微信小程序设计文档 (餐饮行业)

> 💡 本文档用于记录微信小程序 (餐饮行业) 技术选型及核心流程设计过程

一、背景

微信小程序是一种全新的连接用户与服务的方式，它可以在微信内被便捷地获取和传播，同时具有出色的使用体验。对于餐饮行业的商家来说，小程序降低了运营的成本，提高了效率，实现线上线下的融合，整合双向流量，并且商家可以自主实现客流量的精细化运营的能力，成为本地商户提升竞争力的关键。

微信小程序官方提供了原生小程序语法来进行开发，同时社区也有许多解决方案，接下来先介绍目前社区现有方案及其对比，并分析已知问题，最终设定开发目标。

1.1 现有方案

本节主要对小程序开发现有解决方案进行对比，由于社区现有的框架，对于微信小程序底层 API 的调用都提供了保障，所以关注点会集中在开发体验、性能对比、代码编译等。

目前微信小程序社区较为推荐方案如下：

- taro
- uni-app
- mpvue
- wepy

方案对比如下：

	原生小程序	uni-app	wepy	mpvue	taro
小程序 API 调用	✅	✅	✅	✅	✅
跨端	❌	✅	❌	❌	✅
ES Next	❌	✅	✅	✅	✅

npm 资源	✅	✅	✅	✅	✅
自定义组件	✅	✅	✅	✅	✅
开箱即用的工程化	❌	✅	✅	✅	✅
css 预编译器	❌	✅	✅	✅	✅
DSL 语法提效	❌	Vue	Vue	Vue	React
数据流方案	westore	vuex	vuex	vuex	dva
性能对比	最优 (调优下)	优	良	良	优
社区资源对比	中	优	中	中	优

下图为社区提供各框架生态对比图表：

		chameleon	mpvue	Taro	uni-app	WePY
开发工具	DSL	类 Vue	Vue	React	Vue	类 Vue
	IDE/图形化开发工具	无	无	无	有	无
	语法校验工具	自研	无	ESLint 规则	IDE 支持	无
	TypeScript	无	有	有	有	有
	Typing/自动补全	无	API	API + JSX	IDE 支持	无
	样式	sass, less, stylus	sass, less, stylus	sass, less, stylus, CSS Modules	sass, less, stylus	sass, less, stylus
多端支持	小程序支持	微信、百度、支付宝	微信、百度、支付宝、字节跳动	微信、百度、支付宝、字节跳动	微信、百度、支付宝、字节跳动	微信、百度、支付宝
	移动端容器	Weex	无	React Native	Weex	无
	移动端增强	chameleon SDK	无	Expo	nvue	无
	多端编译方式	自研多态协议	环境变量条件编译	环境变量 + 文件条件编译	自研条件编译语法	环境变量条件编译
	H5 兼容 API	自研多态协议	无	有	有	无
	跨端组件库	有	无	有	有	无
组件库/工具库/demo	第三方组件	较少	丰富	较丰富	丰富	丰富
	第三方工具库	较少	较丰富	较丰富	丰富	丰富
	Demo	较少	较丰富	较丰富	较丰富	丰富
	状态管理工具	Vuex	Vuex	Redux/MobX/Dva	Vuex	Redux
	转换微信小程序工具	无	无	有	无	有
	自研组件库	有	无	有	有	无
流行度	GitHub Star	3843	16281	16084	2921	16779
	GitHub Issue/PR	68/8	1369/104	2026/368	215/18	1662/441
	NPM + CNPM 下载量 [1]	241/周	1971/周	4332/周	N/A	976/周
	案例	较少	丰富	丰富	非常丰富	丰富
	开发者人数 [2]	~1000 人	~ 5000 人	~ 5000 人	10000+	~ 5000 人
	自建开发者社区	无	无	无	有	无

* 数据采集于 2019 年 3 月 11 日，以各框架目前稳定版特性为标准

[1] 统计上周各框架 CLI 工具下载量，uni-app 可用 IDE 工具直接开发

[2] 统计框架及相关生态微信/QQ 群的总人数

更详细对比可参见：

- 小程序开发：用原生还是选框架（wepy/mpvue/taro/uni-app）

- 跨端开发框架深度横评之2020版
- 小程序框架全面测评

uni-app 等框架的核心问题在于：

- 转换小程序代码存在不确定性，可能存在潜在的性能问题
- 直接使用框架自行封装的组件生态，可能存在无法及时修复 bug 的情况

> 💡 **如果团队无法忽视以上两个问题，即对性能的上限要求比较高，则建议使用原生语法开发。(从微信小程序层面来说，用原生语法开发可以做到底层到上层每个环节都掌握在自己手里)**

原生语法与 uni-app 的差异在于：

- 跨端支持
- ES Next 语法支持
- 开箱即用的工程化支持
- 前端熟悉的 DSL 语法 (Vue) 提效支持
- 开箱即用的数据流管理支持
- 无法 Learn once, write anywhere
- 需要完全重新学习 DSL

> 💡 **如果用原生语法开发需要达到与现有框架一致的开发体验，它的框架研发成本巨大，渲染引擎、布局引擎、DSL、上层框架每个部分都需要大量人力开发维护。**

总结，从开发体验、社区资源、开源质量来说，**uni-app** 是最优选择。**关于性能，在合理的用户体验设计下，uni-app 与 原生小程序并无明显区别**。如果团队选择原生语法开发，则需要抹平开发体验上的差异，从零搭建更为完善的开发体系。(**注意：uni-app 不限制底层API 调用**)

