



(12) 发明专利申请

(10) 申请公布号 CN 115640045 A

(43) 申请公布日 2023. 01. 24

(21) 申请号 202211674276.9

(22) 申请日 2022.12.26

(71) 申请人 卓望数码技术(深圳)有限公司

地址 518000 广东省深圳市南山区粤海街道高新区社区高新南七道015号深港产学研基地W601

(72) 发明人 张军勇

(74) 专利代理机构 深圳市恒和大知识产权代理有限公司 44479

专利代理师 林大超

(51) Int.Cl.

G06F 8/71 (2018.01)

G06F 8/38 (2018.01)

H04L 67/51 (2022.01)

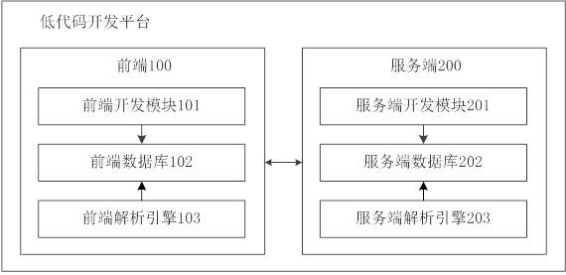
权利要求书2页 说明书14页 附图7页

(54) 发明名称

基于领域驱动设计的低代码开发平台及业务系统创建方法

(57) 摘要

本公开涉及软件开发技术领域,提供了基于领域驱动设计的低代码开发平台及业务系统创建方法。该平台至少包括前端和服务端,前端包括:前端开发模块根据前端设计数据生成前端配置数据;前端数据库存储前端配置数据;前端解析引擎加载并解析前端配置数据,生成前端界面工具;服务端包括:服务端开发模块根据服务端设计数据生成服务端配置数据;服务端数据库存储服务端配置数据;服务端解析引擎加载并解析服务端配置数据,生成业务服务,并将业务服务暴露给前端调用。本发明可实现业务与技术之间的平滑衔接,后续的修改方式简单,通用性好,可适用于不同的使用场景,适用于较复杂或大型业务系统的创建。



1. 一种基于领域驱动设计的低代码开发平台,其特征在于,至少包括前端和服务端;

所述前端包括:

前端开发模块,被配置为根据获取到的前端设计数据,生成前端配置数据,并将所述前端配置数据传递给前端数据库;

所述前端数据库,被配置为接收并存储所述前端配置数据;

前端解析引擎,被配置为根据第一用户的第一触发操作,加载并解析所述前端配置数据,生成前端界面工具,以使所述第一用户可通过所述前端界面工具访问所述服务端暴露的业务服务,并查看所述服务端返回业务数据;

所述服务端包括:

服务端开发模块,被配置为根据获取到的服务端设计数据,生成服务端配置数据,并将所述服务端配置数据传递给服务端数据库;

所述服务端数据库,被配置为接收并存储所述服务端配置数据;

服务端解析引擎,被配置为根据第二用户的第二触发操作,加载并解析所述服务端配置数据,生成业务服务,并通过服务网关将所述业务服务暴露给所述前端,所述业务服务包括业务展示对象和所述业务展示对象的接口协议。

2. 根据权利要求1所述的低代码开发平台,其特征在于,所述根据获取到的前端设计数据,生成前端配置数据,包括:

加载并显示前端设计页面,所述前端设计页面包括多个功能组件;

获取第三用户选中的一个目标功能组件,所述目标功能组件为所述多个功能组件中的任意一个;

加载并显示与所述目标功能组件对应的数据录入模板;

接收所述第三用户基于所述数据录入模板录入的前端设计数据;

根据预设的前端核心管理对象模型与所述前端设计数据之间的关联逻辑关系,自动生成前端配置数据,所述前端配置数据至少包括前端界面元素、所述前端界面元素的排布方式,以及所述前端界面元素与所述服务端暴露的业务服务的服务接口之间的关联关系。

3. 根据权利要求2所述的低代码开发平台,其特征在于,所述前端核心管理对象模型包括多个功能模板,每一个所述功能模板包括多个功能区域模板,每一个所述功能区域模板包括多个功能区域操作模板,每一个功能区域操作模板对应一个功能操作;

所述功能操作包括领域对象和领域对象属性,所述领域对象属性包括绑定属性和可选属性。

4. 根据权利要求1所述的低代码开发平台,其特征在于,所述根据第一用户的第一触发操作,加载并解析所述前端配置数据,生成前端界面工具,包括:

将所述第一用户从所述前端配置数据中选定的数据确定为目标配置数据;

加载并解析所述目标配置数据,执行所述目标配置数据,并渲染生成前端界面工具。

5. 根据权利要求1所述的低代码开发平台,其特征在于,根据获取到的服务端设计数据,生成服务端配置数据,包括:

加载并显示服务端设计页面,所述服务端设计页面包括多个设计组件;

获取第四用户选中的一种目标设计组件,所述目标设计组件为多个设计组件中的任意一个;

加载并显示与所述目标设计组件对应的信息录入模板；

接收所述第四用户基于所述信息录入模板录入的服务端设计数据；

根据预设的服务端核心管理对象模型与服务端设计数据的对应关系，自动生成服务端配置数据，所述服务端配置数据包括但不限于业务服务、领域对象、展示对象、方法、业务逻辑。

6. 根据权利要求5所述的低代码开发平台，其特征在于，所述服务端核心管理对象模型包括业务系统、业务域、管理对象，以及所述管理对象的属性、方法以及所述方法的业务流；

所述管理对象包括服务对象、领域对象、展示对象；

所述服务对象包括应用服务、领域服务、技术服务、第三方服务；

所述领域对象包括聚合根对象、实体对象和值对象；

所述业务流可与不同类型的业务流节点结合以实现不同的业务逻辑；

所述业务流节点可使用关系对象映射规则或者查询语句与业务数据库进行交互。

7. 根据权利要求1所述的低代码开发平台，其特征在于，还包括接口端；

所述接口端包括：

接口管理模块，被配置为根据获取到的接口设计数据，生成接口配置管理数据，并将所述接口配置管理数据传递给接口端数据库；

接口端数据库，被配置为接收并存储所述接口配置管理数据，所述接口配置管理数据包括接口属性、接口调用方法；

接口端解析引擎，被配置为根据获取到的接口报文，加载并解析所述接口配置管理数据，并执行所述接口配置管理数据对应的接口业务逻辑，以实现系统内部或者不同业务系统之间的业务交互。

8. 根据权利要求1所述的低代码开发平台，其特征在于，所述前端解析引擎可通过独立部署或者混合部署的方式部署在所述前端。

9. 一种业务系统创建方法，其特征在于，包括：

前端：

前端开发模块，根据获取到的前端设计数据，生成前端配置数据，并将所述前端配置数据传递给前端数据库；

所述前端数据库，接收并存储所述前端配置数据。

10. 一种业务系统创建方法，其特征在于，包括：

服务端：

服务端开发模块，根据获取到的服务端设计开发数据，生成服务端配置数据，并将所述服务端配置数据传递给服务端数据库；

所述服务端数据库，接收并存储所述服务端配置数据。

基于领域驱动设计的低代码开发平台及业务系统创建方法

技术领域

[0001] 本公开涉及软件开发技术领域,尤其涉及基于领域驱动设计的低代码开发平台及业务系统创建方法。

背景技术

[0002] 低代码是一种创建应用程序的新方法。它的开发速度很快,只需提前制作一份工作流程脑图即可根据需求快速创建应用程序。低代码还可以将企业所有资源(人员、技术和数据)统一在同一个工作流程中。企业使用低代码平台创建应用程序,与传统开发相比,可以将构建应用程序的速度提高10倍,并将维护成本降低50%以上。

[0003] 低代码开发工具可以让开发者参与到应用程序开发的过程中,帮助企业提高敏捷性,降低了应用程序开发过程的复杂性。低代码平台可以帮助开发者在更短的时间内开发更多的应用程序,实现企业的降本增效诉求。

[0004] 现有的低代码平台主要包括以下几种:①基于传统的工作流引擎能力构建的低代码平台;②以云平台的技术能力调用和服务编排为重点的低代码平台;③面向特定领域的低代码平台;④传统的通用低代码平台。

[0005] 然而,现有的低代码平台的设计大多以结构化、表单驱动、模型驱动的设计思想来组织和实现业务为主,无法充分发挥业务人员的作用,做到业务与技术之间的平滑衔接;同时,在生成的代码或代码框架需要二次修改时,后续的修改只能采用传统的代码开发方式,或者需要引入新的脚本语,从而使得修改操作十分复杂且成本较高;或者仅能面向专业领域,或需要绑定特定的平台或产品,通用性较差,大大地限制了其使用场景;此外,目前的低代码平台在面对较复杂或大型业务系统的创建时,会碰到诸多的管理问题,例如,需要解决低代码平台与传统开发方式开发的代码的共存问题、相互调用问题、事务问题以及部署的问题等。

[0006] 可见,目前的低代码平台仍存在无法实现业务与技术之间的平滑衔接,后续的代码或代码框架的修改方式复杂且成本较高,通用性差,使用场景的限制较大,以及难以适用于较复杂或大型业务系统的创建的问题。

发明内容

[0007] 有鉴于此,本公开实施例提供了基于领域驱动设计的低代码开发平台及业务系统创建方法,以解决目前的低代码平台仍存在无法实现业务与技术之间的平滑衔接,后续的代码或代码框架的修改方式复杂且成本较高,通用性差,使用场景的限制较大,以及难以适用于较复杂或大型业务系统的创建的问题。

