

前端工程化实践

分享人：魏华亮




1、为什么需要微前端

2、什么是微前端

3、微前端的实现方式

4、微前端实践分享



为什么需要微前端

micro frontend

△ 为什么需要微前端

遗留系统改造

遗留系统架构不符合当前要求，
需要进行重构改造

统一入口

产品功能聚合，统一接入



多团队解耦

降低维护成本，提高开发效率

技术预研

针对新技术新框架在真实项目
进行预研，而不影响原有功能



什么是微前端

micro frontend

什么是微前端

由多个独立应用聚合而成的大型单页应用，每个前端应用可独立开发、独立部署。



应用A

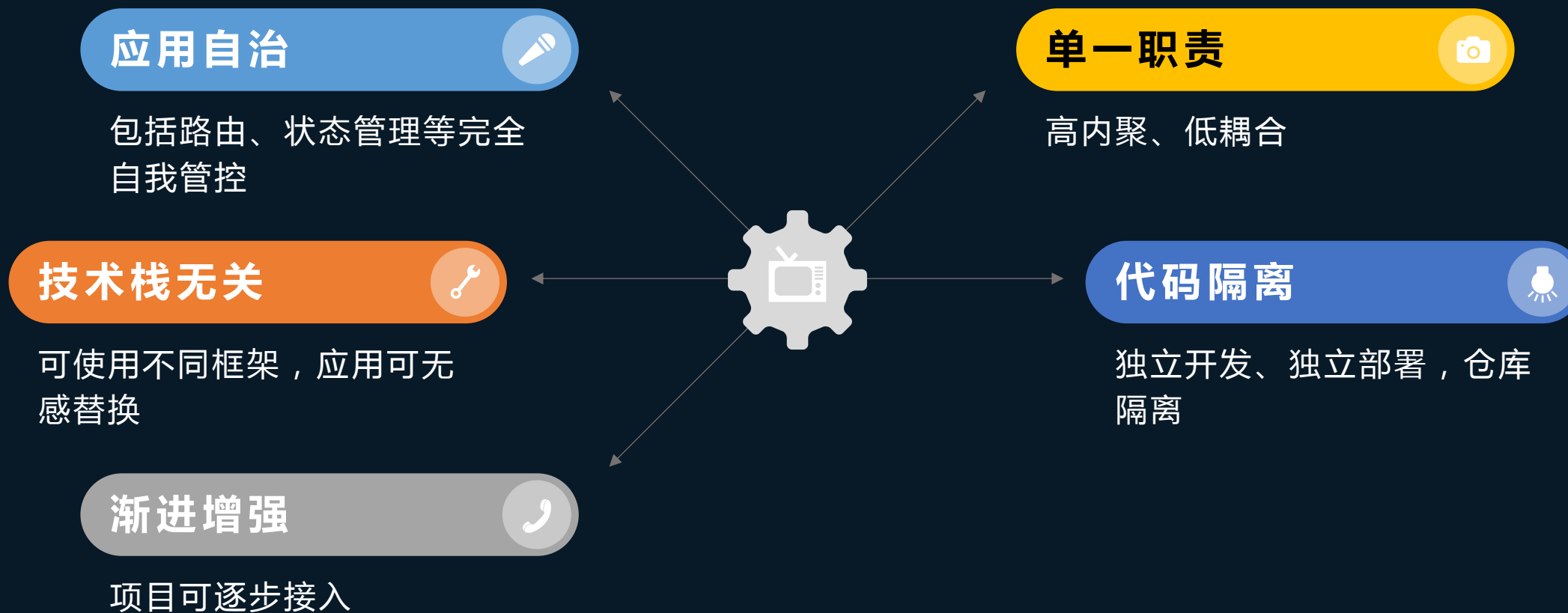


应用B



应用C

△ 什么是微前端-特点





微前端的实现方式

micro frontend

微前端的实现方式

路由分发

通过服务器的反向代理能力，将路由请求到对应的服务上

iframe嵌入

通过主应用来创建iframe作为容器加载子应用

构建时微应用

通过工程化的方式，将多个独立应用组合成一个前端应用

运行时微件化

编译好并上传，可直接嵌入应用上运行的代码块

微服务化