1.2 目标

无论选择原生语法还是框架 (uni-app) 进行小程序开发，需要达成的目标如下：

- 搭建文档中心
- 搭建物料中心

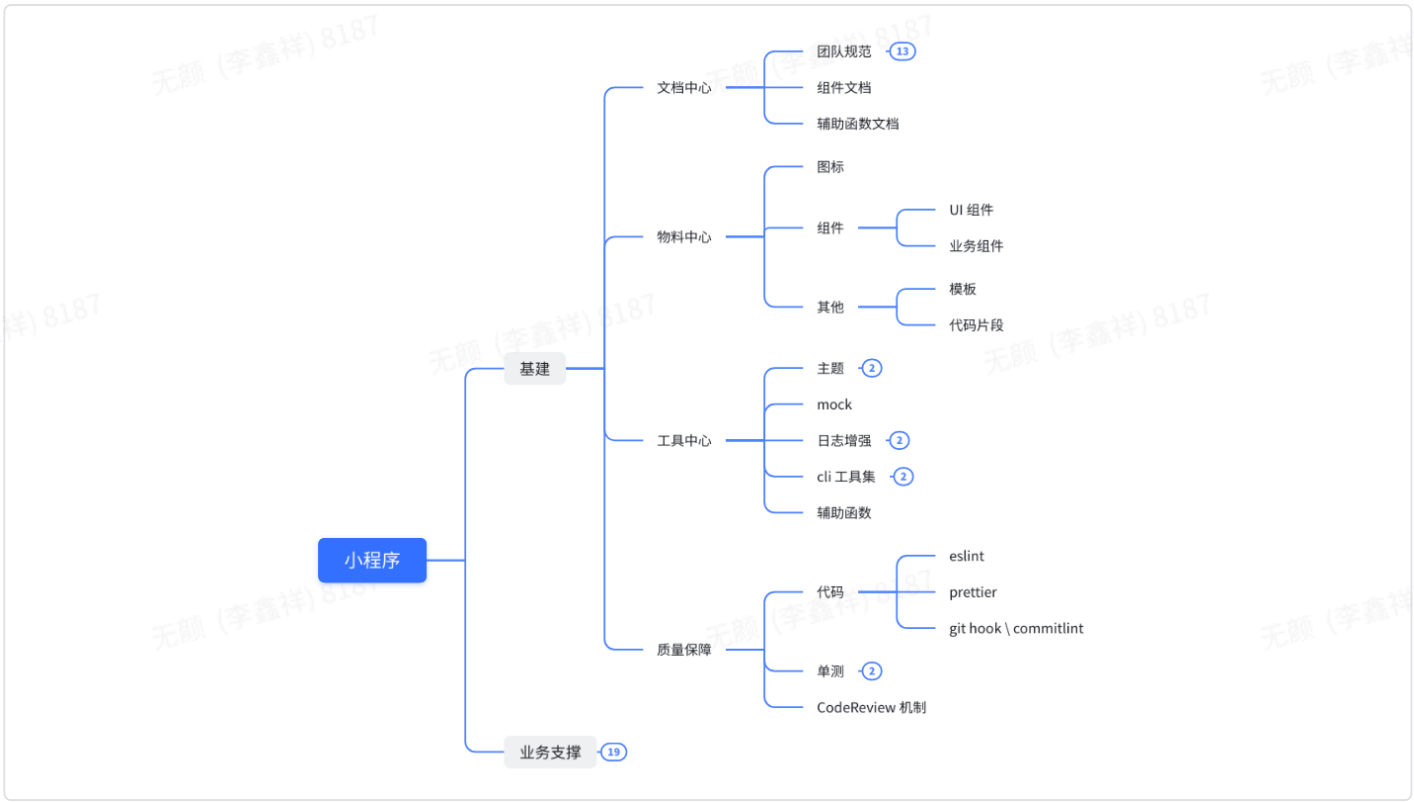
- 搭建工具中心
- 搭建质量保障体系
- 搭建性能测评机制
- 搭建对外合作机制与规约 (UI / UE、Web 接口)

