(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111767035 A
(43)申请公布日 2020.10.13

(21)申请号 202010576147.0

(22)申请日 2020.06.22

(71)申请人 星辰天合(北京)数据科技有限公司
地址 100097 北京市海淀区蓝靛厂金源时代购物中心B区2#B座806-1

(72)发明人 郑晋 张旭明 王豪迈 胥昕

(74)专利代理机构 北京康信知识产权代理有限责任公司 11240
代理人 董文倩

(51)Int.Cl.
G06F 8/30(2018.01)
G06F 8/60(2018.01)
G06F 8/73(2018.01)

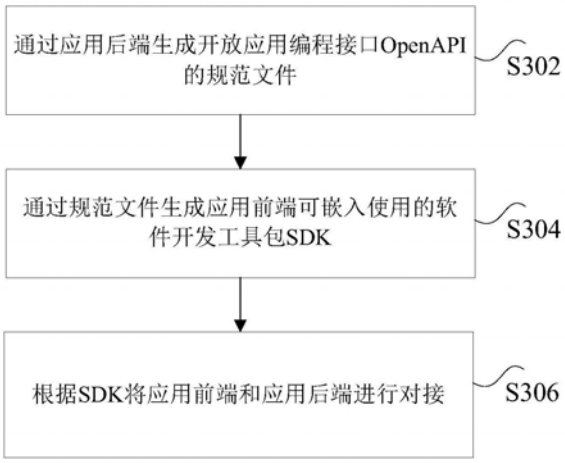
权利要求书2页 说明书7页 附图5页

(54)发明名称

基于OpenAPI的应用接口对接方法及装置

(57)摘要

本发明公开了一种基于OpenAPI的应用接口对接方法及装置。其中,该方法包括:通过应用后端生成开放应用编程接口OpenAPI的规范文件;通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK;根据SDK将应用前端和应用后端进行对接。本发明解决了相关技术中的基于API的应用接口对接方法,对接效率低,对接时间长的技术问题。



CN 111767035 A

1.一种基于OpenAPI的应用接口对接方法,其特征在于,包括:

通过应用后端生成开放应用编程接口OpenAPI的规范文件;

通过所述规范文件生成应用前端可嵌入使用的软件开发工具包SDK;

根据所述SDK将所述应用前端和所述应用后端进行对接。

2.根据权利要求1所述的方法,其特征在于,通过所述应用后端生成所述OpenAPI的所述规范文件,包括:

通过所述应用后端,编写所述OpenAPI的定义部分代码;

通过所述OpenAPI的工具包,对所述定义部分代码进行分析,提取所述定义部分代码的注释信息;

根据所述注释信息,生成所述规范文件。

3.根据权利要求2所述的方法,其特征在于,所述规范文件为所述开放应用编程接口OpenAPI的接口说明文件。

4.根据权利要求1所述的方法,其特征在于,通过所述规范文件生成所述应用前端可嵌入使用的所述SDK包括:

通过所述OpenAPI的工具包,根据所述规范文件创建数据结构;

将所述数据结构应用到所述应用前端可用的预设汇编语言的模板文件上,生成所述预设汇编语言的SDK,其中,所述预设汇编语言的SDK用于供所述应用前端嵌入使用。

5.根据权利要求4所述的方法,其特征在于,所述数据结构包括:第一部分和第二部分,其中,

所述第一部分用于描述所述应用后端包含的接口端点和所述接口端点支持的操作方法;

所述第二部分用于描述应用的实体和所述实体的属性。

6.根据权利要求5所述的方法,其特征在于,所述预设汇编语言为TypeScript语言;

所述第一部分为接口部分,所述第二部分为实体部分;

其中,所述接口部分与所述开放应用编程接口OpenAPI的路径部分相对应,所述实体部分与所述开放应用编程接口OpenAPI的组件部分相对应。

7.根据权利要求1所述的方法,其特征在于,根据所述SDK将所述应用前端和所述应用后端进行对接包括:

将所述SDK引入所述应用前端的代码中;

