

# Documentation application mobile MyPo:

---

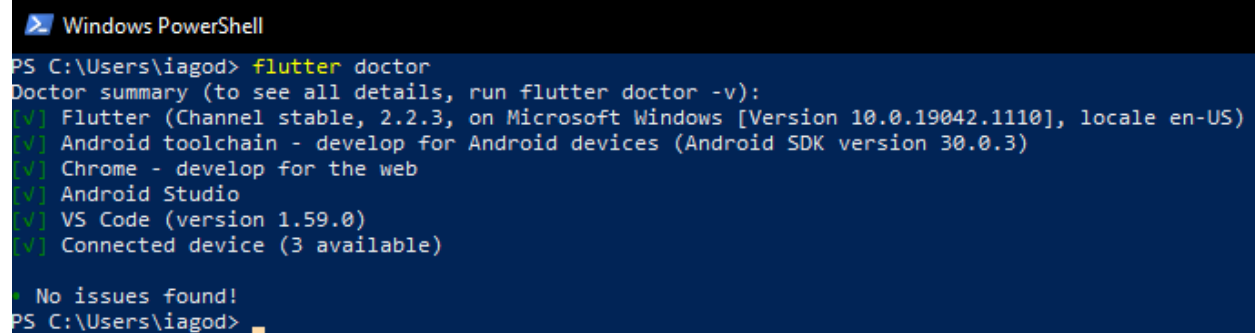
PS :     -mettre à jour la documentation du projet au fur et à mesure  
          - corriger les fautes qui pourraient éventuellement être présent

---

Installer flutter: <https://flutter.dev/docs/get-started/install>

Après installation vérifier que tout est correcte avec la commande

```
flutter doctor
```

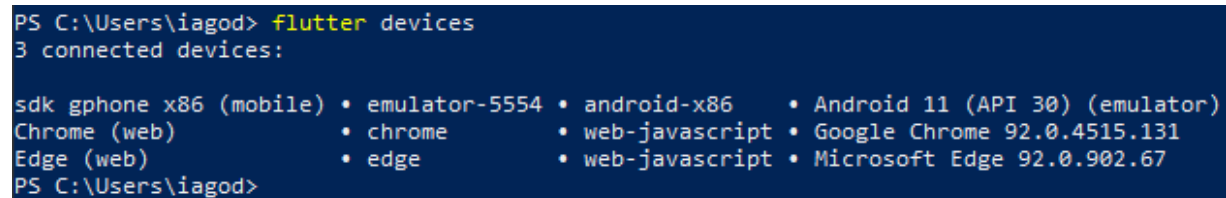


```
Windows PowerShell
PS C:\Users\iagod> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[V] Flutter (Channel stable, 2.2.3, on Microsoft Windows [Version 10.0.19042.1110], locale en-US)
[V] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
[V] Chrome - develop for the web
[V] Android Studio
[V] VS Code (version 1.59.0)
[V] Connected device (3 available)

• No issues found!
PS C:\Users\iagod>
```

Vous pouvez connecter votre telephone, un moteur de recherche ou utiliser un emulateur

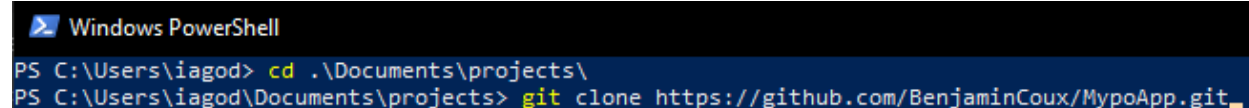
```
flutter devices
```



```
PS C:\Users\iagod> flutter devices
3 connected devices:

sdk gphone x86 (mobile) • emulator-5554 • android-x86      • Android 11 (API 30) (emulator)
Chrome (web)           • chrome       • web-javascript   • Google Chrome 92.0.4515.131
Edge (web)             • edge        • web-javascript   • Microsoft Edge 92.0.902.67
PS C:\Users\iagod>
```

Cloner le dépôt GitHub:



```
Windows PowerShell
PS C:\Users\iagod> cd .\Documents\projects\
PS C:\Users\iagod\Documents\projects> git clone https://github.com/BenjaminCoux/MyPoApp.git
```

Search or jump to... Pulls Issues Marketplace Explore

BenjaminCoux / MyApp Watch 2 Star 1 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main

Go to file Add file Code

About

No description, website, or topics provided.

Readme

Releases

No releases published  
Create a new release

Packages

No packages published  
Publish your first package

Contributors 2

BenjaminCoux

iamiago iago

Languages

Dart 92.3% C++ 5.0% CMake 2.4% C 0.2% Swift 0.1% Kotlin 0.0%

File	Commit Message	Time Ago
.idea	aucune réco	
android	debut env	
assets	changemen	
ios	Changes +	
lib	Merge bran	
test	nouvelle page daccueil	2 months ago
windows	first commit	2 months ago
.gitignore	nb max de characters des messages	10 days ago
DocumentationAppMo...	textfield widget	20 hours ago
README.md	Update README.md	2 days ago
analysis_options.yaml	design	last month
pubspec.lock	fix space dans rapport	4 days ago
pubspec.yaml	textfield widget	20 hours ago
~\$cumentationAppMo...	organisation du code	yesterday
~WRL1806.tmp	organisation du code	yesterday

Se déplacer ensuite dans le dossier créé et ensuite utiliser les commandes

```
flutter pub get
flutter run
```

```
Windows PowerShell
PS C:\Users\iagod\Documents\projects> cd .\MyoApp\
PS C:\Users\iagod\Documents\projects\MyoApp> flutter pub get
Running "flutter pub get" in MyoApp... 5.2s
PS C:\Users\iagod\Documents\projects\MyoApp> flutter run
Using hardware rendering with device sdk gphone x86. If you notice graphics artifacts, consider enabling software rendering with "--enable-software-rendering".
Launching lib\main.dart on sdk gphone x86 in debug mode...
Note: C:\Users\iagod\AppData\Local\Pub\Cache\hosted\pub.dartlang.org\contacts_service-0.6.1\android\src\main\java\Flutter\plugins\contactsservice\contactsservice\ContactsServicePlugin.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Running Gradle task 'assembleDebug'...
Running Gradle task 'assembleDebug'... Done 48.8s
✓ Built build\app\outputs\flutter-apk\app-debug.apk.
Installing build\app\outputs\flutter-apk\app.apk... 1,089ms
Syncing files to device sdk gphone x86... 163ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h Repeat this help message.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

Running with sound null safety

An Observatory debugger and profiler on sdk gphone x86 is available at: http://127.0.0.1:58393/rJCqbSQNwt4=/
The Flutter DevTools debugger and profiler on sdk gphone x86 is available at:
http://127.0.0.1:9101?uri=http%3A%2F%2F127.0.0.1%3A58393%2FrJCqbSQNwt4%3D%2F
```

## Commandes gitHub :

git pull : télécharger les modifications

git push: apporter des modifications via un commit

git add 'fichier1' 'fichier2' : ajout de fichier dans un commit

git commit -m "votre message" : donner un nom au commit

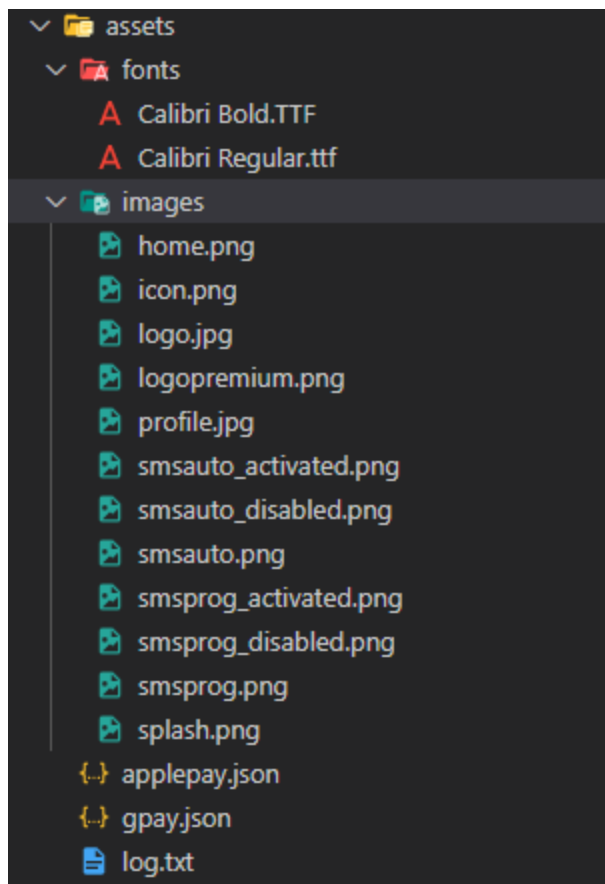
git clone 'adresse' : télécharger un repository

Voir aussi : <https://gist.github.com/aquelito/8596717>

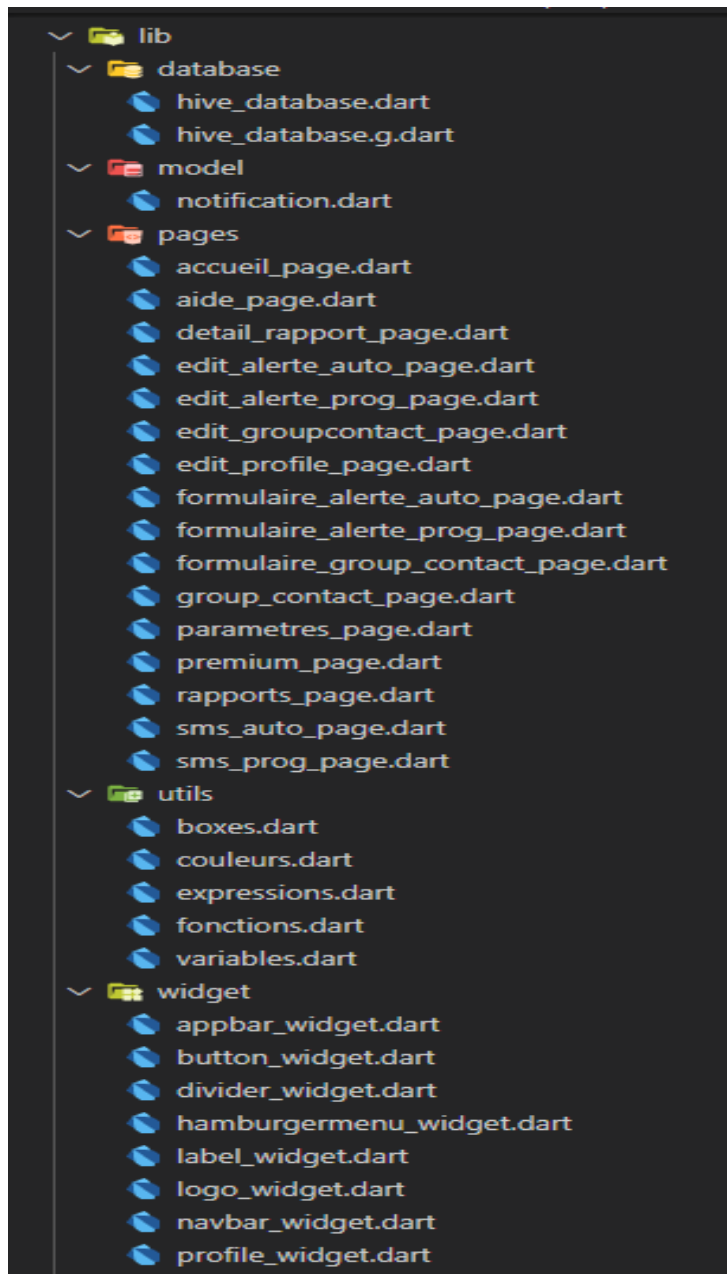
## Structure des dossiers:

Dossier

*Assets*: contient les fonts, les images les fichiers .json, .txt



*Lib* : contient les dossier du projet



**Database** : dossier contenant les fichiers pour une base de données Hive NOSQL et les fichiers automatiquement générés par hive. (Il existe également une base de données SQLITE et SharedPreferences. Hive database est utilisé dans ce projet car c'est un moyen optimal d'utiliser une base de données NOSQL plus rapide que SQLITE et SharedPreferences dans l'écriture et la lecture de données)

Tout d'abord on déclare une class qui hérite de la class HiveObject et les instructions suivantes :

@HiveType(typeId :index) (avant la class)

@HiveField(index) (pour chaque champs)

Comme sur la photo :

[voir la doc de Hive ou des tutos sur internet pour plus d'informations]

```
1  import 'package:hive/hive.dart';
2
3  part 'hive_database.g.dart';
4
5  @HiveType(typeId: 0)
6  class Scheduledmsg_hive extends HiveObject {
7      //extending to hiveobject we can use hive methods such as add save delete etc.
8      @HiveField(0)
9      late String name;
10     @HiveField(1)
11     late String phoneNumber;
12     @HiveField(2)
13     late String message;
14     @HiveField(3)
15     late DateTime date;
16     @HiveField(4)
17     late String repeat;
18     @HiveField(5)
19     late bool countdown;
20     @HiveField(6)
21     late bool confirm;
22     @HiveField(7)
23     late bool notification;
24     @HiveField(8)
25     late DateTime dateOfCreation;
26     @HiveField(9)
27     late List<GroupContact> groupContact;
28     @HiveField(10)
29     late bool status = true;
30 }
```

Ensuite nous utilisons la commande sur le terminal

```
flutter pub run build_runner build
```

Ce code générera automatiquement un fichier .g.dart comme sur la photo

```
lib > database > hive_database.g.dart > ScheduledmsghiveAdapter > read
1  // GENERATED CODE - DO NOT MODIFY BY HAND
2
3  part of 'hive_database.dart';
4
5  // *****
6  // TypeAdapterGenerator
7  // *****
8
9  class ScheduledmsghiveAdapter extends TypeAdapter<Scheduledmsg_hive> {
10   @override
11   final int typeId = 0;
12
13   @override
14   Scheduledmsg_hive read(BinaryReader reader) {
15     final numFields = reader.readByte();
16     final fields = <int, dynamic>{
17       for (int i = 0; i < numFields; i++) reader.readByte(): reader.read(),
18     };
19     return Scheduledmsg_hive()
20       ..name = fields[0] as String
21       ..phoneNumber = fields[1] as String
22       ..message = fields[2] as String
23       ..date = fields[3] as DateTime
24       ..repeat = fields[4] as String
25       ..countdown = fields[5] as bool
26       ..confirm = fields[6] as bool
27       ..notification = fields[7] as bool
28       ..dateOfCreation = fields[8] as DateTime
29       ..groupContact = (fields[9] as List).cast<GroupContact>()
30       ..status = fields[10] as bool;
31   }
32
33   @override
34   void write(BinaryWriter writer, Scheduledmsg_hive obj) {
35     writer
36       ..writeByte(11)
37       ..writeByte(0)
```



Ensuite ceci est le code nécessaire pour le fonctionnement de hive avec flutter, nous devons enregistrer l'adaptateur qui est généré atomiquement lorsqu'on crée des objets de type hive et que l'on utilise la commande :

```
flutter pub run build_runner build
```

```
22 // *****
23 // This function is building the app
24 // *****
Run | Debug | Profile
25 Future main() async {
26   WidgetsFlutterBinding.ensureInitialized();
27
28   await Hive.initFlutter();
29
30   Hive.registerAdapter(Scheduledmsg_hive_adapter());
31   Hive.registerAdapter(Rapportmsg_hive_adapter());
32   Hive.registerAdapter(User_hive_adapter());
33   Hive.registerAdapter(GroupContact_adapter());
34   Hive.registerAdapter(Alert_adapter());
35   Hive.registerAdapter(AlertKey_adapter());
36
37   // loading the <key,values> pair from the local storage into memory
38   try {
39     await Hive.openBox<Scheduledmsg_hive>('scheduledmsg');
40     await Hive.openBox<GroupContact>('group');
41     await Hive.openBox<User_hive>('user');
42     await Hive.openBox<Rapportmsg_hive>('rapportmsg');
43     await Hive.openBox<Alert>('alert');
44     await Hive.openBox<Alert>('alertkey');
45   } catch (e) {
46     debugPrint(e.toString());
47   }
48 }
```

Nous avons plusieurs tables/boxes sur hive

scheduledmsg

group

rapportmsg

alert

alertkey

TABLES : 'scheduledmsg' qui possède des objets de type Scheduledmsg\_hive,

```
1  import 'package:hive/hive.dart';
2
3  part 'hive_database.g.dart';
4
5  @HiveType(typeId: 0)
6  class Scheduledmsg_hive extends HiveObject {
7      //extending to hiveobject we can use hive methods such as add save delete etc.
8      @HiveField(0)
9      late String name;
10     @HiveField(1)
11     late String phoneNumber;
12     @HiveField(2)
13     late String message;
14     @HiveField(3)
15     late DateTime date;
16     @HiveField(4)
17     late String repeat;
18     @HiveField(5)
19     late bool countdown;
20     @HiveField(6)
21     late bool confirm;
22     @HiveField(7)
23     late bool notification;
24     @HiveField(8)
25     late DateTime dateOfCreation;
26     @HiveField(9)
27     late List<GroupContact> groupContact;
28     @HiveField(10)
29     late bool status = true;
30 }
```

'rapportmsg' qui possède des objets de type Rapportmsg\_hive

```
31
32  @HiveType(typeId: 1)
33  class Rapportmsg_hive extends HiveObject {
34      @HiveField(0)
35      late String name;
36      @HiveField(1)
37      late String phoneNumber;
38      @HiveField(2)
39      late String message;
40      @HiveField(3)
41      late DateTime date;
42      @HiveField(4)
43      late String type;
44  }
45
```

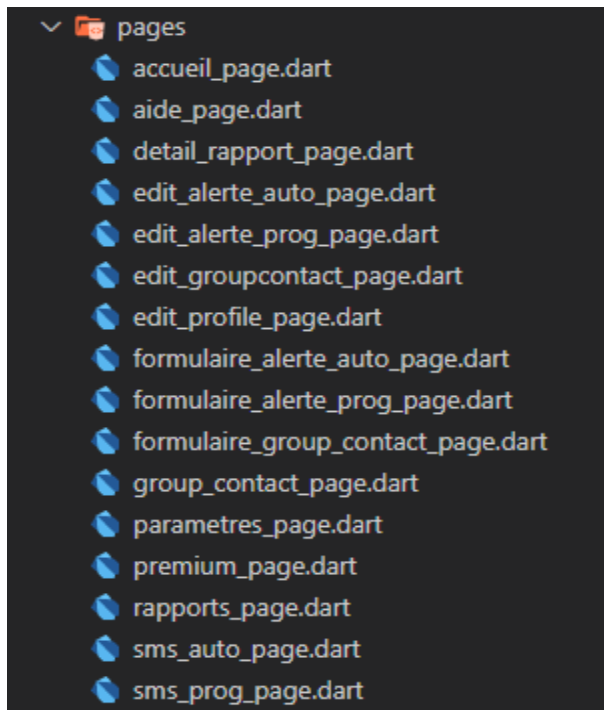
'rapportmsg' qui possède des objets de type Rapportmsg\_hive,  
'user' qui possède des objets de type User\_hive,  
'group' qui possède des objets de type GroupContact,  
'alert' qui possède des objets de type Alert,  
'alertkey' qui possède des objets de type AlertKey.

Chacune d'entre elles possèdent des champs spécifiques que l'on a créé selon nos besoins [c.f lib/database/hive\_database.dart]

*Model* : modèle d'objet utilisés dans l'application (ex notifications dans la photo)

```
lib > model > notification.dart > ...  
1  class ReceivedNotification {  
2      ReceivedNotification({  
3          required this.id,  
4          required this.title,  
5          required this.body,  
6          required this.payload,  
7      });  
8  
9      final int id;  
10     final String? title;  
11     final String? body;  
12     final String? payload;  
13 }  
14
```

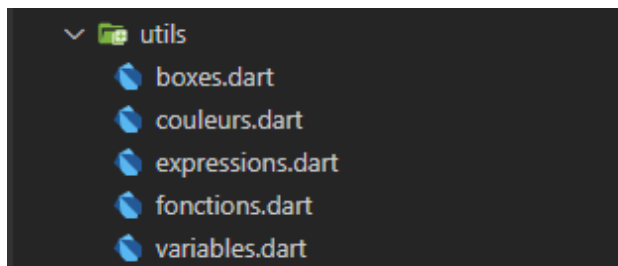
**Pages** : Contient toutes les pages de l'application



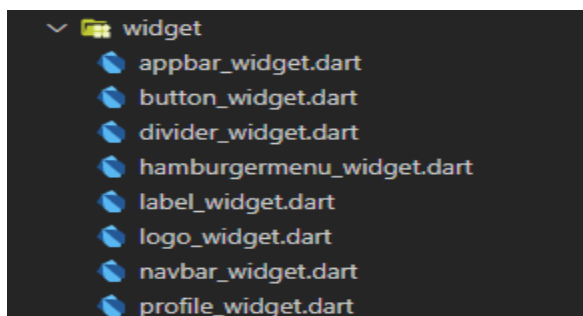
**Utils** : Utilisé essentiellement pour la sauvegarde des données et certaines fonctions utilisées à travers le programme

Contient la classe fonctions avec la fonction pour montrer une notification sur le pied de la page

Contient la classe boxes qui propose les fonctions d'accès aux boxes hive



**Widget** : Contient les widgets utilisés dans les pages tels que la bar de navigation inférieure et supérieure, le logo, le burger menu, la photo du profil etc.



