Boot 进行公共投诉系统后端应用程序的设计和开发

:

:

:

:

二○二○年二月

by

:

February 2020

摘　要

电子政务是治理领域中（信息和通信技术）的一种实施，用于改善政府或公共部门向公众提供的服务。例如智慧城市，在线认证服务，社区投诉服务等。这篇文章的目的是为了发展基于使用 spring boot 微服务框架的应用来构建的公共社区投诉服务应用。微服务架构是用来将一个应用程序划分为多个部分，或者基于业务流程将许多微服务与服务互连，从而成为具有完整业务流程的单个应用程序。该体系结构的优势之一是可以添加更多微服务而不影响其他微服务。该应用程序已部署在可以通过浏览器访问的云环境中。

关键词：电子政务，微服务，投诉服务，spring boot

ABSTRACT

Key words:

1 介绍

电子政务包括使用电子通讯技术，例如互联网，来改善公民对公共服务的访问 [1]。电子政务的实施提高了报告管理系统的管理效率和速度，同时也提高了政府管理流程的透明度。通过它，良好治理的方面体现出来了。但是，印度尼西亚电子政务的实施面临许多问题，例如开发和操作电子政务应用程序的费用问题，技术问题（例如安全性问题，隐私和系统更新）以及人力资源问题，这些方面缺乏能力去管理这样一个系统。云技术成为解决这些问题的替代解决方案之一。该模型允许消费者通过在各种设备中访问的提供的程序在线使用云中存在的应用程序，而不必去担心以上的这些问题。

微服务技术的使用可以给在云中的电子政务系统提供一些优势。微服务中的模块化概念允许在应用程序的管理中单独存在的服务，除此之外对特定服务的开发不会干扰其他服务造成影响。可以在其他服务中区分不同服务的能力构建，以便适当地使用资源。同样，可以使用不同的编程语言来开发服务 [2]。

2 相关工作

Sam Newman [3den 文章中描述] 开发微服务应用程序，首先要进行的工作是确定开发环境或有限上下文的边界。简而言之，要可以从指定的应用程序业务流程中看到的应用程序的有限上下文，然后可以根据用户的功能分组，例如财务部门负责付款，而仓库部门则负责客户订单。然后每个功能都将会放到一起统一在一个模块中，然后，该模块将成为有界上下文，以创建与所制作模块的目的相匹配的微服务。已经制作完成的微服务将应用模块与模块之间是相互松散耦合的，制作单个模块又有高凝聚力的微服务概念。最后，确定对数据库进行读写的模块。

Purnama 和 Yatini [4] 使用 Node.js 开发了一个论文管理应用程序，目的是避免内容或论文标题的任何相似之处，当出现这样的情况时，通常有抄袭的嫌疑。Node.js 是使用微服务构建的，他的目的在于简化应用程序开发的体系结构。当有新的功能添加的时候，无需重新创建应用程序，该新功能可以独立的添加，这样就可以花费更少的时间做应用的进一步开发。

Janssen 和 Joha [5] 认为，公共部门仍然很少使用软件即服务（SaaS）模型，尽管公共/电子政务部门中的 SaaS 有望带来许多优势，例如节省成本，提高效率，但是存在严峻的挑战，例如质量，安全性，隐私性，以及需要在一个地区与其他地区定制不同的系统。

3 方法

3.1 功能需求分析

表 3 功能需求

ID	参与者	功能需求
FR1	管理员	用户注册
FR2	管理员	公民身份证管理
FR3	管理员	分类管理
FR4	公民	身份验证
FR5	公民	发送投诉
FR6	公民	检查投诉
FR7	政府工作单位	显示投诉摘要
FR8	政府工作单位	回答投诉
FR9	政府工作单位	删除投诉
FR10	供应商	查看所有客户

可以通过查看一些类似的应用程序进行功能需求分析，例如，Kediri 市的公共投诉 Web 应用程序 [6]。通过查看应用程序的设计文档可以进行分析，此外，还可以从文献研究中进行分析，以找到一些与电子政务公共报告应用程序的功能要求有关的信息。

功能需求包括参与者需求和功能需求。参与者是将使用该应用程序的用户，该例中使用该应用程序的参与者很少，只包括管理员，供应商，公民和政府部门。功能需求本身就是应用程序的功能。表 5 列出了该应用程序的一些功能要求。