每个应用都是完全独立（技术栈、开发、构建、部署），自主运行，通过模块化的方式进行组合

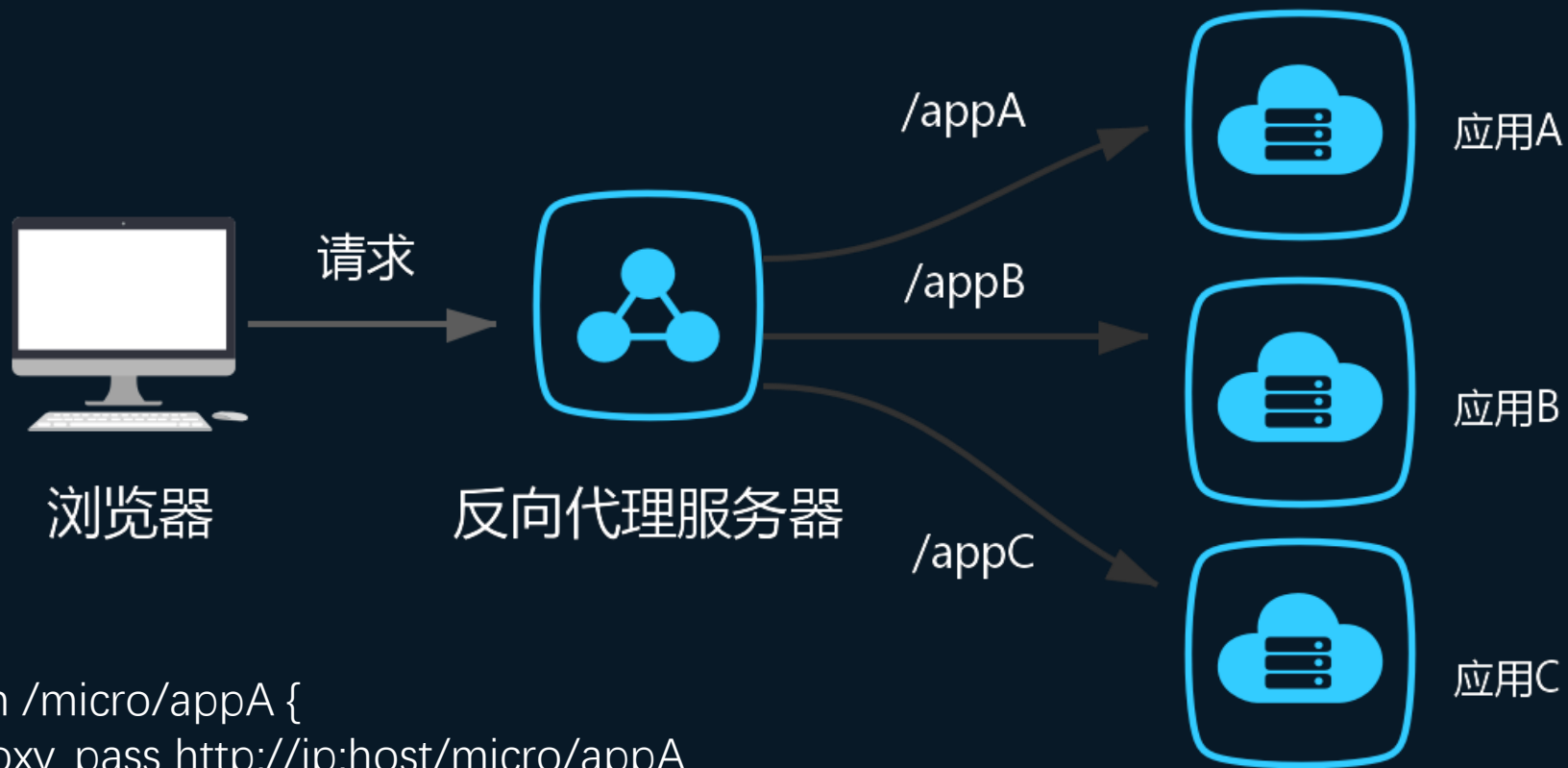
基础设施

团队规范、基础组件、基础库等





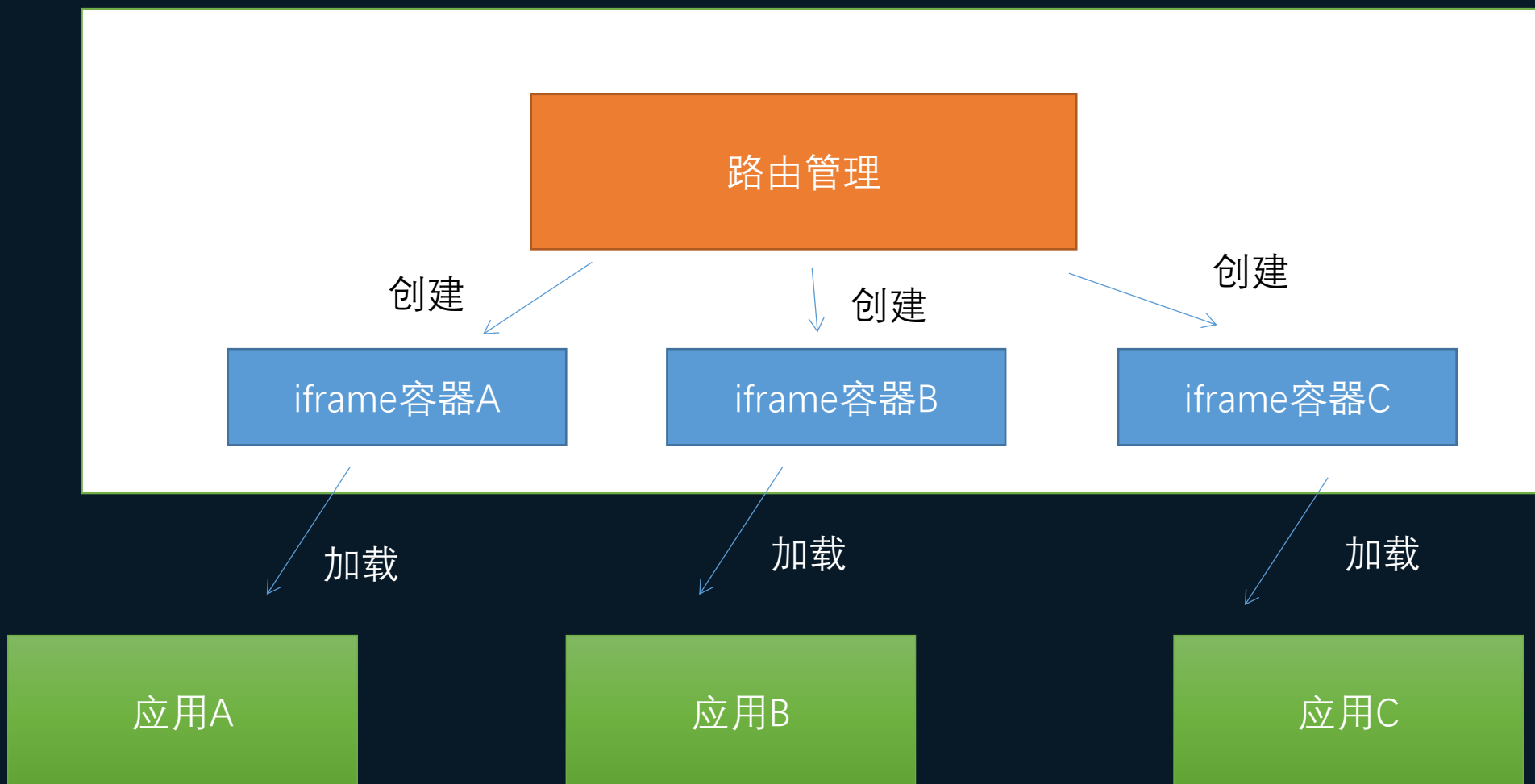
微前端的实现方式—路由分发



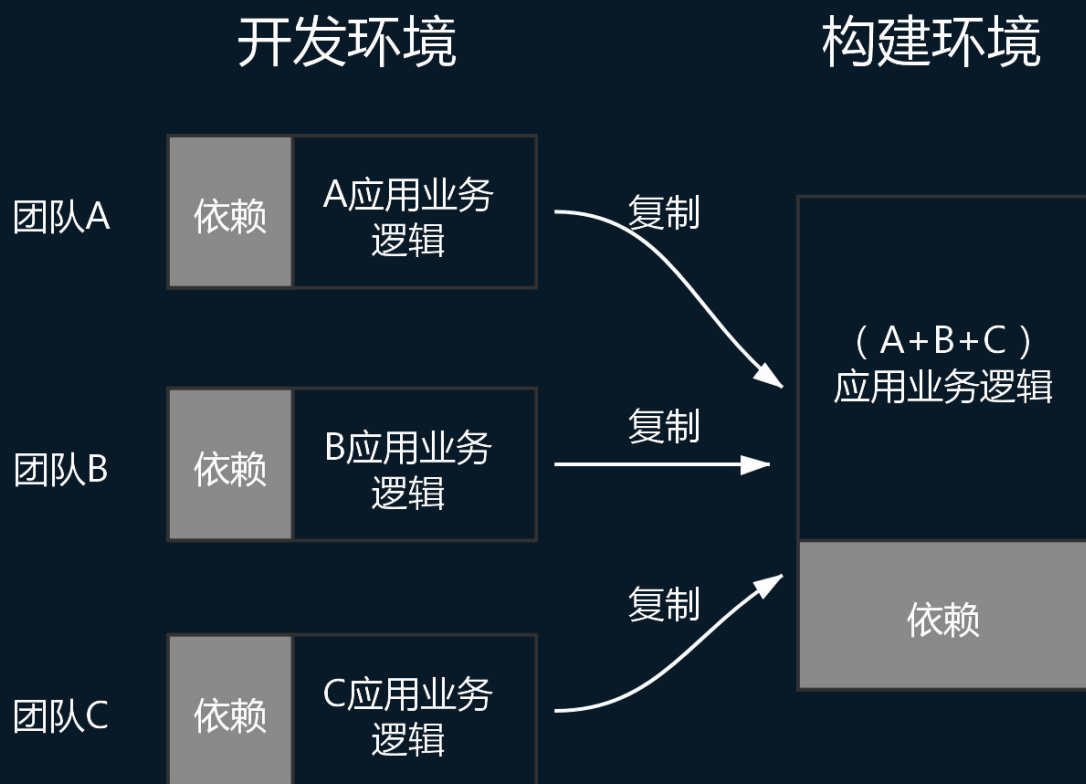
```
location /micro/appA {  
    proxy_pass http://ip:host/micro/appA  
}
```



