RESTful API Implementation in Making a Master Data Planogram Using the Flask Framework
(Case Study: PT Sumber Alfaria Trijaya, Tbk)

Era Susanti\(^1\), Evangs Mailoa\(^2\)
\(^1,2\)Kristen Satya Wacana University
\(^1\)672016309@student.uksw.edu, \(^2\)evangs.mailoa@uksw.edu

Received 24 March 2020; accepted 12 December 2020

Abstract. One of developing retail company and is one of the biggest retail companies in Indonesia, namely Alfamart which is owned by PT. Sumber Alfaria Trijaya, Tbk. Alfamart must have the best marketing strategy and increase innovation for the satisfaction of customers in order to survive in high business competition. One strategy to improve marketing is the arrangement of product displays in stores known as planograms. Planogram is a concept that is used in planning the arrangement and placement of products according to certain categories based on consumer spending habits that aim to increase sales at retail. This research was conducted to create a web-based planogram master application using the Flask framework with the python programming language. The method used in this study is the RESTful API, which is the implementation of web services that work through HTTP links. This research produces a web-based master data application that can be used by users in entering data needed in making a planogram.

Keywords: RESTful API, Python Flask, Planogram

1 Introduction

Technological developments that occur in a country can also affect developments in other fields such as economic growth. One aspect that supports the current economic development is the growing retail business that has sprung up. Retail is a business field that sells products and services that are added value to meet personal, family or group needs in retail quantities. Retail is the last link in the distribution process that connects producers and consumers. It is one of the economic indicators that contributes 19-31% to the Gross Regional Domestic Product (GRDP) [1].

One retail that is developing in Indonesia is Alfamart. Alfamart is a minimarket brand that provides daily needs owned by PT. Sumber Alfaria Trijaya, Tbk. Is one of the largest retail companies in Indonesia, which was founded in 1989 and started business in the field of trade and distribution such as consumer goods, then in 1999 entered the retail sector [2]. In facing high business competition, Alfamart must have a good and attractive marketing strategy and increase innovation for the satisfaction of customers [3]. One way to improve marketing strategies in retail is through the
arrangement of products in stores because it can increase customer buying interest. Implementation of product structuring and display in retail is a planogram.

This research was conducted to create a master data planogram in a web-based marketing application using the Flask framework with a lightweight programming language and is widely used today, namely python. The method used in this study is the RESTful API, which is the implementation of web services that work through HTTP links [10].

This research resulted in a web-based master data application that can be used by users, namely the marketing department in entering the data needed in making a planogram. The data entered include location type data, store type data, rack size, shelf position product classes, non-regular shelves, regional names and regional transfer plan. The results of this study can help facilitate application users, namely the marketing section in managing the data needed when making a planogram.

Based on the background that has been described, the identification of the problem in this research is the management of master planogram data is very necessary to assist in entering the data needed in implementing the planogram.

Based on the background and identification of the problems above, the formulation of the problem in this research is how to implement the RESTful API method in making master planogram data in Marketing Application System, how to use Flask framework in making master planogram data in Marketing Application System.

The purpose of this research is to create a web-based planogram master application at PT. Sumber Alfaria Trijaya, Tbk using the Flask framework and PostgreSQL database. The benefit of this research is to make a web-based master data application that is lighter and easier to use and makes it easier for users, namely the marketing department in managing the data needed when making a planogram.

2 Library Review

Research conducted by Haryanto et al entitled "The Effect of Laying Product Display Based on 5R Principles and Planograms on Increasing Sales: Case Study in Retail X in Lamongan" aims to identify strategies in structuring goods or products and increase sales in Retail X and improve good habits for employees in terms of product preparation. The results of this study are based on the 5R concept and planogram. In this study planogram data that were made are still using the manual or usual way using pareto charts [4].

Research conducted by Gilang entitled "Designing and Implementing Web Services as a Data Exchange Media in Game Applications" discusses the application of web services in exchanging information on different platforms between PHP websites and Android applications using Java. The modules created in this study are the register and login module, the points module and the pulse exchange module. Module testing in this study using postman. The results of this study are that web service can be implemented and can run well according to postman simulations [6].

Research conducted by Mukhammad et al entitled "Analysis and Design of Representational State Transfer (REST) Web Services for Academic Information Systems Integrated Nurul Fikri STT Using YII Framework" resulted in REST web services for academic information systems in Integrated Nurul Fikri STT which
provides services for other systems which requires and handles data exchange on different platforms. This research was made using the Yii Framework 2.0. System testing in this study was carried out using the blackbox method [7].

