























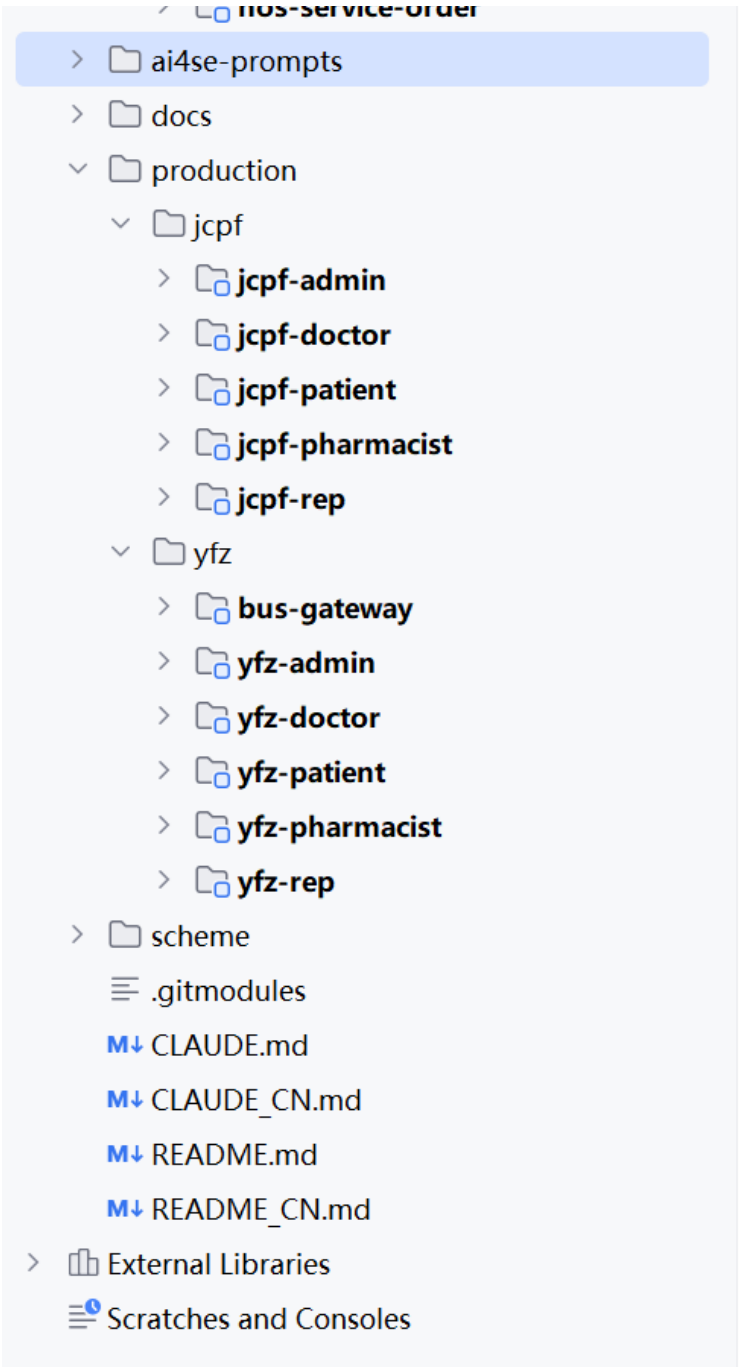


互医平台的解决方案与架构图。

I、项目背景介绍

▼  all D:\Users\develop\11\all	1
>  .claude	
>  .idea	
▼  micro-service	
▼  jcpf	
>  jcpf-base-lib	
>  jcpf-ms-base	
>  jcpf-ms-cms	
>  jcpf-ms-drug	
>  jcpf-ms-his	
>  jcpf-ms-mdm	
>  jcpf-ms-order	
>  jcpf-ms-rp	
>  jcpf-ms-third	
▼  yfz	
>  hos-drug-order	
>  hos-gateway	
>  hos-ms-base-center	
>  hos-ms-cdm	
>  hos-ms-cms	
>  hos-ms-qa	
>  hos-ms-rp	
>  hos-ms-third	
>  hos-service-order	



1、@production\jcpf\ 一共包含五个端的应用：  
分别是 jcpf-admin-京城管理端，jcpf-doctor-京城医生端，jcpf-patient-京城患者端，jcpf-pharmacist-京城药师端，jcpf-rep-京城代表端，这五个端对应的所有微服务调用

