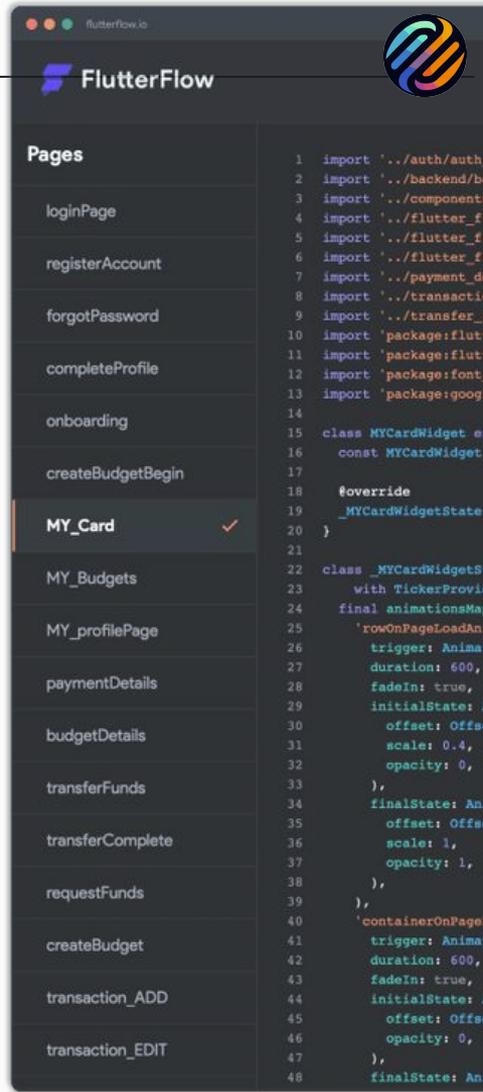


# Low-Code ≠ No-Code: jak FlutterFlow, Cursor AI i Firebase napędzają startup

FlowOrbiter.com

2025

Tomasz Kośmider





# Flutterflow, Visual Development Environment /-2021

Use the modal to find “out of the box” widgets

The screenshot displays the Flutterflow VDE interface. On the left, a sidebar contains a 'Build' menu with icons for adding elements, a search bar for pages or components, a 'Connect' menu, and a 'Widget Tree' showing the hierarchy of the current page. The main workspace shows a design for a 'ProductDetailPage' with a central product image, a title 'Men's Harrington Jacket', a price '\$148', and interactive elements for 'Size' (set to 'S'), 'Color', and 'Quantity' (set to '1'). A 'Commonly Used Elements' panel is open, showing various widget icons like Text, Column, Row, Container, Image, and Button. The top of the interface includes a toolbar with zoom and navigation tools, and a 'Page Parameters' panel on the right showing route settings and visibility options.

Use properties from the theme

The screenshot shows the 'Theme Settings' panel, specifically the 'Colors' section. It features a 'Colors' header with a 'Docs' link and a '+ Add Color' button. Below this, there are tabs for 'System Default Colors', 'Import from Colors', 'Extract from Image', and 'Generate with AI'. The panel is divided into 'Light Mode Theme' and 'Dark Mode Theme' sections. Each section displays a grid of color swatches categorized by 'Brand Colors' (Primary, Secondary, Tertiary, Alternate, Primary Text, Secondary Text, Primary Background, Secondary Background), 'Accent Colors' (Accent 1-4), and 'Semantic Colors' (Success, Error, Warning, Info). The 'Dark Mode Theme' section has a toggle switch that is currently turned on.



