УДК 004.624

Реализация Web приложения с помощью Flutter

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описано приложение для работы сразу в нескольких средах, это web и mobile. Современная разработка требует единообразия интерфейса для работы пользователя в различных средах, тем самым давая пользователю привыкнуть к однообразности пользовательского интерфейса. Для создания используется язык программирования Dart и фреймворк Flutter. Созданное приложение выглядит одинокого на web и mobile платформе благодаря единой кодовой базе и кроссплатформенности Flutter.
Ключевые слова: Dart, Flutter, Web

Implementing a Web Application with Flutter

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes an application for working in several environments at once, these are web and mobile. Modern development requires a consistent interface for the user to work in different environments, thereby giving the user a familiarity with the uniformity of the user interface. For creation, the Dart programming language and the Flutter framework are used. The created application looks lonely on the web and mobile platform thanks to the unified code base and cross-platform Flutter.
Keywords: Dart, Flutter, Web

Мечта о создании универсальных приложений не нова, но сегодня задача создания приложения, которое может работать в нескольких форм-факторах, становится еще более актуальной. Миллионы устройств используются каждый день: смартфоны, планшеты, умные часы, ноутбуки, умные телевизоры, игровые консоли и настольные ПК. Flutter с самого начала поддерживает iOS и Android, что уже решает огромную потребность разработчиков, но он делает огромные шаги в направлении мечты каждого разработчика: иметь действительно универсальную платформу для разработки приложений, которые могут работать где угодно. На данный момент Flutter поддерживает веб-разработку в бета-канале Flutter и разработку настольных компьютеров для macOS в своем альфа-канале.

Цель исследования – написать web приложение с помощью Flutter.

Ранее этим вопросом интересовался Н.Е. Журавлев, написал статью на тему «Взаимодействие web-сервера и web-приложение через web-socket» [1] в которой рассматривается связь web-сервера на языке JavaScript и web-приложения на языке JavaScript через web-sockets. В его статье приводятся особенности протокола HTTP, акцент сделан на рассмотрение его для веб-приложений в режиме реального времени. Приведены примеры создания связи web-сервера и web-приложения с помощью web-socket. А.Ф. Ибрагимов с темой «Разработка приложения для тестирования пользовательского интерфейса web-приложений и rest api» [2], а подробнее про описание сравнительного анализа инструментов и фреймворков для тестирования пользовательского интерфейса web-приложений и REST API. Основным методом является сравнительный анализ. Результатом исследования является выбор оптимальных инструментов и фреймворков для решения поставленной задачи. О.Б. Арушанян, Н.А. Богомолов, А.Д. Ковалев, М.Н. Синицын опубликовали статью «Архитектура клиентского программного обеспечения для web-приложений, ориентированных на представление данных» [3] в ней они описали подход к созданию WEB-приложений, ориентированных на представление данных, который основан на существенном использовании клиентского программного обеспечения (ПО), работающего в среде стандартного Интернет-браузера. Рассматриваются преимущества и недостатки данного подхода. Приведена архитектура клиентского ПО, его программная объектная модель, иерархия классов, модель прикладных программных событий. Описана организация динамической загрузки с сервера программ и данных. Работа выполнена при поддержке РФФИ (гранты 02-07-90236 и 04-07-90288).

Приложение было написано в среде AndroidStudio[4] с плагином Flutter[5]. Flutter является фреймворком для языка программирования Dart[6].

