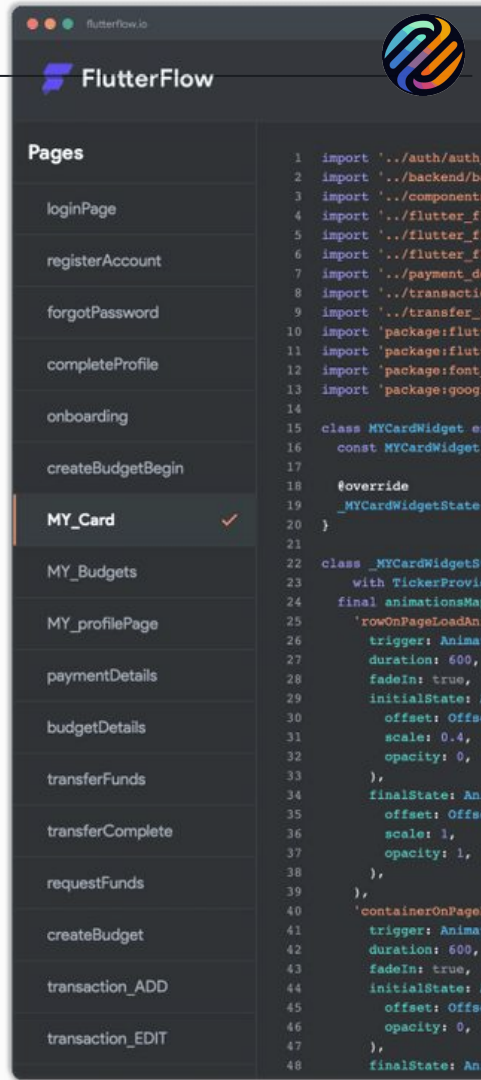
Low-Code ≠ No-Code: jak FlutterFlow, Cursor AI i Firebase napędzają startup



