ISSN : 1978-8282, Online ISSN: 2655-4275

LARAST Apps Administration Public Service Using Restful Api Method and MVVM Architecture

Fifit Alfiah[*1], Herfandi[2], Aditya Agus Wisanto[3]
[1] Informatics Engineering, Faculty of Science and Technology, University of Raharja, Indonesia
[2] Systems Engineering, Sumbawa University of Technology, Indonesia
[3] Magister Computer Science, Budi Luhur University, Indonesia
E-mail: [*1]**fifitalfiah@raharja.info**, [2]herfandi@uts.ac.id, [3]2311600015@student.budiluhur.ac.id

Abstract

Technological developments have made it easier for users both in terms of speed and accuracy in the transition to accessing an application. This convenience in particular gives rise to an idea and idea to be able to create an administrative service application that is used by the community around the Pasar Kemis sub-district. In the administrative service process, they still use queue numbers and for the filing process, they still use duplicated files by photocopying. The duplication resulted in an archive field at the Pasar Kemis District office. The submission of the completed application file does not have the correct information for file retrieval, so valid information is needed regarding the completion of the application file. This application uses the Restful API in the Laravel framework and implements its security using the Laravel Passport as a JSON access token key. Larast Apps uses the MVVM architecture. For HTTP Client Server using Retrofit 2 and OkHttp and Kotlin Coroutines. With this application, it can simplify the process of submitting public administration services. Submitting an application from home through the Android application can avoid crowds at the Pasar Kemis District Office and cut down on the bureaucracy that is already running properly.

Keywords *— MVVM, Android Rest Api, Retrofit 2, Laravel Passport Token*

1. INTRODUCTION

Administrative services are used as a form of service in a government or private agency. Administrative services are a series of activities for structuring and regulating documents in population data through population registration, civil registration, and administrative information management carried out by an agency in the sub-district[1]. The biggest challenge faced by a government agency, especially at the sub-district level in the Pasar Kemis area, is having limited access to administrative services whose citizens still come to the sub-district office directly, this will result in long queues at the Pasar Kemis sub-district office. In addition, another problem is of course the problem in filing requirements where the filing still uses paper duplicated by photocopying will certainly cause an archive field at the Pasar Kemis.

The Pasar Kemis sub-district office has problems other than service and filing, namely providing information to residents and activities that are still massively known to residents[2]. Activities that are still massive in delivering information are usually held by Integrated

Healthcare Center, free dental checks and others as well as information on taking application files that have finished the process.

The purpose of making this administrative service application is to simplify and build a system that can help the community get services at the Pasar Kemis District office faster and more precisely and change the current system, namely manual to computerized so as to improve the quality of performance efficiency of officers and residents who use the application.

The proposed program design from the interview results has been carried out and described in the form of UML (*Unified Modelling Language*) diagrams, namely *Activity* Diagrams, *Sequence Diagrams* and one of them is in the form of *Use Case* Diagrams displayed in this study[3]. The implementation of the administrative service application design using the *Kotlin* programming language with the application of [4] MVVM (*Model View View Model*) writing and data interaction between *clients* using *the RESTful API*[5] is made using *the Laravel Framework* as a bridge container for interaction between *clients* and PHP (*Hypertext Preprocessor) programming language*). To be able to interact between data and *clients* in JSON data format on the *Android mobile* side using the *Retrofit 2 library* that has been provided and developed by *Jake Wharton*[6]. The last stage will be testing endpoints that have been created from an *API* using the *Postman application*[7].

Web service is a JSON-formatted application data interaction service to distribute services over the internet that supports system interoperability. Web service architectures include *Extensible Markup Language Remote Procedure Call* (*XML-RPC*), *Simple Object Access Protocol* (*SOAP),* and *Representation State Transfer* (*REST)*. This research implements the REST architectural style in API development as a *backend* in administrative service applications in Pasar Kemis District[8].

2. RESEARCH METHOD

The method used in the implementation of this administrative service application is using the SDLC (*Software Development Life Cycle*) method. SDLC has a description in it, namely *Extreme Programming* which includes: *Planning*, *Design*, *Coding*, and *Testing*. The use of this method aims to improve the analysis of the functional needs of the *project* to be made. The first thing that will be done is to make *a plan*, namely by making *use of case diagrams* and *class diagrams* as a form of reference for the flow of the running system and the data format to be used. Furthermore, *design* as a *prototype* application that will be implemented, then there is *coding* and *testing* as the final part of the implementation related to community administration service applications that include *Restful API* and its security, *the Framework* that will be used, and the implementation of *the coding* structure, namely MVVM[9].