项目都在 @micro-service\jcpf\目录下

2、@production\yfb\ 一共包含六个端的应用：

分别是 yfb-admin-易赋诊管理端， yfb-doctor-京城医生端， yfb-patient-京城患者端， yfb-pharmacist-易赋诊药师端，  
yfb-rep-易赋诊代表端， bus-gateway-易赋诊定制化需求开发（其中yfb-易赋诊定制化需求开发， zs-正生业务定制化开发）

这六个端对应的所有微服务调用项目都在 @micro-service\yfb\目录下

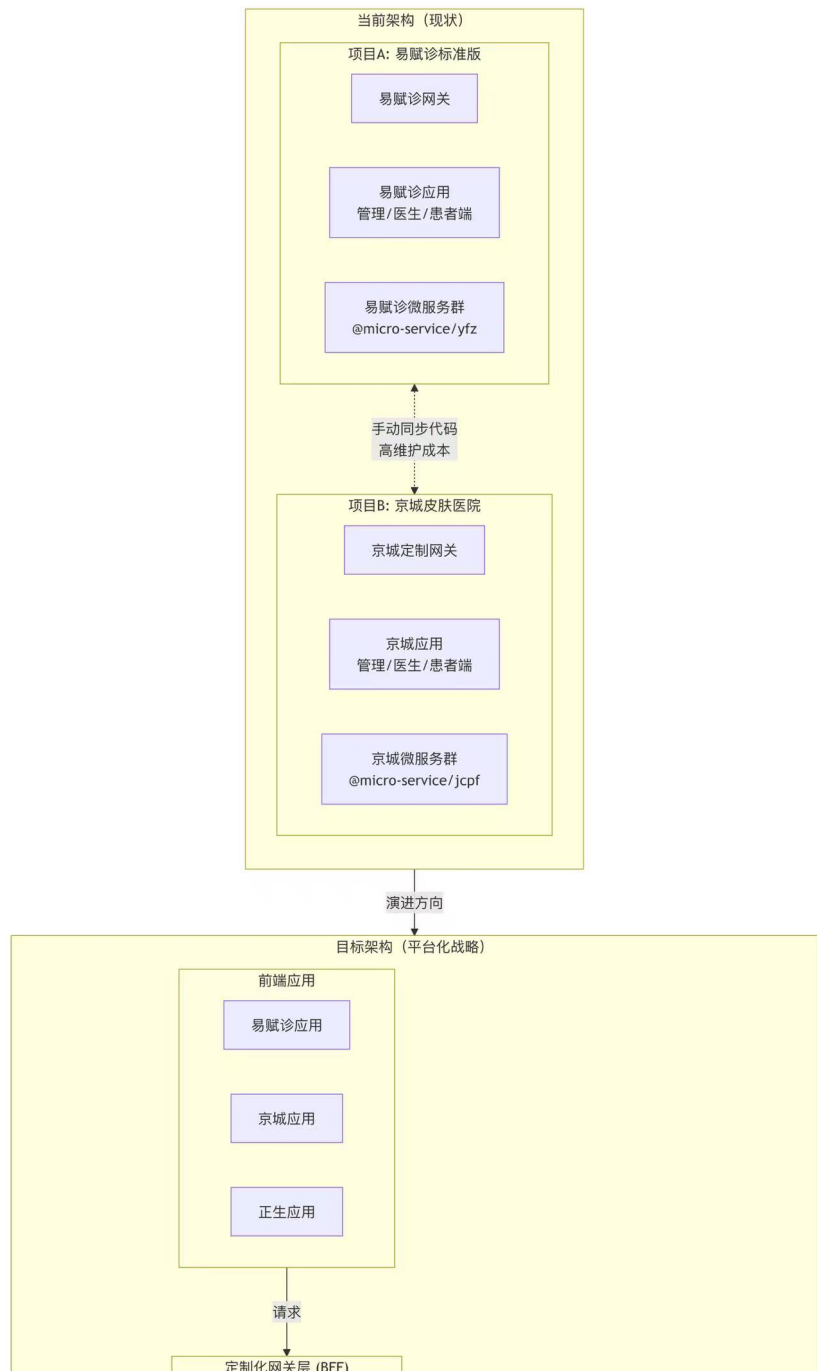
3、最开始是先有的2中项目，也就是易赋诊的项目，我们是以5个端作为一个api入口，去整合业务的，每个端应用的数据库基础服务和三方服务以微服务的形式对端进行暴露，在端上使用openfeign的形式引入微服务并调用，平时的产品都是标准化版本迭代，适用于易赋诊平台所有的租户，这套系统同时支持多租户，并且我们的用户账号是统一数据库表管理的，所有数据都是基于租户id进行传输隔离。

