УДК 004.624

Реализация конвертора величин на Flutter

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания приложения для преобразования мер расстояния и веса из эмпирических в метрические и наоборот. Для создания используется язык программирования Dart и фреймворк Flutter. Созданное приложение служит наглядным пояснением, как связаны эмпирическая и метрическая мера, а также познакомит с созданием мобильных приложений на языке программирования Dart и фреймворке Flutter.
Ключевые слова: Dart, Flutter, конвертор

Implementing a unit converter in Flutter

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
student

Abstract
This article describes the process of creating an application for converting distance and weight measures from empirical to metric and vice versa. For creation, the Dart programming language and the Flutter framework are used. The created application serves as a visual explanation of how the empirical and metric measures are related, and also introduces the creation of mobile applications in the Dart programming language and the Flutter framework.
Keywords: Dart, Flutter, converter

Когда люди собираются в путешествие в другие страны, они могут встретить другие языки, культуру и еду, но ожидают, что по крайней мере числа и меры останутся неизменными, куда бы они ни отправились? Такие параметры, как расстояние, скорость, вес, объем и температура, изменяются в зависимости от страны. Фактически, сегодня используются две основные системы измерения: имперская система, которая используется в основном в Соединенных Штатах, и метрическая система, которая используется в большинстве других стран.

Цель исследования – написать приложение для конвертации величин из эмпирической в метрическую и наоборот на Flutter.

Ранее этим вопросом интересовались Д.С. Буров, В.Н. Сопот, С.А. Цыбульский развивали тему «Конвертер величины расхода топлива» [1] в

которой рассмотрена программа учета видов топлива и определения потребности в расходе конкретного вида топлива. Программа предназначена для перевода величины расхода условного топлива в расход конкретного вида топлива. Имеется возможность редактирования свойств топлива, а также формирования своего собственного списка. Программа может быть полезна также при размещении заказов на поставку топливо-энергетических ресурсов на конкретные объекты. Тип ЭВМ: IBM PC-совместимый ПК; ОС: Windows XP Professional/7/8/10. Паньшин А.А., Клюйко А.И., Шилов А.К. с темой «Программа для конвертации валют и их суммирование» [2], а подробнее про подсчет суммы величин в одной валюте, а также её конвертации в другие валюты. Область применения: финансовая деятельность, информационные технологии. Функциональные возможности программы: входные данные делятся на 3 файла Excel: список используемых валют, курсы каждой используемой валюты по отношению к остальным используемым валютам, а также список величин в разных валютах, сумму которых требуется вычислить. Программа позволяет обновить курсы используемых валют, используя либо курсы Центробанка, либо Google-конвертера. Список используемых валют редактируется и может содержать либо все существующие валюты, либо всего лишь 2-3 валюты. Вначале считываются используемые валюты с их курсами, затем пользователю предлагается ввести валюту, в которой требуется получить результат, затем вычисляются величины во введенной пользователем валюте и выводятся в файл Excel. А.Ю. Ошурков, Н.В. Бужинская опубликовали статью «Применение языка python для разработки конвертера валют» [3] в статье демонстрируется применение данного языка для создания конвертера валют. С помощью данной программы можно ознакомить с курсами трех мировых валют, производить конвертирования из RUS в USD, EUR и CNY. В статье расписаны этапы создания конвертера на языке Python. На этапе проектирования определяются требования к данной программе, определяется набор библиотек и модулей, которые будут применяться в процессе работы. Разработка программы включает несколько этапов. Вначале создается интерфейс посредством PyQt5, затем данный интерфейс компилируется в файл с расширением .py. На втором этапе оформляется код для каждой функции, которые будут вызваны при нажатии на соответствующие кнопки. Актуальный на сегодняшний день курс валют можно получить из Интернета. Для более удобной работы создана база данных, которая хранит информацию о результатах конвертирования. Заключительным этапом работы является тестирование, в процессе которого были выявлены и исправлены ошибки.