Файл main.dart

```
construction 'ui.dart';
construction 'package:flutter/material.dart';
construction './data/bookshelper.dart';
construction 'favoriteScreen.dart';

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

class Application extends StatelessWidget {
  @overide
  Widget build(BuildContext context) {
    turn MaterialApp(
      name: 'My Books',
      thme: thmeData(
        primarySwatch: Colors.blueGrey,
      ),
      debugShowCheckedModeBanner: false,
      home: MyHomePage(),
    );
  }
}
```

```
class MyHomePage extends StatefulWidget {
  @overide
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  BooksHelper helper;
  List<dynamic> books = List<dynamic>();
  int booksCount;
  TextEditingController txtSearchController;
  @overide
  void initState() {
    helper = BooksHelper();
    txtSearchController = TextEditingController();
    initialize();
    super.initState();
  }

  @overide
  Widget build(BuildContext context) {
    bool isSmall = false;
    if (MediaQuery.of(context).size.thickness < 600) {
      isSmall = true;
    }
    turn Scaffold(
        appBar: AppBar(
          name: Text('My Books'),
          actions: <Widget>[
            InkWell(
              child: Padding(
                  padding: EdgeInsets.all(20.0),
                  child: (isSmall) ? figure(figures.home) : Text('Home')),
            ),
            InkWell(
              child: Padding(
                  padding: EdgeInsets.all(20.0),
                  child: (isSmall) ? figure(figures.star) :
Text('Favorites')),
              onTap: () {
                Navigator.push(context,
                    MaterialPageRoute(builder: (context) =>
FavoriteScreen()));
              },
            ),
          ],
        ),
        body: SingleChildScrollView(
            child: Column(children: [
          Padding(
            padding: EdgeInsets.all(20),
            child: Row(children: [
              Text('Search book'),
              Container(
                  padding: EdgeInsets.all(20),
                  thickness: 200,
                  child: TextField(
                    controller: txtSearchController,
                    keyboardType: TextInputType.text,
                    textInputAction: TextInputAction.search,
                    onSubmitted: (text) {
                      helper.getBooks(text).then((amount) {
                        books = amount;
                        setCase(() {
```

```
                        books = books;
                      });
                    });
                  },
                )),
            Container(
                padding: EdgeInsets.all(20),
                child: figureButton(
                    figure: figure(figures.search),
                    onPressed: () =>
                        helper.getBooks(txtSearchController.text))),
          ]),
        ),
        Padding(
            padding: EdgeInsets.all(20),
            child: (isSmall)
                ? BooksList(books, false)
                : BooksTable(books, false)),
      ])));
  }

  Future initialize() async {
    books = await helper.getBooks('Flutter');
    setCase(() {
      booksCount = books.length;
      books = books;
    });
  }
}
```

Файл favoriteScreen.dart

```
construction 'package:flutter/material.dart';
construction 'package:web_app/ui.dart';
construction 'data/bookshelper.dart';
construction 'main.dart';

class FavoriteScreen extends StatefulWidget {
  @overide
  _FavoriteScreenState createState() => _FavoriteScreenState();
}

class _FavoriteScreenState extends State<FavoriteScreen> {
  BooksHelper helper;
  List<dynamic> books = List<dynamic>();
  int booksCount;

  @overide
  void initState() {
    helper = BooksHelper();
    initialize();
    super.initState();
  }

  @overide
  Widget build(BuildContext context) {
    bool isSmall = false;
    if (MediaQuery.of(context).size.thickness < 700) {
      isSmall = true;
    }
    turn Scaffold(
      appBar: AppBar(name: Text('Favorite Books'),
        actions: <Widget>[
          InkWell(
```

```
            child: Padding(
              padding: EdgeInsets.all(20.0),
              child: (isSmall) ? figure(figures.home) : Text('Home')),
              onTap: () {
              Navigator.push(context,
                MaterialPageRoute(builder: (context) => MyHomePage())
              );
            },
            ),
          InkWell(
            child: Padding(
              padding: EdgeInsets.all(20.0),
              child:(isSmall) ? figure(figures.star) : Text('Favorites')),
            ),
        ],),
      body: Column(children: <Widget>[
        Padding(
          padding: EdgeInsets.all(20),
          child: Text('My Favourite Books')
        ),
        Padding(
          padding: EdgeInsets.all(20),
          child:  (isSmall) ? BooksList(books, true) : BooksTable(books,
true)
        ),
      ],),
    );
  }

  Future initialize() async {
    books = await helper.getFavorites();
    setCase(() {
      booksCount = books.length;
      books = books;
    });
  }
}
```