Secure *the RESTful API* [10]used in administrative service applications using *Laravel Passport*. This feature is adopted from *a module* called *League Oauth2 Server* developed by *Alex Bilbie* on his *Github* account. *Laravel Passport* provides a full implementation of *Oauth2 Server* for the development of software built using the *Laravel framework*.

Retrofit 2 is a *secure HTTP* client library for *Android applications*[11], especially *native Android using* Java *and* Kotlin *programming languages*. *Retrofit 2* itself is used as a *REST Client* on the *mobile* side. Using *Retrofit 2* there is no need to build your methods to connect to *the web service*. *Retrofit 2* simplifies the interaction process of exchanging data from *JSON* or XML format by being parsed into *Plain Old Java Objects* (*POJOs*). *Retrofit 2* itself has a request method[12] on the *server* in the form of *POST*, *GET*, *PUT*, *DELETE*, and *PATCH methods*[13]. *Retrofit 2* is built from several libraries that can be used as *OkHttp third-party* protocol to get *a response* from *the client* who made the request[14].

Based on the results of observations and interviews, it will be formulated and the elaboration of important points in it in the form of Table 1 images for a feature and scope of application use. For the next stage, the process of creating *a RESTful API* using *the Laravel Framework* continued in the implementation stage of the MVVM pattern design[15] in the administration service application. The last stage is testing, testing on *the API endpoint* that has been created.

Table 1 explains the features and scope of the administrative service application based on observations and interviews as well as agreements from the Pasar Kemis. The features of the application based on Table 1 consist of 2 types of users with different application bases, namely the use of Android-based applications on the part of residents and website-based applications from the Pasar Kemis District office as the *admin base* to manage related data about administrative services.

Table 1. Feature Formulation and Application Scope

No	Android and Web Features	Warga	Kecamatan
1	Login	✓	✓
2	Register	X	✓
3	View News	✓	✓
4	News Input	X	✓
5	Search News	✓	✓
6	View Event	✓	✓
7	Input an Event	X	✓
8	Get Notifications	✓	✓
9	Push Notification Input	X	✓
10	Announcement Input	X	✓
11	View Family	✓	X
12	Add Family	X	✓
13	View File Tracking	✓	✓
14	Search Files	✓	✓
15	Create a Cover Letter	X	✓
16	Download Cover Letter	✓	✓
17	Account Profile	✓	✓
18	File Access Approval	X	✓
19	Village Access Granting	X	✓
20	File Validation	X	✓
21	Account Addition	X	✓
22	Upload Requirements File	✓	X

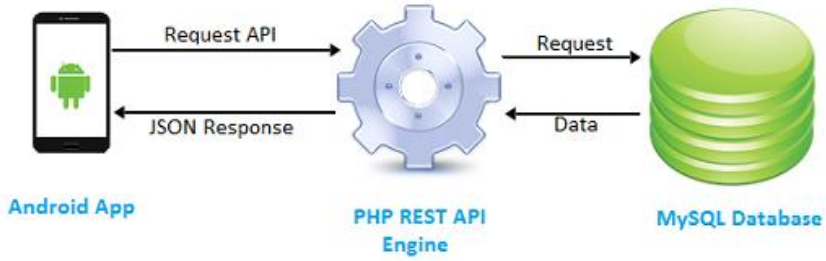

Figure 1. *Client* and *Server RESTful API Architecture*

The administration service application created will be focused on an *Android* base and response that integrates with the *website* using *RESTful API*[10]. The *client* and *server* architecture will be formed as shown in Figure 1, *the client* will interact with each other through *the REST API Engine* with the *database* so that it will have the same aligned data as on the *Pasar Kemis sub-district admin website using the scope in Table 1*. *The REST API Engine* will respond according to *client requests* through *the endpoints that have been provided so that it can communicate to the* server *side* with data in the form of *JSON Response* that has been processed.

