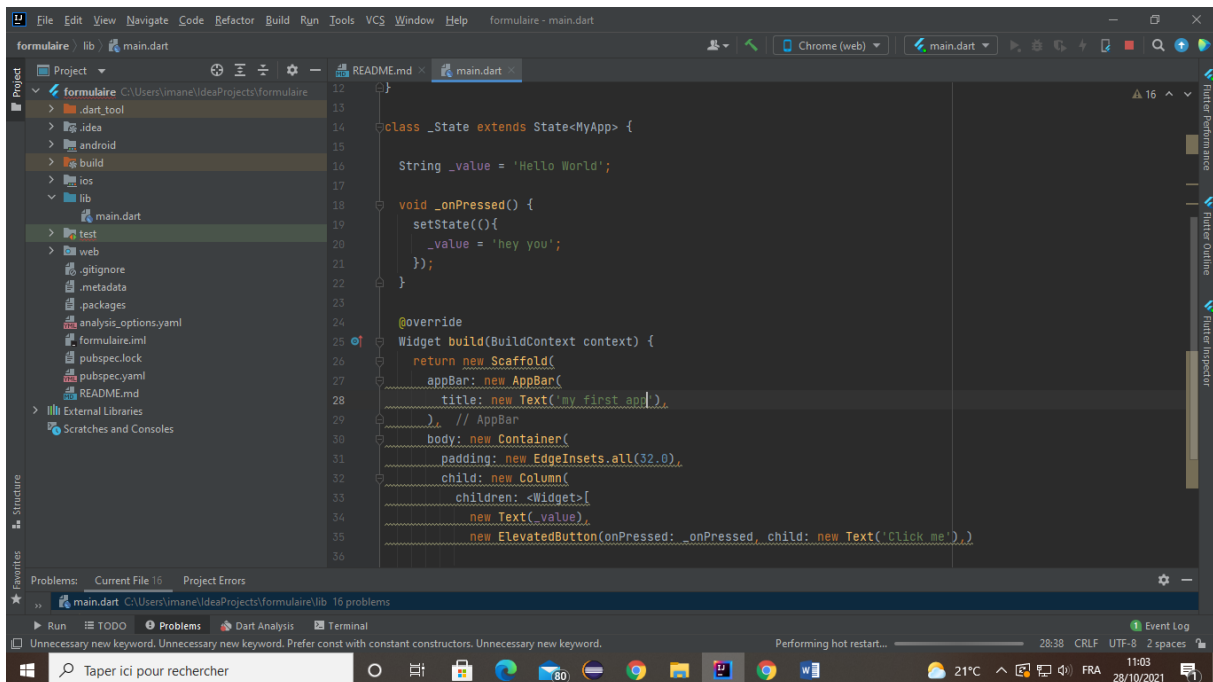


Formulaire

Étape 1 : Boutons

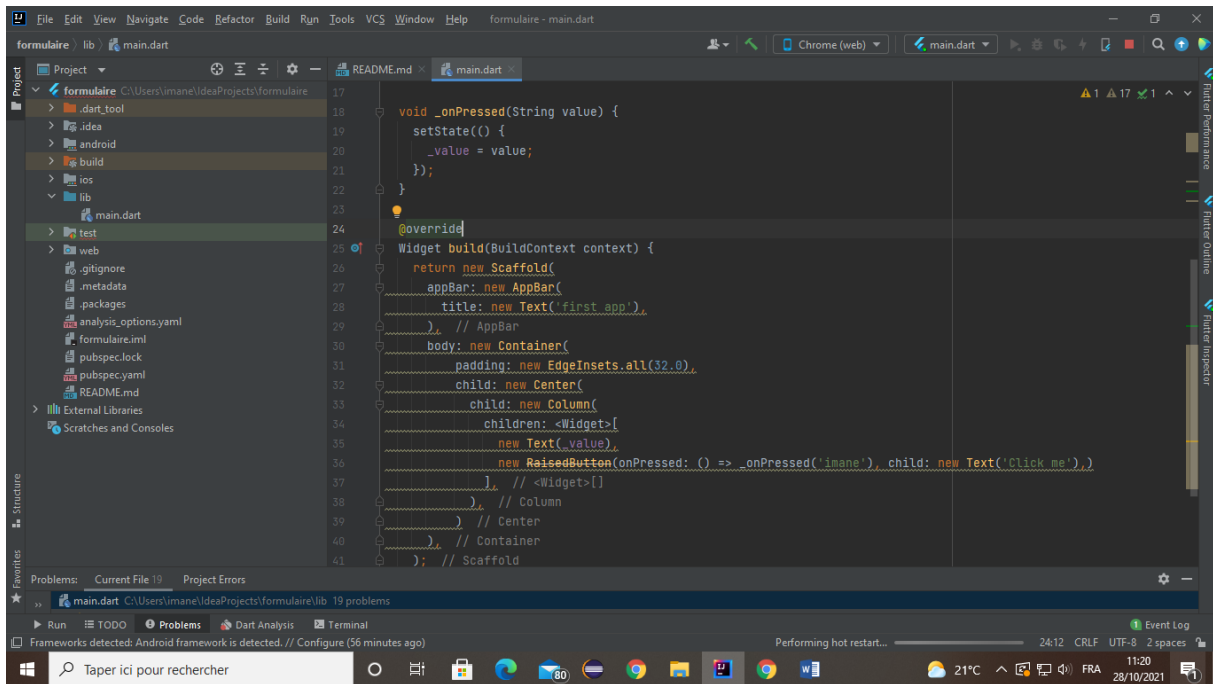
RaisedButtons/ ElevatedButtons

Pour créer notre bouton, on fait appel à la classe `ElevatedButton` qui prend en argument la fonction `onPressed` déjà définie.



```
class _State extends State<MyApp> {  
  String _value = 'Hello World';  
  
  void _onPressed() {  
    setState(){  
      _value = 'hey you';  
    };  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return new Scaffold(  
      appBar: new AppBar(  
        title: new Text('my first app'),  