*Pubspec.yaml* (fichier) contient les dépendances, les packages utilisés, la déclaration des ressources utilisés dans le l'application comme les images, font et

```
pubspec.yaml

28 # versions available, run `flutter pub outdated`.
29 dependencies:
30   flutter:
31     sdk: flutter
32   flutter_localizations:
33     sdk: flutter
34
35   #to store data in sqlite database
36   sqflite: ^2.0.0+3
37   #hive data base (faster) NoSQL
38   hive: ^2.0.4
39   hive_flutter: ^1.1.0
40   hive_generator: ^1.1.0
41   flutter_cupertino_localizations: ^1.0.1
42   telephony: ^0.1.4
43   flutter_multiselect: ^1.0.0
44   path_provider: ^2.0.2
45   shared_preferences: ^2.0.6
46   intl: ^0.17.0
47   contacts_service: ^0.6.1
48   permission_handler: ^8.1.4+2
49   quiver: ^3.0.0
50   weekday_selector: ^1.0.0
51   rxdart: ^0.27.1
52   flutter_local_notifications: ^6.0.0
53   email_validator: ^2.0.1
54   intl_phone_field: ^2.0.1
55   url_launcher: ^6.0.9
56   # The following adds the Cupertino Icons font to your application.
57   # Use with the CupertinoIcons class for iOS style icons.
58   cupertino_icons: ^1.0.2
59   font_awesome_flutter: ^9.1.0
60   image_picker: ^0.8.1+3
```

pubspec.yaml

```
92
93 # The following section is specific to Flutter.
94 flutter:
95   # The following line ensures that the Material Icons font is
96   # included with your application, so that you can use the icons in
97   # the material Icons class.
98   uses-material-design: true
99   assets:
100     - assets/images/logo.jpg
101     - assets/images/profile.jpg
102     - assets/images/icon.png
103     - assets/images/icon.png
104     - assets/images/logopremium.png
105     - assets/images/smsprog.png
106     - assets/images/smsauto.png
107     - assets/images/smsauto_activated.png
108     - assets/images/smsprog_activated.png
109     - assets/images/smsprog_disabled.png
110     - assets/images/smsauto_disabled.png
111     - assets/images/home.png
112     - assets/
```

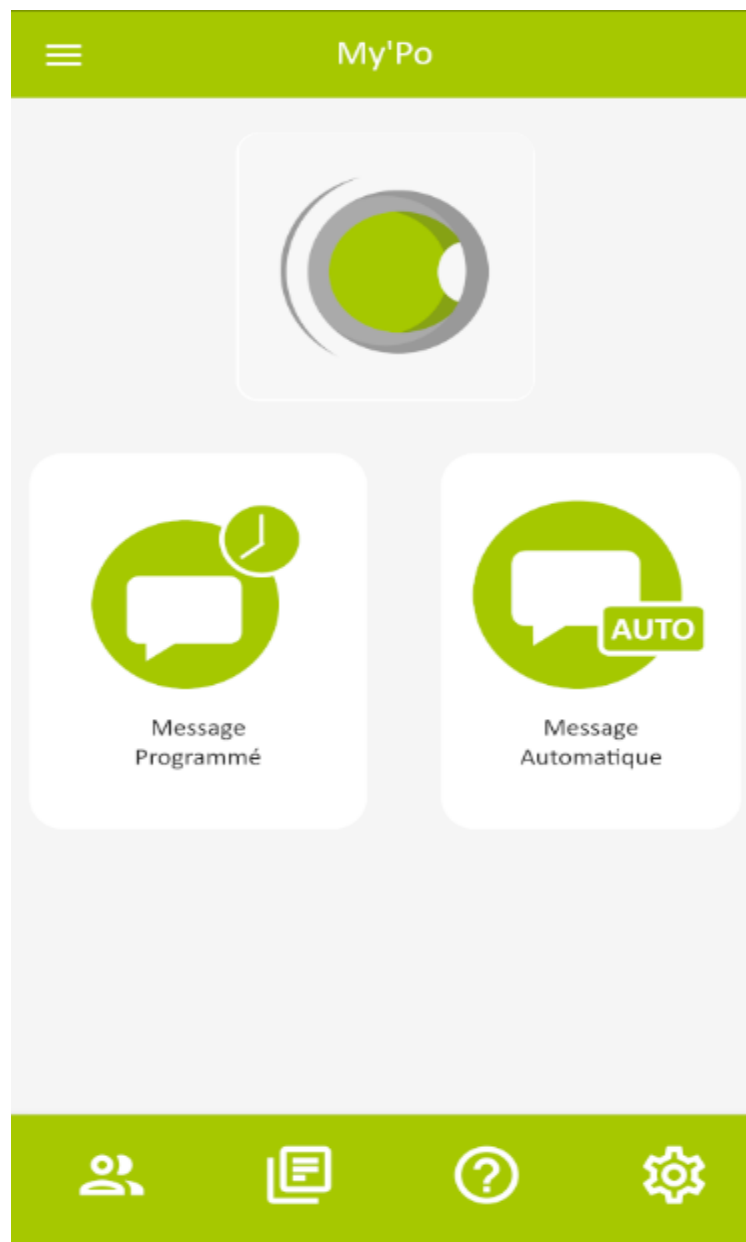




## Les Pages :

accueil\_page.dart : Cette page contient

- La bar de navigation supérieur avec un menu de type burger qui se superpose a l'écran lorsqu'on clique dessous
- Le logo qui occupe 20% de l'écran  
Bouton message programmé qui nous renvoie vers la page des messages programmés
- Bouton message automatique qui nous renvoie vers la page des messages auto
- La bar de navigation inférieure qui permet de naviguer de la page d'accueil a la page des rapports, la page d'aide, et la page des paramètres



On y trouve les fonctions suivantes :

```
class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    bool userDefined = false;
    User_hive? user;
    List users = Boxes.getUser().values.toList().cast<User_hive>();
    if (!users.isEmpty) { ...
    }

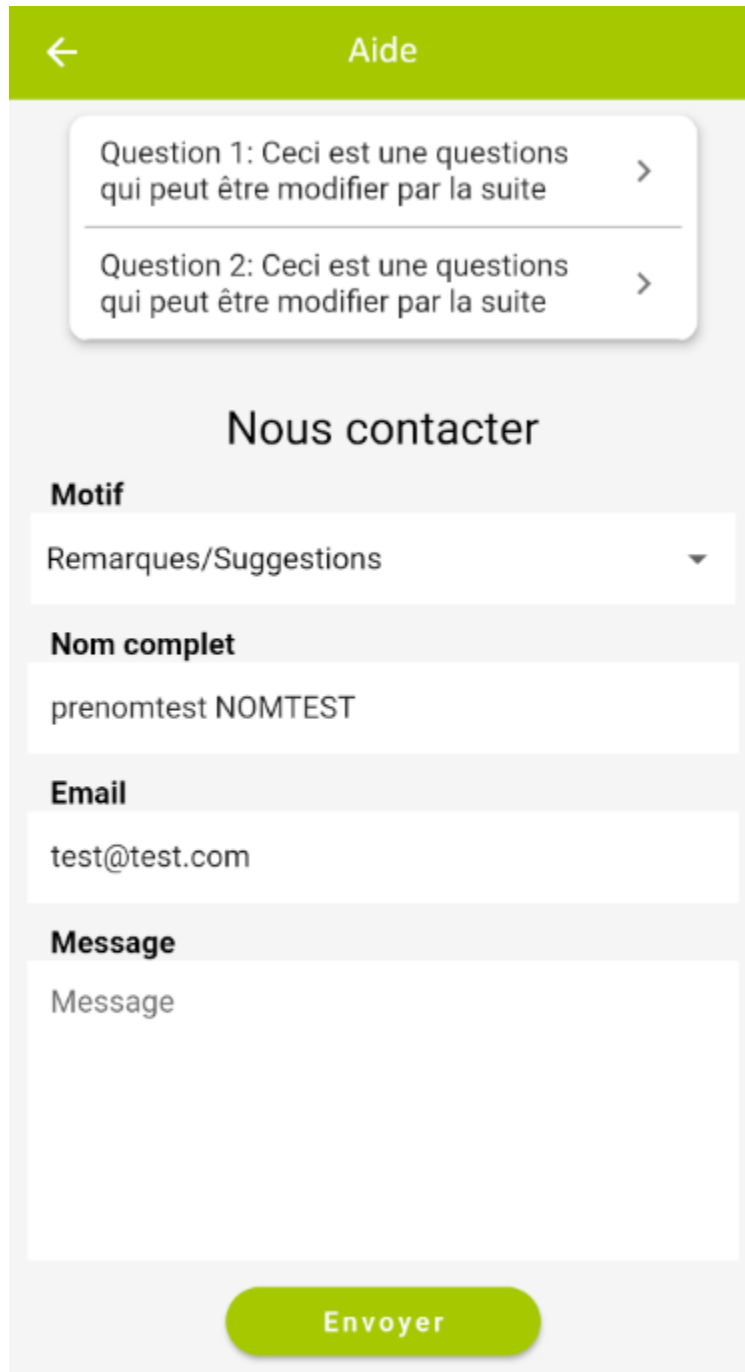
    return Scaffold(
      backgroundColor: d_grey,
      appBar: TopBar(title: userDefined ? user!.name : "My'Po"),
      drawer: HamburgerMenu(),
      body: SingleChildScrollView(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [Logo(imgPath: userDefined ? user!.imagePath : null), Mode()],
        ), // Column // SingleChildScrollView
      bottomNavigationBar: BottomNavigationBarSection(),
    ); // Scaffold
  }
}
```

```
class _ModeState extends State<Mode> {
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Container(
          padding: EdgeInsets.all(5),
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              InkWell( // InkWell ...
              InkWell( // InkWell ...
            ],
          ), // Row
        ), // Container
      ],
    ); // Column
  }
}
```



aide\_page.dart : Cette page contient

- La bar de navigation supérieur avec le retour vers la page d'accueil
- Les questions les plus fréquentes avec les réponses
- Le formulaire de contact pour envoyer un mail à l'entreprise



← Aide

Question 1: Ceci est une questions qui peut être modifier par la suite >

Question 2: Ceci est une questions qui peut être modifier par la suite >

### Nous contacter

**Motif**

Remarques/Suggestions ▼

**Nom complet**

prenomtest NOMTEST

**Email**

test@test.com

**Message**

Message

Envoyer

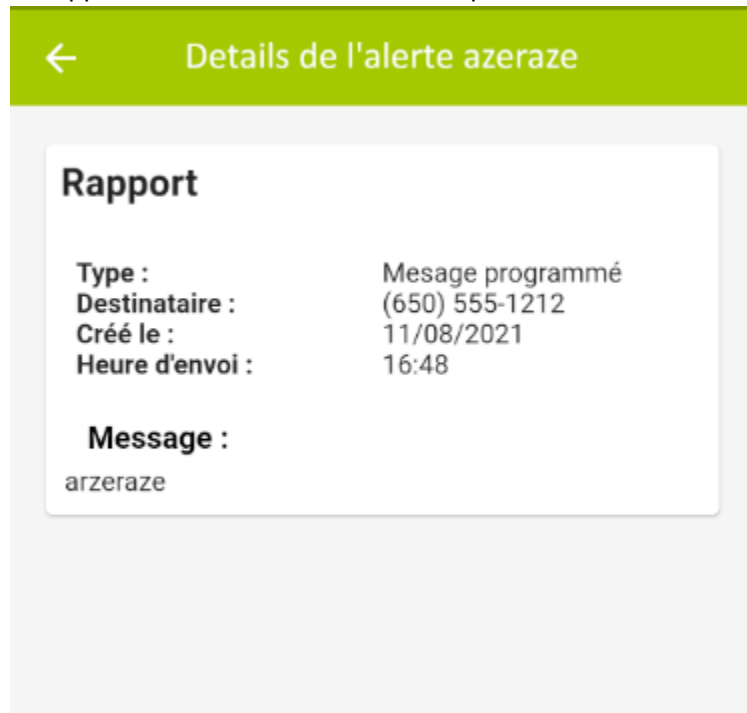
On y trouve les fonctions suivantes :

```
44
45 > Widget buildFormContact() { ...
141
142 > sendMail(String subject, String message) { ...
160
161 myListAide(List<String> questions, int lenght, List<String> reponses,
162 > | BuildContext context) { ...
190
191 > Widget buildDropDown() { ...
213
214 @override
215 > Widget build(BuildContext context) { ...
260 }
```



detail\_rapport\_page.dart : Cette page contient


- La bar de navigation supérieur avec le retour vers la page d'accueil
- Le rapport d'une alerte émises où l'on peu voir :







edit\_alerte\_auto\_page.dart : Cette page contient



Alerte : test

**Nom**

**Message**

**Cible**  

☐ Tous

☐ Contacts uniquement

☐ Numéros non enregistrés

☐ Groupe de contact

☒ SMS reçu

☐ Appel manqué

**Jours**  

lun.

mar.

mer.

jeu.


ven.

sam.


dim.