The design of *a use case diagram* is a scheme used to describe all activities that can be carried out by actors (*users*). Activities that can be done by residents include *logging in*, viewing news and events, making a cover letter for submitting an application, viewing and *downloading* a cover letter when residents have submitted a cover letter, viewing the process of citizen application files that have been submitted, changing *profile* accounts, receiving file notifications when the file has been processed, viewing and *uploading* File requirements.

Activities that can be done by admins include *login*, inputting news, and events, validation of cover letters made by residents, file validation, and changing the filing status during the process of files submitted by residents who have passed through bureaucracy based on the system created, uploading cover letters when residents make filing requests, changing *profile* accounts, provide filing notifications to residents who have accounts, create citizen accounts and input family members. The depiction of the *designed Use Case Diagram* can be seen in Figure 2.

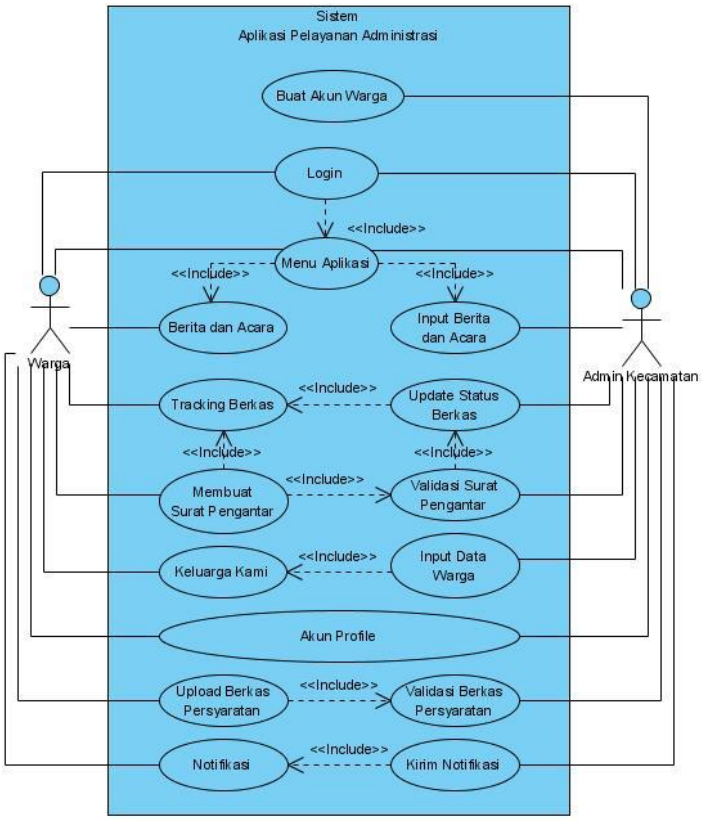

Figure 2. *Use Case* Diagram

DBMS (*Database Management System*) is a software to design and build a database to hold all data connected by the system in a computerized manner. DBMS has a role as an intermediary *between users* and databases, DBMS has a method that adds, subtracts, modifies and deletes and manages the amount of data contained in a database. The database design of the system that has been created is an administration service application using *Visual Paradigm* to describe the database of relationships between tables that have been equipped with *primary* keys and *foreign keys* and each *interrelated field* with added a detailed data type and action such as *Create, Read, Update, Delete*, and *SEARCH*. The action has a function activity in each of the tables listed.

3. RESEARCH RESULTS AND DISCUSSION

The implementation of the design described in the previous section is done by creating *Restful APIs* and Android applications by applying MVVM (*Model View View Model) pattern design coding*. The discussion carried out was an explanation of the application of MVVM in Android applications in the form of *class diagrams* as well as the display of community administration service applications and the display of *Restful API* testing. The first stage is the display of *backend* methods of the *login* process created by applying the *MVVM* pattern *design (Model View View Model*) as a reference for implementing other views related to the architecture. The application of MVVM architecture is shown in Figure 3.