Research conducted by Haryanto et al explains about making planogram data that still uses manual methods, namely charts [4]. Research conducted by Gilang and Mukhammad et al uses a web service method that is implemented by providing services for other systems and exchanging data on different platforms. The difference of this research with previous studies is that in this study using a framework that is based on Python language Flask to implement web services in a web-based planogram data master application.

Planogram is a concept used in planning the arrangement and placement of products according to certain categories based on consumer shopping habits [4]. Planogram is made with the aim of maximizing retail sales. Made based on several inputs such as size of retail area, type of retail location, product size, size and position of the product such as shelves, cabinets and others.

REST (Representational State Transfer) is a web-based communication standard used in implementing web services. REST works through HTTP (Hypertext Transfer Protocol) links in carrying out certain tasks. HTTP functions that can be used in REST are get, post, put and delete functions [5]. It was first introduced by Roy Fielding in 2000. The web service system that applies the principles of REST is called RESTful. The way RESTful works is that the client sends a request via HTTP Request and the server responds to the client's request via HTTP Response.

There are 2 parts of messages to communicate with the server, namely the message header and message body. HTTP header is a small record of every transaction on HTTP while the HTTP body is the data to be sent [8].

Python is a high-level programming language that is often used by developers in making or developing applications today [9]. Supports object-oriented programming and is designed to be easy to learn and use.

Python is a dynamic programming language, which can be used in various types of software or application development [11]. In writing code there are no declarations of variable data types, parameters, functions or methods, which make python code lines shorter and more flexible [12].

Flask is a web framework written in Python. Flask has two external libraries namely the WSGI toolkit and the Jinja2 template engine [12]. Is a type of microframework that does not require a particular library in its use. Flask can use extensions to add features and components that have been provided by third parties and are not installed by default on Flask such as Form Validation, Upload Handling, Database [13].

PostgreSQL is an ORDBMS (Object-Relational Database Management System) ie models such as objects, classes and inheritance can be implemented directly in the database and query schema. The license owned by PostgreSQL is the GPL (General Public License) so that it can be used and modified freely without the need to pay a license. PostgreSQL was developed by Professor Michael Stonebraker with students and programmers at the University of California, Berkeley. The fundamental difference in PostgreSQL with other database systems is that PostgreSQL's ability to define SQL
functions independently [14]. PostgreSQL can run on types of operating systems such as Windows, Linux, UNIX, AIX, BSD, HP-UX, Mac OS X, Solaris and Tru64 [15].

<table>
<thead>
<tr>
<th>Code in C</th>
<th>Code in Python</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main()</code></td>
<td><code>main()</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>{</code></td>
</tr>
<tr>
<td><code>char ch;</code></td>
<td><code>char ch;</code></td>
</tr>
<tr>
<td><code>FILE *fin, *fout;</code></td>
<td><code>FILE *fin, *fout;</code></td>
</tr>
<tr>
<td><code>fin=fopen(&quot;a.txt&quot;,&quot;r&quot;);</code></td>
<td><code>fin=fopen(&quot;a.txt&quot;,&quot;r&quot;);</code></td>
</tr>
<tr>
<td><code>fout=fopen(&quot;b.txt&quot;,&quot;w&quot;);</code></td>
<td><code>fout=fopen(&quot;b.txt&quot;,&quot;w&quot;);</code></td>
</tr>
<tr>
<td><code>while(feof(fin))</code></td>
<td><code>while(feof(fin))</code></td>
</tr>
<tr>
<td><code>{</code></td>
<td><code>{</code></td>
</tr>
<tr>
<td><code>putc(ch,fout);</code></td>
<td><code>putc(ch,fout);</code></td>
</tr>
<tr>
<td><code>ch=getc(fin);</code></td>
<td><code>ch=getc(fin);</code></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td><code>fclose(fin);</code></td>
<td><code>fclose(fin);</code></td>
</tr>
<tr>
<td><code>fclose(fout);</code></td>
<td><code>fclose(fout);</code></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

Table 1. Comparison of lines of code in C and Python [10]

3 Research Methodology

There are several stages in this research so that the making of web-based applications in accordance with the expected results. In this study using the Waterfall SDLC (System Development Life Cycle) method. The steps in this research are as follows:

