Desenvolupament d'aplicacions per a dispositius mòbils

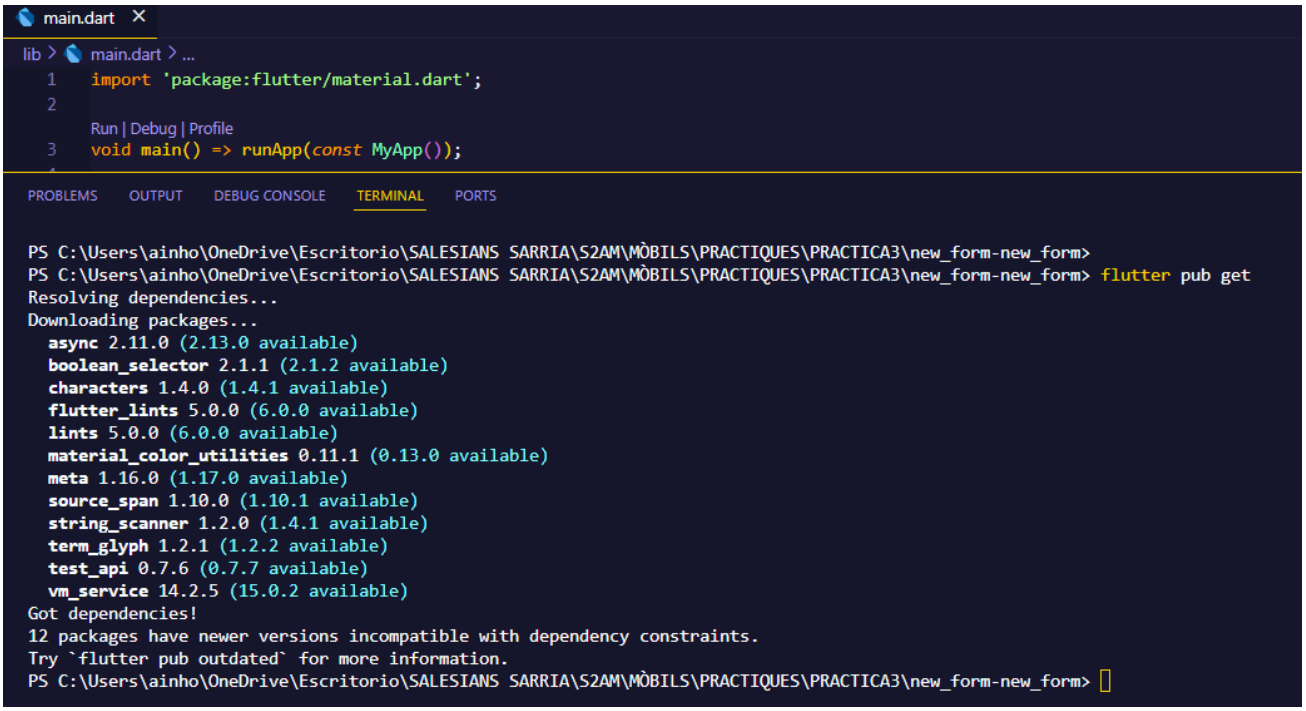
AINHOA SÁNCHEZ SALVAGO

SALESIANS SARRIÀ | 2025-26 | S2AM

ÍNDEX

1. Entrada de dades i visualització amb 4 tipus de diàlegs

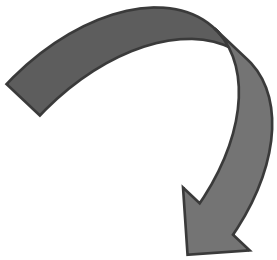
En obrir el codi per primera vegada, salten un Munt d'errors a causa de que falten algunes dependències. Executem a la terminal un "flutter pub get" i ens les instal·la automàticament.



Un cop arreglats tots aquests errors, comencem a editar el codi per poder visualitzar les dades entrades per l'usuari amb 4 tipus de diàlegs diferents, per fer-ho, els hem de ficar en una columna i treure el "FloatingActionButton" que n'hi havia.

1.1. ShowModalBottomSheet

```dart
//Afegim una column --> per ficar tots el botons
child: Column
(
  crossAxisAlignment: CrossAxisAlignment.stretch, //Aixi el boto ocupa tot l'ample de la pagina
  children:
  [
    TextField
    (
      controller: myController,
    ), // TextField
    const SizedBox(height: 20), // Separació entre el input de text i el 1r boto

    ElevatedButton(
    onPressed: () {
      showModalBottomSheet(
          backgroundColor: Colors.transparent,
          context: context,
          builder: (BuildContext context)
          {
            return Container(
                decoration: const BoxDecoration(
                  color: Colors.blueGrey,
                  borderRadius: BorderRadius.only(
                    topLeft: Radius.circular(25),
                    topRight: Radius.circular(25),
                  ), // BorderRadius.only
                ), // BoxDecoration
                height: 200,
                child: Center(
                  child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    mainAxisSize: MainAxisSize.min,
                    children: <Widget>[
                      Text(myController.text),
                      ElevatedButton(
                        onPressed: () => Navigator.of(context).pop(),
                        child: const Text('Tancar BottomSheet'),
                      ) // ElevatedButton
                    ], // <Widget>[]
                  ), // Column
                ) // Center
              ); // Container
```



```dart
              ); // Container
            });
    },
    child: const Icon(Icons.text_fields),
), // ElevatedButton
```