      ), // AppBar  
      body: new Container(  
        padding: new EdgeInsets.all(32.0),  
        child: new Column(  
          children: <Widget>[  
            new Text(_value),  
            new ElevatedButton(onPressed: _onPressed, child: new Text('Click me')),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

La fonction `onPressed()` vous permet de détecter l'état de votre bouton. Dans ce cas il va nous afficher un message 'hey you'. Si vous voulez ajouter un traitement fait par le bouton c'est là qu'il faut l'ajouter.

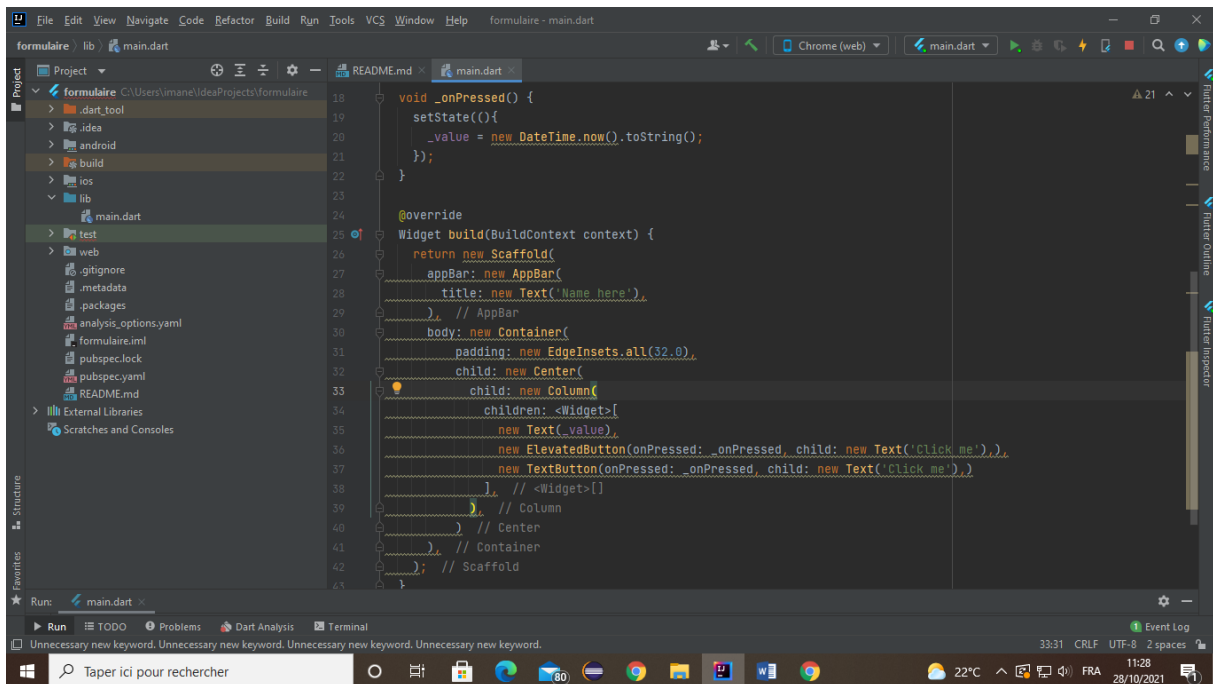


