Cihan University-Erbil Scientific Journal (CUESJ)

REVIEW ARTICLE

A Proposed Model for Building a Secure Restful API to Connect Between Server-side and Mobile Applications Using Laravel Framework with Flutter Toolkits

Osama M. A. AL-Atraqchi

Department of Computer Science, College of Science, Chian University-Erbil, Kurdistan Region, Iraq

ABSTRACT

These days, any business needs a mobile application that connects to a central database, such as those used for e-commerce, e-learning, and restaurant applications building a professional application that connects to the backend requires a Restful Application Programming Interface (API); as a result, picking the best languages, tools, and frameworks to build a mobile application on the client side, as well as a database, API, and dashboard on the back end, is essential and result in a problem. This paper proposes a paradigm for building cross-platform mobile apps using the flutter framework. It includes packages such as HTTP for connecting to APIs, flutter secure storage for storing tokens, GetX for state management, and using MVC architectural style. The result is building a full-stack application (frontend and Backend) called (My Services), a kind of eCommerce application for services only. In addition, the Laravel framework and MySQL database are used for backends such as dashboard and API., and Spatie package is used with Laravel to make a system for rules and permissions, as well as uses Sanctum package for authentication and authorization system.

Keywords: Flutter, laravel, web development, mobile development, backend-frontend.

INTRODUCTION

These days, mobile applications are essential and widely used, and they also provide the best environment for using sensors for Internet of Things (IoT) systems, which are also widely used.[1] However, since we cannot create any application without a central data source to enable user sharing of data and resources, the computability between systems, especially client and server environments, becomes a problem.[2]

Using Application Programming Interface (API) is the best solution to communicate between data sources on the server side and the applications on the client side.[3]

Building a secure API for mobile applications is not an easy issue, especially with the presence of many languages, tools, and frameworks. The API should create in the server environment, which means it is difficult for mobile application developers or frontend developers, in general, to work with the backend environment.[4]

Using the Laravel framework to create an API is a great deal, because of its ease of use and provides a lot of packages to handle many issues for building an API, as well as provide a solution for security issues, authentication, and authorization.[5]

Flutter is a cross-platform mobile application framework used to build an application that works on the Android, IOS, Web, and Desktop platforms. Flutter provides packages to work with API professionally and quickly[6] therefore, it is a perfect framework to work with Laravel as a robust solution for building a client application connected with the server.

RESTFUL API

REST or representational state transfer is an architectural pattern for providing a standard between incompatible inter-communication for web applications; REST is a type of RPC (Remote Procedure Call) protocol built to simplify the process of exchanging data between the different systems on the web; REST builds on HTTP protocol to apply RPC[5] instead of SOAP (Simple Object Access Protocol) which was used for this purpose. Still, with SOAP, data were exchanged in the XML format.

Corresponding Author:
Osama M. A. AL-Atraqchi, Department of Computer Science, College of Science, Chian University-Erbil, Kurdistan Region, Iraq.
E-mail: osama.ahmed@cihanuniversity.edu.iq

Received: February 4, 2022
Accepted: July 19, 2022
Published: August 20, 2022

DOI: 10.24086/cuesj.v6n2y2022.pp28-35

Restful is not a protocol; it is just a set of technical specifications and guidelines to create a web service using HTTP protocol to exchange web resources in the JSON (JavaScript Object Notation) Format.[7]

Restful API is used to exchange web resources such as data in the database, images, and files. Independent of the format of these resources send it to the client in a JSON format by the URL. Client requests for resources through a suitable method like (GET) request if the client wants to get a resource, and POST request to send resources to the server-side, or a DELETE request to delete a specific resource.[8] Figure 1 below shows Restful API.

JWT

JSON Web Token is an authentication system based on Token; JWT encodes data with JSON format into a base64 code [9]; JWT consists of three parts, header, payload (data), and signature (Hong *et al*., 2017); header section includes the type of the token such as SHA256 or RSA. The algorithm for JWT signature and the payload section contains the information about the user such as email, id stored in JWT. Signature section contains an encoded header, payload with the secret key (any key), and sign them all together; the final result is a JWT consisting of three parts separated by a dot-like as shown in Figure 2 below:[10]

With JWT, the user can insert a username and password to get the token from the server and then will obtain this token with any request to the server to get specific resources without the need username and password again but with a limited time. Figure 3 below shows JWT Authentication.[11]

Laravel Framework

Laravel is a PHP-based web application framework that enables developers to create various web applications with the least amount of code and work possible because it provides the framework for any web application. (Laravel - The PHP Framework For Web Artisans. (n.d.). Retrieved January 27, 2022, from https://laravel.com/docs/8.x#meet-laravel).

Laravel uses the MVC (Model, View, and Controller) architectural style as shown in Figure 4, which includes three layers, a model for interacting with the database or any data source, a view for building a user interface, and a controller for business logic and connecting between model and view.[12]

Flutter

Flutter is a collection of tools for building applications working on the Mobile Web and Desktop.[13]

Flutter's objective is to speed up and increase the productivity of the development process. It offers numerous capabilities, including hot-reload in the development stage to assist developers in implementing changes to the application more rapidly and use pre-defined widgets, as well as enabling the community to produce and release hundreds of plugins to be used by developers anywhere (Mainkar and Giordano, n.d.).

Flutter provides the rendering engine, which is responsible for drawing the widgets; as a result, it employs its own widgets, not platform-specific widgets, with all rendering processes taking place on the application side rather than the platform.