微前端的实现方式-iframe嵌入



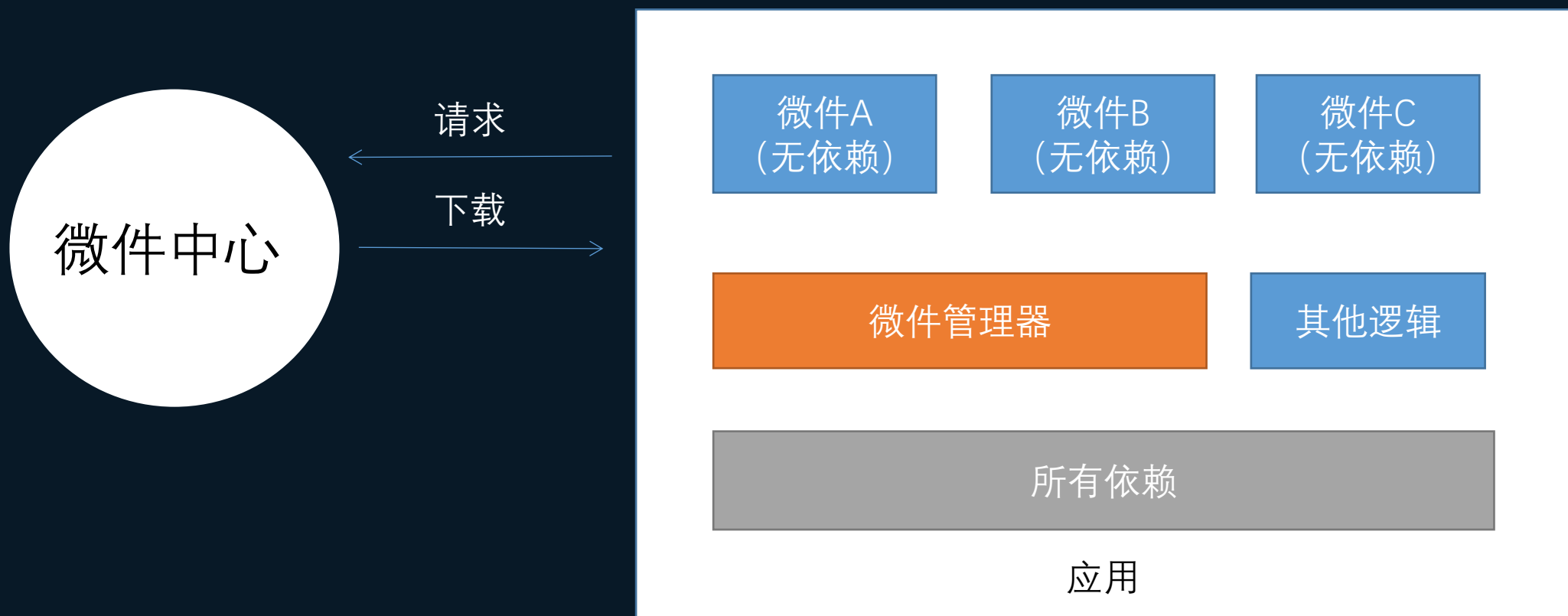
微前端的实现方式—构建时微应用



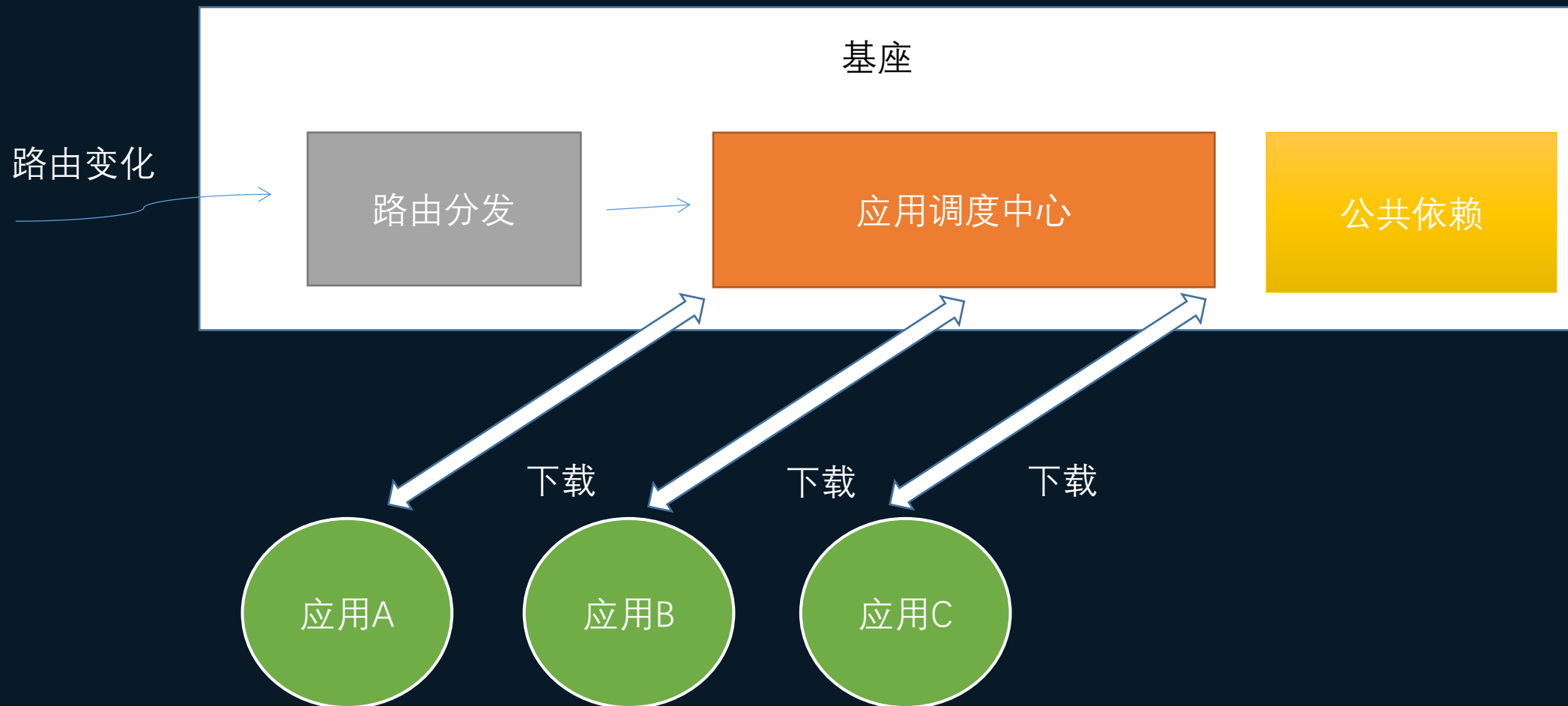
在开发时应用都以单一应用的形式存在，而在构建时，通过构建工具合并一个应用，组合成一个新的应用