**Contenu du message entrant**  

Contient ▼      Accepte ▼




happy new year

Contient 

 Notification après réponse

☒

 Règle de réponse

☐

Annuler


Sauvegarder

On y trouve les fonctions suivantes :

```
76 > String getContient(AlertKey a) { ...
86
87 > Color getColorDropDown(AlertKey a) { ...
94
95 > bool verifieCle(String nom) { ...
112
113 > verifieCibles(List<dynamic> cibles) { ...
127
128 > bool isCiblesSet(List<dynamic> cibles) { ...
140
141 > bool isWeekSet(List<dynamic> week) { ...
150
151 > Widget alertKeys(BuildContext context) { ...
366
367 > buildPopupDialogCancel() { ...
395
396 > bool sameName(String n) { ...
405
406 > /* ...
409
409 > Container buildTextFieldMessage(
410 > |   String placeholder, TextEditingController controller, int nbLines) { ...
463
464 > Container buildTextField(
465 > |   String placeholder, TextEditingController controller, int nbLines) { ...
512
513 > void save() async { ...
520
521 > List<bool> buildboolListEdit(List<GroupContact> db, List<GroupContact> cur) { ...
535
536 > @override
537 > Widget build(BuildContext context) { ...
953 }
```



edit\_alerte\_prog\_page.dart : Cette page contient


 Alerte : test

Nom

test

Numéro(s) de contact(s)


(650) 555-1212

+ 

ou

Groupe(s) de contacts

Groupe de contact

+ 

Message

arzero  
arzeroze  
arze

Date de création

14/08/2021  
Heure: 19:00


Date du prochain envoi

14/08/2021  
Heure: 19:00


Changer la date

Réurrence: Aucune récurrence


Changer la récurrence

 Compte à rebours

☐

 Confirmer avant envoi

☐

 Notification

☒

Annuler

Sauvegarder

On y trouve les fonctions suivantes :

```
111 > void changed() { ...
114
115 > void saveChanges() { ...
137
138 > bool sameName(String n) { ...
148
149 > buildPopupDialogCancel() { ...
177
178   Container buildTextField(
179 >     String placeholder, TextEditingController controller, int nbLines) { ...
223
224   Container buildTextFieldMessage(
225 >     String placeholder, TextEditingController controller, int nbLines) { ...
279
280 > Widget buildDatePicker() => SizedBox( // SizedBox ...
315
316   showSheet(BuildContext context,
317     {required Widget child, required VoidCallback onClicked}) =>
318 >     showCupertinoModalPopup( ...
329
330 > Widget buildRepeatOptions() => SizedBox( // SizedBox ...
344   List<Widget> modelBuilder<M>(
345     List<M> models, Widget Function(int index, M model) builder) =>
346     models
347     .asMap()
348 >     .map<int, Widget>( ...
350     .values
351     .toList();
352
353   @override
354 >   Widget build(BuildContext context) { ...
917 }
```



edit\_group\_contact\_page.dart : Cette page contient

← Editer le groupe azert

**Nom du groupe**

azert

**Description**

aezrt

**Numéro(s) de contact(s)**

Numéro de téléphone +

Raze Razer

Valider


On y trouve les fonctions suivantes :


```
82 > void removeFromName(String name) { ...
89
90 > buildTile(String number) { ...
104
105 > Future<Iterable> getC() async { ...
110
111 > String findName(String val, Iterator<Contact> it) { ...
126
127 > void buildNames() { ...
134
135 > void save() { ...
141
142 > buildList(Iterable<Contact> it) { ...
155
156 > final Future<Iterable<Contact>> list = Future<Iterable<Contact>>.delayed( // Future.delayed ...
160 @override
161 > Widget build(BuildContext context) { ...
308 }
```





edit\_profile\_page.dart : Cette page contient

 Éditer votre profil



Passer à la version Premium

Prénom

prenomtest


Nom

NOMTEST

Email

test@test.com

Téléphone

 +33

663666666

9/9

Annuler


Sauvegarder

On y trouve les fonctions suivantes :

```
75 > saveUserToHive(User_hive? User) { ...
97
98 > Future<File> getImageFileFromAssets(String path) async { ...
107
108 > buildPopupDialogCancel(User_hive? user, bool userDefined) { ...
148
149 > Widget buildUpgradeButton(bool userDefined) => ButtonWidget( // ButtonWidget ...
163
164   Widget buildTextField(
165     String labelText,
166     String placeholder,
167     TextEditingController controller,
168     int nbLines,
169 >   TextInputType keyboardType) { ...
234
235   Widget buildTextFieldNumero(String labelText, String placeholder,
236 >   TextEditingController controller, int nbLines) { ...
301
302   @override
303 >   Widget build(BuildContext context) { ...
523 }
```



formulaire\_alerte\_auto\_page.dart : Cette page contient



Ajouter une alerte

**Titre**

**Message**

**Cible**

☐ Tous

☐ Contacts uniquement

☐ Numéros non enregistrés

☐ Groupe(s) de contacts

☐ SMS reçu

☐ Appel manqué

**Jours**

☐ lun.

☐ mar.

☐ mer.

☐ jeu.

☐ ven.

☐ sam.

☐ dim.

**Contenu du message entrant**

Contient

▼

Accepte

▼



Pas encore de clé pour cette alerte

 Notification après réponse

☐

 Règle de réponse

☐

Valider

On y trouve les fonctions suivantes :

```
98 > void _onFormSaved() { ...
102
103 > String getContient(AlertKey a) { ...
113
114 > void saveAlert(String title, String content, var days, var cibles, bool notif,
115 > | List<AlertKey> keys) async { ...
144
145 > bool sameName(String n) { ...
154
155 > Color getColorDropDown(AlertKey a) { ...
162
163 > // *****
168 > bool verifieCle(String nom) { ...
185
186 > verifieCibles(List<bool> cibles) { ...
200
201 > bool isCiblesSet(List<bool> cibles) { ...
213
214 > bool isWeekSet(List<bool> week) { ...
223
224 > Widget weekSelector(BuildContext context) { ...
241
242 > Widget alertKeys(BuildContext context) { ...
440
441 > Container buildTextField(
442 > | String placeholder, TextEditingController controller, int nbLines) { ...
486
487 > Container buildTextFieldMessage(
488 > | String placeholder, TextEditingController controller, int nbLines) { ...
540
541 > @override
542 > Widget build(BuildContext context) { ...
923 }
```



formulaire\_alerte\_prog\_page.dart : Cette page contient

- Cette page contient le formulaire de création pour une alerte programmée, il est accessible lorsqu'on clique sur ajouter une alerte dans la page des messages programmés
  - La bar de navigation supérieure avec le titre de la page
  - Le bouton retour
- les champs de texte pour :
- Nom de l'alerte
  - Numéro du contact avec icône à droite pour aller sélectionner des contacts du téléphone
  - Le message à envoyer
  - La sélection de la date et de l'heure
  - La sélection de la récurrence
  - L'option compte à rebours
  - L'option de confirmation avant envoi
  - L'option de notification lorsqu'un message est envoyé

← Ajouter une alerte

**Nom de l'alerte**

Nom

**Numéro(s) de(s) contact(s)**

Numéro de téléphone + person

ou

**Groupe(s) de contacts**

Groupe(s) de contacts + group

**Message**

Contenu du message


**Date:** 12/08/2021  
**Heure:** 17:58

Date

**Récurrence:** Aucune récurrence

Récurrence


Compte à rebours


Confirmer avant envoi


Notification

Valider

On y trouve les fonctions suivantes :

```

76 > saveToHive() {...
108
109 > bool sameName(String n) {...
119
120 > List<GroupContact> getGrpfromString(String input) {...
132
133   Container buildTextField(
134 >     String placeholder, TextEditingController controller, int nbLines) {...
180
181   Container buildTextFieldMessage(
182 >     String placeholder, TextEditingController controller, int nbLines) {...
234
235 > Widget buildDatePicker() => SizedBox( // SizedBox ...
253
254   showSheet(BuildContext context,
255     {required Widget child, required VoidCallback onClicked}) =>
256 >     showCupertinoModalPopup(...
267
268 > Widget buildRepeatOptions() => SizedBox( // SizedBox ...
282   List<Widget> modelBuilder<M>(
283 >     List<M> models, Widget Function(int index, M model) builder) =>
284     models
285       .asMap()
286       .map<int, Widget>(
287         (index, model) => MapEntry(index, builder(index, model)))
288       .values
289       .toList();
290
291   @override
292 >   Widget build(BuildContext context) {...
818
819 >

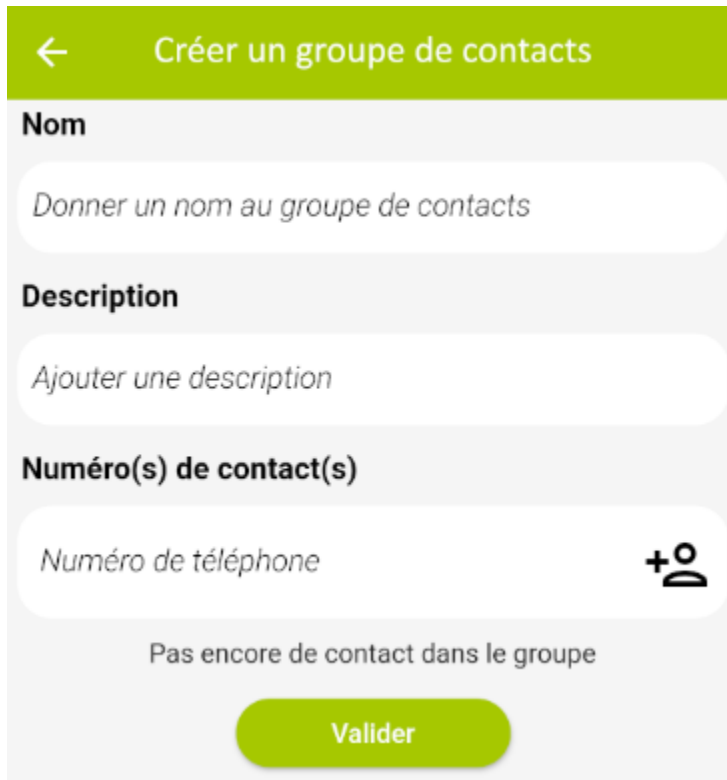
```



