Development of REST API for Digital Advertising Application Using an Iterative Incremental Method (Case Study: Show-Up Apps)

Rifqi Rosidin[1], Deden Witarsyah[1], Muharman Lubis[1] and Haryasena Panduwiyasa[1,2]
[1]*Information System, School of Industrial and System Engineering, Telkom University, Bandung, Indonesia*
[2]*Industrial Engineering, School of Industrial and System Engineering, Telkom University, Bandung, Indonesia*

Keywords: Rest API, JSON, Postman, Black-Box Testing.

Abstract: Technological developments are rapidly growing, especially in web application development. A full-stack developer involves combining presentation with business logic, which is front-end application development that focuses on engagement and user interface, and back-end development focuses on algorithms and databases. The Show Up application which helps business owners in marketing and selling products with social media integration, currently still uses a complicated full-stack development method, making it challenging to develop and integrate because the presentation and business logic are mixed in one part of development. It takes a series of systems that can bridge the two effectively and efficiently. In this study, the Iterative Incremental method is used to develop an API that aims to integrate between sections, including the front-end and back-end programs, using the Representational State Transfer (REST) Rest API approach to the Show-up as a product application so that it can be integrated with various devices such as mobile, web, and desktop, using the JavaScript Object Notation (JSON) standard data format. The effectiveness of the Rest API is tested using the black-box method, with two tests: Unit testing and Performance testing. From unit testing, it is evident that all features work well. Then, with performance testing, the Rest API Show-up application server can receive 1,000 client requests in 1 minute and process 2,000 server requests.

1 INTRODUCTION

Amid the transformation of business digitalization in the industrial 4.0 era, various types of technology are conducted to connect systems quickly, explicitly, and flexibly to support business operations and ensure they are carried out effectively and efficiently (Panduwiyasa et al., 2021a). However, it is often found in several information systems and corporate applications that are constrained in developing the integration process and adding the correct functions to produce a compatible and robust feature when running on two devices or in a different environment (Zeebaree et al.,2020; Rosidin et al., 2020). As a result of this case, it triggers some of the application features to have malfunctioned (Panduwiyasa et al., 2021a). This problem is experienced by the advertising and digital marketing platform, namely the Show-Up Application.

Show-Up is an integrated business platform that facilitates the user to advertise, distribute, and sell the product with the help of social media. The problem started when the integration of data processing and technical functions between the main web-based applications and social media, which were mostly mobile-based, could not be carried out due to differences in the types of devices and the development of the devices, both due to programming languages and application environments (Rosidin et al., 2020; Panduwiyasa et al., 2021b). A step that is needed to be taken as a solution to the integration process of two applications or systems with different types of devices and development environments efficiently is to implement the Application Programming Interface or API (Hidayat et al., 2020).

API is an interface used as an intermediary between different applications based on the development method, programming language or physical system (Barros et al., 2020). The development of API-based applications will allow the back-end and front-end to be used more widely, as they are able to separate concrete system logic from presentation engagement, or what is commonly called the user interface (Neumann et al., 2021). This API concept can be use-

Rosidin, R., Witarsyah, D., Lubis, M. and Panduwiyasa, H.
Development of REST API for Digital Advertising Application Using an Iterative Incremental Method (Case Study: Show-Up Apps).
DOI: 10.5220/0012440200003848

In *Proceedings of the 3rd International Conference on Advanced Information Scientific Development (ICAISD 2023)*, pages 25-31
ISBN: 978-989-758-678-1

ful for users to make it easier to access data; for example, with a web service on the website system, data changes occur, and the data system in Android applications can automatically change. With APIs created by other users, they cannot directly access the database, but they have to request it via the internet and then access the available APIs (Neumann et al., 2021).

This study uses the Representational State Transfer (REST) architecture as the business logic of the Show Up application. This REST architecture was chosen over Simple Object Access Protocol (SOAP) because REST is a client-server architecture where requests are sent by the client to the server, then processed, and the appropriate response is sent back to the client. REST, showing that REST services are not limited to XML but can also support JavaScript Object Notation (JSON), as well as plain text, unlike SOAP which only supports documents in XML format (Neumann et al., 2021; Chatterjee, 2020). The purpose of making the API for the Show Up application product is to facilitate integration between applications such as web, mobile, and desktop, so that users can be flexible in using the platform when accessing the Show Up application[9]. The API development using JSON as a standard form of data communication and JSON Web Token (JWT) as a system user authentication code. The Iterative Incremental method was chosen in this study because business needs are always evolving and API needs are constantly changing, so this method is suitable because system development can be done in stages (Panduwiyasa et al., 2021a).