微前端的实现方式-运行时微件化



△ 微前端的实现方式—微服务化



微前端的实现方式一对比

方式	开发成本	维护成本	可行性	同一框架要求	难度	风险
路由分发	低	低	高	否	低	体验较差，实际为多页应用，应用间跳转需要刷新页面，重新加载所有资源 同一个页面仅可存在一个应用
iframe嵌入	低	低	高	否	低	SEO不友好 打开多个iframe内存占用过大
构建时微应用	中	中	中	是	中	高度依赖构建工具和部署流程
运行时微件化	高	低	中	是	高	需要提供微件的调度系统和框架API
微服务化	高	中	中	否	高	需要为不同框架单独设计通信和加载机制，构建复杂度较高

△ 微前端的实现方式—基础设施

应用配置表

维护各个应用的入口、权限等相关信息，用于应用注册和应用发现

应用数据共享

对于通用的数据，采取一定策略来缓存数据

不同应用作用域隔离

对不同应用进行隔离，防止全局污染

应用生命周期管理

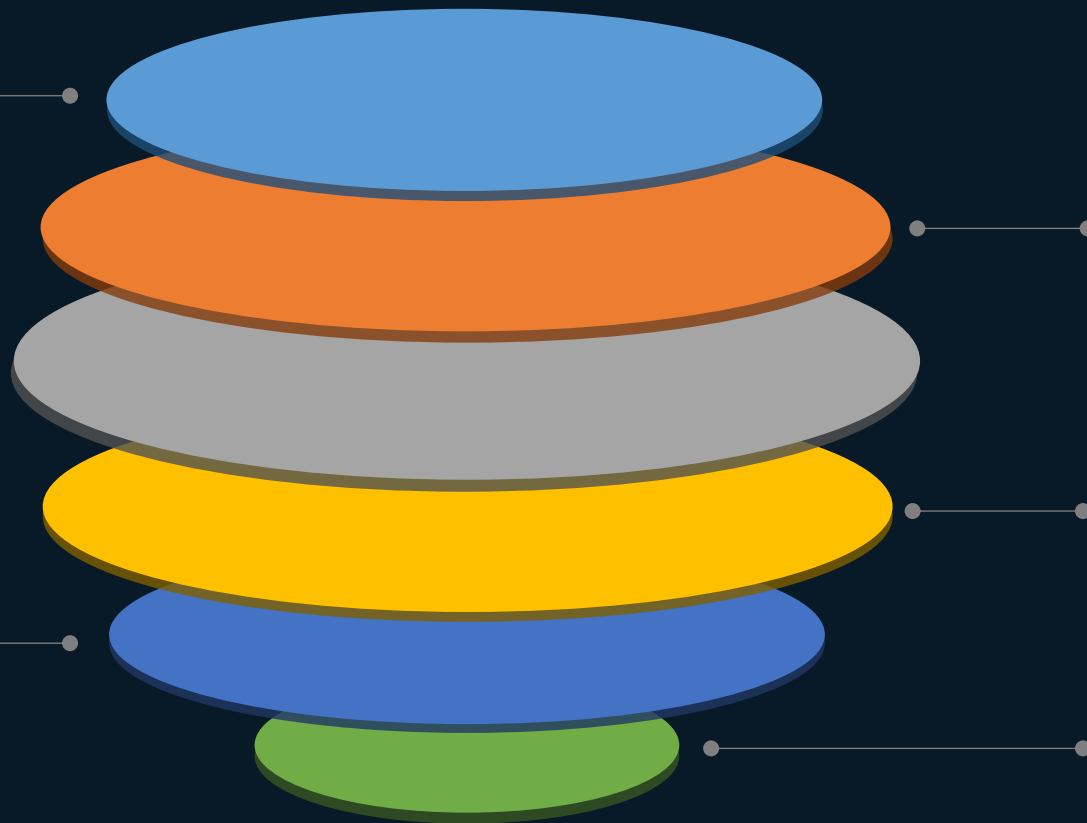
管理独立应用的加载、运行、卸载

应用通信

应用间的通信机制，提供底层库进行支持

子应用规范划分

良好的划分方式，利于解耦





微前端的实现方式-应用注册表

服务注册中心，服务提供方要注册告知服务地址，服务调用方才能发现目标服务。

应用注册表拥有每个应用及对应的入口，入口的直接表现形式可以是路由，或者对应的应用映射。

应用在构建完成或者部署到服务器后，应该在这个应用注册表中进行注册。



微前端的实现方式—应用生命周期

