Hướng dẫn sử dụng Flutter Row

Xem thêm các chuyên mục:

Các hướng dẫn lập trình Flutter

Hãy theo dõi chúng tôi trên Fanpage để nhận được thông báo mỗi khi có bài viết mới. Và tham gia nhóm facebook Những người thích lập trình, nơi bạn có thể đặt câu hỏi và nhận được các câu trả lời từ các thành viên khác hoặc các tác giả.

1- Row

Row là một widget hiển thị các widget con của nó trên một hàng. Một biến thể khác là **Column**, hiển thị các widget con của nó trên một cột.

- Hướng dẫn sử dụng Flutter Column

Để làm cho widget con của **Row** có thể mở rộng lấp đầy khoảng không gian nằm ngang sẵn có bạn có thể gói nó bên trong một đối tượng **Expanded**.

- Hướng dẫn sử dụng Flutter Expanded

Row đặt các con của nó trên một dòng và không thể cuộn, nếu bạn muốn có một bộ chứa (container) tương tự và có thể cuộn được hãy cân nhắc sử dụng **ListView**.

Row Constructor:

Row Constructor

```
Row(
  {Key key,
  List<Widget> children: const <Widget>[],
  MainAxisAlignment mainAxisAlignment: MainAxisAlignment.start,
  MainAxisSize mainAxisSize: MainAxisSize.max,
  CrossAxisAlignment crossAxisAlignment: CrossAxisAlignment.center,
  TextDirection textDirection,
  VerticalDirection verticalDirection: VerticalDirection.down,
  TextBaseline textBaseline: TextBaseline.alphabetic
  }
)
```

Các hướng dẫn lập trình Flutter

- Cài đặt Flutter SDK trên Windows
- Cài đặt Flutter Plugin cho Android Studio
- Tạo ứng dụng Flutter đầu tiên của bạn - Hello Flutter
- Hướng dẫn sử dụng Flutter Scaffold
- Hướng dẫn sử dụng Flutter AppBar
- Hướng dẫn sử dụng Flutter BottomAppBar
- Hướng dẫn sử dụng Flutter TabBar
- Hướng dẫn sử dụng Flutter Banner
- Hướng dẫn sử dụng Flutter SplashScreen
- Hướng dẫn sử dụng Flutter BottomNavigationBar
- Hướng dẫn sử dụng Flutter FancyBottomNavigation
- Hướng dẫn sử dụng Flutter CircularProgressIndicator
- Hướng dẫn sử dụng Flutter LinearProgressIndicator
- Hướng dẫn sử dụng Flutter Container
- Hướng dẫn sử dụng Flutter Center
- Hướng dẫn sử dụng Flutter Align
- Hướng dẫn sử dụng Flutter Row
- Hướng dẫn sử dụng Flutter Column
- Hướng dẫn sử dụng Flutter Stack
- Hướng dẫn sử dụng Flutter IndexedStack
- Hướng dẫn sử dụng Flutter Spacer
- Hướng dẫn sử dụng Flutter Expanded

- Hướng dẫn sử dụng Flutter Center

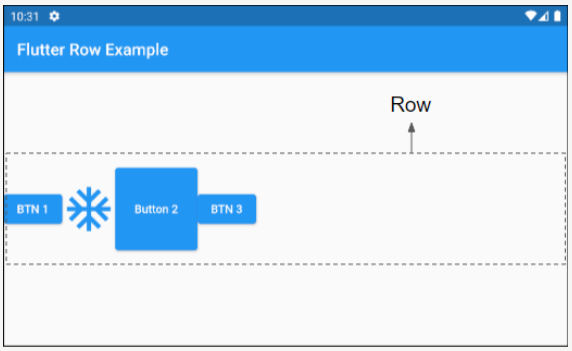
2- children

Property **children** được sử dụng để định nghĩa một danh sách các widget con của **Row**.