通过所述应用前端根据所述代码直接对所述应用后端进行接口对接。

8.根据权利要求7所述的方法,其特征在于,所述SDK还用于所述应用前端和所述应用后端对接的数据进行下列操作至少之一:参数校验、类型检查、自动补全。

9.一种基于OpenAPI的应用接口对接装置,其特征在于,包括:

第一生成模块,用于通过应用后端生成开放应用编程接口OpenAPI的规范文件;

第二生成模块,用于通过所述规范文件生成应用前端可嵌入使用的软件开发工具包SDK;

对接模块,用于根据所述SDK将所述应用前端和所述应用后端进行对接。

10.一种计算机存储介质,其特征在于,所述计算机存储介质包括存储的程序,其中,在所述程序运行时控制所述计算机存储介质所在设备执行权利要求1至8中任意一项所述的

基于OpenAPI的应用接口对接方法。

基于OpenAPI的应用接口对接方法及装置

技术领域

[0001] 本发明涉及网页开发领域,具体而言,涉及一种基于OpenAPI的应用接口对接方法及装置。

背景技术

[0002] 在Web应用程序的开发过程,前后端API对接往往是一个非常重要但是又费时费力的过程。业界目前一般有两种方案:第一种是后端研发人员在开发环境中部署开发好的RESTful接口,前端研发人员使用这个接口环境不断进行调试。第二种是前后端研发人员提前约定并编写好API接口文档,然后根据编写好的文档并行开发,最后再在开发环境上进行真实数据的调试工作。

[0003] 第一种方案的最大问题在于效率低下,后端研发人员需要先完成或部分完成应用接口的开发,并部署好环境,前端开发人员才能进行接口联调。这大大减少了并行开发的时间,极大的影响了整体开发效率。图1是本申请现有技术的第一种方案的操作时序的示意图,如图1所示,其实际用时可以达到4个多月。另外由于对接依赖于真实部署的API接口服务,导致在联调过程中的接口修改需要进行重新编译部署,这又额外增加的对接的时间、人力成本。

[0004] 第二种方案的问题是多了API文档这一层,前后端研发需要基于这个文档进行开发。首先文档的编写需要额外的人力成本,其次还容易出现文档更新不及时或者编写错误导致与实际的API不匹配等问题,这也会导致不必要的对接成本,最后API文档并不能避免前端人员实际调用接口可能存在的参数错误等问题,还是需要后期大量的联调才能暴露和解决类似问题。图2是本申请现有技术的第二种方案的操作时序的示意图,如图2所示,其实际用时可以达到将近5个月。

[0005] 现有的两种方案主要有以下几个缺点:前端需要等待后端开发并部署真实的API环境才能进行对接,开发效率低下;需要编写单独的API文档,造成额外的时间成本,并且增加了出错的机会;无法保证前端编写API对接代码的正确性,后期还是需要在真实环境中花大量时间联调确认。

[0006] 针对上述的问题,目前尚未提出有效的解决方案。

发明内容

[0007] 本发明实施例提供了一种基于OpenAPI的应用接口对接方法及装置,以至少解决相关技术中的基于API的应用接口对接方法,对接效率低,对接时间长的技术问题。

[0008] 根据本发明实施例的一个方面,提供了一种基于OpenAPI的应用接口对接方法,包括:通过应用后端生成开放应用编程接口OpenAPI的规范文件;通过所述规范文件生成应用前端可嵌入使用的软件开发工具包SDK;根据所述SDK将所述应用前端和所述应用后端进行对接。

[0009] 可选的,通过所述应用后端生成所述OpenAPI的所述规范文件,包括:通过所述应

用后端，编写所述OpenAPI的定义部分代码；通过所述OpenAPI的工具包，对所述定义部分代码进行分析，提取所述定义部分代码的注释信息；根据所述注释信息，生成所述规范文件。

[0010] 可选的，所述规范文件为所述开放应用编程接口OpenAPI的接口说明文件。

[0011] 可选的，通过所述规范文件生成所述应用前端可嵌入使用的所述SDK包括：通过所述OpenAPI的工具包，根据所述规范文件创建数据结构；将所述数据结构应用到所述应用前端可用的预设汇编语言的模板文件上，生成所述预设汇编语言的SDK，其中，所述预设汇编语言的SDK用于供所述应用前端嵌入使用。