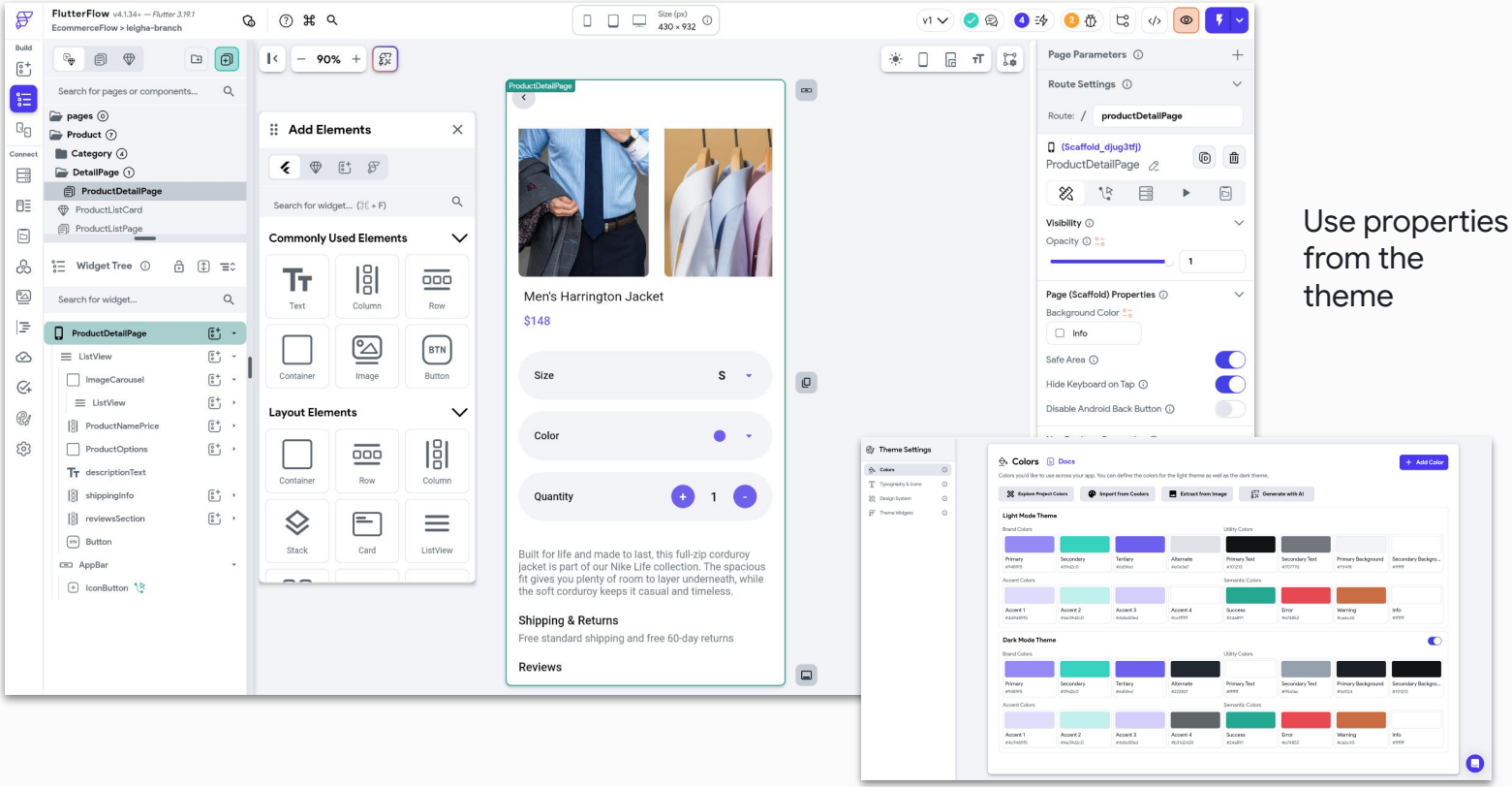
FlowOrbiter.com

2025

Tomasz Kośmider

Flutterflow, Visual Development Environment /~2021

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



Use properties from the theme

Contents

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

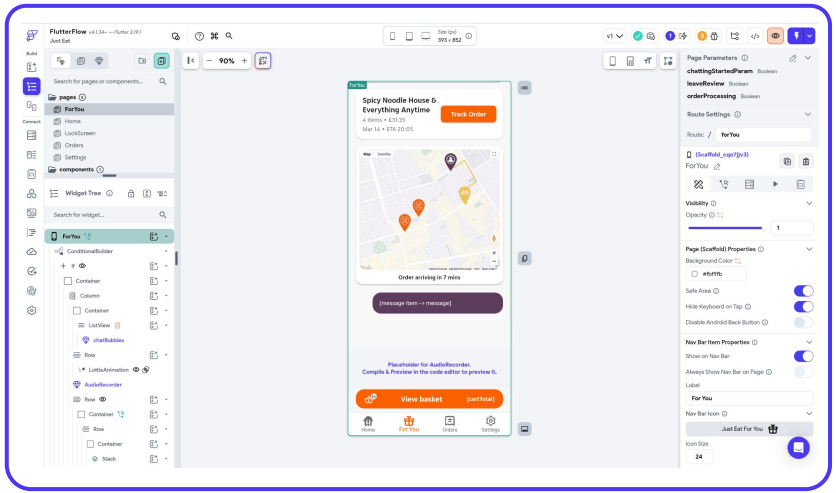
Extend The Platform With Code

Build A Rich Library Of Reusable Assets

Own Your Intellectual Property

FlutterFlow architecture

User interacts with the FlutterFlow application



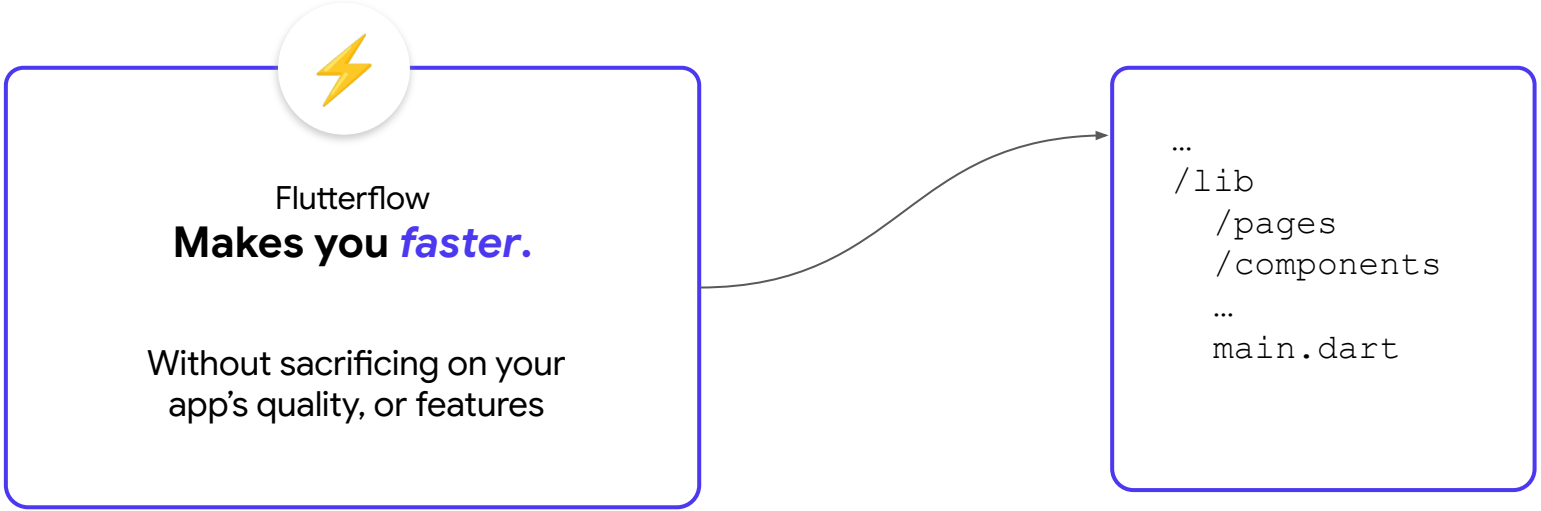
FlutterFlow saves project entities in intermediary protobuf

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

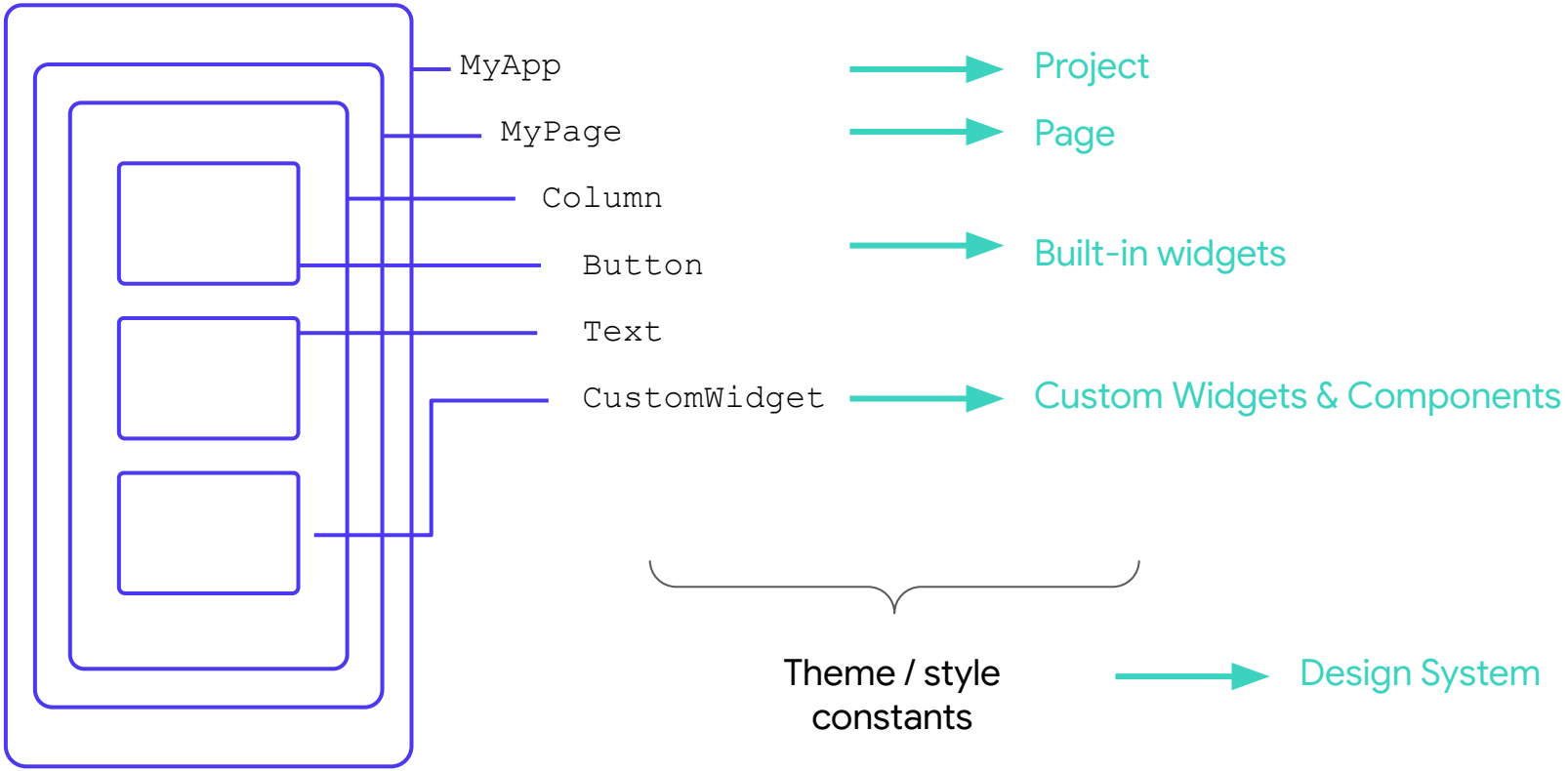
FlutterFlow backend converts to Flutter code