![Figure 1. Stages of Research Methodology](image-url)
The following is an explanation of the stages of research in Figure 1:

a. System Analysis

The system that will run in this study is that users can perform the CRUD function (Create, Read, Update, Delete) in processing data in this application. The function can be carried out on the modules available for making a planogram namely the Location Type, Store Type, Shelve Size, Product Class, Non Regular Shelves, Regional Names, Regional Transfer Plano and Planogram. Users can do the get, post, put and delete methods on each of these modules. At this stage, system requirements analysis is also used in this study.

b. System Design and Design

At this stage the system design will be done based on an analysis of system requirements that have been carried out. Then the system that has been designed and analyzed will be implemented using tools namely the python language with the Flask framework and PostgreSQL database. This stage is also the stage of making system modeling using UML (Unified Modeling Language). Making a system model produces Use Case Diagrams, Class Diagrams and Activity Diagrams. The following are the results of making designs in this study.

Figure 2. Use Case Diagram

Figure 2 is a Use Case Diagram in this study. In this diagram the employee acts as the actor who will run the application. In the picture employees can run operations on several modules. Before running these modules employees must log in first. The main function that can be performed by employees is on the Planogram, but before carrying out functions on the planogram the employee must perform operations on the master data modules. The master data modules that can be run by employees include managing location type data, managing store type data, managing shelve size data, managing product class data, managing non-regular rack data, managing shelve position data, managing regional name data and managing data regional transfer plano. In the master data module, employees can perform functions on each master data module such as viewing, adding, editing and deleting data. Then after performing operations on the
The master data module the employee can run the planogram module where the planogram data is obtained from the master data modules.

<table>
<thead>
<tr>
<th>User</th>
<th>Application</th>
<th>Web Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Chooses the Planogram Menu</td>
<td>Check Form Data Validation</td>
<td>Receive Request Method Response</td>
</tr>
<tr>
<td>User Fills in the Planogram Data Form</td>
<td>Sending Post Method Request Based on User Input</td>
<td>Sending Post Data Method Response</td>
</tr>
<tr>
<td>Getting Data Notification Not Appropriate</td>
<td>Valid?</td>
<td>Check Data Validation</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>Valid?</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Send Notification Failed to Save</td>
<td>Data Storage</td>
</tr>
<tr>
<td></td>
<td>Getting Notified Successfully Saved</td>
<td>Sending Data Storage Response</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receive Data Storage Responses and Send Notifications</td>
</tr>
</tbody>
</table>

Figure 3. Activity Diagram

Figure 3 is an Activity Diagram in this study, namely in the Planogram menu, the process of inserting data. The flow of this diagram is the first step the user chooses the Planogram menu then to insert the data the next step the user fills in the data form needed to insert the data. After the user fills out the form, the application will check the validation of the input form from the user if the input data is valid then the application will send a request for the method post based on the user input, if the input data is invalid then the user will get a notification if the data filled is not appropriate and the user must fill repeat the form until the data is declared valid. After the application sends a post method request, the web service will accept the request and send a response to the post method, then the web service will check the validation of the data to be inserted. If the data is invalid, the user will get a notification failing to save the data, if the data is valid then the web service will data storage and sending data storage responses. After the web service sends a response, the application will receive a data response and send
a notification, after which the user will get a successful save data notification.

c. Implementation

This stage is the stage of implementing the system design that has been made. Namely the creation of web-based applications using tools namely the python programming language, Flask framework and also using the Postgresql database. This stage is the most important process in this research methodology because the core of the system development is at this stage. Making the system based on the design and system design that has been made in the previous stage.

d. Testing

This stage is the testing phase of applications that have been made, whether in accordance with the initial design of the system and ensure all functions contained therein can run well. Testing is an important stage because before the application will be used by the user or the user must ensure that the application can be used properly and there are no errors or bugs in it. This stage also determines user satisfaction when using an application that has been created.

4 Results and Discussion

Based on the results of the system design that has been created and implemented using the python language and framework flask, the results of this study are web-based data planogram master applications. This system was built using the python programming language with flask framework and postgresql database, this system also uses the restful API method. The following are the results of this study.

![Form Login](image)

Figure 4 Login page

Figure 4 is the login page. At login, the user must enter a timesheet or user id and password. This login process is when the user enters the timesheet and press enter then if the data matches the database, the name and store columns automatically fill in the data according to the user's input, then the user enters the password and click the login button. If the data does not match the database an error handling will appear and the user must re-enter the correct data. If the login is successful then the user will enter the main web page.
Figure 5 Menu Page Type Location

Figure 5 is the location type master menu page. In this menu there are location types as well as location names. In this menu the user can perform data view, add data, edit data and delete data.