3.2 微服务建模

提出功能要求后，下一步就是对微服务进行建模，此步骤将功能需求（也称为有限上下文）划分为一些符合有界上下文目的的微服务。换句话说，微服务是由一个或多个微服务组成的组，它们相互连接以执行业务流程或功能。表 3.1 中显示了此应用程序中的微服务。

表 3.1 微服务需求

ID	**微服务需求**
FR1	创建一个新的客户数据
FR2	创建公民 ID 数据
FR2	显示公民身份数据列表
FR2	删除公民 ID 数据
FR3	创建一个类别
FR3	删除一个类别
FR3	显示类别列表
FR4	获取市民 ID 数据
FR5	提出新的投诉
FR5	获取票号
FR6	凭票显示投诉状态
FR7	显示所有具有特定状态的投诉
FR8	更新投诉数据中的答案
FR9	删除投诉
FR10	显示所有注册客户

3.3 设计用例

通过对功能需求的分析生成了设计用例，用例显示了参与者与应用程序之间的交互，如图 3.1 所示。

3.4 应用开发

此阶段是根据功能需求，微服务需求和用例对应用程序进行编码的过程。该研究致力于使用 Java 编程语言和 springboot 框架来开发后端应用程序，并且 Spring-boot 框架具有多个优点所以它受到广大青睐。

Spring 为开发者提供了很多的便利，它支持 MVC 并提供 RESTful Web 服务功能，它的包中也提供了数据库连接，除此之外 Spring 框架还支持依赖项注入。有了依赖注入可以在应用程序中轻松进行配置依赖，因此在应用程序开发过程中变得更加方便。每个 spring 框架都支持面向切面编程（AOP）[7]。而对于使用 spring boot 来说还具有另一个优点，那就是 tomcat 服务器可以轻松地包含在 spring boot 内并且可以直接运行 [8]。