# Contents

---

01

o FlowOrbiter

---

02

Wprowadzenie i Archeologia

---

03

o FlutterFlow

---

04

o Projektach Praktycznie

---

05

Demo & Podsumowanie

---



# FlutterFlow Capabilities



**Standardize Client Side  
Application Development**



**Rapidly Build Modern UX  
For Mobile, Web, & Desktop**



**Connect To Any Backend  
Via REST API / Firebase**



**Visually Implement Client  
Side Business Logic**



**Real-Time Collaboration  
With Product Teams**



**Extend The Platform  
With Code**



**Build A Rich Library Of  
Reusable Assets**



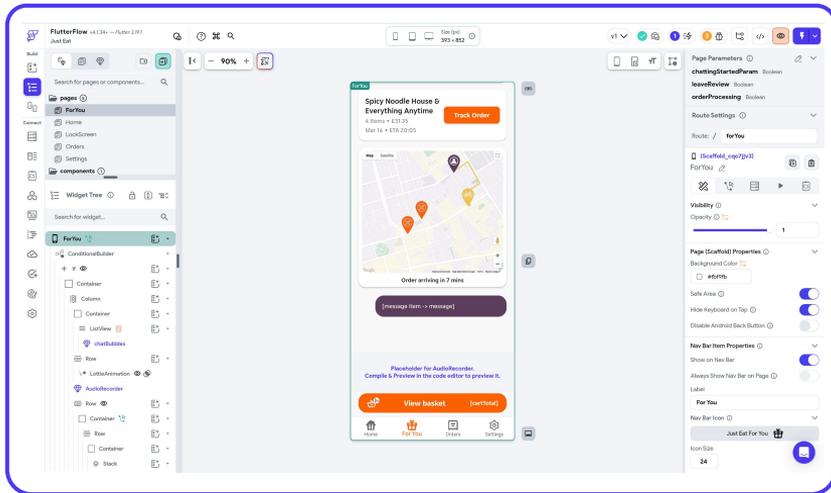
**Own Your Intellectual  
Property**





# FlutterFlow architecture

User interacts with the FlutterFlow application

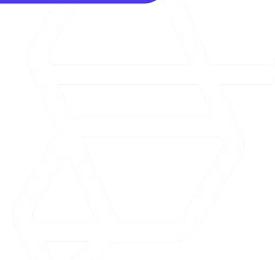


FlutterFlow saves project entities in intermediary protobuf

```
message FlutterFlowProject {  
  string project_id ...  
  string project_name ...  
  repeated Page pages ...  
  ...  
}
```

FlutterFlow backend converts to Flutter code

```
...  
/lib  
  /pages  
  /components  
  ...  
  main.dart
```





# Goal for developers



Flutterflow

**Makes you *faster*.**

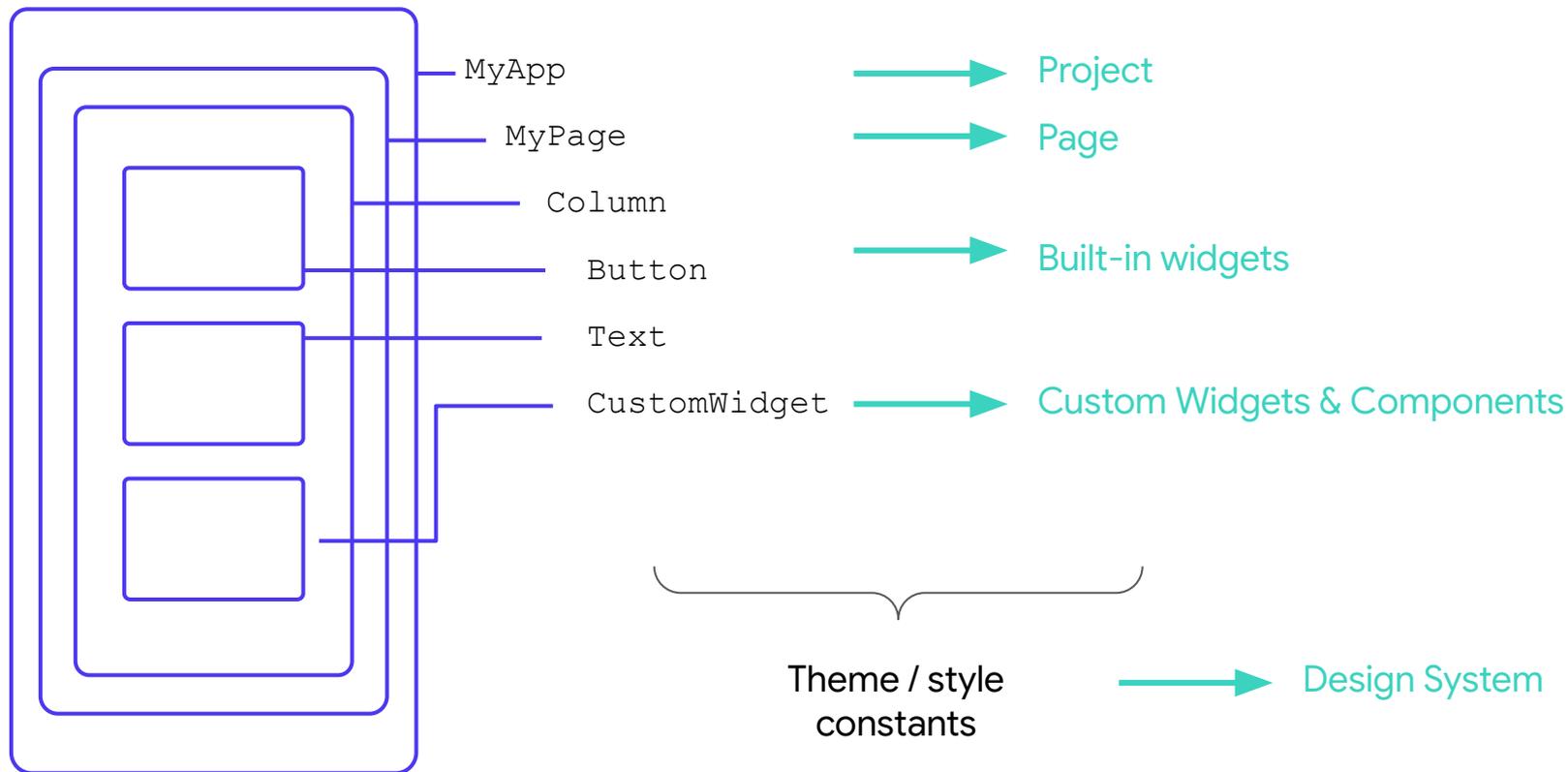
Without sacrificing on your  
app's quality, or features

```
...  
/lib  
  /pages  
  /components  
...  
main.dart
```

