# ValFormulario

# Android Estudio V3.0.17/Flutter

Objetivo:

F.Creación: 1-4-2018

Programa:valformulario

Fuente: [Ver](https://flutter.rocks/2017/10/17/validating-forms-in-flutter/)

[tips](https://github.com/flutter/flutter/blob/master/examples/flutter_gallery/lib/gallery/example_code.dart#L148)

## Pantalla Principal

**import 'package:flutter/material.dart'**;  
**import 'package:valformulario/login\_page.dart'**;  
  
**void** main() {  
 runApp(**new** MyApp());  
}  
  
**class** MyApp **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return new** MaterialApp(  
 title: **'Flutter Demo'**,  
 theme: **new** ThemeData(  
 primarySwatch: Colors.*blue*,  
 ),  
 home: **new** LoginPage(),  
 );  
 }  
}

Se incorpora la pantalla login\_page que va en el mismo directorio, y llama a sus función LoginPage

## LoginPage()

**import 'package:flutter/material.dart'**;  
  
**class** LoginPage **extends** StatefulWidget {  
 @override  
 \_LoginPageState createState() => **new** \_LoginPageState();  
}  
  
**class** \_LoginPageState **extends** State<LoginPage> {  
 **final scaffoldKey** = **new** GlobalKey<ScaffoldState>();  
 **final formKey** = **new** GlobalKey<FormState>();  
  
 String **\_email**;  
 String **\_password**;  
  
 **void** \_submit() {  
 **final** form = **formKey**.**currentState**;  
  
 **if** (form.validate()) {  
 form.save();  
  
 *// Email & password matched our validation rules  
 // and are saved to \_email and \_password fields.* \_performLogin();  
 }  
 }  
  
 **void** \_performLogin() {  
 *// This is just a demo, so no actual login here.* **final** snackbar = **new** SnackBar(  
 content: **new** Text(**'Email:** $**\_email, password:** $**\_password'**),  
 );  
  
 **scaffoldKey**.**currentState**.showSnackBar(snackbar);  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 **return new** Scaffold(  
 key: **scaffoldKey**,  
 appBar: **new** AppBar(  
 title: **new** Text(**'Validating forms'**),  
 ),  
 body: **new** Padding(  
 padding: **const** EdgeInsets.all(16.0),  
 child: **new** Form(  
 key: **formKey**,  
 child: **new** Column(  
 children: [  
 **new** TextFormField(  
 decoration: **new** InputDecoration(labelText: **'Your email'**),  
 validator: (val) =>  
 !val.contains(**'@'**) ? **'Not a valid email.'** : **null**,  
 onSaved: (val) => **\_email** = val,  
 ),  
 **new** TextFormField(  
 decoration: **new** InputDecoration(labelText: **'Your password'**),  
 validator: (val) =>  
 val.**length** < 6 ? **'Password too short.'** : **null**,  
 onSaved: (val) => **\_password** = val,  
 obscureText: **true**,  
 ),  
 **new** RaisedButton(  
 onPressed: \_submit,  
 child: **new** Text(**'Login'**),  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}

Observaciones

Si quisieramos agregar un usuario podriamos

...

// Instance variables in our state class

final formKey = new GlobalKey<FormState>();

String username;

...

new Form(

key: formKey,

child: new TextFormField(

decoration: new InputDecoration(labelText: 'Username'),

validator: (val) => val.isEmpty? 'Username can\'t be empty.' : null,

onSaved: (val) => \_username = val,

),

);

Hay dos cosas interesantes aquí:

* Un validator es un método que recibe el valor actual del campo cuando  [FormFieldState#validate](https://docs.flutter.io/flutter/widgets/FormFieldState/validate.html) es llamado. Este puede retornar una cadena de error o nulo,dependiendo si el campos es válido o no.
* Similarmente, el onSaved parameter recibe el valor actual del campo cuando  [FormFieldState#save](https://docs.flutter.io/flutter/widgets/FormFieldState/save.html) es llamado. Este valor se da cuando todo esta correcto, generalmente se rescata el atributo como en este caso.

En esta ocasión diremos que las key sirven para dar una referencia a nuestros widget

Para validar el formulario  
 **final** form = **formKey**.**currentState**;  
  
 **if** (form.validate()) {  
 form.save();

Llamando a [FormState#validate()](https://docs.flutter.io/flutter/widgets/FormState/validate.html)  llama a cada campo contenido en el formulario y lo valida, si hay errores son desplegados , [FormState#save()](https://docs.flutter.io/flutter/widgets/FormState/save.html)  realiza el mismo proceso pero con los save