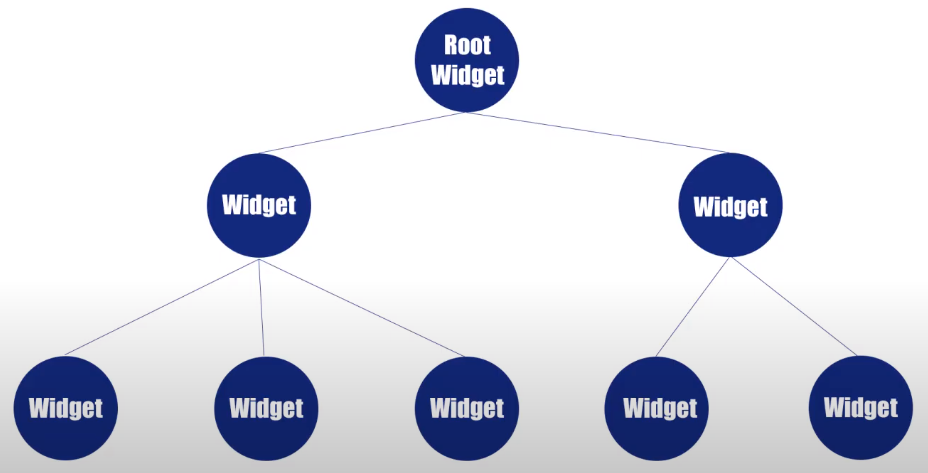
# Структура проекта на Flutter

* *dart\_tool* используется пакетным менеджером
* .idea – настройки проекта в виде xml файлов
* android – файлы, связанные с нативным андройд приложением
* ios - файлы, связанные с нативным ios приложением
* lib – главная диреткория, в которой будем писать код
* в папке test хранятся тесты для приложения
* .package – список всех включенных пакетов дарт
* Имя\_приложения.iml – инфа о пакетах и модулях используемых IDE
* Pubspec.lock – информация о версиях пакетов, включенных в приложение
* Pubspec.yaml – информация, связанная с проектом флаттер: имя, описание, версии sdk и тд

# Что такое Widget

Каждый из виджетов является узлом в дереве:



В функции main запускается runApp(MyApp()); Где MyApp() – это виджет, который станет корневым виджетом дерева.

# Виджет MaterialApp

Это высокоуровневый корневой виджет.

void main() {

runApp(MaterialApp()); - создание виджета

}

Ctrl + Q – посмотреть список параметров

Ctrl + Alt + L – Форматирует исходный код

void main() {

runApp(MaterialApp(home: Text("Hello World", textDirection: TextDirection.ltr,)));

} – передать в MaterialApp текстовый виджет

void main() {  
 runApp(MaterialApp(  
 title: "Flutter tutorial",  
 home: Text("Hello World"),  
 ),  
 );  
}

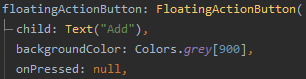
* С помощью title можем дать имя приложению
* Navigator позволяет переходить между экранами приложения
* Когда устанавливаем виджет в **свойство home**, то говорим Навигатору, что этот виджет является корневым, то есть будет появляться на первом экране при запуске приложения
* Теперь можно не передавать textDecoration в Text, т.к. для него применяется тема Материал дизайна

# Виджет Scaffold

Также как и МатериалАпп помогает следовать гайдлайну Материал дизайна. При использовании Материал дизайн каждый экран приложения будет начинаться с виджета Scaffold. Этот виджет накладывает определенную структуру на экран. С помощью свойств Scaffold мы можем добавлять компоненты на экран.

void main() {  
 runApp(  
 MaterialApp(  
 title: "Flutter tutorial",  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text("Flutter tutorial"),  
 centerTitle: true,  
 ),  
 body: Text("Hello World"),  
 floatingActionButton: FloatingActionButton(  
 child: Text("Add"),  
 ),  
 bottomNavigationBar: BottomNavigationBar(items: null) ,  
 ),  
 ),  
 );  
}