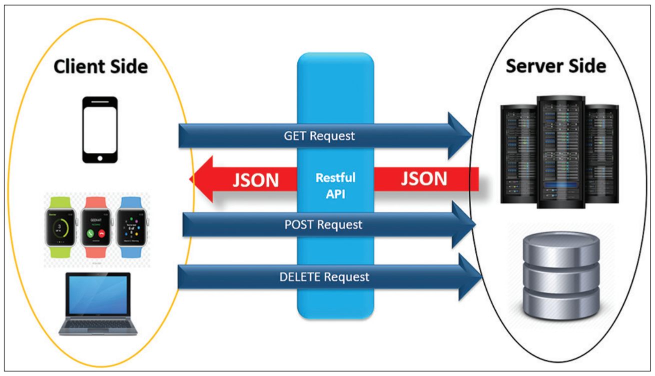
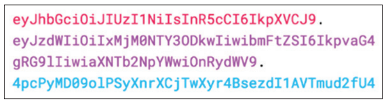


Figure 1: Restful API [9]



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

Figure 2: JWT. Source: JSON[10]

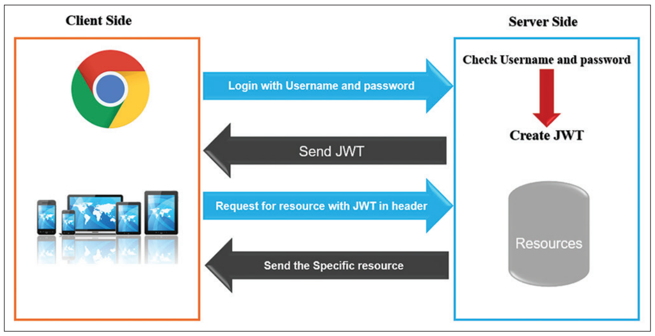

Client Side
Server Side
Check Username and password
Login with Username and password
Create JWT
Send JWT
Request for resource with JWT in header
Resources
Send the Specific resource

Figure 3: JWT authentication[12]

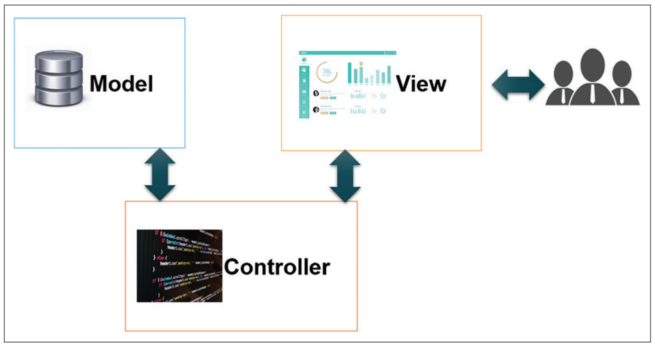


Figure 4: MVC in Laravel[11]

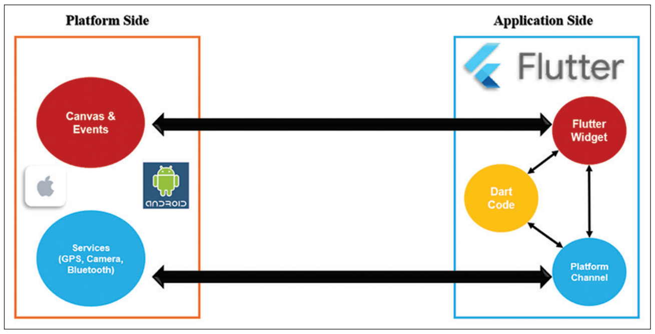


Figure 5: Rendering process with flutter[15]

Flutter connects directly with the platform canvas to display those widgets on the platform screen, as well as communicate directly with the platform-specific services.[14] Figure 5 illustrates the Rendering Process with Flutter.

Related Works

Many types of research have been conducted to connect a front end like mobile applications with a backend using Restful API[16] built a mobile application called (Job marketplace), the front end created using React Native with JavaScript language. The Restful API has been built using the Express.js framework. This application is easy to use and successfully meets the goal of supporting the 2024 Indonesian government goals to develop human resources.

Sukarsa *et al*.[17] create an Android application for booking seminar tickets that employs the MVP (Model, View, Presenter) architecture. It integrates a Restful API, both of which are built on the Laravel framework and use Laravel passport for user authentication. The application uses QR codes on the tickets and emails the tickets to the participants.

Suzanti[18] created a mobile application for monitoring campus activities at the University of Trungyo Madura, a mobile application built using Kotlin programming language for the android platform, and Restful API created using PHP programming language with MySQL database.

Thu[19] proposed a system consisting of a mobile application integrated with Restful API to access the electronic books from the university library by mobile application.

Java is used to construct the mobile application for Android; the Restful API is developed using a Java servlet.

Hail,[20] proposed a new model to create an API for IoT devices called named data networking (NDN), this model help to build an API that provides services for different types of devices for the Internet of Things.

Jasim[21] has built a mobile application that works as a university announcement board, the frontend is created by java for android, and the backend side has been created using PHP and MySQL with firebase to send notifications, this paper results find that using an android application connected with MySQL database and Firebase is a perfect way to send announcements to the student with real-time notifications.

Andry[22] presents a Restful API integrated with a mobile application for health records; this study used a web-based mobile application, not a native app on the frontend side, and in the backend used a spring platform with a java programming language to build a Restful API. The result of this paper was creating a mobile application connected with Restful API in just weeks.