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

Goal for developers

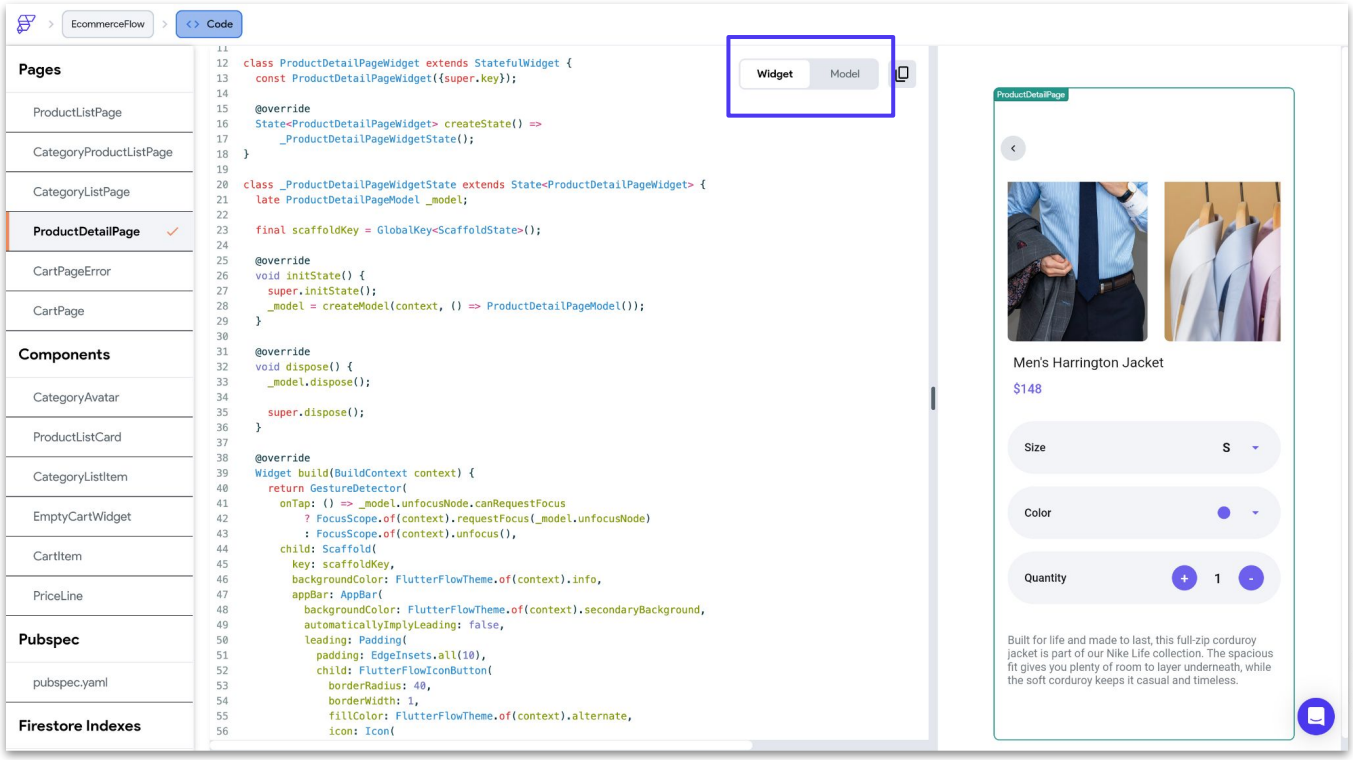


Flutter App Parts to FlutterFlow App Parts



Pages in the generated Code

Separation of UI and data model



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

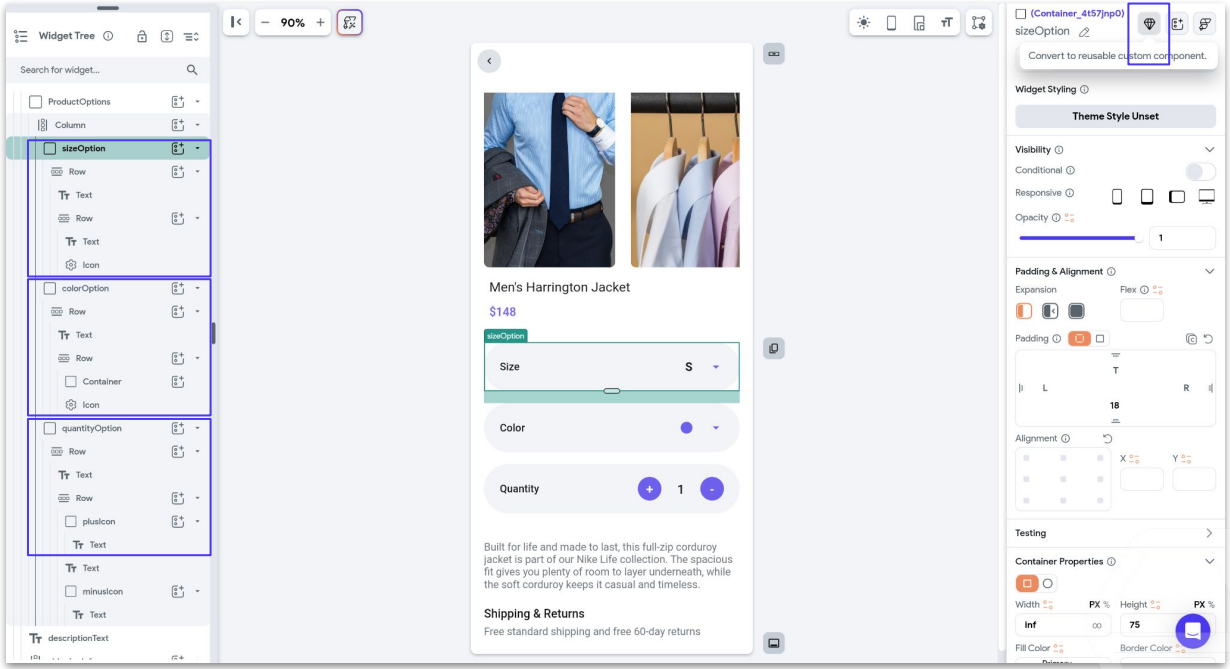
```
class _ProductDetailPageWidgetState extends State<ProductDetailPageWidget> {
 ...
 @override
 Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => _model.unfocusNode.canRequestFocus
        ? FocusScope.of(context).requestFocus(_model.unfocusNode)