```

17
18 void _onPressed(String value) {
19   setState(() {
20     _value = value;
21   });
22 }
23
24 @override
25 Widget build(BuildContext context) {
26   return new Scaffold(
27     appBar: new AppBar(
28       title: new Text('first app'),
29     ), // AppBar
30     body: new Container(
31       padding: new EdgeInsets.all(32.0),
32       child: new Center(
33         child: new Column(
34           children: <Widget>[
35             new Text(_value),
36             new RaisedButton(onPressed: () => _onPressed('imane'), child: new Text('Click me!')),
37           ], // <Widget>[]
38         ), // Column
39       ), // Center
40     ), // Container
41   ); // Scaffold
  
```

TextButton :

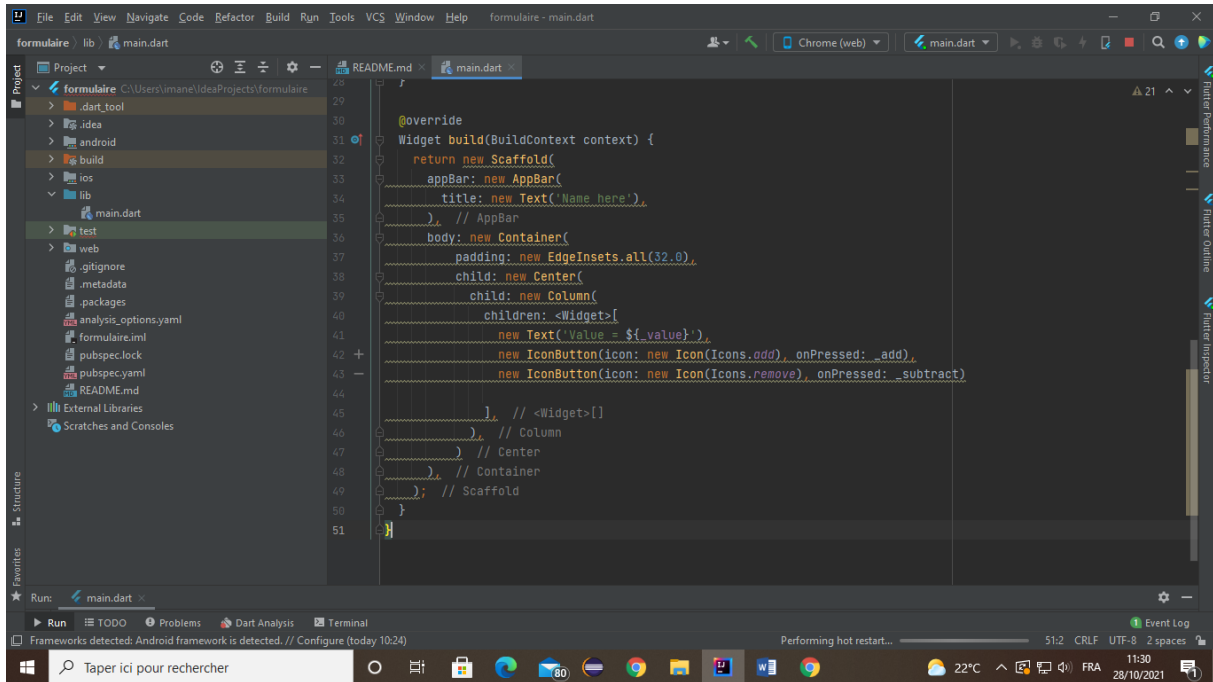
De la meme manière on peut créer un Textbutton.



```

18 void _onPressed() {
19   setState(){
20     _value = new DateTime.now().toString();
21   });
22 }
23
24 @override
25 Widget build(BuildContext context) {
26   return new Scaffold(
27     appBar: new AppBar(
28       title: new Text('Name here'),
29     ), // AppBar
30     body: new Container(
31       padding: new EdgeInsets.all(32.0),
32       child: new Center(
33         child: new Column(
34           children: <Widget>[
35             new Text(_value),
36             new ElevatedButton(onPressed: _onPressed, child: new Text('Click me!')),
37             new TextButton(onPressed: _onPressed, child: new Text('Click me!')),
38           ], // <Widget>[]
39         ), // Column
40       ), // Center
41     ), // Container
42   ); // Scaffold
  
```

IconButtons :