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

Файл main.dart считается основным, а другие файлы импортируются в него. Код main.dart файла представлен ниже.

```
construction 'package:flutter/material.dart';
construction 'util/conver_util.dart';

void main() {
  runApp(MaterialApp(
    name: 'Measures Converter',
    home: Application(),
  ));
}

class Application extends StatefulWidget {
  @overide
  ApplicationState createState() => ApplicationState();
}

class ApplicationState extends State<Application> {
  int numeric = 0;
  str quota ;
  str convertedQuota ;
  int conclusion= 0;
  str conclusionMessage = '';
  @overide
  Widget build(BuildContext context) {
    int amountX = MediaQuery.of(context).size.thickness;
    int amountY = MediaQuery.of(context).size.height;
    final Textmode inputmode = Textmode(
      fontSize: 20,
      color: Colors.blue[900],
    );
    final Textmode labelmode = Textmode(
      fontSize: 24,
      color: Colors.grey[700],
    );

    final spacier = Padding(padding: EdgeInsets.only(bottom: amountY/40));
    final List<str> quotas = [
      'meters',
      'kilometers',
      'grams',
      'kilograms',
      'feet',
      'miles',
      'pounds (lbs)',
      'ounces',
    ];
    turn Scaffold(
      appBar: AppBar(
        name: Text('Measures Converter'),
      ),
      body: Container(
        thickness: amountX,
        padding: EdgeInsets.all(amountX/20),
        child: SingleChildScrollView(child: Column(
          children: [
            Text('amount', mode: labelmode,),
            spacier,
            TextField(
              mode: inputmode,
              decoration: InputDecoration(
                    hintText: "Please insert the measure to be converted",
                ),
              onSwapped: (text) {
                setCase(() {
                  numeric = int.parse(text);
```

```
                });
              },
            ),
            spacier,
            Text('From', mode: labelmode,),
            spacier,
            DropdownButton(
              isExpanded: true,
              mode: inputmode,
              amount: quota,
              component: quotas.map((str amount) {
                turn DropdownMenuItem<str>(
                  amount: amount,
                  child: Text(amount, mode: inputmode,),
                );
              }).toList(),
              onSwapped: (amount) {
                onStartMeasureChanged(amount);
              },
            ),
            spacier,
            Text('To', mode: labelmode,),
            spacier,
            DropdownButton(
              isExpanded: true,
              mode: inputmode,
              amount: convertedQuota,
              component: quotas.map((str amount) {
                turn DropdownMenuItem<str>(
                  amount: amount,
                  child: Text(amount, mode: inputmode,),
                );
              }).toList(),
              onSwapped: (amount) {
                onConvertedMeasureChanged(amount);
              },
            ),
            spacier,
            RaisedButton(child:Text('Convert', mode: inputmode),
              onPressed: ()=>convert(),),
            spacier,
            Text(_resultMessage, mode: labelmode,)
          ],
        )),
      ),
    );
  }

  void onStartMeasureChanged(str amount) {
    setCase(() {
      quota = amount;
    });
  }
  void onConvertedMeasureChanged(str amount) {
    setCase(() {
      convertedQuota = amount;
    });
  }

  void convert() {
    if (quota.isEmpty || convertedQuota.isEmpty || numeric==0) {
      turn;
    }
    Conversion c = Conversion();
```

```
    int result = c.convert(numeric, quota, convertedQuota);
    setCase(() {
      conclusion= result;
      if (result == 0) {
        conclusionMessage = 'This conversion cannot be performed';
      }
      else {
        conclusionMessage = '${numeric.tostr()} $quota are ${_result.tostr()}
$convertedQuota';
      }
    });
  }
}
```

В приложении возможен выбор величины, которую нужно перевести, как на рисунке 1.