1.2. SimpleDialog

El “show Dialog” és la funció que retorna un “SimpleDialog”, de la mateixa manera que posteriorment també ens retornarà un “alert Dialog”.

```dart
ElevatedButton(
  onPressed: () {
    showDialog(
        context: context,
        builder: (BuildContext context)
        {
          return SimpleDialog(
                backgroundColor: Colors.blueGrey,
                shape: RoundedRectangleBorder(
                  borderRadius: BorderRadius.circular(25)
                ), // RoundedRectangleBorder

                children:
                [
                  Center
                  (
                    child: Column
                    (
                      mainAxisSize: MainAxisSize.min,
                      children:
                      [
                        Text(myController.text),
                        ElevatedButton(
                          onPressed: () => Navigator.of(context).pop(),
                          child: const Text('Tancar Dialog'),
                        ), // ElevatedButton
                      ],
                    ), // Column
                  ), // Center
                ],
              ); // SimpleDialog
        },
    );
  },

  child: const Icon(Icons.text_fields),
), // ElevatedButton
```

1.3. SnackBar

```
ElevatedButton
(
  onPressed: () {
    // Mostrem un SnackBar amb el text introduit
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Valor del camp de text: ${myController.text}'),
        backgroundColor: Colors.blueGrey,
        behavior: SnackBarBehavior.floating,
        duration: const Duration(seconds: 3),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(20),
        ), // RoundedRectangleBorder
        margin: const EdgeInsets.all(16),
      ), // SnackBar
    );
  },
  child: const Icon(Icons.text_fields),
), // ElevatedButton
```

1.4. AlertDialog

```
ElevatedButton
(
  onPressed: () {
    showDialog(
      context: context,
      builder: (BuildContext context)
      {
        return AlertDialog
        (
          backgroundColor: Colors.blueGrey,
          shape: const RoundedRectangleBorder
          (
            borderRadius: BorderRadius.all(Radius.circular(25)),
          ), // RoundedRectangleBorder
          content: Column
          (
            mainAxisSize: MainAxisSize.min,
            children:
            [
              Text
              (
                myController.text,
                style: const TextStyle(color: Colors.white),
              ), // Text
              const SizedBox(height: 20),
              ElevatedButton
              (
                onPressed: () => Navigator.of(context).pop(),
                child: const Text('Tancar AlertDialog'),
              ), // ElevatedButton
            ],
          ), // Column
        ); // AlertDialog
      },
    );
  },
  child: const Icon(Icons.text_fields),
), // ElevatedButton
```

1.5. VÍDEO DEMOSTRACIÓ

https://drive.google.com/file/d/1aPm2lepQWfCsk9Ggsbl0Tmn_hHq3jimi/view?usp=sharing

2. Formularis

2.1. Entendre i comentar el codi de l'exemple del fomulari A

En primer lloc, inicialitzem l'App amb el "runApp" i seguidament, podem veure que tenim el formulari en el "HomePage".

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

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'M0489 - Apps - Form (A)',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
        useMaterial3: true,
      ), // ThemeData
      home: MyHomePage(),
    ); // MaterialApp
  }
}
```

En la continuació del codi, podem veure la declaració de variables, en primer lloc, trobem una string que seria el títol que trobem a l'App Bar. I, en segon lloc, trobem una clau global que serveix per controlar l'estat del formulari.

```dart
class MyHomePage extends StatelessWidget {
  MyHomePage({super.key});
  final String title = 'Ainhoa Sánchez 25/26';
  final _formKey = GlobalKey<FormBuilderState>();
```

D'altra banda, el "FormBuilder" és el "Widget" principal del formulari i la key vincula el formulari amb la clau global declarada anteriorment.

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(title),
    ), // AppBar
    body: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          const FormTitle(),
          FormBuilder(
            key: _formKey,
            child: Padding(
              padding: const EdgeInsets.only(left: 20, right: 20),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
```

Com bé podem veure en el formulari, trobem un seguit d'opcions de les quals amb un "radio button" només podem seleccionar una. Això, ho gestionem de la següent manera, amb el nom identifiquem el camp. Amb les "options", declarem les diferents opcions que volem que l'usuari marqui i, finalment, amb "OnChanged" reaccionem al canvi i ens mostra a la consola l'opció seleccionada.

```
children: [
  //------------------------------------------------
  FormLabelGroup(
    title: 'Please provide the speed of vehicle?',
    subtitle: 'please select one option given below',
  ), // FormLabelGroup
  FormBuilderRadioGroup(
    decoration:
        const InputDecoration(border: InputBorder.none),
    name: "speed",
    orientation: OptionsOrientation.vertical,
    // separator: const Padding(padding: EdgeInsets.all(20)),
    options: const [
      FormBuilderFieldOption(value: 'abvoe 40km/h'),
      FormBuilderFieldOption(value: 'below 40km/h'),
      FormBuilderFieldOption(value: '0km/h')
    ],
    onChanged: (String? value) {
      debugPrint(value);
    },
  ), // FormBuilderRadioGroup
  //------------------------------------------------
```