01

加载应用

决定加载哪个应用，
通过应用注册表中找
到对应应用，获取静
态资源，并安装应用。
例如挂载dom。

02

运行应用

应用自我管理

03

卸载应用

删除应用的生命周
期，卸载应用，例
如删除dom、取消
事件绑定



微前端的实现方式—应用数据共享策略

01

- 数据交互越多、耦合程度越高，
- 最小使用原则进行共享

02

中心化数据管理策略

- 设计安全数据存取方式
- 严格控制不同应用数据修改权限

03

常见的数据交互方式

- URI参数传递
- Storage共享
- 客户端存储，例如IndexDB、WEB SQL
- 服务端存储



微前端的实现方式—应用通信机制

同级通信

挂载在同一个
HTMLDocument下的应用
通信

- 通过全局的自定义事件（CustomEvent）或者通过自定义发布-订阅模式组件

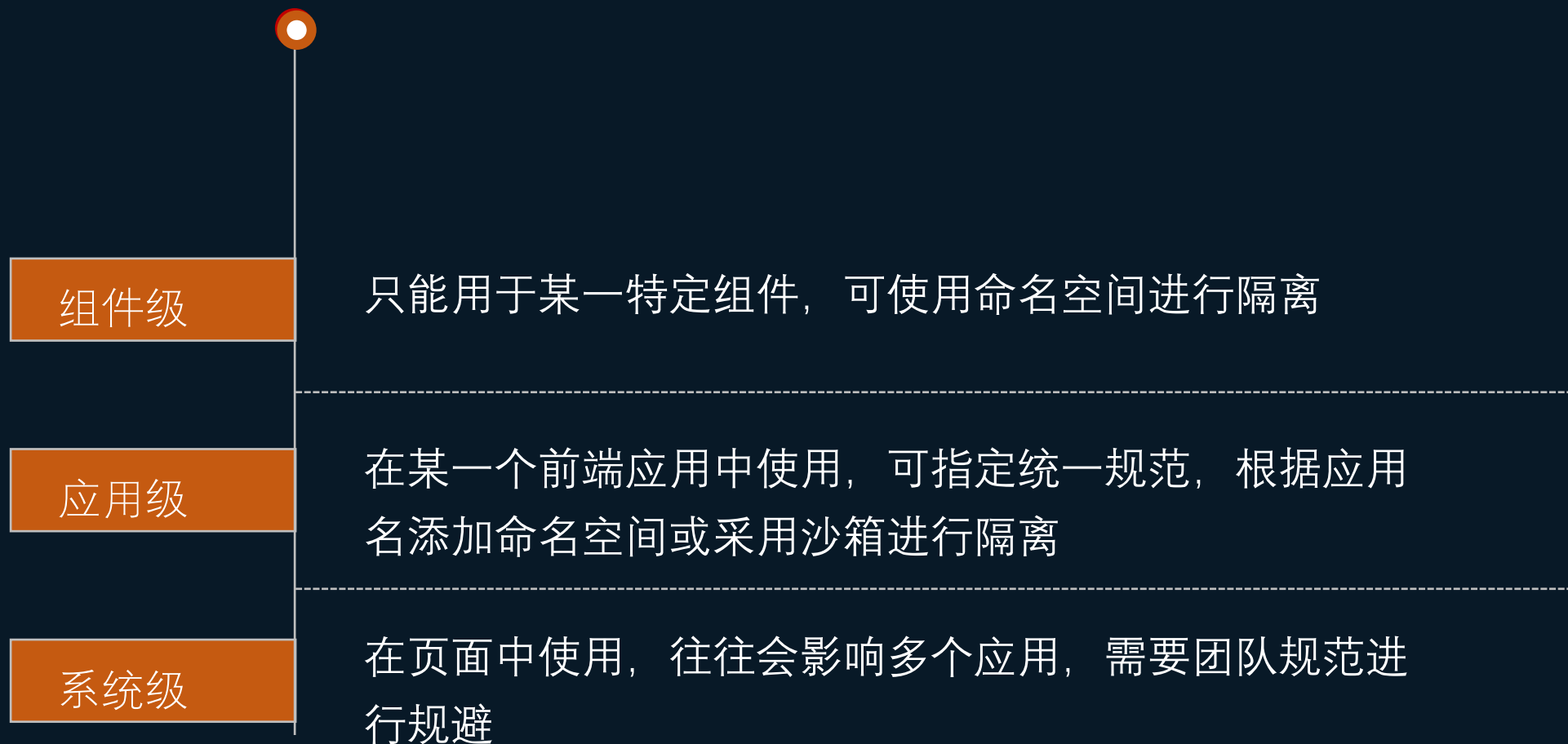
父子级通信

采用iframe形式来挂载其他应用

- 通过postMessage在父子窗口间进行通信
- 通过parent.window寻找父窗口，再发出全局的自定义事件
- 通过父窗口进行中转



微前端的实现方式—作用域隔离





微前端的实现方式—子应用规划规范

- ✓ 根据业务流程进行划分
- ✓ 根据前端领域模型进行划分
- ✓ 参考后端微服务进行划分
- ✓ 根据功能变更是否频繁划分
- ✓ 根据权限进行划分
- ✓ 根据开发团队进行划分