[0008] 本公开实施例的第一方面,提供了一种基于领域驱动设计的低代码开发平台,至少包括前端和服务端;

所述前端包括:

前端开发模块,被配置为根据获取到的前端设计数据,生成前端配置数据,并将所

述前端配置数据传递给前端数据库；

所述前端数据库，被配置为接收并存储所述前端配置数据；

前端解析引擎，被配置为根据第一用户的第一触发操作，加载并解析所述前端配置数据，生成前端界面工具，以使所述第一用户可通过所述前端界面工具访问所述服务端暴露的业务服务，并查看所述服务端返回业务数据；

所述服务端包括：

服务端开发模块，被配置为根据获取到的服务端设计数据，生成服务端配置数据，并将所述服务端配置数据传递给服务端数据库；

所述服务端数据库，被配置为接收并存储所述服务端配置数据；

服务端解析引擎，被配置为根据第二用户的第二触发操作，加载并解析所述服务端配置数据，生成业务服务，并通过服务网关将所述业务服务暴露给所述前端，所述业务服务包括业务展示对象和所述业务展示对象的接口协议。

[0009] 本公开实施例的第二方面，提供了一种业务系统创建方法，包括：

前端：

前端开发模块，根据获取到的前端设计数据，生成前端配置数据，并将所述前端配置数据传递给前端数据库；

所述前端数据库，接收并存储所述前端配置数据。

[0010] 本公开实施例的第三方面，提供了一种业务系统创建方法，包括：

服务端：

服务端开发模块，根据获取到的服务端设计开发数据，生成服务端配置数据，并将所述服务端配置数据传递给服务端数据库；

所述服务端数据库，接收并存储所述服务端配置数据。

[0011] 本公开实施例与现有技术相比，其至少存在的有益效果是：提供了一种基于领域驱动设计的低代码开发平台，该平台至少包括前端和服务端；所述前端包括：前端开发模块，被配置为根据获取到的前端设计数据，生成前端配置数据，并将所述前端配置数据传递给前端数据库；所述前端数据库，被配置为接收并存储所述前端配置数据；前端解析引擎，被配置为根据第一用户的第一触发操作，加载并解析所述前端配置数据，生成前端界面工具，以使所述第一用户可通过所述前端界面工具访问所述服务端暴露的业务服务，并查看所述服务端返回业务数据；所述服务端包括：服务端开发模块，被配置为根据获取到的服务端设计数据，生成服务端配置数据，并将所述服务端配置数据传递给服务端数据库；所述服务端数据库，被配置为接收并存储所述服务端配置数据；服务端解析引擎，被配置为根据第二用户的第二触发操作，加载并解析所述服务端配置数据，生成业务服务，并通过服务网关将所述业务服务暴露给所述前端，所述业务服务包括业务展示对象和所述业务展示对象的接口协议。本公开实施例提供的低代码开发平台可很好地实现业务与技术之间的平滑衔接，后续的代码或代码框架的修改方式简单且成本较低，通用性好，可适用于不同的使用场景，且可适用于较复杂或大型业务系统的创建。

附图说明

[0012] 为了更清楚地说明本公开实施例中的技术方案，下面将对实施例或现有技术描述

中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本公开的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其它的附图。

[0013] 图1是本公开实施例提供的一种基于DDD的领域划分设计方式的示意图;

图2是本公开实施例提供的一种基于领域驱动设计的低代码开发平台的系统架构的示意图;

图3是本公开实施例提供的一种前端核心管理对象模型的示意图;

图4是本公开实施例提供的一种服务端核心管理对象模型的示意图;

图5是本公开实施例提供的另一种基于领域驱动设计的低代码开发平台的系统架构的示意图;

图6是本公开实施例提供的一种接口端核心管理对象模型的示意图;

图7是本公开实施例提供的一种基于领域驱动设计的低代码开发平台的整体系统架构的示意图;

图8是本公开实施例提供的一种业务系统创建方法的流程示意图;

图9是本公开实施例提供的另一种业务系统创建方法的流程示意图。

具体实施方式

[0014] 以下描述中,为了说明而不是为了限定,提出了诸如特定系统结构、技术之类的具体细节,以便透彻理解本公开实施例。然而,本领域的技术人员应当清楚,在没有这些具体细节的其它实施例中也可以实现本公开。在其它情况中,省略对众所周知的系统、装置、电路以及方法的详细说明,以免不必要的细节妨碍本公开的描述。

[0015] 下面将结合附图详细说明根据本公开实施例的基于领域驱动设计的低代码开发平台及业务系统创建方法。

[0016] 在现有技术中,传统的基于 workflow 引擎厂商构建的低代码平台一般主要基于其 workflow 引擎能力构建,重点是灵活的流程编排和工单的自动生成能力。云平台厂商构建的低代码平台一般以云平台的技术能力调用和服务编排为重点,这类低代码平台一般会与其云平台强绑定。一些专业领域的低代码平台一般对其专业领域的业务能力进行抽象和封装,形成通用的 API (应用程序接口),再结合一定的低代码能力(如代码生成、配置化等)构建的面向特定领域的低代码平台。传统的通用低代码平台一般是软件开发厂商或者一些大型公司,在其长期的软件开发或系统建设过程中,逐渐积累下来的技术能力或技术组件,再结合一定的低代码能力(如代码生成、配置化等)构建的低代码平台,这类低代码平台由于其发展演进时间比较长,核心以及配套功能相对会比较完善、成熟,但一般历史负担会比较重,其架构和核心思想一般都会比较落后。

[0017] 目前的低代码平台普遍存在下述问题:

(1) 现有的低代码平台的设计思想比较陈旧,无法做到业务与技术之间的平滑衔接:现有的低代码平台大多以结构化、表单驱动、模型驱动的设计思想来组织和实现业务为主。而结构化和表单驱动这两种设计思想在面向比较复杂或大型业务系统的建设时,无法有效地对庞大的业务内容进行清晰的管理。此外,模型驱动的设计思想目前还是比较主流的低代码平台的设计思想,但这种思想无法充分发挥业务人员的作用,做到业务与技术之

间的平滑衔接。

[0018] (2) 现有的低代码平台在后续的代码或代码框架的修改方式复杂且成本较高：现有的低代码平台大多数是采用生成代码或生成代码框架的方式，而当生成的代码或代码框架需要二次修改时，只能采用传统的代码开发方式来进行修改；甚至有些低代码平台在实现低代码能力时需要引入另一种脚本语言，这就要求开发设计人员要额外学习一种新的脚本语言，这种修改方式十分繁复且成本较高。

[0019] (3) 现有的低代码平台的通用性差且使用场景限制较大：现有的某些低代码平台是面向专业领域的低代码平台，没有通用性；或者是绑定特定平台或产品的，这就限制了使用场景。

[0020] (4) 现有的低代码平台难以适用于较复杂或大型业务系统的创建：现有的低代码平台大多数以实现简单功能或者演示 (Demo) 系统为主，在面对现实中的较为复杂或大型的业务系统的创建时，总是会遇到很多的问题。比如，现有低代码平台无法适用于所有的业务场景，这也就意味着低代码平台必须要解决与传统开发方式开发的代码的共存问题、相互调用问题、事务问题以及部署的问题；再比如，目前的低代码平台对其管理的内容一般采用堆砌的方式进行组织管理，并没有采用比较先进的设计及管理思想进行抽象管理，在面对较复杂或大型业务系统时，往往会面临较大的管理问题。

[0021] 为了解决上述技术问题，本公开实施例提出了一种基于领域驱动设计的低代码开发平台，该低代码开发平台采用领域驱动设计 (简称“DDD”) 思想作为其核心设计思想。基于领域驱动设计思想，将业务领域划分为前端领域、服务端领域和接口端领域三个领域，由此设计该低代码开发平台的整体系统架构包括前端、服务端和接口端。通过在该低代码开发平台的前端、服务端和接口端均部署一个解析引擎，并利用部署在低代码开发平台中的解析引擎实现三个领域 (包括前端、服务端和接口端) 的低代码能力，其中，解析引擎的部署方式灵活，支持多种开发、部署模式，以应对复杂业务系统的开发和部署需求，从而实现了业务与技术之间的平滑衔接，且后续的代码或代码框架的修改方式简单且成本较低，此外，本公开的低代码开发平台通用性好，不受使用场景的限制，尤其可适用于较复杂或大型业务系统的创建。

[0022] 一方面，本公开实施例以领域驱动设计 (DDD) 思想作为低代码开发平台本身设计的核心思想；另一方面，在基于该低代码开发平台设计开发业务系统时也可以领域驱动设计 (DDD) 思想作为其设计开发的核心思想。

[0023] 领域驱动设计是一种由领域模型来驱动着系统设计的思想。领域模型是对业务模型的抽象，DDD是把业务模型翻译成系统架构设计的一种方式。

[0024] 图1示出了一种基于DDD的领域划分设计方式的示意图。结合图1，基于DDD的设计思想，可以将本公开实施例所涉及的低代码开发平台的业务领域划分为以下三个领域：①面向核心业务服务领域的服务端领域 (即服务端)；②面向人机交互领域的前端领域 (即前端)；③面向系统间交互的接口端领域 (即接口端)。其中，服务端领域主要包括领域模型、领域服务、应用服务；前端领域主要包括用户界面类型，例如，常规WEB界面、数据可视化类、统计分析类和定制类型等；接口端领域主要包括实时服务接口、文件接口、定制类型接口等。

