Flutter

Fiche 7 : SMS et localisation

Table des matières

1 Objectifs à valider

ID	Objectifs
F12	Envoi de SMS
F13	Récupération de la localisation
F14	Persistance des données

2 Concepts

2.1 Introduction

Pour commencer le tutoriel, créez un nouveau projet (New Flutter Project) nommé *tuto7* dans votre repository de cours.

Pour ce tutoriel, nous aurons besoin de deux packages différents. Lancez la commande suivante pour installer le package *location* : **flutter pub add location**

Pour le deuxième package, nous ne pouvons pas utiliser *flutter pub add* pour l'installer. Le package est bien répertorié sur le gestionnaire de paquets *pub.dev* (voir sa page de présentation : https://pub.dev/packages/flutter_sms). Mais la version publiée sur le gestionnaire n'est pas à jour. Une version plus récente existe sur GitHub (https://github.com/fluttercommunity/flutter_sms).

Pour utiliser cette version, il faut modifier le fichier *pubspec.yaml* manuellement. Nous allons dire à flutter où chercher ce package. Ajouter les lignes suivantes, après la dépendance *location* que nous venons de rajouter :

```
flutter_sms:
  git:
    url: https://github.com/fluttercommunity/flutter_sms.git
    ref: master
```

Lancez ensuite la commande **flutter pub get** pour installer les dépendances de l'application.

Pour que le projet puisse compiler sur Android avec la librairie *location*, il faut également modifier le fichier de configuration *settings.gradle* au sein du dossier *android* du projet. Modifiez la version du plugin *org.jetbrains.kotlin.android* à la version 1.9.23.

Il faut également modifier le fichier de configuratin *build.gradle* au sein du dossier *android/app* (et pas celui du dossier *android*). Modifiez la version minimale d'Android compatible en passant le *minSdkVersion* à la version 21.

Le package *flutter_sms* nous permet d'envoyer des SMS depuis notre application, et le package *location* nous permet de récupérer la localisation de l'utilisateur.

[commit avec message : T07.1 Initialisation]

2.2 Détection de la plateforme

Créez un widget *HomeScreen* dans un dossier *views*. Copiez-y le code suivant :

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

  @override
  Widget build(BuildContext context) {
    String platform;

    if (kIsWeb) {
      platform = "Web";
    } else if (Platform.isAndroid) {
      platform = "Android";
    } else if (Platform.isIOS) {
      platform = "iOS";
    } else if (Platform.isWindows) {
      platform = "Windows";
    } else if (Platform.isMacOS) {
      platform = "macOS";
    } else if (Platform.isLinux) {
      platform = "Linux";
    } else {
      platform = "Unknown";
    }

    return Scaffold(
      appBar: AppBar(
        title: const Text("Tutoriel 7"),
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      ),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Center(
          child: Column(
            children: [
              Text("Hello from $platform!"),
            ],
          ),
        ),
      ),
    );
  }
}
```

Ce widget affiche un message dépendant de la plateforme sur laquelle tourne l'application, en utilisant une variable *platform*. Pour détecter cette plateforme, nous utilisons les constantes de la classes **Platform** du package **dart:io**. Si l'application tourne sur un téléphone Android, la constante *Platform.isAndroid* sera vraie. Si elle tourne un téléphone iOS, c'est la constante *Platform.isIOS* qui sera vraie. Nous pouvons déterminer sur quel OS tourne l'application grâce à ces constantes.

Pour détecter si l'application tourne dans un navigateur web, nous utilisons la constante **kIsWeb** du package **foundation**. Grâce aux différents *if/else if/else* présent dans ce code, nous pouvons donc déterminer quel est le contexte de l'application.

[commit avec message : T07.2 Détection de la plateforme]

2.3 Envoi de SMS

Modifiez la colonne du widget *HomeScreen* pour qu'elle contienne le code suivant :