有了后端之外，还需要开发前端应用程序来和后端进行通信以此进行业务流

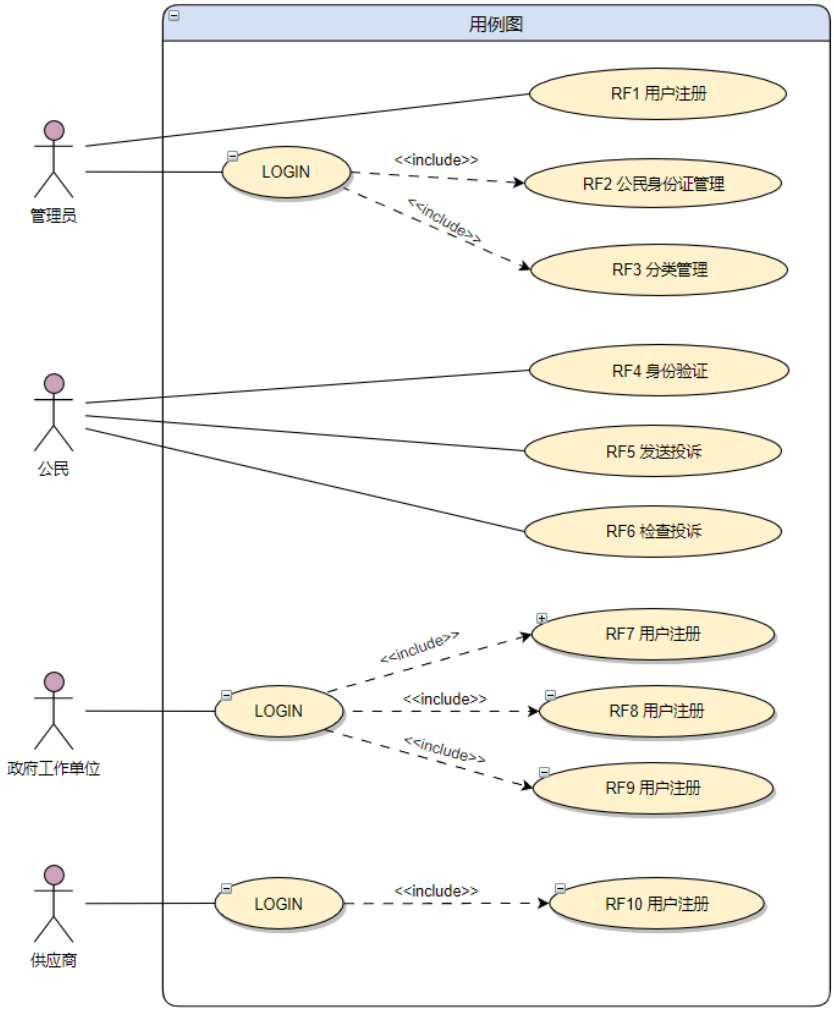

图 3.1 应用用例图

Fig. 3.1

程。在该项目中，前端开发使用 typescript 和 Angular2 框架进行。

3.5 微服务黑盒测试

黑盒测试是一项旨在检查功能性应用程序是否正确运行而又不知道该应用程序内部是否存在任何问题的测试 [9]。它的测试工作是通过创建测试用例来完成的，该用例的形式为功能应用程序预期的测试输入和输出。可以在不使用算法或低粒度级别的应用程序上进行黑盒测试 [10]，因此不需要花费很多时间 [11]。虽然完成微服务的测试，需要进行两项测试，即白盒测试和黑盒测试，但本研究仅执行黑盒测试，因为该应用程序不使用任何复杂的算法并且具有较低的粒度详细信息，所以黑盒测试就足够了。

4 应用业务流程

公共投诉应用程序由特定的单位或团队操作和管理，在某些情况下，在 Bandung 等地区 [12]，这种应用程序如果能在城市管理中心进行操作和管理，那么该应用程序将工作的最好。

此公共投诉应用程序是基于云的一个应用程序，所以地方政府可以通过向供应商支付一些钱来租用搭建应用程序服务的云平台。政府也可以指派管理员来运行应用程序，管理员拥有上传公民 ID 数据的权限。

公民通过云平台上的应用程序提出投诉。首先，市民必须根据管理员上传的数据 ID 来验证其身份，之后，他们发起的投诉将保存在数据库中，并准备由政府工作部门进行管理。在处理之前，系统有一个选择阶段来过滤一些投诉信息，选择过程由城市管理中心的政府工作单位进行管理。有效的投诉内容将按照管理员创建的类别进行分类，然后系统将其发送到城市管理中心外的相对应的正确的政府部门，以便可以得到正确的答案以及合理的后续行动来解决该条投诉。如果投诉人的报告得到处理并取得成效之后，则发送投诉的公民可以将投诉状态更改为完成，以告知政府投诉已得到充分解决。

5 应用架构

使用云技术的公共投诉应用程序具有如图 5.1 所示的云体系结构。前端应用程序以及后端应用程序以及应用程序数据库将与应用程序数据库一起存储在云中，以节省资源使用。公众可以通过浏览器访问该应用程序。通过 Web 应用程序提交的投诉将被发送到数据库中，以供政府进一步管理，在这种情况下，政府是指挥中心和其他政府工作单位。供应商可以通过后端或前端访问应用程序以对应用程序执行维护。

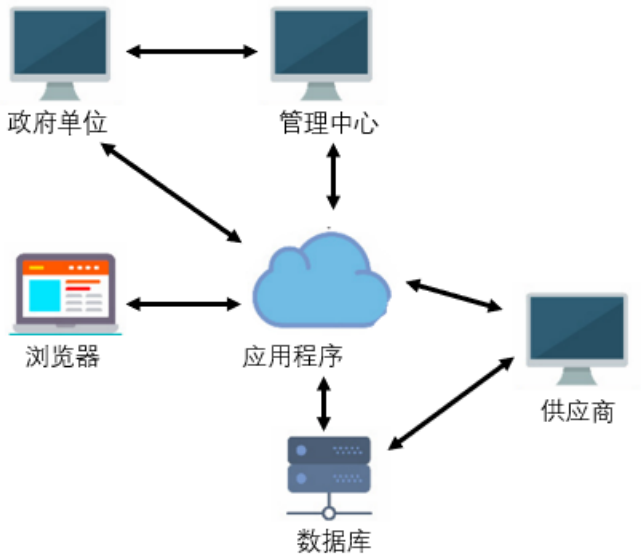

图 5.1 应用架构

Fig. 5.1

6 微服务架构

有一个工具称为 Swagger，它可以使开发人员更轻松地查看应用程序上所有可用的微服务。Swagger 是一个标准框架，它使开发人员可以快速找到和理解应用程序上的所有服务，而无需访问程序代码、应用程序开发文档，也无需检查应用程序服务网络 [13]。Swagger 是基于使用 springboot 框架创建的控制器类来映射微服务。类 controller（或可以称为 controller）是 springboot 中的类文件，由根据业务需要执行输入输出过程的方法组成。在 controller 中，可以将方法制成 REST API，通过 Swagger 就可以显示已经在应用程序中创建的所有 controller。在每个 swagger 中，都可以通过使用 controller 本身所用方法的描述来访问 REST API（所用的 REST 方法通常是 GET，POST，DELETE，PUT），然后是 REST API URL，以及 controller 中定义的方法名称，前端应用程序将使用该 REST API 通过已定义的通信方法去访问微服务。