[0025] 可以理解的是，本公开实施例基于DDD的设计思想将低代码开发平台设计成一个包括前端、服务端以及接口端的解析型通用综合软件开发平台。该低代码开发平台设可以

满足各种业务场景的业务系统的设计开发,可实现业务与技术之间的平滑衔接,与此同时,在后续需要针对在先创建的业务系统进行修改时,可直接通过解析引擎来修改相应的配置数据即可,无需对代码或代码框架进行修改,大大降低了后续修改操作的复杂度和成本。

[0026] 图2是本公开实施例提供的一种基于领域驱动设计的低代码开发平台的系统架构的示意图。如图2所示,该低代码开发平台,至少包括前端100和服务端200。

[0027] 前端100包括前端开发模块101、前端数据库102和前端解析引擎103。

[0028] 前端开发模块101,被配置为根据获取到的前端设计数据,生成前端配置数据,并将所述前端配置数据传递给前端数据库。

[0029] 在一些实施例中,所述根据获取到的前端设计数据,生成前端配置数据,包括如下步骤:

加载并显示前端设计页面,所述前端设计页面包括多个功能组件;

获取第三用户选中的一个目标功能组件,所述目标功能组件为所述多个功能组件中的任意一个;

加载并显示与所述目标功能组件对应的数据录入模板;

接收所述第三用户基于所述数据录入模板录入的前端设计数据;

根据预设的前端核心管理对象模型与所述前端设计数据之间的关联逻辑关系,自动生成前端配置数据,所述前端配置数据至少包括前端界面元素、所述前端界面元素的排布方式,以及所述前端界面元素与所述服务端暴露的业务服务的服务接口之间的关联关系。

[0030] 第三用户,通常是指业务人员或者前端开发设计人员。当然,还可以是智能机器人等。

[0031] 功能组件可以理解为便于第三用户理解和操作前端设计页面进行前端界面开发、设计以及测试等动作的一些功能按钮等。示例性的,多个功能组件可包括“页面创建”功能按钮、“页面跳转”功能按钮、“退出页面”按钮、“页面保存”按钮等。

[0032] 在一实施例中,第三用户可以根据实际需求通过点击/触摸前端设计页面上显示的多个功能组件中的任意一个,选中的功能组件即被确定为目标功能组件。

[0033] 在一示例性实施例中,若检测到第三用户对“页面创建”这一功能按钮的选中操作,则加载并显示与“页面创建”这一功能按钮对应的数据录入模板。该数据录入模板通常可包括一个或者多个信息录入区域,例如,可包括名称、请求上下文、版本、用户、返回的参数结构、关联的展示对象及该展示对象的属性、页面发布信息等信息录入区域。

[0034] 第三用户可以基于当前界面上所展示的数据录入模板上的一个或者多个信息录入区域的提示文本来录入相应的信息(即前端设计数据)。

[0035] 当接收到第三用户基于所述数据录入模板录入的前端设计数据时,根据预设的前端核心管理对象模型与所述前端设计数据之间的关联逻辑关系,自动生成前端配置数据。

[0036] 在本公开实施例中,基于DDD的设计思想,可将基于低代码开发平台开发的业务系统的复杂业务对象和业务逻辑抽象为一套包括各类管理对象以及业务逻辑的核心管理对象模型。

[0037] 作为一示例,前端核心管理对象模型如图3所示。其中,图3中的黑色实心菱形箭头和空心菱形箭头均代表“组合关系”,空心三角箭头代表“泛化、继承关系”,简单箭头代表

“使用关系、关联关系”。

[0038] 如图3所示,该前端核心管理对象模型包括多个功能模板,每一个所述功能模板包括多个功能区域模板,每一个所述功能区域模板包括多个功能区域操作模板,每一个功能区域操作模板对应一个功能操作;所述功能操作包括领域对象和领域对象属性,所述领域对象属性包括绑定属性和可选属性。

[0039] 作为一示例,一个功能模板通常代表一类功能,例如,可将“增、删、改、查”定义为一类功能。一个功能模板中通常可包括多个功能区域模板,例如,“增”功能对应一个功能区域模板,“删”功能对应一个功能区域模板等。一个功能区域模板包括多个功能区域模板包括多个功能区域操作模板,以“增”功能对应的功能区域模板为例,在该功能区域模板中可以包括多个领域对象的数据等的增加区域操作模板。

[0040] 领域对象通常是指前端所涉及的操作对象或实体,例如,可以是某个商品等。领域对象属性中的绑定属性可以是领域对象的唯一标识字段等属性值,可选属性可以是展示对象、展示方式等。

[0041] 在本公开实施例中,上述的各种模板都可以通过实例化,实现其相应的功能。例如,对于“增”功能的功能区域模板可通过实例化,以实现其预先配置的增加数据表、字段等方面的功能。

[0042] 在一示例性实施例中,在接收到第三用户基于所述数据录入模板录入的前端设计数据时,即可建立起预设的前端核心管理对象模型中各个领域对象、业务逻辑与前端设计数据之间的关联逻辑关系,自动生成一套前端配置数据。该套前端配置数据至少包括前端界面元素、所述前端界面元素的排布方式,以及所述前端界面元素与所述服务端暴露的业务服务的接口之间的关联关系。

[0043] 其中,前端界面元素通常是指组成前端界面的一些功能组件,例如,各类功能按钮等。前端界面元素的排布方式是指这些功能组件在前端界面上的排布规则,包括各个功能组件在前端界面上的排列方式(如随机排列,按照一定顺序进行排列等)、各个功能组件在前端界面上的显示大小、颜色等。前端界面元素与服务端暴露的业务服务的接口之间的关联关系具体是指各功能组件与服务端暴露的业务服务的接口之间的绑定关系、对应关系。例如,前端界面上的“查看发票详情”与服务端暴露的“展示对象”绑定。

[0044] 在本公开实施例中,基于DDD思想对业务系统三大领域的划分(如图1所示)以及对管理对象和业务逻辑的抽象得到的前端核心管理对象模型(如图3所示),它们与业务系统所归属的行业或技术领域无关,而是对业务系统的一种通用的抽象,能够实现通用低代码开发平台提供设计基础,也为面向复杂业务系统的低代码开发提供了管理基础。同时,由于DDD设计思想的特性,使基于本低代码开发平台进行开发的业务系统能够达到业务与技术之间的无缝衔接。

[0045] 所述前端数据库102,被配置为接收并存储所述前端配置数据。

[0046] 在一些实施例中,前端数据库102可以将从前端开发模块101处接收到的前端配置数据按照如下表1的存储方式进行对应存储。

[0047] 表1

前端配置数据ID	前端配置数据明细	数据存储地址	数据创建/修改时间
01	AA	aa	20xx年xx月xx日

02	BB	bb	20xx年xx月xx日
...	...	...	...

前端解析引擎103,被配置为根据第一用户的第一触发操作,加载并解析所述前端配置数据,生成前端界面工具,以使所述第一用户可通过所述前端界面工具访问所述服务端暴露的业务服务,并查看所述服务端返回业务数据。

[0048] 在一些实施例中,所述根据第一用户的第一触发操作,加载并解析所述前端配置数据,生成前端界面工具,包括:

将所述第一用户从所述前端配置数据中选定的数据确定为目标配置数据;

加载并解析所述目标配置数据,执行所述目标配置数据,并渲染生成前端界面工具。

[0049] 第一用户通常是指使用基于本公开提供的低代码开发平台开发的业务系统的系统用户。

[0050] 第一触发操作,通常是指系统用户在操作前端界面时对展示在当前界面上的“前端解析引擎”功能按钮的点击、触摸等选中操作。

[0051] 当检测到第一用户对当前界面界面上的“前端解析引擎”的选中操作时,执行“前端解析引擎”对应的程序代码,加载并解析存储在所述前端数据库中的前端配置数据。例如,当第一用户继续选中表1中ID为01的前端配置数据时,将该被选中的前端配置数据01确定为目标配置数据。接着,从前端数据库加载并解析所述目标配置数据,执行所述目标配置数据,并渲染生成前端界面工具。假设该目标配置数据是“查看发票详情”的页面配置数据,那么通过上述“前端解析引擎”可从前端数据库中加载并解析与“查看发票详情”对应的页面配置数据,执行该页面配置数据,并在当前界面上生成“查看发票详情”的页面。通过该“查看发票详情”的页面,第一用户可以访问服务端暴露的与“查看发票详情”关联的发票信息,其中,发票信息包括发票号码、发票金额、开票信息等。

[0052] 在一些实施例中,所述前端解析引擎可通过独立部署或者混合部署的方式部署在所述前端或者平台的某指定位置。

[0053] 在一种情况下,前端解析引擎可通过独立部署的方式部署在前端。具体的,可采用远程方法调用的方式实现通过本公开的低代码平台开发的业务逻辑方法与通过传统编码方式开发的业务逻辑方法之间的互相调用。该远程方法调用的方式不支持本地数据库事务。

[0054] 在另一种情况下,前端解析引擎还可通过混合部署的方式部署在前端。具体的,本公开的前端解析引擎可采用Jar包的方式引入到传统的代码工程中,与传统的代码进行混合部署。这种部署方式除了采用远程方法调用的方式实现通过本公开的低代码开发平台开发的业务逻辑方法与通过传统编码方式开发的业务逻辑方法之间的互调之外,还可以采用直接本地方法调用的方式实现二者之间的互相调用。通过本地方法调用的方式能够支持本地数据库事务。