- `saveToHive()` :  
Permet de sauvegarder le message sur notre box intitulé 'scheduledmsg' de hive database si tous les champs sont remplis correctement
- `buildTextField(String labelText, String placeholder, TextEditingController controller, int nbLines)` :  
Construit un champ pour rentrer du texte avec le nom du champ(labeltext), le texte d'indice(placeholder) le controleur de ce champ(controller) et le nombre de lignes du champ
- `buildDatePicker()` :  
Construit un sélectionneur de date avec le format de la librairie Cupertino, date initiale étant la date actuelle, format 24h
- `buildRepeatOptions()` :  
Construit un sélectionneur de récurrence avec la librairie Cupertino selon les options de



formulaire\_group\_contact\_page.dart : Cette page contient

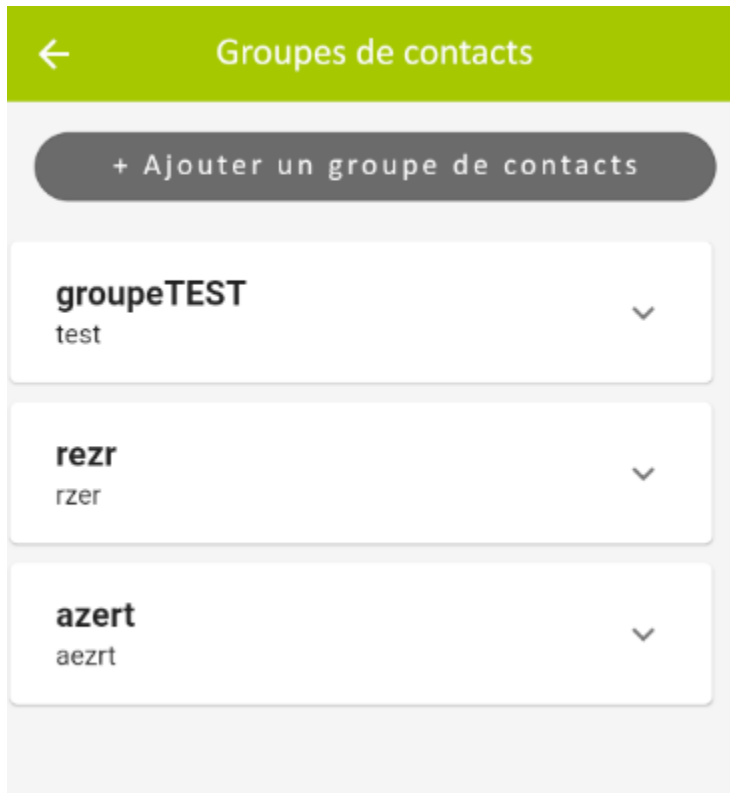


On y trouve les fonctions suivantes :

```
36 > int findNumber(String name) { ...
44
45 > void remove(String name) { ...
56
57 > buildTile(String number) { ...
70
71 > void save() { ...
81
82 > buildList() { ...
94
95 > String findName(String val) { ...
110
111 @override
112 > Widget build(BuildContext context) { ...
257 }
```



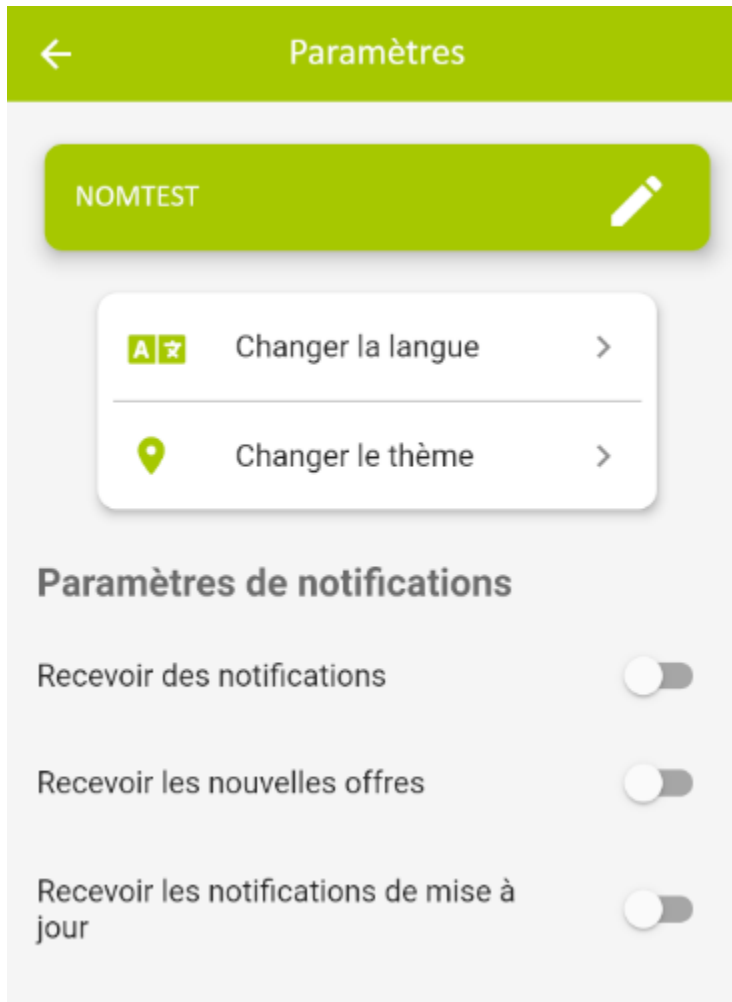
group\_contact\_page.dart : Cette page contient



On y trouve les fonctions suivantes :

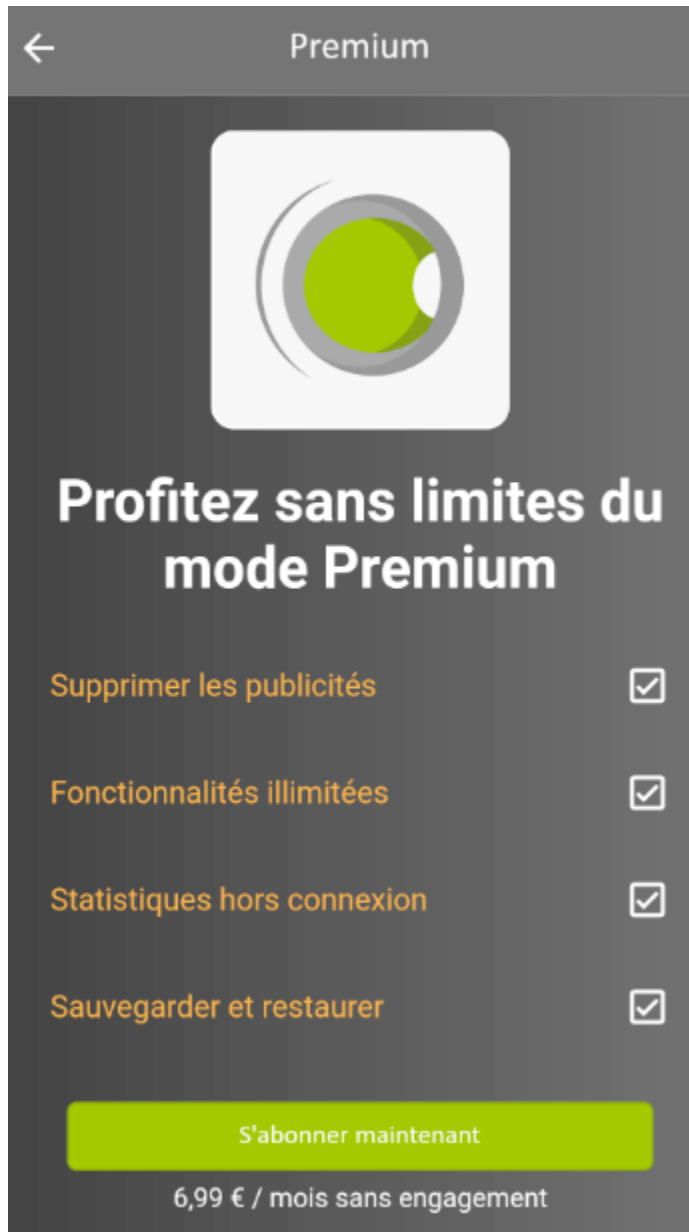
```
28 > buildGroup(BuildContext context, GroupContact contact) { ...
52
53 > String getNbAlerte(String title) { ...
88
89 > void addToDB(GroupContact g) async { ...
97
98 > buildButtons(BuildContext context, GroupContact contact) => Row( // Row ...
153
154 > buildListofContact(int lenght, List<GroupContact> list) { ...
168
169 > buildPopupDialog(GroupContact contact) { ...
199
200 @override
201 > Widget build(BuildContext context) { ...
239 }
```

parametres\_page.dart : Cette page contient



On y trouve les fonctions suivantes :

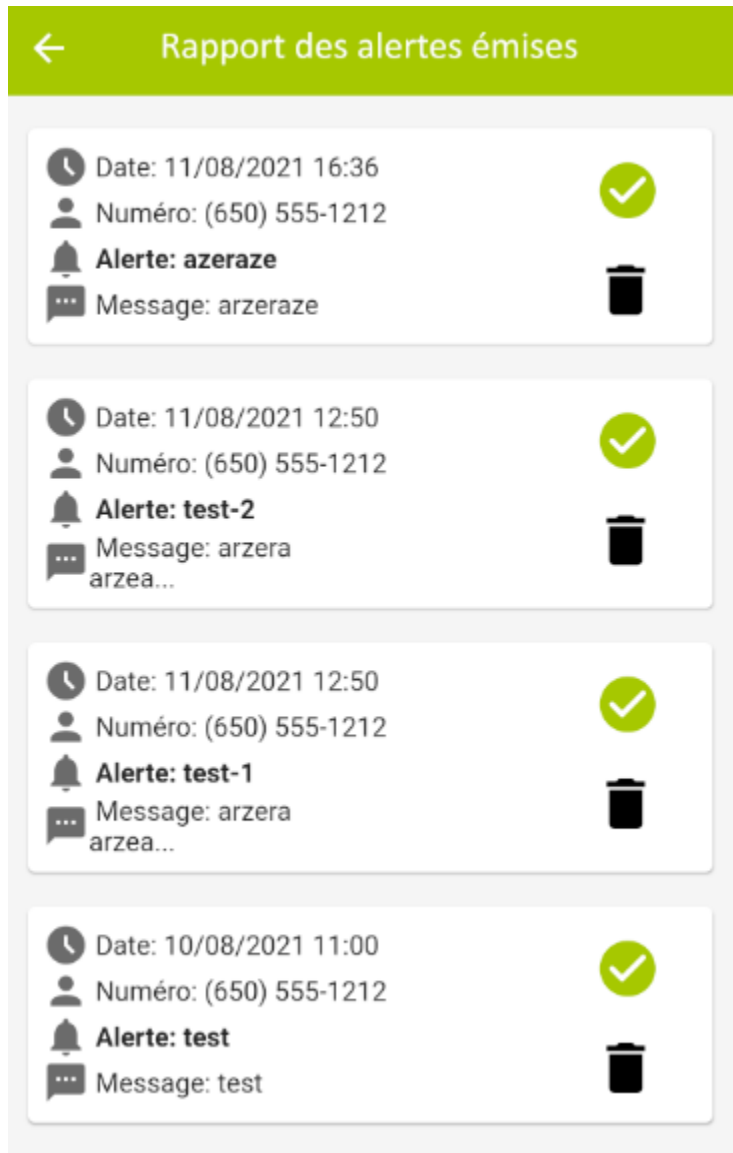
premium\_page.dart : Cette page contient