> Bạn có thể thêm các widget con vào **children**, hoặc loại bỏ các widget ra khỏi **children**, tuy nhiên bạn phải tuân thủ một quy tắc sẽ được đề cập trong mục **"Add/Remove Children"**.

```
List<Widget> children: const <Widget>[]
```

Chúng ta bắt đầu với ví dụ đầu tiên, một **Row** với 4 widget con:



main.dart (children ex1)

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'o7planning.org',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatelessWidget {
  MyHomePage({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          title: Text("Flutter Row Example")
      ),
      body: Center(
          child: Row (
              children: [
                ElevatedButton(child: Text("BTN 1"), onPressed:(){}),
                Icon(Icons.ac_unit, size: 64, color: Colors.blue),
                ElevatedButton(
                    child: Text("Button 2"),
                    onPressed:(){},
                    style: ButtonStyle(
                        minimumSize: MaterialStateProperty.all(Size.square(100))
                    )
                ),
                ElevatedButton(child: Text("BTN 3"), onPressed:(){}),
              ]
          )
      ),
    );
```

- Hướng dẫn sử dụng Flutter SizedBox
- Hướng dẫn sử dụng Flutter RotatedBox
- Hướng dẫn sử dụng Flutter Card
- Hướng dẫn sử dụng Flutter CircleAvatar
- Hướng dẫn sử dụng Flutter IconButton
- Hướng dẫn sử dụng Flutter FlatButton
- Hướng dẫn sử dụng Flutter TextButton
- Hướng dẫn sử dụng Flutter ElevatedButton
- Hướng dẫn sử dụng Flutter SnackBar
- Hướng dẫn sử dụng Flutter Tween
- Hướng dẫn sử dụng Flutter SimpleDialog
- Hướng dẫn sử dụng Flutter AlertDialog
- Navigation và Routing trong Flutter
- Hướng dẫn sử dụng Flutter ShapeBorder
- Hướng dẫn sử dụng Flutter Border
- Hướng dẫn sử dụng Flutter ContinuousRectangleBorder
- Hướng dẫn sử dụng Flutter RoundedRectangleBorder
- Hướng dẫn sử dụng Flutter CircleBorder
- Hướng dẫn sử dụng Flutter StadiumBorder
- Hướng dẫn sử dụng Flutter EdgeInsetsGeometry
- Hướng dẫn sử dụng Flutter EdgeInsets
- Hướng dẫn sử dụng Flutter Alignment
- Hướng dẫn sử dụng Flutter Positioned

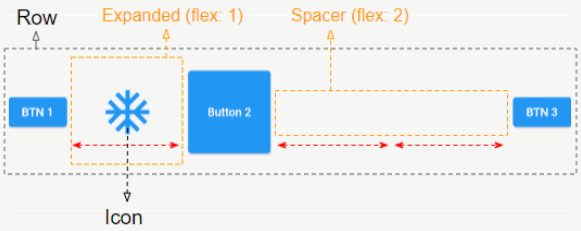
Tài liệu mới nhất

- Hướng dẫn sử dụng Java BiFunction
- Hướng dẫn sử dụng Java Map
- Hướng dẫn sử dụng Java NavigableSet
- Hướng dẫn sử dụng Java SortedSet
- Hướng dẫn sử dụng Java ListIterator
- Hướng dẫn sử dụng Flutter SnackBar
- Hướng dẫn sử dụng Java Iterator
- Hướng dẫn sử dụng Java InputStream
- Hướng dẫn sử dụng Java OutputStream
- Hướng dẫn sử dụng Java Writer

Các hướng dẫn lập trình Flutter

```
  }
}
```

Một vài loại widget con với hệ số **flex > 0** có khả năng mở rộng chiều rộng của nó để lấp đầy không gian còn lại theo chiều ngang chẳng hạn **Expanded, Spacer**,.. chúng thường được sử dụng để điều chỉnh khoảng cách giữa các widget con của **Row**, dưới đây là một ví dụ như vậy:



children (ex2)

```
Row (
  children: [
    ElevatedButton(child: Text("BTN 1"), onPressed:(){}),
    Expanded(
      flex: 1,
      child: Icon(Icons.ac_unit, size: 64, color: Colors.blue),
    ),
    ElevatedButton(
      child: Text("Button 2"),
      onPressed:(){},
      style: ButtonStyle(
          minimumSize: MaterialStateProperty.all(Size.square(100))
      )
    ),
    Spacer(
      flex: 2
    ),
    ElevatedButton(child: Text("BTN 3"), onPressed:(){}),
  ]
)
```

- Hướng dẫn sử dụng Flutter Spacer
- Hướng dẫn sử dụng Flutter Expanded

3- Add/Remove children

Bạn có thể thêm một vài widget con vào một **Row** hoặc loại bỏ một vài widget con ra khỏi **Row**, và cách làm như dưới đây có thể mang đến một kết quả không như mong đợi:

** Not Working! **