```
child: Column(
  children: [
    Text("Hello from $platform!"),
    const SizedBox(height: 16),
    if (["Web", "Android", "iOS"].contains(platform))
      ElevatedButton(
        onPressed: () async {
          await sendSMS(
            message: "Test SMS",
            recipients: ["0456555321"],
          );
        },
        child: const Text("Send SMS"),
      )
    else
      const Text("Your platform doesn't allow you to send SMS…"),
    ],
),
```

Ce code utilise la fonction asynchrone **sendSMS** du package **flutter_sms** pour envoyer un SMS lorsque l'utilisateur appuie sur le bouton « Send SMS ». L'application n'envoie pas le SMS directement, mais utilise l'application SMS par défaut de l'appareil pour l'éditer et l'envoyer. Le message envoyé est « Test SMS » au numéro de téléphone « 0456555321 ». Les numéros commençant par 0456 ne sont pas attribués en Belgique, il ne s'agit donc pas d'un vrai numéro de téléphone.

Cette fonction n'est pas disponible pour toutes les plateformes. C'est seulement le cas sur les OS mobiles (Android ou iOS), et sur certains navigateurs web. À la place du bouton d'envoi, l'application affiche un message d'erreur pour les plateformes ne permettant pas l'envoi de SMS.

Attention, cela ne fonctionne pas non plus sur un simulateur Android. La fonction affiche une erreur dans le terminal dans ce cas. Si vous souhaitez tester l'envoi de sms, il est nécessaire d'utiliser un vrai téléphone et non un simulateur.

[commit avec message : T07.3 Envoi de SMS]

2.4 Récupération de la localisation

Créez un fichier *location_dialog.dart* et copiez-y le code suivant :

```
class LocationDialog extends StatefulWidget {
  const LocationDialog({super.key});

  @override
  State<LocationDialog> createState() => _LocationDialogState();
}

class _LocationDialogState extends State<LocationDialog> {
  LocationData? location;

  @override
  void initState() {
    _getLocation().then((value) => setState(() => location = value));
    super.initState();
```

```
  }

  Future<LocationData?> _getLocation() async {
    Location location = Location();

    var serviceEnabled = await location.serviceEnabled();
    if (!serviceEnabled) {
      serviceEnabled = await location.requestService();
      if (!serviceEnabled) {
        return null;
      }
    }

    var permissionGranted = await location.hasPermission();
    if (permissionGranted == PermissionStatus.denied) {
      permissionGranted = await location.requestPermission();
      if (permissionGranted != PermissionStatus.granted) {
        return null;
      }
    }

    return await location.getLocation();
  }

  @override
  Widget build(BuildContext context) {
    return AlertDialog(
      title: const Text("Your location"),
      content: Column(
        mainAxisSize: MainAxisSize.min,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text("Latitude: ${location?.latitude ?? 0.0}"),
          Text("Longitude: ${location?.longitude ?? 0.0}"),
        ],
      ),
      actions: [
        TextButton(
          onPressed: () => Navigator.pop(context),
          child: const Text("Dismiss"),
        ),
      ],
    );
  }
}
```

Ce code permet d'afficher un **AlertDialog** contenant la localisation de l'utilisateur. Les **Dialog** sont des widgets s'affichant par-dessus l'écran actuel. Les **AlertDialog** sont des *Dialog* affichant en plus un titre et une barre de boutons contenant les actions possibles de l'utilisateur par rapport à ce *Dialog*. Lorsque l'utilisateur tape à côté du *Dialog*, celui-ci est effacé et l'affichage revient à l'écran actuel. Il est également possible de revenir à cet écran en utilisant la fonction **Navigator.pop**, comme montré dans ce cas-ci avec le bouton d'action « Dismiss ».

Pour récupérer la localisation de l'utilisateur, ce widget utilise une fonction *_getLocation*. Dans cette fonction, il utilise les fonctions **location.serviceEnabled** et **location.requestService** pour s'assurer que l'utilisateur a bien activé la fonctionnalité de localisation sur son appareil. Il utilise ensuite les fonctions **location.hasPermission** et **location.requestPermission** pour s'assurer que l'utilisateur a bien donné la permission à cette application de récupérer sa localisation précise. Il utilise finalement la fonction

location.getLocation pour récupérer un objet de type **LocationData** contenant les coordonnées de latitude et de longitude de l'utilisateur et les afficher au sein du Dialog.

[commit avec message : T07.4 Récupération de la localisation]

2.5 Configuration de l'application

Sur certaines plateformes, il est nécessaire de configurer l'application pour avoir la possibilité de demander la localisation de l'utilisateur. Ce n'est pas le cas sur le web, qui permet à n'importe quel site web de demander sa localisation à l'utilisateur.

Sur Android, il faut ajouter les lignes suivantes au fichier **android > app > src > main > AndroidManifest.xml**, dans le tag manifest et avant le tag application :

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Sur iOS, il faut rajouter les lignes suivantes au fichier **iOS > Runner > Info.plist**, à la fin du tag dict :