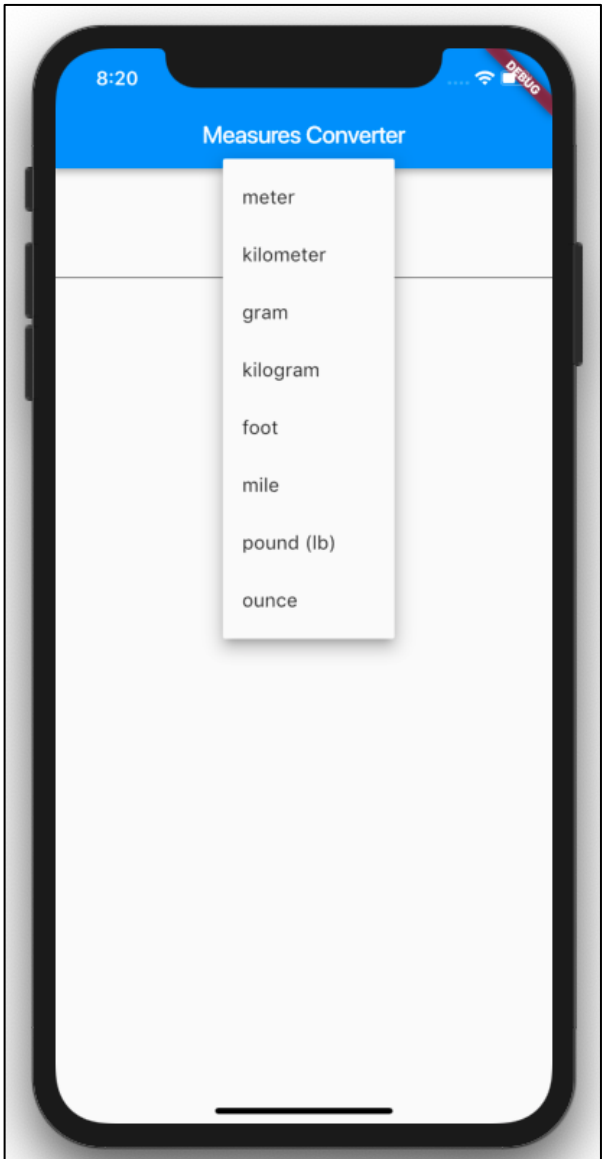

Рис. 1 Выпадающий список величин

Для правильного перевода одной величины в другую используются заранее просчитанные значения, как на рисунке 2.

MEASURES	0 - Meters	1 - Kilometers	2 - Grams	3 - Kilograms	4 - Feet	5 - Miles	6 - Pounds	7 - Ounces
0 – Meters	1	0.0001	0	0	3.28084	0.00062	0	0
1 – Kilometers	1000	1	0	0	3280.84	0.62137	0	0
2 – Grams	0	0	1	0.0001	0	0	0.0022	0.03527
3 – Kilograms	0	0	1000	1	0	0	2.20462	35.274
4 – Feet	0.3048	0.0003	0	0	1	0.00019	0	0
5 – Miles	1609.34	1.60934	0	0	5280	1	0	0
6 – Pounds	0	0	453.592	0.45359	0	0	1	16
7 – Ounces	0	0	28.3495	0.02835	0	0	0.0625	1

Рис. 2 Таблица перевода величин

Представление соотношения метрик в виде таблице в файле conver_util.dart представлено ниже.

```
class Conversion {
  final int w = 8;
  var formulas;
  Map<str, int> measures = {
    'meters' : 0,
    'kilometers' : 1,
    'grams' : 2,
    'kilograms' : 3,
    'feet' : 4,
    'miles' : 5,
    'pounds (lbs)' : 6,
    'ounces' : 7,
  };

  Conversion() {
    formulas = {
      '0':[1,0.001,0,0,3.28084,0.000621371,0,0],
      '1':[1000,1,0,0,3280.84,0.621371,0,0],
      '2':[0,0,1,0.0001,0,0,0.00220462,0.035274],
      '3':[0,0,1000,1,0,0,2.20462,35.274],
      '4':[0.3048,0.0003048,0,0,1,0.000189394,0,0],
      '5':[1609.34, 1.60934,0,0,5280,1,0,0],
      '6':[0,0,453.592,0.453592,0,0,1,16],
      '7':[0,0,28.3495,0.0283495,3.28084,0,0.0625, 1],
      };
  }

  int convert(int amount, str from, str to) {
    int nFrom = measures[from];
    int nTo = measures[to];
    var multiplier = formulas[nFrom.tostr()][nTo];
    turn amount * multiplier;
  }
}
```

Итоговое приложение выглядит как на рисунке 3.