[0012] 可选的，所述数据结构包括：第一部分和第二部分，其中，所述第一部分用于描述所述应用后端包含的接口端点和所述接口端点支持的操作方法；所述第二部分用于描述应用的实体和所述实体的属性。

[0013] 可选的，所述预设汇编语言为TypeScript语言；所述第一部分为接口部分，所述第二部分为实体部分；其中，所述接口部分与所述开放应用编程接口OpenAPI的路径部分相对应，所述实体部分与所述开放应用编程接口OpenAPI的组件部分相对应。

[0014] 可选的，根据所述SDK将所述应用前端和所述应用后端进行对接包括：将所述SDK引入所述应用前端的代码中；通过所述应用前端根据所述代码直接对所述应用后端进行接口对接。

[0015] 可选的，所述SDK还用于所述应用前端和所述应用后端对接的数据进行下列操作至少之一：参数校验、类型检查、自动补全。

[0016] 根据本发明实施例的另一方面，还提供了一种基于OpenAPI的应用接口对接装置，包括：第一生成模块，用于通过应用后端生成开放应用编程接口OpenAPI的规范文件；第二生成模块，用于通过所述规范文件生成应用前端可嵌入使用的软件开发工具包SDK；对接模块，用于根据所述SDK将所述应用前端和所述应用后端进行对接。

[0017] 根据本发明实施例的另一方面，还提供了一种计算机存储介质，所述计算机存储介质包括存储的程序，其中，在所述程序运行时控制所述计算机存储介质所在设备执行上述中任意一项所述的基于OpenAPI的应用接口对接方法。

[0018] 根据本发明实施例的另一方面，还提供了一种处理器，所述处理器用于运行程序，其中，所述程序运行时执行上述中任意一项所述的基于OpenAPI的应用接口对接方法。

[0019] 在本发明实施例中，采用通过应用后端生成开放应用编程接口OpenAPI的规范文件；通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK；根据SDK将应用前端和应用后端进行对接的方式，通过生成OpenAPI的规范文件后，根据规范文件生成应用前端可嵌入使用的SDK，从而使应用前端根据SDK进行对接，与应用后端较早的并行开发进行对接，达到了在较短的时间内完成应用前端和应用后端的对接的目的，从而实现了极大的提高对接效率，缩短对接时间的技术效果，进而解决了相关技术中的基于API的应用接口对接方法，对接效率低，对接时间长的技术问题。

附图说明

[0020] 此处所说明的附图用来提供对本发明的进一步理解，构成本申请的一部分，本发明的示意性实施例及其说明用于解释本发明，并不构成对本发明的不当限定。在附图中：

[0021] 图1是本申请现有技术的第一种方案的操作时序的示意图；

[0022] 图2是本申请现有技术的第二种方案的操作时序的示意图；

[0023] 图3是根据本发明实施例的基于OpenAPI的应用接口对接方法的流程图；

[0024] 图4是根据本发明实施方式的应用接口对接方法的操作时序的示意图；

[0025] 图5是根据本发明实施方式的接口定义部分的代码的示意图；

[0026] 图6是根据本发明实施方式的生成OpenAPI Spec文件的代码的示意图；

[0027] 图7是根据本发明实施方式的生成SDK的代码的示意图；

[0028] 图8是根据本发明实施方式的整体过程的流程图；

[0029] 图9是根据本发明实施例的一种基于OpenAPI的应用接口对接装置的示意图。

具体实施方式

[0030] 为了使本技术领域的人员更好地理解本发明方案，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分的实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都应当属于本发明保护的范围。

[0031] 需要说明的是，本发明的说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描述的本发明的实施例能够以除了在这里图示或描述的那些以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0032] 下面对本实施例出现的专业术语进行解释说明。

[0033] OpenAPI:OpenAPI定义了一个标准的、语言无关的RESTful API接口描述规范

[0034] TypeScript:TypeScript是一种由微软开发的开源、跨平台的编程语言。它是JavaScript的超集，最终会被编译为JavaScript代码。

