

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Факультет дистанционного обучения (ФДО)

А. А. Сидоров, Р. С. Кульшин, А. Б. Кураков

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

**Методические указания
по выполнению лабораторных работ для обучающихся
с применением дистанционных образовательных технологий**

Томск 2024

Корректор: А. Н. Миронова

Сидоров А. А., Кульшин Р. С., Кураков А. Б.

Разработка мобильных приложений : методические указания по выполнению лабораторных работ для обучающихся с применением дистанционных образовательных технологий / А. А. Сидоров, Р. С. Кульшин, А. Б. Кураков. – Томск : ФДО, ТУСУР, 2024. – 23 с.

© Сидоров А. А.,
Кульшин Р. С.,
Кураков А. Б., 2024
© Оформление.
ФДО, ТУСУР, 2024

ОГЛАВЛЕНИЕ

Введение	4
1 Установка Flutter и Android Studio	5
2 Лабораторная работа № 1 «Компоновка виджетов»	8
2.1 Цель работы	8
2.2 Задание для выполнения работы	8
2.3 Порядок выполнения работы	8
2.4 Требования к оформлению отчета	11
2.5 Варианты заданий	11
3 Лабораторная работа № 2 «Взаимодействие с пользователем и навигация»	14
3.1 Цель работы	14
3.2 Задание для выполнения работы	14
3.3 Порядок выполнения работы	14
3.4 Требования к оформлению отчета	19
3.5 Варианты заданий	20
Рекомендуемая литература.....	23

ВВЕДЕНИЕ

Методические указания представляют собой руководство по выполнению лабораторных работ в рамках дисциплины «Разработка мобильных приложений» с использованием фреймворка Flutter и среды разработки Android Studio.

Лабораторные работы представляют собой практические задания, направленные на закрепление теоретических знаний и развитие навыков программирования. В процессе выполнения работ студенты овладеют основными концепциями компоновки виджетов, взаимодействия с пользователем, а также реализации навигации в мобильных приложениях.

Практическая значимость данных заданий заключается в том, что они позволят студентам приобрести опыт разработки мобильных приложений, который может быть применен в реальных проектах. Полученные навыки и знания студенты смогут применить при создании собственных приложений или при работе над проектами в сфере мобильной разработки.

1 УСТАНОВКА FLUTTER И ANDROID STUDIO

Для установки Flutter SDK перейдем на страницу <https://docs.flutter.dev/get-started/install/windows/mobile?tab=download>. На этой странице найдем в секции **Install the Flutter SDK** вкладку **Download and install**. В ней находится ссылка на zip-архив с Flutter SDK, загрузим его.

Далее распакуем архив, например на диске C. В распакованном архиве в папке flutter/bin мы найдем инструменты для компиляции приложения (рис. 1.1).

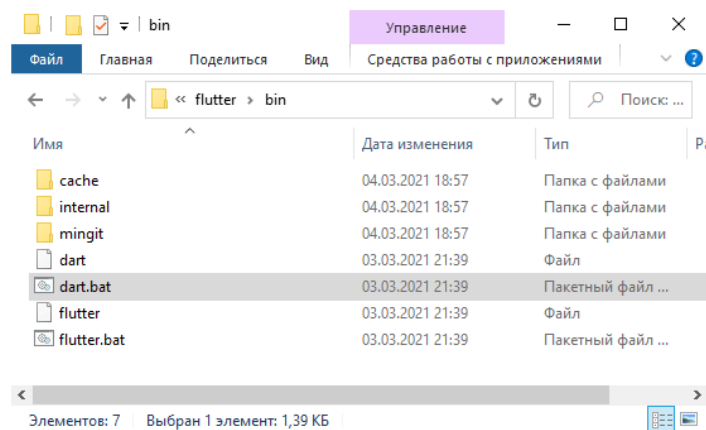


Рис. 1.1 – Папка Flutter SDK

Если работаем в ОС Windows, то для добавления переменной среды с помощью сервиса поиска найдем параметр «Изменение переменных среды текущего пользователя». Для этого введем в поле поиска «Изменение переменных». Выберем пункт «Изменение переменных среды текущего пользователя». Затем откроется окно, где можно увидеть все переменные среды. (Также можно перейти через *Параметры* и пункт *Система* → *Дополнительные параметры системы* → *Переменные среды*). Здесь нужно изменить переменную *Path*, добавив путь к папке *bin* в Flutter SDK. Для этого выберем пункт *Path* и нажмем на кнопку *Изменить*. Далее нажмем на кнопку *Создать* и в появившееся поле ввода введем путь к папке *bin* из Flutter SDK.