二、整体设计

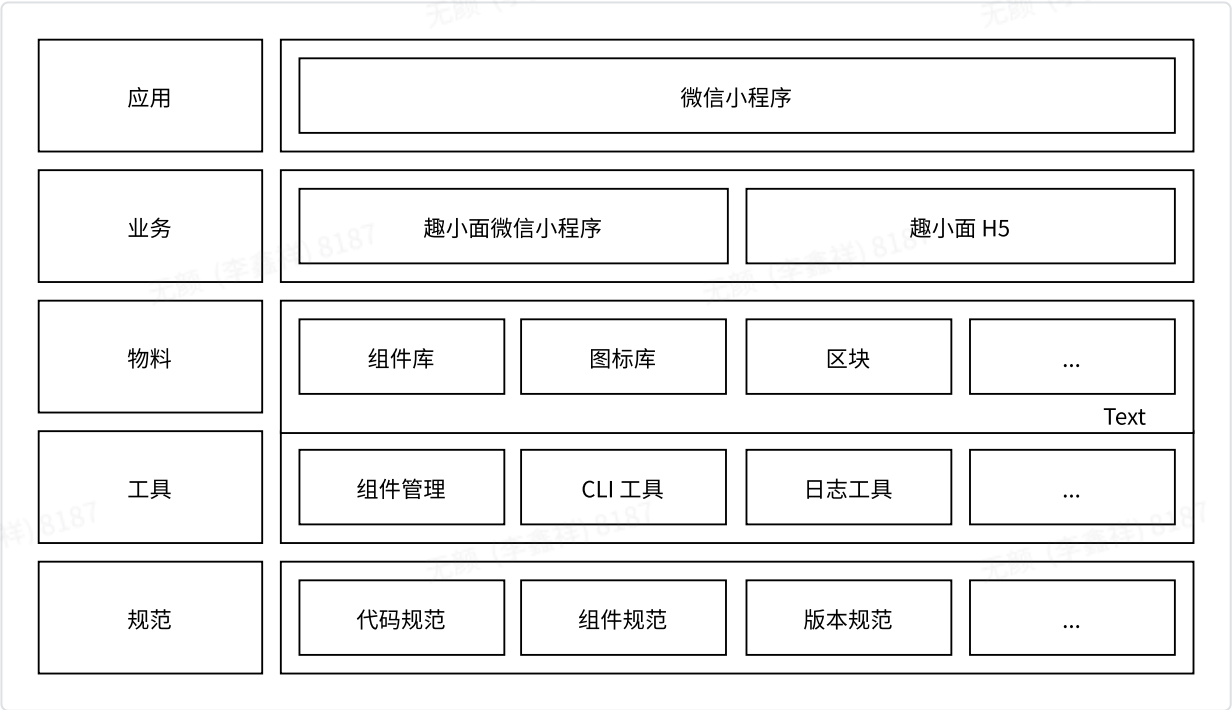
微信小程序整体设计内容包括：

- 基建
- 业务支撑

内容细分：



整体设计：



2.1 基建

2.1.1 基建内容

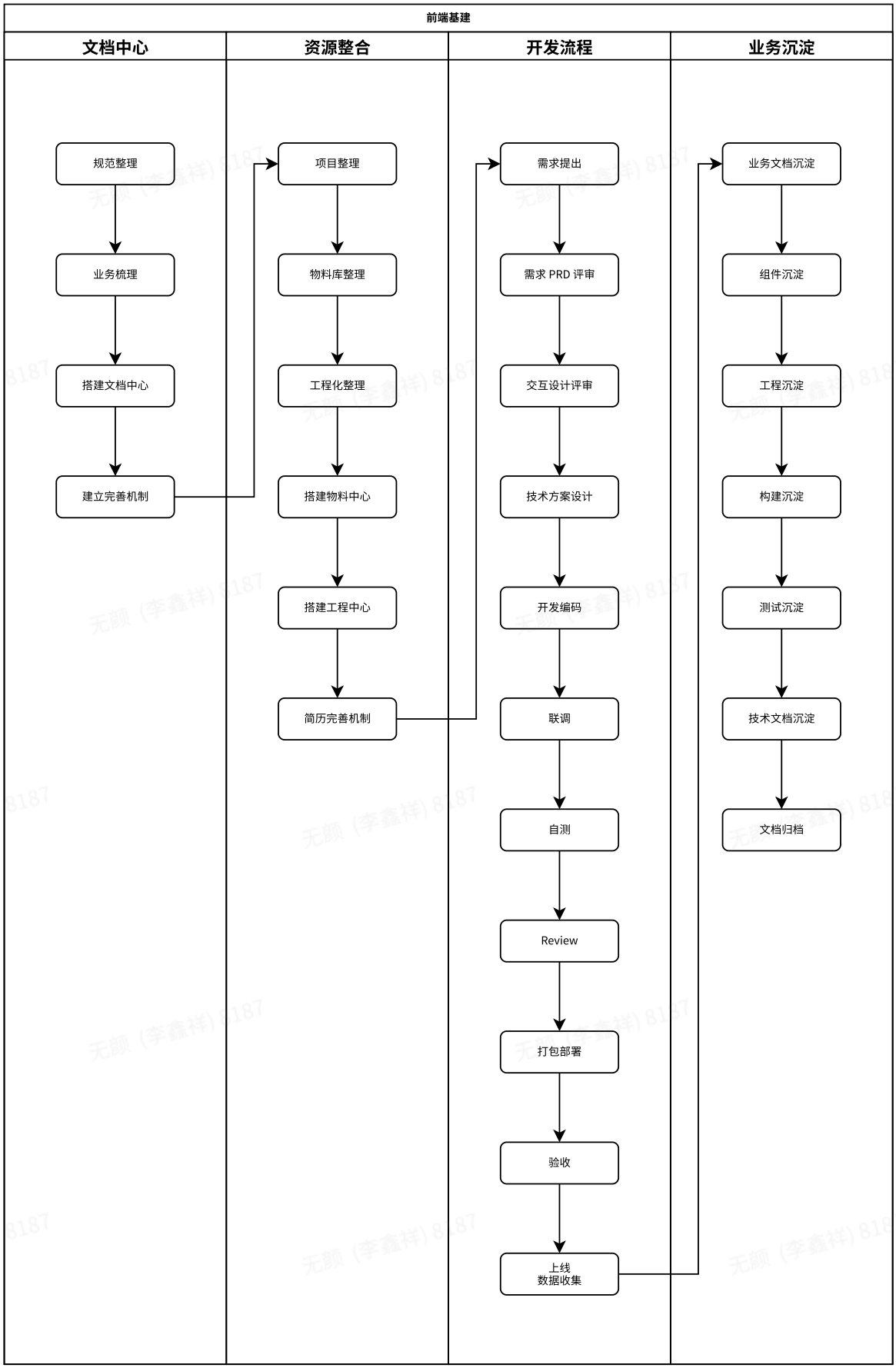
> 💡 以提效为目的，服务于业务的建设称之为基建

内容包括但不限：

- **团队规范：**团队标准化共识，标准化规范是团队高效协作的必要前提
- **研发流程：**业务研发链路，标准化流程直接影响上下游的协作和效率
- **基础资产：**工具链、物料库（组件、区块、模板等）、团队 DSL 沉淀等
- **工程管理：**项目全生命周期的低成本管控，如：项目创建、本地环境配置、打包部署
- **安全防控：**三方包依赖安全、代码合规性检查、安全风险检测等防控机制
- **质量保障：**自测 checkList、单测、UI 自动化测试、链路自动化测试
- **性能检测：**通过自动化、工具化的方式发现页面性能瓶颈，提供优化建议
- **统计监控：**埋点方案、数据采集、数据分析、线上异常监控等