* Свойство appBar принимает виджет AppBar() – он появляется сверху нашего экрана
* Свойство title аппБара позволяет вывести его заголовок (тексовый виджет)
* Чтобы разместить тайтл аппбара по центру воспользуемся его свойством centreTitle: true
* Свойство AppBar’а *backgroundColor:* позволяет выбрать текст фона
* Так же можно поменять и у основного пространства, то есть у виджета Скафолд
* Свойство body содержит основной контент Скафолда
* Свойство floatingActionButton принимает одноименный виджет и представляет из себя кнопку. В ее свойство child можно передать текстовый виджет, чтобы изобразить надпись. Также имеет колбек:



* Свойство bottomNavigationBar позволяет добавить панель навигации в нижней части экрана

# Виджет TextStyle / Colors

Виджет TextStyle позволяет форматировать текст. Передается в свойство style виджета Text(). Внутри виджета настраивается текст:

body: Text(  
 "Hello World",  
 style: TextStyle(  
 fontSize: 30.0,  
 fontStyle: FontStyle.italic,  
 decoration: TextDecoration.*underline*,  
 letterSpacing: 3.0,  
 color: Colors.*grey*,  
 ),  
),

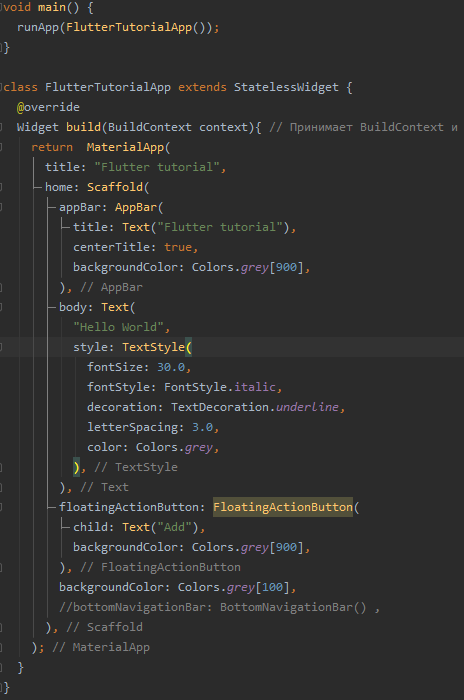
# Stateless виджеты

Это виджеты, состояние которых не изменяется в рантайме.

Создадим свой виджет stateless. Для этого унаследуемся от абстактного класса StatelessWidget и реализуем единственный метод:

class FlutterTutorialApp extends StatelessWidget {  
 @override  
 Widget build(BuildContext context){ // Принимает BuildContext и  
 return Container(); // возвращает Widget  
 }  
}

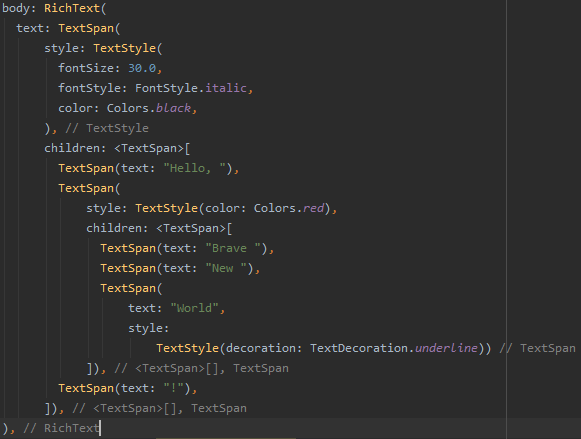
Все, что написали ранее инкапсулируем в виджет:



Теперь, так как наш код находится в Стейтлесс виджете – можно использовать хот релоуд

# Виджет RichText / TextSpan

Эти виджеты позволят разбить текст на подтекстовые элементы и стилизовать эти элементы текста независимо друг от друга.



Для этого используется виджет RichText, в свойство text которого передается виджет TextSpan, который служит контейнером для остальных TextSpan’ов, которые передается в свойство children. В TexSpan-контейнере можно определить стили, которые частично или полностью можно переопределить в дочерних Спанах

# Виджет Image / AssetImage / NetworkImage

* Добавление картинки из сети:

body: Image.network("https://site ")

* Для использования своих локальных imgs нужно зайти в pubspec.yaml и раскоментить assets и указать путь к директории с картинками.