2 LITERATURE REVIEW

2.1 REST (Representational State Transfer)

REST (Representational State Transfer) is a web-based architecture using the HTTP (Hypertext Transfer Protocol) protocol as data communication (Li et al., 2016) . The implementation of REST architecture on the server can be done independently of each other (Hu, 2021). In this case, the client implementation is independently done without interfering with the server or backend. The REST architecture has standard request methods, namely GET, POST, PUT, and DELETE (Hu, 2021). GET is used to retrieve data from the server. PUT is used to update data on the server. POST is used to send data to the server (Jánoky et al., 2018). Then DELETE is used to delete data on the server. Web services based on the REST architecture are known as RESTful Web API services. This web service uses HTTP methods to implement REST architecture concepts. There are several principles for designing REST, respectively:

1. Addressability

   Each resource requires at least one associated URI for a REST service. URI is used to specify a resource or set of resources.

2. Statelessness

   A client-to-server request must contain all the information necessary to understand the request and cannot use the context stored on the server.

3. Cacheable

   Data marked as cacheable is stored on the system and can be reused in the future in response to the same subsequent request, rather than producing the same result iteratively. Cache constraints are used to allow response data to be marked as cacheable or non-cacheable.

4. Uniform Interface

   Applying general software engineering principles to component interfaces creates a unified interface, simplifies overall system architecture, and improves dialog visibility.

2.2 JSON Web Token

JSON Web Token is an open standard (RFC 7519) that defines a way to transmit information as a JSON object (Jánoky et al., 2018). this information can be verified and trusted because it is digitally assigned JWT can be signed using hidden (HMAC algorithm) or public/private key pair using RSA or ECDSA. Claims can then be encrypted, as JSON Web Encryption (JWE), or can be digitally signed or protected using JSON Web Signature (JWS). JWE specified in RFC7516 (Jones et al., 2015). The JSON format is specified in RFC8259 (Brier, 2020). JWS consists of three parts. Headers are used to secure claims, the payload, or body, describes the claim in JSON format, and the least is message signature or authentication code on base64 url to encode header and payload.

2.3 Software Testing

Software testing is the process of running a program or system involving any activity to find problems, bugs or other properties that interfere with the capabilities of the program or system and determine that the program or system meets the required results (Garousi et al., 2020). The software testing steps are divided into three sets, respectively:

1. Unit testing is a testing method in which the programmer unit tests the program to see if it is usable or not.
2. Integration testing, namely integration testing of two or more application units.
3. Software testing validation testing for to ensure that the software meets functional requirements

2.4 Blackbox Testing

Black box is a testing method where data testing comes from predetermined functional requirements without considering the final structure of the program, this test is also known as functional testing (Brier, 2020; Sutiah and Supriyono, 2021). Because it is only related to the functionality of the software module, black box testing also refers to functional testing, a test method carried out on the implementation of functions and verification of input and output data (Nidhra, 2012). Black box testing tries to find errors in several categories, including:

1. Functions that are not appropriate or non-exist
2. Interface error
3. Errors in data structures or database access
4. Performance errors
5. Error initialization and termination

3 METHODOLOGY

The method used in application development research is iterative incremental. This method was chosen because this method can be used in making Rest API (Rosidin et al., 2020), Iterative incremental model is a system development method that develops based on the waterfall model problem. This model combines elements of waterfall model in an iterative condition. The stages of development in this method consist of four main stages (Susanto and Meiryani, 2019).