rapports\_page.dart : Cette page contient

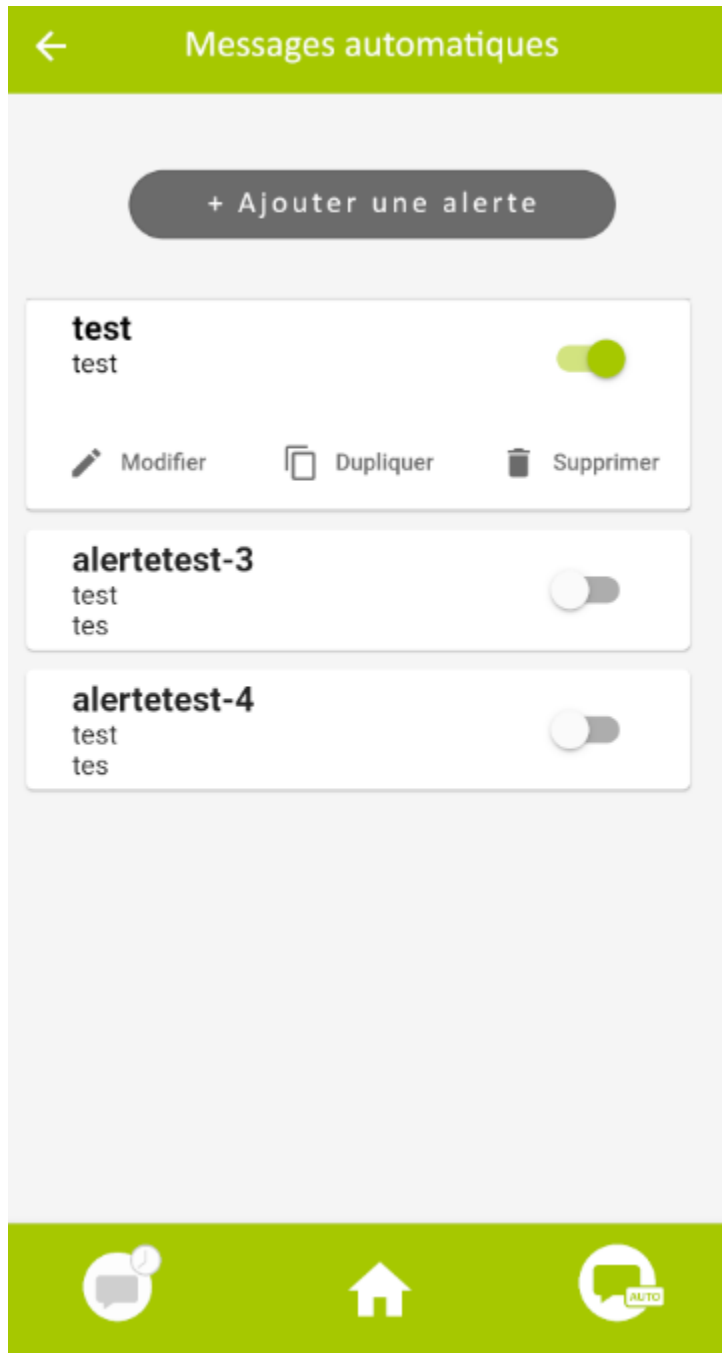


On y trouve les fonctions suivantes :

```
62 > preview(String text) { ...  
70  
71 > Widget buildMessage(BuildContext context, Rapportmsg_hive message) { ...  
163  
164 > buildPopupDialog(Rapportmsg_hive message) { ...  
191  
192 @override  
193 > Widget build(BuildContext context) { ...  
216 }
```



sms\_auto\_page.dart : Cette page contient



On y trouve les fonctions suivantes :

```
24 final FlutterLocalNotificationsPlugin flutterLocalNotificationsPlugin =
25     FlutterLocalNotificationsPlugin();
26
27 > class SmsAuto extends StatefulWidget { ...
31
32 class _SmsAutoState extends State<SmsAuto> {
33     List<Alert> alerts = <Alert>[];
34
35 > Future<void> readAlert() async { ...
40
41     @override
42 > void initState() { ...
46
47     @override
48 > void dispose() {} ...
51
52     @override
53 > Widget build(BuildContext context) { ...
72 }
73
74 // ignore: must_be_immutable
75 > class SwitchButton extends StatefulWidget { ...
81
82 class _StateSwitchButton extends State<SwitchButton> {
83     bool state = false;
84
85     final box = Boxes.getAutoAlert();
86
87     // ignore: unused_field
88     String _message = "";
89
```

```
90     final telephony = Telephony.instance;
91
92 >     /* ...
95     @Override
96 >     void initState() { ...
100
101 >     /* ...
107 >     onMessage(SmsMessage message) async { ...
132
133     //This function return the status of the message after sending it
134
135 >     onSendStatus(SendStatus status) { ...
140
141 >     /* ...
144
145 >     Future<void> initPlatformState() async { ...
158
159 >     /** ...
162 >     changeActive(Alert alerte, bool s) async { ...
166
167     @Override
168 >     Widget build(BuildContext context) { ...
183 | }

```

```

186 > class Alertes extends StatefulWidget { ...
192
193   class _AlertesState extends State<Alertes> {
194     // ignore: unused_field
195     @override
196 >   void initState() { ...
199
200     @override
201 >   void dispose() { ...
205
206 >   /* ...
209 >   void delete(Alert alert) async { ...
212
213 >   /* ...
216 >   buildPopupMenuDialog(Alert alerte) { ...
246
247 >   void addToDB(Alert alert, String title) async { ...
260
261 >   /* ...
264 >   String getNbAlerte(String title) { ...
299
300 >   /* ...
303 >   myList(List<Alert> alerts, int lenght, BuildContext context) { ...
315
316 >   Widget buildMsg(BuildContext context, Alert alert) { ...
341
342 >   buildButtons(BuildContext context, Alert alert) => Row( // Row ...
406
407 >   /* ...
410   Future<List> callAsyncFetch() =>
411     Future.delayed(Duration(milliseconds: 1), () => widget.alerts);
412
413 >   /* ...
416   @override
417 >   Widget build(BuildContext context) { ...
468
469 >   //pop menu hidden
470 >   void onButtonPressed() { ...
526 }

```

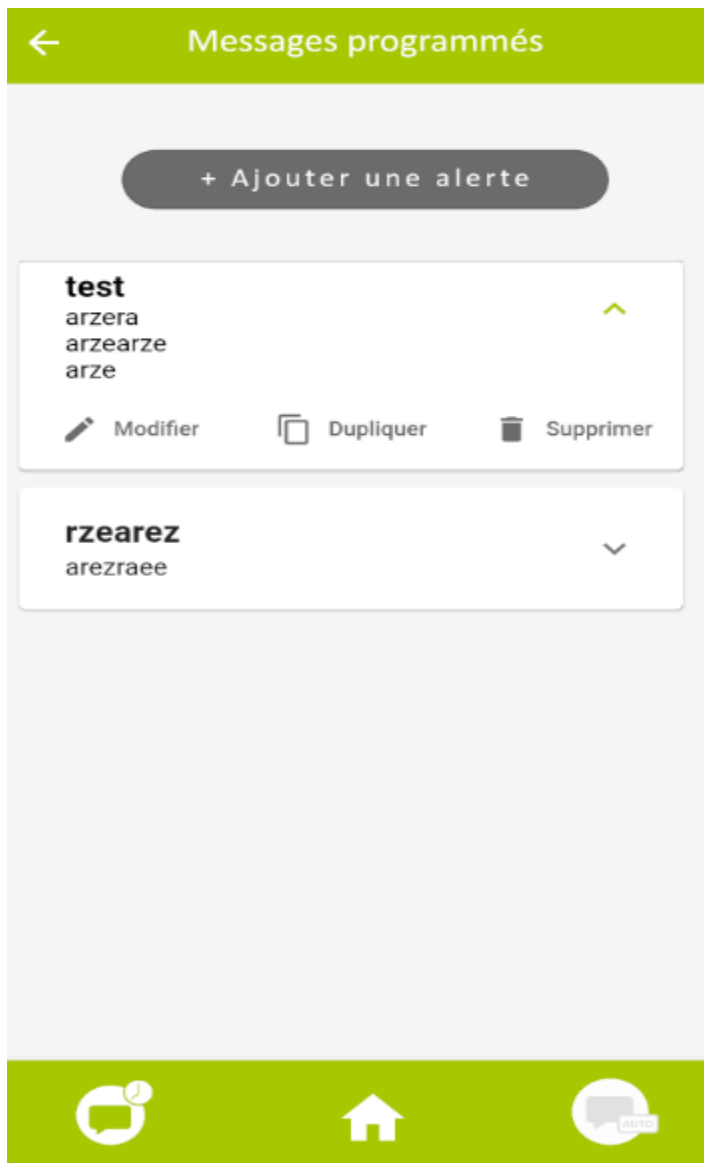
```
529 > /** ...
532 > String getLastWord(String str) { ...
556
557 > /** ...
560 > Alert createAlert(Alert a, bool activated) { ...
564
565 > /** ...
568 > String getFirstWord(String str) { ...
582
583 > /** ...
586 > bool isActive(String? body, Alert alert, bool isInContact) { ...
623
624 > Future<bool> isContactInContactList(SmsMessage message) async { ...
640
641 > onBackgroundMessage(SmsMessage message) async { ...
667
668 > Future<List<Alert>> getAlerts() async { ...
699
700 > /** ...
703 > bool dontAllow(String? body, Alert a) { ...
725
726 > bool dayIsRight(Alert alert, String day) { ...
743
744 > Future<void> _showNotification(String title, String body) async { ...
756
757 > /** ...
761 > List<AlertKey> buildKeys(dynamic input) { ...
773
774 > bool isInGroup(Alert a, String? adress) { ...
```





`sms_prog_page.dart` : Cette page contient

- La bar de navigation supérieur avec un bouton retour sur la page d'accueil  
Le bouton pour ajouter une nouvelle alerte
- Un message qui est affiché si on ne possède aucune alerte
- La bar de navigation inférieure qui permet de revenir à la page d'accueil et changer entre les pages de message programmé ou message automatique