2.1.2 基建落地

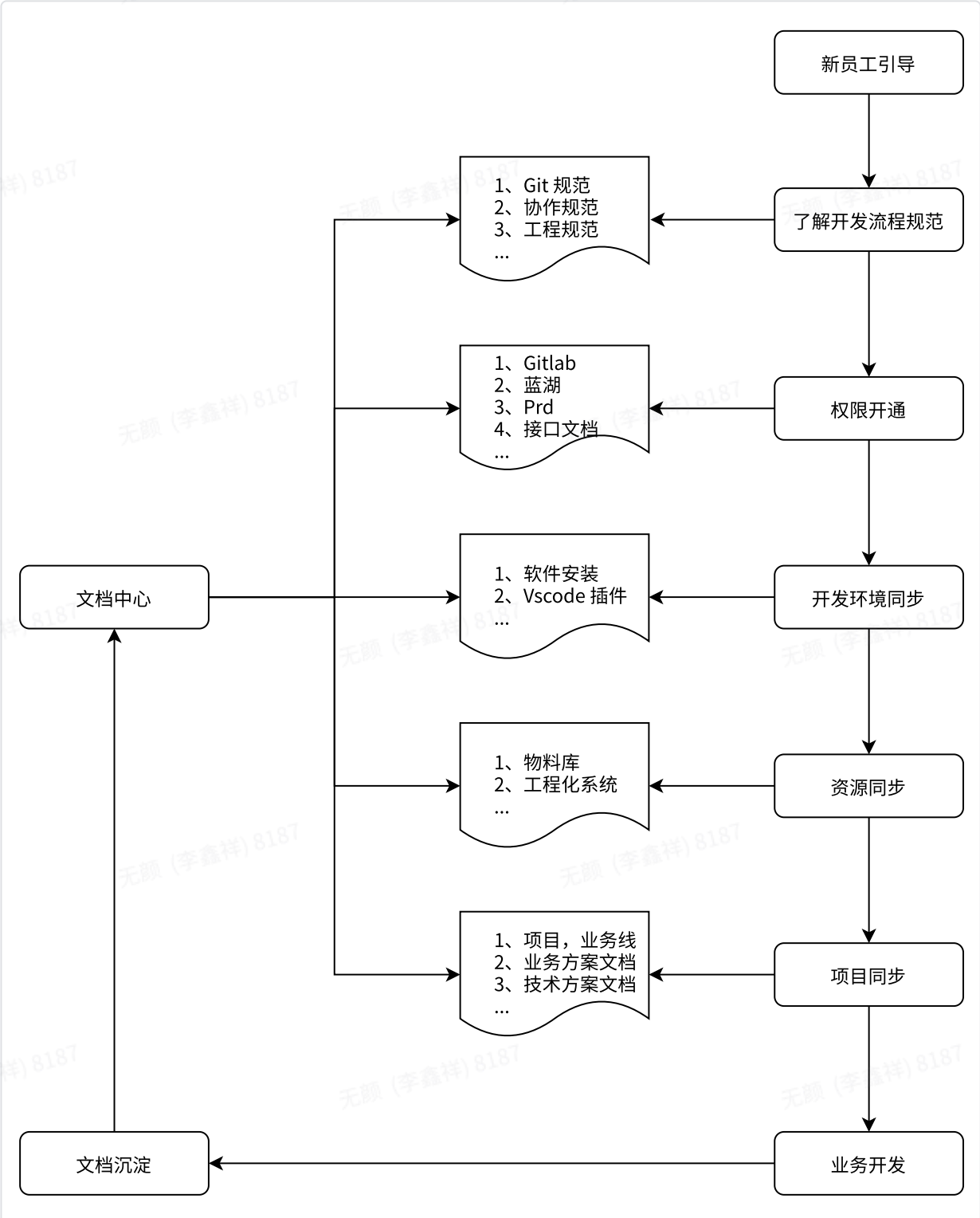
从零到一搭建基建以及基建的完善机制流程如下：


前端基建
文档中心
资源整合
开发流程
业务沉淀
规范整理
业务梳理
搭建文档中心
建立完善机制
项目整理
物料库整理
工程化整理
搭建物料中心
搭建工程中心
简历完善机制
需求提出
需求 PRD 评审
交互设计评审
技术方案设计
开发编码
联调
自测
Review
打包部署
验收
上线
数据收集
业务文档沉淀
组件沉淀
工程沉淀
构建沉淀
测试沉淀
技术文档沉淀
文档归档