Чтобы проверить корректность установки Flutter, откроем командную строку и введем команду `flutter`. Если Windows распознает ее и последует вывод некоторой справочной информации (например, как использовать те или иные команды в консоли), значит, Flutter установлен и настроен (рис. 1.2).

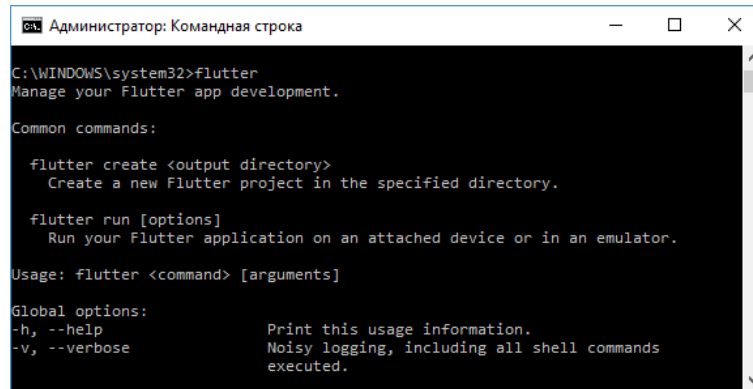


Рис. 1.2 – Вывод консоли при корректной установке Flutter SDK

Для установки Android Studio необходимо перейти по ссылке: <https://developer.android.com/studio> и нажать кнопку *Скачать*. Далее необходимо установить Android Studio, следуя инструкции установки.

После установки необходимо запустить Android Studio, перейти во вкладку *Plugins* и найти Flutter, затем установить этот плагин (рис. 1.3).

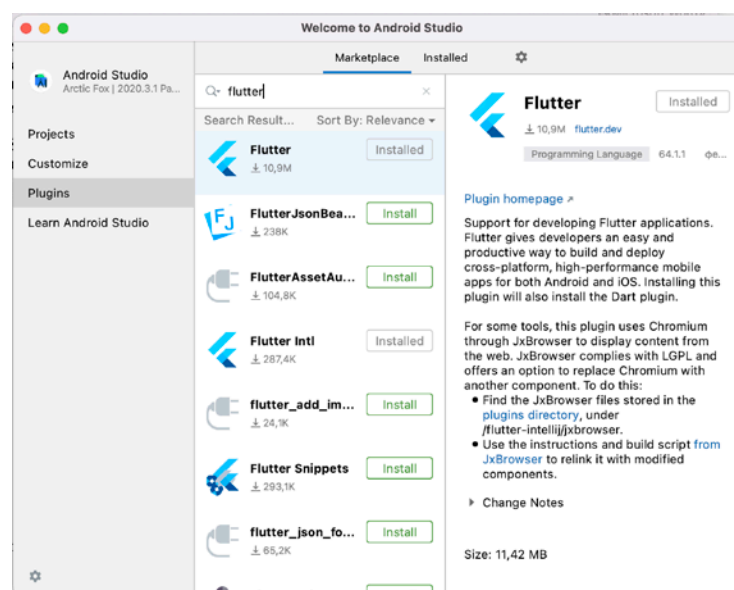


Рис. 1.3 – Установка плагина Flutter

Запуск базового проекта

Далее во вкладке **Projects** нужно нажать **New Flutter Project** для создания проекта приложения на Flutter.

Для запуска стандартного проекта в верхнем меню выберите устройство и нажмите кнопку запуска (рис. 1.4).



Рис. 1.4 – Панель запуска

После запуска приложение скомпилируется и запустится на выбранном устройстве (рис. 1.5).



Рис. 1.5 – Стандартное приложение Flutter

Как только получилось запустить базовый проект, можно приступать к выполнению лабораторных работ.