Figure 6 Planogram Home Page

Figure 6 is the main page of the planogram. In this planogram menu the user can perform the data view, add data, edit data and delete data functions. If the user wants to see the planogram data, the user must enter the store type data, rack number and shelve number on the form in this menu. If the data entered is in accordance with the database, the planogram data will appear according to the type of store, shelf number and shelve number. If the user wants to add a planogram data, the user must add master data because these planogram data are taken from the master data.

Figure 7 Planogram Posting Page
Figure 7 is the planogram post page contained in the planogram menu. The function of this planogram post is to copy data from the head_temporary and det_temporary tables to the head_t and det_t tables. So before the planogram is posted, the planogram data is stored in a temporary table. The user can choose the specific data to be posted by entering one type of store, but if the user wants to post all planogram data then select all type of store. Planogram data posted based on the type of store that has been inputted by the user.

Program Code 1 User Access Controller on Modules in the Application

```python
class Akses_User(Resource):
    def __init__(self):
        self.skema_aksesuser = SkemaAksesUser()

def get(self):
    id = request.form.get('id')
    menu = request.form.get('menu')
    print("Id adalah : ", id)
    print("Menu adalah : ", menu)
    hak_akses = Akses.query.filter_by(id=id, menu=menu).first()
    akses = {}
    if hak_akses:
        akses['insert_update'] = hak_akses.insert_update
        akses['delete'] = hak_akses.delete
    else:
        akses['insert_update'] = 'False'
        akses['delete'] = 'False'
    return akses
```

Program code 1 is a controller for the user access function that is to give certain access rights to the user based on data in the database when the user accesses the application. There are 2 categories of user permissions namely insert_update and delete. Insert_update access rights are user access given when adding and editing data in a module, if insert_update is true then the user has access to add and edit data while if insert_update is false then the user does not have access to add and edit data. Delete access rights are user access given when deleting data in a module, if delete is true then the user has access to delete data while if delete is false then the user does not have access to delete data.

Program Code 2 Controller Function Get Data on Module Type Location

```python
class Jenis_Lokasi(Resource):
    def __init__(self):
        self.skema_jenislokasi = SkemaJenisLokasi()
        self.skema_jenisLokasis = SkemaJenisLokasi(many=True)
        self.skema_detailokasis = SkemaDetailCabang(many=True)
        self.skema_aksesuser = SkemaAksesUser()

def get(self):
    jenis_lokasi = Lokasi.query.filter_by(kode_cabang='A000')
    hasil = skema_jenisLokasis.dump(jenis_lokasi)
    return jsonify(hasil)
```

Program code 2 is a controller function that gets data on module type location.
Program code 2 is a function to display data in the Location Type module. When displaying data, this module uses a schema and uses a dump to display the schema. After the dump process, the results are converted to json form with jsonify () code.

Program Code 3 Function Controller Get Position Data on the Planogram Module

```python
class getPss(Resource):
    def get(self):
        listpss = []
pss = PassShelve.query.all()
for a in pss:
    l = {}
l['id'] = a.kd_pss
l['value'] = a.kd_pss
listpss.append(l)
return jsonify(listpss)
```

Program code 3 functions to display position data on the Planogram module. In this function there is an empty list to hold the position data to be displayed. Then after querying the data the data is added to the list and the results are converted to json form.

System testing in this application is done using the black box testing method. Black box testing is a method of program testing that emphasizes the functional program being tested and checks the input and output data of the program [16]. The following are the results of tests that have been carried out by the black box method in this study.

