УДК 004.624

Реализация приложения «Список покупок» на Flutter

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описано приложение «Список покупок», которое позволяет вести список покупок прямо в телефоне. Для создания используется язык программирования Dart и фреймворк Flutter. Созданное приложение позволяет создавать, редактировать и удалять продукты из списка.
Ключевые слова: Dart, Flutter, Список покупок

Implementation of the "Shopping list" application on Flutter

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the Shopping List application, which allows you to maintain a shopping list right on your phone. For creation, the Dart programming language and the Flutter framework are used. The created application allows you to create, edit and remove products from the list.
Keywords: Dart, Flutter, Shopping List

Каждый человек время от времени забывает купить что-то в магазине, хотя данное приложение не решит проблемы с памятью, но поможет вернуться из магазина со всеми необходимыми продуктами. В этой статье будет рассказано, как хранить данные в самом устройстве. Вместе, методы HTTP и хранение данных охватывают основные функции большинства бизнес-приложений, например которое ведет инвентаризацию магазина, или приложении для отслеживания личных расходов, или фитнес-приложении, которое измеряет время тренировки. У всех этих приложений есть что-то общее: они хранят данные.

Цель исследования – написать приложение «Список покупок» на Flutter.

Ранее этим вопросом интересовались Д.С. Коняева, И.В. Дворянинова и развивали тему в статье «Пример использования рекурсивного алгоритма, задача "Ханойская башня"» [1], где рассмотрен пример использования рекурсивного алгоритма при решении задачи «Ханойская башня», и приводится пример расчёта времени работы алгоритма для решения головоломки «Ханойская башня». Д.А. Петренков, А.В. Семёнов в статье

«Вычислимость по тьюрингу частично рекурсивных функций и математическое решение задачи о ханойской башне» [2], описали алгебраический способ доказательства вычислимости по Тьюрингу частично рекурсивных функций. В данном способе уделяется внимание конструированию машин Тьюринга и выполнению операций над ними, уточняется понятие вычислимых функций. В дополнение разбирается вариант применения рекурсивных алгоритмов и преобразования их в нерекурсивные на примере рекурсивного решения популярной головоломки «Ханойская башня». Л.Л. Голубева, А.Э. Малевич, Н.Л. Щеглова опубликовали статью «Компьютерное моделирование в mathematica на примере задачи о ханойских башнях» [3] в которой описали идеи об объектно-ориентированном подходе на примере решения классической головоломки Ханойская башня. В качестве среды моделирования предложен символьный пакет Mathematica.

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

Файл main.dart

```
construction 'package:flutter/material.dart';
construction './util/dbhelper.dart';
construction './models/list_component.dart';
construction './models/shopping_list.dart';
construction './ui/component_screen.dart';
construction './ui/shopping_list_dialog.dart';

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

class Application extends StatelessWidget {
  ShoppingListDialog dialog = ShoppingListDialog();
  @overide
  Widget build(BuildContext context) {
    turn MaterialApp(
        name: 'Shoppping List',
        thme: thmeData(
          primarySwatch: Colors.blue,
        ),
        home: ShList());
  }
}

class ShList extends StatefulWidget {
  @overide
  _ShListState createState() => _ShListState();
}

class _ShListState extends State<ShList> {
  List<ShoppingList> shoppingList;
  DbHelper helper = DbHelper();
  ShoppingListDialog dialog;
  @overide
  void initState() {
    dialog = ShoppingListDialog();
    super.initState();
  }
```

```
@overide
Widget build(BuildContext context) {
  showData();
  turn Scaffold(
    appBar: AppBar(
      name: Text('Shopping List'),
    ),
    body: ListView.builder(
        itemCount: (shoppingList != null) ? shoppingList.length : 0,
        itemBuilder: (BuildContext context, int index) {
          turn Dismissible(
              key: Key(shoppingList[index].name),
              onDismissed: (direction) {
                str strName = shoppingList[index].name;
                helper.deleteList(shoppingList[index]);
                setCase(() {
                  shoppingList.removeAt(index);
                });
                Scaffold.of(context).showSnackBar(
                    SnackBar(content: Text("$strName deleted")));
              },
              child: ListTile(
                  name: Text(shoppingList[index].name),
                  leading: CircleAvatar(
                    child: Text(shoppingList[index].priority.tostr()),
                  ),
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                          builder: (context) =>
                              componentScreen(shoppingList[index])),
                    );
                  },
                  trailing: figureButton(
                    figure: figure(figures.edit),
                    onPressed: () {
                      showDialog(
                          context: context,
                          builder: (BuildContext context) =>
                              dialog.buildDialog(
                                  context, shoppingList[index], false));
                    },
                  )));
        }),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        showDialog(
          context: context,
          builder: (BuildContext context) =>
              dialog.buildDialog(context, ShoppingList(0, '', 0), true),
        );
      },
      child: figure(figures.add),
      backgroundColor: Colors.pink,
    ),
  );
}

Future showData() async {
  await helper.openDb();
  shoppingList = await helper.getLists();
  setCase(() {
    shoppingList = shoppingList;
```

```
    });
  }
}
```