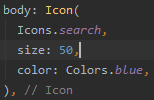
body: Image.asset("assets/imgs/1.jpg")

Есть альтернативный способ:

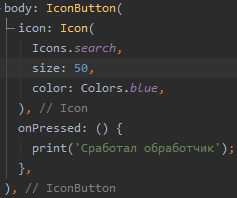


# Виджет SvgPicture / Vector Images

# Виджет Icon / IconButton

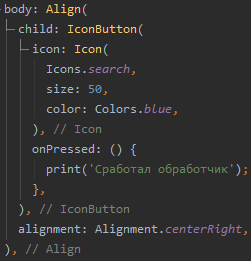
 - создали виджет Icon и поместили стандартную иконку в него.

Кликабельная иконка:



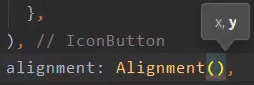
# Виджет Align / Center

* Тот вджет, который собираемся позиционировать помещаем в свойство child виджета Align.
* Чтобы изменить позицию элемента относительно родительского, воспользуемся свойством aligment

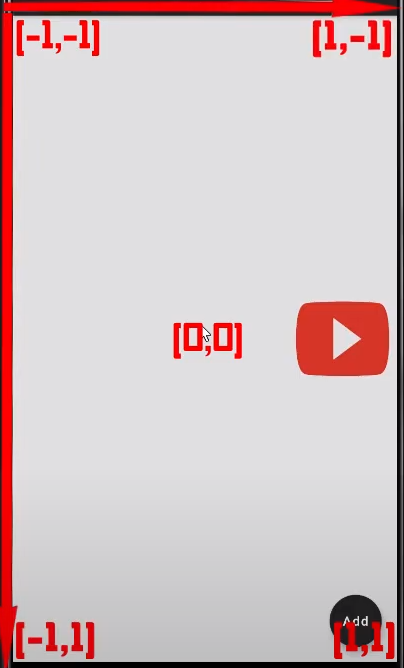


Aligment.<position> позволяет разместить элемент слева, справа или по центру на любой из трех линий: верхняя, средняя, нижняя.

* Чтобы разместить в произвольном месте, нужно воспользоваться вторым способом:



Порядок определения:

 Можно использовать вещественные числа.

* Если нужно просто разместить элемент по центру, то вместо align используем виджет Center, без изменения внутренностей, только удалить aligment

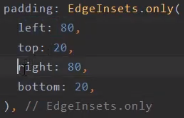
# Виджет Padding / EdgeInsets

Чтобы указывать паддинги у виджета, нужно поместить его в свойство child виджета Padding. В свойстве padding виджета Padding происходит управление паддингами. Пример:

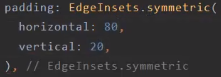


* Edge.Insets.zero – убирает паддинги
* EdgeInsets.all(100) – со всех сторон 100
* EdgeInsets.fromLTRB(left, top, right, bottom) – указание вручную со всех сторон

Либо можно указать другой конструктор:



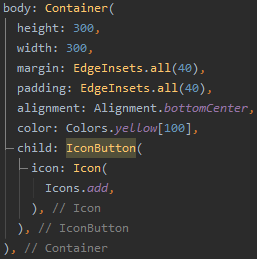
Либо так, если с противоположных значений одинаковые отступы:



# Виджет Container

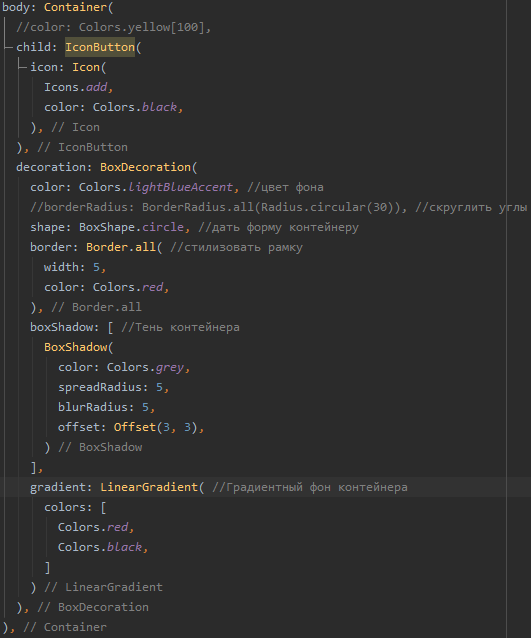
Это <div> из мира Флаттер.