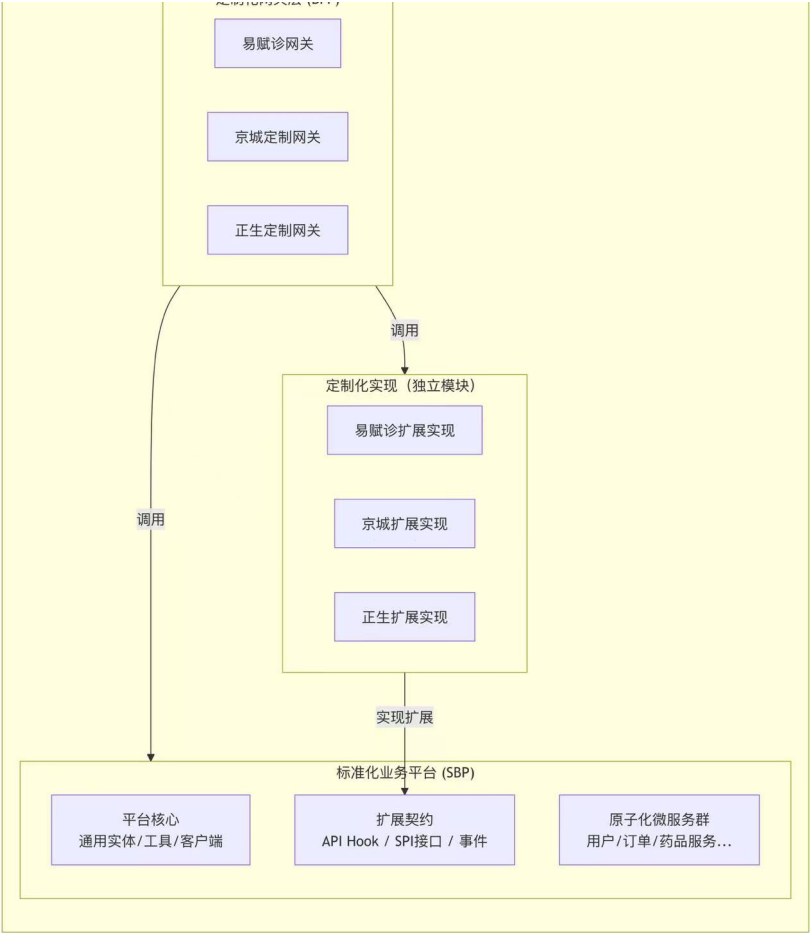
4、后来京城皮肤医院是在易赋诊这套模式的基础上，我们单独把所有的端和微服务拷贝出来单独做了一份，叫智慧医院的定制化开发服务。这个项目是单租户的，因为京城医院对接his系统，所有数据都强依赖于his系统，所以我们在程序和数据库上做了强隔离，但是这也后来形成了一套单独的技术方案，我们后续再对接其他智慧医院，只需要替换底层his供应商，上层业务以及微服务不需要大改。

5、易赋诊有一个定制化的需求开发，那就是正生项目， 这个项目目前是在bus-gateway/zs这个模块下，其所有的外部调用均依赖自易赋诊的标准版微服务，所有定制化部分（非易赋诊标准版功能）均在网关层实现。

3、目前互医这套技术方案主要围绕易赋诊和京城皮肤应对标准化功能迭代、易赋诊定制化开发以及进行多租户管理的

II、互医平台的架构演进





1. 当前架构 (现状)
- 问题：易赋诊和京城项目是两套独立且几乎相同的代码库，通过手动同步来尝试保证功能一致，存在一定的维护成本和分支污染风险。
2. 每个业务架构遵循的原则 (平台化战略)
- 标准化业务平台 (SBP)：成为所有业务的唯一核心。提供高度抽象的原子化微服务（如用户、订单服务）和统一的扩展契约（Hook、SPI、事件）。
  - 定制化实现：各项目（易赋诊、京城、正生）的独特逻辑从平台中剥离，以独立模块的形式存在，通过实现平台的扩展契约接口来完成定制，杜绝了修改平台核心代码的可能。
  - 定制化网关 (BFF)：为不同的前端应用提供量身定制的API聚合与适配。它同时调用SBP的标准API和定制化