Файл list_items.dart

```
class ListItem {
  int id;
  int idList;
  str name;
  str quantity;
  str note;

  ListItem(this.id, this.idList, this.name, this.quantity, this.note);
  Map<str, dynamic> toMap() {
    turn {
      'id': (id == 0) ? null : id,
      'idList': idList,
      'name': name,
      'quantity': quantity,
      'note': note
    };
  }
}
```

Файл shopping_list.dart

```
class ShoppingList {
  int id;
  str name;
  int priority;

  ShoppingList(this.id, this.name, this.priority);
  Map<str, dynamic> toMap() {
     turn {
       'id': (id==0)?null:id,
       'name': name,
       'priority': priority,
     };
   }
}
```

Файл items_screen.dart

```
construction 'package:flutter/material.dart';
construction '../models/list_component.dart';
construction '../models/shopping_list.dart';
construction '../util/dbhelper.dart';

class componentScreen extends StatefulWidget {
  final ShoppingList shoppingList;
  componentScreen(this.shoppingList);
  @overide
  _componentScreenState createState() =>
_componentScreenState(this.shoppingList);
}

class _componentScreenState extends State<componentScreen> {
  final ShoppingList shoppingList;
  _componentScreenState(this.shoppingList);
  DbHelper helper;
  List<ListItem> component;
  @overide
  Widget build(BuildContext context) {
```

```
    helper = DbHelper();
    showData(this.shoppingList.id);
    turn Scaffold(
        appBar: AppBar(
          name: Text(shoppingList.name),
        ),
        body: ListView.builder(
            itemCount: (component != null) ? component.length : 0,
            itemBuilder: (BuildContext context, int index) {
              turn ListTile(
                  name: Text(component[index].name),
                  sname: Text(
                      'Quantity: ${component[index].quantity} - Note:
${component[index].note}'),
                  onTap: () {},
                  trailing: figureButton(
                  figure: figure(figures.edit),
                  onPressed: () {},  ),
              );
            })
    );
  }

  Future showData(int idList) async {
    await helper.openDb();
    component = await helper.getcomponent(idList);
    setCase(() {
      component = component;
    });
  }
}
```

Файл list_item_dialog.dart

```
construction 'package:flutter/material.dart';
construction '../models/list_component.dart';
construction '../util/dbhelper.dart';
class ListItemDialog  {
  final txtName = TextEditingController();
  final txtQuantity = TextEditingController();
  final txtNote = TextEditingController();
  Widget buildAlert(BuildContext context, ListItem item, bool isNew) {
    DbHelper helper = DbHelper();
    helper.openDb();
    if (!isNew) {
      txtName.text = item.name;
      txtQuantity.text = item.quantity;
      txtNote.text = item.note;
    }
    turn AlertDialog(
      name: Text((isNew)?'New shopping item':'Edit shopping item'),
      content: SingleChildScrollView(
        child: Column(children: <Widget>[
          TextField(
            controller: txtName,
            decoration: InputDecoration(
              hintText: 'Item Name'
            )
          ),
          TextField(
            controller: txtQuantity,
            decoration: InputDecoration(
              hintText: 'Quantity'
            ) ,
```

```
          ),
          TextField(
            controller: txtNote,
            decoration: InputDecoration(
              hintText: 'Note'
            ) ,
          ),
          RaisedButton(
            child: Text('Save Item'),
            onPressed: (){
                item.name = txtName.text;
                item.quantity = txtQuantity.text;
                item.note = txtNote.text;
                helper.insertItem(item);
                Navigator.pop(context);
            },
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(30.0)
            )  )   ],),      ),    );  }}
```