[0035] SDK:软件开发工具包一般都是一些软件工程师为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件时的开发工具的集合。

[0036] 根据本发明实施例，提供了一种基于OpenAPI的应用接口对接方法的方法实施例，需要说明的是，在附图的流程图示出的步骤可以在诸如一组计算机可执行指令的计算机系统中执行，并且，虽然在流程图中示出了逻辑顺序，但是在某些情况下，可以以不同于此处的顺序执行所示出或描述的步骤。

[0037] 图3是根据本发明实施例的一种基于OpenAPI的应用接口对接方法的流程图，如图3所示，该方法包括如下步骤：

[0038] 步骤S302，通过应用后端生成开放应用编程接口OpenAPI的规范文件；

[0039] 步骤S304，通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK；

[0040] 步骤S306，根据SDK将应用前端和应用后端进行对接。

[0041] 通过上述步骤，采用通过应用后端生成开放应用编程接口OpenAPI的规范文件；通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK；根据SDK将应用前端和应用后

端进行对接的方式，通过生成OpenAPI的规范文件后，根据规范文件生成应用前端可嵌入使用的SDK，从而使应用前端根据SDK进行对接，与应用后端较早的并行开发进行对接，达到了在较短的时间内完成应用前端和应用后端的对接的目的，从而实现了极大的提高对接效率，缩短对接时间的技术效果，进而解决了相关技术中的基于API的应用接口对接方法，对接效率低，对接时间长的技术问题。

[0042] 上述通过应用后端生成OpenAPI的规范文件在应用后端进行，由于OpenAPI的规范文件是OpenAPI协议的较小一部分，因此上述步骤通过应用后端在较短的时间内完成，通常用时可以在10天左右，上述规范文件可以为OpenAPI的Spec文件。前端开发人员无需等待真实的API开发和部署，就可以根据Spec文件进行数据对接工作，保证了并行开发的效率。

[0043] 上述前端在使用上述规范文件进行数据对接工作时，不能直接对规范文件进行操作，通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK，根据SDK进行数据对接工作。从而保证了在开发早期就可以使应用前端和应用后端并行开发，极大的缩短了OpenAPI的开发时间，进而解决了相关技术中的基于API的应用接口对接方法，对接效率低，对接时间长的技术问题。

[0044] 可选的，通过应用后端生成OpenAPI的规范文件，包括：通过应用后端，编写OpenAPI的定义部分代码；通过OpenAPI的工具包，对定义部分代码进行分析，提取定义部分代码的注释信息；根据注释信息，生成规范文件。

[0045] 由于OpenAPI定义了一个标准的、语言无关的RESTful API接口描述规范，因此，可以直接利用语言原生的注释语法编写API相关的定义，在应用后端成功编写好上述OpenAPI的接口定义部分的代码后，通过OpenAPI的工具包，例如openapi-tools，对定义部分代码进行分析，提取定义部分代码的注释信息；根据注释信息，组合成符合OpenAPI规范的Spec文件，即规范文件。由于生成OpenAPI Spec的代码或注释本身就是后端程序代码的一部分，避免了单独编写API文档的时间成本和容易出错的问题。

[0046] 可选的，规范文件为开放应用编程接口OpenAPI的接口说明文件。也即是上述Spec文件。

[0047] 可选的，通过规范文件生成应用前端可嵌入使用的SDK包括：通过OpenAPI的工具包，根据规范文件创建数据结构；将数据结构应用到应用前端可用的预设汇编语言的模板文件上，生成预设汇编语言的SDK，其中，预设汇编语言的SDK用于供应用前端嵌入使用。

[0048] 由于规范文件不能被应用前端直接使用，因此，通过前端可用的预设汇编语言，将规范文件生成预设汇编语言的SDK，可以供应用前端直接使用。具体的，通过OpenAPI的工具包，根据规范文件创建数据结构；将数据结构应用到应用前端可用的预设汇编语言的模板文件上，生成预设汇编语言的SDK。