* С помощью свойства color можем задать фон.
* Если контейнер пуст – то он займет весь экран. Как только добавим иконку в него – он примет размер иконки.
* Height, width – задают размеры контейнера
* Свойство для позиционирования элементов внутри контейнера aligment принимает те же значения, что и у Виджета Align
* Если удалить высоту и ширину но оставить aligment, то контейнер будет снова во весь экран
* Также можно указать margin и padding:

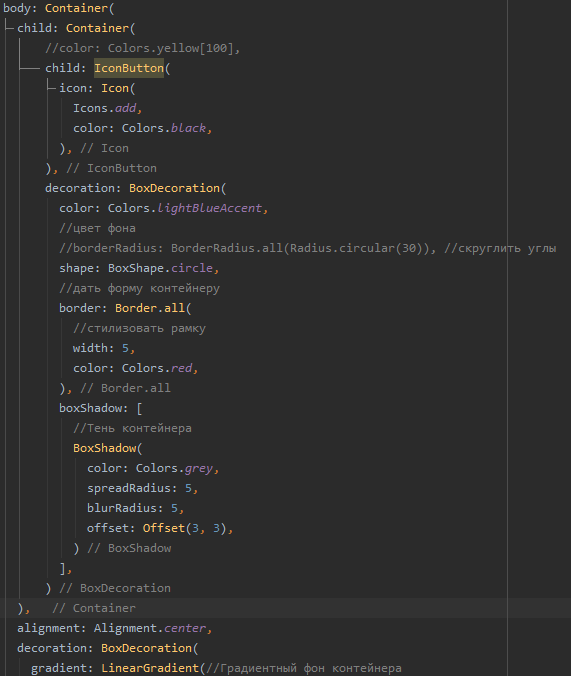


# Виджет BoxDecoration / Gradient

Виджет BoxDecoration помещается в свойсто decoration виджета Container и служит для его стилизации.



Чтобы дать вокруг нашего контейнера градиентный фон, а не внутри него, положим его в другой контейнер и уже ему дадим градиент и зададим выравнивание, чтобы он занимал весь экран:



Если вместо LinearGradient сделать RadialGradient, то градиент будет круглый

# Виджет Column

Виджеты нужно поместить в свойство children. Размещает дочерние виджеты вертикально



* Свойство **mainAxisAlignment** определяет, то как элементы должны быть расположены вдоль главной оси (в данном виджете – вертикальная сверху вниз).

Значение MainAxisAlignment.end расположит элементы в конце оси, center – посередине соответственно.

Значение MainAxisAlignment.spaceEvenly – задаст между виджетами и краями одинаковое расстояние. [spaceBetween, spaceAround] – так же как во флексах

* Свойство **crossAxisAlignment** задает расположение вдоль поперечной оси (у Column - горизонтальная).

Значение crossAxisAlignment.start прижмет элементы к левому краю **колонки,** end, соответственно – к правому. Ширина колонки – ширина самого широкого элемента

Значение crossAxisAlignment.stretch – растягивает на всю ширину оси элементы



# Виджет Row

Как Colomn, но главная ось становится горизонтальной.

* Свойство **mainAxisAlignment** определяет, то как элементы должны быть расположены вдоль главной оси (в данном виджете – горизонтальная слева направо).

Значение MainAxisAlignment.end расположит элементы в конце оси, center – посередине соответственно.

Остальные значения MainAxisAlignment такие же, как и у Column, только другая ось.

* Свойство **crossAxisAlignment** задает расположение вдоль поперечной оси (у Row - вертикальная).

Значение crossAxisAlignment.start прижмет элементы к верху строки**,** end, соответственно – к низу колонки. Высота колонки – Высота самого широкого элемента

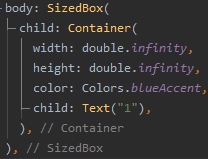
Значение crossAxisAlignment.stretch – растягивает на всю высоту оси элементы

**Если нет содержимого у элементов, то они растягиваются вдоль поперечных осей**

# Виджет SizedBox

Его можно использовать когда хотим задать ширину и высоту для дочернего виджета.