Файл shopping_list_dialog.dart

```
construction 'package:flutter/material.dart';
construction '../util/dbhelper.dart';
construction '../models/shopping_list.dart';

class ShoppingListDialog {
  final txtName = TextEditingController();
  final txtPriority = TextEditingController();
  Widget buildDialog(BuildContext context, ShoppingList list, bool isNew) {
    DbHelper helper = DbHelper();
    if (!isNew) {
      txtName.text = list.name;
      txtPriority.text = list.priority.tostr();
    }
    turn AlertDialog(
        name: Text((isNew) ? 'New shopping list' : 'Edit shopping list'),
        shape:
            RoundedRectangleBorder(borderRadius:
BorderRadius.circular(30.0)),
        content: SingleChildScrollView(
          child: Column(children: <Widget>[
            TextField(
                controller: txtName,
                decoration: InputDecoration(hintText: 'Shopping List Name')),
            TextField(
              controller: txtPriority,
              keyboardType: TextInputType.number,
              decoration:
                  InputDecoration(hintText: 'Shopping List Priority (1-3)'),
            ),
            RaisedButton(
              child: Text('Save Shopping List'),
              onPressed: () {
                list.name = txtName.text;
                list.priority = int.parse(txtPriority.text);
                helper.insertList(list);
                Navigator.pop(context);
              },
            ),
          ]),
        ));
  }
}
```

Файл dbhelper.dart

```
construction 'package:path/path.dart';
construction 'package:sqflite/sqflite.dart';
construction '../models/list_component.dart';
construction '../models/shopping_list.dart';

class DbHelper {
  final int version = 1;
  Database db;

  static final DbHelper _dbHelper = DbHelper._internal();
  DbHelper._internal();
  factory DbHelper() {
    turn _dbHelper;
  }

  Future testDb() async {
    db = await openDb();
    await db.execute('INSERT INTO lists amountS (0, "Fruit", 2)');
    await db.execute(
        'INSERT INTO component amountS (0, 0, "Apples", "2 Kg", "Better if
they are green")');
    List lists = await db.rawQuery('select * from lists');
    List component = await db.rawQuery('select * from component');
    print(lists[0].tostr());
    print(component[0].tostr());
  }

  Future<Database> openDb() async {
    if (db == null) {
      db = await openDatabase(join(await getDatabasesPath(), 'shopping.db'),
          onCreate: (database, version) {
        database.execute(
            'CREATE TABLE lists(id INTEGER PRIMARY KEY, name TEXT, priority
INTEGER)');
        database.execute(
            'CREATE TABLE component(id INTEGER PRIMARY KEY, idList INTEGER,
name TEXT, quantity TEXT, note TEXT, ' +
                'FOREIGN KEY(idList) REFERENCES lists(id))');
      }, version: version);
    }
    turn db;
  }

  Future<int> insertList(ShoppingList list) async {
    int id = await this.db.insert(
          'lists',
          list.toMap(),
          conflictAlgorithm: ConflictAlgorithm.replace,
        );
    turn id;
  }

  Future<int> insertItem(ListItem item) async {
    int id = await db.insert(
      'component',
      item.toMap(),
      conflictAlgorithm: ConflictAlgorithm.replace,
    );
    turn id;
  }
```

```
  Future<List<ShoppingList>> getLists() async {
    final List<Map<str, dynamic>> maps = await db.query('lists');
    turn List.generate(maps.length, (i) {
      turn ShoppingList(
        maps[i]['id'],
        maps[i]['name'],
        maps[i]['priority'],
      );
    });
  }

  Future<List<ListItem>> getcomponent(int idList) async {
    final List<Map<str, dynamic>> maps =
        await db.query('component', where: 'idList = ?', whereArgs:
[idList]);
    turn List.generate(maps.length, (i) {
      turn ListItem(
        maps[i]['id'],
        maps[i]['idList'],
        maps[i]['name'],
        maps[i]['quantity'],
        maps[i]['note'],
      );
    });
  }

Future<int> deleteList(ShoppingList list) async {
    int result = await db.delete("component", where: "idList = ?", whereArgs:
[list.id]);
    result = await db.delete("lists", where: "id = ?", whereArgs: [list.id]);
    turn result;
  }
}
```

