BUT2 RA Développement mobile - cours 4 :

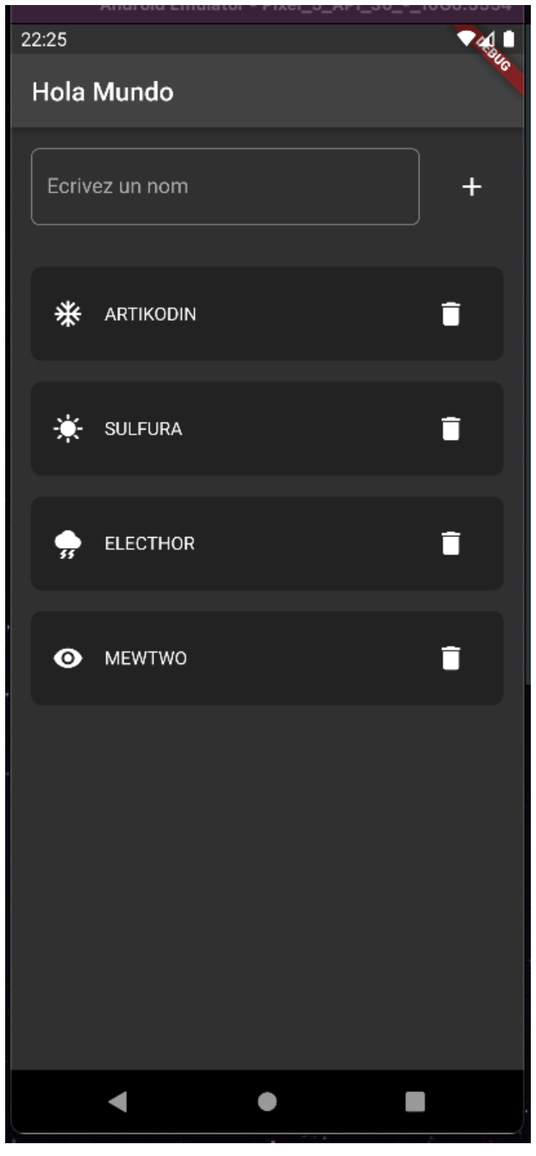
Idéalement, utiliser VSCode et Linux pour ce cours. Sinon, utilisez DartPad : `https://dartpad.dev/`

Ce cours/TD est la suite directe du cours précédant.

- Rappel
- Ajouter un élément dans une liste
- Passer de Stateless à Stateful
- Provoquer un changement à l'écran
- Amélioration légère de l'UX
- Le controlleur du Textfield
- Utiliser ce qui est écrit dans le Textfield
- Ajouter des conditions avant l'ajout
- Supprimer un élément de la liste

Rappel

Pour rappel, on continue à partir de l'écran réalisé dans le cours précédant :



Ajouter un élément dans une liste

On souhaite pouvoir cliquer sur le bouton "+" et ajouter un élément dans notre liste appelée "pokedex". Pour l'instant, on ignore ce qui est écrit dans le `Textfield`, afin de simplifier les choses.

On ajoute cette fonction dans le paramètre onPressed de notre bouton "+"

```
pokedex.add(
  const Pokemon('Test', Icons.question_mark),
);
```

Quand on clique sur "+", on rencontre cependant cette erreur :

Passer de Stateless à Stateful

L'erreur de la page précédante est causée par le fait que notre liste est dans une classe Stateless. Il faut donc que notre classe `HomePage` devienne Stateful. N'oubliez pas de HOT RESTART votre appli après ces changements.

```
class HomePage extends StatelessWidget { // <===== Stateless
  const HomePage({super.key});
    ...
```

va devenir :

```
class HomePage extends StatefulWidget { // <===== Stateful
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  ... // <===== notre liste "pokedex" se trouve ici maintenant
```

De meme, notre liste (qui se trouve maintenant dans la classe `_HomePageState`) va passer de :

```
final pokedex = const <Pokemon>[
  Pokemon('Artikodin', Icons.ac_unit),
  ...
];
```

à cela :

```
final pokedex = <Pokemon>[ // <===== on enlève le mot clé "const" ici
  Pokemon('Artikodin', Icons.ac_unit),
  ...
];
```

(NB : normalement, il faudrait ajouter `const` devant chaque Pokemon, mais pour simplifier le cours, nous allons ignorer cela tant que votre IDE VSCode ne vous met pas de warnings/erreurs.)

Provoquer un changement à l'écran

A ce stade, lorsque l'on clique sur le bouton "+", il ne se passe rien. Le nouvel objet Pokemon est bien ajouté à la liste mais il faut dire à Flutter de rafraichir la page pour afficher la nouvelle liste.

Pour cela, on appelle la fonction `setState`, juste après la méthode `.add()` (toujours dans le `onPressed` du `IconButton`) :

```
setState(() {});
```

Amélioration légère de l'UX

L'UX correspond à User Experience. C'est la facon dont votre utilisateur va utiliser et percevoir votre application.