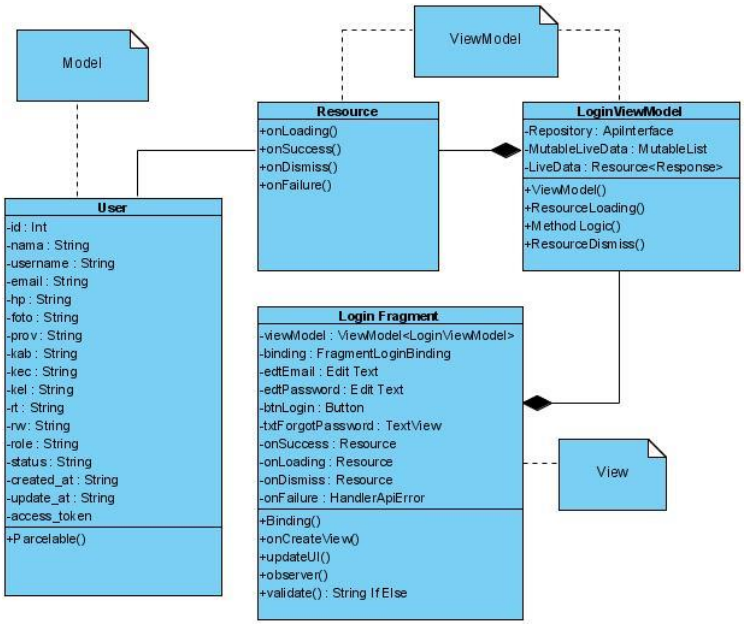

Figure 3. *Class Diagram* Desain *Pattern* MVVM

Next, display a *login display* along with the main page (after successfully *logging in*) in the administration service application. In the *login* display, you can only input your *email* and *password* as well as the forgot *password feature* that is specifically for residents using the application as a prevention of forgetting passwords that have been given by RT / RW from the District admin.

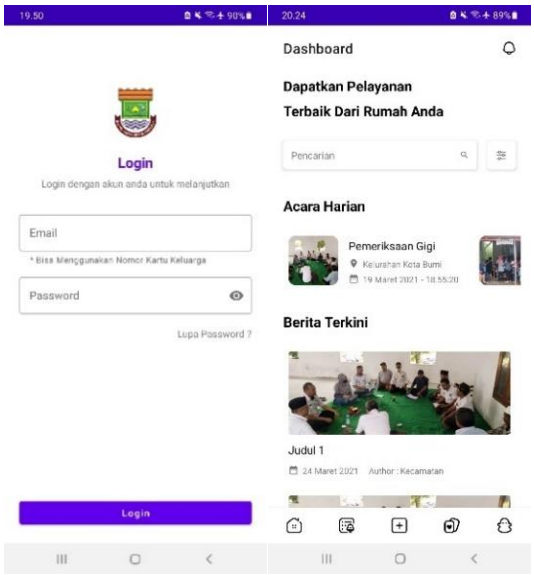

Figure 4. Login Screen Display

A login screen display in Figure 4 is a display that appears after the *splash screen* the first time the application runs, the display looks like it does not have *registered* access because it does not have the authority to use the feature at the scope of citizens. The use of account register feature can only be used on *website* applications owned by the District admin. If you have successfully *logged in*, it will be redirected to the *dashboard screen*.