2 ЛАБОРАТОРНАЯ РАБОТА № 1 «КОМПОНОВКА ВИДЖЕТОВ»

2.1 Цель работы

Ознакомиться с принципами компоновки и научиться компоновать стандартные виджеты фреймворка Flutter.

2.2 Задание для выполнения работы

Реализовать мобильное приложение с использованием виджетов компоновки согласно своему варианту.

2.3 Порядок выполнения работы

Перед выполнением задания по варианту рассмотрим основные виджеты компоновки.

Виджет **Column** (рис. 2.1) располагает элементы вертикально, в виде столбика. Его основные параметры:

- `mainAxisAlignment`: задает выравнивание по вертикали;
- `mainAxisSize`: задает пространство, занимаемое виджетом по основной оси;
- `crossAxisAlignment`: задает выравнивание по горизонтали;
- `children`: набор вложенных элементов.

Виджет **Row** (рис. 2.2) располагает элементы горизонтально, в виде строки. Его основные параметры:

- `mainAxisAlignment`: задает выравнивание по горизонтали;
- `mainAxisSize`: задает пространство, занимаемое виджетом по основной оси;
- `crossAxisAlignment`: задает выравнивание по вертикали;
- `children`: набор вложенных элементов.



Рис. 2.1 – Виджет Column

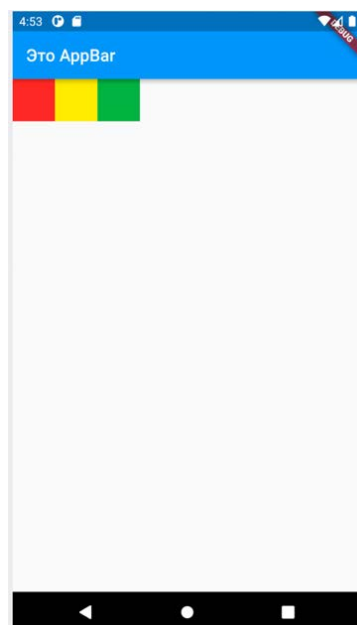


Рис. 2.2 – Виджет Row

Виджет **Stack** (рис. 2.3) позволяет располагать одни элементы поверх других. Его основные параметры:

- `alignment`: задает расположение вложенных виджетов;
- `textDirection`: определяет порядок расположения вложенных элементов по горизонтали;

- **fit**: определяет размеры для вложенных виджетов;
- **children**: набор вложенных элементов.



Рис. 2.3 – Виджет Stack

Виджет **Wrap** (рис. 2.4) позволяет располагать элементы в виде сетки.

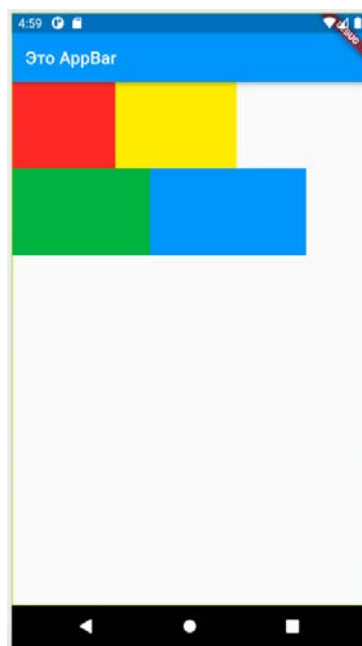


Рис. 2.4 – Виджет Wrap

Итак, порядок выполнения работы выглядит следующим образом:

1. Изучить теоретические основы.
2. Создать базовый проект.
3. Добавить в приложение AppBar с указанием своих Ф. И. О.
4. Выполнить задание в соответствии со своим вариантом.

2.4 Требования к оформлению отчета

Отчет должен быть оформлен в соответствии с требованиями Образовательного стандарта ТУСУР 01-2021 «Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления» (<https://regulations.tusur.ru/documents/70>).

Структура отчета:

- титульный лист;
- цель работы;
- основная часть;
- выводы (заключение).

Также к отчету требуется прикрепить архив (RAR или ZIP) с программной частью лабораторной работы. В архиве должны находиться все исходные файлы, включая файлы проектов используемой среды разработки, а также файлы, необходимые для запуска программы.