Notre problème ici est que le nouvel élément s'ajoute à la fin. Au bout de 5 ou 7 ajouts (selon la taille du téléphone), on a l'impression que l'ajout ne marche plus, alors qu'il suffit de scroller pour voir que notre code fonctionne.

Ce n'est pas à l'utilisateur d'etre obligé de scroller, c'est au développeur d'adapter son code pour permettre une expérience fluide.

Ainsi, on remplace la fonction `.add()` par une fonction qui va insérer le nouvel élément à l'index zéro.

Le controlleur du Textfield

Pour cela, nous allons devoir ajouter un controlleur dans notre `TextField`.

Dans notre classe `_HomePageState`, on ajoute un controlleur :

```
late final TextEditingController controller;
```

- `late` signifie qu'il sera instancié plus tard (car nous n'avons pas écrit à quoi il est égal)
- `late final` signifie qu'il ne changera plus après la première instanciation.

Pour s'assurer que le controlleur sera instancié une seule fois, on override la méthode `initState()`.

```
@override
void initState() {
  controller = TextEditingController();
  super.initState();
}
```

Pour libérer des ressources dans l'application, il faut penser à disposer ce que l'on a instancié dans `initState()`, via la méthode `dispose()`

```
@override
void dispose() {
  controller.dispose();
  super.dispose();
}
```

Et enfin, il faut lier le controlleur que l'on vient de créer avec le widget `Textfield` via ses paramètres.

```
TextField(
  controller: controller,
  ...
```

Utiliser ce qui est écrit dans le Textfield

Dans le bouton, on peut récupérer le contenu du Textfield avec `controller.text` dans notre méthode `onPressed`

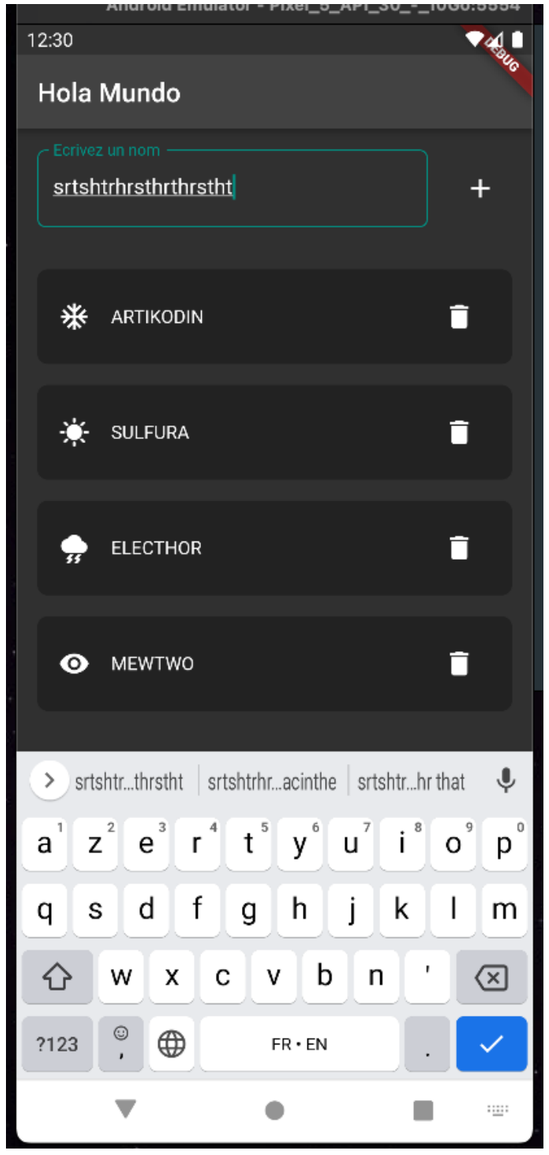
```
IconButton(
  icon: const Icon(Icons.add),
  onPressed: () {
    final nameToAdd = controller.text;
    ...
```