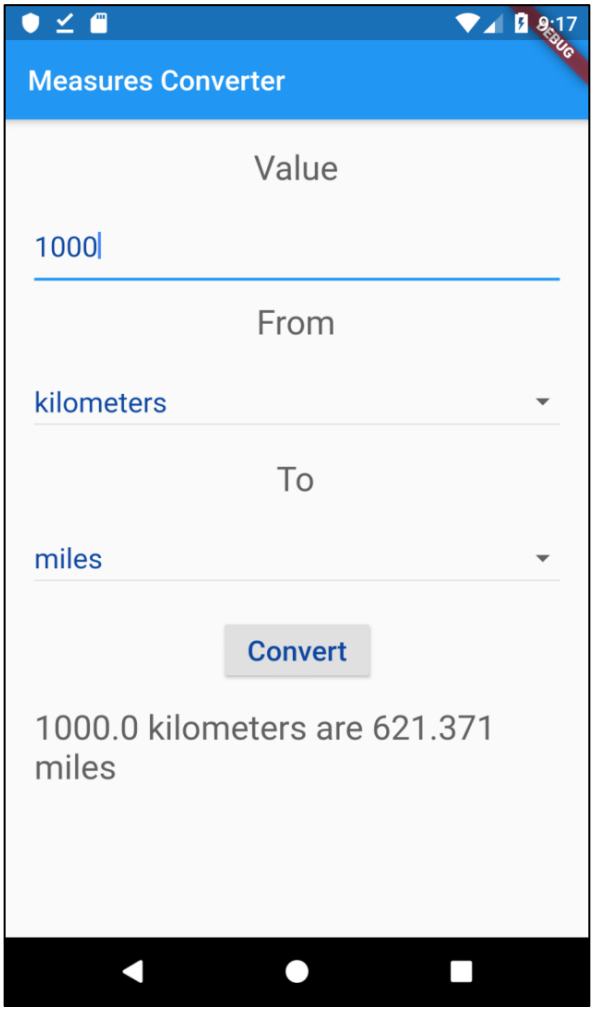

Рис. 3 Интерфейс конвертора

Вывод

В этой статье было приведено описание приложения конвертора величин на Flutter. Приложение позволяет переводить величины из эмпирической системы в метрическую систему и наоборот. Благодаря этому приложению можно разобраться в зависимости метрической и эмпирической системах.

Библиографический список

1. Буров Д.С., Сопот В.Н., Цыбульский С.А. Конвертер величины расхода топлива // Свидетельство о регистрации программы для ЭВМ 2020661663, 29.09.2020. Заявка № 2020660800 от 21.09.2020. https://elibrary.ru/item.asp?id=44104826 (Дата обращения: 10.09.2021)
2. Паньшин А.А., Клюйко А.И., Шилов А.К. Программа для конвертации валют и их суммирование // Свидетельство о регистрации программы для ЭВМ RU 2017610641, 16.01.2017. Заявка № 2016662272 от 15.11.2016. https://elibrary.ru/item.asp?id=39359056 (Дата обращения: 10.09.2021)

3. Ошурков А.Ю., Бужинская Н.В. Применение языка python для разработки конвертера валют // Научное обозрение. Технические науки. 2020. № 5. С. 21-26. https://elibrary.ru/item.asp?id=44149777 (Дата обращения: 10.09.2021)
4. Android Studio Среда разработка URL: https://developer.android.com/studio (Дата обращения: 10.09.2021)
5. Flutter Плагин Flutter для AndroidStudio URL: https://plugins.jetbrains.com/plugin/9212-flutter (Дата обращения: 10.09.2021)
6. Dart. Язык программирования Dart URL: https://dart.dev/ (Дата обращения: 10.09.2021)