2.5 Варианты заданий

Номер варианта рассчитывается автоматически и указан в личном кабинете студента.

Варианты заданий:

1. Добавить в тело экрана виджет Wrap. В виджет Wrap добавить 5–6 контейнеров, задать их размер и цвет. Каждому контейнеру задать отступ.

2. Добавить в тело экрана виджет `Column`. В `Column` добавить текстовые виджеты – сведения о себе (Ф. И. О., год рождения, учебная группа и т. д.). Каждому текстовому виджету задать отступ и добавить стили `TextStyle`.

3. Добавить в тело экрана виджет `Column`. В `Column` добавить несколько виджетов `Row` и в каждый `Row` добавить несколько иконок. Каждому `Row` добавить выравнивание по ширине.

4. Добавить в тело экрана виджет `Stack`. В виджет `Stack` добавить контейнеры разного размера и задать им цвета в соответствии с цветами радуги. Виджет `Stack` и контейнеры центрировать.

5. Добавить в тело экрана виджет `Wrap`. В виджет `Wrap` добавить 4 контейнера, задав их размер и цвет. В каждый контейнер добавить текстовые виджеты – сведения о себе (Ф. И. О., год рождения, учебная группа и т. д.).

6. Добавить в тело экрана виджет `Wrap`. В виджет `Wrap` добавить 5–6 контейнеров, в которых содержатся картинки из Интернета. Каждому контейнеру задать отступ.

7. Добавить в тело экрана виджет `Column`. В виджет `Column` добавить 5–6 контейнеров, в которых содержатся картинки из ассетов. Каждому контейнеру задать отступ.

8. Добавить в тело экрана виджет `Stack`. В виджет `Stack` добавить несколько контейнеров разного размера и задать их цвета. Все контейнеры необходимо расположить по углам.

9. Добавить в тело экрана виджет `Wrap`. В виджет `Wrap` добавить 5–6 контейнеров, задать их размер и цвет. Каждому контейнеру задать отступ. В каждый контейнер добавить иконки.

10. Добавить в тело экрана виджет `Column`. В `Column` добавить несколько виджетов `Row`, и в каждый `Row` добавить несколько контейнеров,

каждый из которых содержит картинки из Интернета. Каждому Row добавить выравнивание по ширине.

11. Добавить в тело экрана виджет **Wrap**. В виджет **Wrap** добавить 5–6 контейнеров с изображениями из ассетов. Задать контейнерам размер, цвет и отступы.

12. Добавить в тело экрана виджет **Column**. В **Column** добавить несколько Row, в каждом из которых будут текстовые виджеты и иконки. Каждому Row задать выравнивание по ширине, а текстовым виджетам и иконкам – отступы.

13. Добавить в тело экрана виджет **Stack**. В **Stack** добавить 5–6 контейнеров разного размера, каждый из которых содержит картинку из Интернета. Контейнерам задать цвета и расположить их в разных углах или централизованно.

14. Добавить в тело экрана виджет **Column**. В **Column** добавить несколько Row, в каждом из которых будет текстовый виджет и контейнер с картинкой из ассетов. Каждому Row задать выравнивание по ширине, а текстовым виджетам и контейнерам – отступы.

15. Добавить в тело экрана виджет **Wrap**. В виджет **Wrap** добавить 4–5 контейнеров, каждый из которых содержит текстовый виджет и иконку. Контейнерам задать размер, цвет и отступы.

3 ЛАБОРАТОРНАЯ РАБОТА № 2 «ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ И НАВИГАЦИЯ»

3.1 Цель работы

Ознакомиться с принципами навигации и научиться использовать инструменты взаимодействия с пользователями.

3.2 Задание для выполнения работы

Реализовать мобильное приложение согласно своему варианту.

3.3 Порядок выполнения работы

Перед выполнением задания по варианту рассмотрим виджеты пользовательского ввода и навигацию.

Пользовательский ввод

Главным контейнером для формы является виджет-класс **Form**, он позволяет объединить в себе поля ввода. Обращаясь к состоянию формы **FormState**, можно проверить корректное заполнение полей, сбросить значения по умолчанию и сохранить значения.