[0055] 结合图2,服务端200包括服务端开发模块201、服务端数据库202和服务端解析引擎203。

[0056] 服务端开发模块201,被配置为根据获取到的服务端设计数据,生成服务端配置数据,并将所述服务端配置数据传递给服务端数据库。

[0057] 在一实施例中,根据获取到的服务端设计数据,生成服务端配置数据的步骤与上述步骤“根据获取到的前端设计数据,生成前端配置数据”的实现方式类似。

[0058] 根据获取到的服务端设计数据,生成服务端配置数据的步骤可具体包括如下步骤:

加载并显示服务端设计页面,所述服务端设计页面包括多个设计组件;

获取第四用户选中的一种目标设计组件,所述目标设计组件为多个设计组件中的任意一个;

加载并显示与所述目标设计组件对应的信息录入模板;

接收所述第四用户基于所述信息录入模板录入的服务端设计数据;

根据预设的服务端核心管理对象模型与服务端设计数据的对应关系,自动生成服务端配置数据,所述服务端配置数据包括但不限于业务服务、领域对象、展示对象(包括展示对象所用的接口协议等)、方法、业务逻辑(业务流)。

[0059] 第四用户,通常是指业务人员或者开发人员,当然,还可以是智能机器人等。

[0060] 设计组件可以为便于第四用户理解和操作服务端设计页面进行服务端界面开发、设计以及测试等动作的一些功能设计按钮等。示例性的,多个设计组件可以是包括“业务系统”创建按钮、“业务域”创建按钮、“系统数据库”创建按钮、“数据导入”按钮、“应用服务”创建按钮等。

[0061] 在一实施例中,第四用户可以根据需求通过点击/触摸服务端设计页面上显示的多个设计组件中的任意一个,选中的设计组件即为目标设计组件。

[0062] 在一示例性实施例中,若检测到第四用户对“业务系统”创建按钮的选中操作,则加载并显示与“业务系统”创建按钮对应的信息录入模板,该信息录入模板可包括名称、描述、请求上下文、版本、归属租户等信息录入框。

[0063] 第四用户可以基于当前界面上所展示的信息录入模板上的一个或者多个信息录入框的提示文本来录入相应的信息(即服务端设计数据)。

[0064] 接着,在接收到第四用户基于该信息录入模板录入的服务端设计数据时,根据预设的服务端核心管理对象模型与服务端设计数据之间的关联关系,自动生成服务端配置数据。

[0065] 作为一示例,服务端核心管理对象模型如图4所示。其中,图4中的黑色实心菱形箭头代表“组合关系”,空心三角箭头代表“泛化、继承关系”,简单箭头代表“使用关系、关联关系”。

[0066] 结合图4,所述服务端核心管理对象模型包括业务系统、业务域、管理对象,以及所述管理对象的属性、方法以及所述方法的业务流;所述管理对象包括服务对象、领域对象、展示对象;所述服务对象包括应用服务、领域服务、技术服务、第三方服务;所述领域对象包括聚合根对象、实体对象和值对象;所述业务流可与不同类型的业务流节点结合以实现不同的业务逻辑;所述业务流节点可使用关系对象映射规则或者查询语句与业务数据库进行交互。此外,通过对业务流节点的扩展可以增加各类技术能力的支持,如文件上传/下载、TXT/EXCEL/XML/JSON等类型文件的导入/导出、FTP/SFTP的存取、缓存服务访问、消息中间件服务访问、工作流能力访问等。

[0067] 在一示例性实施例中,在接收到第四用户基于所述信息录入模板录入的服务端设

计数据时,即可建立起预设的服务端核心管理对象模型中各个管理对象、业务逻辑与服务端设计数据之间的关联逻辑关系,自动生成一套服务端配置数据。

[0068] 所述服务端数据库202,被配置为接收并存储所述服务端配置数据。

[0069] 在一些实施例中,服务端数据库202可以将服务端开发模块201处接收到的服务端配置数据按照如下表2的存储方式进行对应存储。

[0070] 表2

服务端配置数据ID	服务端配置数据明细	数据存储地址	数据创建/修改时间
01	A	a	20xx年xx月xx日
02	B	b	20xx年xx月xx日
...	...	...	...

服务端解析引擎203,被配置为根据第二用户的第二触发操作,加载并解析所述服务端配置数据,生成业务服务,并通过服务网关将所述业务服务暴露给所述前端,所述业务服务包括业务展示对象和所述业务展示对象的接口协议。

[0071] 在一些实施例中,根据第二用户的第二触发操作,加载并解析所述服务端配置数据,生成业务服务,具体可包括如下步骤:

将所述第四用户从所述服务端配置数据中选定的数据确定为目标数据;

加载并解析所述目标数据,执行所述目标数据,并生成业务服务。

[0072] 第二触发操作,通常是指第四用户在操作服务端界面时对展示在当前界面上的“服务端解析引擎”功能按钮的点击、触摸等选中操作。

[0073] 当检测到第四用户对当前界面界面上的“服务端解析引擎”的选中操作时,执行“服务端解析引擎”对应的程序代码,加载并解析存储在服务端数据库中的服务端配置数据。例如,当第四用户继续选中表2中ID为01的服务端配置数据时,将该被选中的服务端配置数据01确定为目标数据。接着,从服务端数据库加载并解析所述目标数据,执行所述目标数据,并生成业务服务。假设该目标数据是与“发票管理服务”相关的服务配置数据,那么通过上述“服务端解析引擎”可从服务端数据库中加载并解析与“发票管理服务”对应的服务配置数据,并在当前界面上生成“发票管理服务”(即业务服务),同时,通过服务网关将所述业务服务暴露给所述前端,所述业务服务包括业务展示对象和所述业务展示对象的接口协议。即将根据“发票号码”查询发票详情方法暴露给前端调用。

[0074] 在本公开实施例中,服务端解析引擎的主要功能是从服务端数据库中加载并解析设计开发形成的服务端配置数据,并根据配置执行业务逻辑,提供完整的业务服务。与此同时,本公开实施例可通过服务网关统一暴露服务端需要对外暴露的业务服务,并提供鉴权、流控、性能监控、业务监控和链路监控等通用技术能力。

[0075] 可以理解的是,服务端解析引擎可采用独立部署或者混合部署的方式部署在服务端或者平台的某指定位置。

[0076] 下面以在本公开提供的低代码开发平台上开发创建“财务管理系统”为例,详细说明采用本公开提供的低代码开发平台进行业务系统的创建的具体操作过程。

[0077] 需要说明的是,在实施具体的开发建设工作之前,已经完成“财务管理系统”的领域建模以及数据库表设计工作并已经在数据库中配置相应的数据库表。具体的创建步骤示例如下(以下的步骤都在低代码开发平台的界面化集成开发环境中进行):

(1) 创建业务系统,即“财务管理系统”,系统的访问URL为:financeMGR。

[0078] (2) 在业务系统(“财务管理系统”)中创建业务域(对应领域建模中的业务领域)。

[0079] (3) 在业务系统(“财务管理系统”)中创建数据库。

[0080] (4) 根据数据库中的表批量导入并自动生成业务系统(“财务管理系统”)中的各类基础管理对象,包括实体对象(如“发票”实体对象)、展示对象(如“发票”展示对象)、数据库表、数据映射规则信息、关系对象映射信息、实体/展示对象的属性、数据库表的字段、实体对象的基础方法(增、删、改、查方法)、方法的参数(出参和入参)、方法对应的业务流以及业务流节点。

[0081] (5) 在业务系统(“财务管理系统”)中创建应用服务(以“发票管理服务”为例)。

[0082] (6) 在“发票管理服务”中创建根据“发票号码”查询发票详情方法,配置该方法的入参数为字符串类型的“发票号码”,出参为“发票”展示对象。

[0083] (7) 对根据“发票号码”查询发票详情方法的业务流进行编排,主要逻辑为构建参数、调用“发票”实体对象的“根据主键查询”方法、将“发票”实体对象转换为“发票”展示对象并返回。

[0084] (8) 暴露根据“发票号码”查询发票详情方法给前端调用,访问URL为:/invoiceService/byId。

[0085] (9) 在业务系统(“财务管理系统”)中创建“查看发票详情”页面,配置该页面对应的后端服务为前面配置的根据“发票号码”查询发票详情方法(URL:为/financeMGR/invoiceService/byId,返回的参数结构为“发票”展示对象)。

[0086] (10) 在“查看发票详情”页面配置要显示的信息并关联到“发票”展示对象对应的属性。

[0087] (11) 发布“查看发票详情”页面,配置访问URL为:/invoice/showDetailInfo。

[0088] (12) 用户可以通过URL:/financeMGR/invoice/showDetailInfo来访问前面发布的“查看发票详情”界面功能。

[0089] 其中,上述步骤(1)至(8)均在服务端执行,步骤(9)至(12)均在前端执行。

[0090] 本公开实施例提供的技术方案,采用解析引擎的方式可将业务系统涉及各类管理对象以及业务逻辑都沉淀到低代码开发平台的数据库中,形成一套配置数据;之后可通过低代码开发平台的解析引擎加载并解析沉淀的配置数据,实现具体的业务能力。此外,本公开还可通过解析引擎的方式对基于低代码开发平台开的业务系统进行持续的维护、升级以及版本迭代,也能够通过对解析引擎的扩展和优化,在不影响业务系统实现的情况下,附加各种配套功能并优化升级。