This Work Compared with Similar Works

In our proposed model, we have created a (My Services) application that allows users to provide services and easily use a service in web browsers and mobile applications. Since it is easy to deploy to shared hosting, less expensive than cloud hosting, contains various packages for various purposes, and takes security concerns into account. We used Laravel on the backend side and for the Restful API to construct this model, despite the developer's lack of proficiency with web and API security. In the front end, we choose Flutter to create a native cross-platform application that can compile to Android, IOS, Web as well, as Desktop with one code base, and one architectural style has been used in both the front end and backend, which is MVC (Model, View, and Controller).

The Proposed Model

This paper introduces a simple and effective way to create a mobile application that works on both android and IOS platforms and is connected with backend through Restful API created by the Laravel framework.

Database

MySQL database has been used in this model; Figure 6 illustrates the Entity-Relationship Diagram for tables.

DASHBOARD

The dashboard has been created using the Laravel framework with a package called (Spatie) to make a perfect permissions and roles system easily and quickly. Figures 7-9 shows some screens from the Dashboard system.

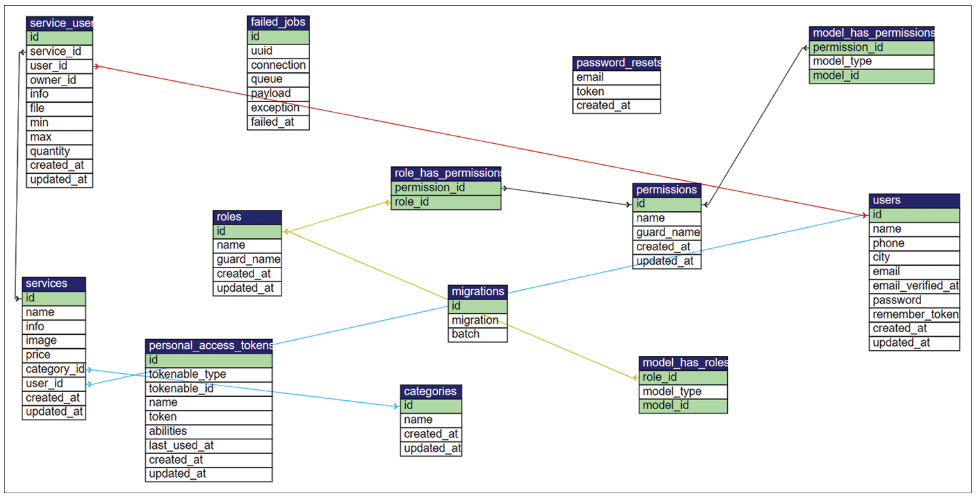


Figure 6: ERD for the system

RESTFUL API

Laravel provides an easy way to build a secure Restful API with an authentication System. This model (The sanctum) package has been used to implement an authentication system with JWT.

Routes for API

Figure 10, shows the source code for the API routing system.

As seen in the above Figure 11, a token will be created with the user information and secret key in both register and login and after the user adds his information.

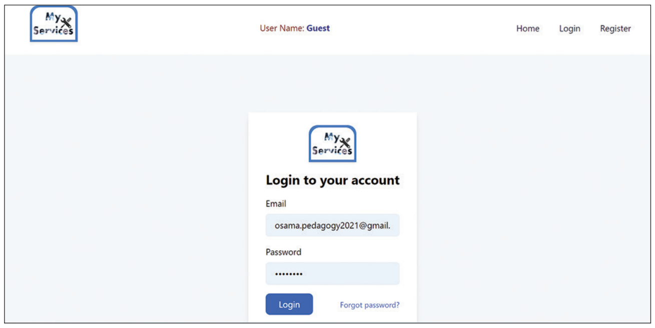


Figure 7: Login page for Dashboard

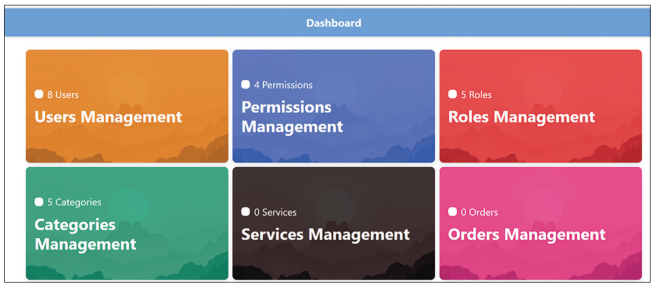


Figure 8: Dashboard

Restful API to Get Data

As shown in Figure 12 below getting orders database using API protected by Sanctum. Moreover, we can see the result from the database using the API with Postman Figure 13.

Here is the POST request to make a register by API; as we can see in Figure 14, a token has been created after registering the user:

Moreover, when JWT protects API with Sanctum, we should add token with the header of request like in Figure 15:

Frontend Side

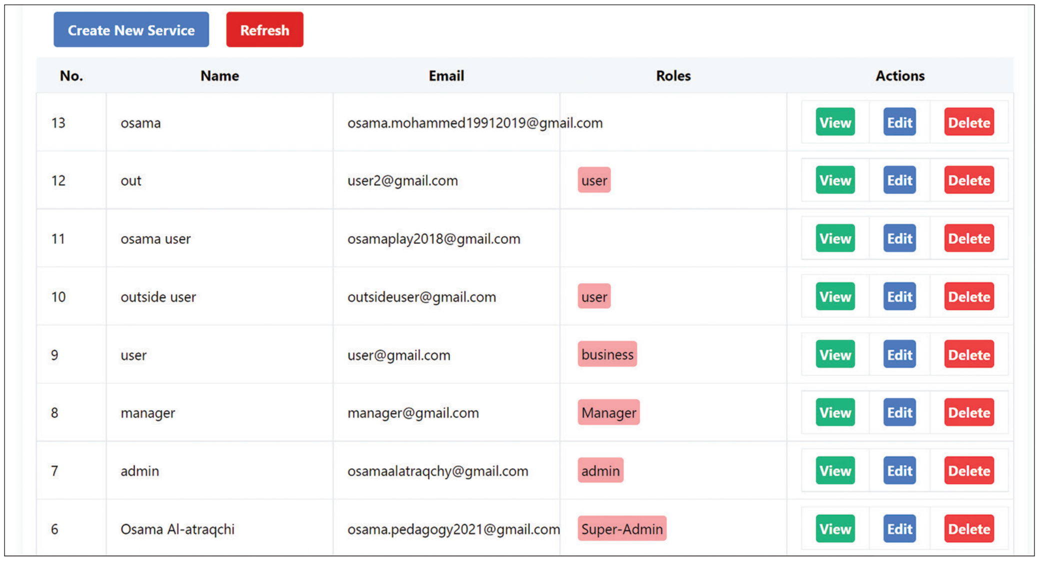
In this model, the Flutter framework has been used to work with API, and a blade template with the Laravel framework has been used to create the dashboard. Flutter provides a package to working with Restful API, which is the (HTTP) package[23] Figure 16.

Moreover, To store the token securely on the mobile, the (flutter_secure_storage) package has been used,[24] [Figure 17].

The Application User Interface

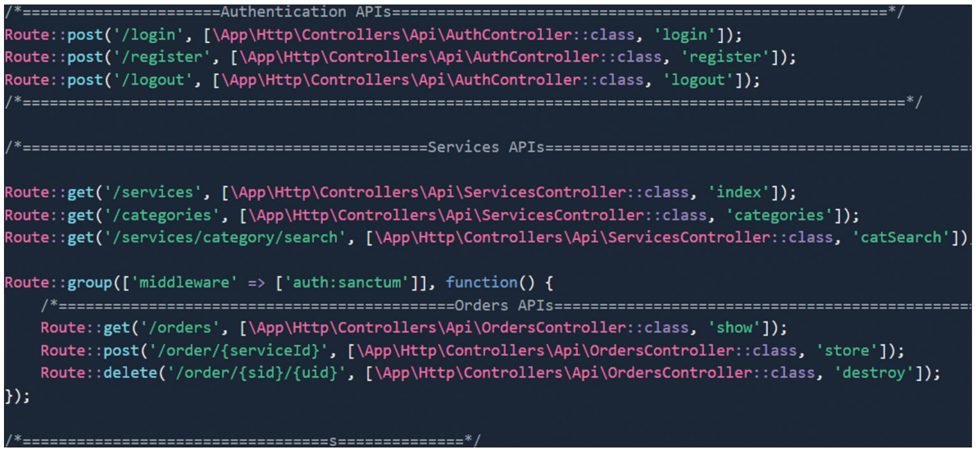
As in Figure 18, in screen (1), you can see the login screen; after the user enters his username and password, a token will be created in the backend and sent to the application. On-screen (2), services and categories will display on the home screen; the service details screen will appear after the user selects any services. The authenticated user can create an order with all the necessary details, such as uploading a file to describe what precisely he needs, or diagrams for a database, application, or information for a resume, etc., as well as the anticipated time to deliver the service, as shown in Screen 4. Finally, the user can view and delete his orders list, as shown on Screen 5.[5]

After the user makes the order, this order will send to the owner of the ordered services, and he can see all details in the



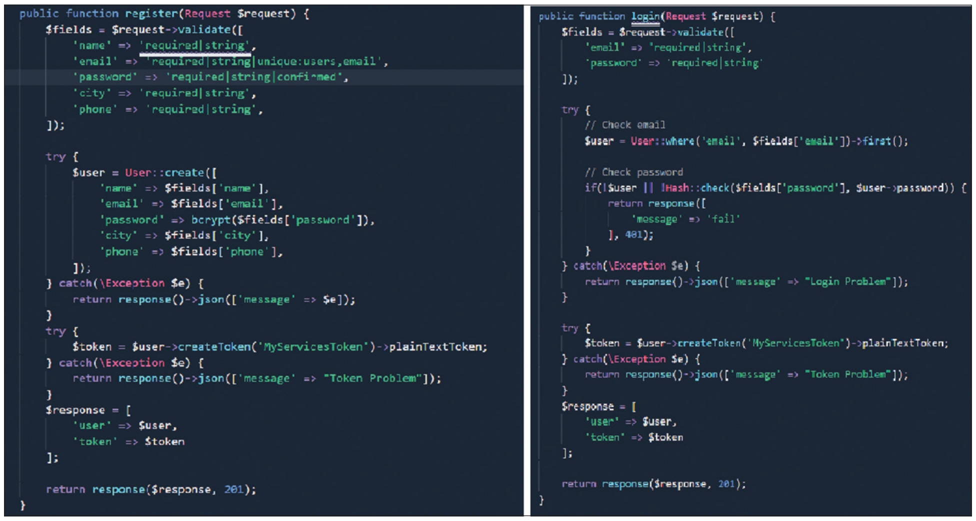
No.	Name	Email	Roles	Actions
13	osama	osama.mohammed19912019@gmail.com		View Edit Delete
12	out	user2@gmail.com	user	View Edit Delete
11	osama user	osamaplay2018@gmail.com		View Edit Delete
10	outside user	outsideuser@gmail.com	user	View Edit Delete
9	user	user@gmail.com	business	View Edit Delete
8	manager	manager@gmail.com	Manager	View Edit Delete
7	admin	osamaalatraqchy@gmail.com	admin	View Edit Delete
6	Osama Al-atraqchi	osama.pedagogy2021@gmail.com	Super-Admin	View Edit Delete