```
class SomeWidgetState extends State<SomeWidget> {
  List<Widget> _children;

  void initState() {
    _children = [];
  }

  void someHandler() {
    setState(() {
        _children.add(newWidget);
    });
  }

  Widget build(BuildContext context) {
    // Reusing `List<Widget> _children` here is problematic.
    return Row(children: this._children);
  }
}
```

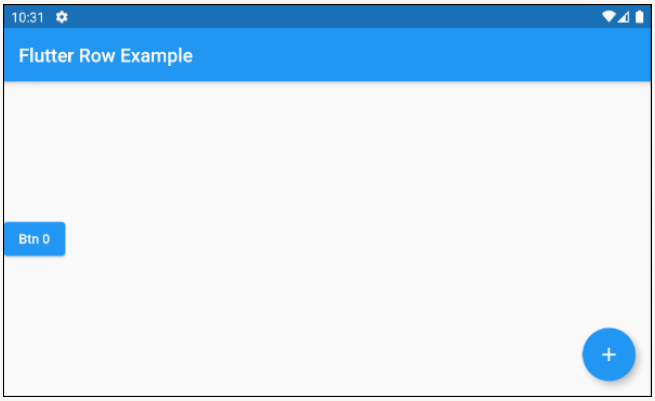
Để giải quyết vấn đề nêu trên bạn cần tuân thủ các quy tắc sau:

1. Các widget con cần được chỉ định giá trị **Key** một cách rõ ràng, điều này giúp cho **Flutter** nhận biết các widget con cũ hoặc mới khi số lượng các widget con thay đổi.
2. Cần tạo một đối tượng **List** mới cho **Row.children** nếu có một widget con nào đó thay đổi, hoặc số lượng các widget con thay đổi.

** Worked! **

```
class SomeWidgetState extends State<SomeWidget> {
  List<Widget> _children;

  void initState() {
    this._children = [];
  }

  // Add or remove some children..
  void someHandler() {
    setState(() {
      // The key here allows Flutter to reuse the underlying render
      // objects even if the children list is recreated.
      this._children.add(newWidget(key: ...));
    });
  }

  Widget build(BuildContext context) {
    // Always create a new list of children as a Widget is immutable.
    var newChildren = List.from(this._children);
    this._children = newChildren;

    return Row(children: this._children);
  }
}
```

Ví dụ:



main.dart (children ex3)

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'o7planning.org',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return MyHomePageState();
  }
}

class MyHomePageState extends State<MyHomePage> {
  List<Widget> _children = [];
  int idx = 0;
```

```
  @override
  void initState()  {
    super.initState();

    this._children = [
      ElevatedButton(
          key: Key(this.idx.toString()),
          child: Text("Btn " + idx.toString()),
          onPressed: (){}
      )
    ];
  }

  void addChildHandler()  {
    this.idx++;
    this.setState(() {
      var newChild = ElevatedButton(
          key: Key(this.idx.toString()),
          child: Text("Btn " + idx.toString()),
          onPressed: (){}
      );
      this._children.add(newChild);
    });
  }

  @override
  Widget build(BuildContext context) {
    // Create new List object:

    this._children = this._children == null? [] : List.from(this._children);

    return Scaffold(
      appBar: AppBar(
          title: Text("Flutter Row Example")
      ),
      body: Center(
          child: Row (
              children:  this._children
          )
      ),
      floatingActionButton: FloatingActionButton(
          child: Icon(Icons.add),
          onPressed: ()  {
            this.addChildHandler();
          }
      ),
    );
  }
}
```