[0091] 图5是本公开实施例提供的另一种基于领域驱动设计的低代码开发平台的系统架构的示意图。如图5所示,该低代码开发平台,还包括接口端300。

[0092] 该接口端300具体可包括接口管理模块301、接口端数据库302和接口端解析引擎303。

[0093] 接口管理模块301,被配置为根据获取到的接口设计数据,生成接口配置管理数据,并将所述接口配置管理数据传递给接口端数据库。

[0094] 接口设计数据,是指第四用户基于接口端的界面上展示的数据录入模板录入的接口信息,该接收信息包括接口类型、接口协议、接口所涉及的两个或以上系统、调用的服务

端的方法等。

[0095] 在一些实施例中,根据获取到的接口设计数据,生成接口端配置数据的步骤与上述步骤“根据获取到的前端设计数据,生成前端配置数据”的实现方式类似,在此不再赘述。

[0096] 其中,根据获取到的接口设计数据,生成接口端配置数据中所用到的接口端核心管理对象模型如图6所示。

[0097] 作为一示例,接口端核心管理对象模型如图6所示。其中,图6中的黑色实心菱形箭头和空心菱形箭头均代表“组合关系”,空心三角箭头代表“泛化、继承关系”。

[0098] 如图6所示,该接口端核心管理对象模型主要包括系统接口定义、接口业务服务、接口模拟器。系统接口定义主要包括实时接收(发送)类接口(包括对端服务信息,如对端请求的业务服务或者发送的业务服务等)、文件接收(发送)类接口(包括文件地址信息,如IP地址等)。接口业务服务主要包括接口报文处理服务和接口服务生成服务。接口模拟器主要包括自定义接口模拟器、实时接收(发送)类接口模拟器、文件接收(发送)类接口模拟器。

[0099] 接口端数据库302,被配置为接收并存储所述接口配置管理数据,所述接口配置管理数据包括接口属性、接口调用方法。

[0100] 接口属性包括接口类型、接口协议、接口所涉及的两个或以上系统。接口调用方法通常是指调用的服务端的方法。

[0101] 在一些实施例中,接口端数据库302可以将将从接口端开发模块301处接收到的接口端配置数据按照如下表1、2类似的存储方式进行对应存储,在此不再赘述。

[0102] 在一些实施例中,接口端数据库302还可用于保存接口端接收或者发送的业务数据。

[0103] 接口端解析引擎303,被配置为根据获取到的接口报文,加载并解析所述接口配置管理数据,并执行所述接口配置管理数据对应的接口业务逻辑,以实现系统内部或者不同业务系统之间的业务交互。

[0104] 在一些实施例中,接口端解析引擎303可包括接口报文发送解析引擎和接口报文接收解析引擎。它们主要是从接口端数据库中加载、解析并执行接口端配置数据,使接口业务按预设的逻辑运行起来,以实现系统内部或者不同业务系统之间的业务交互。

[0105] 在一些实施例中,接口端还可包括接口端测试模块,该测试模块主要提供接口的模拟测试功能。

[0106] 在一些实施例中,接口端还可包括接口网关。所有接口都通过接口网关跟外部系统(第三方系统)进行对接,提供与业务逻辑无关的鉴权、流控、性能监控、业务监控、消息转换、协议转换、消息摘要及验证、消息压缩及解压缩等能力。

[0107] 可以理解的是,接口端解析引擎可采用独立部署或者混合部署的方式部署在接口端或者平台的某指定位置。

[0108] 为应对复杂业务系统的开发需求,本公开实施例提供的低代码开发平台可支持以下几种开发模式:

①零代码模式:功能的开发全部在低代码平台上通过配置开发实现并上线生效。零代码级的开发对开发人员的技术没有特别的要求,业务人员和技术人员都可以满足。

[0109] ②低代码模式:通过低代码平台的配置开发结合规划扩展点的扩展代码开发来实现完整的功能,并通过打包部署的方式来实现功能的上线。该模式对开发人员的技术有一

定的要求,一般需要技术人员或者具备一定技术功底的业务人员才可以满足。

[0110] ③低代码+个性化定制开发模式:功能的一部分可以采用低代码平台来配置开发,剩下的部分需要由开发人员进行个性化定制开发实现。主要使用人员是技术人员。

[0111] ④传统开发模式:整个功能完全采用传统的开发方式进行编码开发。主要使用人员是技术人员。

[0112] 图7是本公开实施例提供的一种基于领域驱动设计的低代码开发平台的整体系统架构的示意图。

[0113] 结合图7,采用该低代码开发平台进行业务系统开发的工作流程大致如下:

对于前端而言,业务或(软件)开发人员可通过界面配置开发平台(前端开发模块)进行界面开发设计操作,该界面配置开发平台可根据业务或开发人员进行界面开发设计操作时输入的前端设计数据,生成前端配置数据,并将该前端配置数据传递给界面配置开发平台数据库(前端数据库),界面配置开发平台数据库将接收到的前端配置数据进行保存,完成前端开发设计工作。系统用户(第一用户)可通过操作界面的方式启动界面解析执行引擎(前端解析引擎),该界面解析执行引擎可根据系统用户(第一用户)的第一触发操作,加载并解析所述前端配置数据,生成前端界面工具,以使所述第一用户可通过所述前端界面工具访问所述服务端的执行引擎(服务端解析引擎)暴露的业务服务,并查看所述服务端返回业务数据。

[0114] 对于服务端而言,业务或(软件)开发人员可通过设计实现一体化平台(服务端开发模块)进行服务端界面的开发设计操作,该设计实现一体化平台可以根据业务或开发人员进行服务端界面的开发设计操作时输入的服务端设计数据,生成服务端配置数据,并将所述服务端配置数据传递给设计实现一体化平台数据库(服务端数据库),设计实现一体化平台数据库将接收到的服务端配置数据进行保存,完成服务端开发设计工作。业务或开发人员可通过操作界面的方式来启动执行引擎(服务端解析引擎),该执行引擎可根据业务或开发人员的第二触发操作,加载并解析所述服务端配置数据,生成业务服务,并通过服务网关将所述业务服务暴露给所述前端,以便于前端调用,所述业务服务包括业务展示对象和所述业务展示对象的接口协议等。

[0115] 在一些实施例中,第三方系统或第三方平台可以通过直接访问服务端的服务网关以调用服务端的业务服务。示例性的,第三方系统(如产商品中心)可通过下述操作流程调用服务端的业务服务:(i)在审批流程完成时,通过服务网关调用服务端的产品下发接口报文生成服务;(ii)接口部分:生成接口发送报文,并将该接口发送报文通过接口报文发送引擎发送给接口平台数据库,以使接口平台数据库将该接口发送报文写入/插入到接口发送日志表中。

[0116] 对于接口端而言,业务或开发人员可通过操作接口配置管理平台(接口管理模块)配置接口设计数据,该接口配置管理平台可以根据业务或开发人员所配置的接口设计数据,生成接口配置管理数据,并将所述接口配置管理数据传递给接口配置数据库(接口端数据库),接口配置数据库将接收到的接口配置数据库进行保存,完成接口端设计工作。

[0117] 在一实施例中,该接口端还可包括接口测试平台,通过该接口测试平台可以通过数据依赖(通过一个关系中属性间值的相等与否体现出来的数据间的相互关系)的方式对接口配置数据库中的接口配置数据进行模拟测试。

[0118] 在另一些实施例中,第三方系统或第三方平台可以通过下述方式调用服务端的业务服务:

步骤(a)~(f):第三方系统或第三方平台可以通过向接口网关发送报文(消息或文件(扫描件)), (a)接口网关在接收第三方系统或平台发送的报文(消息或文件(扫描件))后,将接收到的报文(消息或文件(扫描件))实时发送给接口端数据库; (b)接口端数据库在接收到接口网关的报文后,在接口接收日志表中记录接口接收日志; (c)接口端数据库通过扫描获取由接口端解析引擎(接收报文处理引擎)发送过来的未处理日志; (c1)接口网关实时调用服务端的服务网关的接口内部服务; (d)接收报文处理引擎通过服务端的服务网关调用接口内部服务; (d1)接口网关在完成调用接口内部服务的动作时会触发接口端数据库记录监控日志; (f)接收报文处理引擎在完成调用接口内部服务的动作时也会触发接口端数据库记录监控日志。

[0119] 步骤(1)~(3): (1)接口端数据库扫描扫描未发送的接口日志; (2)接口报文发送引擎向接口网关发送报文(消息或文件); (3)接口网关将从接口报文发送引擎处接收到的发送报文发送给第三方系统或第三方平台,同时,接口端数据库会记录监控日志。

[0120] 本发明实施例提供的低代码开发平台,基于DDD思想将业务系统划分为前端、服务端和接口端三大领域,同时通过对业务系统中所涉及的管理对象和业务逻辑进行通用的抽象,可以很好地实现业务与技术之间的无缝衔接;通过解析引擎的方式实现低代码应用开发设计,能够无缝的对基于低代码开发平台开发的业务系统进行持续的维护、升级以及版本迭代,也能够通过对解析引擎的扩展和优化,在不影响业务系统实现的情况下,附加各种配套功能并优化升级,大大降低了业务系统的开发和维护成本。此外,本发明实施例提供的低代码开发平台可支持多种开发、部署模式,在面对复杂业务系统的开发时更具适应性和实用性。

[0121] 本发明经过Demo原型验证以及正式产品核心功能的检验,实现的低代码开发平台能够摆脱只适用简单业务功能或者Demo演示系统开发的使用场景的限制,能够更好的适用复杂或大型业务系统的研发场景的需要。