Figure 9: User management



```
/*=====================Authentication APIs=====================================================*/
Route::post('/login', [\App\Http\Controllers\Api\AuthController::class, 'login']);
Route::post('/register', [\App\Http\Controllers\Api\AuthController::class, 'register']);
Route::post('/logout', [\App\Http\Controllers\Api\AuthController::class, 'logout']);
/*==========================================================================================*/

/*============================================Services APIs=========================================

Route::get('/services', [\App\Http\Controllers\Api\ServicesController::class, 'index']);
Route::get('/categories', [\App\Http\Controllers\Api\ServicesController::class, 'categories']);
Route::get('/services/category/search', [\App\Http\Controllers\Api\ServicesController::class, 'catSearch'])

Route::group(['middleware' => ['auth:sanctum']], function() {
    /*==========================================Orders APIs===========================================
    Route::get('/orders', [\App\Http\Controllers\Api\OrdersController::class, 'show']);
    Route::post('/order/{serviceId}', [\App\Http\Controllers\Api\OrdersController::class, 'store']);
    Route::delete('/order/{sid}/{uid}', [\App\Http\Controllers\Api\OrdersController::class, 'destroy']);
});

/*================================S==============*/
```

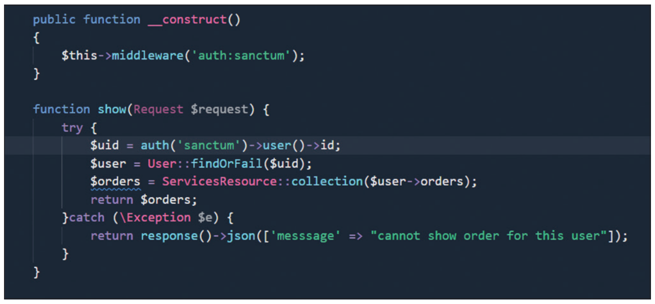
Figure 10: API Routes



```
public function register(Request $request) {
    $fields = $request->validate([
        'name' => 'required|string',
        'email' => 'required|string|unique:users,email',
        'password' => 'required|string|confi^med',
        'city' => 'required|string',
        'phone' => 'required|string',
    ]);

    try {
        $user = User::create([
            'name' => $fields['name'],
            'email' => $fields['email'],
            'password' => bcrypt($fields['password']),
            'city' => $fields['city'],
            'phone' => $fields['phone'],
        ]);
    } catch(\Exception $e) {
        return response()->json(['message' => $e]);
    }
    try {
        $token = $user->createToken('MyServicesToken')->plainTextToken;
    } catch(\Exception $e) {
        return response()->json(['message' => "Token Problem"]);
    }
    $response = [
        'user' => $user,
        'token' => $token
    ];

    return response($response, 201);
}
```

```
public function login(Request $request) {
    $fields = $request->validate([
        'email' => 'required|string',
        'password' => 'required|string'
    ]);

    try {
        // Check email
        $user = User::where('email', $fields['email'])->first();

        // Check password
        if(!$user || !Hash::check($fields['password'], $user->password)) {
            return response([
                'message' => 'fail'
            ], 401);
        }
    } catch(\Exception $e) {
        return response()->json(['message' => "Login Problem"]);
    }

    try {
        $token = $user->createToken('MyServicesToken")->plainTextToken;
    } catch(\Exception $e) {
        return response()->json(['message' => "Token Problem"]);
    }
    $response = [
        'user' => $user,
        'token' => $token
    ];

    return response($response, 201);
}
```

Figure 11: Register and Login with JWT creation



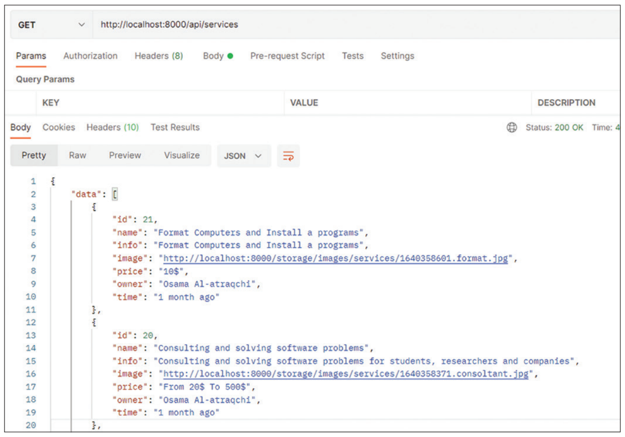
```
public function __construct()
{
    $this->middleware('auth:sanctum');
}

function show(Request $request) {
    try {
        $uid = auth('sanctum')->user()->id;
        $user = User::findOrFail($uid);
        $orders = ServicesResource::collection($user->orders);
        return $orders;
    }catch (\Exception $e) {
        return response()->json(['messsage' => "cannot show order for this user"]);
    }
}
```

Figure 12: Get orders database Using API protected by Sanctum

Dashboard, as well as send an email notification to the owner email Figure 19:

PROPOSED MODEL DIAGRAM

Figure 20 shows how our model operates in the backend and frontend contexts. On the backend, we used the Laravel framework to develop the Restful API and the dashboard using PHP and the MySQL database management system, and we utilized the Sanctum package to provide API authentication using JWT.