[0049] 可选的，上述规范文件的数据结构包括：第一部分和第二部分，其中，第一部分用于描述应用后端包含的接口端点和接口端点支持的操作方法；第二部分用于描述应用的实体和实体的属性。

[0050] 上述预设汇编语言为TypeScript语言，可以供应用前端直接使用；上述第一部分为接口部分，也即是api部分，第二部分为实体部分，也即是model部分；其中，接口部分与开放应用编程接口OpenAPI的路径部分相对应，实体部分与开放应用编程接口OpenAPI的组件部分相对应。上述OpenAPI的路径部分为paths部分，描述了后端程序包含的接口端点

(endpoint)及其支持的操作方法(method)。上述OpenAPI的组件部分为components部分,描述了应用程序的实体及其属性。从而将规范文件成功转化为应用前端可用的SDK,使应用前端可以根据SDK实现对规范文件的执行,保证了应用前端和应用后端的并行开发。

[0051] 可选的,根据SDK将应用前端和应用后端进行对接包括:将SDK引入应用前端的代码中;通过应用前端根据代码直接对应用后端进行接口对接。

[0052] 只需要将SDK引入前端应用的代码中,就可以直接对接后端的接口,实现应用前端和应用后端的并行开发。

[0053] 可选的,SDK还用于应用前端和应用后端对接的数据进行下列操作至少之一:参数校验、类型检查、自动补全。

[0054] 借助于TypeScript SDK本身的类型检查、自动补全等特性,前端开发可以高效准确的调用后端API接口,可以大大缩减联调所需的时间,对开发后期应用前端和应用后端的对接,有效提高对接的效率和准确性。进而提高了应用前端和应用后端的对接效率,缩短对接时间。进而解决了相关技术中的基于API的应用接口对接方法,对接效率低,对接时间长的技术问题。

[0055] 需要说明的是,本实施例还提供了一种可选的实施方式,下面对该实施方式进行详细说明。

[0056] 本实施方式的核心是基于应用程序后端的代码或注释生成标准的OpenAPI Spec文件,通过自动化的方式将OpenAPI Spec转换成前端可用的TypeScript SDK,再利用TypeScript的强类型检查等特性高效、准确的进行API对接。

[0057] 图4是根据本发明实施方式的应用接口对接方法的操作时序的示意图,如图4所示,本实施方式的步骤如下:

[0058] 首先,后端开发人员在初期只需要编写少量OpenAPI相关的代码或注释,来生成OpenAPI Spec文件。前端开发人员无需等待真实的API开发和部署,就可以进行数据对接工作,保证了并行开发的效率。

[0059] 其次,由于生成OpenAPI Spec的代码或注释本身就是后端程序代码的一部分,避免了单独编写API文档的时间成本和容易出错的问题。

[0060] 最后,使用OpenAPI Spec文件生成的TypeScript SDK可以直接被前端代码引用。借助于TypeScript本身的类型检查、自动补全等特性,前端开发可以高效准确的调用后端API接口,可以大大缩减联调所需的时间。

[0061] 本实施方式的方案描述了一种基于OpenAPI的应用接口对接方法,并提供了执行相应方法的工具包openapi-tools。

[0062] 本实施方式包含的主要对接步骤包括:

[0063] 基于后端代码生成OpenAPI Spec;

[0064] 基于OpenAPI Spec生成TypeScript SDK;

[0065] 前端代码嵌入和使用TypeScript SDK。

[0066] OpenAPI定义了一个标准的、语言无关的RESTful API接口描述规范。使用例如Node.js、PHP、Go等语言编写程序后端代码的时候,可以直接利用语言原生的注释语法编写API相关的定义,例如,图5所示的代码,图5是根据本发明实施方式的接口定义部分的代码的示意图。

[0067] 当接口定义部分的代码在后端程序中编写好之后，就可以利用openapi-tools对源码进行分析，提取出这些注释信息，并组合成符合OpenAPI规范的Spec文件。具体的代码例如图6所示的代码，图6是根据本发明实施方式的生成OpenAPI Spec文件的代码的示意图。这一步类似许多主流的编程语言、框架对应的swagger-php、swagger-express、goswagger等工具做的事情。