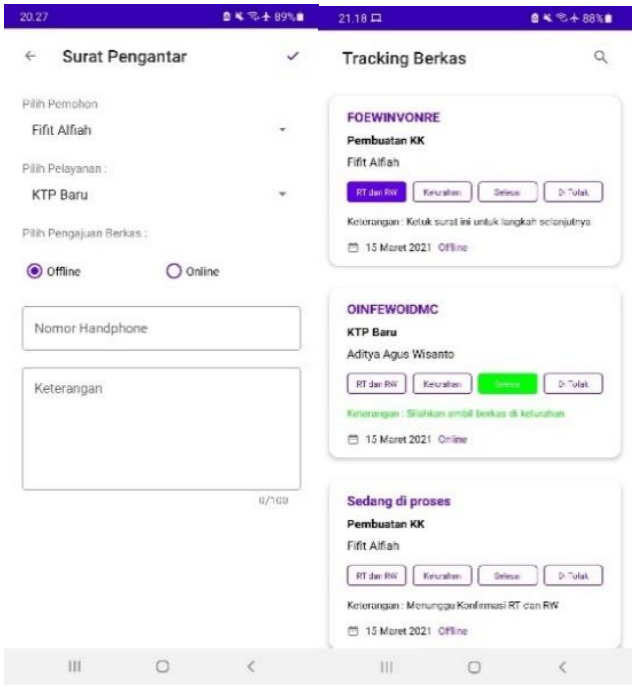

Figure 5. Cover Letter Menu Display and *File* Tracking

Figure 5 displays a *form* for creating a cover letter. Selecting the applicant to make a cover letter will create a *filter* method from the *logged-in* account to find out their respective family members so that it will not cause a *human error* related to the selection of applicants in submitting a cover letter. In the cover letter display, there are two types of file submissions, each of which has a different process for the next level such as *online* and *offline file* submission where online file submission can only be accessed through the *Android* application by uploading the required files, while offline file submission can only be accessed independently, namely coming to subdistrict for further processing. The screen display of the advanced application file tracking menu from the process of making a cover letter. The *file tracking* process functions as a monitoring of the running of a file process in the sub-district through structured information from administrative service applications. Files that have been submitted by residents have a change in status to the filing process at each subsequent bureaucratic step. The filing can be used to find out the application file process that runs in Pasar Kemis. In addition, *tracking* files that have an *offline* or *online* status indicate that the file is processed as it should have been selected in the creation of the cover letter. The status has a different function, if the submission file is *offline* then the button display will change to *download* a cover letter, then vice versa if *online* the *button* button will be directed to access the file through our family menu as shown in Figure 6.

Figure 6. *File* Tracking Menu Display

Each family member has one file *upload* view to support online filing needs. In addition, family member data is equipped with files that will support the online filing process that has been provided. The file *upload* display serves as a place for online file archives that will be used as a medium for application requirements at the Pasar Kemis sub-district office as shown in Figure 7.

Figure 7. Online File View

The last stage is *testing the RESTful API endpoint* that has been created to support the needs of administrative service applications using the *Postman* application for testing and testing *the RESTful API* starting from *the login*, retrieving *the API* that has the *GET* method, and storing the *login* token in the *Header* when each time it performs *requests* that have the scope of *Auth* security on *routing* in the *Laravel framework*. The type of JSON data creation has 2 types, namely JSON *Array* and *JSON Object*. JSON *Array* functions as retrieval of more than one data while JSON *Object* signifies data retrieval of only one of many. Usually JSON *Object* as a parameter for retrieving variables in a JSON Array. The first stage will be testing *endpoints* which will be shown in Figure 8.

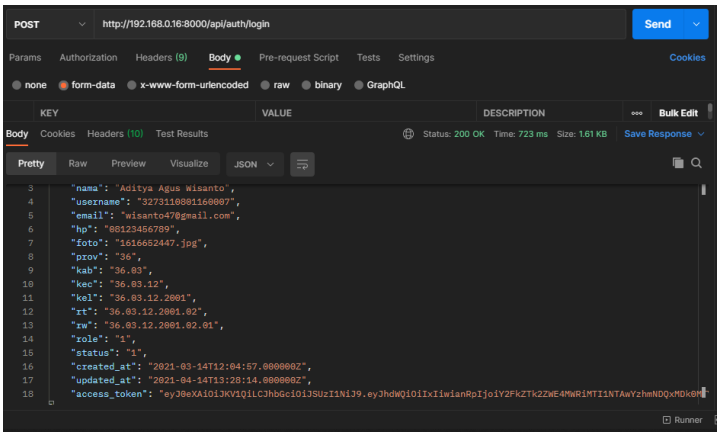

Figure 8. Login *Endpoint View*

You can see *the JSON* Object *variable* in Figure 8, which is *access_token*, *access_token* as a substitute for the *email key* and *password* when you want to *make a request* to the *server* as needed. The character length of *the access_token* is 70 characters with the data type used is a string. *Access_token* will always change when logging *in* to the application.