2.1.3 文档中心

💡 无论前端支撑什么业务，都推荐**文档先行**

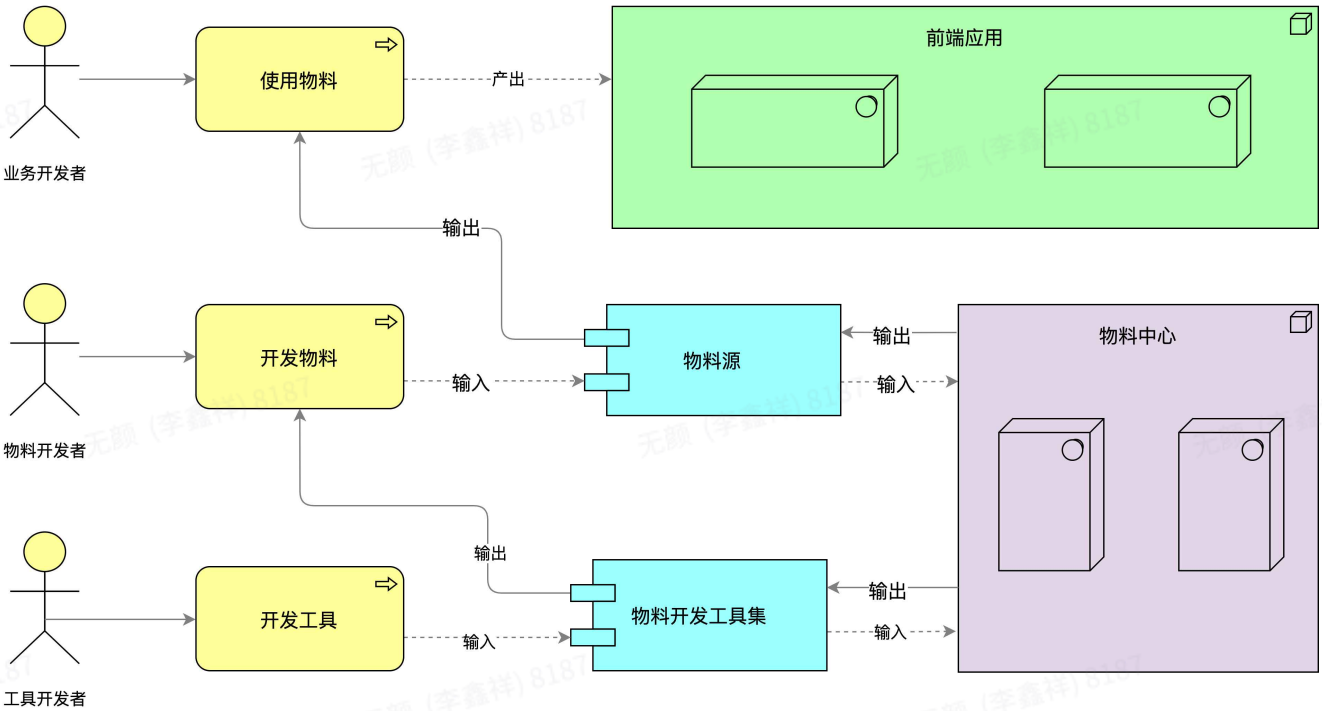
公司内部推荐使用基于飞书的文档体系，如：客管前端文档



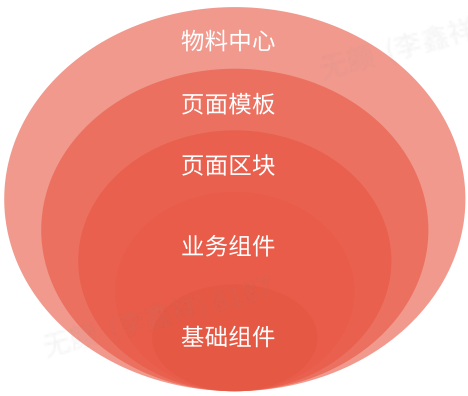
2.1.4 物料中心

> 物料中心对于微信小程序意义在于沉淀出高质量 (UI 一致、测试一致) 组件、区块、模板等

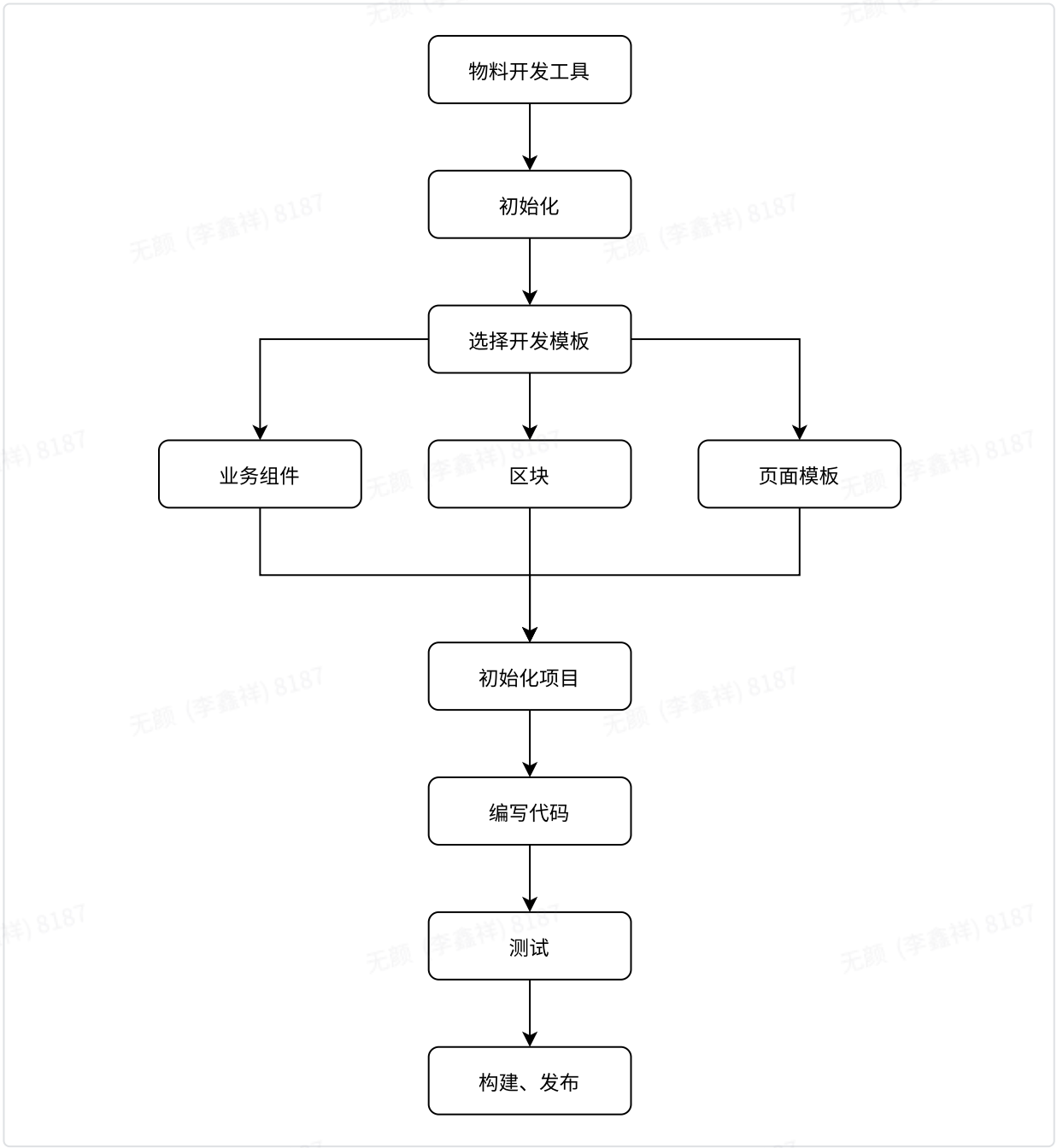
物料关系图：



物料集合：



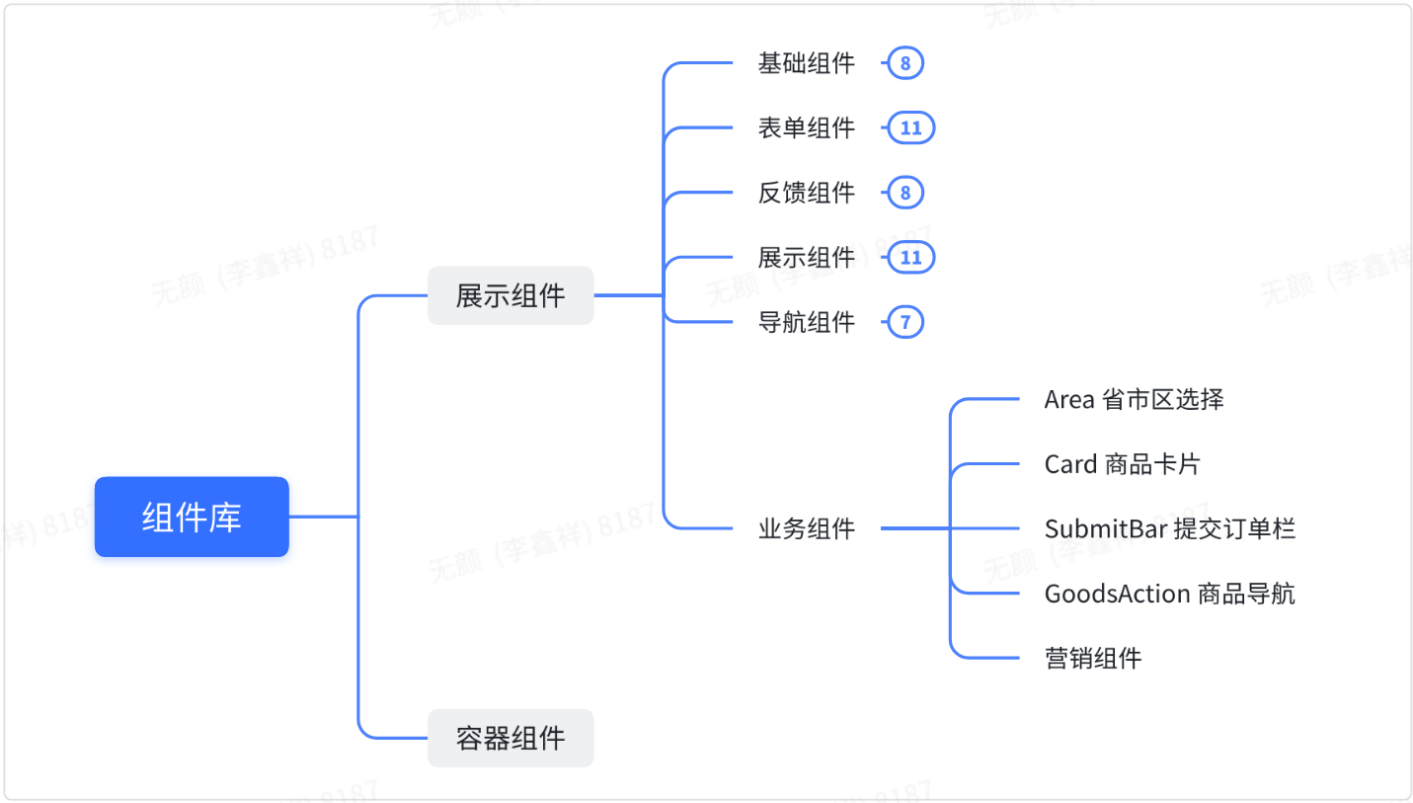
物料开发流程：



2.1.5 组件库

💡 微信小程序的组件应与 UI 侧保持一致，并且提供的 H5 相同版本，所有设计符合设计系统。

业务支撑中涉及的常用组件：



2.1.6 性能测评

谷歌为 Web 应用定义的以用户为中心的性能指标体系：

体验	指标
页面能否正常访问	首次内容绘制 (First Contentful Paint, **FCP**)