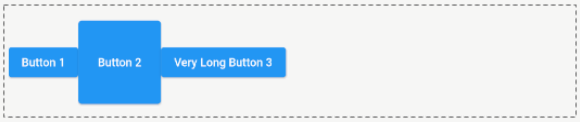
4- mainAxisAlignment

Property **mainAxisAlignment** được sử dụng để chỉ định cách mà các widget con sẽ được bố trí trên trục chính (main axis). Đối với **Row**, trục chính (main axis) là chính là trục nằm ngang.

```
MainAxisAlignment mainAxisAlignment: MainAxisAlignment.start

// MainAxisAlignment enum:

MainAxisAlignment.start
MainAxisAlignment.center
MainAxisAlignment.end

MainAxisAlignment.spaceBetween
MainAxisAlignment.spaceAround
MainAxisAlignment.spaceEvenly

```

MainAxisAlignment.start

Với **textDirection** = **TextDirection.ltr** (Mặc định) và **mainAxisAlignment** = **MainAxisAlignment.start** thì các widget con của **Row** sẽ được đặt sát nhau từ trái sang phải.



MainAxisAlignment.start

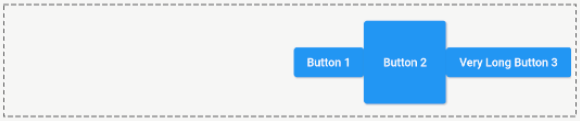
```
Row (
  mainAxisAlignment: MainAxisAlignment.start,
  children: [
    ElevatedButton(child: Text("Button 1"), onPressed:(){}),
    ElevatedButton(
        child: Text("Button 2"),
        onPressed:(){},
        style: ButtonStyle(
            minimumSize: MaterialStateProperty.all(Size.square(100))
        )
    ),
    ElevatedButton(child: Text("Very Long Button 3"), onPressed:(){})
  ]
)
```

MainAxisAlignment.center


```
mainAxisAlignment: MainAxisAlignment.center
```

MainAxisAlignment.end

Với **textDirection = TextDirection.ltr** (Mặc định) và **mainAxisAlignment = MainAxisAlignment.end** thì các widget con của **Row** sẽ được đặt sát nhau từ phải sang trái.

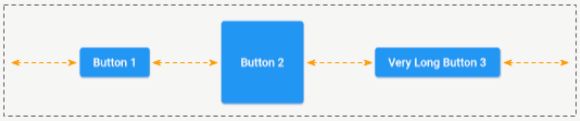



```
mainAxisAlignment: MainAxisAlignment.end
```

MainAxisAlignment.spaceBetween


```
mainAxisAlignment: MainAxisAlignment.spaceBetween
```

MainAxisAlignment.spaceEvenly




```
mainAxisAlignment: MainAxisAlignment.spaceEvenly
```

MainAxisAlignment.spaceAround


```
mainAxisAlignment: MainAxisAlignment.spaceAround
```

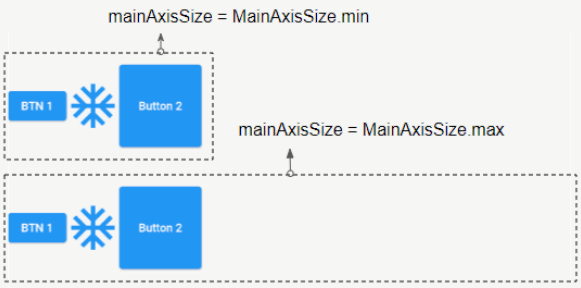
5- mainAxisSize

Property **mainAxisSize** chỉ định bao nhiêu không gian nằm ngang nên được chiếm giữ bởi **Row**, giá trị mặc định của nó là **MainAxisSize.max** điều này có nghĩa là **Row** sẽ cố gắng chiếm giữ không gian nằm ngang nhiều nhất có thể.

Nếu có một widget con với **"flex > 0 && fit != FlexFit.loose"** thì **Row** sẽ cố gắng chiếm giữ nhiều không gian nhiều nhất có thể mà không phụ thuộc vào giá trị của **mainAxisSize**.