6.1 controller 类

表 6.1 显示了应用程序中存在的 controller 类以及对应相关的微服务。

6.2 REST 方法

表 6.2 中包含应用程序中存在的 REST 方法信息以及相关的微服务。

6.3 REST URL

表 6.3 包含相关微服务的 URL，该 URL 用于在前后端之间进行通信。

6.4 架构图

前端可以分为四个用户界面（UI），分别是管理员的控制界面（Admin UI），政府工作单位的 Operator UI，公民创建和复核投诉的 UI 界面以及由供应商使用的 Vendor UI ，这里面的每个 UI 将与某个微服务相关联，如图 6.1 所示。

表 6.1 controller 类

controller 类	微服务需求
Customer Controller	创建一个新的客户数据
Citizen Controller	创建公民 ID 数据
Citizen Controller	显示公民身份数据列表
Citizen Controller	删除公民 ID 数据
Category Controller	创建一个类别
Category Controller	删除一个类别
Category Controller	显示类别列表
Citizen Controller	获取市民 ID 数据
Complaint Controller	提出新的投诉
Complaint Controller	获取票号
Complaint Controller	凭票显示投诉状态
Complaint Controller	显示所有具有特定状态的投诉
Complaint Controller	更新投诉数据中的答案
Complaint Controller	删除投诉
Customer Controller	显示所有注册客户

表 6.2 Rest 方法

Rest 方法	微服务需求
POST	创建一个新的客户数据
POST	创建公民 ID 数据
GET	显示公民身份数据列表
DELETE	删除公民 ID 数据
POST	创建一个类别
DELETE	删除一个类别
GET	显示类别列表
POST	获取市民 ID 数据
GET	提出新的投诉
GET	获取票号
GET	凭票显示投诉状态
GET	显示所有具有特定状态的投诉
PUT	更新投诉数据中的答案
DELETE	删除投诉
GET	显示所有注册客户

表 6.3 REST URL

REST URL	**微服务需求**
/customer/new	创建一个新的客户数据
/citizen/upload	创建公民 ID 数据
/rest/citizen/allby/idkokab	显示公民身份数据列表
/rest/citizen/delete/by/idkokab	删除公民 ID 数据
/rest/admin/category/new	创建一个类别
/rest/admin/category/delete/categoryid	删除一个类别
/rest/admin/category/all/idkokab	显示类别列表
/citizen/checknik	获取市民 ID 数据
/complaint/new	提出新的投诉
/complaint/new	获取票号
/complaint/find/ticket/ticketcode	凭票显示投诉状态
rest/complaint/kokabandstatus	显示所有具有特定状态的投诉
rest/complaint/update/answer/id	更新投诉数据中的答案
complaint/delete/id	删除投诉
/rest/user-dev/customer/all	显示所有注册客户