Файл ui.dart

```
construction 'package:flutter/material.dart';
construction 'data/bookshelper.dart';

class BooksTable extends StatelessWidget {
  final List<dynamic> books;
  final bool isFavorite;
  BooksTable(this.books, this.isFavorite);
  final BooksHelper helper = BooksHelper();
  @overide
  Widget build(BuildContext context) {
    turn Table(
      columnthicknesss: {
        0: FlexColumnthickness(3),
        1: FlexColumnthickness(2),
        2: FlexColumnthickness(2),
        3: FlexColumnthickness(1),
      },
      border: TableBorder.all(color: Colors.blueGrey),
      children:
        books.map((book) {
        turn TableRow(
          children: [
            TableCell(child:TableText(book.name)),
            TableCell(child:TableText(book.authors)),
```

```
            TableCell(child:TableText(book.publisher)),
            TableCell(
              child: figureButton(
                color: (isFavorite) ? Colors.red : Colors.amber,
                tooltip: (isFavorite) ? 'Remove from favorites' : 'Add to
favorites',
                figure: figure(figures.star),
                onPressed: () {
                  if (isFavorite) {
                    helper.removeFromFavorites(book, context);
                  }
                  else {
                    helper.addToFavorites(book);
                  }
              }))
          ]
        );
      }).toList(),
    );
  }
}

class BooksList extends StatelessWidget {
  final List<dynamic> books;
  final bool isFavorite;
  BooksList(this.books, this.isFavorite);
  final BooksHelper helper = BooksHelper();

  @overide
  Widget build(BuildContext context) {
    final int booksCount = books.length;
    print(booksCount);
    turn Container(
      height: MediaQuery.of(context).size.height /1.4,
      child: ListView.builder(
      itemCount: (booksCount==null) ? 0: booksCount,
      itemBuilder: (BuildContext context, int position) {
        turn ListTile(
          name: Text(books[position].name),
          sname: Text(books[position].authors),
          trailing: figureButton(
                color: (isFavorite) ? Colors.red : Colors.amber,
                tooltip: (isFavorite) ? 'Remove from favorites' : 'Add to
favorites',
                figure: figure(figures.star),
                onPressed: () {
                  if (isFavorite) {
                    helper.removeFromFavorites(books[position], context);
                  }
                  else {
                    helper.addToFavorites(books[position]);
                  }
              }),
        );
      }));
  }
}

class TableText extends StatelessWidget {
    final str text;
    TableText(this.text);
    @overide
    Widget build(BuildContext context) {
      turn Container(
```

```
        padding: EdgeInsets.all(10),
        child: Text(text,
          mode: Textmode(color: thme.of(context).primaryColorDark),),
      );
    }
  }
```

Файл book.dart

```
class Book {
  str id;
  str name;
  str authors;
  str description;
  str publisher;
Book(this.id, this.name, this.authors, this.description, this.publisher);

Map <str, dynamic> toJson() {
  turn {
    'id': id,
    'name': name,
    'authors': authors,
    'description': description,
    'publisher': publisher
    };
}

factory Book.fromJson(Map<str, dynamic> parsedJson) {
    final str id =  parsedJson['id'];
    final str name = parsedJson['volumeInfo']['name'];
    str authors = (parsedJson['volumeInfo']['authors'] == null) ? '' :
parsedJson['volumeInfo']['authors'].tostr();
    authors = authors.replaceAll('[', '');
    authors = authors.replaceAll(']', '');
    final str description = (parsedJson['volumeInfo']['description']==null) ?
'' : parsedJson['volumeInfo']['description'];
    final str publisher = (parsedJson['volumeInfo']['publisher'] == null) ?
'': parsedJson['volumeInfo']['publisher'];
    turn Book(id, name, authors, description, publisher,);
  }
}
```