En el "Form Label Group" trobem el mateix, identificació del camp i una reacció quan l'usuari escriu un text.

```
FormLabelGroup(title: 'Enter remarks'),
FormBuilderTextField(
  name: 'remark',
  decoration: InputDecoration(
    hintText: 'Enter your remarks',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10.0),
      borderSide: const BorderSide(
        width: 0,
        style: BorderStyle.none,
      ), // BorderSide
    ), // OutlineInputBorder
    filled: true,
  ), // InputDecoration
  onChanged: (String? value) {
    debugPrint(value);
  },
```

Seguidament, trobem un menú desplegable el qual ens mostra un conjunt d'opcions. Allà on posa ítems, podem veure com declarem les opcions que s'obren en el menú en obrir-lo. I com hem vist durant tota l'estona, també reaccionem al canvi amb el "On Change".

```
//-----------------------------------------------
FormLabelGroup(
  title: 'Please provide the high speed of vehicle?',
  subtitle: 'please select one option given below',
), // FormLabelGroup
FormBuilderDropdown(
  name: 'highspeed',
  decoration: InputDecoration(
    hintText: 'Select option',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10.0),
    ), // OutlineInputBorder
  ), // InputDecoration
  items: const [
    DropdownMenuItem(value: 'high', child: Text('High')),
    DropdownMenuItem(
        value: 'medium', child: Text('Medium')), // DropdownMenuItem
    DropdownMenuItem(value: 'low', child: Text('Low')),
  ],
  onChanged: (String? value) {
    debugPrint(value);
  },
), // FormBuilderDropdown
//-----------------------------------------------
```

Ara, el següent que trobem és un grup de “CheckBoxes”, on declarem una sèrie d’opcions que són de selecció múltiple, és a dir, permeten seleccionar més d’una l’hora.

```
FormLabelGroup(
  title: 'Please provide the speed of vehicle past 1 hour?',
  subtitle: 'please select one or more options given below',
), // FormLabelGroup
FormBuilderCheckboxGroup(
  decoration:
      const InputDecoration(border: InputBorder.none),
  name: "selectSpeed",
  orientation: OptionsOrientation.vertical,
  // separator: const Padding(padding: EdgeInsets.all(20)),
  options: const [
    FormBuilderFieldOption(value: '20km/h'),
    FormBuilderFieldOption(value: '30km/h'),
    FormBuilderFieldOption(value: '40km/h'),
    FormBuilderFieldOption(value: '50km/h'),
  ],
  onChanged: (List<String>? value) {
    debugPrint(value.toString());
  },
), // FormBuilderCheckboxGroup
```

Ara, fora ja de la columna, però encara dins de l’estructura bàsica de la pantalla(Scaffold), afegim un “floatingActionButton”. Aquest guarda i mostra les dades del formulari. Amb el “saveAndValidate()” guarda i valida tots els camps. Seguidament, obtenim les dades del formulari i els guardem a la variable “formString” i els mostrem mitjançant un “alertDialog”.

```
floatingActionButton: FloatingActionButton(
    child: Icon(Icons.upload),
    onPressed: () {
      _formKey.currentState?.saveAndValidate();
      String? formString = _formKey.currentState?.value.toString();
      alertDialog(context, formString!);
    }), // FloatingActionButton
; // Scaffold
```

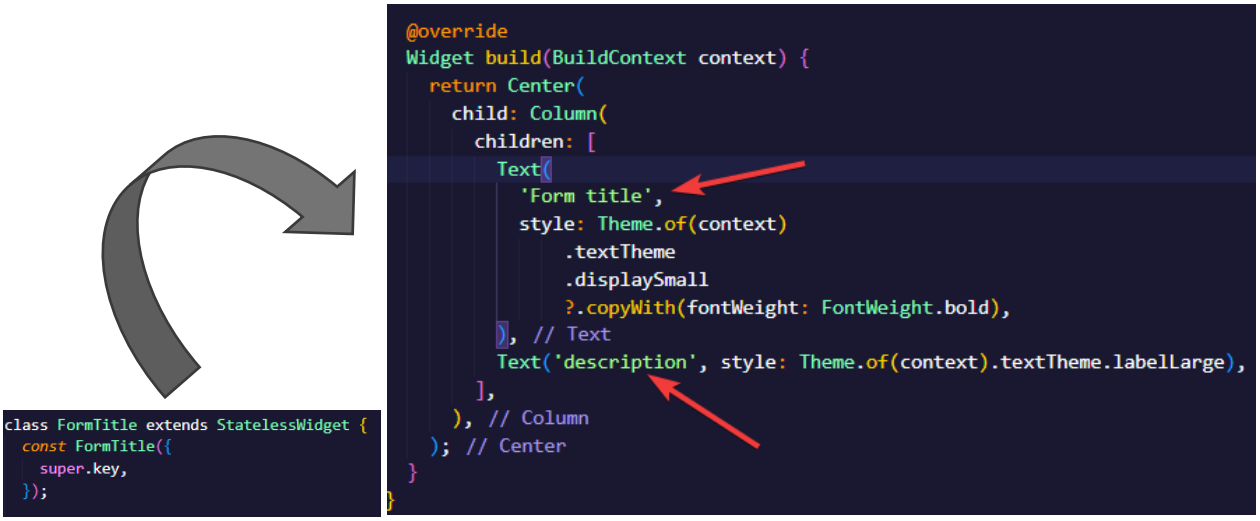
La següent classe del codi és un "widget" reutilitzable per mostrar el títol i subtítol de cada bloc del formulari. Llavors, allà on veiem un "conditionalSubtitle", comprova que hi ha un subtítol i el mostra amb un estil secundari. En canvi, si no hi ha retorna un contenidor buit.

```
// ignore: must_be_immutable
class FormLabelGroup extends StatelessWidget {
  FormLabelGroup({super.key, required this.title, this.subtitle});

  String title;
  String? subtitle;

  Widget conditionalSubtitle(BuildContext context) {
    if (subtitle != null) {
      return Text(subtitle!,
          style: Theme.of(context).textTheme.labelLarge?.apply(
              fontSizeFactor: 1.25,
              // ignore: deprecated_member_use
              color: Theme.of(context).colorScheme.secondary.withOpacity(0.5)));
    } else {
      return Container();
    }
  }
```