[0122] 上述所有可选技术方案,可以采用任意结合形成本申请的可选实施例,在此不再一一赘述。

[0123] 下述为本公开方法实施例,可以用于执行本公开方法实施例。对于本公开方法实施例中未披露的细节,请参照本公开低代码开发平台所涉及到的方法实施例。

[0124] 图8是本公开实施例提供的一种业务系统创建方法的流程示意图。如图8所示,该业务系统创建方法包括:

前端:

步骤S801,前端开发模块,根据获取到的前端设计数据,生成前端配置数据,并将所述前端配置数据传递给前端数据库;

步骤S802,所述前端数据库,接收并存储所述前端配置数据。

[0125] 在一些实施例中,所述根据获取到的前端设计数据,生成前端配置数据,包括:

加载并显示前端设计页面,所述前端设计页面包括多个功能组件;

获取第三用户选中的一个目标功能组件,所述目标功能组件为所述多个功能组件中的任意一个;

加载并显示与所述目标功能组件对应的数据录入模板；

接收所述第三用户基于所述数据录入模板录入的前端设计数据；

根据预设的前端核心管理对象模型与所述前端设计数据之间的关联逻辑关系，自动生成前端配置数据，所述前端配置数据至少包括前端界面元素、所述前端界面元素的排布方式，以及所述前端界面元素与服务端暴露的业务服务的接口之间的关联关系。

[0126] 在一些实施例中，所述根据第一用户的第一触发操作，加载并解析所述前端配置数据，生成前端界面工具，包括：

将所述第一用户从所述前端配置数据中选定的数据确定为目标配置数据；

加载并解析所述目标配置数据，执行所述目标配置数据，并渲染生成前端界面工具。

[0127] 图9是本公开实施例提供的另一种业务系统创建方法的流程示意图。如图9所示，该业务系统创建方法包括：

服务端：

步骤S901，服务端开发模块，根据获取到的服务端设计开发数据，生成服务端配置数据，并将所述服务端配置数据传递给服务端数据库；

步骤S902，所述服务端数据库，接收并存储所述服务端配置数据。

[0128] 应理解，上述实施例中各步骤的序号的大小并不意味着执行顺序的先后，各过程的执行顺序应以其功能和内在逻辑确定，而不对本公开实施例的实施过程构成任何限定。

[0129] 综上所述，本公开实施例提供了一种基于领域驱动设计的低代码开发平台及业务系统创建方法，该低代码开发平台采用领域驱动设计(DDD)作为总的系统架构设计核心思想。一方面，基于该核心思想将业务系统划分为前端领域、服务端领域和接口端领域三大领域，并且基于该核心思想分别针对性地根据前端领域、服务端领域以及接口端领域所涉及的管理对象以及业务逻辑进行抽象，形成了一套针对前端领域的通用的前端核心管理对象模型、一套针对服务端领域的通用的服务端核心管理对象模型以及一套针对接口端领域的通用的接口端核心管理对象模型。三大领域所对应的管理对象模型是对业务系统的一种通用的抽象，具有与业务系统归属的行业或技术领域的无关性，从而可实现通用的低代码能力，并可实现业务与技术的无缝衔接。另一方面，本公开实施例提供的低代码开发平台通过在各端部署的解析引擎来实现低代码能力，能够无缝的对基于低代码开发平台开发的业务系统进行持续的维护、升级以及版本迭代，也能够通过对解析引擎的扩展和优化，在不影响业务系统实现的情况下，附加各种配套功能并优化升级，从而具备良好的持续迭代升级能力。此外，本公开实施例提供的低代码开发平台由于具备上述的功能特性以及支持多种开发、部署模式，所以在面对复杂业务系统开发时更具适应性和实用性。

[0130] 以上实施例仅用以说明本公开的技术方案，而非对其限制；尽管参照前述实施例对本公开进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质的脱离本公开各实施例技术方案的精神和范围，均应包含在本公开的保护范围之内。