Export the code whenever  
you want



# Flutter App Parts to FlutterFlow App Parts





# Pages in the generated Code

## Separation of UI and data model

The image shows a code editor with a sidebar on the left containing a project structure. The sidebar lists 'Pages' (ProductListPage, CategoryProductListPage, CategoryListPage, **ProductDetailPage**, CartPageError, CartPage), 'Components' (CategoryAvatar, ProductListCard, CategoryListItem, EmptyCartWidget, CartItem, PriceLine), 'Pubspec' (pubspec.yaml), and 'Firestore Indexes'. The main editor displays the following Dart code:

```
11 class ProductDetailPageWidget extends StatefulWidget {
12   const ProductDetailPageWidget({super.key});
13
14
15   @override
16   State<ProductDetailPageWidget> createState() =>
17     _ProductDetailPageWidgetState();
18 }
19
20 class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {
21   late ProductDetailPageModel _model;
22
23   final scaffoldKey = GlobalKey<ScaffoldState>();
24
25   @override
26   void initState() {
27     super.initState();
28     _model = createModel(context, () => ProductDetailPageModel());
29   }
30
31   @override
32   void dispose() {
33     _model.dispose();
34     super.dispose();
35   }
36 }
37
38 @override
39 Widget build(BuildContext context) {
40   return GestureDetector(
41     onTap: () => _model.unfocusNode.canRequestFocus
42       ? FocusScope.of(context).requestFocus(_model.unfocusNode)
43       : FocusScope.of(context).unfocus(),
44     child: Scaffold(
45       key: scaffoldKey,
46       backgroundColor: FlutterFlowTheme.of(context).info,
47       appBar: AppBar(
48         backgroundColor: FlutterFlowTheme.of(context).secondaryBackground,
49         automaticallyImplyLeading: false,
50         leading: Padding(
51           padding: EdgeInsets.all(10),
52           child: FlutterFlowIconButton(
53             borderRadius: 40,
54             borderWidth: 1,
55             fillColor: FlutterFlowTheme.of(context).alternate,
56             icon: Icon(
```

A blue box highlights the 'Widget' and 'Model' tabs in the editor's top right corner. To the right of the code editor is a preview of the 'ProductDetailPage' UI, which displays a product card for a 'Men's Harrington Jacket' priced at '\$148'. The card includes a 'Size' dropdown menu (set to 'S'), a 'Color' dropdown menu (set to a blue color), and a 'Quantity' selector (set to '1'). Below the product card, there is a descriptive paragraph: 'Built for life and made to last, this full-zip corduroy jacket is part of our Nike Life collection. The spacious fit gives you plenty of room to layer underneath, while the soft corduroy keeps it casual and timeless.'



# Page's Widget code

```
class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return GestureDetector(  
      onTap: () => _model.unfocusNode.canRequestFocus  
        ? FocusScope.of(context).requestFocus(_model.unfocusNode)  
        : FocusScope.of(context).unfocus(),  
      child: Scaffold(  
        key: scaffoldKey,  
        backgroundColor: FlutterFlowTheme.of(context).info,  
        appBar: AppBar(  
          leading: Padding(  
            padding: EdgeInsets.all(10),  
            child: FlutterFlowIconButton(  
              ...  
            ),  
          ),  
        body: SafeArea(...)  
      ),  
    );  
  }  
}
```

**Pages** in FF are just widgets, only they use a scaffold and routing is automatically configured using `go_router`

Widget properties are set using the configurations in the property panel



# Page's Widget code

```
class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return GestureDetector(  
      onTap: () => _model.unfocusNode.canRequestFocus  
        ? FocusScope.of(context).requestFocus(_model.unfocusNode)  
        : FocusScope.of(context).unfocus(),  
      child: Scaffold(  
        key: scaffoldKey,  
        backgroundColor: FlutterFlowTheme.of(context).info,  
        appBar: AppBar(  
          leading: Padding(  
            padding: EdgeInsets.all(10),  
            child: FlutterFlowIconButton(  
              ...  
            ),  
          body: SafeArea(...)  
        ),  
      ),  
    );  
  }  
}
```

Many of the out of the box widgets come right from the Flutter framework



# Page's Widget code

```
class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return GestureDetector(  
      onTap: () => _model.unfocusNode.canRequestFocus  
        ? FocusScope.of(context).requestFocus(_model.unfocusNode)  
        : FocusScope.of(context).unfocus(),  
      child: Scaffold(  
        key: scaffoldKey,  
        backgroundColor: FlutterFlowTheme.of(context).info,  
        appBar: AppBar(  
          leading: Padding(  
            padding: EdgeInsets.all(10),  
            child: FlutterFlowIconButton(  
              ...  
            ),  
          body: SafeArea(...)  
        )  
      )  
    )  
  }  
}
```