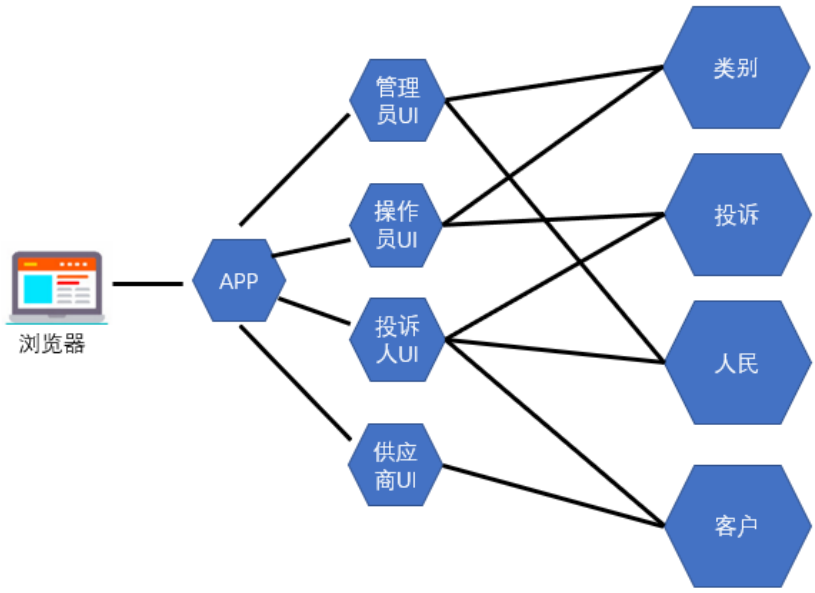

图 6.1 微服务架构

Fig. 6.1

7 结论

基于本研究已完成的过程，可以得出如下结论：

- 公众投诉既是一个应用程序也是服务，它是使用 springboot 框架通过微服务架构开发的。现在，此应用程序已部署在了云平台中，在部署应用程序时，将后端程序制作为一个 jar 包，然后将前端程序以及所使用的数据库都部署到一个云平台上面；要访问该应用程序，只需要使用一个浏览器，通过浏览器可以创建与云服务器中应用程序的通信连接。
- 在开发微服务应用程序时，应将整个系统的功能需求分解为几个微服务，然后这些微服务通过明确的业务流程相互连接来形成一个统一的应用程序。
- 使用微服务架构的优势在于不仅可以添加更多功能需求，而且微服务之间不会互相影响。这样，开发人员就可以独立添加更多功能，来创造业务价值，从而节省进一步节省开发的时间。

致 谢

这项研究由“Penelitian Laboratorium PUPT ITS 2017”研究补助金支持/资助，合同号：566 / PKS / ITS / 2017。

参考文献

参考文献

[1] D. A. Putra and N. Aminuddin, "Tactical Steps E-Government Development For Regional Government in Pringsewu Regency; Langkah –Langkah Taktis Pengembangan E-Government Untuk Pemerintahan Daerah (Pemda) Kabupaten Pringsewu," Technology Acceptance Model, p. 67, 2014.

[2] S. Sharma, "Mastering Microservices with Java," in Mastering Microservices with Java, Birmingham, PACKT publishing, 2016.

[3] S. Newman, "How to Model Services," in Building Microservices: Designing Fine-Grained Systems, O’ Reilly Media, 2014.

[4] H. Purnama and I. Yatini, "Thesis Managemen Application in STMIK AKAKOM Yogyakarta Using Microservice Architecture With Node.Js," in Seminar Riset Teknologi Informasi (SRITI), 2016.

[5] Marijn Janssen and Anton Joha, "CHALLENGES FOR ADOPTING CLOUD-BASED SOFTWARE AS A SERVICE (SAAS) IN THE PUBLIC SECTOR," in European Conference on Information Systems, 2011.

[6] Electronic Data Management Department of Kediri City Government, "Application Development Document of E-Complaint Kediri City".

[7] Mr.Anil Kumar, Dr. Arvind Kumar and Mr. M. Iyyappana, "Applying Separation of Concern for Developing Softwares Using Aspect Programming Concepts," International Conference on Computation, pp. 906-9014, 2016.

[8] I. A. Pradana, "Mengenal Spring Boot," 3 February 2017. [Online]. Available: https://www.codepolitan.com/spring-boot-pengenalan588da0c4bedd1. [Accessed 16 February 2017].

[9] M. Kumar, S. K. Singh and Dwivedi, "A Comparative Study of Black Box Testing and White Box Testing Techniques," International Journal of Advance Research in Computer Science and Management Studies, 2015.

[10] S. R. Jan, S. T. U. Shah, Z. U. Johar, Y. Shah and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," IJSRSET, 2016.

[11] A. Orso and G. Rothermel, "Software testing: a research travelogue (2000–2014)," in Proceedings of the on Future of Software Engineering, New York, 2014.

[12] A. Nurmatari, "Aplikasi Seluruh SKPD akan Terintegrasi di Bandung Command Center," 7 December 2016. [Online]. Available: https://news.detik.com/berita-jawa-barat/d-3365344/aplikasi-seluruh-skpd-akan-terintegrasi-di-bandung-command-center. [Accessed 1 Maret 2017].

[13] Swagger, "Getting Started With Swagger," [Online]. Available: https://swagger.io/getting-started/. [Accessed 10 July 2017]

[14] M. B. Jones, "draft-ietf-oauth-json-web-token-32," [Online]. Available: https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32. [Accessed 26 Juli 2017].

[15] Yung-Feng Lu, Chin-Fu Kuo and Yi-Yen Fang, "Efficient Storage Encryption for Android Mobile Devices," in Proceedings of the International Conference on Research in Adaptive and Convergent System, Odense, 2016.