Application, Information System
and Software Development Journal
Appissode Journal
Vol. 1, No. 1, Dec 2023, Page 27-33
E-ISSN: 3032-5625, P-ISSN: 3032-5269

Academic Data Publication Using REST Architecture

Lady Silk Moonlight[1], Yesy Diah Rosita[2], Teguh Arifianto[3]
[1]Politeknik Penerbangan Surabaya, Indonesia
[2]Institut Teknologi Telkom Purwokerto, Indonesia
[3]Politeknik Perkeretaapian Indonesia Madiun, Indonesia

Corresponding Author: Lady Silk Moonlight
Aeronautical Communication
Politeknik Penerbangan Surabaya, Indonesia
Email: lady@poltekbangsby.ac.id

Article History

Received Nov 21, 2023
Accepted Dec 9, 2023
Published Dec, 2023

Keywords

application programming interface, representational state transfer, REST, web services.

Abstract

Academic data publication is not a brand-new issue when it comes to data dissemination. Numerous internationally renowned universities have completed this. When it comes to publishing data, there are issues with both how the data is presented and how flexible it is for consumers to access it. Web services using REST (Representational State Transfer) and SOA (Service Oriented Architecture) enable clients to access resources through services. Two applications were created for testing purposes: one served as a resource supplier by offering a REST web service, and the other as a user of the web service. The second application was able to successfully retrieve resources by visiting the URI on the REST web service, according to the test results of the web service system that was established.

1 INTRODUCTION

The globe is becoming more interconnected and open as a result of developments in information and communication technologies. Alongside this advancement in technology, there is a corresponding rise in the complexity of data needed to fulfil present and future information demands. Both internal and external data are required to suit the demands of users and organisations. This is because the data collected can be utilised again for comparison, as a supplement to existing data, or both, in order to provide the relevant information. An organization's difficulty when implementing system integration and data publication is to create a standard architecture that will enable it to publish data to satisfy both general and specialised organisational needs. The problem with data publication is not only displaying data on a web page or application, but also providing flexibility in displaying or publishing data in the form of a standard architecture so that users can easily use it, so that published data is obtained according to user needs. The publication of university academic data is not something new or should not be published, this can be seen from several leading universities in the world that have published academic data such as Stanford University, University of California Berkeley, and other universities.

In order to ensure that published data is obtained in accordance with user needs, the issue with data publication extends beyond just presenting data on a web page or application. It also involves offering flexibility in how data is displayed or published in the form of a standard architecture so that users can readily use it. In addition, there is a wide range of data types that be submitted for publishing, including academic data as well as data from online stores, ticketing, flights, and ticketing.

Many universities have made their academic data public. Stanford University, for instance, has made its student biodata, registration data, student data searches, course classes, and programme data available. In addition, the University of California Berkeley makes available academic information about classes and majors, the information section, registration information, student profiles, class schedules, registration fee information, and more.

A REST-based web service model was created for UNPAR's SISFO needs, which included concerns about security, reliability, independence, and system scalability [1]. Another research by Cosmas Dedy Kurniawan raised the Integration of Banking Financial Reporting Based on Service Oriented Architecture where the results of his research concluded that REST web services in the implementation of Service Oriented Architecture obtained applications that can provide services that are easy to access and manage [2].

In this research, the author tries to use the REST web service with the objective of offering resources for other apps that will be used for publishing academic data in higher education. Therefore, it will be highly helpful for government agencies like DIKNAS, DIKTI, and BAN-PT, where requests for information are now handled manually or require humans to enter the data again into the relevant party's information system. Furthermore, the REST API that was created will eventually become the norm for academic data.

2 THEORETICAL BASE

Service Oriented Architecture (SOA)

The service-oriented paradigm is made up of several designs. This paradigm is used by applications to create logical solutions in service-oriented solution logic [3]. The service itself is the fundamental tenet of service orientation. Services are independent software applications with various design features that facilitate the accomplishment of service-oriented computing-related strategic objectives. Every service is assigned a unique function that consists of a collection of abilities associated with the service's primary purpose.

SOA is about connecting customers requirements with enterprise capabilities, regardless of technology landscape or arbitrary organizational boundaries [4] [5]. A service-oriented architecture (SOA) is a system made up of a number of loosely coupled components or services that can communicate with one another using common message exchange protocols and standard formats. These services are autonomous, may be found on several computers, and each has distinct objectives and outcomes.