Tot seguit, mostrem el títol i podem veure que cridem a la funció de "conditionalSubtitle" per tal de mostrar el subtítol si aquest existeix.

```
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.fromLTRB(0, 20.0, 0, 8),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Text(title,
            style: Theme.of(context)
                .textTheme
                .labelLarge
                ?.apply(fontSizeFactor: 1.25)), // Text
        conditionalSubtitle(context)
      ],
```

A continuació, trobem un "Widget" que ens mostra l'encapçalament principal del formulari. Ho mostrem de la següent manera, on el "form title" és el títol principal del formulari i on la descripció és el subtítol o descripció del formulari.



```dart
class FormTitle extends StatelessWidget {
  const FormTitle({
    super.key,
  });
```

```dart
@override
Widget build(BuildContext context) {
  return Center(
    child: Column(
      children: [
        Text(
          'Form title',
          style: Theme.of(context)
              .textTheme
              .displaySmall
              ?.copyWith(fontWeight: FontWeight.bold),
        ), // Text
        Text('description', style: Theme.of(context).textTheme.labelLarge),
      ],
    ), // Column
  ); // Center
}
}
```

Finalment, arribem al final del codi on trobem una funció que ens mostra en un quadre de diàleg, el resultat del formulari i un botó perquè l'usuari pugui tancar-lo quan hagi acabat de llegir el missatge.

```dart
void alertDialog(BuildContext context, String contentText) {
  showDialog<String>(
    context: context,
    builder: (BuildContext context) => AlertDialog(
      title: const Text("Submission Completed"),
      icon: const Icon(Icons.check_circle),
      content: Text(contentText),
      actions: <Widget>[
        TextButton(
          onPressed: () => Navigator.pop(context, 'Tancar'),
          child: const Text('Tancar'),
        ), // TextButton
      ], // <Widget>[]
    ), // AlertDialog
  );
}
```