```

28
29
30
31 @override
32 Widget build(BuildContext context) {
33   return new Scaffold(
34     appBar: new AppBar(
35       title: new Text('Name here!'),
36     ), // AppBar
37     body: new Container(
38       padding: new EdgeInsets.all(32.0),
39       child: new Center(
40         child: new Column(
41           children: <Widget>[
42             new Text('Value = $_value'),
43             new IconButton(icon: new Icon(Icons.add), onPressed: _add),
44             new IconButton(icon: new Icon(Icons.remove), onPressed: _subtract)
45           ], // <Widget>[]
46         ), // Column
47       ), // Center
48     ), // Container
49   ); // Scaffold
50 }
51

```

Créez les deux fonctions `_add` et `_subtract` afin d'incrémenter ou décrémenter la valeur de `_value`.

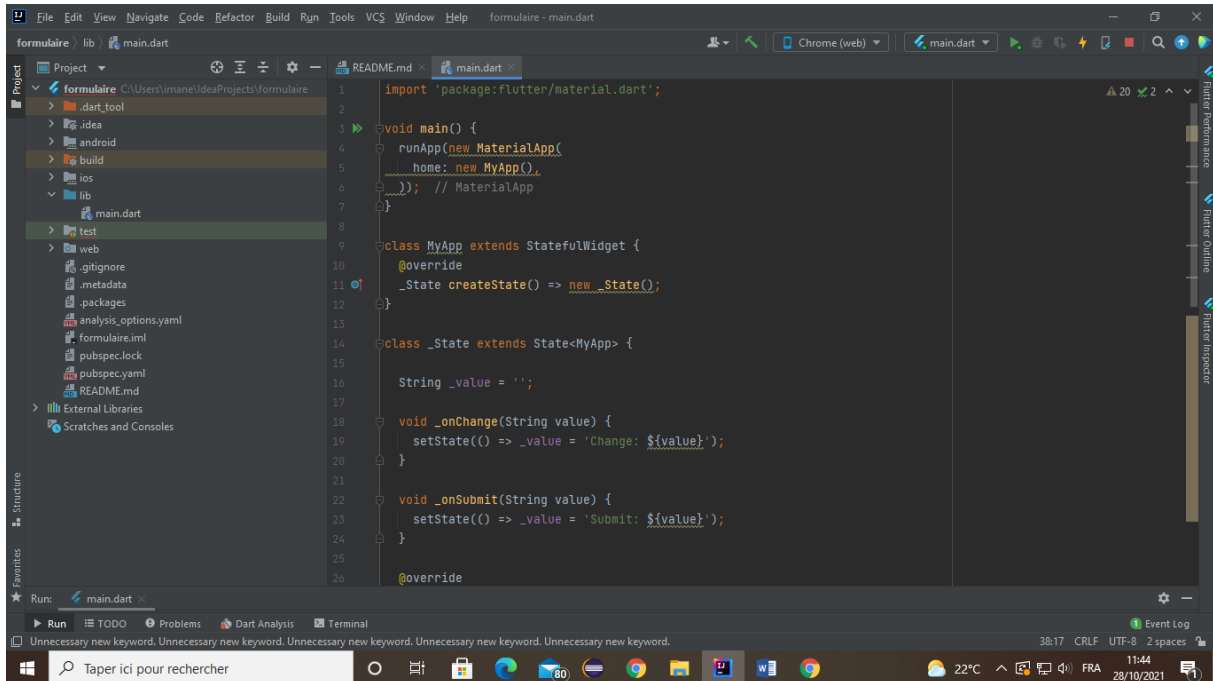
Etape 2 : Champs de texte :

TextField est une classe qui vous permet de créer un champ de texte.

Les TextField ont deux fonctions qui vont avec :

`onChange()` : quand le contenu du champ change

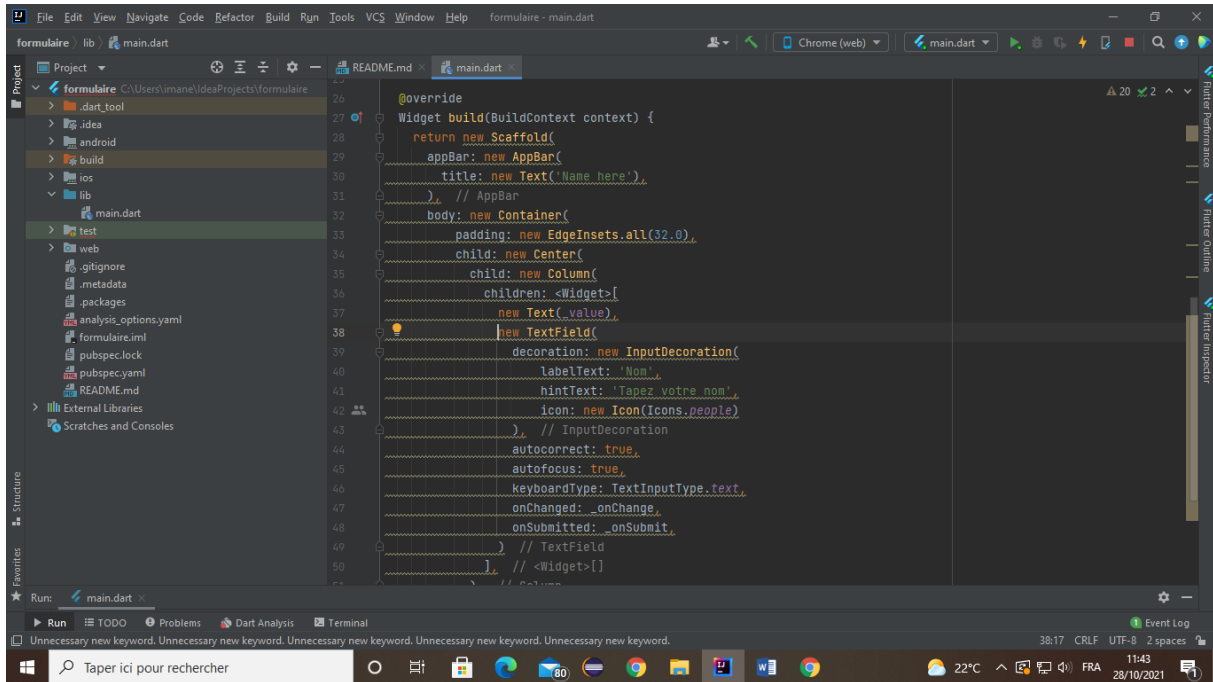
`onSubmit()` : quand le contenu du champ est soumis.