[0068] 生成了OpenAPI Spec文件后，第二步就是用它生成TypeScript语法的SDK。openapi-tools会自动根据OpenAPI Spec文件创建一个数据结构，这个数据结构包含api和model两部分。其中api部分对应OpenAPI的paths部分，它描述了后端程序包含的接口端点(endpoint)及其支持的操作方法(method)。Model部分对应了OpenAPI的components部分，描述了应用程序的实体及其属性。然后将这个数据结构应用到TypeScript语法的模板文件上，最后就得到了与OpenAPI Spec对应的前端SDK了。具体的代码如图7所示，图7是根据本发明实施方式的生成SDK的代码的示意图。

[0069] 最后，只需要将SDK引入前端应用的代码中，就可以直接对接后端的接口，并且该方案还提供了参数智能补全、自动校验等功能，提高了对接的效率和准确性。

[0070] 整体过程如图8所示，图8是根据本发明实施方式的整体过程的流程图，具体如下：

[0071] 1、后端接口定义；

[0072] 2、OpenAPI Spec文件；

[0073] 3、前端SDK；

[0074] 4、使用SDK。

[0075] 本实施方式基于同一套装置自动实现从后端接口定义到前端SDK的转换和生成工作；生成的SDK具有参数校验、类型检查、自动补全等功能。

[0076] 本实施方式使得应用接口对接工作更好的解耦，优化了并行开发的时间，提高了应用整体开发的效率，并有利于缩短项目整体交付时间。

[0077] 本实施方式已经在实际项目中被采用并验证，结果证明能够极大的提高前后端接口对接工作的效率和准确性。

[0078] 图9是根据本发明实施例的一种基于OpenAPI的应用接口对接装置的示意图，如图9所示，根据本发明实施例的另一方面，还提供了一种基于OpenAPI的应用接口对接装置，包括：第一生成模块92，第二生成模块94和对接模块96，下面对该装置进行详细说明。

[0079] 第一生成模块92，用于通过应用后端生成开放应用编程接口OpenAPI的规范文件；第二生成模块94，与上述第一生成模块92相连，用于通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK；对接模块96，与上述第二生成模块94相连，用于根据SDK将应用前端和应用后端进行对接。

[0080] 通过上述装置，采用第一生成模块92通过应用后端生成开放应用编程接口OpenAPI的规范文件；第二生成模块94通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK；对接模块96根据SDK将应用前端和应用后端进行对接的方式，通过生成OpenAPI的规范文件后，根据规范文件生成应用前端可嵌入使用的SDK，从而使应用前端根据SDK进行对接，与应用后端较早的并行开发进行对接，达到了在较短的时间内完成应用前端和应用后端的对接的目的，从而实现了极大的提高对接效率，缩短对接时间的技术效果，进而解决了相关技术中的基于API的应用接口对接方法，对接效率低，对接时间长的技术问题。

[0081] 根据本发明实施例的另一方面，还提供了一种计算机存储介质，计算机存储介质包括存储的程序，其中，在程序运行时控制计算机存储介质所在设备执行上述中任意一项的基于OpenAPI的应用接口对接方法。

[0082] 根据本发明实施例的另一方面，还提供了一种处理器，处理器用于运行程序，其中，程序运行时执行上述中任意一项的基于OpenAPI的应用接口对接方法。

[0083] 上述本发明实施例序号仅仅为了描述，不代表实施例的优劣。

[0084] 在本发明的上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详述的部分，可以参见其他实施例的相关描述。

[0085] 在本申请所提供的几个实施例中，应该理解到，所揭露的技术内容，可通过其它的方式实现。其中，以上所描述的装置实施例仅仅是示意性的，例如所述单元的划分，可以为一种逻辑功能划分，实际实现时可以有另外的划分方式，例如多个单元或组件可以结合或者可以集成到另一个系统，或一些特征可以忽略，或不执行。另一点，所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口，单元或模块的间接耦合或通信连接，可以是电性或其它的形式。

[0086] 所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元显示的部件可以是或者也可以不是物理单元，即可以位于一个地方，或者也可以分布到多个单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0087] 另外，在本发明各个实施例中的各功能单元可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。