微前端的实现方式—架构模式

基座模式

基座承担微前端应用的基础，由一个主应用和一系列业务子应用组成，并由这个主应用来管理其他子应用，包括子应用的生命周期管理和通信

自组织模式

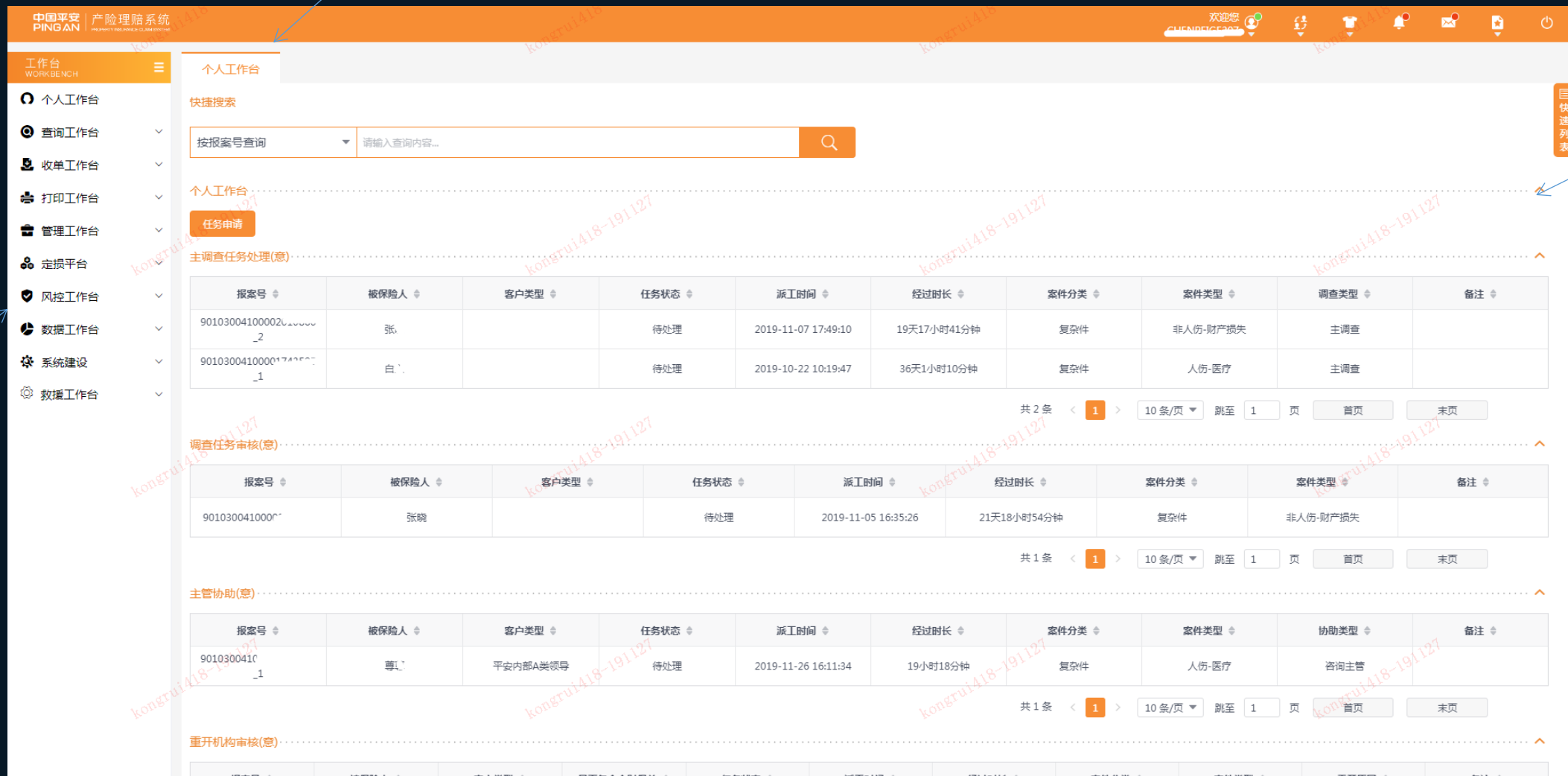
系统内部各子系统之间可以自行按照某种规则形成一定的功能，每个子系统都有一个小型的基座管理功能



微前端实践分享

micro frontend

快速列表





微前端实践分享-案例一项目特点



多团队协作

- 不同模块属于多个团队



启动周期短

- 需要快速上线



多应用共存

- 一个页面需要运行多个应用



技术栈统一

- 新项目，所有团队技术栈统一



统一入口

- B端产品，统一入口

—



微前端实践分享-案例一项目架构

统一登录

登录、注销

业务逻辑

菜单管理

TAB管理

权限管理

路由管理

通信代理

消息管理

换肤

子应用

业务逻辑

子应用菜单

状态管理

依赖

消息处理

... ..