Based on the Figure 1, At each stage of development in this methodology there are inputs and outputs. All of the input and output can be concluded (Rosidin et al., 2020);

1. The inception stage,
   Is the first stage that focuses on the beginning of research process, such as making a system background, compiling business problems and identifying risks, defining the scope of research to understand and resolve the problem with ideated solution,

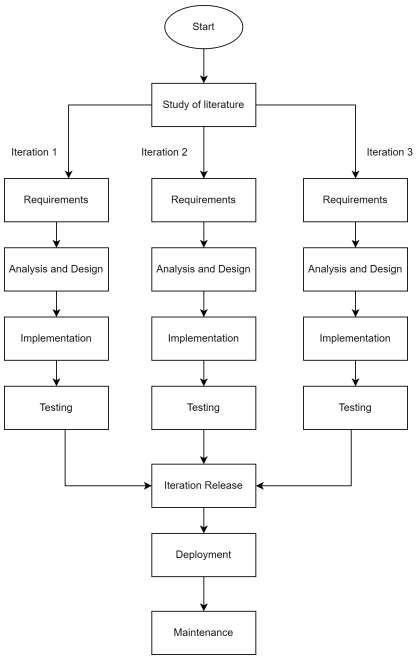

Figure 1: Iterative Incremental Method.

2. The elaboration stage,
   Is the second phase that focuses on analysis and design, publishes the basis of the project architecture, makes construction plans that support research objectives and preparing the process,
3. The construction stage,
   Is the third stage that focuses on developing software to produce a prototype or customized features,
4. The transition stage,
   The final stage focuses on application testing, including performance and functional testing, quality assurance and user testing to make the application ready for use and consumption by users. The process will restart from the beginning after the product requires more concrete development or updates.

If grouped according to the stages of development, the Iterative Incremental process in this study can be presented as shown in Table 1.

There are five proven advantages of iterative incremental over other methods, respectively (Bhuvaneswari and Prabaharan, 2013):

1. Iterative Incremental allows the developer to generate the software quickly during the software development life cycle (SDLC).
2. Iterative Incremental is more flexible and less costly to change scope and requirements if compared to another methods.

Table 1: Description of research activities.

Phase	Activities	Output
Inception	Literature Reviews	Background, Problems, and Research objectives
Elaboration	Analysis and design system	Use case, entity relationship diagram and Enpoint API
Construction	Implementation Testing	Program code Testing API
Transition	Release Deployment	The Rest API is ready to use Rest API installed on the server

3. It is easier to test and debug during a smaller iteration.
4. Customers or users of the application can respond to each build.
5. It easier to manage risk and failure because every part of the development is identified and handled during the iteration.

3.1 Use Case Diagram

Based on the case study, Show Up application consists of a pair user, namely external and internal users or commonly called admin users. External users have several functions, namely authentication, managing ads and transactions. While the admin can manage users and manage transactions. Figure 2 is a use case diagram describing the functional system to be created:

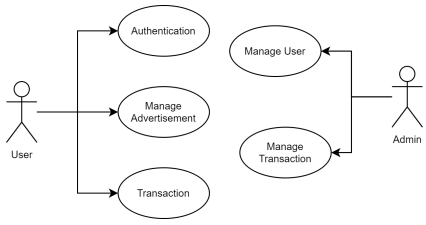

Figure 2: Show-Up Apps Use Case Diagram.

3.2 Entity Relational Diagram

An Entity Relational Diagram (ERD) aims to describe the data storage structure of applications used in making REST APIs (Susanto and Meiryani, 2019).

Figure 3 is an ERD which consists of several entities. User entities are used to store user data both internally and externally. Advertising entities are used to store advertising data generated by users. The advertisement photo entity is used to store advertisement photo datas. The bids entity is used to store advertisement bid data. The transferring entity is used to store advertising payment transfer data. The bank account entity is used to store account data. The transaction entity store advertising transaction data with any social media influencers.

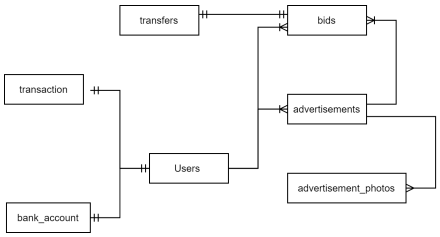

Figure 3: Show-Up Apps Entity Relational Diagram.

3.3 REST API Architecture

This architectural design aims to describe how the system works in communication between systems. Figure 4 is a system architecture diagram where the REST API Show Up acts as a liaison between systems. REST API Show Up will process data sent from a mobile or web application or commonly called a “Client”, then it will be processed on the server, then the server will return the data to the client(Wolde and Boltana, 2021).