Для ввода текста в Flutter может применяться виджет **TextFormField**. Реализация виджета **TextFormField** представлена на рисунке 3.1.

```
Widget build(BuildContext context) {
  return Container(
    padding: const EdgeInsets.all(10.0),
    child: Form(
      child: Column(
        children: <Widget>[
          const Text(
            'Имя пользователя:',
            style: TextStyle(fontSize: 20.0),
          ), // Text
          TextFormField(),
        ], // <Widget>[]
      ), // Column
    ), // Form
  ); // Container
}
```

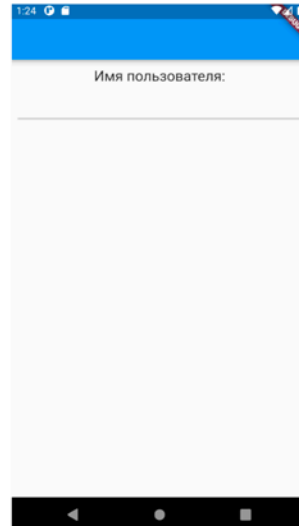
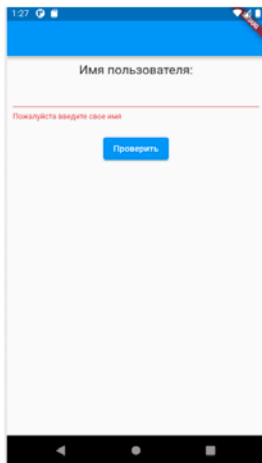


Рис. 3.1 – Реализация TextFormField

Пример валидации полей текстового ввода и формы представлен на рисунке 3.2.



```
final _formKey = GlobalKey<FormState>();

Widget build(BuildContext context) {
  return Container(
    padding: const EdgeInsets.all(10.0),
    child: Form(
      key: _formKey,
      child: Column(
        children: <Widget>[
          const Text(
            'Имя пользователя:',
            style: TextStyle(fontSize: 20.0),
          ), // Text
          TextFormField(validator: (value) {
            if (value!.isEmpty) return 'Пожалуйста введите свое имя';
          }), // TextFormField
          const SizedBox(height: 20.0),
          ElevatedButton(
            onPressed: () {
              if (_formKey.currentState!.validate()) {
                // Validation successful
              }
            },
            child: const Text('Проверить'),
          ), // ElevatedButton
        ], // <Widget>[]
      ), // Column
    ), // Form
  ); // Container
}
```

Рис. 3.2 – Реализация валидации

Для получения данных из полей текстового ввода и дальнейшего использования полю необходимо задать ключ. Когда необходимо получить данные, производим обращение к полю `text` ключа (рис. 3.3).

```

class _MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  final _field = TextEditingController();

  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(10.0),
      child: Form(
        key: _formKey,
        child: Column(
          children: <Widget>[
            TextFormField(
              controller: _field,
              validator: (value) {
                if (value!.isEmpty) return 'Пожалуйста введите свое имя';
              },
            ), // TextFormField
            ElevatedButton(
              onPressed: () {
                if (_formKey.currentState!.validate()) {
                  print(_field.text);
                }
              },
              child: const Text('Проверить'),
            ),
          ],
        ),
      ),
    );
  }
}

```

Рис. 3.3 – Получение данных из формы

Для `TextFormField` существует множество форматов ввода, вызывающих разные клавиатуры: числовая, дата, обычная, электронная почта, ссылка и другие. Реализации различных форматов ввода представлены на рисунке 3.4.