        : FocusScope.of(context).unfocus(),
    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

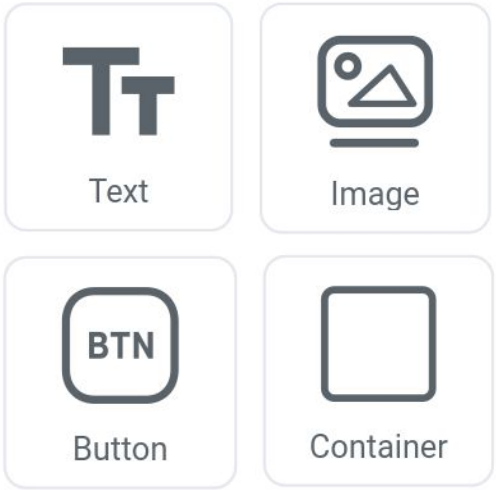
Same principles of encapsulation and reusability apply in FlutterFlow

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



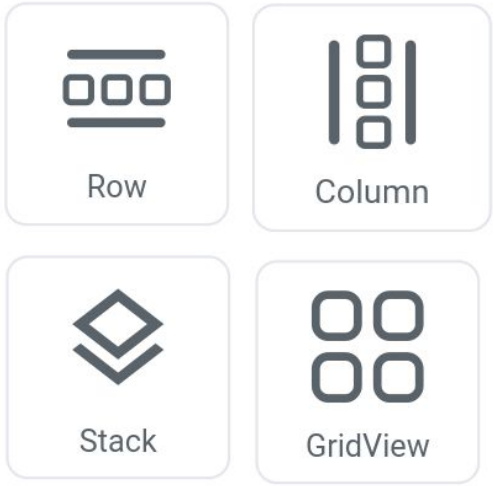
Types of widgets

Base element or single-child widgets



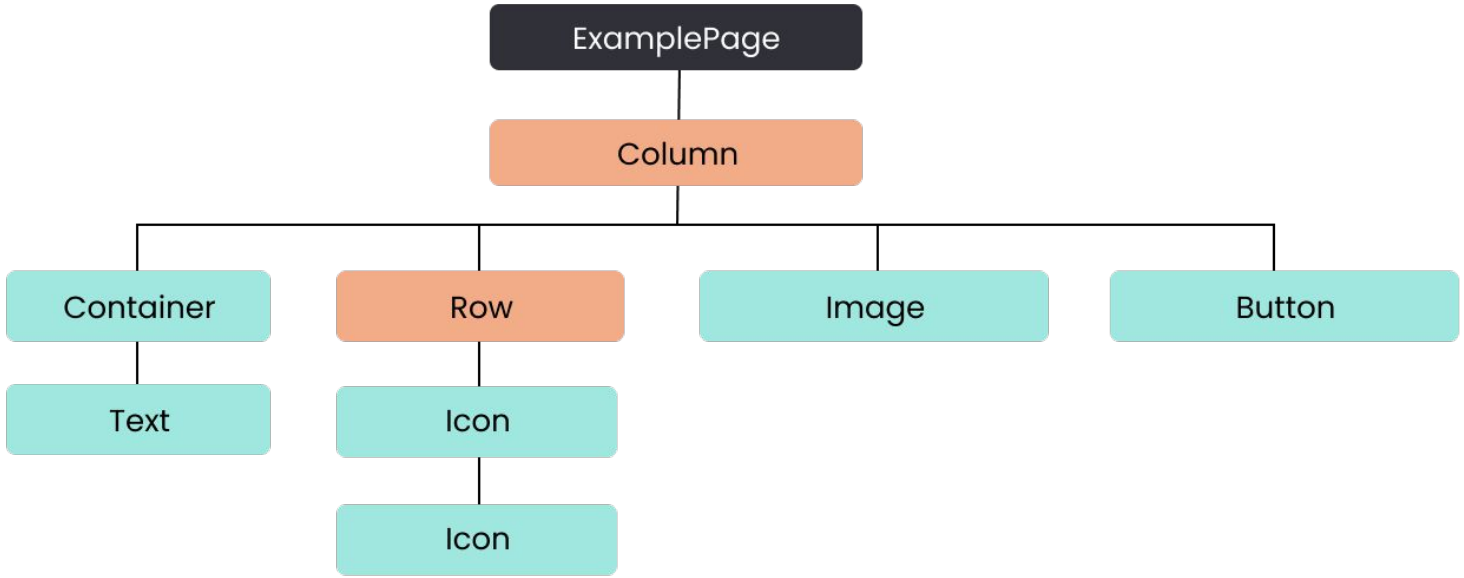
and more..

Multiple-child widgets

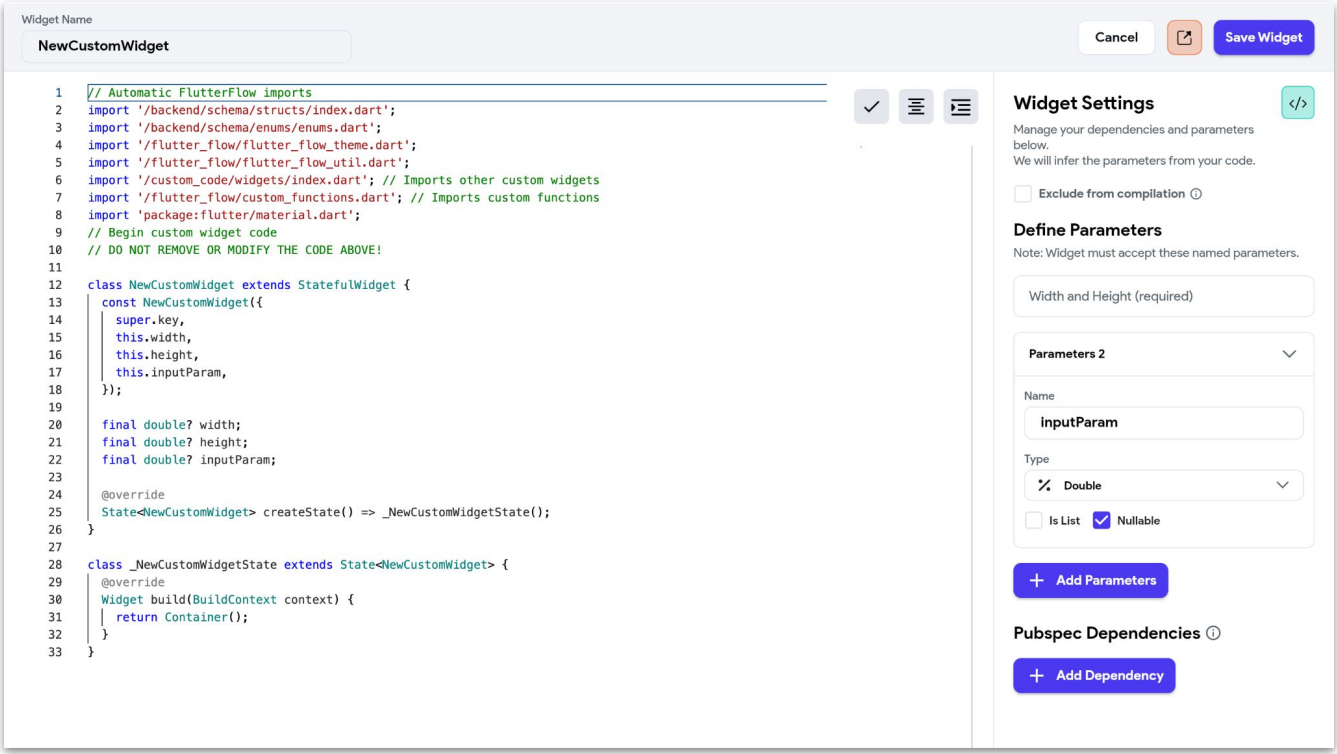


and more..

Widget Tree Hierarchy



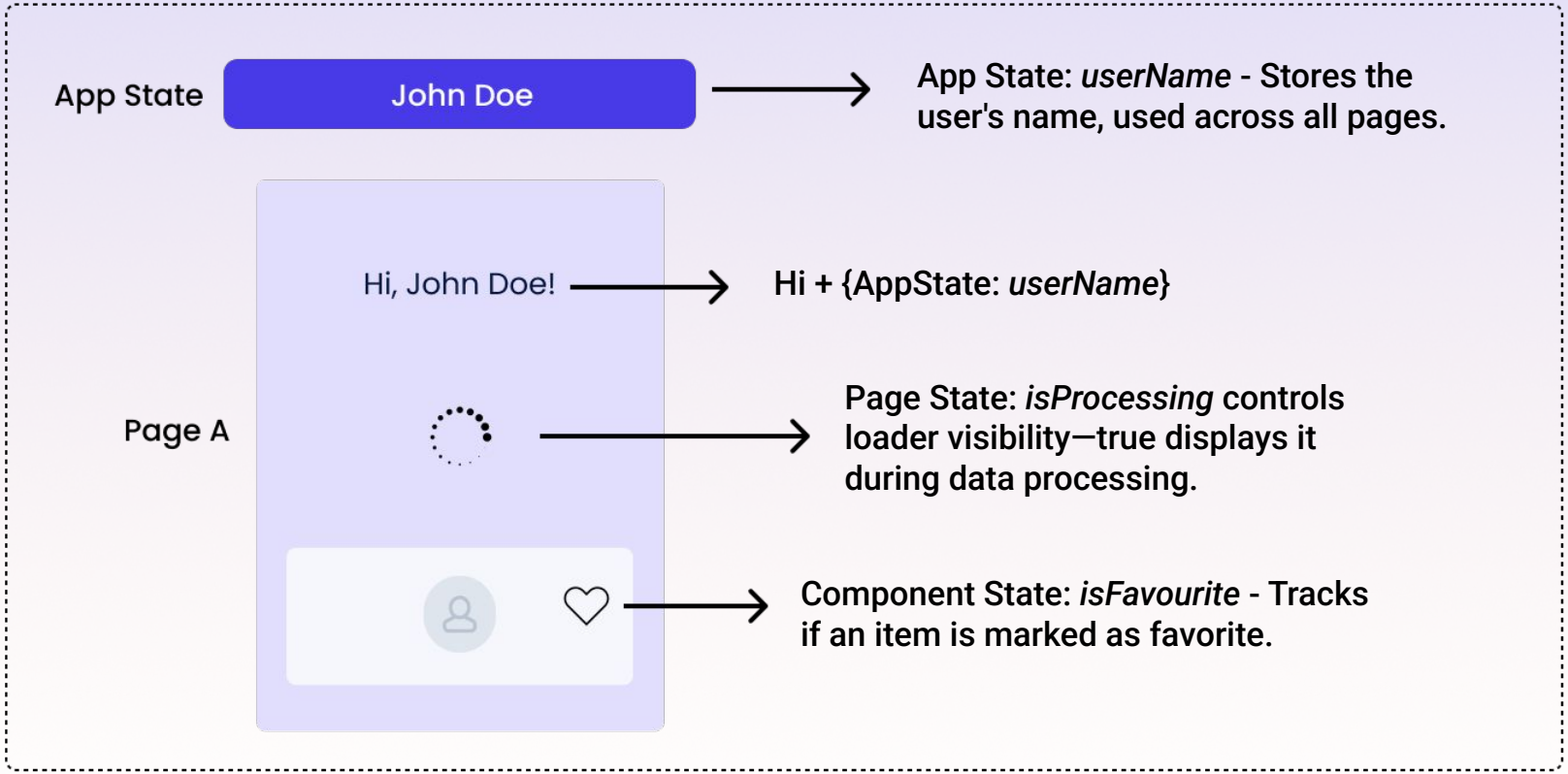