页面内容是否有用	首次有效绘制 (First Meaningful Paint, **FMP**)
页面功能是否可用	可交互时间 (Time to Interactive, **TTI**)

对于大多数小程序而言，上述指标对应的含义为：

- FCP：白屏加载结束
- FMP：首屏渲染完成
- TTI：所有内容加载完成

小程序官方性能指标：

- 首屏时间不超过 5 秒；
- 渲染时间不超过 500ms；

- 每秒调用 `setData` 的次数不超过 20 次；
- `setData` 的数据在 `JSON.stringify` 后不超过 256kb；
- 页面 WXML 节点少于 1000 个，节点树深度少于 30 层，子节点数不大于 60 个；
- 所有网络请求都在 1 秒内返回结果；

详见 小程序性能评分规则

体验评分工具：

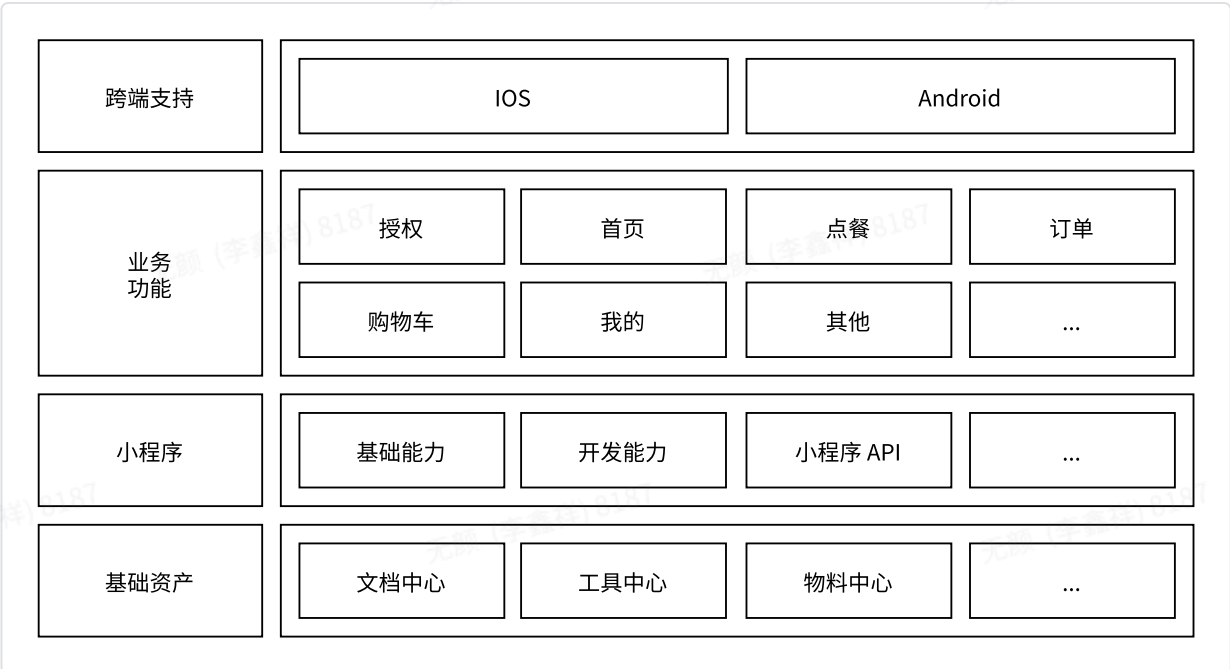
小程序提供了 体验评分工具（`Audits` 面板） 来测量上述的指标数据，其集成在开发者工具中，在小程序运行时实时检查相关问题点，并为开发者给出优化建议。