[0088] 所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来，该计算机软件产品存储在一个存储介质中，包括若干指令用以使得一台计算机设备(可为个人计算机、服务器或者网络设备等)执行本发明各个实施例所述方法的全部或部分步骤。而前述的存储介质包括:U盘、只读存储器(ROM,Read-Only Memory)、随机存取存储器(RAM,Random Access Memory)、移动硬盘、磁碟或者光盘等各种可以存储程序代码的介质。

[0089] 以上所述仅是本发明的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明原理的前提下，还可以做出若干改进和润饰，这些改进和润饰也应视为本发明的保护范围。


2020
Jan
Feb
Mar
Apr
May
2020
后端接口开发
60 days
Jan 1 - Mar 24
前端应用对接
30 days
Mar 25 - May 5

图1

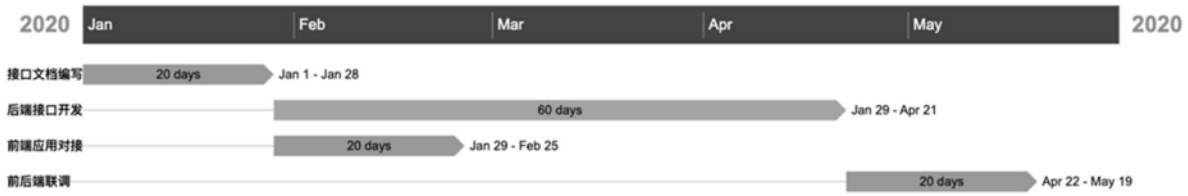

2020
Jan
Feb
Mar
Apr
May
2020
接口文档编写
20 days
Jan 1 - Jan 28
后端接口开发
60 days
Jan 29 - Apr 21
前端应用对接
20 days
Jan 29 - Feb 25
前后端联调
20 days
Apr 22 - May 19

图2

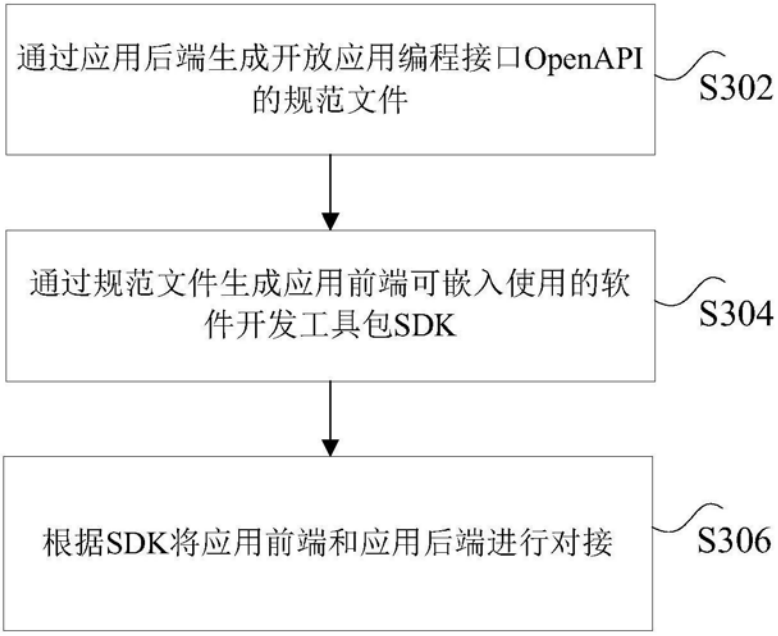

通过应用后端生成开放应用编程接口OpenAPI的规范文件
S302
通过规范文件生成应用前端可嵌入使用的软件开发工具包SDK
S304
根据SDK将应用前端和应用后端进行对接
S306