Creating custom widgets

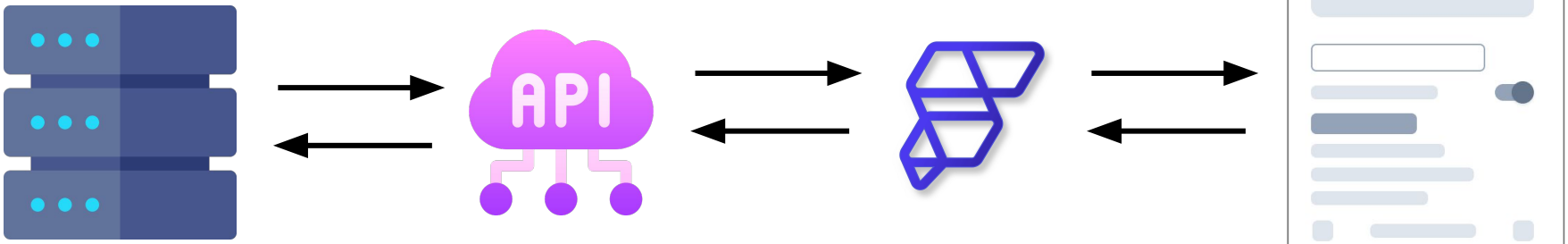


Pass in parameters

Can include a pubspec dependency

State managment





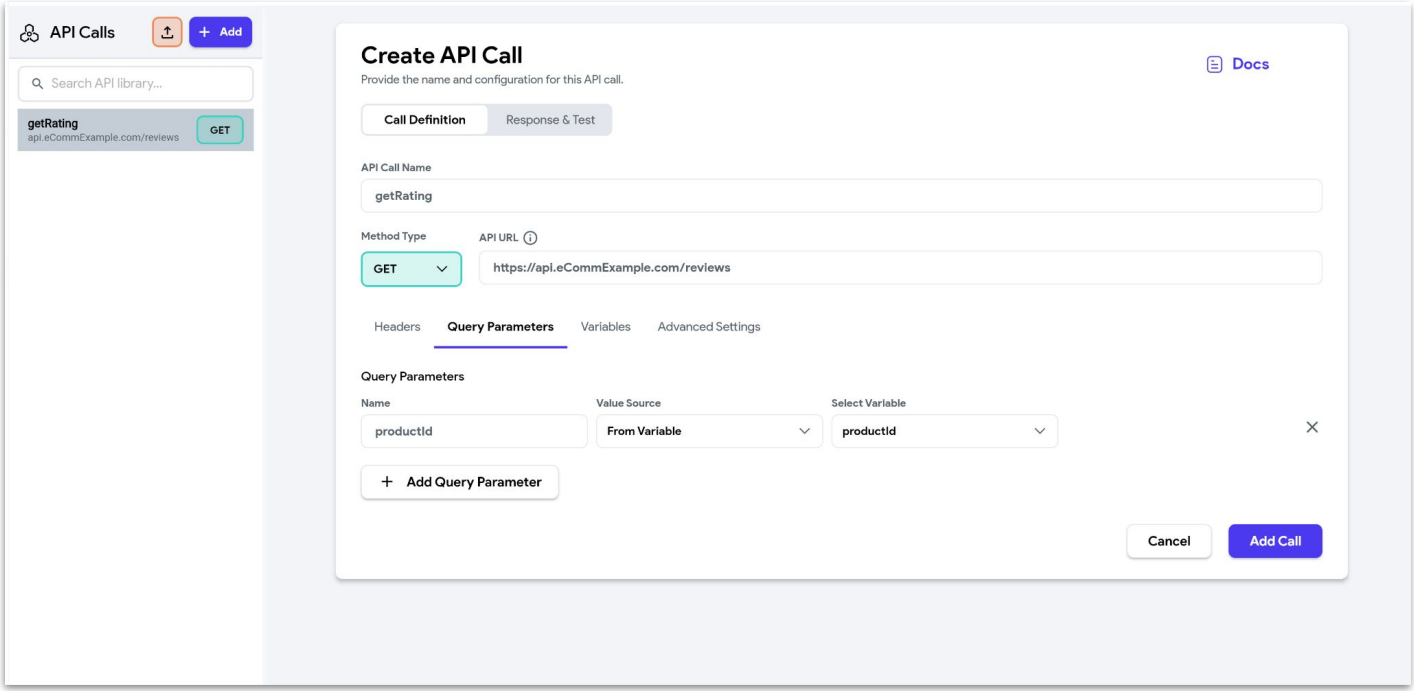
Database
(e.g Firebase, Supabase, own SQL server)

FlutterFlow uses its own API management to connect with APIs in a low-code way

Your FlutterFlow app

Connecting to the backend

Example, calling an API to get product ratings



Use API spec files to generate this config for you!

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

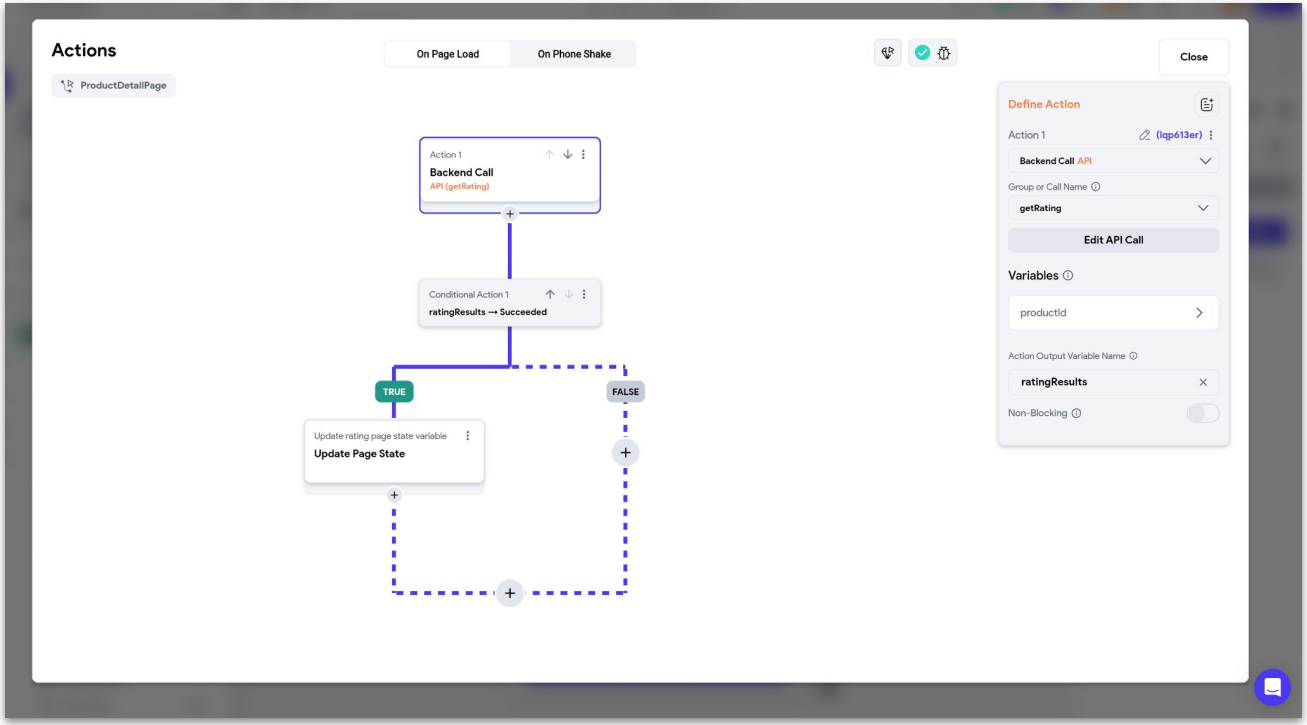
API Call