Some are created by FlutterFlow, to better match the needs of building apps with your own branding



# Page's Widget code

```
class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {  
  ...  
  @override  
  Widget build(BuildContext context) {  
    return GestureDetector(  
      onTap: () => _model.unfocusNode.canRequestFocus  
        ? FocusScope.of(context).requestFocus(_model.unfocusNode)  
        : FocusScope.of(context).unfocus(),  
      child: Scaffold(  
        key: scaffoldKey,  
        backgroundColor: FlutterFlowTheme.of(context).info, →  
        appBar: AppBar(  
          leading: Padding(  
            padding: EdgeInsets.all(10),  
            child: FlutterFlowIconButton(  
              ...  
            ),  
          body: SafeArea(...)  
        )  
      )  
    )  
  }  
}
```

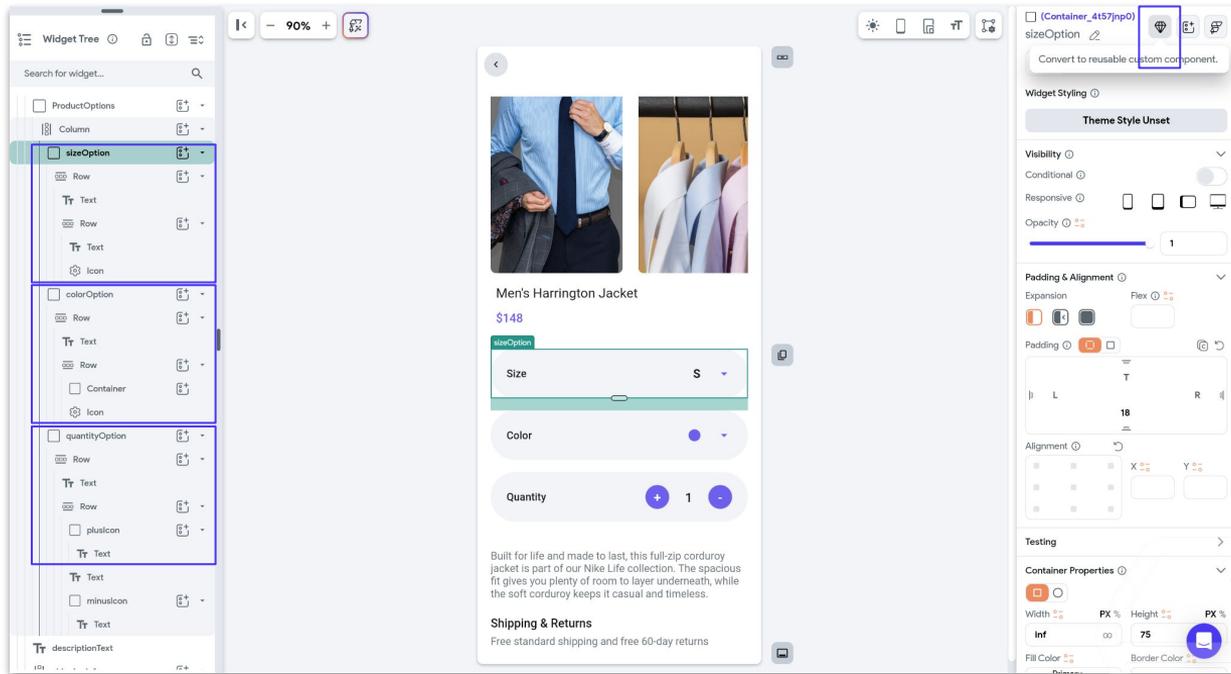
Using properties from theme leverages custom theme class that switches between light / dark



# Creating a component from the widget tree

Same principles of encapsulation and reusability apply in FlutterFlow

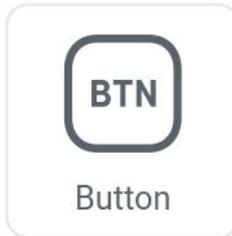
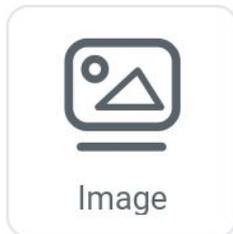
Convert widget tree to components, which are simply custom widget classes





# Types of widgets

## Base element or single-child widgets



and more..