更多细节参阅

- 如何打造高性能小程序门户
- 微信原生开发与 uni-app 比较

2.2 业务支撑

面向 IOS、Android 等客户端输出微信小程序应用功能，对应架构图：

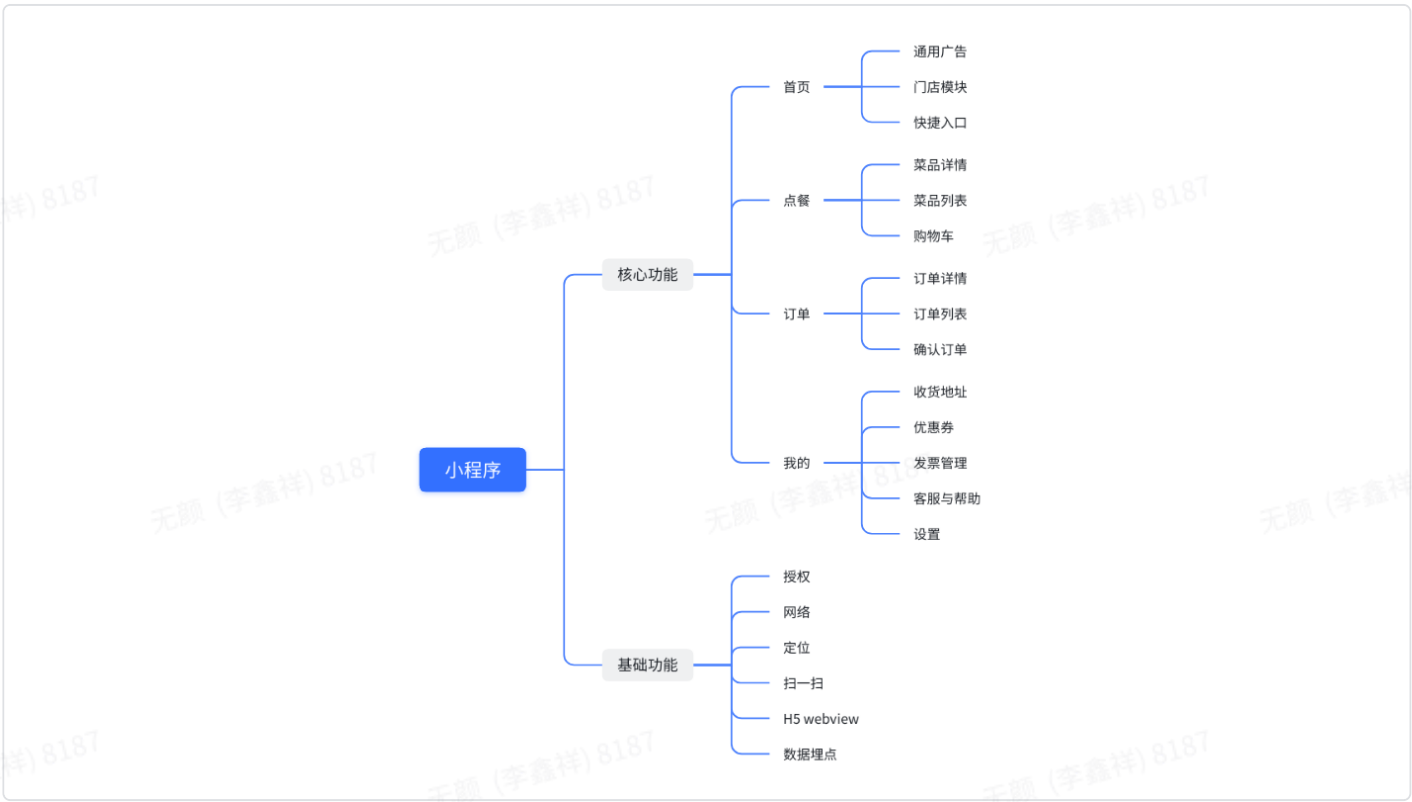


在业务开发过程中沉淀的基础资产，结合微信小程序能力，可承载餐饮行业的通用业务模型

三、核心流程

3.1 功能细分

餐饮行业功能细分如下：



3.2 登录授权

TODO

3.3 首页

TODO

3.4 点餐

TODO

3.5 订单

TODO

3.6 我的

TODO

四、关键技术

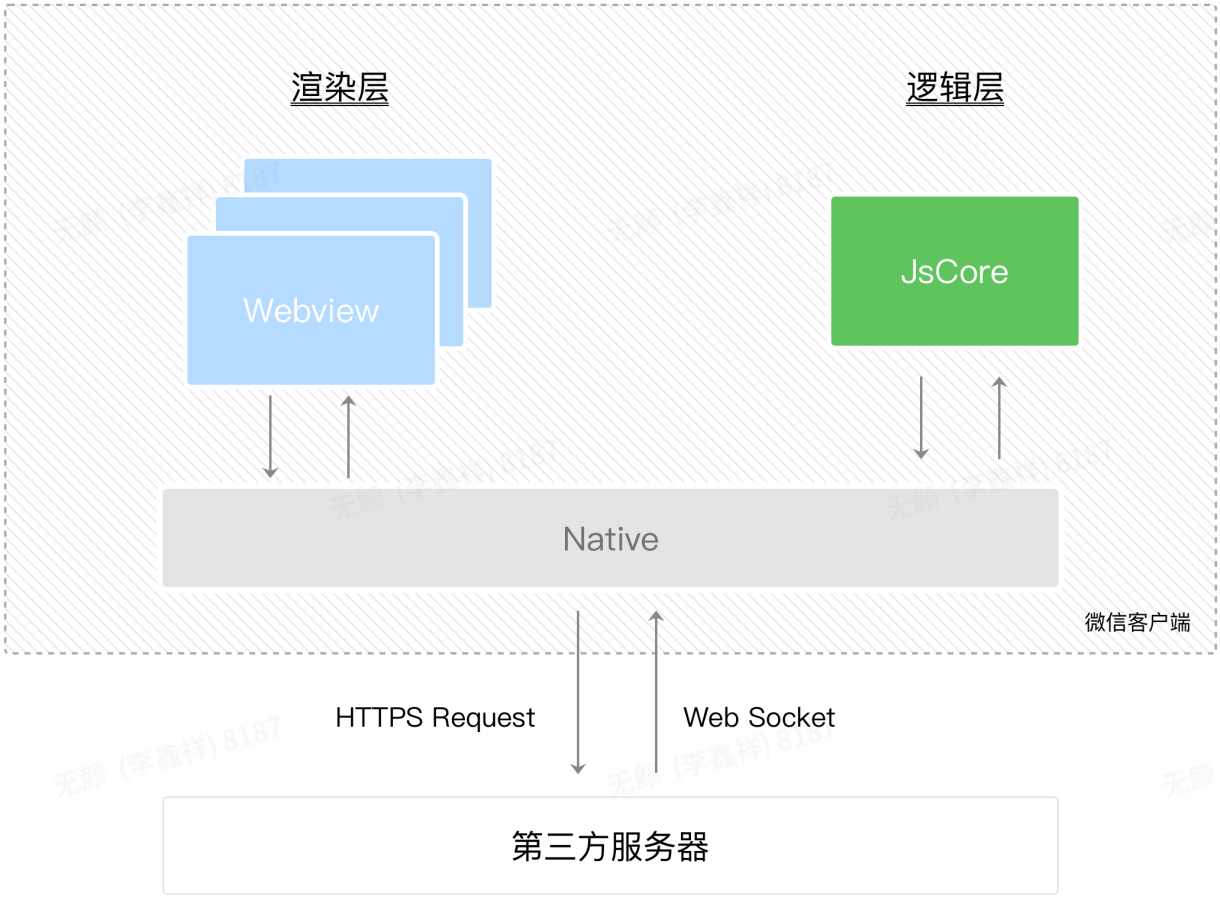
4.1 微信小程序性能提升

> 💡 **本节仅对必备性能调优加以说明，社区框架对性能调优都有自己的方案，如果采用微信原生语法进行开发，开发者应跳出框架着重关注自定义的性能调优。**
>
> 微信小程序官方开发指南

微信小程序是大前端跨平台技术的其中一种产物，与当下其他热门的技术 React Native、Weex、Flutter 等不同，小程序的最终渲染载体依然是浏览器内核，而不是原生客户端。

而对于传统的网页来说，UI 渲染和 JS 脚本是在同一个线程中执行，所以经常会出现 “阻塞” 行为。微信小程序基于性能的考虑，启用了**双线程模型**：

- **视图层**：也就是 webview 线程，负责启用不同的 webview 来渲染不同的小程序页面；
- **逻辑层**：一个单独的线程执行 JS 代码，可以控制视图层的逻辑；



业务开发过程，可能会遇到如下问题：

- 小程序启动慢
- 白屏时间长
- 页面渲染慢
- 运行内存不足

下面从几个关键阶段来针对性地给出解决方案。

启动阶段：

- 准备运行环境
- 下载小程序代码包
- 加载小程序代码包
- 初始化小程序首页

启动阶段提升性能方式：（推荐阅读：小程序工程探索）

- JS、CSS Tree-Sharking

- 减少代码包中的静态资源文件
- 逻辑后移，精简业务逻辑
- 复用模板插件
- 分包加载
- 部分页面 H5 化 (如营销 H5) 小程序开发文档

白屏阶段：

- 代码包下载完成
- FMP (首次有效绘制)，影响因素：网络资源加载时间、渲染时间

白屏阶段提升性能方式：(推荐阅读：小程序运行机制)

- 启用本地缓存
- 数据预拉取
- 跳转时预拉取
- 分包预下载
- 非关键渲染数据延迟请求 (建议了解：关键渲染路径)
- 接口聚合，请求合并
- 图片资源优化
- 骨架屏

渲染阶段：

- 初始化 webview 线程环境、基础库
- 逻辑层到视图层的数据初始化
- 数据驱动视图 (WXML、WXSS 结合完成页面渲染)

渲染阶段提升性能方式：

- 合并 `setData` 调用
- 控制 `setData` 数据量