```

1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(new MaterialApp(
5     home: new MyApp(),
6   )); // MaterialApp
7 }
8
9 class MyApp extends StatefulWidget {
10   @override
11   _State createState() => new _State();
12 }
13
14 class _State extends State<MyApp> {
15
16   String _value = '';
17
18   void _onChange(String value) {
19     setState(() => _value = 'Change: ${value}');
20   }
21
22   void _onSubmit(String value) {
23     setState(() => _value = 'Submit: ${value}');
24   }
25 }
26
27 @override

```



```

@override
Widget build(BuildContext context) {
  return new Scaffold(
    appBar: new AppBar(
      title: new Text('Name here'),
    ), // AppBar
    body: new Container(
      padding: new EdgeInsets.all(32.0),
      child: new Center(
        child: new Column(
          children: <Widget>[
            new Text(_value),
            new TextField(
              decoration: new InputDecoration(
                labelText: 'Nom',
                hintText: 'Tapez votre nom',
                icon: new Icon(Icons.people)
              ), // InputDecoration
              autocorrect: true,
              autofocus: true,
              keyboardType: TextInputType.text,
              onChanged: _onChange,
              onSubmitted: _onSubmit,
            ) // TextField
          ], // <Widget>[]
        )
      )
    )
  );
}

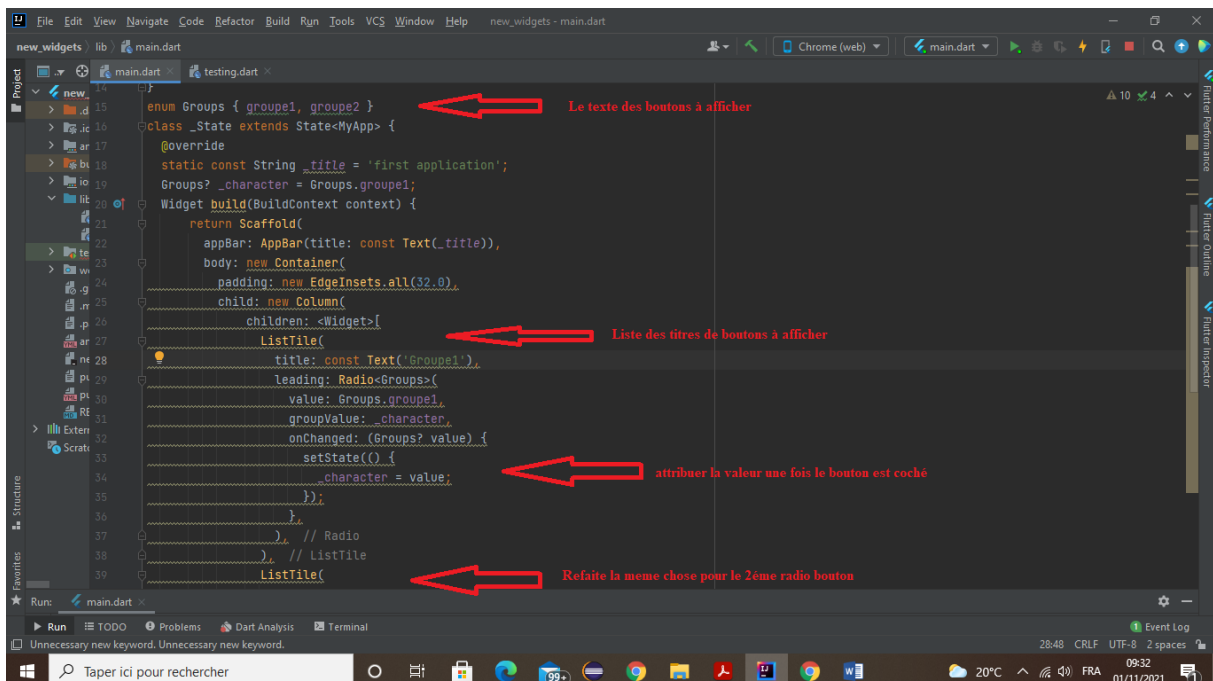
```

Exercice :

Réalisez un formulaire à remplir.

Etape 3 : Radio buttons

Concernant les radios buttons, il faut déclarer une liste qui va s'afficher.