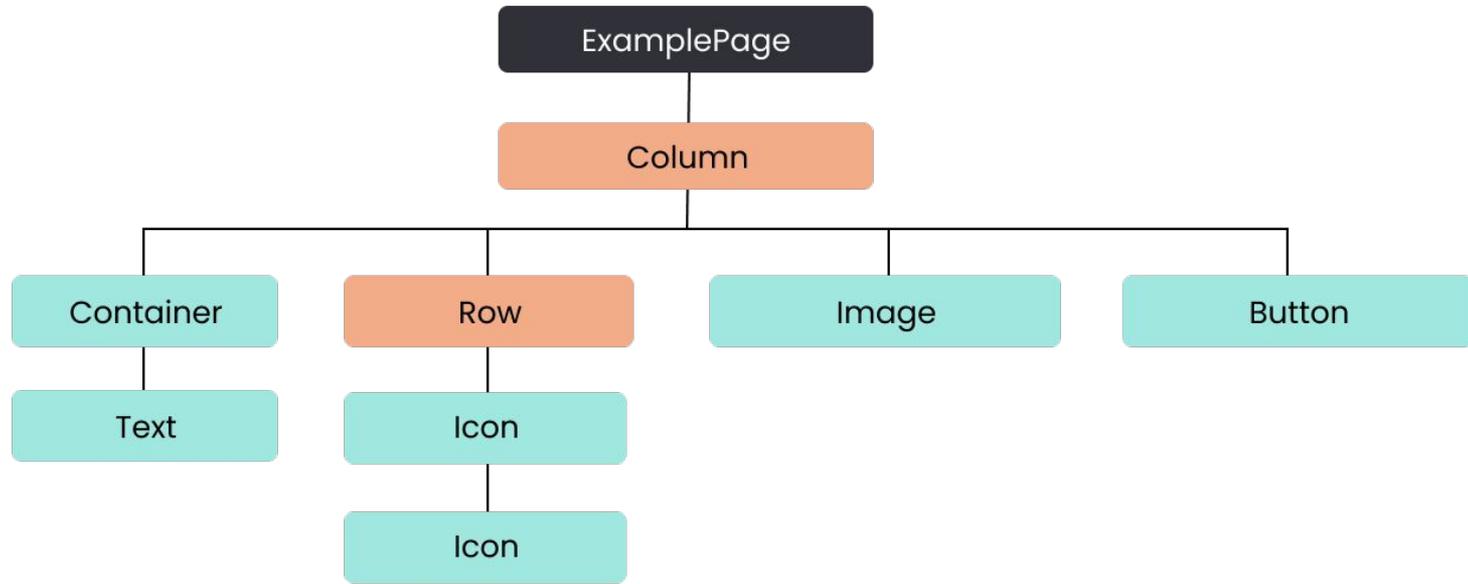
## Multiple-child widgets



and more..



# Widget Tree Hierarchy



Base Widgets    Multiple Child Widgets

# Creating custom widgets



Widget Name  
NewCustomWidget

Cancel Save Widget

```
1 // Automatic FlutterFlow imports
2 import '/backend/schema/structs/index.dart';
3 import '/backend/schema/enums/enums.dart';
4 import '/flutter_flow/flutter_flow_theme.dart';
5 import '/flutter_flow/flutter_flow_util.dart';
6 import '/custom_code/widgets/index.dart'; // Imports other custom widgets
7 import '/flutter_flow/custom_functions.dart'; // Imports custom functions
8 import 'package:flutter/material.dart';
9 // Begin custom widget code
10 // DO NOT REMOVE OR MODIFY THE CODE ABOVE!
11
12 class NewCustomWidget extends StatefulWidget {
13   const NewCustomWidget({
14     super.key,
15     this.width,
16     this.height,
17     this.inputParam,
18   });
19
20   final double? width;
21   final double? height;
22   final double? inputParam;
23
24   @override
25   State<NewCustomWidget> createState() => _NewCustomWidgetState();
26 }
27
28 class _NewCustomWidgetState extends State<NewCustomWidget> {
29   @override
30   Widget build(BuildContext context) {
31     return Container();
32   }
33 }
```

Widget Settings

Manage your dependencies and parameters below.  
We will infer the parameters from your code.

Exclude from compilation

Define Parameters

Note: Widget must accept these named parameters.

Width and Height (required)

Parameters 2

Name  
inputParam

Type  
Double

Is List  Nullable

+ Add Parameters

Pubspec Dependencies

+ Add Dependency

Pass in parameters

Can include a pubspec dependency





# State management

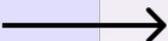
App State

John Doe



App State: *userName* - Stores the user's name, used across all pages.

Hi, John Doe!