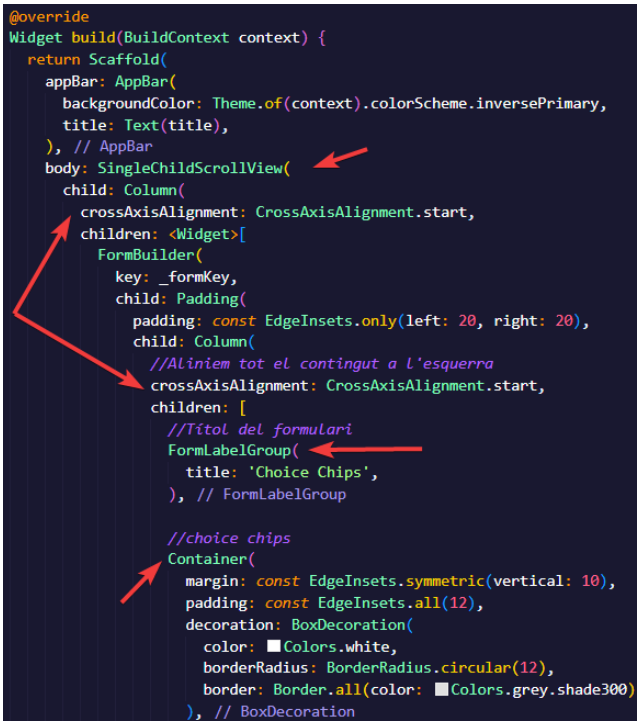
3. Implementació de formularis

3.1. Formulari C

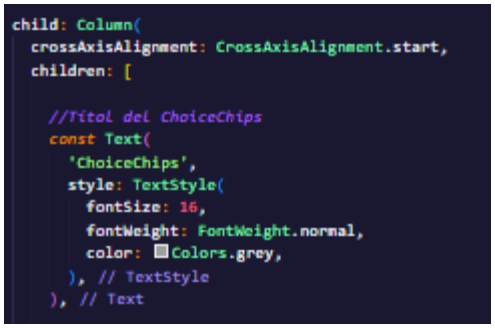
3.1.1. Choice Chips

Per començar a executar el codi del formulari C, he seguit d'exemple el codi del formulari B el qual ja hem comentat anteriorment. Afegim, en primer lloc, dins l'estructura principal del codi una barra superior amb el títol. En segon lloc, afegim el cos del nostre formulari on hem anat creant les noves funcionalitats i aspecte que volem que tingui aquest formulari, si aquest és molt llarg, amb el "SingleChildScrollView" fem que sigui possible fer scroll a la pantalla. En tercer lloc, trobem el formulari principal del projecte ("FormBuilder"). Finalment, comencem a

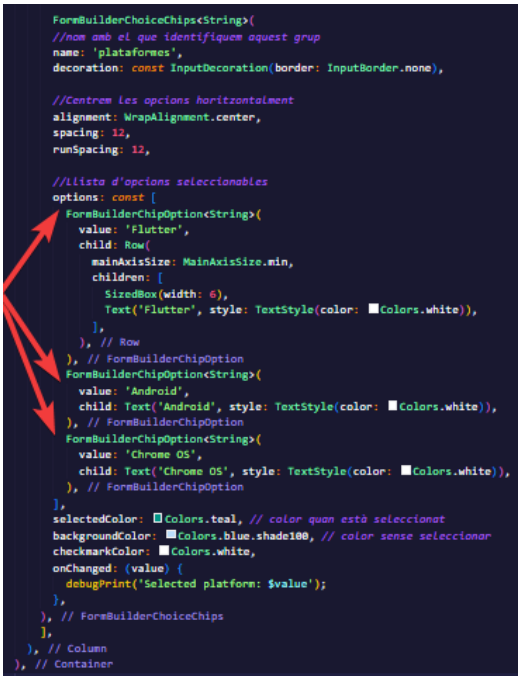
implementar la primera part de la nostra pantalla, és a dir, comencem amb els ChoiceChips i el primer que fem és crear el seu estil.



Dins d'aquest contenidor afegim una columna la qual conté el títol que volem que ens aparegui dins i definim l'estil que volem que tingui.



Un cop fet això, creem el contingut principal del container col·locant un grup de "choiceChips" seleccionables, per fer-ho, afegim un "FormBuilderChoiceChips". Ara, dins d'aquest creem una llista amb les opcions que l'usuari pot seleccionar, cadascuna d'aquestes opcions tenen un "value" que, cada cop que una està seleccionada, el guardem i amb el mètode de "OnChanged" mostrem cada cop que se selecciona una de les opcions el valor a la consola (opcional).



3.1.2. Switch

Continuem amb el segon container, per començar definim la decoració que volem que aquest tingui i l'espaiat o marges.

```dart
//Container pel switch
Container(
//Marges i decoració del container
margin: const EdgeInsets.symmetric(vertical: 10),
padding: const EdgeInsets.all(16),
decoration: BoxDecoration(
  color: Colors.white,
  borderRadius: BorderRadius.circular(12),
  border: Border.all(color: Colors.grey.shade300),
), // BoxDecoration
```

Un cop acabat això, definim el títol del switch i quin estil volem que tingui.

```dart
//Titol del switch
const Text(
  'Switch',
  style: TextStyle(
    fontSize: 16,
    fontWeight: FontWeight.normal,
    color: Colors.grey,
  ), // TextStyle
), // Text

const SizedBox(height: 12),
```

Per tal de crear el Switch, afegim un "FormBuilderSwitch", dins d'aquest definim el nom amb què volem identificar-lo i també, el text que volem que es mostri al costat del switch. Finalment, amb el mètode OnChanged, guardem el valor (si està activat o no) i ho mostrem a la consola.

```
//Switch
FormBuilderSwitch(
  name: 'switch',
  title: const Text(
    'This is a switch',
    style: TextStyle(
      fontSize: 16,
    ), // TextStyle
  ), // Text
  initialValue: false,
  activeColor: Colors.teal,
  onChanged: (value) {
    debugPrint('Switch changed: $value');
  },
), // FormBuilderSwitch
],
), // Column
), // Container
```