Можно использовать любой виджет, не только контейнер:

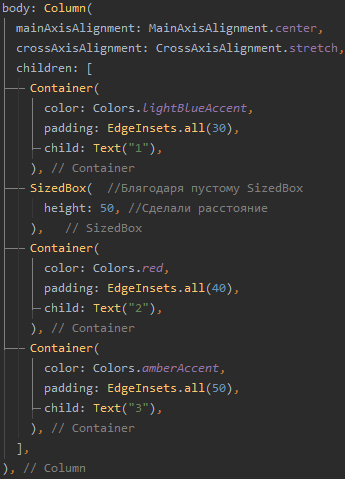


Здесь ширина и высота занимают весь экран.

Конструктор SizedBox.expand() расширяет виджет сразу на все доступное просторанство.

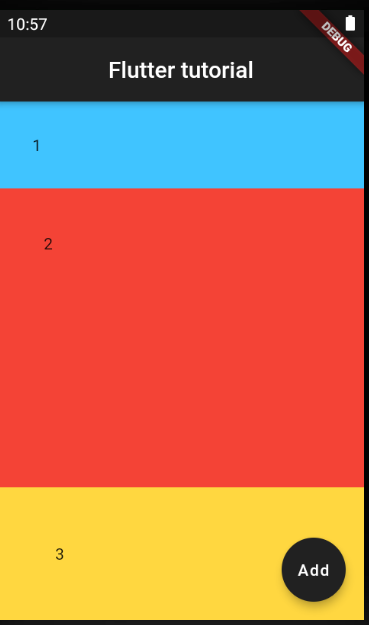
Конструктор SizedBox.shrink() наоборот делает дочерний элемент максимально маленького размера (может вообще исчезнуть).

Пример, как сделать кастомное расстояние между элементами:



# Виджет Expanded

Позволяет дочернему виджету заполнить все доступное пространство. Пример:

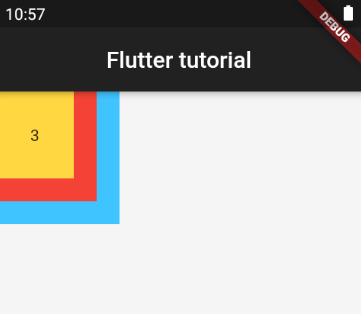


Но заполняет пустое пространство только вдоль **главной** оси. Если обернуть в Expand еще один контейнер, они будут делить свободное место.

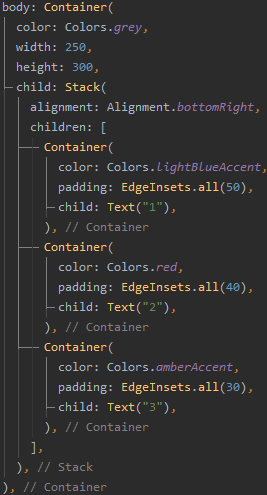
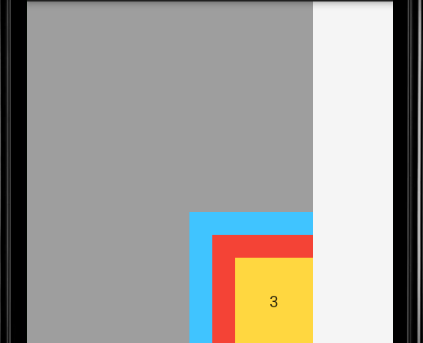
У Expand есть свойство **flex** (аналог flex-grow) с помощью которого можно управлять какую долю свободного пространства он займет. Пример: У первого Expand – flex: 1, у второго – 2, у третьего – 2. Всего 5 долей. Первый будет занимать 1/5 пространства, второй 2/5, третий тоже 2/5

# Виджет Stack / Positioned

Позволяет наложить один виджет на другой.