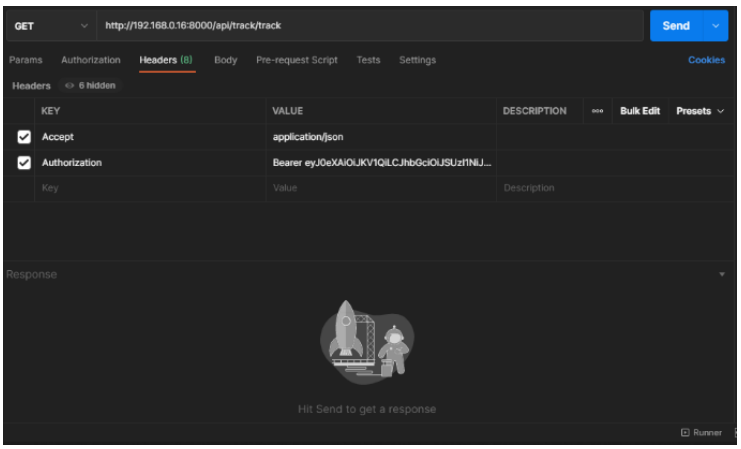

Figure 9. Display *Header Authorization*

Figure 9 shows the token storage in *the Headers* that will be used to make *requests* that are still the scope of *Auth* in *routing* in *the Laravel framework.*

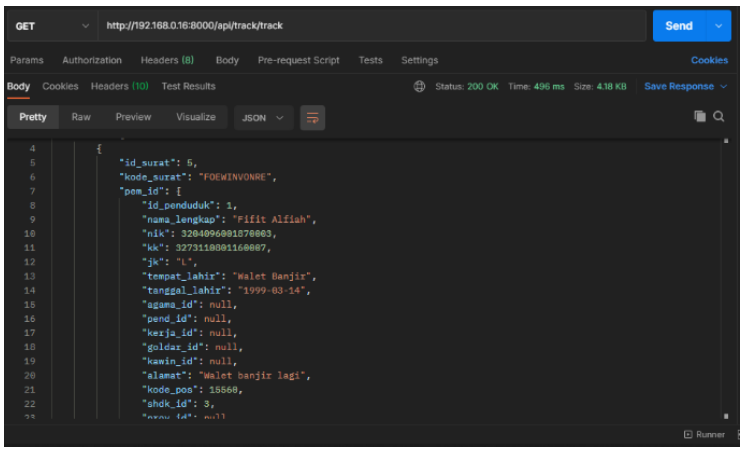

Figure 10. Get Data View Succeeded

Successful retrieval *of JSON* data using *the endpoint tracking* file in Figure 10 using a token stored from the login response results using *the Authorization Bearer header* with *the scope of the routing endpoint* in Figure 11.

```
Route::namespace('Api')->group(function () {
    Route::prefix('auth')->group(function () {
        Route::post('register', 'AuthController@register');
        Route::post('login', 'AuthController@login');
        Route::post('forgot', 'AuthController@forgot');
    });
    Route::group(['prefix' => 'family', 'middleware' => 'auth:api'], function () {
        Route::get('family', 'FamilyController@family');
    });
    Route::group(['prefix' => 'file', 'middleware' => 'auth:api'], function () {···
    });
    Route::group(['prefix' => 'letter', 'middleware' => 'auth:api'], function () {
        Route::get('citizens', 'LetterController@citizens');
        Route::get('service', 'LetterController@service');
        Route::post('letter', 'LetterController@letter');
    });
    Route::group(['prefix' => 'track', 'middleware' => 'auth:api'], function () {
        Route::get('track', 'TrackController@track');
        Route::get('searchtrack', 'TrackController@searchtrack');
    });
    Route::group(['prefix' => 'news', 'middleware' => 'auth:api'], function () {
        Route::get('news', 'NewsController@news');
        Route::get('daily', 'NewsController@daily');
        Route::get('searchnews', 'NewsController@searchnews');
```

Figure 11. *Laravel Routing Framework View*

The implementation of Android applications has a *pattern* of writing the code used by developers. *The* writing pattern in the administration service application applies the *MVVM (*Model View ViewModel*)* pattern. *The pattern* has a clean writing style and utilizes the features of *LiveData architecture components*, *Data Binding*, *ViewModel*, *Room*, and *Navigation Architecture Components* contained in *fragments*.

MVVM (*Model View View Model*) itself is used to separate *User Interface* and *Application Logic* [8], the basic concept of MVVM lies in *view* and *model*. *The View* will always observe data changes made to the *logic* contained in the *View Model* while the data management in the *View Model* will reference according to the variables stored in the *Model*. So, the *View Model* has a role in the *View* and *Model* when the data has a *request* from the *user logic* [9].