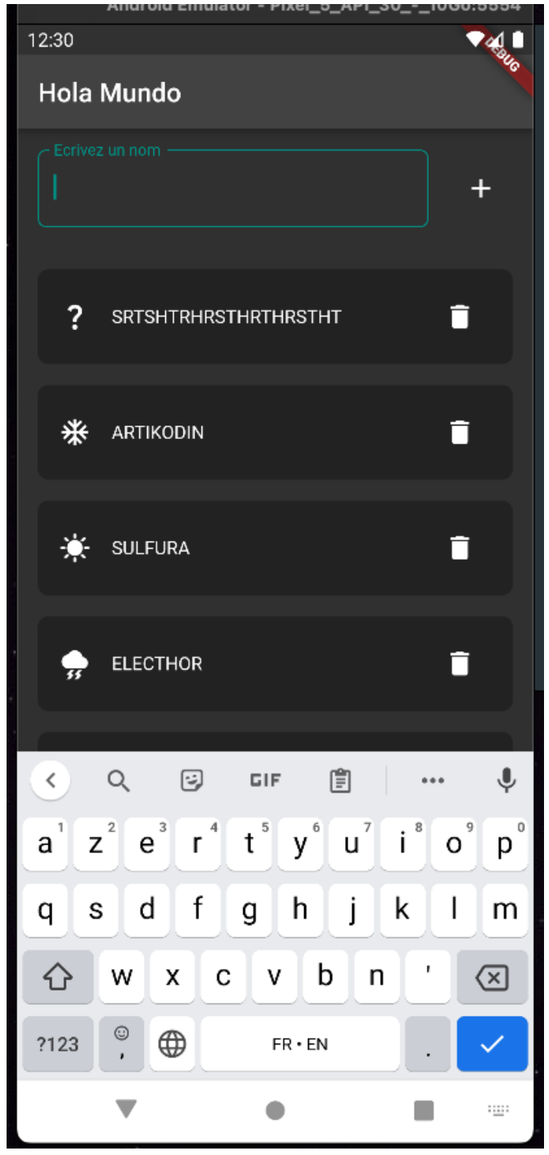
Vous pouvez alors définir le nom du Pokémon avec le contenu écrit dans le `Textfield`.

```
Pokemon(
  nameToAdd,
  Icons.question_mark,
),
```

Cherchez également un moyen de vider le champ de text après l'ajout (vous pouvez le faire avec le controller).

Vous devez obtenir le résultat suivant :





Ajouter des conditions avant l'ajout

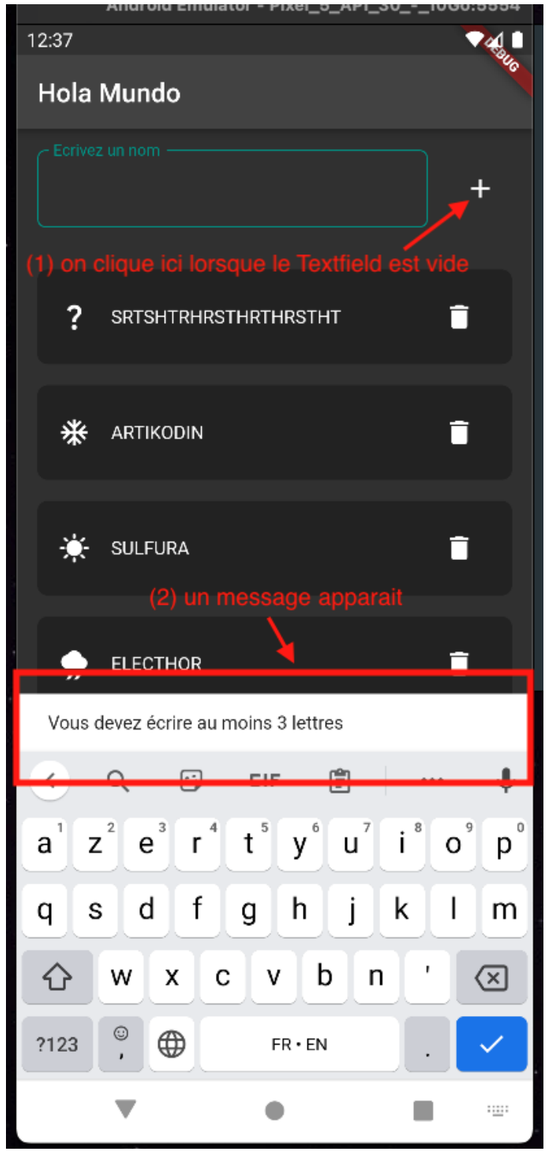
On va ajouter des conditions sur le mot écrit dans le `Textfield`. Avant l'ajout, il faut que :

- Le mot fasse au minimum 3 lettres
    - Pensez à traiter le cas où l'utilisateur écrit uniquement des espaces
- Le mot ne doit pas etre déjà présent dans la liste
    - Pensez aux cas où la liste contient un mot comme "Sulfura" et que l'utilisateur écrit "sulfura" (sans majuscule), ou "sulfura " (avec des espaces dans le champ de texte")

Si ces conditions ne sont pas remplie, une snackbar va s'afficher pour indiquer à l'utilisateur ce qui ne va pas.

Vous devez obtenir le résultat suivant :

Au moins 3 lettres



Le mot est déjà utilisé

Supprimer un élément de la liste

Vous devez chercher comment supprimer un élément de la liste.

Le bouton de supression est dans notre widget `TheAmazingRow`. Or, notre liste de pokemon est dans son widget parent `HomePage`. Vous devez donc trouver un moyen d'agir sur un élément du widget parent (`HomePage`), en cliquant dans le widget enfant (`TheAmazingRow`).