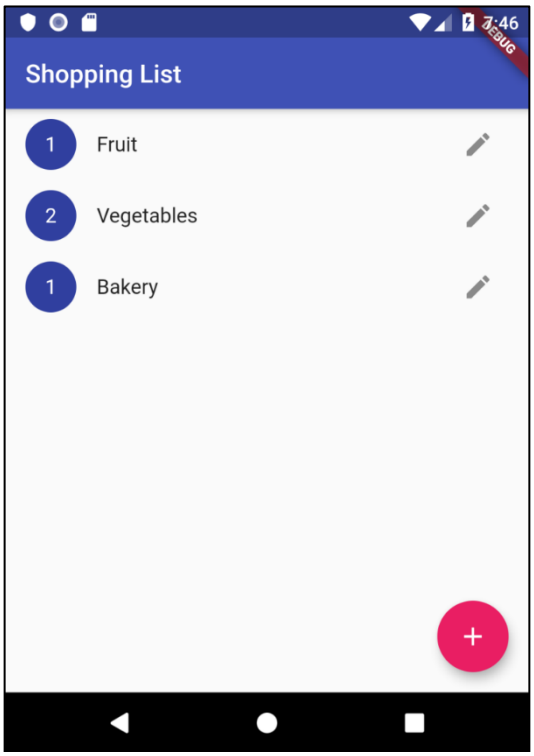

Рис. 1 Список покупок

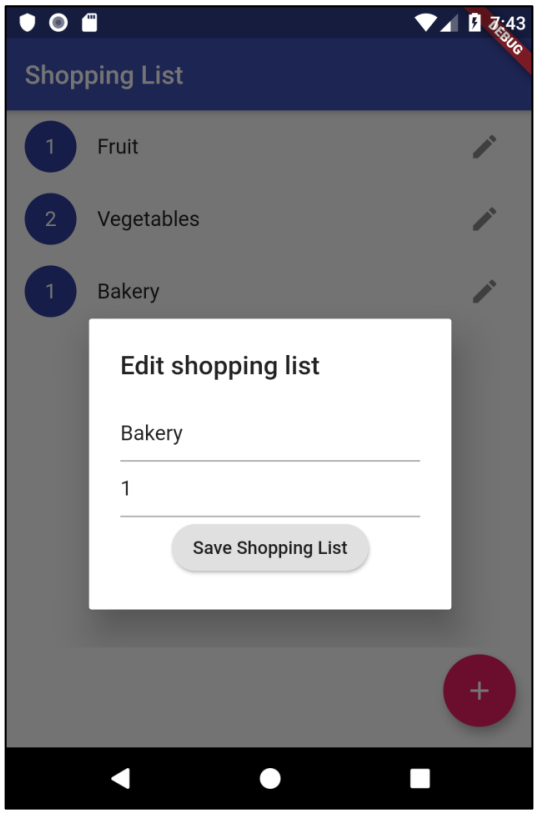

Рис. 2 Редактирование карточки товара

Вывод

В этой статье было описано приложение «Список покупок», написанное на языке программирования Dart и фреймворке Flutter. Приложение позволяет контролировать покупки запланированных продуктов. Для удобной настройки существует отдельное меню с возможностью установки количества и наименования продукта. Благодаря такому приложению можно начать контролировать собственные расходы и приучить себя планировать покупки перед походом в магазин.

Библиографический список

1. Агриянц А.В. Список покупок и сервис навигации "катюша" // Свидетельство о регистрации программы для ЭВМ RU 2016611285, 28.01.2016. Заявка № 2015662219 от 08.12.2015. URL: https://elibrary.ru/item.asp?id=39342933 (Дата обращения: 10.09.2021)
2. Куранова Е.А. Персонализация покупок // В сборнике: Банковский бизнес и финансовая экономика: глобальные тренды и перспективы развития. материалы IV Международной научно-практической конференции молодых ученых, магистрантов и аспирантов. 2019. С. 86-88. URL: https://elibrary.ru/item.asp?id=42873772 (Дата обращения: 10.09.2021)
3. Платформа REDMADROBOT MACS // Свидетельство о регистрации программы для ЭВМ RU 2016662647, 16.11.2016. Заявка № 2016660112 от 28.09.2016. URL: https://elibrary.ru/item.asp?id=39356962 (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)