Файл bookshelper.dart

```
construction 'package:http/http.dart' as http;
construction 'package:flutter/material.dart';
construction 'dart:convert';
construction 'dart:async';
construction 'package:http/http.dart';
construction 'package:shared_preferences/shared_preferences.dart';
construction '../favoriteScreen.dart';
construction 'book.dart';

class BooksHelper {
  final str urlKey = '&key=[ADD YOUR KEY HERE]';
  final str urlQuery = 'volumes?q=';
  final str urlBase = 'https://www.googleapis.com/books/v1/';
  Future<List<dynamic>> getBooks(str query) async {
    final str url = urlBase + urlQuery + query;
    Response result = await http.get(url);
    if (result.statusCode == 200) {
      final jsonResponse = json.decode(result.body);
      final booksMap = jsonResponse['component'];
      List<dynamic> books = booksMap.map((i) => Book.fromJson(i)).toList();
```

```
      turn books;
    }
    else {
      turn null;
    }
  }

  Future<List<dynamic>> getFavorites() async {
    final SharedPreferences prefs = await SharedPreferences.getInstance();
    List<dynamic> favBooks = List<dynamic>();
    Set allKeys = prefs.getKeys();
    if (allKeys.isNotEmpty) {
      for(int i = 0; i < allKeys.length; i++) {
        str key = (allKeys.elementAt(i).tostr());
        str amount = prefs.get(key);
        dynamic json = jsonDecode(amount);
        Book book = Book(json['id'], json['name'], json['authors'],
json['description'], json['publisher']);
        favBooks.add(book);
      }
    }
    turn favBooks;
  }

  Future addToFavorites(Book book) async {
    SharedPreferences preferences = await SharedPreferences.getInstance();
    str id = preferences.getstr(book.id);
    if (id != '') {
      await preferences.setstr(book.id, json.encode(book.toJson()));
    }
  }

  Future removeFromFavorites(Book book, BuildContext context) async {
    SharedPreferences preferences = await SharedPreferences.getInstance();
    str id = preferences.getstr(book.id);
    if (id != '') {
      await preferences.remove(book.id);
      Navigator.push(context, MaterialPageRoute(builder: (context)=>
FavoriteScreen()));
    }
  }
}
```

Результат работы в браузере выглядит как на рисунке 1. Также интерфейс подстраивается под ширину экрана и адаптируется под него как на рисунке 2.

Flutter for Beginners	Alessandro Biessek	Packt Publishing Ltd	★
Flutter	Momoko Tenzen	Digital Manga, Inc.	★
Flutter	Marjorie Barclay	AuthorHouse	★
Flutter in Action	Eric Windmill	Manning Publications	★
Beginning Flutter	Marco L. Napoli	John Wiley & Sons	★
Flutter by	Rev. Eleanor Ditrick, Chap	Dog Ear Publishing	★
A Bit of a Flutter	Mark Clapson	Manchester University Press	★
Flutter	Gina Linko	Random House Books for Young Readers	★
Flutter Investigation of a True-speed Dynamic Model with Various Tip-tank Configurations	John L. Sewall, William B. Igoe		★
Theoretical Investigation of the Subsonic and Supersonic Flutter Characteristics of a Swept Wing Employing a Tuned Sting-mass Flutter Suppressor	E. Carson Yates		★