模块的独有API，是处理前端个性化需求的理想场所。

- 清晰依赖：所有定制化模块和网关都依赖SBP，而SBP不依赖任何定制化模块，保证了核心平台的稳定和独立。依赖关系从“网状”变为“星型”，这样的话可以降低了一定的软件工程复杂度。

3.业务融合(未来)（易赋诊、京城、正生...）

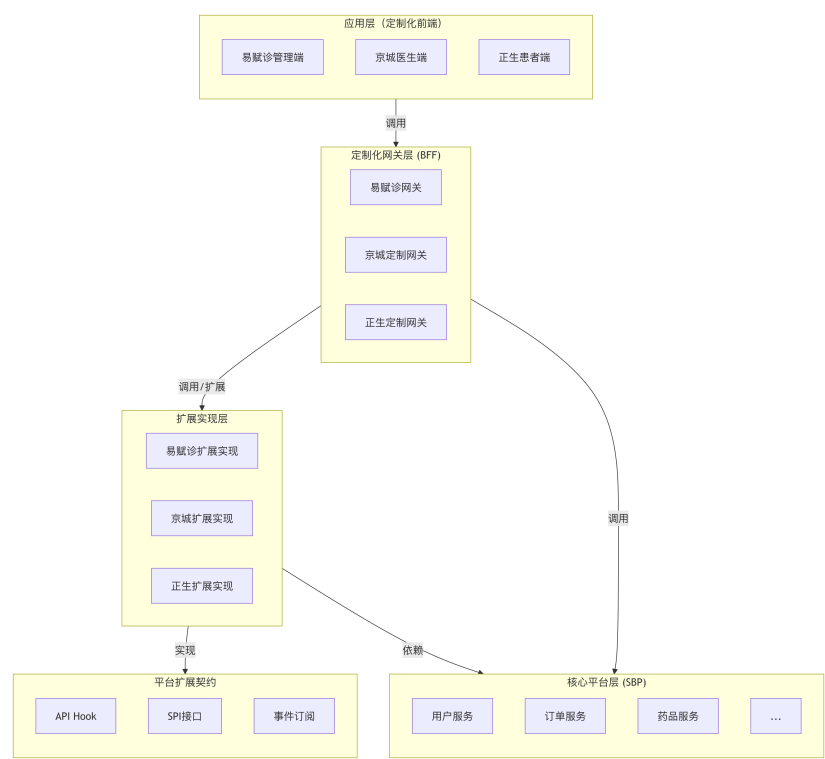
III、互医的解决方案：平台化战略

核心思想：从“项目集合”彻底演进为“一个标准化平台 + N个个性化解决方案”的架构模式。杜绝任何形式的代码复制，通过平台强大的抽象能力和可扩展性来支撑标准化和定制化需求开发。

---

一、架构图

架构如下图所示，主要展示了平台核心、扩展机制以及定制应用之间的关系：



二、核心组件详解

1. 标准化业务平台 (SBP - Standard Business Platform)

- 定位：整个架构的基石和引擎。它不再属于任何特定客户，而是提供医疗领域高度抽象的原子化业务能力（如用户管理、订单创建、处方审核等）。
- 技术实现：
  - 将现有@micro-service/yfz/和@micro-service/jcpf/中的共性服务抽离、重构而成。
  - 采用“接口与实现分离”原则。平台只定义核心业务模型和标准接口。
  - 包含一个强大的扩展点框架（如下述“扩展契约”）。

## 2. 平台扩展契约 (Platform Extension Contract)

这是连接平台与定制化的“插座”，是解决方案的核心。

- API Hook (南北向扩展): 允许在标准API的执行链路上（如/api/orders/beforeCreate, /api/orders/afterCreate）注入自定义逻辑。
- SPI接口 (Service Provider Interface, 东西向扩展): 用于彻底替换或提供平台定义的抽象接口的自定义实现。例如，定义一个HisService接口，京城项目可以提供自己的JingChengHisServiceImpl来对接其HIS系统，而易赋诊标准版提供默认的或空实现。
- 事件订阅 (Event Subscription): 平台在关键业务节点发出领域事件（如OrderCreatedEvent），定制项目可以订阅这些事件并触发自身逻辑，实现松耦合的集成。

ps: SPI 用于定义服务接口与实现类的解耦规范，允许第三方开发者通过标准化方式扩展服务功能。其核心是通过接口标准化与动态加载机制，实现服务提供者的灵活替换，为Java模块化扩展提供基础支撑