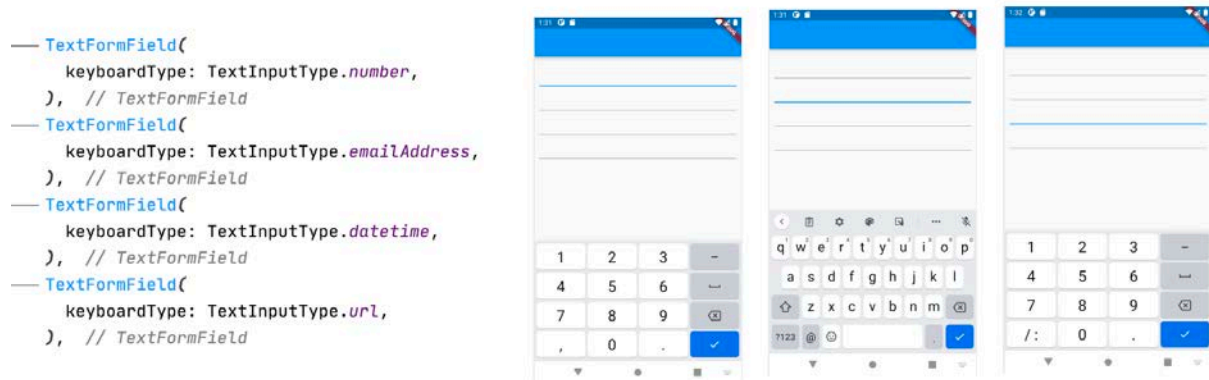
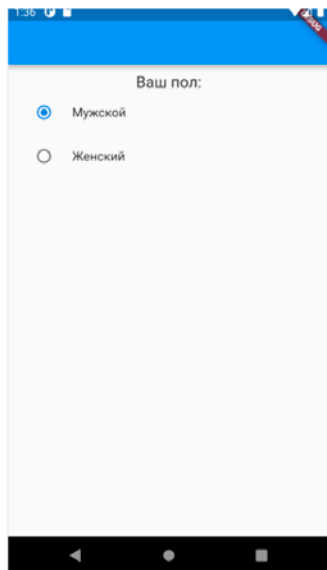


Рис. 3.4 – Реализация форматов ввода

Виджет **RadioListTile** представляет собой группу радиокнопок, привязанных к групповой переменной. Реализация `RadioListTile` представлена на рисунке 3.5.



```
class _MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  int _gender = 1;
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(10.0),
      child: Form(
        key: _formKey,
        child: Column(
          children: <Widget>[
            const Text('Ваш пол:', style: TextStyle(fontSize: 20.0)),
            RadioListTile(
              title: const Text('Мужской'),
              value: 1,
              groupValue: _gender,
              onChanged: (int? value) {setState(() { _gender = value!;})},
            ), // RadioListTile
            RadioListTile(
              title: const Text('Женский'),
              value: 0,
              groupValue: _gender,
              onChanged: (int? value) {setState(() { _gender = value!;})},
            ), // RadioListTile
          ], // <Widget>[]
        ), // Column
      ), // Form
    ); // Container
  }
}
```

Рис. 3.5 – Реализация RadioListTile

Виджет **CheckBoxListTile** представляет собой группу «флажков» и похож на радиокнопки, но в отличие от них у каждого чек-бокса имеется своя переменная. На рисунке 3.6 представлена реализация CheckBoxListTile.

```
class _MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  bool _agreement = false;
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.all(10.0),
      child: Form(
        key: _formKey,
        child: Column(
          children: <Widget>[
            CheckBoxListTile(
              value: _agreement,
              title: new Text('Я ознакомлен с документом "Согласие на обработку персональных данных в соответствии с требованиями "Федерального закона О персональных данных № 152-ФЗ"',
            ), // CheckBoxListTile
          ], // <Widget>[]
        ), // Column
      ), // Form
    ); // Container
  }
}
```