* С помощью свойства aligment стека можно управлять выравниванием в нем элементов. Размер стека – это размер самого большого в нем элемента.
* Можно обернуть Stack в контейнер или SizedBox и задать им размеры – Stack примет размер родителя:

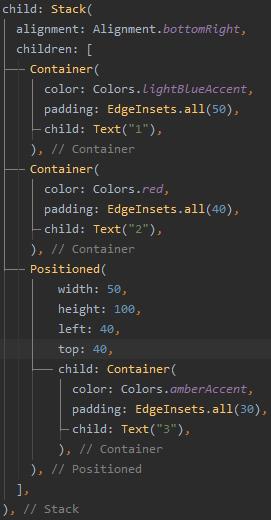
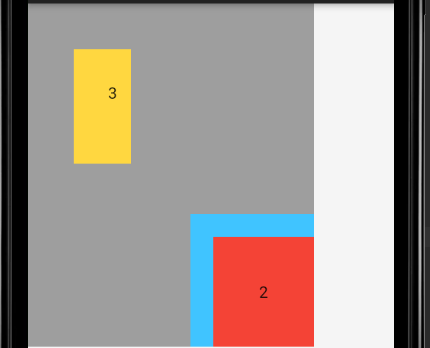


* Stack имеет свойство overflow

**Positioned**

Любой контейнер в Стэке можем положить в виджет Positioned и можем менять его положение.

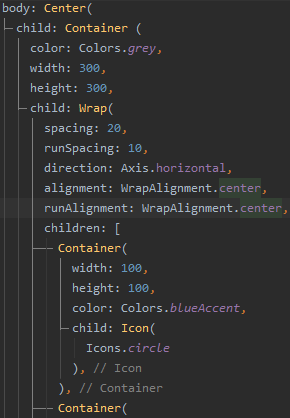
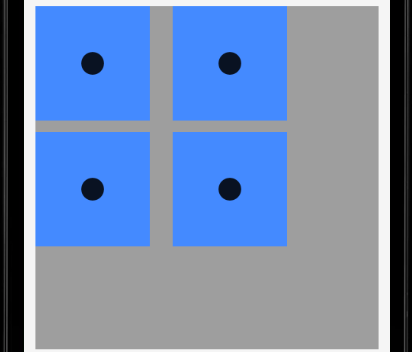
* В Positioned можем указать ширину и высоту обернутого виджета, эти размеры будут перезаписывать размеры обернутого контейнера.
* С помощью свойств left/top (bottom/right) виджета Positioned можем указать положение обернутого контейнера относительно Стека

# Виджет Wrap

Позволяет переносить дочерние виджеты на следующую строку, если им не хватает места.

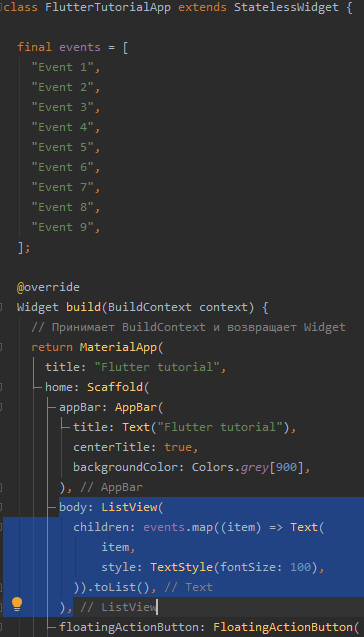
* Направление оси задается свойством direction.
* Расстояние между элементами по главной оси задается свойством spacing, вдоль поперечной – свойством runSpacing
* Выравнивание вдоль главной оси задает свойство aligment, вдоль поперечной runAligment

* Свойство verticalDirection управляет порядком заполнения элементов виджета Wrap. Значение verticalDirection.down – это по умолчанию

# Виджет ListView

Позволяет создать список из виджетов, который можно скроллить:



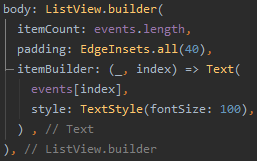
* С помощью свойства scrollDirection виджета ListView можно изменить направление скроллинга, например:



* Свойство reverse: true изменить порядок детей ListView на противоположный

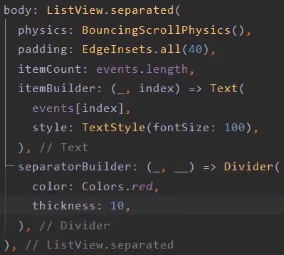
**Конструктор ListView.builder**

Создает только те элементы, которые отображаются на экране, которые находятся заграницей – уничтожаются:



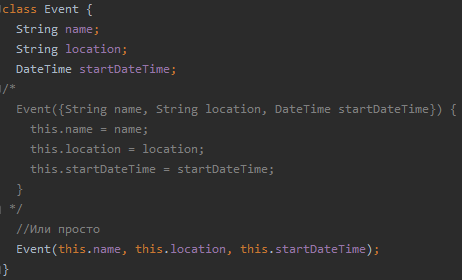
**Конструктор ListView.separated**

Создает разделитель между элементами

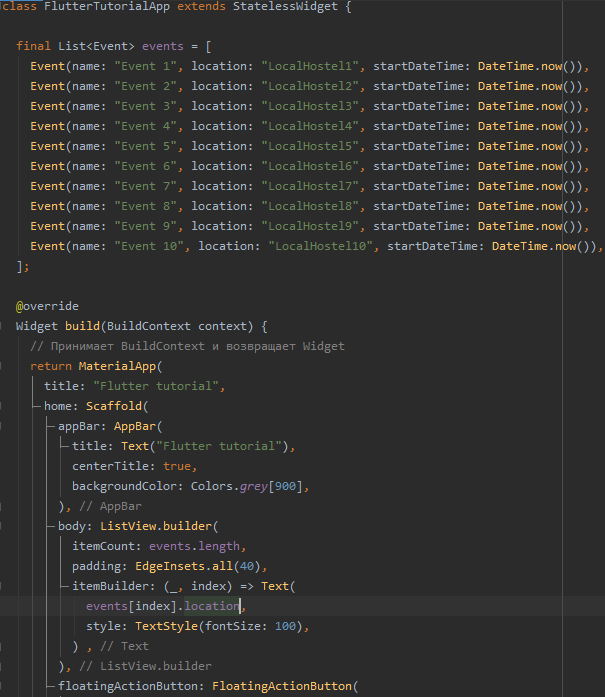


# Создаем Event class (события в реальном мире)

Правая кнопка мыши по папке lib и создаем в ней новый файл event.main.

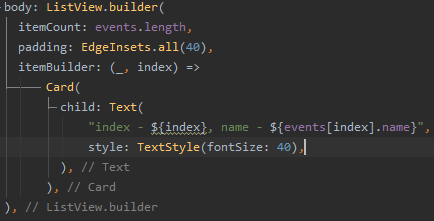


Помещаем экземпляры класса Event в виджет ListView:



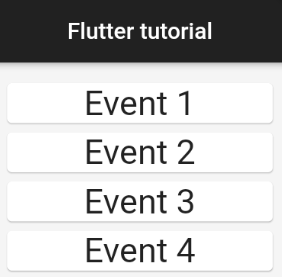
# Виджет Card

Визуально отделяет содержимое карточки. Пример:



Сверстаем карточку:

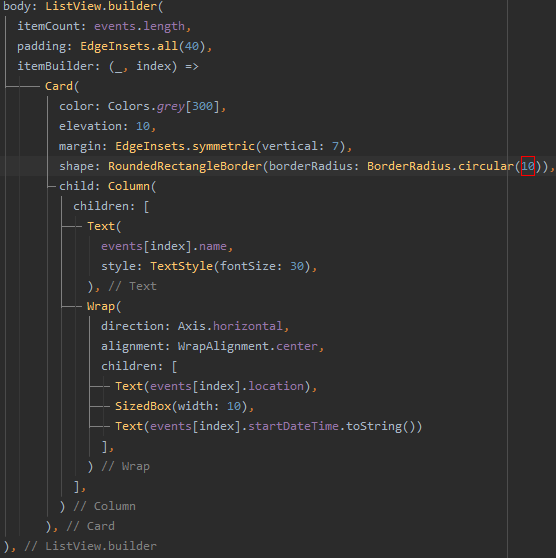
* Шаг 1. Заголовок карточки



* Шаг 2. Добавление информации о событии. Добавление переноса строки.



* Шаг 3. Стилизация карточки. Работа со свойствами.
  + Свойство elevation задает как высоко визуально карточка будет висеть над поверхностью
  + Свойство shadowColor виджета Card задает цвет тени
  + Есть свойство margin, чтобы указать расстояние между карточками
  + Свойство shape задает форму карточки



# Виджет ListTile

До этого, чтобы структурировать контент внутри карточек мы юзали Coloumn, Row, Wrap.

Для того же самого мы можем использовать ListTile.