## 3. 定制化实现 (Customization Implementation)

- 定位: 各个项目（易赋诊、京城、正生）的独有逻辑存放地。它们依赖SBP，并通过实现上述“扩展契约”来完成所有定制工作。
- 技术实现:
  - 每个项目是一个独立的Maven模块或Git仓库（如 repository/customization-jc, repository/customization-zs）。
  - 其中包含: SPI实现类、Hook处理器、事件监听器以



及完全独立的Controller和DTO（用于提供标准平台之外的额外API）。

#### 4. 定制化网关 (BFF Gateway)

- 定位：为特定项目或特定端提供聚合和适配接口的角色。
- 技术实现：
  - 京城或正生的端应用，不再直接调用SBP的标准API。
  - 而是调用自己的专属网关，该网关：
    1. 调用SBP的标准API。
    2. 调用自己定制化实现模块提供的额外API。
    3. 将结果聚合后返回给前端。
  - 此举将复杂的聚合逻辑从前端和后端核心平台中剥离，保持了各部分的纯净。

---

### 三、新架构下的工作流示例：京城医院创建订单

1. 京城医生端前端调用`京城定制网关`的 /jc-api/orders 接口。
2. `京城定制网关` 首先调用`SBP-订单服务`的标准创建订单接口。
3. `SBP-订单服务` 在执行过程中，会自动执行已注册的Hook，并调用由`京城定制实现`提供的 beforeOrderCreate 逻辑（进行HIS数据转换）。
4. `SBP-订单服务` 完成核心创建逻辑，并发出 OrderCreatedEvent 事件。
5. `京城定制实现` 模块中监听此事件的监听器被触发，执行后续与HIS同步等操作。

- 6. `京城定制网关` 接收到SBP的返回结果，并再次调用`京城定制实现` 提供的接口，获取一些定制化数据。
- 7. `京城定制网关` 将所有数据聚合后返回给京城医生端前端。

---

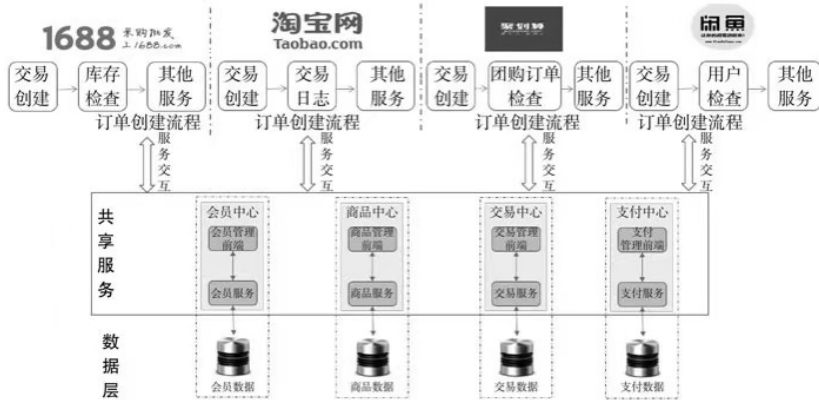
四、互医的后端业务实施路线图

- 1. Phase 1: 解耦与抽象
  - 成立“平台核心组”与“定制化组”。
  - 停止所有新功能的项目级复制。
  - 抽离Common Core，建立强版本管理。
  - 在SBP中为第一个核心领域（如用户服务）设计并实现“扩展契约”（Hook、SPI）。
  - 选择京城项目的一个新需求，用新架构实现，作为范本。
- 2. Phase 2: 重构与迁移
  - 逐步将现有易赋诊和京城项目中的共性业务逻辑重构、下沉到SBP。
  - 将现有定制逻辑重写为“定制化实现”模块中的Hook、SPI和事件监听器。
- 3. Phase 3: 平台化与自治
  - SBP趋于稳定，提供全面的扩展能力。
  - 各个定制项目（包括新客户）完全基于扩展机制进行开发。

IV、互医未来的架构演进

融合平台化战略，做成类似阿里巴巴的共享服务体系建

设。



这张图展示了共享服务是如何支持前端业务的（并不准确表达各业务的交易流程），以1688（B2B电商平台）、淘宝（C2C电商平台）、聚划算（团购平台）、