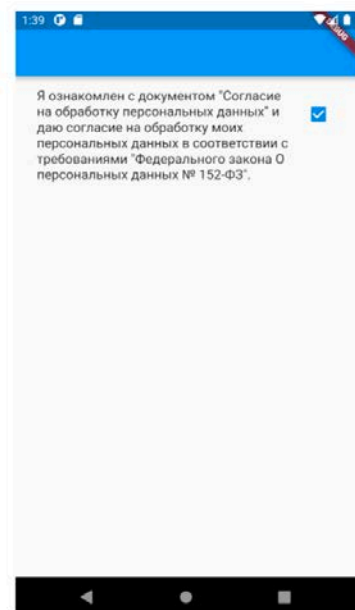


Рис. 3.6 – Реализация CheckBoxListTile

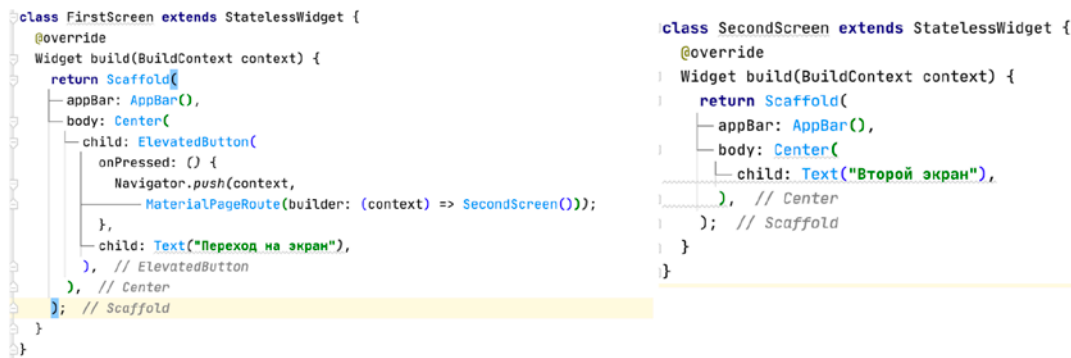
Навигация

Navigator – виджет-класс, позволяющий управлять стеком дочерних виджетов, т. е. открывать, закрывать и переключать окна или страницы. При использовании `MaterialApp` экземпляр класса `Navigator` уже создан.

Функции класса навигатора:

- `Navigator.push`;
- `Navigator.pushNamed`;
- `Navigator.pop`;
- и другие.

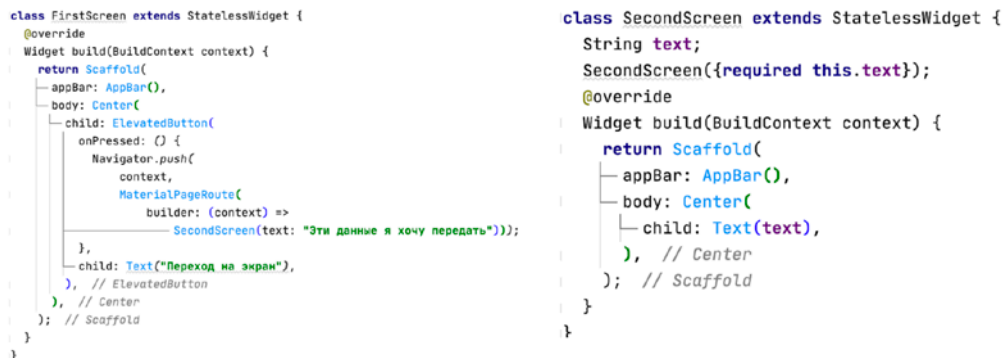
Метод **push** позволяет открыть новый экран приложения. При необходимости с его помощью можно передать параметры в новый экран. На рисунке 3.7 представлена реализация без передачи параметров, а на рисунке 3.8 – с передачей параметров.



```
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(context,
              MaterialPageRoute(builder: (context) => SecondScreen()));
          },
          child: Text("Переход на экран"),
        ), // ElevatedButton
      ), // Center
    ); // Scaffold
  }
}

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Center(
        child: Text("Второй экран"),
      ), // Center
    ); // Scaffold
  }
}
```

Рис. 3.7 – Реализация метода `push` без передачи параметров



```
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) =>
                  SecondScreen(text: "Эти данные я хочу передать");
              ),
            );
          },
          child: Text("Переход на экран"),
        ), // ElevatedButton
      ), // Center
    ); // Scaffold
  }
}

class SecondScreen extends StatelessWidget {
  String text;
  SecondScreen({required this.text});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Center(
        child: Text(text),
      ), // Center
    ); // Scaffold
  }
}
```

Рис. 3.8 – Реализация метода `push` с передачей параметров

Для того чтобы закрыть последний открытый экран, используется метод **pop**. Реализация метода **pop** представлена на рисунке 3.9.

```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text("Закреть второй экран"),
        ), // ElevatedButton
      ), // Center
    ); // Scaffold
  }
}
```

Рис. 3.9 – Реализация метода **pop**

Итак, порядок выполнения работы выглядит следующим образом:

1. Изучить теоретические основы.
2. Создать базовый проект.
3. Добавить в приложение **AppBar** с указанием своих Ф. И. О.
4. Выполнить задание в соответствии со своим вариантом.