Hi + {AppState: *userName*}

Page A

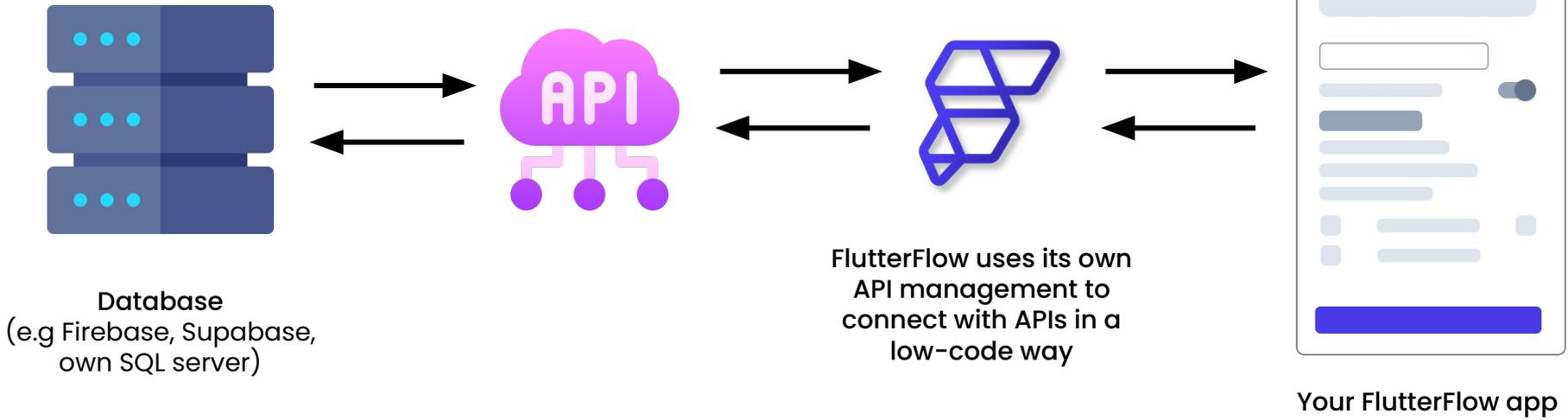


Page State: *isProcessing* controls loader visibility—true displays it during data processing.



Component State: *isFavourite* - Tracks if an item is marked as favorite.







# Connecting to the backend

Example, calling an API to get product ratings

**API Calls** ↑ + Add

Search API library...

**getRating**  
api.eCommExample.com/reviews GET

### Create API Call

Provide the name and configuration for this API call. Docs

**Call Definition** Response & Test

API Call Name

Method Type API URL ⓘ  
GET

Headers **Query Parameters** Variables Advanced Settings

**Query Parameters**

Name	Value Source	Select Variable	
<input type="text" value="productId"/>	<span>From Variable</span>	<span>productId</span>	<span>×</span>

+ Add Query Parameter

Cancel Add Call

Use API spec files to generate this config for you!

Easily test API response and convert into a custom data type!





# API Call

```
class GetRatingCall {
    static Future<ApiCallResponse> call({
        String? productId = '',
    }) async {
        return ApiManager.instance.makeApiCall(
            callName: 'getRating',
            apiUrl: 'https://api.eCommExample.com/reviews',
            callType: ApiCallType.GET,
            headers: {},
            params: {
                'productId': productId,
            },
            ...
        );
    }
}
```

Create a new class based  
on my API call  
configurations



# Calling the API

Call API on page load, and update app state variable

The screenshot displays a workflow editor for a page named 'ProductDetailPage'. The workflow is triggered by 'On Page Load'. It consists of the following steps:

- Action 1: Backend Call** (API (getRating))
- Conditional Action 1** (ratingResults → Succeeded)
- TRUE Path:** Update rating page state variable (Update Page State)
- FALSE Path:** (Empty)

The 'Define Action' panel on the right shows the configuration for the 'Backend Call API' action:

- Action 1:** Backend Call API (lgp613er)
- Group or Call Name:** getRating
- Variables:** productid
- Action Output Variable Name:** ratingResults
- Non-Blocking:** (Checked)

Pass in variables to API call

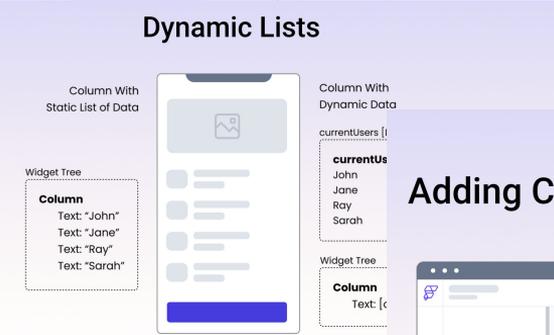
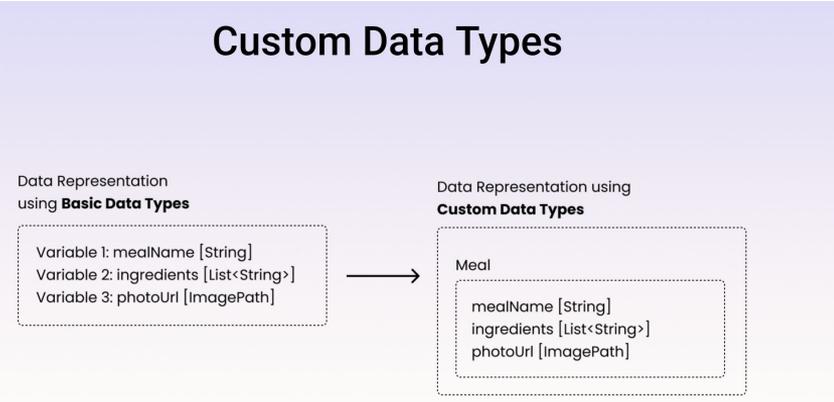
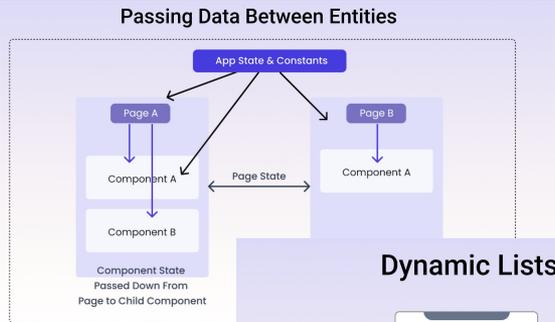
Can parse response into specific data type