图3

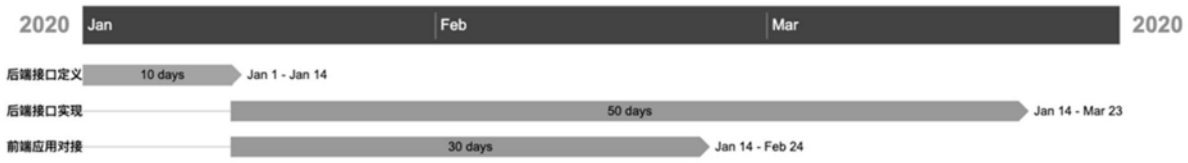

2020
Jan
Feb
Mar
2020
后端接口定义
10 days
Jan 1 - Jan 14
后端接口实现
50 days
Jan 14 - Mar 23
前端应用对接
30 days
Jan 14 - Feb 24

图4

```
// User defines a user
type User struct {
  // id of user
  ID int64 `orm:"auto;column(id)" json:"id" view:"nested"`
  // name of user
  Name string `json:"name" view:"nested"`
  // email of user
  Email string `json:"email"`
  // password of user
  Password string `json:"-"`
  // enable the user or not
  Enabled bool `json:"enabled"`
  // roles of user
  Roles string `json:"-"`
  // external id of user
  ExternalID string `orm:"null;column(external_id)"`
  // identity platform of user
  IdentityPlatform *IdentityPlatform `orm:"null;rel(fk);"`
  // setting of dashboard
  DashboardSetting string `orm:"null;type(text)" json:"-"`
  // time of creating user
  Create time.Time `orm:"auto_now_add;type(datetime)" json:"create"`
  // time of last password update
  PasswordLastUpdate time.Time `orm:"auto_now_add;"`

  // roles of user
  RolesView []string `orm:"-" json:"roles"`
}
```

图5

```
"User": {
    "type": "object",
    "properties": {
        "name": {
            "type": "string",
            "description": "name of user"
        },
        "roles": {
            "items": {
                "type": "string"
            },
            "type": "array",
            "description": "roles of user"
        },
        "identity_platform": {
            "description": "identity platform of user",
            "$ref": "#/components/schemas/IdentityPlatform_Nestview"
        },
        "create": {
            "type": "string",
            "description": "time of creating user",
            "format": "date-time"
        },
        "enabled": {
            "type": "boolean",
            "description": "enable the user or not"
        },
        "id": {
            "type": "integer",
            "description": "id of user",
            "format": "int64"
        },
        "password_last_update": {
            "type": "string",
            "description": "time of last password update",
            "format": "date-time"
        },
        "external_id": {
            "type": "string",
            "description": "external id of user"
        },
        "email": {
            "type": "string",
            "description": "email of user"
        }
    },
    "title": "User"
},
```


图6

```
export interface User {
    /**
     * name of user
     * @type {string}
     * @memberof User
     */
    name?: string;
    /**
     * roles of user
     * @type {Array<string>}
     * @memberof User
     */
    roles?: Array<string>;
    /**
     *
     * @type {IdentityPlatformNestview}
     * @memberof User
     */
    identityPlatform?: IdentityPlatformNestview;
    /**
     * time of creating user
     * @type {Date}
     * @memberof User
     */
    create?: Date;
    /**
     * enable the user or not
     * @type {boolean}
     * @memberof User
     */
    enabled?: boolean;
    /**
     * id of user
     * @type {number}
     * @memberof User
     */
    id?: number;
    /**
     * time of last password update
     * @type {Date}
     * @memberof User
     */
    passwordLastUpdate?: Date;
    /**
     * external id of user
     * @type {string}
     * @memberof User
     */
    externalId?: string;
    /**
     * email of user
     * @type {string}
     * @memberof User
     */
    email?: string;
}
```

图7

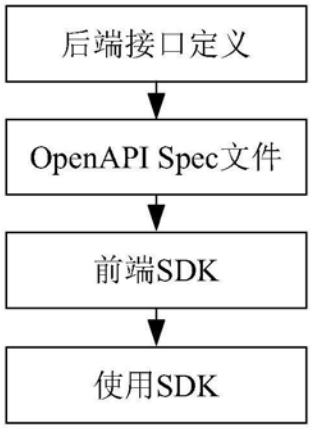

后端接口定义
OpenAPI Spec文件
前端SDK
使用SDK

图8


第一生成模块92
第二生成模块94
对接模块96

图9