```

enum Groups { groupe1, groupe2 }

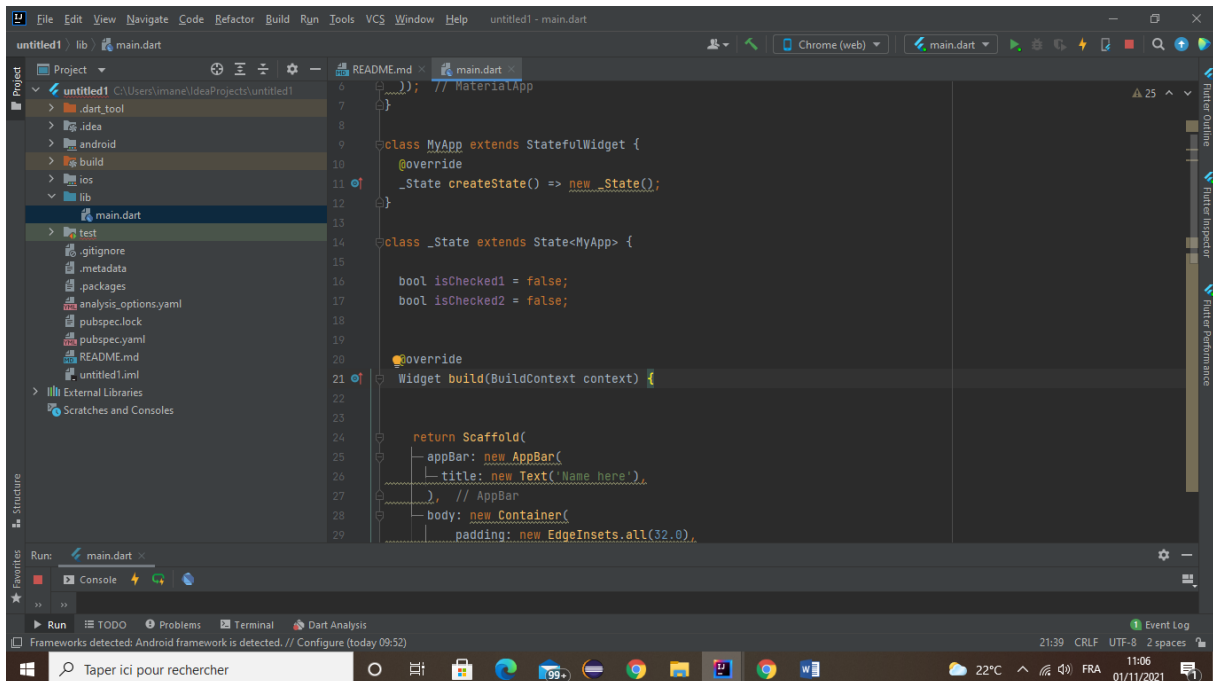
class _State extends State<MyApp> {
  @override
  static const String _title = 'first application';
  Groups? _character = Groups.groupe1;

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text(_title)),
      body: new Container(
        padding: new EdgeInsets.all(32.0),
        child: new Column(
          children: <Widget>[
            ListTile(
              title: const Text('Groupe1'),
              leading: Radio<Groups>(
                value: Groups.groupe1,
                groupValue: _character,
                onChanged: (Groups? value) {
                  setState(() {
                    _character = value;
                  });
                },
              ), // Radio
            ), // ListTile
          ],
        )
      )
    );
  }
}

```

- CheckBox

On déclare deux variables qui vont représenter l'état des checkboxes. On les initialise à false.



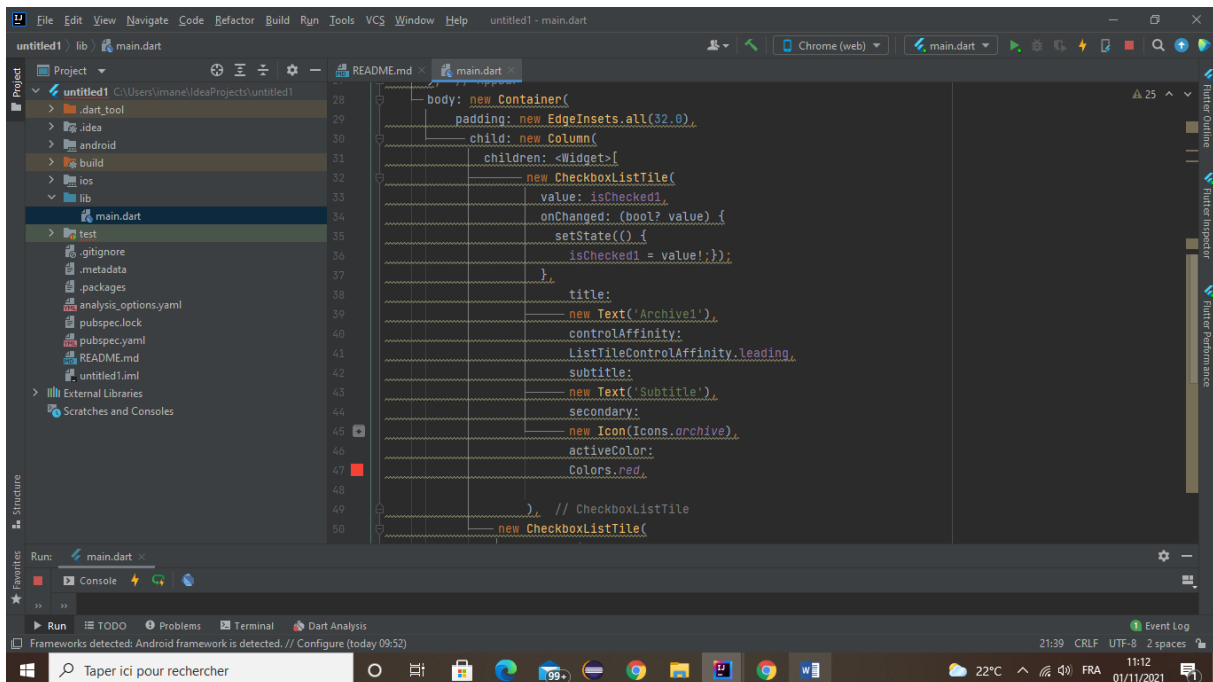
```