```
<key>NSLocationWhenInUseUsageDescription</key>
<true/>
```

Si vous voulez faire tourner l'application sur d'autres plateformes que ces trois-là, faites des recherches pour vérifier quelles configurations apporter à l'application pour avoir le droit de demander la localisation de l'utilisateur.

[commit avec message : T07.5 Configuration de l'app]

2.6 Appel à un Dialog

Modifiez la colonne du widget *HomeScreen* pour qu'elle contienne le code suivant :

```
child: Column(
  children: [
    Text("Hello from $platform!"),
    const SizedBox(height: 16),
    if (["Web", "Android", "iOS"].contains(platform))
      ElevatedButton(
        onPressed: () async {
          await sendSMS(
            message: "Test SMS",
            recipients: ["0456555321"],
          );
        },
        child: const Text("Send SMS"),
      )
    else
      const Text("Your platform doesn't allow you to send SMS…"),
    const SizedBox(height: 16),
    ElevatedButton(
      onPressed: () => showDialog(
        context: context,
        builder: (context) => const LocationDialog(),
      ),
```

```
      child: const Text("Retrieve location"),
    ),
  ],
),
```

Ce code utilise la fonction **showDialog** pour faire appel au *Dialog* créé précédemment lorsqu'on appuie sur le bouton « Retrieve location ».

[commit avec message : T07.6 Appel à un dialog]

2.7 Test de l'application sur un téléphone mobile

Lancez l'application pour vérifiez son fonctionnement. Certaines fonctionnalités de cette application ne seront pas disponibles sur toutes les plateformes ou sur émulateur. Testez l'application sur un téléphone mobile. Pour lancer l'application sur un téléphone Android, suivez le tutoriel suivant : https://developer.android.com/studio/run/device?hl=fr. Pour lancer l'application sur un téléphone iOS, suivez le tutoriel suivant : https://medium.com/front-end-weekly/how-to-test-your-flutter-ios-app-on-your-ios-device-75924bfd75a8