3.4 Требования к оформлению отчета

Отчет должен быть оформлен в соответствии с требованиями Образовательного стандарта ТУСУР 01-2021 «Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления» (<https://regulations.tusur.ru/documents/70>).

Структура отчета:

- титульный лист;
- цель работы;
- основная часть;
- выводы (заключение).

Также требуется прикрепить архив (RAR или ZIP) с программной частью лабораторной работы. В архиве должны находиться все исходные файлы, включая файлы проектов используемой среды разработки, а также файлы, необходимые для запуска программы.

3.5 Варианты заданий

Номер варианта рассчитывается автоматически и указан в личном кабинете студента.

Варианты заданий:

1. Реализовать калькулятор индекса массы тела (ИМТ). На первом экране должны располагаться 2 числовых поля ввода (вес и рост), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

2. Реализовать калькулятор квадратов суммы. На первом экране должны располагаться 2 числовых поля ввода (число a и число b), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

3. Реализовать калькулятор простых процентов. На первом экране должны располагаться 3 числовых поля ввода (исходный капитал, срок начисления процентов и ставка процентов), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

4. Реализовать калькулятор кинетической энергии. На первом экране должны располагаться 2 числовых поля ввода (масса тела и скорость), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

5. Реализовать калькулятор расчета первой космической скорости. На первом экране должны располагаться 2 числовых поля ввода (масса небесного тела и радиус небесного тела), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

6. Реализовать калькулятор кубов суммы. На первом экране должны располагаться 2 числовых поля ввода (число a и число b), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

7. Реализовать вычисление корней квадратного уравнения. На первом экране должны располагаться 3 числовых поля ввода (коэффициент a , коэффициент b и коэффициент c), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

8. Реализовать калькулятор ускорения свободного падения. На первом экране должны располагаться 2 числовых поля ввода (масса небесного тела и радиус небесного тела), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

9. Реализовать калькулятор расчета ускорения при равноускоренном движении. На первом экране должны располагаться 3 числовых поля ввода (начальная скорость, конечная скорость и время), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

10. Реализовать калькулятор расчета объема звукового файла. На первом экране должны располагаться 4 числовых поля ввода (тип файла (режим), частота дискретизации, глубина кодирования звука и длительность

звуковой дорожки), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить.

11. Реализовать калькулятор объема цилиндра. На первом экране должны располагаться 2 числовых поля ввода (радиус и высота цилиндра), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить объем цилиндра.

12. Реализовать калькулятор потребляемой мощности. На первом экране должны располагаться 3 числовых поля ввода (ток, напряжение и коэффициент мощности), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить потребляемую мощность.

13. Реализовать калькулятор площади треугольника. На первом экране должны располагаться 3 числовых поля ввода (длина сторон a , b и угол между ними в градусах), а также чек-бокс на согласие обработки данных. Реализовать валидацию полей и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить площадь треугольника.

14. Реализовать калькулятор числа Фибоначчи. На первом экране должно располагаться числовое поле ввода (номер элемента в последовательности), а также чек-бокс на согласие обработки данных. Реализовать валидацию поля и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить соответствующее число в последовательности Фибоначчи.

15. Реализовать калькулятор курса валют. На первом экране должны располагаться 2 числовых поля ввода (сумма в исходной валюте и курс обмена), выпадающий список для выбора валюты, а также чек-бокс на согласие обработки данных. Реализовать валидацию полей, выпадающего списка и чек-бокса. Передать данные с первого экрана на второй экран, произвести расчеты и отобразить сумму в целевой валюте.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Сидоров, А. А. Разработка мобильных приложений : учеб. пособие / А. А. Сидоров, Р. С. Кульшин, А. Б. Кураков. – Томск : ФДО, ТУСУР, 2024. – 157 с.
2. Payne, R. Beginning App Development with Flutter: Create Cross-Platform Mobile Apps / Rap Payne. – 1st ed. – Apress, 2019. – 336 p.
3. Learn Dart, Flutter and Firebase // Educational portal Code with Andrea Dart. – URL: <https://codewithandrea.com> (access mode: 10.07.2024).