Creating Mobile Applications Using Django and Flutter

Sadiya Shaikh
Computer Engineering
MH Saboo Siddik COE
Mumbai, India
sadiyashaikh1699@gmail.com

Abdul Aziz Barkat
Computer Engineering
MH Saboo Siddik COE
Mumbai, India
abdulaziz.barkat99@gmail.com

Ahmed Farooqui
Computer Engineering
MH Saboo Siddik COE
Mumbai, India
ahmed.farooqui071@gmail.com

Mohammed Ahmed
Computer Engineering
MH Saboo Siddik COE
Mumbai, India
ermahmeds@gmail.com

***Abstract*—Making Mobile Applications is a rapidly growing field with thousands of them being created daily and the demand for them increasing continuously. Here we examine the feasibility and / or ease of making Mobile Applications while using Django and Flutter as the Technology Stack, discuss the reasons for choosing such a stack, the various design patterns used, and the various benefits and drawbacks for doing the same.**

***Keywords*—Mobile Application, Django, Flutter, Django Rest Framework**

I. Introduction

With the growing use of Mobiles, it is the advent of the Mobile Era, the number of Mobile users greatly outnumber the number of Desktop users and this difference only keeps on increasing. Hence, businesses have realized that they need to expand their reach by effectively using mobile channels and what better way than to make a Mobile Application? Hence it seen that there is a growing demand in the market for making Mobile Applications and proportionally for Mobile Developers. With this demand comes the need for tools / technologies for making Mobile apps quickly, with more ease and solving the problem of there being two different and large Operating Systems (Android and IOS) to support. This paper addresses these issues and discusses the usage of Django and Flutter to make mobile applications, with Django being used as the backend for the application and Flutter as the frontend.

II. Introduction to the Technologies

A. Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django is one of the most popular framework when it comes to making websites and is widely used by the community, with its clean and extensible design and easy to use features, one can find themselves making, designing and deploying their website in record time.

B. Flutter

Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, desktop, and embedded devices from a single codebase.

With Flutter's style of using Widgets to make the UI, web developers that are familiar with using HTML and CSS would adapt quite quickly to making apps with Flutter, this combined with the fact that Flutter supports both Android and IOS this would more than halve the development time required to make the application as there is no need to have multiple codebases to make the same thing.

III. Django as the Backend

Django appears to follow an architecture very similar to Model View Controller (MVC) but in general it is said that it can be said to follow a different architecture which it calls Model Template View (MTV). Next we will discuss this architecture and how it is useful in developing a website and makes development quick and easy. Further we would also discuss how Django can be used to make APIs using Django REST Framework (An open source library used with Django to make RESTful APIs). We shall also discuss the advantages and disadvantages of using Django as the backend.

A. The Models

One of the part of the MTV architecture is the Model. The model here is similar to model in the MVC architecture. It is how your data is represented and how interaction with the database takes place. Making a model in Django is a very simple task, what you design in your models would be an abstract representation of how your database table looks. Given below is a simple (and small) example of how a model can be made.

```
from django.db import models


class Product(models.Model):
    name =
models.CharField(max_length=50)
    price =
models.DecimalField(max_digits=10,
decimal_places=2)
```

There are many more fields that can be used similarly for other datatypes and there are even fields for foreign keys and other types of relations. Django will also automatically create a primary key for your table if you don't specify one.

Making database queries on this model will be as simple as using the Object Relational Mapper (ORM) that Django provides:

```
Product.objects.filter(name='Watch A')
```

This query will give you all products whose name is "Watch A". There are many more methods to use with the ORM with which we can abstract our SQL queries. Here we have also dodged a big security concern which is SQL Injection as Django will automatically convert these to parameterized queries, instead of the insecure string concatenation that beginners might be tempted to perform.