Exercices

2.8 Introduction

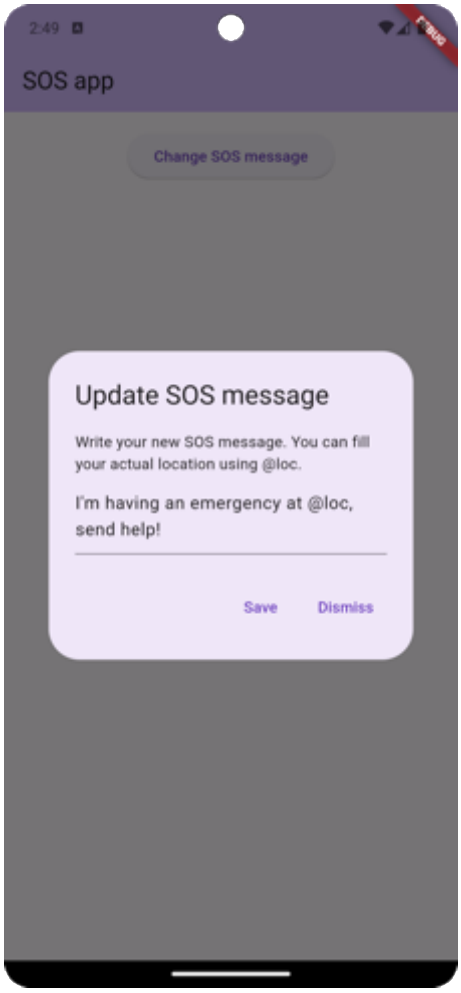
Veuillez créer un nouveau projet (New Flutter Project) nommé *ex7* dans votre repository de cours.

Dans cet exercice, vous allez créer une application permettant d'envoyer des messages d'urgence. L'application enregistre le message à envoyer et une liste de numéros de téléphones à contacter en cas d'urgence. Des dialogues permettent de modifier ces informations. Un bouton permet d'envoyer le message à tous les numéros de téléphones enregistré en cas d'urgence.

Pour cet exercice, nous vous demanderons de faire tourner l'application sur un téléphone mobile, iOS ou Android. Utilisez un téléphone physique si possible, les émulateurs ne permettant pas de tester correctement toutes les fonctionnalités de l'application.

2.9 Modification du message d'urgence

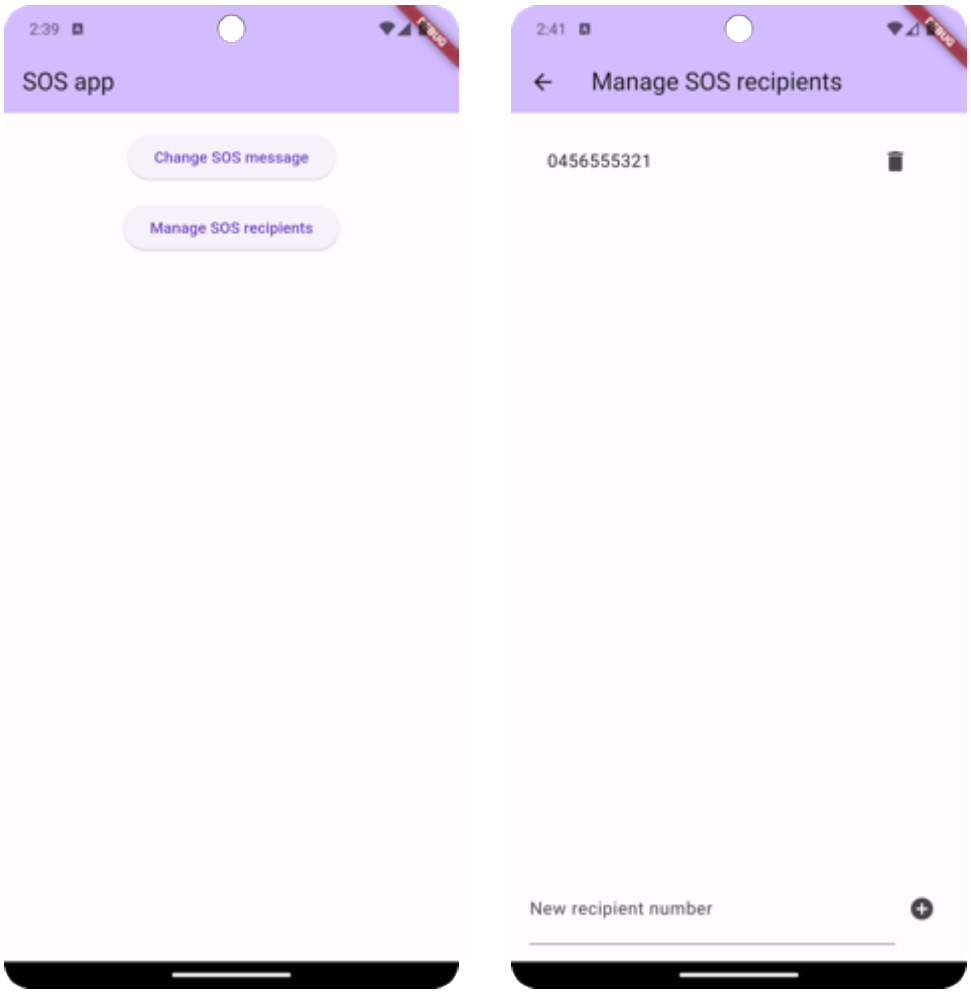
Créez un bouton permettant d'ouvrir un Dialog affichant un champ de texte. Ce Dialog permet soit d'enregistrer le contenu du champ de texte comme le nouveau message d'urgence à envoyer, soit d'annuler et de garder le message d'urgence précédent. Lorsque l'on affiche le Dialog, ce champ de texte doit contenir initialement le message d'urgence précédent. À l'ouverture de l'application, le message d'urgence initial doit être : « I'm having an emergency at @loc, send help! ». À cette étape, votre application pourra ressembler aux captures d'écran suivantes.



⚡ **[commit avec message : F12.1 Modification du message d'urgence]** ⚡

2.10 Enregistrement des numéros de téléphones d'urgence

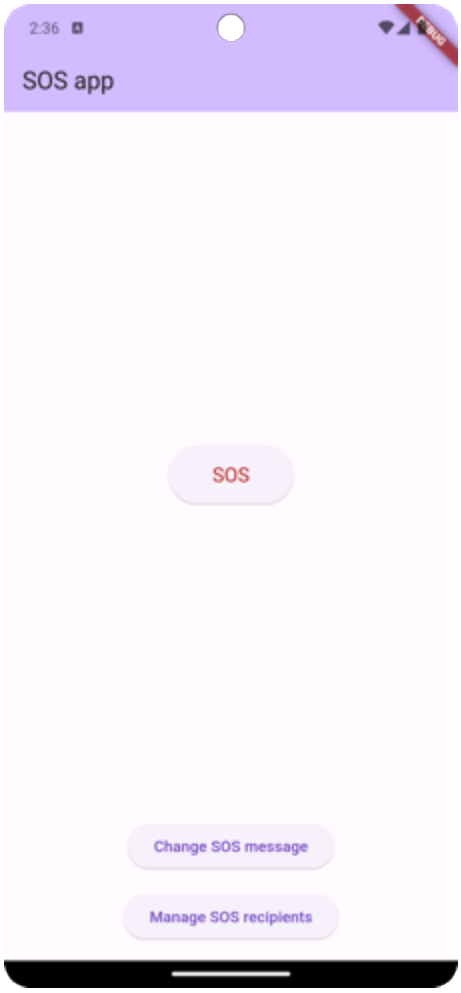