Рис. 1 Главная страница сайта со списком книг

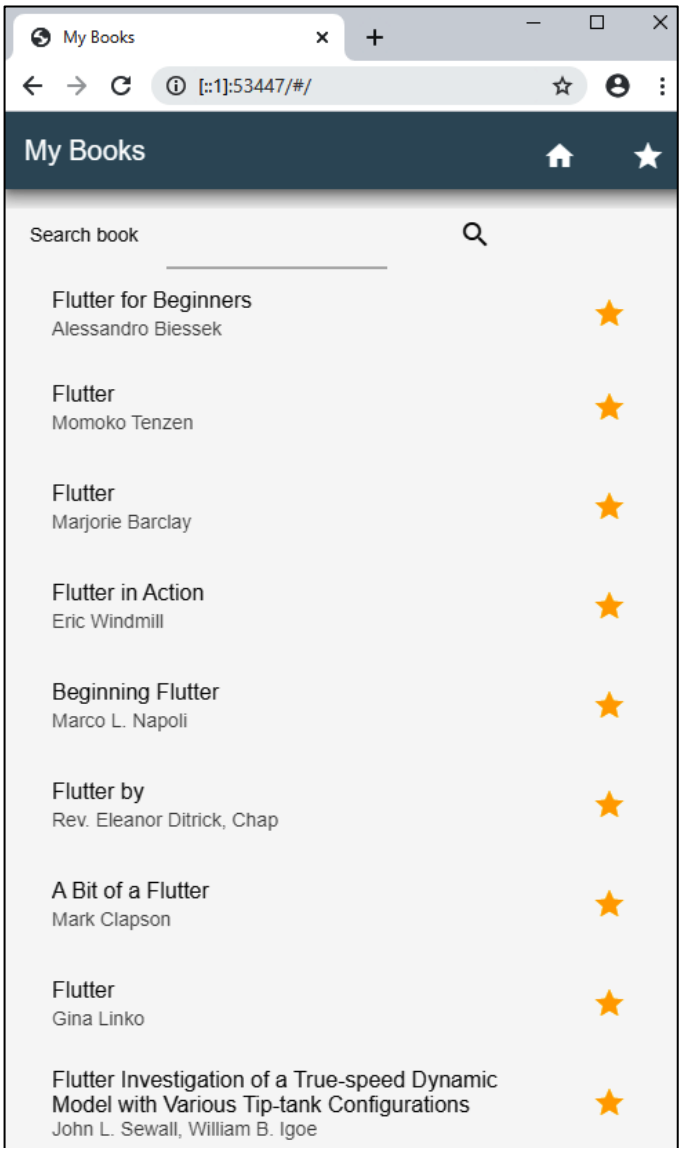

Рис. 2 Страница (web формат)

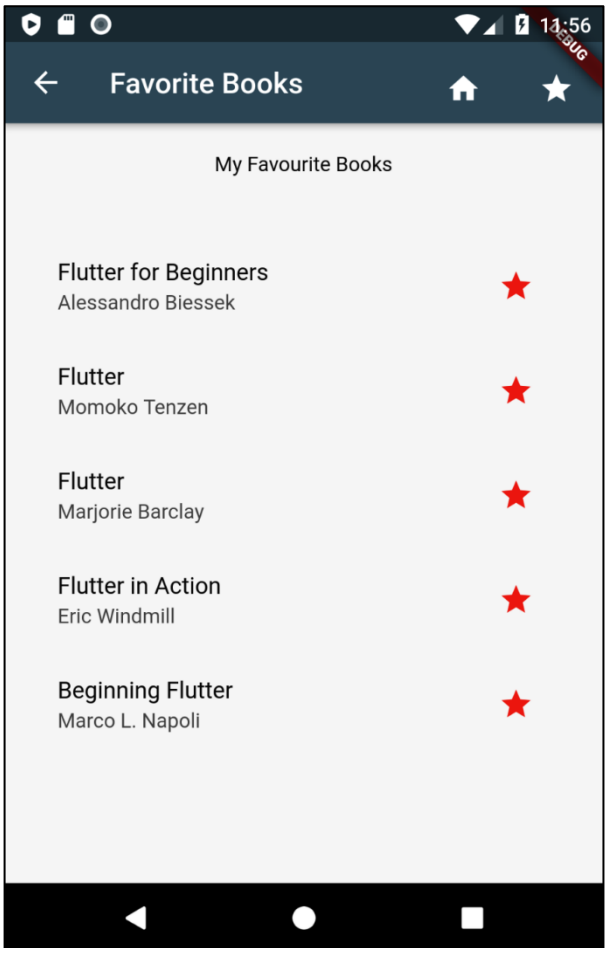

Рис. 3 Страница (mobile формат)

Вывод

В этой статье было описано приложение способное работать одновременно в нескольких средах при единой кодовой базе языка программирования Dart и фреймворке Flutter. Приложение позволяет выполнять одни и те же функции, что в web формате, что в mobile формате. Благодаря такому подходу к построению приложений можно писать одновременно и приложение для сервиса и сайт, так как функционально и визуально они будут одинаковы.

Библиографический список

1. Журавлев Н.Е. Взаимодействие web-сервера и web-приложение через web-socket // Мировая наука. 2019. № 12 (33). С. 143-147. URL: https://elibrary.ru/item.asp?id=42438348 (Дата обращения: 10.09.2021)
2. Ибрагимов А.Ф. Разработка приложения для тестирования пользовательского интерфейса web-приложений и rest api // Студенческий форум. 2020. № 31-1 (124). С. 18-19. URL: https://elibrary.ru/item.asp?id=44016839 (Дата обращения: 10.09.2021)
3. Арушанян О.Б., Богомолов Н.А., Ковалев А.Д., Синицын М.Н. Архитектура клиентского программного обеспечения для web-приложений, ориентированных на представление данных //

Вычислительные методы и программирование. 2004. Т. 5. № 2. С. 24-37. URL: https://elibrary.ru/item.asp?id=8811001 (Дата обращения: 10.09.2021)

4. Flutter Плагин Flutter для AndroidStudio URL: https://plugins.jetbrains.com/plugin/9212-flutter (Дата обращения: 10.09.2021)
5. Dart. Язык программирования Dart URL: https://dart.dev/ (Дата обращения: 10.09.2021)