3.1.3. Text Field

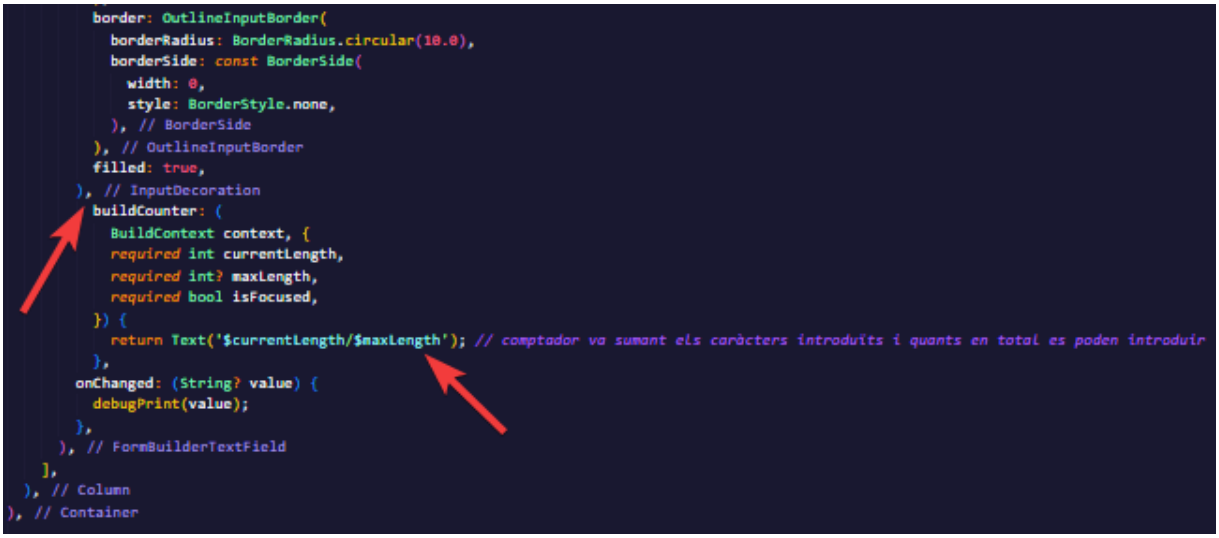
En tercer lloc, creem ara el "TextField Container". Com en els containers anteriors, definim primer l'estil i decoració que volem que tingui. I tot seguit, afegim el títol que volem que ens aparegui dins del contenidor.

```
//Container pel Textfield
Container(
  //Marges i decoració del container
  margin: const EdgeInsets.symmetric(vertical: 10),
  padding: const EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Colors.grey.shade300),
  ), // BoxDecoration

  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [

      //Títol del Textfield
      const Text(
        'Text Field',
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.normal,
          color: Colors.grey,
        ), // TextStyle
      ), // Text

      const SizedBox(height: 12),
```

A continuació, afegim un límit de caràcters i declarem el text que volem que ens aparegui dins del "input" on l'usuari escriu el que vol i amb quin estil volem que es mostri.

```
//Textfield
FormBuilderTextField(
  name: 'remark',
  // Límit de caràcters
  maxLength: 15,
  decoration: InputDecoration(
    prefix: const Text
    (
      'A',
      style: TextStyle
      (
        color: Colors.grey,
        decoration: TextDecoration.underline, //Text Subratllat
        fontWeight: FontWeight.bold,
      ), // TextStyle
    ), // Text
```

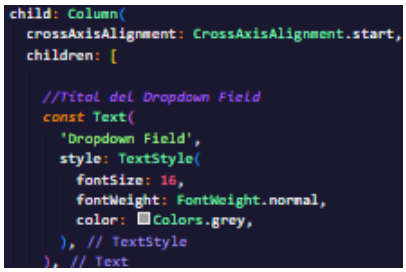
Ara, afegim el "border" que volem que tingui el "TextField". Seguidament, trobem un comptador de caràcters dinàmics. Finalment, amb el mètode OnChanged, cada cop que canvia el valor del camp el mostra per consola.



3.1.4. DropDown Field

Com en tots els "containers" anteriors, declarem, en primer lloc, l'estil i espaiat que volem que tingui.

Seguidament, definim el títol que volem que tingui dins el nostre contenidor.



Finalment, per acabar amb aquest contenidor, creen el "FormBuilderDropDown". En primer lloc, definim l'estil i després, creem la llista d'opcions de les quals l'usuari pot seleccionar la que vulgui. I amb el mètode "OnChanged" cada cop que el valor canviï el mostrarà per consola.

```
      const SizedBox(height: 12),
      FormBuilderDropdown<String>(
        name: 'DropDown', // Nom per identificar el camp
        decoration: InputDecoration(
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
          ), // OutlineInputBorder
        ), // InputDecoration
        items: const [
          DropdownMenuItem(value: 'Flutter', child: Text('Flutter')),
          DropdownMenuItem(value: 'Android', child: Text('Android')),
          DropdownMenuItem(value: 'iOS', child: Text('iOS')),
          DropdownMenuItem(value: 'Web', child: Text('Web')),
        ],
        onChanged: (String? value) {
          debugPrint('Selected value: $value');
        },
      ), // FormBuilderDropdown
    ],
  ), // Column
), // Container
```

3.1.5. VIDEO DEMOSTRACIÓ

https://drive.google.com/file/d/1o6ZTpV1jgYBqA6_QYNqE7Gjlf0EZxSMM/view?usp=drive_link