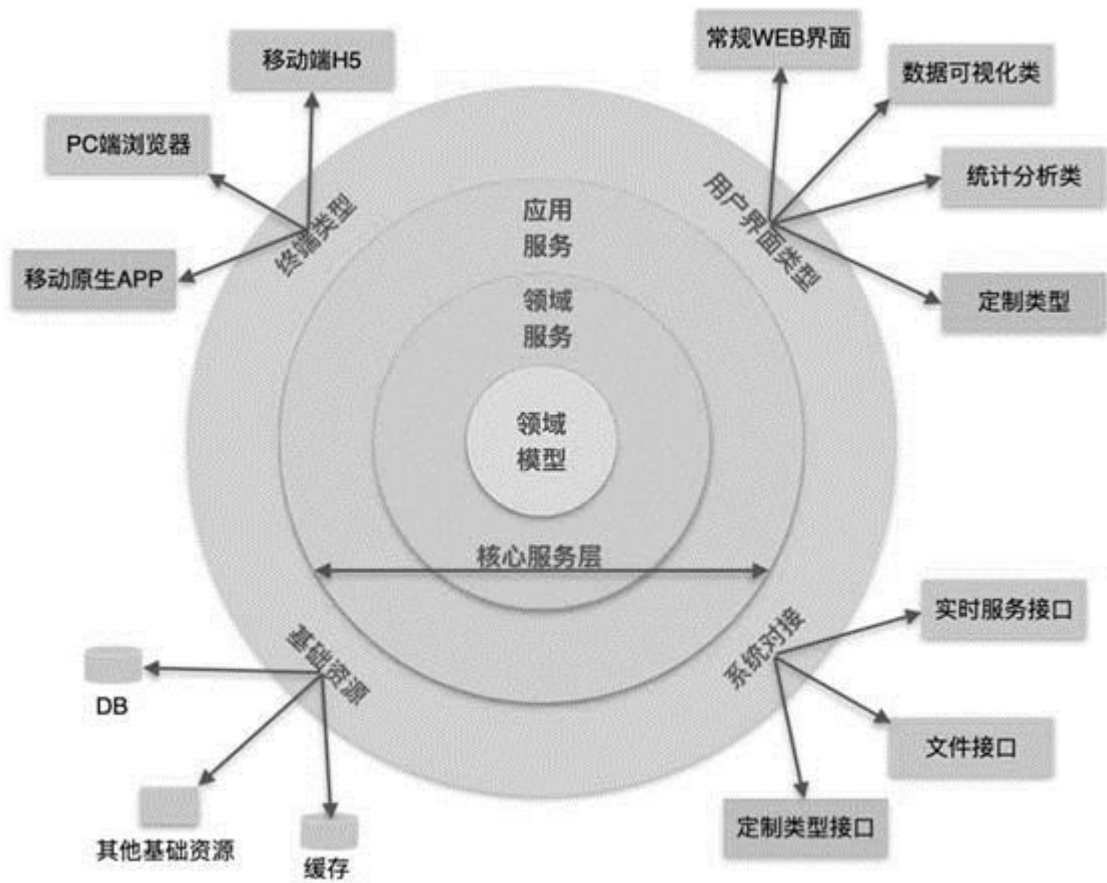


图1

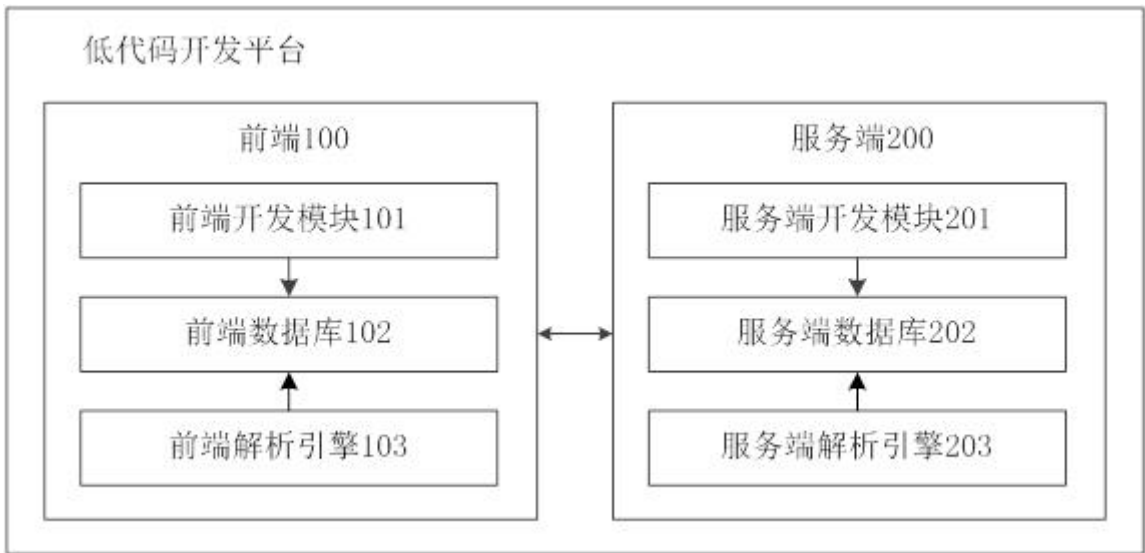


图2

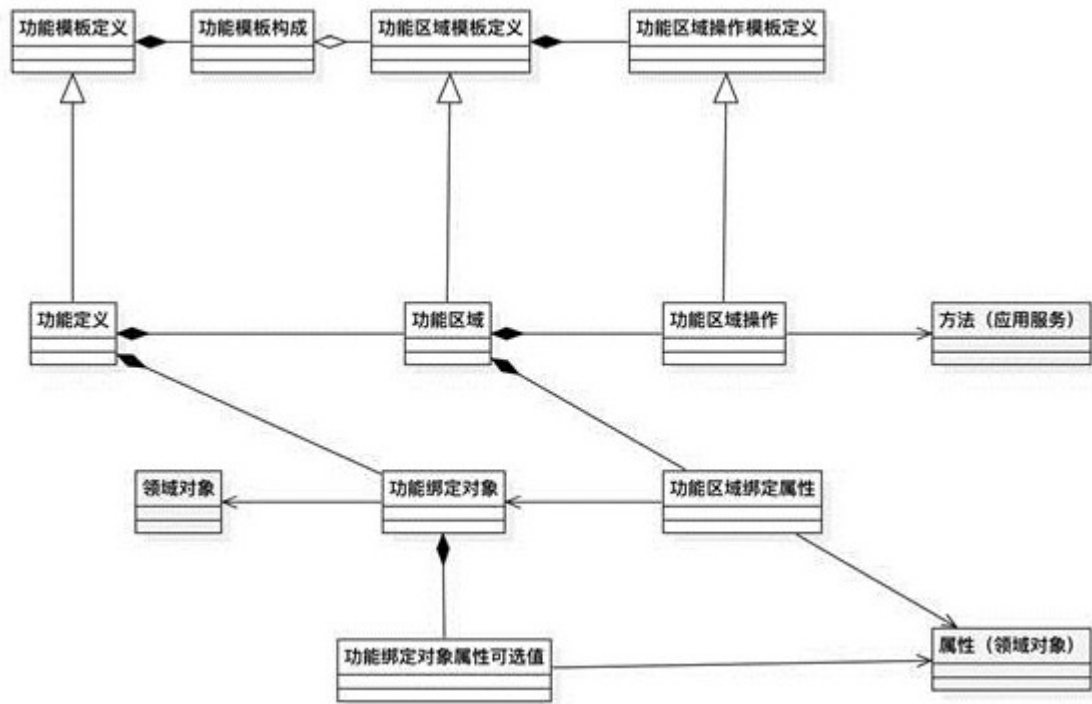


图3

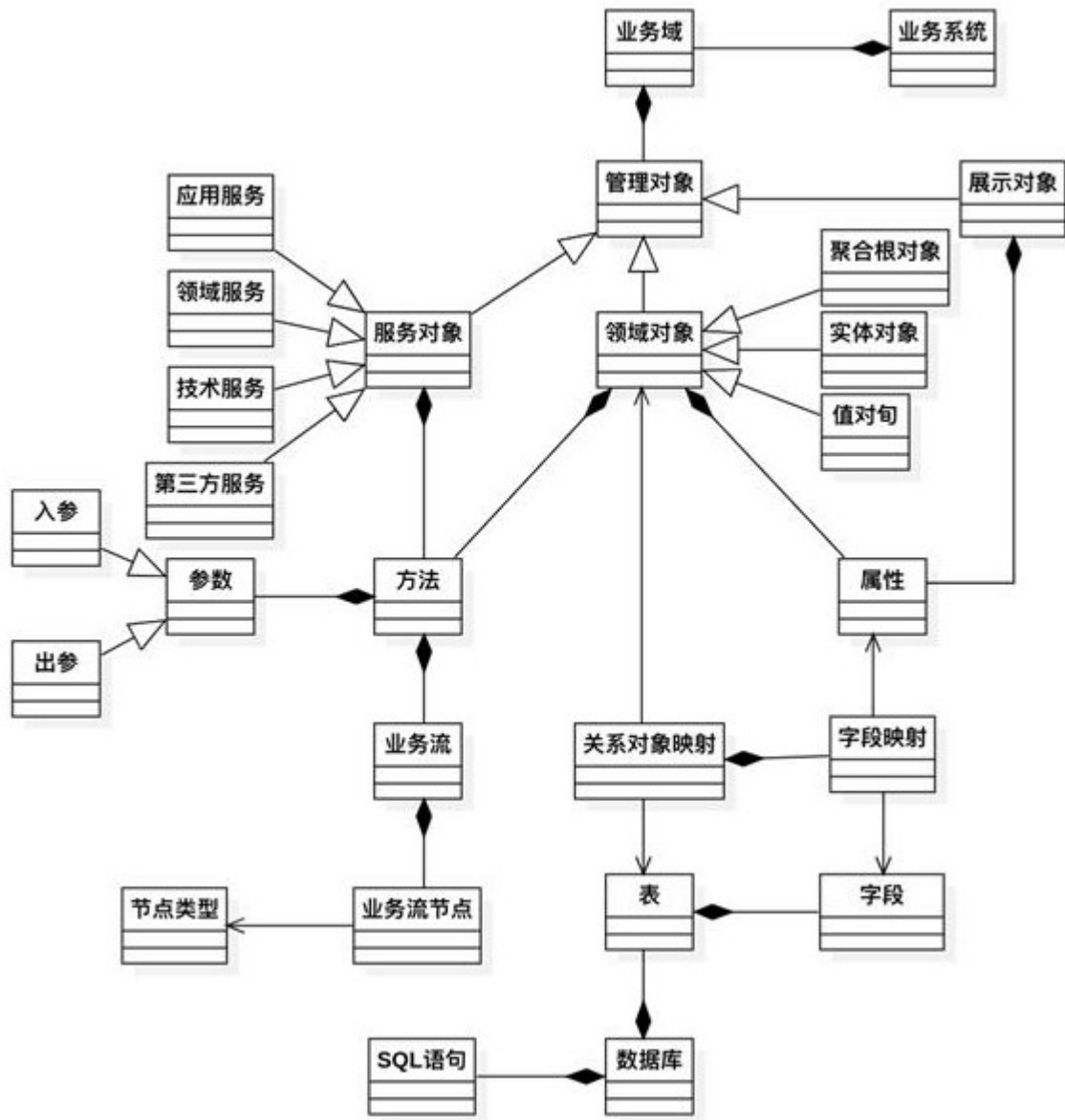


图4

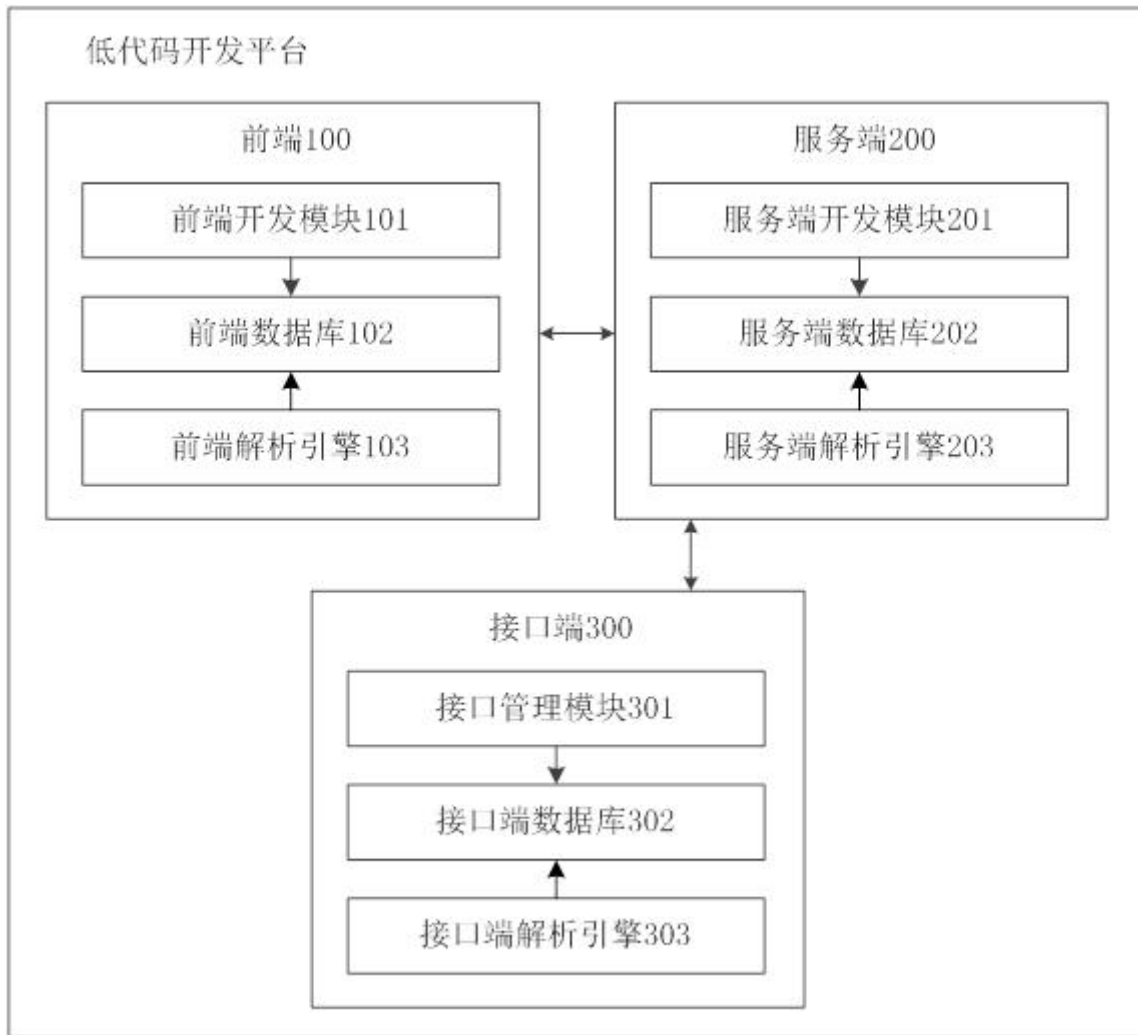