By emphasising services as the primary instrument where the solution logic is represented in enabling the realisation of strategic goals connected with service orientation, Service-Oriented Architecture (SOA) defines a type of architecture that attempts to boost the efficiency, agility, and productivity of an organisation or enterprise. the idea of software architecture that makes use of services already present in a network system. It is crucial to remember that SOA places a high priority on loose coupling between software components, allowing for the reuse of current services in subsequent software development projects [3].

Web Services

Since web services are reusable by numerous other programmes, they are inherently more valuable than standard application functions. Because web services are reusable or may be used repeatedly by other programmes, they are more helpful than other outdated application functionalities. This is made possible by the industry-agreed standards used in web service interfaces [6].

When an application's processing functions are all web services, other applications can utilise them rather than having to recreate them. Web services can facilitate updates and lower error rates.

Web services offer open services for information cooperation and data integration that are accessible online to different parties utilising their own devices. The following are a few benefits of Web Services [7]:

1. The following are a few benefits of Web Services:Web services that are cross-platform are designed to function or be utilised across several operating systems, regardless of whether the application is desktop or mobile.
2. Programming languages of different kinds can use web services since they are language neutral and do not depend on any specific language.
3. A bridge that connects to the database without requiring a database driver or knowledge of the kind of database management system. Clients can access resources through a dedicated path provided by web services, eliminating the need for them to consider the DBMS in use.
4. Simplifying the data interchange procedure. using online services Web services are a unique route that clients are given from the server to facilitate easier and safer data interchange between businesses and their clients.
5. Reusing the various application components. Building web services can be done by leveraging previously developed functionalities rather than having to recreate the logical operations of a system.

Representational State Transfer (REST)

There has been an upward trend in the number of papers in the domain of RESTful APIs testing during recent years [8]. Distributed hypermedia systems use the Representational State Transfer (REST) software architecture. Web-based platforms host the majority of REST implementations [9]. Roy T. Fielding first presented REST in 2000 at the University of California [10]. Fielding groups REST into six categories, which are as follows:

1. Uniform interface
   You MUST choose API interfaces for system resources that are visible to API customers and adhere to them scrupulously, as the constraint name implies. There should only be one logical URI for each resource in the system, and that URI should allow users to retrieve additional or related data. A web page can always be used as a synonym for a resource.
   A single resource should not be excessively huge and should include every possible representation of everything. A resource should always include links (HATEOAS) pointing to relative URIs so that users can retrieve similar content. Furthermore, there should be adherence to certain standards for resource representations throughout the system, such as name conventions, link formats, and data formats (XML or/and JSON).

2. Client-server
   This restriction basically states that client and server apps MUST be independent of one another and capable of evolving independently of one another. All that a client needs to know is the resource URI. This is typical procedure in web development these days, so you don't need to do anything fancy. Make it easy. Stateless

3. Cacheable
   According to this requirement, client and server apps basically HAVE to be able to develop independently of one another. All that a client should be aware of are resource URIs. Nothing unusual is needed from your end; this is common procedure in web development these days. Keep things straightforward.

4. Stateless
   Roy fielding got inspiration from HTTP, so it reflected in this constraint. Make all client-server interactions stateless. The server will not store anything about the latest HTTP request the client made. It will treat every request as new. No session, no history.
   If the client application needs to be a stateful application for the end-user, where the user logs in once and does other authorized operations after that, then each request from the client should contain all the information necessary to service the request – including authentication and authorization details.

5. Layered system
   Using a layered system design, such as deploying the APIs on server A, storing data on server B, and authenticating requests in server C, is made possible by REST. Typically, a client is unable to discern whether it is connected to the end server directly or through a middleman.

6. Code on demand
   This restriction is, however, negotiable. The majority of the time, you will be delivering XML or JSON files that are static resource representations. However, you are allowed to return executable code when necessary to support a specific aspect of your application. For example, clients can call your API to obtain the rendering code for a UI widget. That's okay.