- 应用层数据 diff
- 控制事件绑定、节点属性
- 控制组件颗粒度
- 组件层面的 diff

其他提升性能方式：

- 内存预警
- 合理控制计时器
- 防抖节流
- 大图、长列表优化

4.2 uni-app 技术细节

五、时间计划

5.1 基建计划

基建任务拆分如下：

- [ ] 文档中心 ⏰ 9月30日 14:30
  - [ ] 团队规范
    - [x] ~~代码规范~~
    - [ ] 组件规范
    - [ ] 设计系统规范
    - [ ] 接口对接规范
  - [ ] 组件文档
    - [ ] 组件管理文档 (开发与迭代)
    - [ ] 展示组件 API 与示例
    - [ ] 业务组件 API 与示例
  - [ ] 工具文档
    - [ ] 辅助函数 (通用工具函数、hooks 等)
    - [ ] 物料开发 CLI、组件开发 CLI 等

- [ ] 物料中心 ⏰ 10月30日 14:30
  - [ ] 组件库 (展示组件、业务组件)

- [ ] 基础组件 (Button、Cell 等定制)
- [ ] 表单组件 (Search、Field 等定制)
- [ ] 导航组件 (Tab、Tabbar 等定制)
- [ ] 反馈组件 (ActionSheet、Loading 加载等定制)
- [ ] 展示组件 (Progress、Steps、Skeleton 等定制)
- [ ] 业务组件 (Area 省市区选择、Card 商品卡片等定制)

- [ ] 图标库 (基础图标、业务图标)
- [ ] Block、Templates
- [ ] Vscode 代码片段、插件等

- [ ] 工具中心 ⏰ 10月30日 14:30
  - [ ] Mock 工具
  - [ ] 调试工具 (日志增强)
  - [ ] 辅助函数
  - [ ] CLI 工具集
  - [ ] 符合设计系统的主题工具

5.2 业务计划 (TODO)

业务功能任务拆分如下：

- [ ] 授权
- [ ] 首页
  - [ ] 通用广告
  - [ ] 门店模块
  - [ ] 快捷入口
- [ ] 点餐
  - [ ] 菜品列表
  - [ ] 菜品详情
  - [ ] 购物车
- [ ] 订单
  - [ ] 订单列表

- [ ] 订单详情
- [ ] 确认订单

- [ ] 我的
  - [ ] 收货地址
  - [ ] 优惠券
  - [ ] 发票管理
  - [ ] 客服与帮助
  - [ ] 设置

- [ ] 其他
  - [ ] 网络
  - [ ] 定位
  - [ ] 扫一扫
  - [ ] H5 webview

六、参考资料

- 微信小程序开发文档
- 小程序开发：用原生还是选框架（wepy/mpvue/taro/uni-app）
- 跨端开发框架深度横评之2020版