1  import 'package:flutter/material.dart'; // MaterialApp
2
3  class MyApp extends StatefulWidget {
4    @override
5    _State createState() => new _State();
6  }
7
8  class _State extends State<MyApp> {
9
10     bool isChecked1 = false;
11     bool isChecked2 = false;
12
13     @override
14     Widget build(BuildContext context) {
15
16       return Scaffold(
17         appBar: new AppBar(
18           title: new Text('Name here!'),
19         ), // AppBar
20         body: new Container(
21           padding: new EdgeInsets.all(32.0),

```

Pour l'affichage du ckeckbox, vous avez le choix entre afficher un simple carré pour cela vous faites appel au widget checkbox().

Sinon on peut choisir un widget checkboxlist() afin d'ajouter des propriété au bouton tel que le titre, l'icône, la couleur



```

28     body: new Container(
29       padding: new EdgeInsets.all(32.0),
30       child: new Column(
31         children: <Widget>[
32           new CheckboxListTile(
33             value: isChecked1,
34             onChanged: (bool? value) {
35               setState(() {
36                 isChecked1 = value!;
37               });
38             },
39             title:
40               new Text('Archive1'),
41             controlAffinity:
42               ListTileControlAffinity.leading,
43             subtitle:
44               new Text('Subtitle'),
45             secondary:
46               new Icon(Icons.archive),
47             activeColor:
48               Colors.red,
49           ), // CheckboxListTile
50           new CheckboxListTile(

```

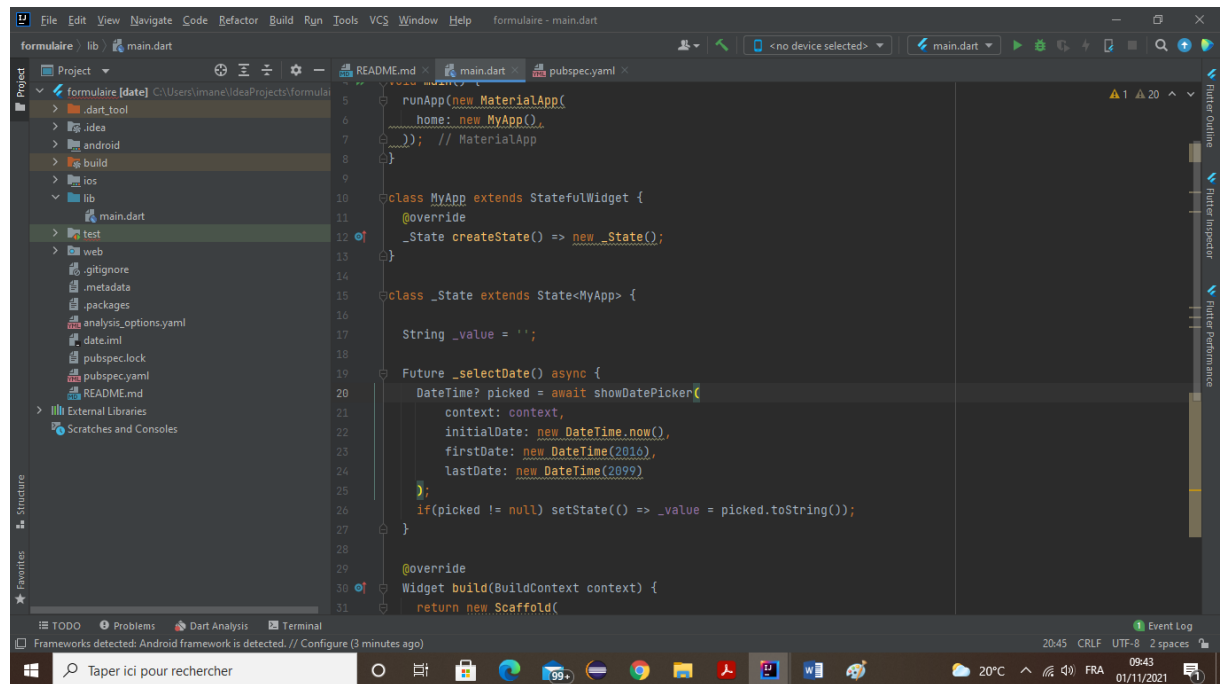
Vous faites la même chose avec le deuxième checkbox.

Etape 4 : Calendrier

Pour créer un calendrier, on a besoin tout d'abord de déclarer un intervalle de date à afficher :

- La date initiale
- La date finale
- La date courante : date de système

Une fois notre marge de date créée, il faut récupérer la date choisie et la définir comme nouvelle état.

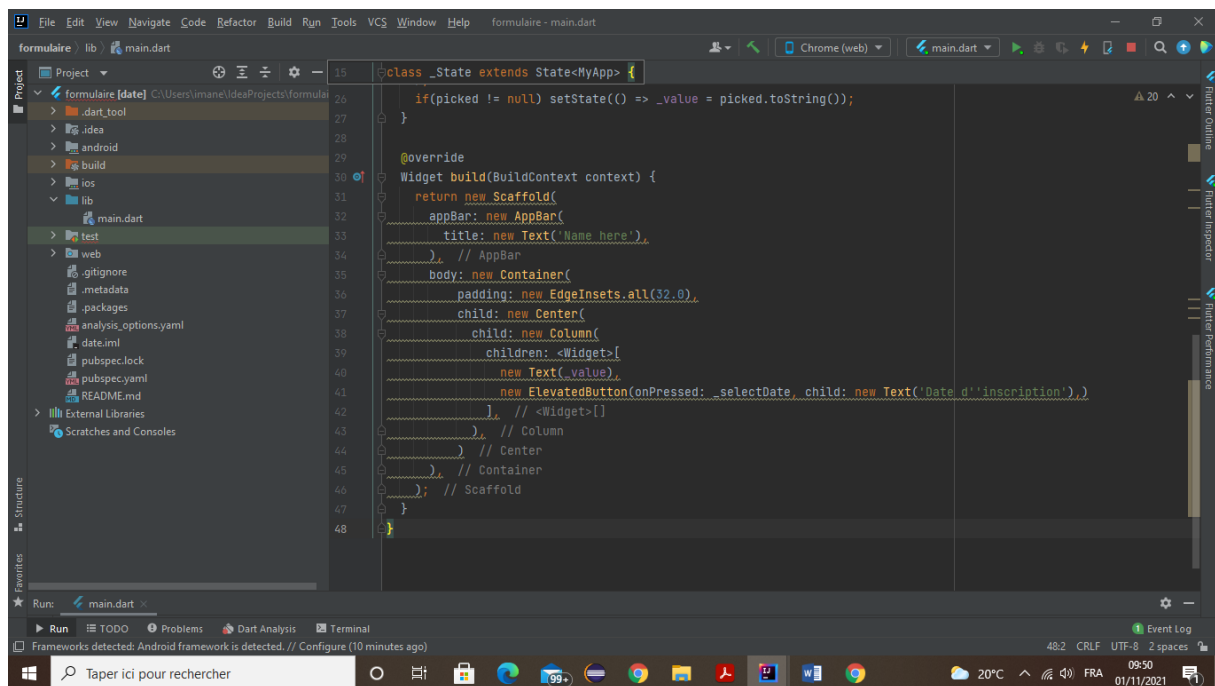