Dans ce fichier on trouve les fonctions

```
31  @override
32  > void initState() { ...
40
41  > saveMsgToRapport(Scheduledmsg_hive message) { ...
51
52  > void send(Scheduledmsg_hive msg) async { ...
70
71  > void sendSms() async { ...
99
100 > bool fiveMinutesBeforeSend(Scheduledmsg_hive msg) { ...
108
109 > void updateDate(Scheduledmsg_hive msg) { ...
137
138 > bool canbeSent(Scheduledmsg_hive msg) { ...
147
148 > Future selectNotification(String payload) async { ...
154
155     Future<void> _showNotification(
156 >     | String number, List<GroupContact> list, String body) async { ...
186
187 > Future<void> _showNotificationBis(String title, String body) async { ...
199
200 > confirmSend(Scheduledmsg_hive msg) { ...
259
260 > myList(List<Scheduledmsg_hive> alerts, int lenght, BuildContext context) { ...
272
273     Future<List> callAsyncFetch() =>
274     | Future.delayed(Duration(milliseconds: 1), () => alerts);
275
276 > Widget buildMsg(BuildContext context, Scheduledmsg_hive message) { ...
299
300 > void addToDB(Scheduledmsg_hive alert, String title) async { ...
315
316 > String getNbAlerte(String title) { ...
351
352 > buildButtons(BuildContext context, Scheduledmsg_hive message) => Row( // Row ...
422
423 > buildPopupDialog(Scheduledmsg_hive message) { ...
454
455     @override
456 > Widget build(BuildContext context) { ...
525 }
```

- `saveMsgToRappor(Scheduledmsg_hive message)` :  
sauvegarde un message dans la boîte/box des rapport
- `sendSms()` :  
Vérifie si le message possède des options avant envoi
- `send()` :  
Envoi un message avec la librairie telephony
- `updateDate(Scheduledmsg_hive msg)` :  
Prend en paramètre un message et met à jour la date d'envoi du message programmé en fonction de la récurrence choisi (tout les ans, tout les mois, aucune etc.)
- `canBeSent(Scheduledmsg_hive msg)` :  
Retourne vrai si l'heure actuelle est supérieure ou égale a l'heure programmé pour envoi
- `confirmSend(Scheduledmsg_hive msg)` :  
Créer un pop up pour demander si l'utilisateur souhaite envoyer le msg si non un délai de 5 minutes est ajouter a l'heure d'envoi
- `buildListOfMsg(List<Scheduledmsg_hive> messages)`:  
Prend en paramètre la liste de messages de type `scheduledmsg_hive` et construit la liste de messages avec `ListView.Builder`
- `buildMsg(BuildContext context, Scheduledmsg_hive message)`:  
Créer le modèle pour un message
- `buildButtons(BuildContext context, Scheduledmsg_hive message)`:  
Créer les boutons affichés dans le message (modifier supprimer)
- `buildPopupDialog(Scheduledmsg_hive message)` :  
Créer un pop up pour demander si l'utilisateur souhaite supprimer ou non l'alerte

## Utils :

Le dossier utils contient un ensemble de pages pour mieux organiser le code

- **boxes.dart :**

class responsable de la récupération des boîtes/boxes de hive database

On trouve la méthode getScheduledmsg() qui peut être utilisé pour récupérer tous les messages sauvegarder sur la boîte 'scheduledmsg' exemple (final allMessages =

Boxes.getScheduledmsg() )

```
lib > utils > boxes.dart > ...
1  import 'package:hive/hive.dart';
2  import 'package:mypo/database/hive_database.dart';
3
4  class Boxes {
5      static Box<Scheduledmsg_hive> getScheduledmsg() =>
6          Hive.box<Scheduledmsg_hive>('scheduledmsg');
7
8      static Box<Rapportmsg_hive> getRapportmsg() =>
9          Hive.box<Rapportmsg_hive>('rapportmsg');
10
11     static Box<GroupContact> getGroupContact() => Hive.box<GroupContact>('group');
12
13     static Box<Alert> getAutoAlert() => Hive.box<Alert>('alert');
14
15     static Box<Alert> getAlertKey() => Hive.box<Alert>('alertkey');
16
17     static Box<User_hive> getUser() => Hive.box<User_hive>('user');
18 }
```

- **couleurs.dart :**

Contient toutes les couleurs utilisés dans l'application

```
1  import 'package:flutter/material.dart';
2
3  const d_green = Color(0xFFA6C800);
4  Color d_grey = Colors.grey.shade100;
5  const d_gray = Color(0xFFBABABA);
6  const d_darkgray = Color(0xFF6C6C6C);
7  const d_lightgray = Color(0XFFFAFAFA);
8  |
```

- **expressions.dart :**

Contient toutes les expressions régulières qu'on utilise pour valider certains champs

```
lib > utils > expressions.dart > ...
1  final regularExpression =
2    |   RegExp(r'^[a-zA-Z0-9_\-\@,.\äåâÀèëéÉèÈíîïôóóúúÚçÇñÑ@ \.;\]+$');
3
4  final phoneExpression = RegExp(r'^[0-9_\-\+() \.;\]+$');
5  final alphanumeric = RegExp(r'^[a-zA-Z0-9.:#_-\éàô ]+$');
6  final numeroExpression =
7    |   RegExp(r'^[+]*([0,1]{0,1}[0-9]{1,4}){0,1}[-\s\./0-9]*$');
8
```

- **fonctions.dart :**

Contient certaines fonctions réutilisés pas mal de fois dans les autres pages

```
lib > utils > fonctions.dart > ...
1  import 'package:flutter/material.dart';
2
3  class Fonctions {}
4
5  void showSnackBar(BuildContext context, String s) {
6    final snackBar = SnackBar(
7      content: Text(s, style: TextStyle(fontSize: 20)),
8    ); // SnackBar
9    ScaffoldMessenger.of(context)
10     ..removeCurrentSnackBar()
11     ..showSnackBar(snackBar);
12  }
```

- **variables.dart :**

Contient certaines variables qui sont réutilisés dans différents champs et représentent la même valeur

```
lib > utils > variables.dart > ...
1  const double scrollbarThickness = 10;
2  const double bottomNavBarIconSize = 40;
3
```



## Widget :

Le dossier widget contient un ensemble de widget réutilisables selon nos besoin dans nos pages, pour ce faire nous n'avons qu'à l'importer et l'appeler avec les paramètres nécessaires

- **appbar\_widget.dart :**

widget responsable pour la bar de navigation supérieure

```
class TopBar extends StatelessWidget implements PreferredSizeWidget {  
  final String title;  
  Size get preferredSize => new Size.fromHeight(50);  
  
  const TopBar({  
    Key? key,  
    required this.title,  
  }) : super(key: key);  
  @override  
  Widget build(BuildContext context) {  
    return AppBar(  
      elevation: 0,  
      title: Text(title, style: TextStyle(fontFamily: 'calibri')),  
      centerTitle: true,  
      backgroundColor: d_green,  
    ); // AppBar  
  }  
}
```

Pour appeler ce widget

- TopBar(« Mon titre ») :  
Retourne in widget avec le titre donné en paramètre



- `button_widget.dart`:

widget responsable pour la création d'un bouton avec un titre et une fonction sur le clique

```
lib > widget > button_widget.dart > ...
1  import 'package:flutter/material.dart';
2  import 'package:mypo/utils/couleurs.dart';
3
4  class ButtonWidget extends StatelessWidget {
5      final String text;
6      final VoidCallback onClicked;
7
8      const ButtonWidget({
9          Key? key,
10         required this.text,
11         required this.onClicked,
12     }) : super(key: key);
13
14     @override
15     Widget build(BuildContext context) => ElevatedButton(
16         style: ElevatedButton.styleFrom(
17             textStyle: TextStyle(fontFamily: 'calibri', fontSize: 18),
18             primary: d_green,
19             onPrimary: Colors.white,
20             shape: StadiumBorder(),
21             padding: EdgeInsets.symmetric(horizontal: 30, vertical: 10)),
22         child: Text(text),
23         onPressed: onClicked,
24     ); // ElevatedButton
25 }
```

Pour utilise ce widget

- `ButtonWidget(String text, VoidCallback onClicked)` :  
permet d'afficher un bouton avec le titre donné en paramètre et la fonction à exécuter sur le clique donné également en paramètre

- divider\_widget.dart :

widget responsable pour la création d'un diviseur

```
lib > widget > divider_widget.dart > ...
1  import 'package:flutter/material.dart';
2
3  Container buildDivider() {
4    return Container(
5      margin: EdgeInsets.symmetric(horizontal: 8),
6      width: double.infinity,
7      height: 1,
8      color: Colors.grey.shade400); // Container
9  }
10
11 Container buildDividerTransparent() {
12   return Container(
13     margin: EdgeInsets.symmetric(horizontal: 8),
14     width: double.infinity,
15     height: 1,
16     color: Colors.transparent); // Container
17 }
```

Pour utilise ce widget

- buildDivider() ou buildDividerTransparent()

- hamburgermenu\_widget.dart :

widget responsable pour la création d'un bouton de type burger

le menu avec les options compte, rapport, paramètres et aide s'affichent lorsqu'on clique et nous redirige vers l'une des pages sélectionnée

```

1  import 'package:flutter/material.dart';
2  import 'package:mypo/pages/aide_page.dart';
3  import 'package:mypo/pages/edit_profile_page.dart';
4  import 'package:mypo/pages/rapports_page.dart';
5  import 'package:mypo/pages/parametres_page.dart';
6  import 'package:mypo/widget/divider_widget.dart';
7
8  class HamburgerMenu extends StatefulWidget {
9    @override
10   _HamburgerMenuState createState() => _HamburgerMenuState();
11 }
12
13 class _HamburgerMenuState extends State<HamburgerMenu> {
14   @override
15   Widget build(BuildContext context) {
16     return Drawer(
17       child: Container(
18         color: Colors.white,
19         child: ListView(children: <Widget>[
20 >       new ListTile( // ListTile ...
33         buildDividerTransparent(),
34 >       new ListTile( // ListTile ...
47         buildDividerTransparent(),
48 >       new ListTile( // ListTile ...
62         buildDividerTransparent(),
63 >       new ListTile( // ListTile ...
77         buildDividerTransparent(),
78       ]), // <Widget>[] // ListView
79     )); // Container // Drawer
80   }
81 }

```

Pour utiliser ce widget

- HamburgerMenu() :  
permet d'afficher un bouton de type burger  
widget de type drawer

- logo\_widget.dart :

widget responsable pour la création du logo avec une taille fixe de 25% de l'écran

```
lib > widget > logo_widget.dart > Logo

1  import 'dart:io';
2
3  import 'package:flutter/material.dart';
4  import 'package:mypo/utils/couleurs.dart';
5  import 'package:mypo/pages/edit_profile_page.dart';
6
7  /*
8  | -that class creates the logo in the middle
9  |
10 class Logo extends StatelessWidget {
11   final String? imgPath;
12   Logo({Key? key, required this.imgPath}) : super(key: key);
13
14   @override
15   Widget build(BuildContext context) {
16     var heightOfScreen = MediaQuery.of(context).size.height * 0.20;
17     bool isImgSet = false;
18     if (imgPath != null) {
19       isImgSet = true;
20     }
21
22     return isImgSet
23       ? Column(
24         children: [buildImage(context)],
25       )
26       : Center(
27         child: Container(
28           margin: EdgeInsets.fromLTRB(0, 20, 0, 20),
29           height: heightOfScreen,
30           width: heightOfScreen,
31           decoration: BoxDecoration(
32             color: d_grey,
33             borderRadius: BorderRadius.all(Radius.circular(12)),
34             image: DecorationImage(
35               image: AssetImage('assets/images/icon.png'),
36             ), // DecorationImage // BoxDecoration
37           ), // Container
38         ); // Center
39   }
40
41   > buildImage(BuildContext context) { ...
81 }
```