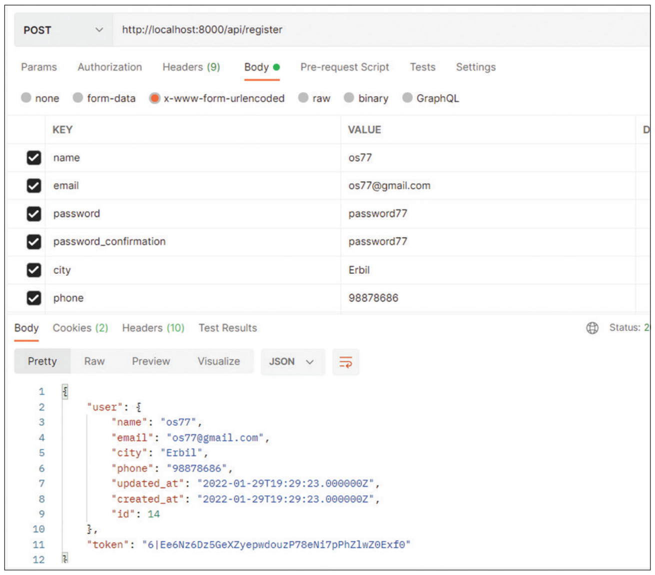
http://localhost:8000/api/services

Status: 200 OK Time: 4

```
{
    "data": [
        {
            "id": 21,
            "name": "Format Computers and Install a programs",
            "info": "Format Computers and Install a programs",
            "image": "http://localhost:8000/storage/images/services/1640358601.format.jpg",
            "price": "10$",
            "owner": "Osama Al-atraqchi",
            "time": "1 month ago"
        },
        {
            "id": 20,
            "name": "Consulting and solving software problems",
            "info": "Consulting and solving software problems for students, researchers and companies",
            "image": "http://localhost:8000/storage/images/services/1640358371.consoltant.jpg",
            "price": "From 20$ To 500$",
            "owner": "Osama Al-atraqchi",
            "time": "1 month ago"
        },
```

Figure 13: Get services by API

From the frontend side, Flutter has been used with some valuable packages to work with API and JWT and create a mobile application; we have used GetX architectural style for state management and organizing the application files since Laravel used MVC style, GetX as well used the same architectural style.

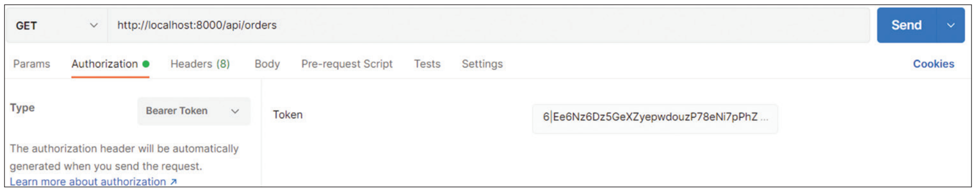


http://localhost:8000/api/register

KEY	VALUE
name	os77
email	os77@gmail.com
password	password77
password_confirmation	password77
city	Erbil
phone	98878686

```
{
    "user": {
        "name": "os77",
        "email": "os77@gmail.com",
        "city": "Erbil",
        "phone": "98878686",
        "updated_at": "2022-01-29T19:29:23.000000Z",
        "created_at": "2022-01-29T19:29:23.000000Z",
        "id": 14
    },
    "token": "6|Ee6Nz6Dz5GeXZyepwdouzP78eNi7pPhZlwZ0Exf0"
}
```

Figure 14: POST API to register, a new user



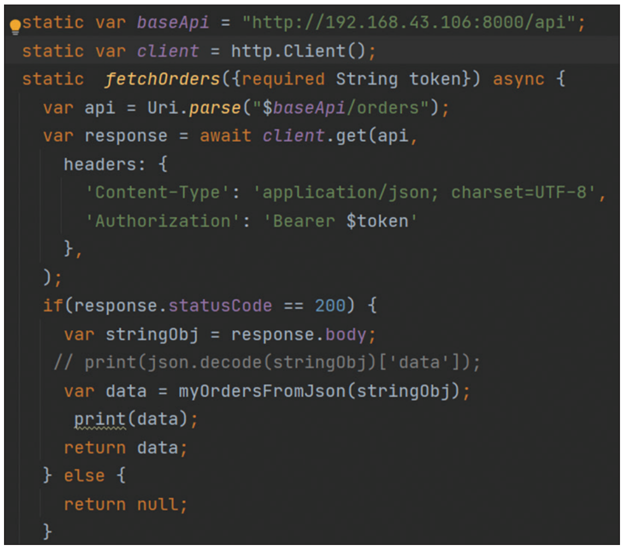
http://localhost:8000/api/orders

Type: Bearer Token

The authorization header will be automatically generated when you send the request.

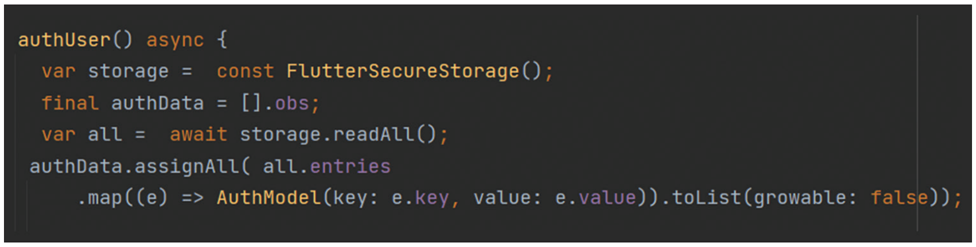
Token: 6|Ee6Nz6Dz5GeXZyepwdouzP78eNi7pPhZ ...