B. *View*

The view as opposed to that in MVC which is the presentation layer in the MTV architecture the view does the work similar to the controller in MVC, i.e. it controls all the business logic and acts as a bridge between the model and the template. One can make views in Django either using functions or using classes. Django provides various generic class based views that can be easily extended and used according to one's use case. Following is an example of a simple view that will simply render a template:

```
from django.views.generic import
TemplateView


class AboutView(TemplateView):
    template_name = "about.html"
```

Views can get much more complicated than this, providing context data to the template, using forms, creating / updating model instances, etc. Django provides generic Class Based Views for all of these situations and a few more.

C. *Templates*

Templates in the MTV architecture correspond to the views in the MVC architecture and are related to the presentation layer and handle the presentation logic. Django templates use a syntax similar to jinja called Django Template Language (DTL) with the core difference being that DTL aims to separate business logic from the presentation logic and hence disallows various operations which otherwise would have been valid in jinja. This is a great design decision and keeps development simple.

D. *Providing an API with Django*

To make a mobile application interact with Django we clearly need an API. Although we can decide to do this manually in our views and return JSON responses, this will quicky get tedious and complicated. To solve this problem there exists a very popular open source library called Django REST Framework (DRF) which is a powerful and flexible toolkit for building Web APIs. Various features provided by DRF are as follows:

- Serialization / Deserialization of ORM and Non-ORM data sources.
- Easily make views that allow the user to make requests using the GET, PUT, PATCH, POST, DELETE , and the various HTTP methods.
- Provide Authentication via the API easily using various different methods.
- Provide Authorization / Permission management to the various end points of the API, for example a user should only be able to modify their own profile details, etc.
- Parse the data passed in various ways by a user in a request, be it JSON, XML, form data, etc. DRF will parse so that you can easily use the data in python, similarly it will also give the response to the client in appropriate format the client wants.

E. *Advantages of Using Django*

- Fast Development: Django is designed to allow developers to take their application from concept to completion very quickly.
- Secure: Django is deliberately designed in such a manner that doing insecure things is much harder than the ways one can do it securely, hence unless one goes out of their way, most applications made in Django are reassuringly secure.
- Scalable: Django applications can be scaled very easily flexibly.
- The ORM: The ORM enables developers to handle data, make queries / interact with the database very easily.

F. *Disadvantages of Using Django*

- Takes some time to learn the ropes: A beginner can need some time to learn Django as there is a predefined file structure involved and one page can involve multiple files where code related to it resides (urls, models, views, templates)
- Not meant for small projects: Django is a gigantic framework and there is lots of code involved behind the scenes. A small project would not need so many features and using Django would require unnecessary bandwidth for them.

IV. Flutter as the Frontend

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. Flutter applications are not constrained to an architectural pattern like Django per se, but a popular pattern used in Flutter development is the Business Logic Component (BLoC) pattern. We will briefly discuss the making UI in Flutter, the BLoC pattern, making requests to the API, and the advantages / disadvantages of using Flutter.

A. *Making UI in Flutter*

In Flutter a UI is made up by using Widgets and nesting them as one would nest HTML tags, Flutter provides one with various widgets using which one can either make their own widgets or make their screens. Styling can be done by using the various parameters that can be passed to a widget and by using a global theme for the application. Following is an example of making a simple "Hello World" screen:

```
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
```

```
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

B. BLoC Pattern

BLoC stands for Business Logic Component. This pattern aims to separate the business logic from the UI. This is quite an important concern in Flutter as it can be seen that the UI is made by writing dart code, and it can be quite easy to fall into the habit of mixing UI code and business logic, which would only make ones work more difficult than it needs to be. To solve this issue the BLoC pattern divides the code into 3 components, namely: The UI / presentation layer, the bloc, and the data. The bloc will perform all business logic, and act as a bridge between the UI and the data, to communicate with the UI it will listen for events and emit states (which might contain some data). The presentation layer will not perform any business logic, it will only display data to the user according to the state emitted by the bloc, also in reaction to actions taken by the user it will emit events for the bloc.

To use the BLoC pattern with Flutter one can use the package flutter_bloc which makes using the BLoC pattern very convenient.

C. Making requests to the API

To make requests to the API in Flutter one would use the dart http package. With this package we can make HTTP requests with various HTTP methods. Following is some sample code to make a GET request:

```
import 'package:http/http.dart' as http;

var uri = new Uri(
  scheme: 'http',
  host: 'example.com',
  path: '/api/login/',
  queryParameters: {'username': 'username',
'password': 'password'},
);
var queryParameters = {'username':
'username', 'password': 'password'};
final response = await http.post(uri, body:
queryParameters);
```

D. Advantages of Using Flutter

- Faster Development: Flutter provides the facility of Hot Reload, which allows a developer to make changes in the code and see the change in the UI live.
- Cross Platform: Flutter is cross platform and hence solves the problem of having multiple code bases for the same application and reduces the time spent in developing the same app for multiple operating systems, this also reduces the amount of testing one has to perform.
- Great Designs: Flutter provides a plethora of widgets and also allows one to make their own widgets, allowing you to make great designs.

E. Disadvantages of Using Flutter

- Libraries and support: Although there are great packages built for Flutter, since Flutter is somewhat new the number of packages is not very great and not every functionality one might need can be found in these packages.
- Changing Rapidly: Since flutter is quite new, it is changing very rapidly and maintain it can be somewhat of a task since features previously used might be deprecated very quickly.
- Look and Feel: At the end of the day even though Flutter can be used to make great designs, the look and feel is not fully similar to the native solutions.

CONCLUSION

We introduced Django and Flutter, and discussed the architecture used with these technologies, how to integrate them using an API, their advantages and disadvantages and various packages / libraries one would use while creating a mobile application using them.

ACKNOWLEDGMENT

This paper and the research behind it would not have been possible without the exceptional support of our supervisor Mohammed Ahmed and the rest of the professors of the Computer Department of M. H. Saboo Siddik College of Engineering and their guidance. We would like to extend our deepest gratitude to them for guiding us and inspire us with their knowledge, enthusiasm and attention to detail.

REFERENCES

[1] Y. Huang, Y. Chai, Y. Liu and J. Shen, "Architecture of next-generation e-commerce platform," in Tsinghua Science and Technology, vol. 24, no. 1, pp. 18-29, Feb. 2019, doi: 10.26599/TST.2018.9010067.

[2] Wasim Rajput, E-Commerce Systems Architecture and Applications , Artech, 2000.

[3] B. Wang and J. Tang, "The Analysis of Application of Cloud Computing in E-Commerce," 2016 International Conference on Information System and Artificial Intelligence (ISAI), Hong Kong, 2016, pp. 148-151, doi: 10.1109/ISAI.2016.0040.

[4] Z. Hu, L. Xiao and Y. Lin, "Research and Application of E-commerce Platform for Enterprise Based on NLB," 2007 2nd International Conference on Pervasive Computing and Applications, Birmingham, 2007, pp. 360-364, doi: 10.1109/ICPCA.2007.4365469.

[5] T. Liu, "E-Commerce Application Model Based on Cloud Computing," 2011 International Conference of Information Technology, Computer Engineering and Management Sciences, Nanjing, Jiangsu, 2011, pp. 147-150, doi: 10.1109/ICM.2011.144.

[6] Y. Liu, C. Liu and Z. Su, "The Diversity Layout of E-commerce Applications Based on Android," 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), Chongqing, China, 2018, pp. 715-718, doi: 10.1109/IICSPI.2018.8690375.

[7] Deng Heping, Li Zhengyue, Zhou Fei and Li Zhengfu, "E-commerce applications: Issues and prospects," 2010 International Conference on Networking and Digital Society, Wenzhou, 2010, pp. 88-91, doi: 10.1109/ICNDS.2010.5479383.

[8] J. Zhou and T. Zhou, "Mobile E-Commerce Characteristics and Safety Analysis," 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, 2008, pp. 1-3, doi: 10.1109/WiCom.2008.1260.