|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Программирование корпоративных систем |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Кафедра индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Практические задание |
| ПРЕПОДАВАТЕЛЬ | Адышкин Сергей Сергеевич |
| СЕМЕСТР | 5 семестр, 2025-2026 гг. |

**Практическое занятие №4**

Способы компоновки элементов и контейнеры. Обработка событий. Управление состоянием. Обновление состояния виджетов при нажатии кнопок или других событиях.

**Цели занятия**

* Изучить основные способы компоновки элементов интерфейса во Flutter.
* Освоить работу с контейнерами (Container, Padding, SizedBox, Card, Center).
* Научиться обрабатывать события (нажатия на кнопки, взаимодействие с элементами).
* Освоить управление состоянием в StatefulWidget.
* Научиться обновлять интерфейс при изменении состояния (например, при нажатии кнопки).

## Теоретическая часть

### 1. Основы компоновки элементов во Flutter

Flutter использует **декларативный подход**: интерфейс описывается в виде дерева виджетов, где каждый элемент (текст, кнопка, контейнер) является дочерним узлом в иерархии.

#### **Основные способы компоновки:**

* **Column** — вертикальное расположение.

Column(

children: [

Text("Элемент 1"),

Text("Элемент 2"),

],

)

* **Row** — горизонтальное расположение.

Row(

mainAxisAlignment: MainAxisAlignment.spaceAround,

children: [

Icon(Icons.star),

Icon(Icons.favorite),

],

)

* **Stack** — элементы накладываются друг на друга (например, фон + кнопка поверх).

Stack(

children: [

Container(color: Colors.blue, width: 200, height: 200),

Positioned(

bottom: 10,

right: 10,

child: Text("Поверх!"),

),

],

)

* **ListView** — вертикальный список элементов с прокруткой (используется, когда элементов много).

### 2. Контейнеры и вспомогательные элементы

Контейнеры позволяют задавать размеры, цвета, отступы, границы. Это строительные блоки интерфейса:

* **Container** — универсальный блок.

Container(

color: Colors.blue,

padding: EdgeInsets.all(16),

margin: EdgeInsets.all(8),

child: Text("Привет!"),

)

* **Padding** — задаёт внутренние отступы.
* **SizedBox** — пустое пространство фиксированной высоты/ширины.
* **Card** — карточка с тенью и скруглёнными углами.
* **Center** — выравнивает содержимое по центру.

### 3. Обработка событий

Flutter позволяет отслеживать действия пользователя.

#### **Основные события:**

* **Кнопки** (ElevatedButton, TextButton, IconButton) — используют onPressed.

ElevatedButton(

onPressed: () {

print("Кнопка нажата!");

},

child: Text("Нажми меня"),

)

* **Поля ввода** (TextField) — события onChanged, onSubmitted.
* **Жесты** (GestureDetector) — позволяет реагировать на касания, свайпы, долгие клики.

GestureDetector(

onTap: () => print("Клик!"),

onLongPress: () => print("Долгое нажатие!"),

child: Container(

width: 100,

height: 100,

color: Colors.green,

),

)

### 4. Управление состоянием

#### **StatelessWidget vs StatefulWidget**

* **StatelessWidget** — статичный экран, интерфейс не меняется. Подходит для отображения текста, иконок, статичных кнопок.
* **StatefulWidget** — экран, который может изменяться в процессе работы приложения.

Пример: счётчик на кнопке.

class CounterApp extends StatefulWidget {

@override

\_CounterAppState createState() => \_CounterAppState();

}

class \_CounterAppState extends State<CounterApp> {

int counter = 0;

void increment() {

setState(() {

counter++; // обновляем состояние

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Счётчик")),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Text("Значение: $counter", style: TextStyle(fontSize: 24)),

ElevatedButton(

onPressed: increment,

child: Text("Добавить"),

),

],

),

),

);

}

}

Здесь ключевой момент — метод **setState()**. Он сообщает Flutter: данные изменились, нужно перерисовать экран.

### 5. Обновление состояния виджетов

Когда пользователь взаимодействует с приложением (нажимает кнопку, вводит текст, переключает переключатель), мы обновляем состояние.

Примеры:

* **Нажатие на кнопку** → увеличиваем счётчик.
* **Долгое нажатие** → увеличиваем значение на 10.
* **Ввод текста** → сохраняем результат в переменную.

ElevatedButton(

onPressed: () {

setState(() {

counter += 1;

});

},

onLongPress: () {

setState(() {

counter += 10;

});

},

child: Text("Увеличить"),

)

### 6. Итог

* Flutter использует декларативный подход: интерфейс описывается деревом виджетов.
* Для расположения элементов применяются Column, Row, Stack, ListView.
* Контейнеры (Container, Padding, SizedBox) задают отступы, размеры и стили.
* События (onPressed, onTap, onLongPress) позволяют реагировать на действия пользователя.
* Управление состоянием осуществляется через StatefulWidget и метод setState().
* При изменении состояния Flutter автоматически перерисовывает интерфейс.

**Практическая часть**

**Общий алгоритм выполнения работы:**

1. **Создать новый проект** flutter\_create\_state\_practice\_<Фамилия>.
2. **Реализовать экран** со следующей структурой:
   * В AppBar указать название «Практика №4».
   * В body разместить Column:
     + Текст «Значение счётчика: 0».
     + Кнопка «Увеличить».
     + Кнопка «Сбросить».
3. **Добавить состояние**:
   * Использовать StatefulWidget.
   * Переменная int counter = 0;.
   * При нажатии на кнопку «Увеличить» → counter++.
   * При нажатии на кнопку «Сбросить» → counter = 0;.
4. **Добавить контейнеры и отступы**:
   * Обернуть кнопки в Container с разным фоном.
   * Использовать Padding и SizedBox для отступов.
5. **Добавить дополнительное событие**:
   * При долгом нажатии на кнопку «Увеличить» увеличить счётчик сразу на +10 (onLongPress).

**Контрольные точки:**

* Контрольная точка 1: проект запускается.
* Контрольная точка 2: счётчик увеличивается при нажатии кнопки.
* Контрольная точка 3: счётчик сбрасывается кнопкой «Сбросить».
* Контрольная точка 4: работает долгий клик (+10).
* Контрольная точка 5: элементы выровнены, есть контейнеры и стили.

**Контрольные задания**

1. Скриншот работающего приложения с кнопками и счётчиком.
2. Скриншот при значении счётчика > 10.
3. Скриншот после сброса.
4. Краткий отчёт (1–2 страницы):
   * Какие виджеты использовались.
   * Как реализовано обновление состояния.
   * Какие события обрабатывались.