```
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



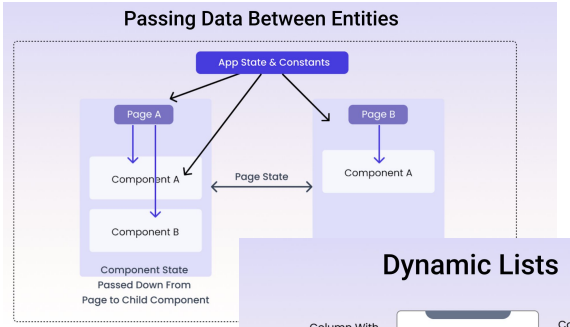
Pass in variables to API call

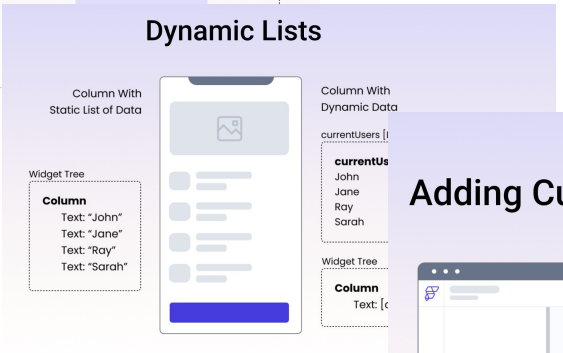
Can parse response into specific data type