Architectures other than MVVM (*Model View View Model*) in the implementation of administrative service applications are *Room architecture* commonly referred to as DAO (*Database Access Object*). The use of DAO in the implementation of Android applications as *an offline database*. By using DAO, *developers* do not establish a direct connection to the *database*. DAO is very suitable for use with MVVM pattern design because it has the same intermediate layer, namely *Controller* and *Model* that will interact with View Model and *Model* [10].

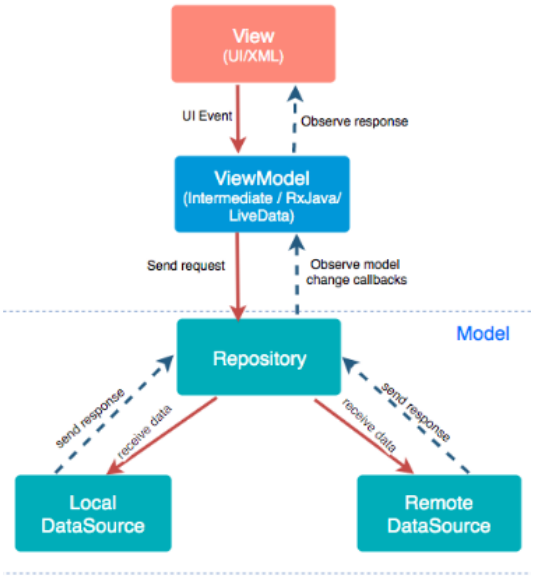

Figure 12. *MVVM* Pattern Design Diagram

In addition to testing using *software*, the administrative service application also conducts trial use to areas that are still covered by Pasar Kemis District, namely the Kuta Bumi area. App trials use the time parameter per week, they are tested within 2 months. Sampling in the Kuta Bumi area is because the community is very consumptive in the use of *smartphones* for a long time in daily use and understanding of technology above the average area in Pasar Kemis.

Table 2. Consumptive Use of *Smartphones*

No	Neighborhoods	Time	Use of hp
1	Gelam Jaya	5 – 8 hours/day	70% of People
2	Kuta Baru	9 – 12 hours/day	70% of People
3	Kuta Jaya	6 – 7 hours/day	65% of People
4	Kuta Bumi	11 – 14 hours/day	85% of People
5	Pangadegan	8 – 9 hours/day	60% of People
6	Pasar Kemis	10 – 12 hours/day	75% of People
7	Sindang Sari	9 – 12 hours/day	50% of People
8	Suka Asih	9 – 12 hours/day	50% of People
9	Suka Mantri	5 – 8 hours/day	65% of People

Table 2 describes the period and use of *smartphones* in each region using a sampling method with a large population in each region. The data in Table 2 is taken from the sub-district with data taken in the December 2020 report using a special survey at the village level. Seen in the data in Table 2 produces data as a form of information in the sample stage of testing Android-based public administration service applications. The results of the data information in Table 2 can obtain a summary of questionnaires related to interviews with residents in testing the application of community administration services.

Table 3. Questionnaire Summary Results

No	Question	Score	Answer
1	How about file-based administration services uploaded online anytime and anywhere without having to come to the sub-district?	80	70% of People
2	Get notified when a file is complete	100	100% Community
3	Can files be seen from the file tracking feature?	90	90% of People
4	How long does it take to get an account to access the application?	65	75% of People

From the results of a simple questionnaire distributed by the subdistrict using the *form*, it has not been maximized in the account field to access the application, because residents are still waiting for further processes on the sub-district in creating an account.

4. CONCLUSION

This administrative service application aims to facilitate an application submission activity in Pasar Kemis District using an Android *mobile-based smartphone*. The application of *the MVVM* (Model View View Model) *pattern* design in the implementation of administrative service applications makes it easier for *developers* who want to develop by adding certain features and more easily debugging in administrative service applications.

On the other hand, data security assisted by using *Laravel Passport* based on *OAuth2* can tighten the security of data on the *server*. Application testing is carried out by *testing* accompanied by a *sampling* method as a trial in certain regions and filling out questionnaires in the intended area, where testing is carried out on features, ease of use of the application, and efficiency of time used by the application.