图5

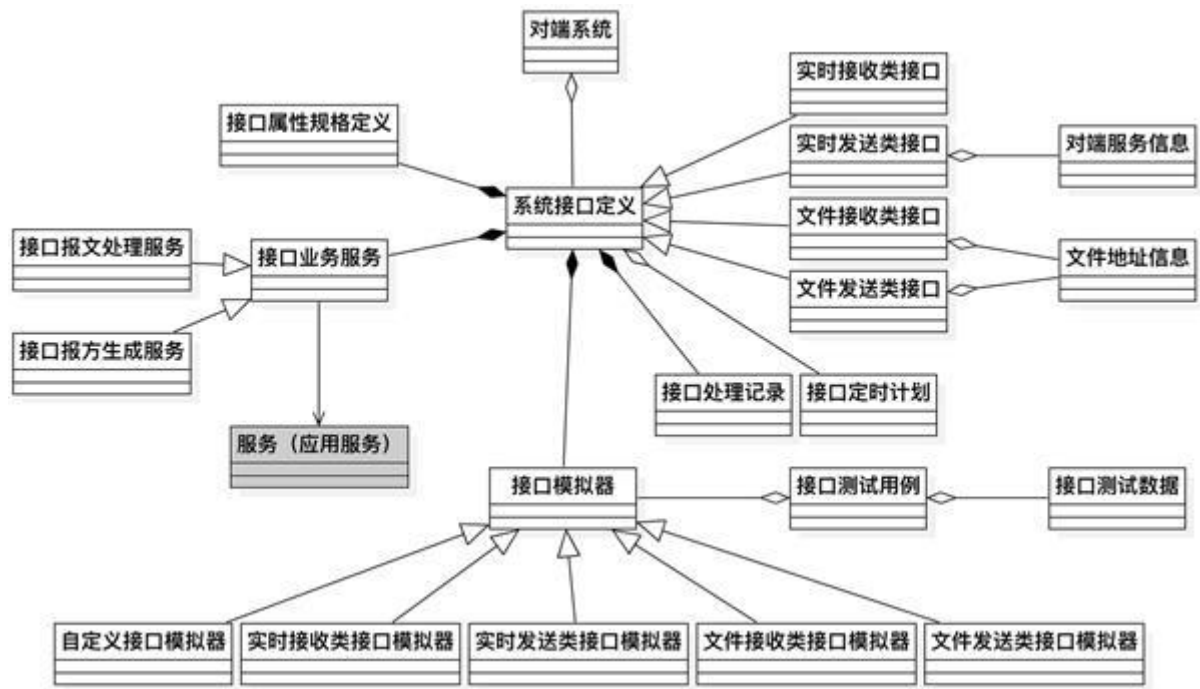


图6

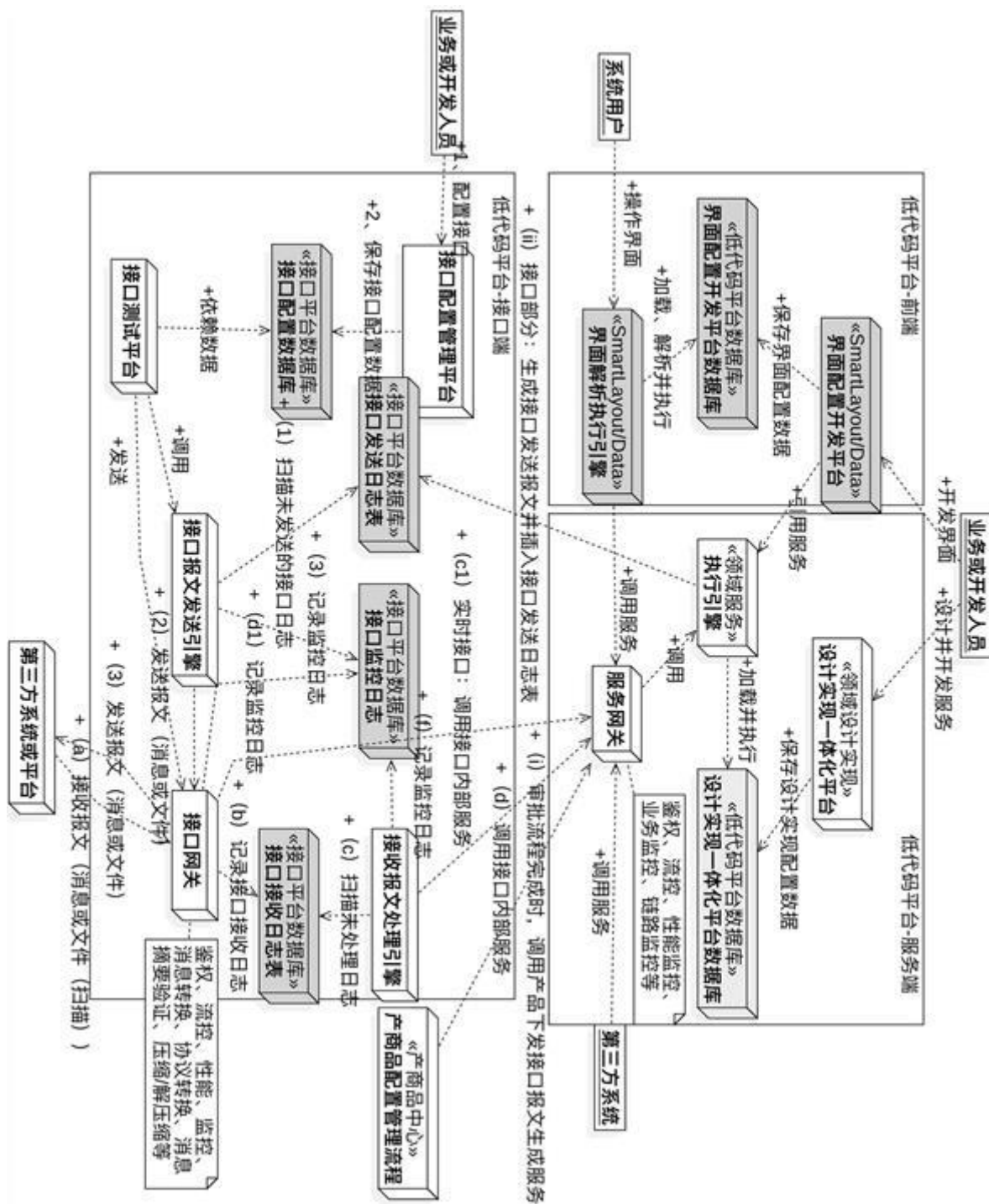


图7

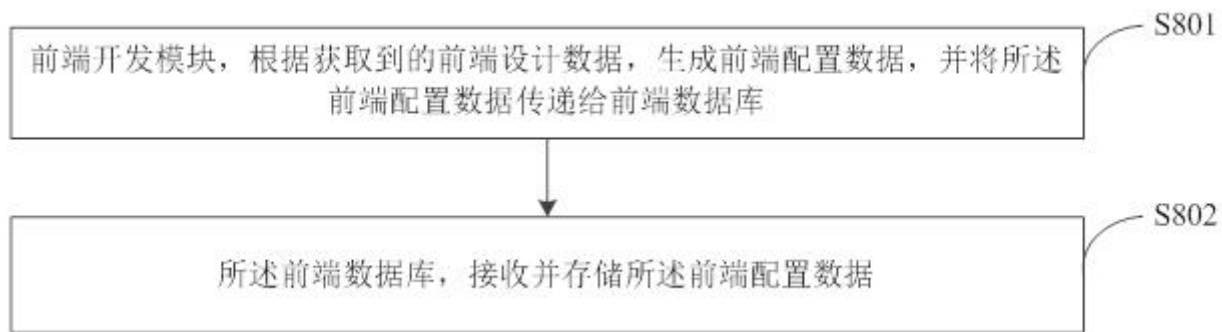


图8

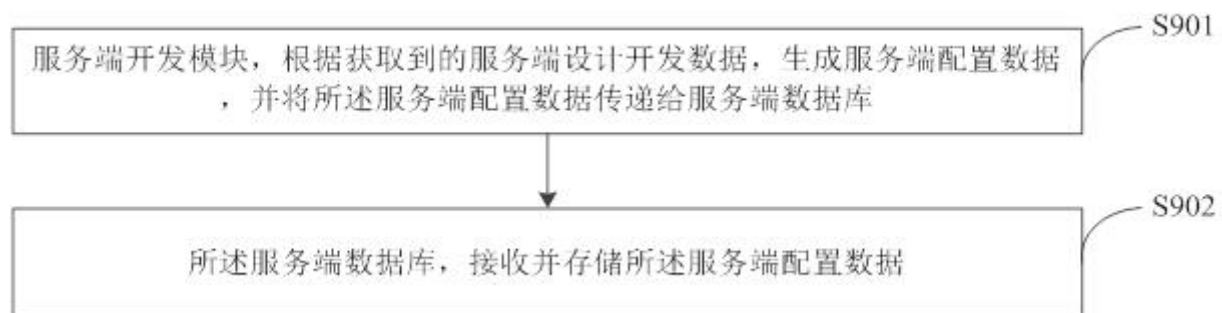


图9