Créez un bouton permettant de naviguer vers un écran affichant les numéros de téléphones d'urgence actuels. Pour chaque numéro de la liste, un bouton permet de l'en retirer. En bas de la liste, un champ de texte et un bouton permettent de rajouter un numéro de téléphone. Au lancement de l'application, la liste des contacts d'urgence doit être vide. À cette étape, votre application pourra ressembler aux captures d'écran suivantes.



⚡ **[commit avec message : F12.2 Enregistrement des numéros de téléphones d'urgence]** ⚡

2.11 Envoi du message d'urgence

Ajoutez un bouton à votre application permettant d'envoyer par SMS le message d'urgence enregistré aux contacts d'urgence enregistrés. Si la plateforme de l'utilisateur ne permet pas l'envoi de SMS, affichez à la place un Dialog avec un message d'erreur. À cette étape, votre application pourra ressembler à la capture d'écran suivante.



⚡ **[commit avec message : F12.3 Envoi du message d'urgence]** ⚡

2.12 Message d'urgence avec localisation

Modifiez l'envoi du message d'urgence pour qu'en son sein le texte « @loc » soit remplacé par les coordonnées GPS de la localisation de l'utilisateur, par exemple « (lat : 50.849268, lon : 4.450863) ». Faites en sorte de récupérer la localisation de l'utilisateur au moment où il/elle appuie sur le bouton d'envoi du message d'urgence.

⚡ **[commit avec message : F13.1 Message localisé]** ⚡

2.13 Persistance des données

Faites persister les données de l'application, le message et les numéros de téléphones d'urgence, à travers plusieurs lancement de l'application.

Cette compétence n'a pas été vue durant le tutoriel. Vous devrez l'apprendre par vous-même. Utilisez le package **Shared Preferences**. Voici sa documentation :

https://pub.dev/packages/shared_preferences

⚡ **[commit avec message : F14.1 Persistance des données]**