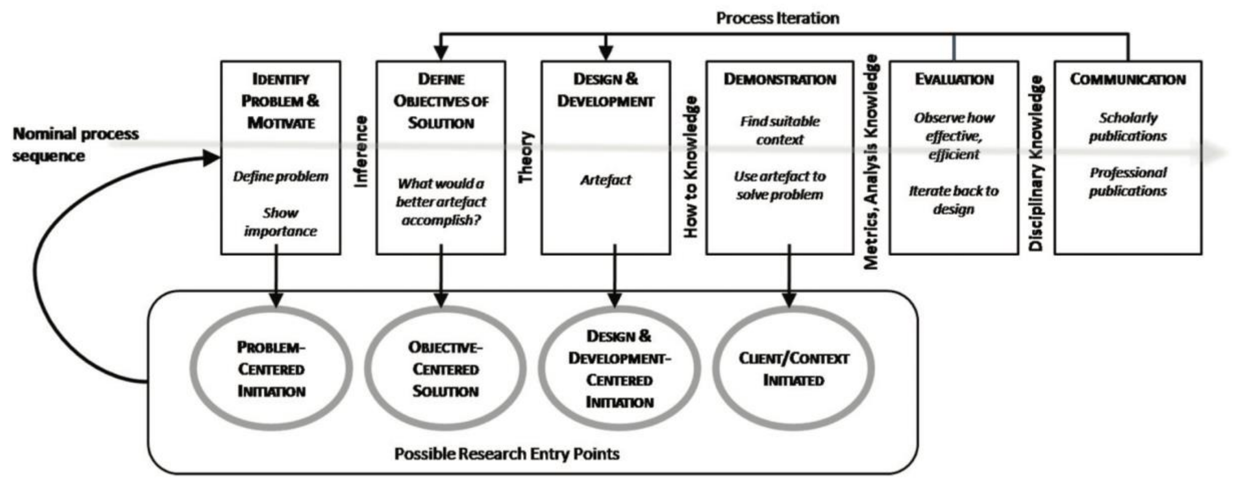
URI Format:

The following URI format is described in [10]:

1. To denote a hierarchy of resource relationships, use the trailing slash (/), as in the following example:
2. A slash (/) shouldn't be used to close it. REST APIs shouldn't finish in a slash because this may lead to more misunderstanding. For instance, many web frameworks and components produce two URIs that are identical:
3. To make URIs easier to understand, hyphens (-) should be utilised. When creating a URI, use hyphens (-) in place of spaces to make long names easier to read and translate.
4. URIs do not use underscores (_). URIs are frequently underlined in text reader apps (editors, browsers, etc.) to indicate that they can be clicked. It is preferable to use a hyphen (-) rather than an underscore (_) to prevent confusion.
5. URIs have to utilise lowercase letters. RFC 3986 (http://www.ietf.org/rfc/rfc3986.txt) specifies URIs as case-sensitive except for schemes and components.
6. The URI does not contain file extensions Typically, a URI's file name and extension are separated by a period (.).

3 METHODS

Design Science Research approach (DSRM) is the research approach employed in this study. This methodology's application is centred on system development and issue solving [11].



Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of Solution
What would a better artefact accomplish?
Theory
Design & Development
Artefact
How to Knowledge
Demonstration
Find suitable context
Use artefact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development-Centered Initiation
Client/Context Initiated
Possible Research Entry Points

Figure 1.Design Science Research (DSRM)

The current acedemic system model, if shown in diagram form, looks like this:

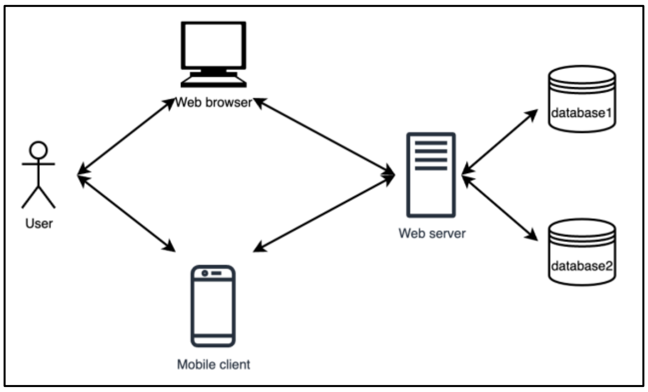


Figure 2. Existing system architecture

A new model for web services in the academic information system is as follows:

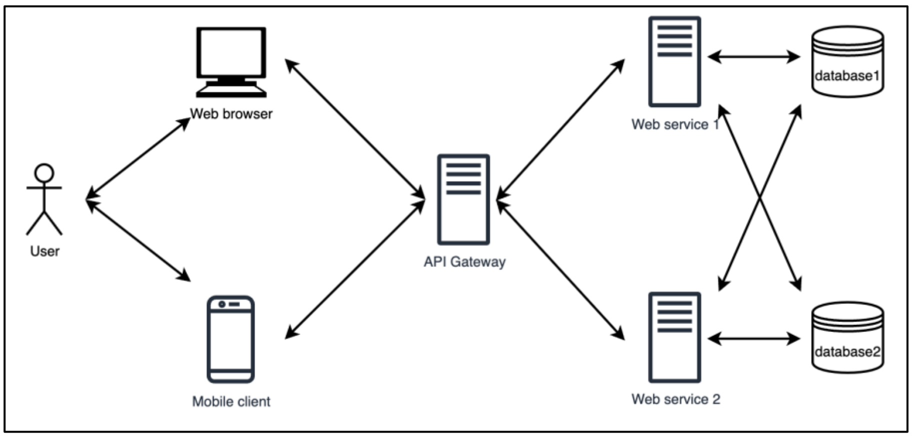


Figure 3. Developed system architecture

The following is the resource design for student data:

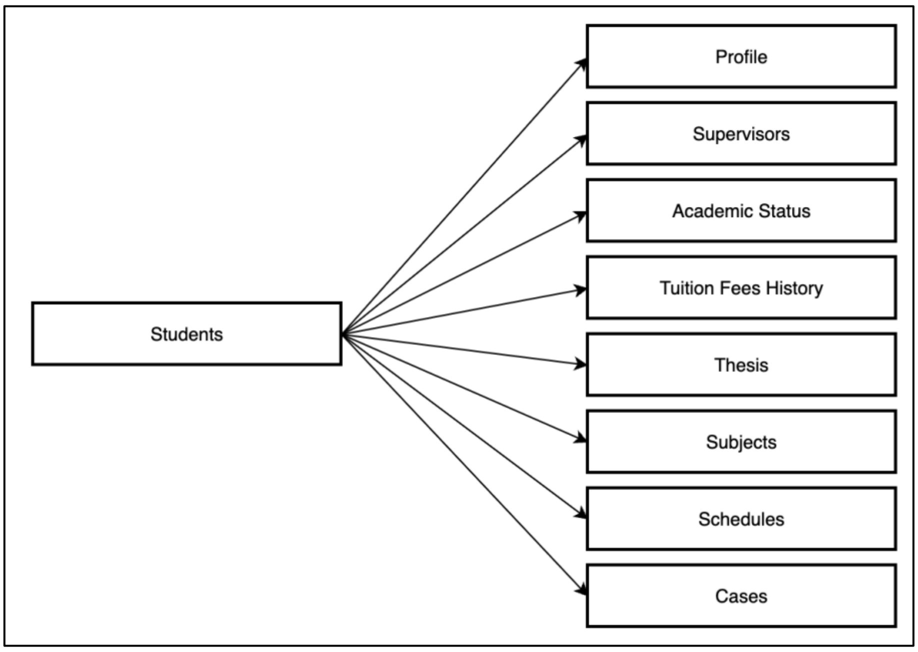


Figure 4. Student's public resources architecture

The lecturer's data resource design looks like this:

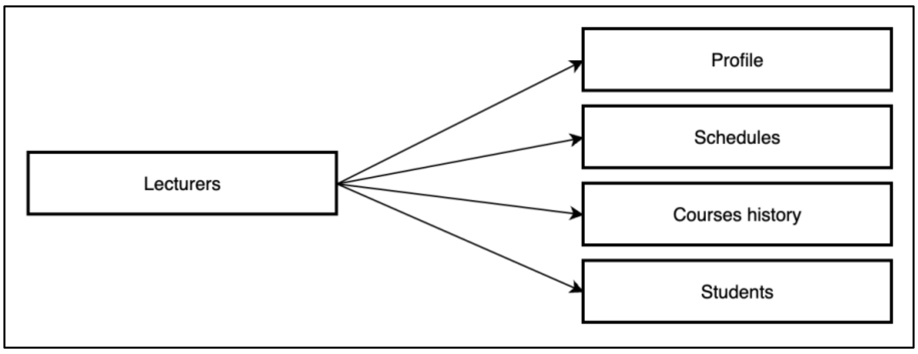


Figure 5.Lecturers public resources architecture

The following is the resource design for the course data:

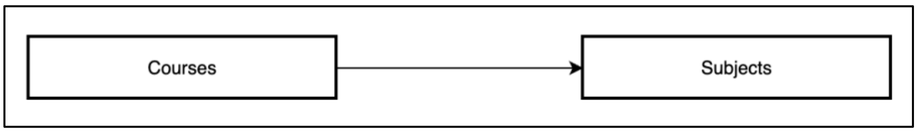


Figure 6. Courses public resources architecture

Following the creation of the student data resource hierarchy, the academic data URI design specifications are as follows for additional information:

1. Students
    a. Profile
    b. Supervisors
    c. Academic status
    d. Tuition fees history
    e. Thesis
    f. Subjects
    g. Schedules
    h. Cases
2. Lecturers
    a. Profile
    b. Schedules

    c. Courses history
    d. Students
3. Courses
    a. Subjects

4 RESULTS

The following are a few findings from this study:

1. A web service based on REST was implemented successfully.
2. The second application is used as an example to access the built service during testing.
3. Resources with the given query filter are successfully displayed by the visited URI.
4. HTTP GET or POST can be used to access services.
5. Because the web service developed in this research only employs private keys for authentication, its security is lacking.

REFERENCES

[1] M. Ghifary and G. Karya, "Pemodelan dan Implementasi Antarmuka Web Services Sistem Informasi UNPAR," Universitas Katolik Parahyangan, Bandung, 2011.

[2] C. D. Kurniawan, "Integrasi Pelaporan Keuangan Perbankan Berbasis Service Oriented Architecture," Universitas Atma Jaya, Yogyakarta, 2013.

[3] T. Erl, SOA : Principles of Service Design, New Jersey: Prentice Hall, 2008.

[4] K. Gabhart and B. Bhattacharya, Service Oriented Architecture Field Guide for Executives, New Jersey: John Wiley & Sons, Inc, 2008.

[5] I. Ahmad, E. Suwarni, R. I. Borman, Asmawati, F. Rossi and Y. Jusman, "Implementation of RESTful API Web Services Architecture in Takeaway Application Development," in *2021 1st International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, Yogyakarta, 2021.

[6] J. Hurwitz, R. Bloor, M. Kaufman and F. Halper, Service Oriented Architecture, Indianapolis: Wiley Publishing, 2009.

[7] E. Sutanta and K. Mutofa, "Strategi Pengembangan Web Services Untuk Integrasi Antar Sistem Aplikasi dan Website Dalam E-Government Di Pemkab Bantul Yogyakarta," *Jurnal Ilmiah SISFOTENIKA,* vol. 2, no. 2, pp. 1-10, 2012.

[8] A. Golmohammadi, M. Zhang and A. Arcuri, "Testing RESTful APIs: A Survey," *ACM Transactions on Software Engineering and Methodology,* vol. 33, no. 1, pp. 1-41, 2023.

[9] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences,* vol. 12, 2022.

[10] M. Masse, REST API Design Rulebook, Sebastopol: O'Reilly Media, 2012.

[11] K. Peffers, T. Tuunanen, M. Rothenberger and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems,* vol. 24, no. 3, pp. 45-77, 2007.

BIOGRAPHY OF AUTHORS

Lady Silk Moonlight

Lady Silk Moonlight was born in Surabaya, Indonesia, in 1987. She received a Bachelor of Engineering degree in Informatics Engineering from Trunojoyo University in 2009 and a Master of Engineering degree in Informatics from Bandung Institute of Technology (ITB) in 2013. Her research interests include information technology, artificial intelligence, information systems, computer science, digital image processing, and computer networks. She is a lecturer at D3 Aeronautical Communication, Politeknik Penerbangan Surabaya, Indonesia.

Yesy Diah Rosita

Yesy Diah Rosita received Master of Computer from Sekolah Tinggi Teknik Surabaya, Surabaya City, Indonesia. Her thesis about classification of infant's cry sound using Artificial Neural Network that published on IEEE, 2016. Now, she is a lecturer in Telkom University National Campus that has been located in Purwokerto, Middle Java Province, Indonesia. She interests in Speech Recognition, Deep Learning, and Decision Support System. She is also a contributor of Mathwork which is a company that specializes in mathematical computing software.

Teguh Arifianto

Teguh Arifianto was born in Tuban, Indonesia, in 1988. He received a Bachelor of Engineering degree in Informatics Engineering from Trunojoyo University in 2011 and a Master of Engineering degree in Electrical Engineering from Surabaya Sepuluh Nopember Institute of Technology (ITS) in 2016. His research interests include digital image processing, artificial intelligence, information systems, computer networks, information technology, and railway electrical. He is a lecturer at D3 Railway Electrical Technology, Politeknik Perkeretaapian Indonesia Madiun, Indonesia.