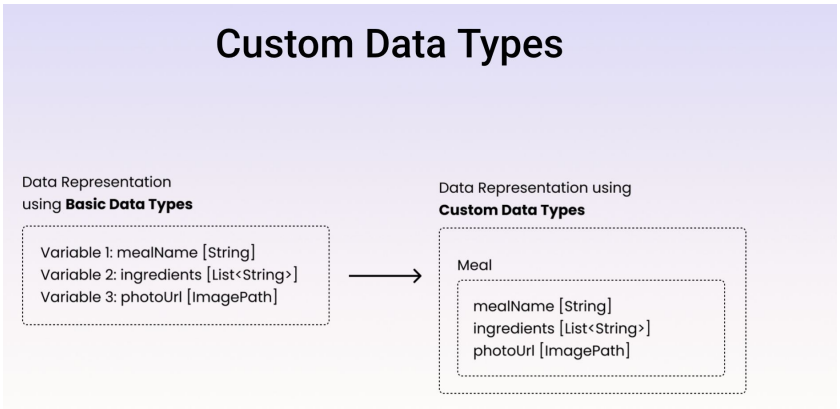
Can handle errors

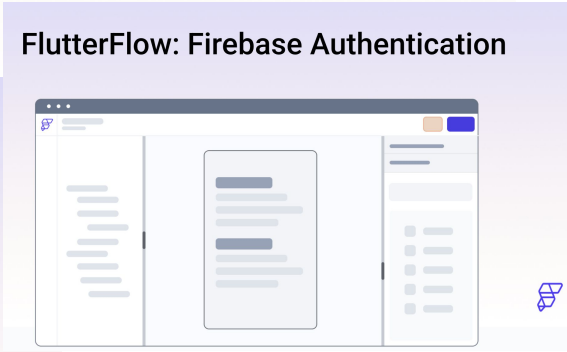
There is much more

But this is not a training session...