Ngược lại, nếu **mainAxisSize = MainAxisSize.min** thì **Row** sẽ có một chiều rộng vừa đủ cho tất cả các widget con của nó.

```
MainAxisSize mainAxisSize: MainAxisSize.max

// MainAxisSize enum:

MainAxisSize.max
MainAxisSize.min


```



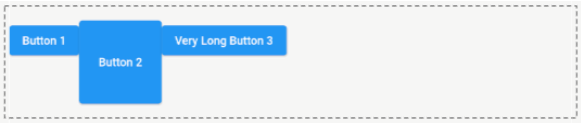
6- crossAxisAlignment

Property **crossAxisAlignment** được sử dụng để chỉ định cách mà các widget con sẽ được bố trí trên trục chéo (cross axis). Đối với **Row**, trục chéo (cross axis) chính là trục thẳng đứng.

```
CrossAxisAlignment crossAxisAlignment: CrossAxisAlignment.center

// CrossAxisAlignment enum:

CrossAxisAlignment.start
CrossAxisAlignment.end
CrossAxisAlignment.center
CrossAxisAlignment.baseline
CrossAxisAlignment.stretch

```

CrossAxisAlignment.start

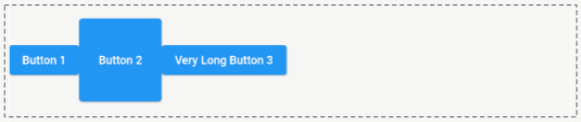
Với **verticalDirection** = **VerticalDirection.down** (Mặc định) và **crossAxisAlignment** = **CrossAxisAlignment.start** thì các widget con của **Row** sẽ được đặt sát cạnh trên cùng của **Row**.



CrossAxisAlignment.start

```
Row (
  crossAxisAlignment: CrossAxisAlignment.start,

  children: [
    ElevatedButton(child: Text("Button 1"), onPressed:(){}),
    ElevatedButton(
        child: Text("Button 2"),
        onPressed:(){},
        style: ButtonStyle(
            minimumSize: MaterialStateProperty.all(Size.square(100))
        )
    ),
    ElevatedButton(child: Text("Very Long Button 3"), onPressed:(){})
  ]
)
```

CrossAxisAlignment.center (Default).



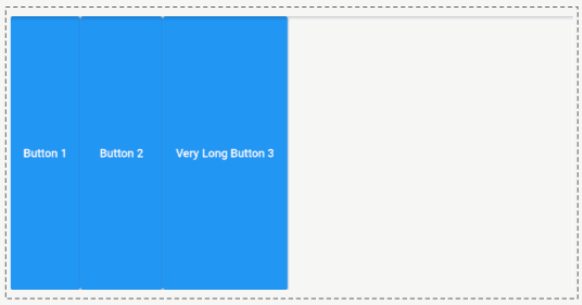

```
crossAxisAlignment: CrossAxisAlignment.center
```

CrossAxisAlignment.end

Với **verticalDirection = VerticalDirection.down** (Mặc định) và **crossAxisAlignment = CrossAxisAlignment.end** thì các widget con của **Row** sẽ được đặt sát cạnh dưới cùng của **Row**.


```
crossAxisAlignment: CrossAxisAlignment.end
```

CrossAxisAlignment.stretch

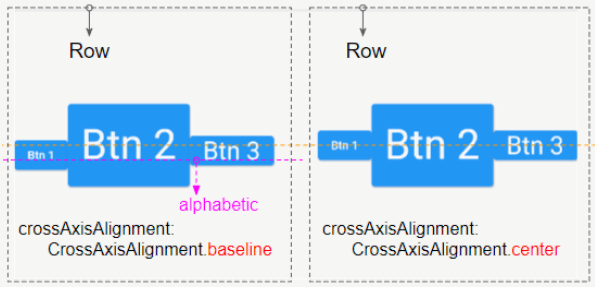



```
crossAxisAlignment: CrossAxisAlignment.stretch
```

CrossAxisAlignment.baseline

```
crossAxisAlignment: CrossAxisAlignment.baseline
```

Ví dụ:




```
crossAxisAlignment: CrossAxisAlignment.baseline,
textBaseline: TextBaseline.alphabetic
```

7- textDirection

Property **textDirection** chỉ định cách mà các widget con của **Row** sẽ được bố trí trên trục chính (trục nằm ngang) và cách giải thích từ **"start"** và **"end"**.

```
TextDirection textDirection

// TextDirection enum:

TextDirection.ltr (Left to Right) (Default)
TextDirection.rtl (Right to Left)
```

Nếu **textDirection = TextDirection.ltr** (Mặc định), từ **"start"** sẽ tương ứng với **"left"** và từ **"end"** sẽ tương ứng với **"right"**.