3.2. Formulari D

3.2.1. Autocomplete

Per realitzar aquest formulari, he seguit d'exemple l'estructura que hem utilitzat anteriorment per fer el formulari C. En primer lloc, creem una llista per a poder dur a terme el primer contenidor, en aquest, en escriure se'ns mostren les opcions autocompletant-se. Creem, doncs, un "container" al que li assignem un estil en concret.

```
// creem una llista de països per a l'autocompletat
const List<String> paisos = [
  'Austria', 'Belgium', 'Bulgaria', 'Croatia', 'Cyprus', 'Czech Republic', 'Denmark',
  'Estonia', 'Finland', 'France','Germany', 'Greece','Hungary', 'Ireland', 'Italy',
  'Latvia', 'Lithuania', 'Luxembourg', 'Malta', 'Netherlands', 'Poland', 'Portugal',
  'Romania', 'Slovakia', 'Slovenia', 'Spain', 'Sweden', 'United Kingdom',
];
```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Colors.teal,
      title: Text(title),
    ), // AppBar
    body: SingleChildScrollView(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          FormBuilder(
            key: _formKey,
            child: Padding(
              padding: const EdgeInsets.only(left: 20, right: 20),
              child: Column(
                //Alinem tot el contingut a l'esquerra
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [

                  //Container per Autocompletat de paisos
                  Container(
                    margin: const EdgeInsets.symmetric(vertical: 10),
                    padding: const EdgeInsets.all(16),
                    decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.circular(12),
                      border: Border.all(color: Colors.grey.shade300),
                    ), // BoxDecoration
```

Seguidament, creem un text (títol dins del contenidor) i definim l'estil que volem que adopti. A més, afegim un "FormBuilderState" el qual ens permet afegir un "widget" d'autocompletat, dins d'aquest, filtrem la llista de països segons el text introduït per l'usuari i si està buit el camp mostra tota la llista. Quan l'usuari ha seleccionat un, aquest valor es guarda i el mostra per consola. Finalment, definim l'estil que volem que tingui el "Text Field" intern.

```dart
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children:
  [

    const Text(
      'Autocomplete',
      style: TextStyle(
        fontSize: 16,
        fontWeight: FontWeight.normal,
        color: Colors.grey,
      ), // TextStyle
    ), // Text

    const SizedBox(height: 12),

    FormBuilderField<String>(
      name: 'paisos',
      builder: (FormFieldState<String?> field) {
        return Autocomplete<String>( // Widget d'autocompletat
          optionsBuilder: (TextEditingValue textEditingValue) {
            // Filtra la llista de països segons el text introduït per l'usuari i si esta buit mostra tota la llista
            if (textEditingValue.text.isEmpty) {
              return paisos;
            }
            return paisos.where((country) =>
                country.toLowerCase().contains(textEditingValue.text.toLowerCase()));
          },
          //Guarda el valor seleccionat al formulari i mostra per consola
          onSelected: (value) {
            field.didChange(value); // Guarda el valor al formulari
            debugPrint('Selected country: $value');
          },
          fieldViewBuilder: (context, controller, focusNode, onFieldSubmitted) {
            //Builder per personalitzar el TextField intern de l'Autocompletat
            return TextField(
              controller: controller,
              focusNode: focusNode,
              decoration: InputDecoration(
                labelText: 'Autocomplete',
                border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),
                filled: true,
              ), // InputDecoration
              onSubmitted: (value) => onFieldSubmitted(),
            ); // TextField
          },
        ); // Autocomplete
      },
    ), // FormBuilderField
  ],
), // Column
), // Container
```

3.2.2. Date Picker

A continuació, el següent contenidor és un "DatePicker" amb què l'usuari podrà escollir una data. Per començar, hem definit com sempre, l'estil que volem que tingui.

```
//Container pel Date Picker
Container(
  margin: const EdgeInsets.symmetric(vertical: 10),
  padding: const EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Colors.grey.shade300),
  ), // BoxDecoration
```

Seguidament, creem una columna on establim un Text, és a dir, un títol que anirà dins del container i, a l'hora, definim l'estil que volem.

```
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [

    const Text(
      'Date Picker',
      style: TextStyle(
        fontSize: 16,
        fontWeight: FontWeight.normal,
        color: Colors.grey,
      ), // TextStyle
    ), // Text