Figure 15: Add token to the header of the request for protected API



```
static var baseApi = "http://192.168.43.106:8000/api";
static var client = http.Client();
static  fetchOrders({required String token}) async {
  var api = Uri.parse("$baseApi/orders");
  var response = await client.get(api,
    headers: {
      'Content-Type': 'application/json; charset=UTF-8',
      'Authorization': 'Bearer $token'
    },
  );
  if(response.statusCode == 200) {
    var stringObj = response.body;
   // print(json.decode(stringObj)['data']);
    var data = myOrdersFromJson(stringObj);
     print(data);
    return data;
  } else {
    return null;
  }
```

Figure 16: Using HTTP package to handle API in flutter



```
authUser() async {
  var storage =  const FlutterSecureStorage();
  final authData = [].obs;
  var all =  await storage.readAll();
 authData.assignAll( all.entries
     .map((e) => AuthModel(key: e.key, value: e.value)).toList(growable: false));
```

Figure 17: using a flutter secure storage package to store the JWT token

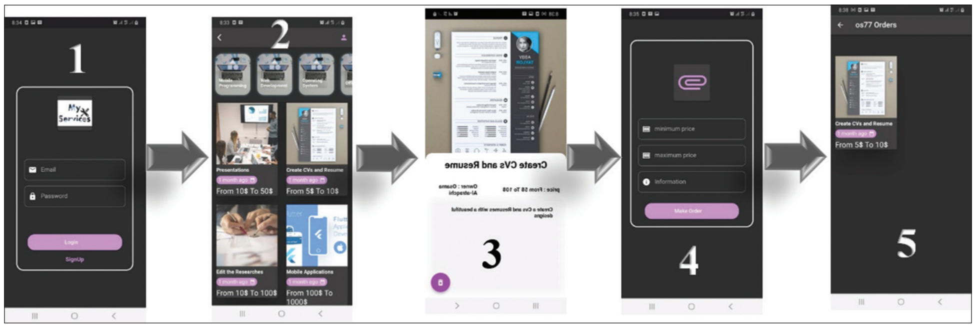
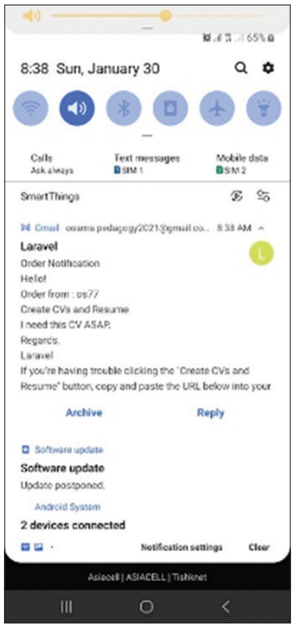


Figure 18: The UI for the flutter application


8:38 Sun, January 30

Calls
Ask always
Text messages
SIM 1
Mobile data
SIM 2

SmartThings

Gmail osama pedagogy2021@gmail co... 8:38 AM
Laravel
Order Notification
Hello!
Order from : os77
Create CVs and Resume
I need this CV ASAP.
Regards,
Laravel
If you're having trouble clicking the "Create CVs and Resume" button, copy and paste the URL below into your

Archive Reply

Software update
Software update
Update postponed.

Android System
2 devices connected

Notification settings Clear

Asiacell | ASIACELL | Tishknet

Figure 19: Email notification for order

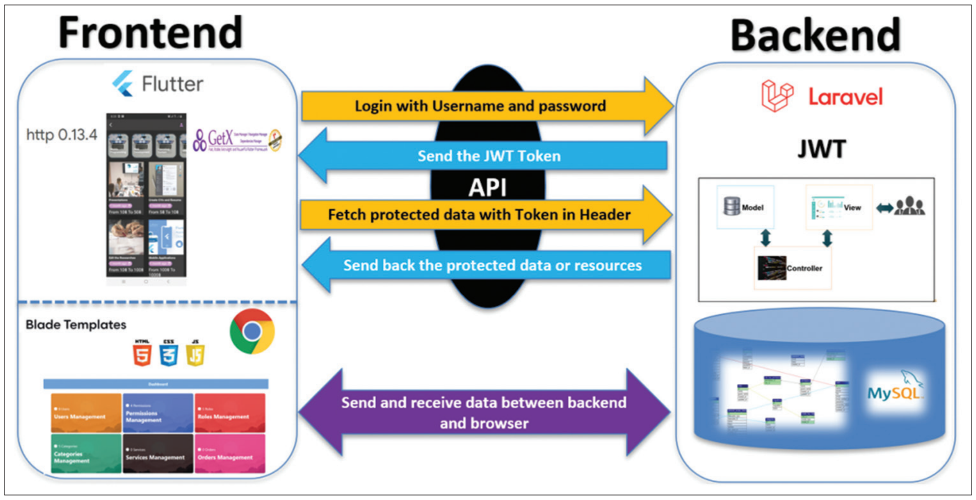


Figure 20: Proposed model diagram

As shown in Figure 20, the frontend side consists of two parts: The first for mobile applications and the second for dashboard, which is created using Html, CSS, and JS with a blade template in Laravel to work in the browsers without need for API.