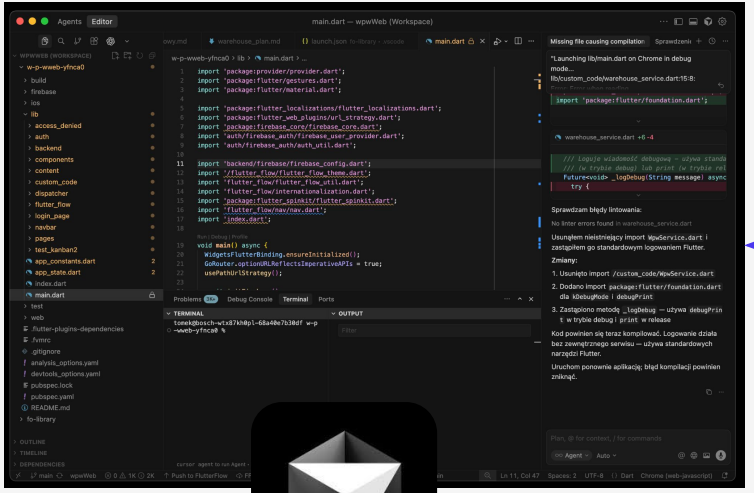


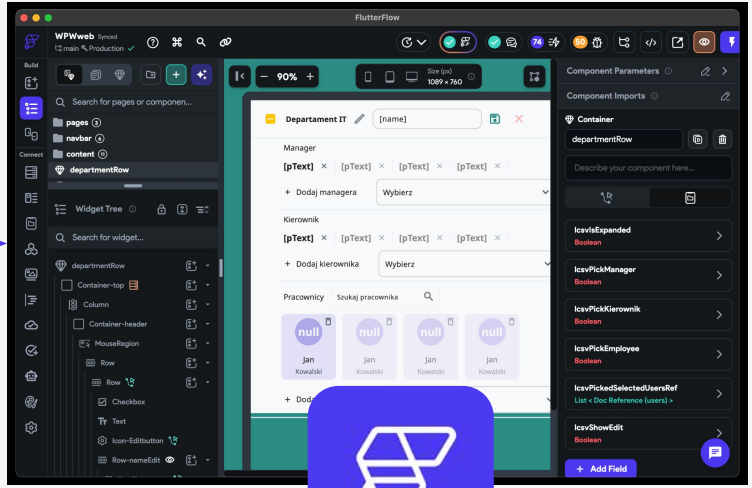


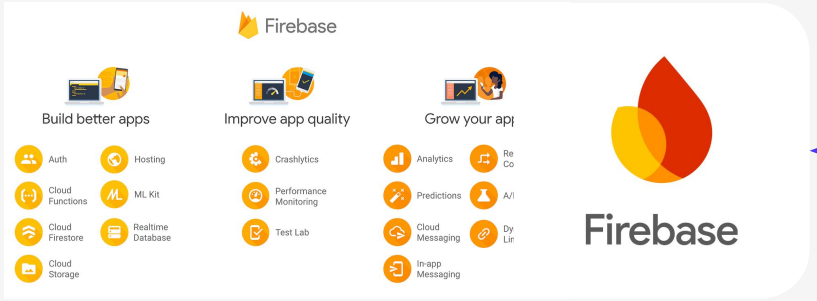




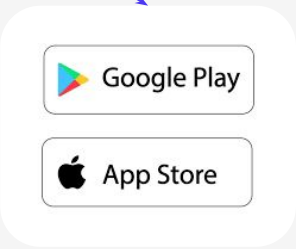
Contents



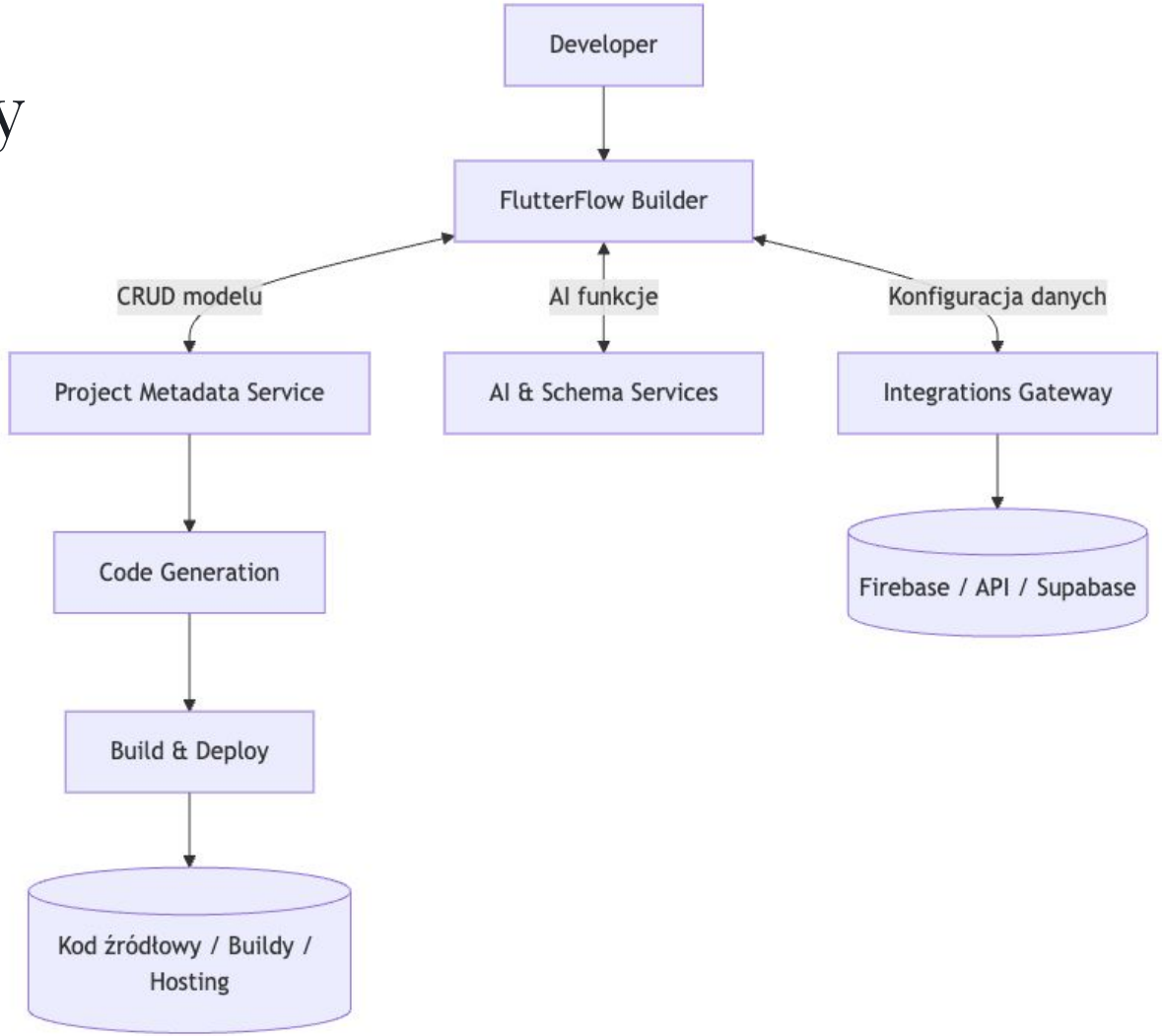



Firebase
Build better apps
Improve app quality
Grow your app
Auth
Hosting
Cloud Functions
ML Kit
Cloud Firestore
Realtime Database
Cloud Storage
Crashlytics
Performance Monitoring
Test Lab
Analytics
Predictions
Cloud Messaging
In-app Messaging
Firebase

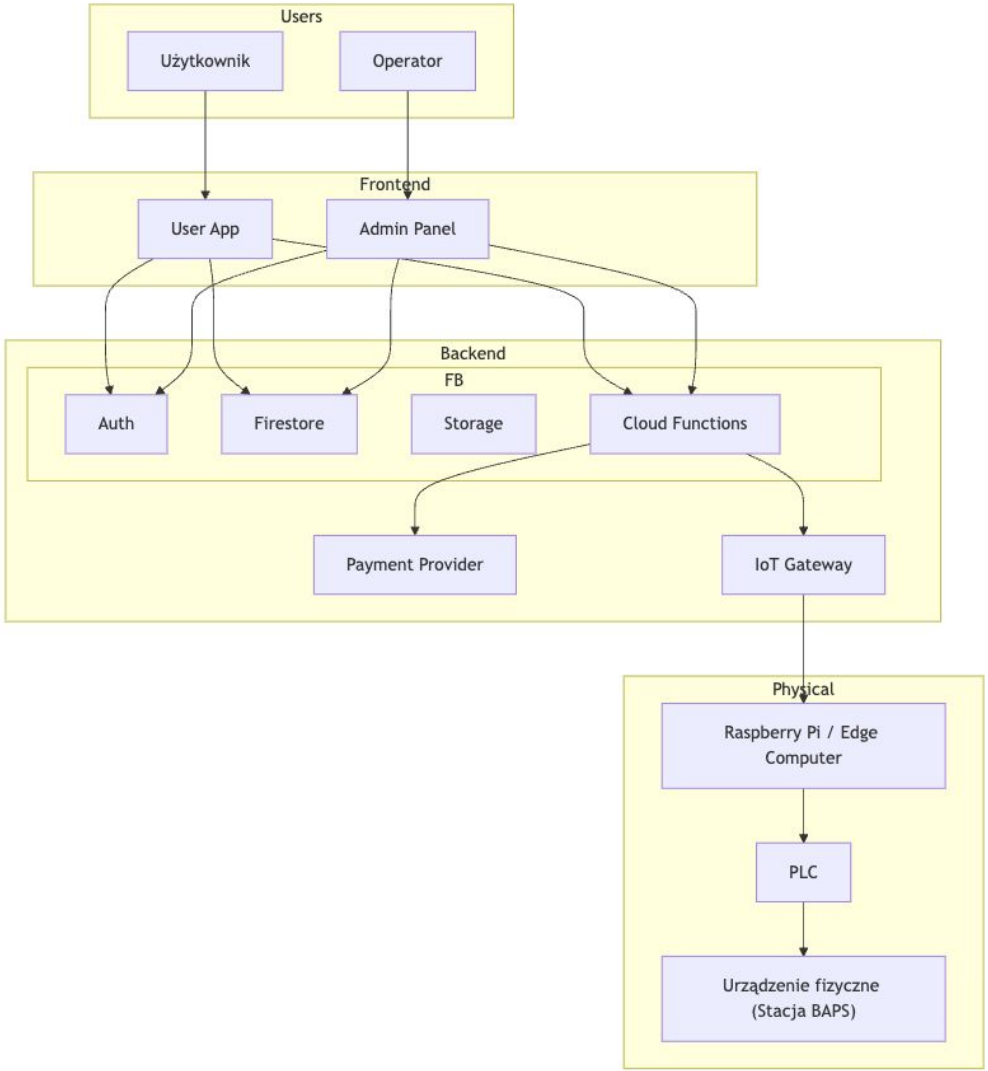

GitHub


Google Play
App Store

Widok uproszczony – kompletny przepływ pracy developera



Przykład projektu IoT
BAPS (kontekst)



Contents

Any questions? daway!

tomasz.kosmider@floworbiter.com