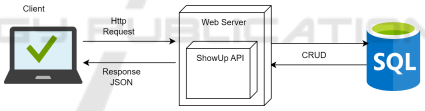

Figure 4: REST API Architecture.

4 RESULT AND DISCUSSION

4.1 Product Implementation (Show-Up Apps)

In the process of implementing the Show-Up Application, from production to post-development, several steps need to be taken, including the following activities, respectively:

1. Authentication

   This authentication feature is divided into 2, namely login and register. Login is used for authentication when you want to enter the application so that you can access the features in the Show Up application. This login process uses a jwt token which is useful for identifying users

who are logging into the application. Furthermore, the register is used for user registration so that later they can log into the application. Here is the authentication API url based to Table 2.

Table 2: API Authentication of Show-Up Application.

Method	Enpoint	Body
POST	/api/register	• name • email • password • password confirmation
POST	/api/login	• email • password

2. Manage User This feature is used to manage users such as viewing registered users and updating profile data. The following is the enpoint and body of the API on the Manage users feature. Following to Table 3 is the API of Manage User activity

Table 3: API Manage User.

Method	Enpoint	Body
GET	/api/users	-
GET	/api/users/me	-
PUT	/api/users/:id	• name • password • alamat • follower instagram • username instagram

3. Manage Advertisement

   This feature is used to manage ads such as creating, viewing, updating and deleting advertisement that have been managed by the user. In this feature, users can see a list of influencer offers. Following to Table 4 is the API of Manage Advertisement activity.

4. Transaction

   The transaction feature is used to carry out bargaining activities between advertisers and influencers. Then after making an offer, the ad owner must pay according to the amount of the price offered to the influencer. Following to Table 5 is the API of Transaction activity

4.2 Software Testing

The Software testing used in this research are divided into two part, respectively; Unit testing and Load testing. Unit testing is a test that is carried out by ensuring the functionality of the REST API that is made to run properly (Vegas et al., 2009). Load testing is a testing technique that tests the robustness of the system used (Jin et al., 2020). The following are the results of unit testing and load testing.

Table 4: Manage Advertisement.

Method	Enpoint	Body
POST	/api/advertisements	• title • description • photos
GET	/api/advertisements/:id	-
GET	/api/advertisments	-
PUT	/api/advertisments/:id	• title • description • photos
DELETE	/ api/advertisments/:id	-

Table 5: API Transaction.

Method	Enpoint	Body
POST	/api/transactions/bid	• advertisement id
		• price
POST	/api/transactions/bid/ transfer	• bid id • confirmation photo
POST	/api/admin/payment/ approve	• alamat
POST	/api/admin/payment/ reject	• bid id

1. Unit Testing

   Figure 6 describes the results of testing the REST API using the Postman application (Wolde and Boltana, 2021). In this testing, what is tested is the functionality of the system and the processing time to access each enpoint. Based on the Table 6, all enpoints can run well with good execution results, the fastest execution time is 176 milliseconds and the longest time is 344 milliseconds, with average 284 milliseconds:

2. Load Testing

   Table 7 is the result of load testing with server specifications of 1 GB RAM memory and 1 Core CPU. This test uses the loader.io application by making several requests at once in 1 minute. This server is only able to accommodate 1000 requests in 1 minute with a consequence of 55% error

Table 6: Result Testing API.

No	Enpoint	Time (ms)	status
1	POST: /api/register	344	success
2	POST: /api/login	245	success
3	GET: /api/users/me	349	success
4	POST: /api/advertisement	351	success
5	GET: /api/advertisement/:id	297	success
6	GET: /api/advertisement	300	success
7	PUT: /api/advertisement/:id	334	success
8	DELETE: /api/advertisement/:id	186	success
9	POST: /api/transactions/bid	420	success
10	POST: /api/transactions/bid/ transfer	233	success
11	POST: /api/admin/payment/ approve	174	success
12	POST: /api/admin/payment/reject	181	success
	AVERAGE	284	

while requests above 1000 servers will experience down.

Table 7: Result Load Testing.