5. REFERENCES

[1] E. Junirianto and D. S. Wita, "Pengembangan Aplikasi Antrian Online MAL Pelayanan Publik Samarinda," *Informatika Mulawarman: Jurnal Ilmiah Ilmu Komputer*, vol. 15, no. 2, p. 127, Sep. 2020, doi: 10.30872/jim.v15i2.3117.

[2] N. Dwi Tsoraya *et al.*, "Pengenalan Aplikasi Pelayanan Publik Digital 'Tangerang Gemilang", Journal of Community Service and Engagement (JOCOSAE), Vol. 3 No. 1, February 2023, pp. 40-50.

[3] S. E. R. Putri Gunawan and D. Hertati, "Inovasi Pelayanan Pengaduan Masyarakat Melalui Aplikasi Wargaku Berbasis Android di Dinas Komunikasi dan Informatika Kota Surabaya," *Jurnal Ilmiah Universitas Batanghari Jambi*, vol. 22, no. 3, p. 1360, Oct. 2022, doi: 10.33087/jiubj.v22i3.2462.

[4] B. Wisnuadhi, G. Munawar, and U. Wahyu, “Performance Comparison of Native Android Application on MVP and MVVM,” Advances in Engineering Research, volume 198, International Seminar of Science and Applied Technology (ISSAT 2020).

[5] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, “RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions,” *Applied Sciences (Switzerland)*, vol. 12, no. 9. MDPI, May 01, 2022. doi: 10.3390/app12094369.

[6] S. Chakraborty and P. S. Aithal, “MVVM Demonstration Using C# WPF,” *Google Scholar Citation: IJAEML International Journal of Applied Engineering and Management Letters (IJAEML) A Refereed International Journal of Srinivas University*, vol. 7, no. 1, pp. 2581–7000, 2023, doi: 10.5281/zenodo.7538711.

[7] H. A. Epiloksa, D. S. Kusumo, and M. Adrian, “Effect Of MVVM Architecture Pattern on Android Based Application Performance,” *Jurnal Media Informatika Budidarma*, vol. 6, no. 4, p. 1949, Oct. 2022, doi: 10.30865/mib.v6i4.4545.

[8] I. O. Suzanti, N. Fitriani, A. Jauhari, and A. Khozaimi, “REST API Implementation on Android Based Monitoring Application,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jul. 2020. doi: 10.1088/1742-6596/1569/2/022088.

[9] D. Indrawan, D. S. Kusumo, and S. Y. Puspitasari, “Analysis Of The Implementation Of MVVM Architecture Pattern On Performance Of Ios Mobile-Based Applications,” *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 8, no. 1, pp. 59–65, Feb. 2023, doi: 10.29100/jipi.v8i1.3293.

[10] S. Karlsson, A. Causevic, and D. Sundmark, “QuickREST: Property-based Test Generation of OpenAPI-Described RESTful APIs,” in *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 131–141. doi: 10.1109/ICST46399.2020.00023.

[11] D. S. Maulana, N. Ramadhani, "Aplikasi E-Pelayanan Publik Pada Sistem Administras Kantor Kelurahan Karang Anyer Kabupaten Asahan", Journal of Science and Social Research, Oct 2022, V (3): 709 – 714, Available online at http://jurnal.goretanpena.com/index.php/JSSR.

[12] S. Widyaningtyas, T. Wahyono, "Implementasi REST API Menggunakan Retrofit Pada Aplikasi Monitoring Grooming Berbasis Android", IT-EXPLORE, Jurnal Penerapan Teknologi Informasi dan Komunikasi, Volume 03 Nomor 02 Tahun 2024.

[13] I. K. D. Senapartha, “Implementasi Single Sign-On Menggunakan Google Identity, REST dan OAuth 2.0 Berbasis Scrum,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 7, no. 2, Aug. 2021, doi: 10.28932/jutisi.v7i2.3437.

[14] W. Sheikh and N. Sheikh, “A Model-View-ViewModel (MVVM) Application Framework for Hearing Impairment Diagnosis-Class Inheritance Architecture,” in *2020 Intermountain Engineering, Technology and Computing, IETC 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/IETC47856.2020.9249215.

[15] P. I. Sari, K. Nashirin, M. Arifudin, Y. Setiawan, and B. O. Learning, “Android Mobile Application System For Pet Care Services Using MVVM Architecture,” Indonesian Journal of Multidisciplinary Science, 2(11) August 2023.