Pour utiliser ce widget

- Logo() :  
permet d'afficher un conteneur avec le logo à l'intérieur

- `navbar_widget.dart` :  
 widget responsable pour la création des bar de navigation inférieure  
 ce fichier comporte les différentes bar de navigation utilisés dans l'application

```

12  /*
13  |  -this class is responsible of the bottom nav bar of 2 elements for the sms auto page with images as icon
14  */
15  // ignore: must_be_immutable
16  class BottomNavigationBarSmsAutoTwo extends StatelessWidget {
17    /*
18    |  - Building the bottom navigation bar
19    */
20    // TO-DO align vertically the navigation bar
21    @override
22  >  Widget build(BuildContext context) { ...
81  }
82
83  /*
84  |  -this class is responsible of the bottom nav bar of 2 elements for the sms prog page with images as icon
85  */
86  // ignore: must_be_immutable
87  class BottomNavigationBarSmsProgTwo extends StatelessWidget {
88    /*
89    |  - Building the bottom navigation bar
90    */
91    @override
92  >  Widget build(BuildContext context) { ...
151 }
152
153  /*
154  |  -This class creates the widget responsible of the bottom navigation bar of 4 elements
155  */
156 > class BottomNavigationBarSection extends StatefulWidget { ...
161
162  class _StateBottomNavigationBarSection
163  |  extends State<BottomNavigationBarSection> {
164  |  final String value = 'test';
165  |
166  |  @override
167  >  Widget build(BuildContext context) { ...
247 }

```

Pour utiliser ce widget

- `BottomNavigationBarSmsAutoTwo()` :
- `BottomNavigationBarSmsProgTwo()` :
- `BottomNavigationBarSection()` :

- profile\_widget.dart :

widget responsable pour la création de l'image du profile avec l'icone pour modification

```
lib > widget > profile_widget.dart > ...

1  import 'dart:io';
2
3  import 'package:flutter/material.dart';
4  import 'package:mypo/utils/couleurs.dart';
5
6  class ProfileWidget extends StatelessWidget {
7    final String imagePath;
8    final bool isEdit;
9    final VoidCallback onClicked;
10
11    const ProfileWidget({
12      Key? key,
13      required this.imagePath,
14      this.isEdit = false,
15      required this.onClicked,
16    }) : super(key: key);
17
18    @override
19    > Widget build(BuildContext context) { ...
20
21    > Widget buildImage() { ...
22
23    > Widget buildEditIcon(Color color) => buildCircle( ...
24
25    buildCircle({
26      required Widget child,
27      required double all,
28      required Color color,
29    }) =>
30    > ClipOval( // ClipOval ...
31  }
```

Pour utilise ce widget

- ProfileWidget(String imagePath, bool isEdit, VoidCallback onClicked) :  
permet de superposes des widgets sur les autres pour créer le bouton d'ajout sur l'image

- `textfield_widget.dart` :

widget responsable pour la création d'un champs de text

```
lib > widget > textfield_widget.dart > textField

1  import 'package:flutter/material.dart';
2
3  Container textField(
4      String placeholder, TextEditingController controller, int nbLines) {
5      return Container(
6          decoration: BoxDecoration(
7              color: Colors.white,
8              borderRadius: BorderRadius.all(
9                  Radius.circular(18),
10             ), // BorderRadius.all
11          ), // BoxDecoration
12      margin: EdgeInsets.fromLTRB(10, 10, 10, 10),
13      child: Padding(
14          padding: const EdgeInsets.all(0),
15          child: TextField(
16              controller: controller,
17              onChanged: (String value) => {},
18              maxLines: nbLines,
19              keyboardType: TextInputType.text,
20              decoration: InputDecoration(
21                  labelStyle: TextStyle(color: Colors.black),
22                  focusedBorder: OutlineInputBorder(
23                      borderRadius: BorderRadius.circular(12),
24                      borderSide: BorderSide(color: Colors.transparent)), // OutlineInputBorder
25                  enabledBorder: OutlineInputBorder(
26                      borderRadius: BorderRadius.circular(12),
27                      borderSide: BorderSide(color: Colors.transparent)), // OutlineInputBorder
28                  border: OutlineInputBorder(
29                      borderRadius: BorderRadius.circular(12),
30                      borderSide: BorderSide(color: Colors.transparent)), // OutlineInputBorder
31                  contentPadding: EdgeInsets.all(8),
32                  hintText: placeholder,
33                  hintStyle: TextStyle(
34                      fontSize: 16,
35                      fontStyle: FontStyle.italic,
36                      fontWeight: FontWeight.w300,
37                      color: Colors.black,
38                  ), // TextStyle
39              ), // InputDecoration
40          ), // TextField
41      ), // Padding
42  ); // Container
43  }
```

- 

Pour utilise ce widget

- `textField(« placeholder », « controller », numeroDeLignes)` :  
permet d'afficher un conteneur avec un champs de texte à l'intérieure





# Fonctionnalités :

## Permissions:

Package: [https://pub.dev/packages/permission\\_handler](https://pub.dev/packages/permission_handler)

On demande les permissions dans l'initialisation de l'état de la class MyApp( classe principale de l'application ) qui est la classe qui sera attaché à l'écran avec la fonction runApp(MyApp()) avec argument MyApp() dans la fonction main()

```
92   class MyApp extends StatefulWidget {
93     @override
94     _MyAppState createState() => _MyAppState();
95   }
96
97   class _MyAppState extends State<MyApp> {
98     // ignore: unused_field
99     late Timer _timer;
100    int i = 0;
101    @override
102    initState() {
103      super.initState();
104      askPermissions();
105    }
106
107    askPermissions() async {
108      try {
109        await Permission.contacts.request().isGranted;
110        await Permission.sms.request().isGranted;
111      } catch (e) {
112        debugPrint(e.toString());
113      }
114    }
115
116    @override
117    Widget build(BuildContext context) {
```

## Envoie de mail :

Package : [https://pub.dev/packages/url\\_launcher](https://pub.dev/packages/url_launcher)

```
153 // *****
154 // This function sends email
155 // input : the subject and the message
156 // output : open the email application with filled content
157 // *****
158
159 sendMail(String subject, String message) {
160   String? encodeQueryParameters(Map<String, String> params) {
161     return params.entries
162       .map((e) =>
163         '${Uri.encodeComponent(e.key)}=${Uri.encodeComponent(e.value)}')
164       .join('&');
165   }
166
167   final Uri emailLauncherUri = Uri(
168     scheme: 'mailto',
169     path: 'contact@mypo.fr',
170     query: encodeQueryParameters(<String, String>{
171       'subject': 'APP MOBILE: ${subject}',
172       'body': '${message}',
173     }));
174
175   launch(emailLauncherUri.toString());
176 }
```

## Messages automatiques:

Package: <https://pub.dev/packages/telephony>

# Telephony

! This plugin currently only works on Android Platform

On appelle la fonction `initPlatformState` dans l'initialisation de l'état de la classe `SmsAuto`

```
95   @override
96   void initState() {
97     super.initState();
98     initPlatformState();
99   }
```

```
141   /*
142   | -This function ask for permissions and initiate the process to send messages
143   */
144
145   Future<void> initPlatformState() async {
146     final bool? result = await telephony.requestPhoneAndSmsPermissions;
147     final bool? resultContact = await Permission.contacts.request().isGranted;
148
149     if (result != null && result && resultContact!) {
150       telephony.listenIncomingSms(
151         onNewMessage: onMessage,
152         onBackgroundMessage: onBackgroundMessage,
153         listenInBackground: true);
154     }
155
156     if (!mounted) return;
157   }
```

Il faut ensuite definir les fonctions : `onMessage` et `onBackgroundMessage`

```

101  /*
102  -This function reacts to messages on the foreground
103  it gets incoming messages
104  Check if it contains any of the keys of activated alerts
105  Send back the alert message to the person
106  */
107  onMessage(SmsMessage message) async {
108      List<Alert> alerts = Boxes.getAutoAlert().values.toList().cast<Alert>();
109      debugPrint("onMessage called (Foreground)");
110      Future<bool> test = isContactInContactList(message);
111      int i = 0;
112      while (i < alerts[i].keys.length) {
113          Alert a = alerts[i];
114          bool tmp = isActive(message.body, a, await test);
115          bool grp = isInGroup(a, message.address);
116          if (tmp && grp) {
117              print(tmp);
118              Telephony.instance
119                  .sendSms(to: message.address.toString(), message: a.content);
120              if (a.notification) {
121                  String title = a.title;
122                  String content = a.content;
123                  String number = message.address.toString();
124                  _showNotification("Une reponse de l'alerte $title à été envoyée",
125                      "La reponse '$content' à été envoyée à $number");
126              }
127              return;
128          }
129          i++;
130      }
131  }

```

```

640 onBackgroundMessage(SmsMessage message) async {
641     debugPrint("onMessage called (background)");
642
643     final alerts = await getAlerts();
644     Future<bool> test = isContactInContactList(message);
645     int i = 0;
646     while (i < alerts[i].keys.length) {
647         Alert a = alerts[i];
648         bool tmp = isActive(message.body, a, await test);
649         bool grp = isInGroup(a, message.address);
650         if (tmp && grp) {
651             print(tmp);
652             Telephony.instance
653                 .sendSms(to: message.address.toString(), message: a.content);
654             if (a.notification) {
655                 String title = a.title;
656                 String content = a.content;
657                 String number = message.address.toString();
658                 _showNotification("Une reponse de l'alerte $title à été envoyée",
659                     "La reponse '$content' à été envoyée à $number");
660             }
661             return;
662         }
663         i++;
664     }
665 }

```

# Message programmé

Package : <https://pub.dev/packages/telephony>

On appelle la fonction périodiquement toutes les 20secondes la fonction sendSms() dans la fonction initState de la classe SmsProg()

```
31  @override
32  void initState() {
33    super.initState();
34    timer = Timer.periodic(
35      Duration(seconds: 20),
36      (Timer t) => {
37        sendSms(),
38      }); // Timer.periodic
39  }
```

```
71  void sendSms() async {
72    List<Scheduledmsg_hive> messages =
73      Boxes.getScheduledmsg().values.toList().cast<Scheduledmsg_hive>();
74    if (!messages.isEmpty) {
75      for (int i = 0; i < messages.length; i++) {
76        if (canbeSent(messages[i])) {
77          if (messages[i].confirm) {
78            confirmSend(messages[i]);
79          } else {
80            send(messages[i]);
81            saveMsgToRapport(messages[i]);
82
83            updateDate(messages[i]);
84            if (messages[i].notification) {
85              _showNotification(messages[i].phoneNumber,
86                messages[i].groupContact, messages[i].message);
87            }
88          }
89        } else if (fiveMinutesBeforeSend(messages[i]) &&
90          messages[i].countdown) {
91          String nom = messages[i].name;
92          _showNotificationBis(
93            "5 minutes avant l'envoi de l'alerte $nom", "accedez à l'appli");
94          messages[i].countdown = false;
95        }
96      }
97    }
98  }
```