Request Client	Response Time (ms)	Response success	Average Error Rate (%)	Status Server
250	236	240	4	Ok
500	253	475	5	Ok
800	257	774	6.25	Ok
1000	279	900	10	Ok
2000	302	900	55	Down

5 CONCLUSIONS

Based on the results of research that has been done, the conclusions obtained are as follows:

1. The research method used in this study is the Iterative Incremental method, as the system's business requirements are dynamic and easy to change. The Laravel framework and MySQL database were used for REST API implementation.
2. The application authentication is performed using the JSON Web Token as the identifier of the logged-in user. The features of the REST API in Show-Up applications consist of authentication, managing ads, managing transactions, and managing users.
3. The Black-Box testing was used to test the REST API, with two tests: unit testing and performance testing. The unit test used the Postman application to ensure the functionality was running properly; based on the test results, all endpoints ran as expected, with the fastest execution time of 176 milliseconds and the longest is 344 milliseconds. Performance testing was used to test the robustness of the system showing that the system could receive 1000 client requests in one minute and 2000 server requests were down.

REFERENCES

Barros, A., Ouyang, C., and Wei, F. Static analysis for improved modularity of procedural web application programming interfaces. *IEEE Access*, 8.

Bhuvaneswari, T. and Prabaharan, S. A survey on software development life cycle models. *International Journal of Computer Science and Mobile Computing*,2(May).

Brier, J. Json web token Best Current Practices. *Malaysian Palm Oil Council (MPOC*, 21(1).

Chatterjee, S. A comparative study on soap and restful web services. *International Research Journal of Engineering and Technology*, (May).

Garousi, V. and Rainer, A. and Lauvås, P. and Arcuri, A. Software-testing education: A systematic literature mapping, vol.165. *Journal of Systems and Software*, 2020.

Hidayat, M., Adityo, R., and Siswanto, A. Design of restaurant billing system (e bill resto) by applying synchronization of data billing in branch companies to main companies based on rest api. In *Proceeding - ICoSTA 2020: International Conference on Smart Technology and Applications: Empowering Industrial IoT by Implementing Green Technology for Sustainable Development*.

Hu, T. Seapp: A secure application management framework based on rest apin access control in sdn-enabled cloud environment. *J Parallel Distrib Comput*, 147.

Jin, W., Qian, J., and Yan, S. Shared-mode resource allocation for cloud-based load testing. *IEEE Access*, 8.

Jones, M., Bradley, J., and Sakimura, N. Rfc 7519: Json web token (jwt). In *Internet Engineering Task Force (IETF*.

Jánoky, L., Levendovszky, J., and Ekler, P. An analysis on the revoking mechanisms for json web tokens. *Int J Distrib Sens Netw*, 14(9).

Li, L., Chou, W., Zhou, W., and Luo, M. Design patterns and extensibility of rest api for networking applications. *IEEE Transactions on Network and Service Management*, 13(1).

Neumann, A., Laranjeiro, N., and Bernardino, J. An analysis of public rest web service apis. *IEEE Trans Serv Comput*, 14(4).

Nidhra, S. Black box and white box testing techniques - a

literature review. *International Journal of Embedded Systems and Applications*, 2(2).

Panduwiyasa, H., Puspitasari, W., and Saputra, M. Integrated invoice and accounting system design for food sector smes using openerp with quickstart methodology. In *Synergizing Management, Technology and Innovation in Generating Sustainable and Competitive Business Growth*.

Panduwiyasa, H., Saputra, M., Azzahra, Z., and Aniko, A. Accounting and smart system: Functional evaluation of iso/iec 25010:2011 quality model (a case study. *IOP Conf Ser Mater Sci Eng*, 1092(1):012065,.

Rosidin, R., Andreswari, R., and Alam, E. Building titipmasa.id application using iterative incremental method.

Susanto, A. and Meiryani. Database management system. *International Journal of Scientific and Technology Research*, 8(6).

Sutiah, S. and Supriyono, S. Software testing on e-learning madrasahs using blackbox testing. *IOP Conf Ser Mater Sci Eng*, 1073(1).

Vegas, S., Juristo, N., and Basili, V. Maturing software engineering knowledge through classifications: A case study on unit testing techniques. *IEEE Transactions on Software Engineering*, 35(4).

Wolde, B. and Boltana, A. Rest api composition for effectively testing the cloud. *Journal of Applied Research and Technology*, 19(6).

Zeebaree, M., Ismael, G., Nakshabandi, O., Sattar, S., and Aqel, M. Impact of innovation technology in enhancing organizational management. *Estudios de Economia Aplicada*, 38(4).