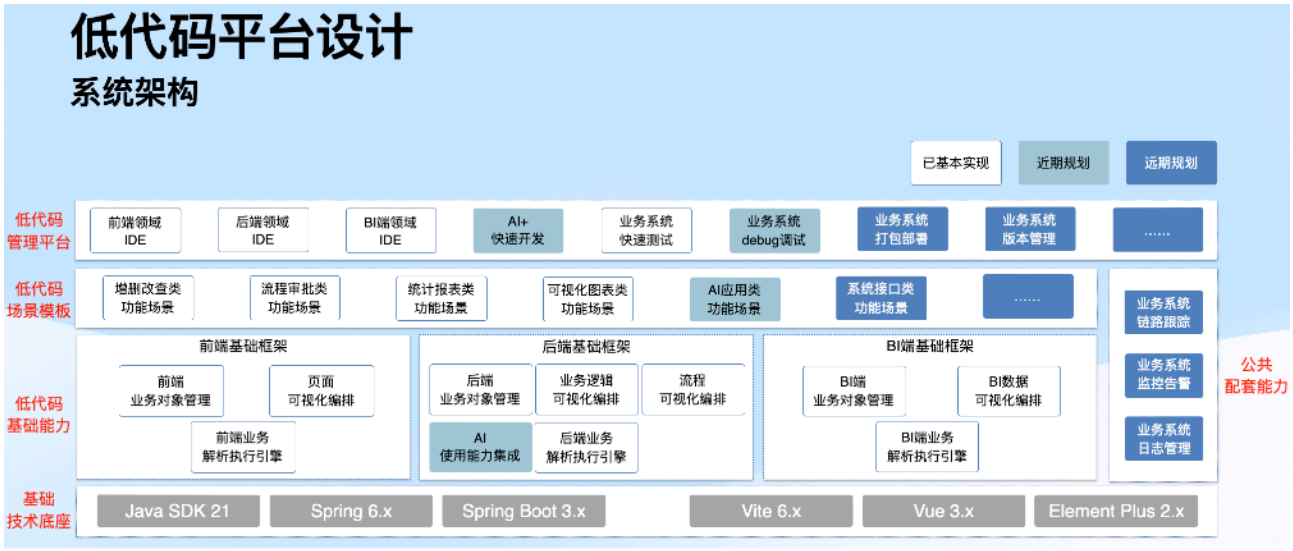
赛道确定后，那么zPaaS低代码开发平台的设计目标也基本可以确定：

- **覆盖企业级应用的低/零代码的开发场景**，支撑目标为一个企业级业务系统中6、70%功能的低/零代码开发。
- **结合领域驱动设计、面向对象设计、系统分层设计等流行设计思想**对基于低代码开发的业务系统中的业务对象进行抽象和管理。
- **支持基础的前、后端以及BI功能的低/零代码开发能力以及模板化的快速开发能力**，至少覆盖增删改查类功能场景、流程审批类功能场景、统计报表类功能场景、可视化图表类功能场景、系统接口类功能场景以及AI应用类功能场景的模板化开发能力。
- 在满足一定条件的情况下，**支持低代码开发与定制化开发的无缝衔接**，包括相互之间的方法调用、支持本地化事务、认证鉴权适配以及页面相互嵌套等，同时支持多种部署模式，包括混合部署、独立部署等。

- 在基础的低/零代码开发能力之外，**支持一些较复杂能力的低/零代码开发**，如下拉选择的灵活配置、属性间的联动/显示/隐藏配置、属性翻译、弹出选择/查看（功能集成）、文件导出/导入等。
- **采用解析执行引擎的模式**，支持基于低代码平台开发的业务系统以及低代码平台本身的持续迭代演进。
- **拥抱AI**，包括集成AI能力，使AI能力能够更无缝的结合到业务系统的方方面面以及将AI能力深度应用到低代码平台，使低代码平台本身能够更加的智能。

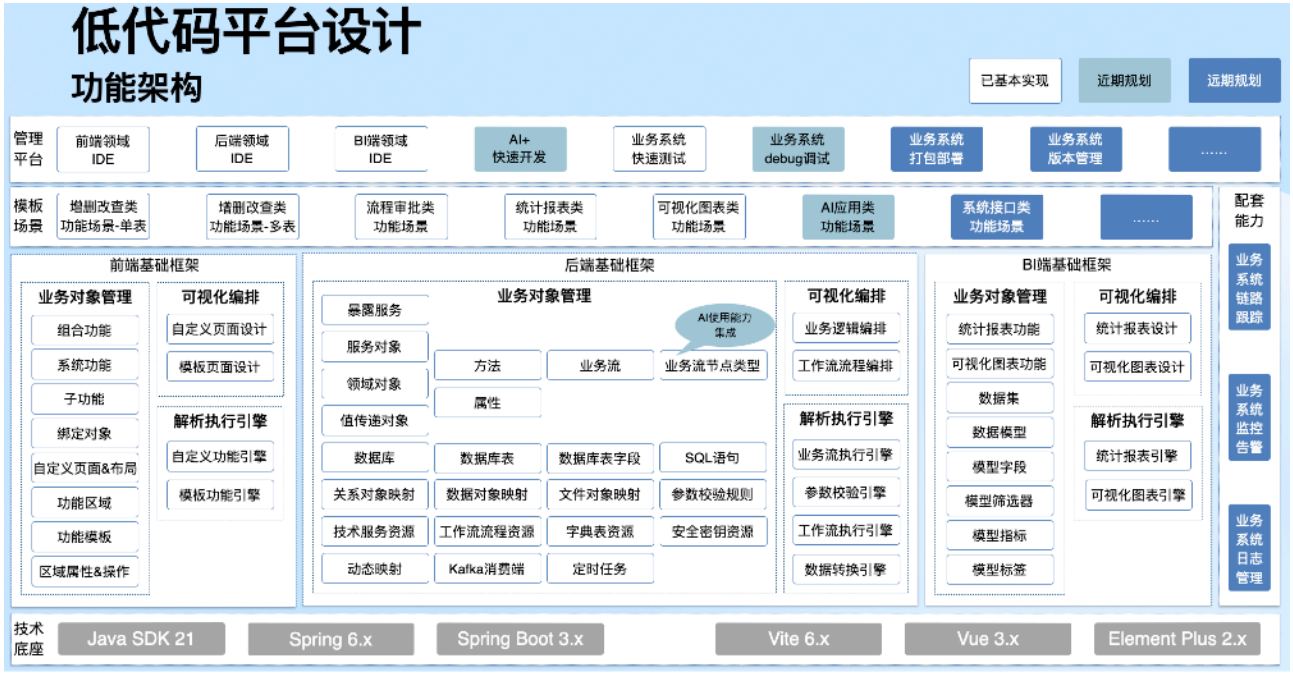
3. 系统架构

zPaaS低代码开发平台的系统架构规划如下：



4. 功能架构

zPaaS低代码开发平台的功能架构规划如下：



初期支持的业务流节点类型以及技术服务资源规划如下：

5. 其他

后续将通过更多的文章，逐步对低代码平台的设计理念、思想以及具体的设计进行细化，随着设计与开发的进展，低代码平台的源码将开源到GitHub：https://github.com/zjyzju/zPaaS-lowcode，敬请关注。

如果有低代码平台以及AI相关的讨论，可以发送邮件到我的邮箱：zjyzju@163.com 。