```

1  runApp(new MaterialApp(
2    home: new MyApp(),
3  )); // MaterialApp
4
5  class MyApp extends StatefulWidget {
6    @override
7    _State createState() => new _State();
8  }
9
10 class _State extends State<MyApp> {
11   String _value = '';
12
13   Future _selectDate() async {
14     DateTime? picked = await showDatePicker(
15       context: context,
16       initialDate: new DateTime.now(),
17       firstDate: new DateTime(2010),
18       lastDate: new DateTime(2099)
19     );
20     if(picked != null) setState(() => _value = picked.toString());
21   }
22
23   @override
24   Widget build(BuildContext context) {
25     return new Scaffold(

```

On ajoute un petit bouton pour nous afficher le calendrier.



```

15 class _State extends State<MyApp> {
16   String _value = '';
17
18   Future _selectDate() async {
19     DateTime? picked = await showDatePicker(
20       context: context,
21       initialDate: new DateTime.now(),
22       firstDate: new DateTime(2010),
23       lastDate: new DateTime(2099)
24     );
25     if(picked != null) setState(() => _value = picked.toString());
26   }
27
28   @override
29   Widget build(BuildContext context) {
30     return new Scaffold(
31       appBar: new AppBar(
32         title: new Text('Name here'),
33       ), // AppBar
34       body: new Container(
35         padding: new EdgeInsets.all(32.0),
36         child: new Center(
37           child: new Column(
38             children: <Widget>[
39               new Text(_value),
40               new ElevatedButton(onPressed: _selectDate, child: new Text('Date d\'inscription')),
41             ], // <Widget>[]
42           ), // Column
43         ), // Center
44       ), // Container
45     ); // Scaffold
46   }
47 }

```