Ngược lại, nếu **textDirection = TextDirection.rtl**, từ **"start"** sẽ tương ứng với **"right"** và từ **"end"** tương ứng với **"left"**.

8- verticalDirection

Property **verticalDirection** chỉ định cách mà các widget con của **Row** sẽ được bố trí trên trục chéo (trục thẳng đứng) và cách giải thích từ **"start"** và **"end"**.

```
VerticalDirection verticalDirection: VerticalDirection.down

// VerticalDirection enum:

VerticalDirection.down (Default)
VerticalDirection.up
```

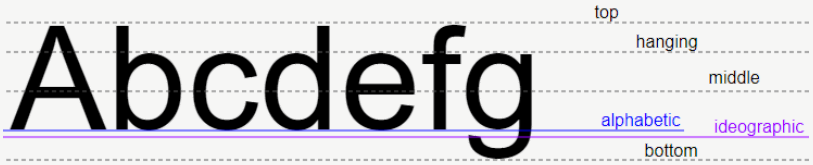
Nếu **verticalDirection = VerticalDirection.down** (Mặc định), từ **"start"** sẽ tương ứng với **"top"** và từ **"end"** sẽ tương ứng với **"bottom"**.

Ngược lại, nếu **verticalDirection = VerticalDirection.up**, từ **"start"** sẽ tương ứng với **"bottom"** và từ **"end"** tương ứng với **"top"**.

9- textBaseline

Nếu căn lề (align) các widget con dựa trên đường cơ sở (baseline), property **textBaseline** sẽ chỉ định loại đường cơ sở nào sẽ được sử dụng.

```
TextBaseline textBaseline: TextBaseline.alphabetic

// TextBaseline enum:

TextBaseline.alphabetic (Default)
TextBaseline.ideographic
```



Ví dụ:

```
crossAxisAlignment: CrossAxisAlignment.baseline,
textBaseline: TextBaseline.alphabetic
```

Xem thêm các chuyên mục:

Các hướng dẫn lập trình Flutter

Có thể bạn quan tâm

Đây là các khóa học trực tuyến bên ngoài website o7planning mà chúng tôi giới thiệu, nó có thể bao gồm các khóa học miễn phí hoặc giảm giá.

- Udemy Flutter News Portal App-Firestore Backend(Android&ios App)
- Udemy The Complete Flutter UI Masterclass | iOS & Android in Dart
- Udemy Dart 2 Complete Bootcamp - Go Hero from Zero in Dart Flutter
- Udemy Mastering Flutter
- Udemy Flutter Blog app Using Firestore Build ios & Android App
- Udemy Mobile E-Commerce with Flutter, Redux, and Stripe
- Udemy Happy Flutter - Sport News Apps Flutter
- Udemy Create a Post Reader App with Flutter
- Udemy The Complete 2020 Flutter Development Bootcamp with Dart
- Udemy Real-World Projects with Flutter
- Udemy Getting Started with Flutter and Firebase
- Udemy Dart and Flutter: The Complete Developer's Guide
- Udemy Master Flutter - Learn Dart & Flutter by Developing 20 Apps
- Udemy Flutter Build a Complex Android and ios Apps Using Firestore
- Udemy Dart and Flutter From Zero to Hero - Practical Dev Bootcamp
- Udemy Dart & Flutter: The Complete Mobile Apps Development Course
- Udemy The Complete Flutter and Firebase Developer Course
- Udemy Intro to Flutter For iOS & Android
- Udemy Flutter Zero to Professional: cross platform App iOS/Android
- Udemy Flutter Web Development Course Build Complete FlutterWeb App
- Udemy Flutter & Dart: A Complete Showcase Mobile App™
- Udemy Flutter & Firebase: Build a Complete App for iOS & Android
- Udemy Flutter & Dart - The Complete Flutter App Development Course
- Udemy The Complete Flutter UI Course | Build Amazing Mobile Apps
- Udemy Build a Social Network with Flutter and Firebase