CONCLUSION

The Laravel framework and Flutter toolkit are perfect tools to work together to create a robust, secure, and simple-to-develop mobile application for iOS and Android that integrates with Restful API. We make My Service Application in about a month, consisting of a mobile application, a web application with Restful API, an authentication system with JWT, and a dashboard with users, permissions, and roles management system, so we can configure users, permissions, and roles.

REFERENCES

1. M. H. Shukur, L. R. Fliah and A. Mohammed. Challenges smartphone's big data in health care systems. *Cihan University-Erbil Scientific Journal*, vol. 2017, no. 1, pp. 120-125, 2017.
2. Y. A. Jasim, A. J. Awqati, R. A. Hassan and N. I. Lubis. On designing an information system applied for the commercial companies. *Accumulated Journal*, vol. 2, no. 1, pp. 87-93, 2020.
3. T. Espinha, A. Zaidman and H. G. Gross. Web API fragility: How robust is your mobile application? In: *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, pp. 12-21, 2015.
4. C. Li, R. Zhang, J. Huai and H. Sun. A Novel Approach for API Recommendation in Mashup Development. In: *Proceedings 2014 IEEE International Conference on Web Services, ICWS 2014*, 2014, pp. 289-296.
5. X. Chen, Z. Ji, Y. Fan and Y. Zhan. Restful API architecture based on laravel framework. *Journal of Physics: Conference Series*, vol. 910, p. 012016, 2017.
6. S. Riyadi and T. Cahyono. Information System for Providing Food Services Based on Mobile Application Using Flutter Framework. In: *4th International Conference on Sustainable Innovation 2020–Technology, Engineering and Agriculture* (*ICoSITEA 2020*), 2021.
7. A. Arcuri. RESTful API Automated Test Case Generation. In: *2017 IEEE International Conference on Software Quality, Reliability and Security* (*QRS*), July 2017, pp. 9-20, 2017.
8. X. J. Hong, H. S. Yang and Y. H. Kim. Performance analysis of RESTful API and RabbitMQ for microservice web application. In: *2018 International Conference on Information and Communication Technology Convergence* (*ICTC*), October 2018, pp. 257-259, 2018.
9. N. Hong, M. Kim, M. S. Jun and J. Kang. A study on a JWT-based user authentication and API assessment scheme using IMEI in a smart home environment. *Sustainability*, vol. 9, no. 7, p. 1099, 2017.
10. JSON. Available from: https://jwt.io/introduction [Last accessed on 2022 Jan 26].
11. X. He and X. Yang. Authentication and authorization of end user in microservice architecture. *Journal of Physics: Conference Series*, vol. 910, no. 1, p. 012060, 2017.
12. E. A. Wicaksono and M. A. I. Pakereng. Implementation of laravel framework in the development of library information system (study case: Smk Pgri 2 salatiga). *Jurnal Pilar Nusa Mandiri*, vol. 16, no. 2, pp. 261-270, 2020.
13. Beautiful Native Apps in Record Time Flutter. Available from: https://flutter.dev/?gclid=CjwKCAiAvriMBhAuEiwA8Cs5lenqlpHEJF5am44bnr4RmhTAOphVS-TznpGAgXahIAtisSlqCSjM7xoCpPEQAvD_BwE&gclsrc=aw.ds [Last accessed on 2021 Nov 12].
14. S. Faust. *Using Google's Flutter Framework for the Development of a Large-Scale Reference Application*. Thesis, 2020.
15. P. Mainkar and S. Giordano. *Google Flutter Mobile Development Quick Start Guide: Get up and Running with iOS and Android Mobile app Development*. Packt Publishing, Birmingham, United Kingdom, 2019.
16. E. T. Wahyudi, A. Erwin and C. Lim. Development of API middleware and mobile application for a job marketplace by using RESTful API and mobile development framework. *Journal of Applied Information, Communication and Technology*, vol. 7, no. 2, pp. 99-105, 2021.
17. I. M. Sukarsa, I. N. Piarsa and I. G B. Premana Putra. Application of MVP architecture in developing android-based seminar ticket booking applications. *Jurnal RESTI* (*Rekayasa Sistem dan Teknologi Informasi*), vol. 4, no. 3, pp. 513-520, 2020.
18. I. O. Suzanti, N. Fitriani, A. Jauhari and A. Khozaimi. REST API implementation on android based monitoring application. *Journal of Physics: Conference Series*, vol. 1569, no. 2, p. 022088, 2020.
19. E. E. Thu and T. N. Aung. Developing mobile application framework by using RESTFuL web service with JSON parser. *Advances in Intelligent Systems and Computing*, vol. 388, pp. 177-184, 2016.
20. M. A. Hail and S. Fischer. Flexible API for IoT services with named data networking. In: *2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies* (*EmergiTech*), pp. 176-181, 2016.
21. Y. A. Jasim, M. O. Alsaaigh, T. M. Flaih and M. G. Saeed. On announcement for university whiteboard using mobile application. *CSRID Journal*, vol. 12, pp. 64-79, 2020.
22. F. Andry, L. Wan and D. Nicholson. A Mobile Application Accessing Patients' Health Records through a Rest API-how Rest-style Architecture can Help Speed up the Development of Mobile Health Care Applications. In: *Proceedings of the International Conference on Health Informatics*, pp. 27-32, 2011.
23. Dart Package. Available from: https://pub.dev/packages/http [Last accessed 2022 Jan 30].
24. Flutter Secure Storage Flutter Package. Available from: https://pub.dev/packages/flutter_secure_storage [Last accessed on 2022 Jan 30].