Can handle errors

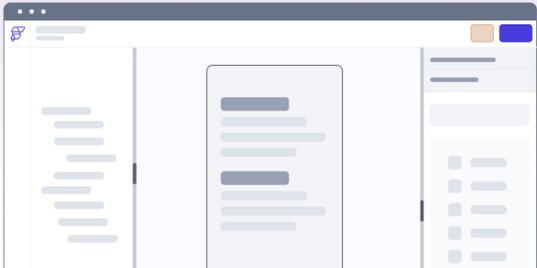


# There is much more

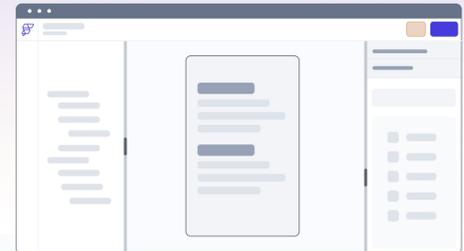
But this is not a training session...



## FlutterFlow: Adding Customizations with Flutter



## FlutterFlow: Firebase Authentication





# Contents

---

01

o FlowOrbiter

---

02

Wprowadzenie i Archeologia

---

03

o FlutterFlow

---

04

o Projektach Praktycznie

---

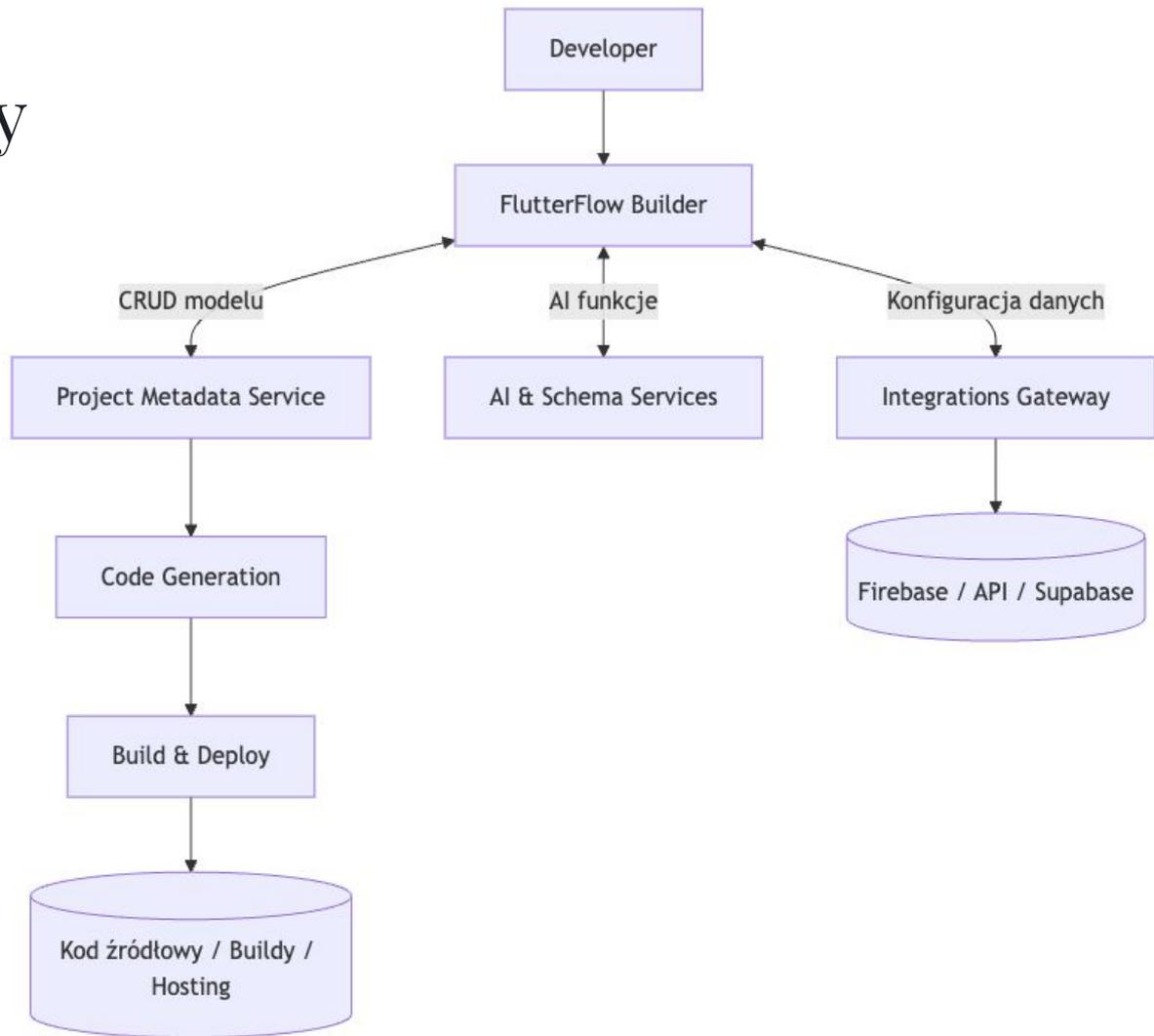
05