Table 2. Testing on the Login Page

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Input</th>
<th>Expected Output</th>
<th>Output</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login</td>
<td>Username and Password not filled</td>
<td>Error message appears</td>
<td>The message &quot;Username dan Password harus diisi!&quot;</td>
<td>Valid</td>
</tr>
<tr>
<td>2</td>
<td>Login</td>
<td>Username is filled in with incorrect data</td>
<td>Error message appears</td>
<td>The message &quot;Data tidak valid!&quot;</td>
<td>Valid</td>
</tr>
<tr>
<td>3</td>
<td>Login</td>
<td>Password filled with incorrect data</td>
<td>Error message appears</td>
<td>The message &quot;Username atau password salah!&quot;</td>
<td>Valid</td>
</tr>
<tr>
<td>4</td>
<td>Login</td>
<td>Username and Password filled in the appropriate data</td>
<td>Login successful</td>
<td>Successfully logged in and entered the main page</td>
<td>Valid</td>
</tr>
</tbody>
</table>
Table 3. Testing on the Data Type Master Page Location

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Input</th>
<th>Expected output</th>
<th>Output</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Insert</td>
<td>One or all of the fields on the form are not filled in</td>
<td>Error message appears</td>
<td>The message &quot;Data Kosong!&quot; Appears</td>
<td>Valid</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Fill out the form with location type data that already exists in the database</td>
<td>A duplicate data message appears</td>
<td>The message &quot;Data Jenis Lokasi sudah ada!&quot;</td>
<td>Valid</td>
</tr>
<tr>
<td>3</td>
<td>Insert</td>
<td>Form filled with data in accordance with the provisions</td>
<td>Successfully added data</td>
<td>The message &quot;Data berhasil ditambahkan!&quot; and data added successfully</td>
<td>Valid</td>
</tr>
<tr>
<td>4</td>
<td>Update</td>
<td>The Location Name field not filled in</td>
<td>Error message appears</td>
<td>The message &quot;Nama Lokasi Kosong!&quot;</td>
<td>Valid</td>
</tr>
<tr>
<td>5</td>
<td>Update</td>
<td>Data to be filled in accordance with the provisions</td>
<td>Successfully edited data</td>
<td>The message &quot;Data berhasil diupdate!&quot; and the data was edited successfully</td>
<td>Valid</td>
</tr>
</tbody>
</table>

Table 4. Testing on the Planogram Page

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Input</th>
<th>Expected Output</th>
<th>Output</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>View Data</td>
<td>One field not filled</td>
<td>Error message appears</td>
<td>The statement &quot;Tidak ada data&quot; appears</td>
<td>Valid</td>
</tr>
<tr>
<td>2</td>
<td>View Data</td>
<td>The data that is filled out does not match the database</td>
<td>Error message appears</td>
<td>The statement &quot;Tidak ada data&quot; appears</td>
<td>Valid</td>
</tr>
<tr>
<td>3</td>
<td>View Data</td>
<td>Data is filled in accordance with the database</td>
<td>Successfully displayed data</td>
<td>Data successfully displayed in accordance</td>
<td>Valid</td>
</tr>
</tbody>
</table>
The following is a display of tests conducted using the Black Box method.

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Input</th>
<th>Expected Output</th>
<th>Output</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Insert</td>
<td>Data is filled in accordance with the provisions</td>
<td>Successfully added data</td>
<td>The message &quot;Data berhasil disimpan!&quot; and data added successfully</td>
<td>Valid</td>
</tr>
</tbody>
</table>

Figure 8 Display Login Test

Figure 8 is a test display on the login page when the password data is filled with incorrect data. The output of the test is the message appears as in the picture above and the output is as expected.

Figure 9 Display of the Master Test Location Type

Figure 9 is a test display on the location type master data page on the data insert function when the location type data is filled with data that already exists in the database. The output of the test is the message appears as in the picture above and the output is as expected.
Figure 10 Display of the Master Test Location Type

Figure 10 is a test display on the location data master page on the data update function when the data entered is in accordance with the provisions. The output of the test is the message appears as in the picture above and the output is as expected.

Figure 11 Display Planogram Testing

Figure 11 is a test display on the planogram page in the data view function when the data that is filled is in accordance with the database. The output of the test is data that appears in accordance with the input and outputs as expected.
Figure 12 is a test display on the planogram page of the data insert function when the data that is filled is in accordance with the database. The output of the test is data that appears in accordance with the input and outputs as expected.

5 Conclusions

The results of this study are web based planogram master application. The method used in this study is the RESTful API, which is the implementation of web services that work through HTTP links. The language used in this application is a language that is being widely used at this time, namely python using the Flask framework. This application aims to make it easy for users to enter master data that will be used when making a planogram.

Suggestions from the authors for developing this application are:
- The use of Python Flask can be further developed and can be added by JSON Web Token to protect the API from being easily accessed by unauthorized people
- Maybe in the future this application can be developed based on mobile and made more dynamic and efficient with the mobile version.

References


Era & Evangs., RESTful API Implementation in Making a Master Data ...