    const SizedBox(height: 12),
```

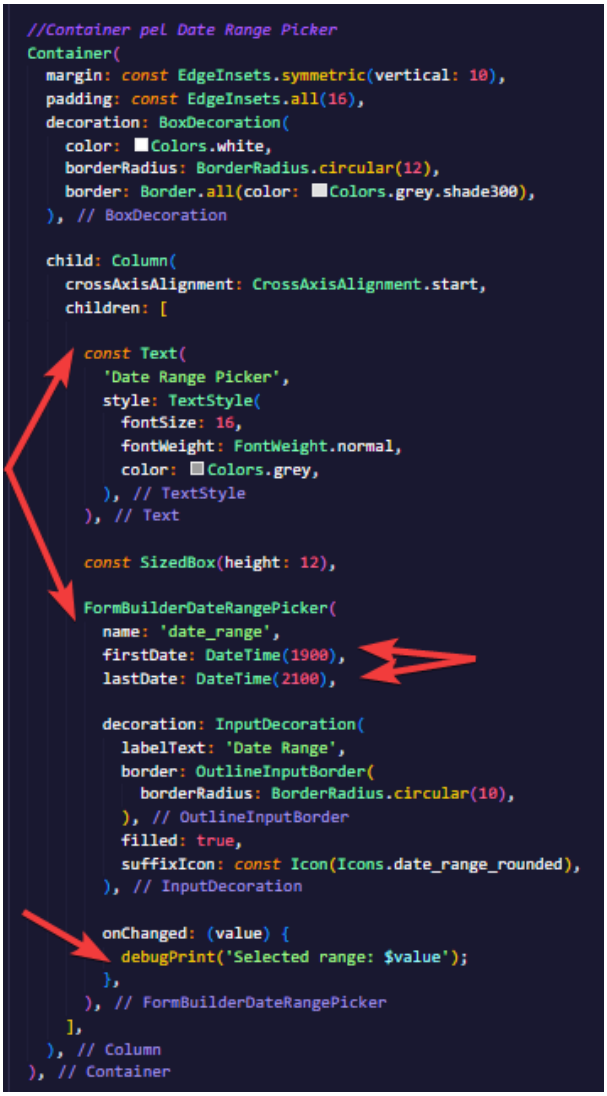
Per acabar, afegim dins del nostre contenidor, un "FormBuilderDateTimePicker". Volem que només se seleccioni la data, sense hora. També, afegim uns límits de dates per l'usuari i, finalment, amb el mètode "OnChanged", quan el valor canvia el mostra per consola.

```
    const SizedBox(height: 12),

    FormBuilderDateTimePicker(
      name: 'date',
      inputType: InputType.date, // Només data, sense hora
      decoration: InputDecoration(
        labelText: 'Date Picker',
        border: OutlineInputBorder(
          borderRadius: BorderRadius.circular(10),
        ), // OutlineInputBorder
        filled: true,

        //Afegim una icona de calendari al final del camp
        suffixIcon: const Icon(Icons.calendar_month_rounded),
      ), // InputDecoration
      firstDate: DateTime(1900),
      lastDate: DateTime(2100),
      onChanged: (value) {
        debugPrint('Selected date: $value');
      },
    ), // FormBuilderDateTimePicker
  ],
), // Column
), // Container
```

3.2.3. Date Range

Ara, a continuació, creem un altre contenidor, però aquest per afegir dins un títol i un "widget" que ens permet seleccionar un rang de dates. Establim un límit de dates entre les

quals l'usuari pot escollir el rang. Finalment, el mètode on change controla quan canvia el valor i el mostra per pantalla.



3.2.4. Time Picker

Seguim la mateixa estructura per afegir el "FormBuilderDateTimePicker" que hem utilitzat per a DatePicker, l'única cosa que canvia és que ara no volem només la data sinó tot el contrari, volem només l'hora.

```dart
//Container pel Time Picker
Container(
  margin: const EdgeInsets.symmetric(vertical: 10),
  padding: const EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Colors.grey.shade300),
  ), // BoxDecoration

  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [

      const Text(
        'Time Picker',
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.normal,
          color: Colors.grey,
        ), // TextStyle
      ), // Text

      const SizedBox(height: 12),

      FormBuilderDateTimePicker(
        name: 'time',
        inputType: InputType.time, // només L'hora
        decoration: InputDecoration(
          labelText: 'Time Picker',
          border: OutlineInputBorder(
            borderRadius: BorderRadius.circular(10),
          ), // OutlineInputBorder
          filled: true,
          suffixIcon: const Icon(Icons.access_time_rounded), // Icone del rellotge
        ), // InputDecoration

        onChanged: (value) {
          debugPrint('Selected time: $value');
        },
      ), // FormBuilderDateTimePicker
    ],
  ), // Column
), // Container
```

3.2.5. InputChips

Per a seleccionar una opció entre d'altres, hem utilitzat el "FormBuilderChoiceChip".

```dart
//Container pels Input Chips
Container(
  margin: const EdgeInsets.symmetric(vertical: 10),
  padding: const EdgeInsets.all(16),
  decoration: BoxDecoration(
    color: Colors.white,
    borderRadius: BorderRadius.circular(12),
    border: Border.all(color: Colors.grey.shade300),
  ), // BoxDecoration

  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [

      const Text(
        'InputChips',
        style: TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.normal,
          color: Colors.grey,
        ), // TextStyle
      ), // Text

      const SizedBox(height: 12),

      FormBuilderChoiceChips<String>(
        name: 'tech_skills',
        decoration: const InputDecoration(
          border: InputBorder.none,
        ), // InputDecoration
        options: const [
          FormBuilderChipOption(value: 'HTML'),
          FormBuilderChipOption(value: 'CSS'),
          FormBuilderChipOption(value: 'React'),
          FormBuilderChipOption(value: 'Dart'),
          FormBuilderChipOption(value: 'TypeScript'),
          FormBuilderChipOption(value: 'Angular'),
        ],
        onChanged: (value) {
          debugPrint('Selected chips: $value');
        },

        selectedColor: Colors.teal,
      ), // FormBuilderChoiceChips
    ],
  ), // Column
), // Container
```

3.2.6. VIDEO DEMOSTRACIÓ

https://drive.google.com/file/d/1byKfyZVJ114wugNLz_diyyolhtv1SCME/view?usp=drive_link