Demo & Podsumowanie

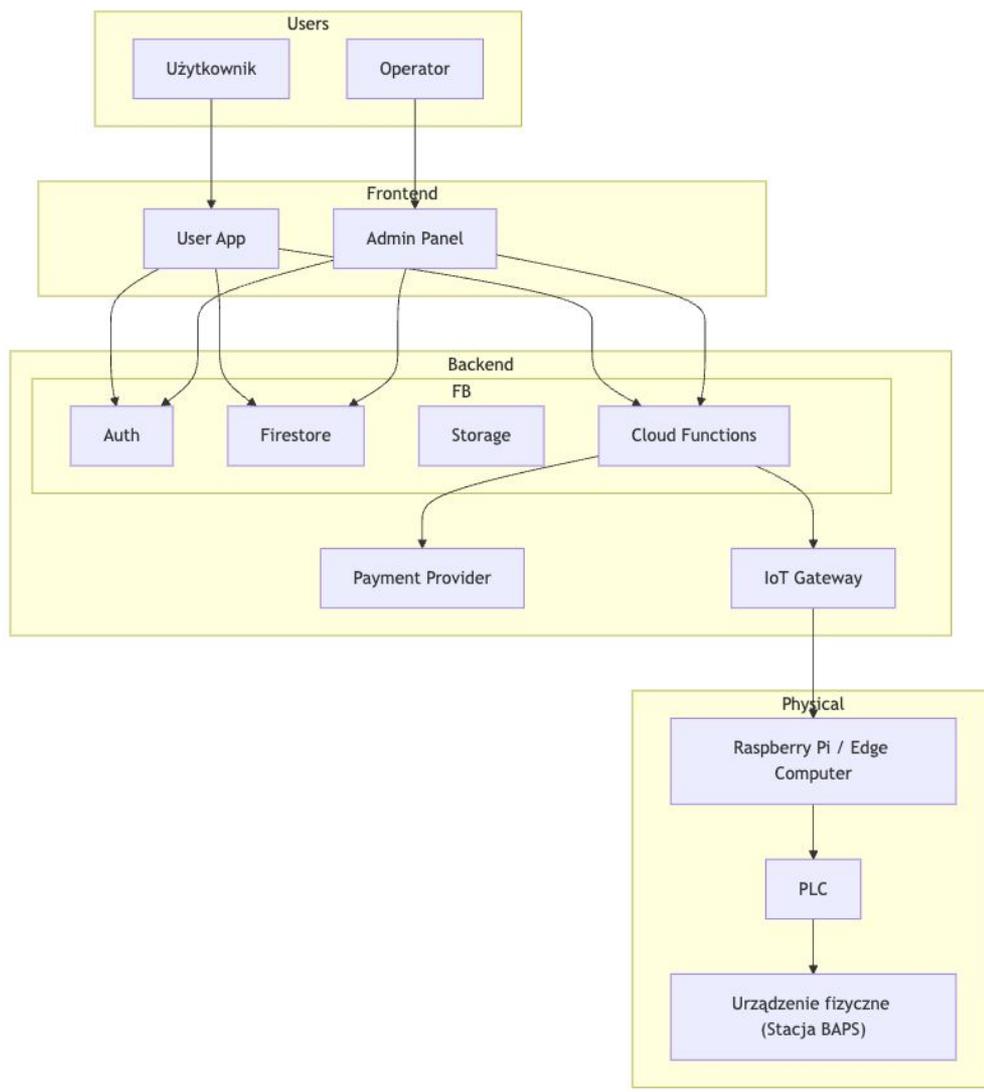
---



# Widok uproszczony – kompletny przepływ pracy developera



# Przykład projektu IoT BAPS (kontekst)





# Contents

---

01

o FlowOrbiter

---

02

Wprowadzenie i Archeologia

---

03

o FlutterFlow

---

---

04

o Projektach Praktycznie

---

05

Demo & Podsumowanie

---



*Any* questions?  
daway!

[tomasz.kosmider@floworbiter.com](mailto:tomasz.kosmider@floworbiter.com)

---