共享依赖

微前端实践分享-案例二



两条独立的业务



两条独立的业务



微前端实践分享-案例二项目特点



多团队协作

- 不同模块属于多个团队



移动端

- 性能、用户体验要求高



多应用共存

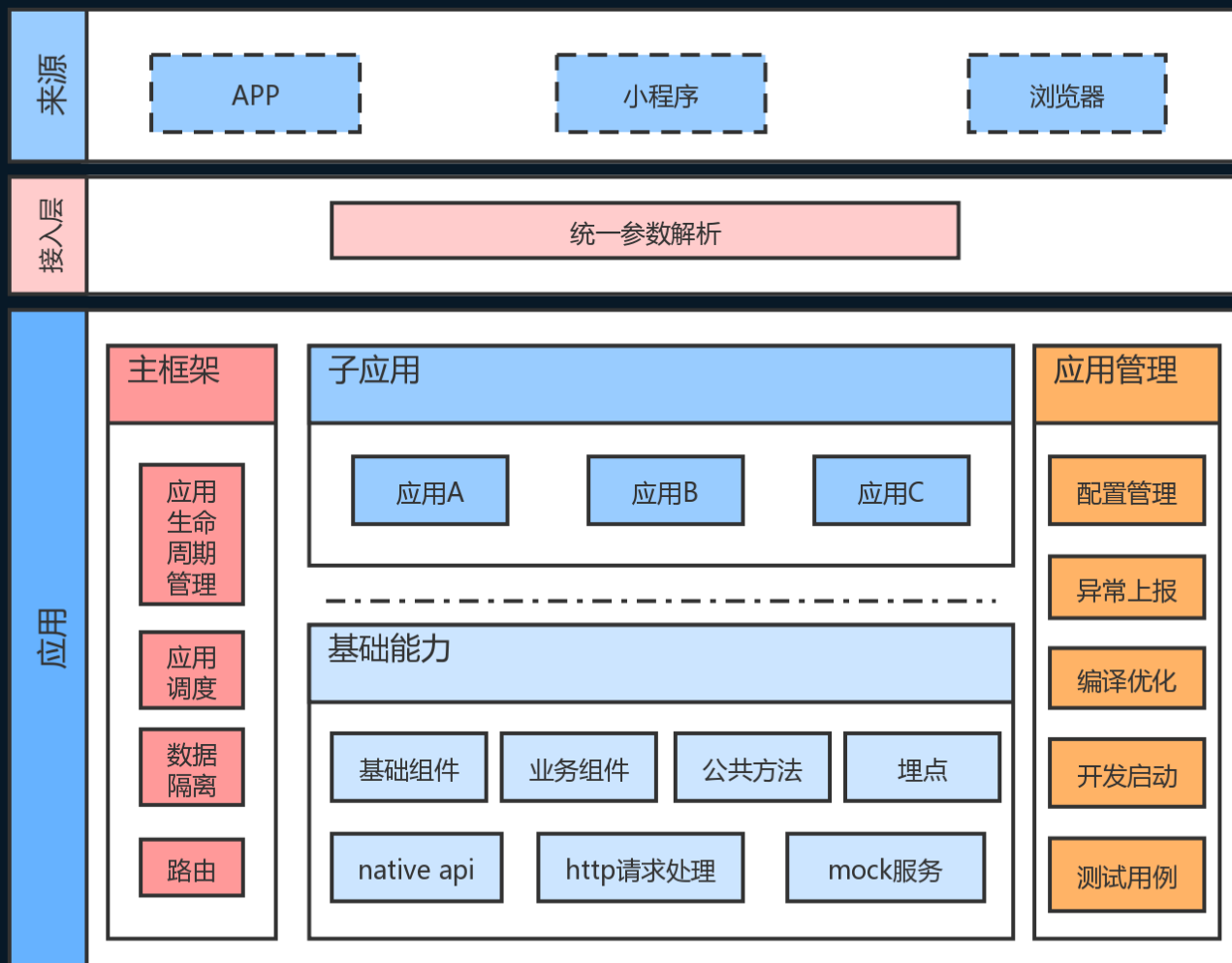
- 一个页面需要运行多个应用



跨平台

- 可运行在多个APP、浏览器、小程序

微前端实践分享-案例二架构图





微前端是一种架构方案，对团队的技术沉淀要求更高。

一切技术方案都是为了解决当前和可预见的未来的问题，并非一成不变。

A wireframe sphere, composed of a network of white lines and dots, floats in the center of the image. The background is a deep blue night sky filled with stars and a faint Milky Way. Below the sphere, a horizon line separates the dark sky from a vast expanse of white, fluffy clouds. The sphere's reflection is visible in the clouds below.

Q & A



谢谢聆听

THANKS