IT MINDS

Flutter Pokédex

Setup your IDE

1. Download and install the flutter SDK
2. Setup your favorite code editor
3. Optional: Make sure everything works

Clone repository

Clone the GitHub repository to get started.

Start the Pokédex

After you have cloned the project, you will need to run `flutter packages get`. This will fetch the dependencies of the project. When this completes you are ready to start the app.

In Visual Studio Code this is done by first clicking on “No Device” in the bottom right and thereafter selecting the installed emulator, you wish to use.

In Android Studio click on “AVD Manager”-icon in the upper right corner. This will give you a menu in which you can choose your favorite emulator.

When the emulator is up and running, you can start the app in debug mode by navigating to Debug -> Start Debugging in the topmenu, or you can just start the app by running `flutter run`. Debug mode is of course preferable.

Build your Pokédex

This is your Pokédex now.

It isn't pretty and you can't scroll.

Let's fix that.

IT MINDS

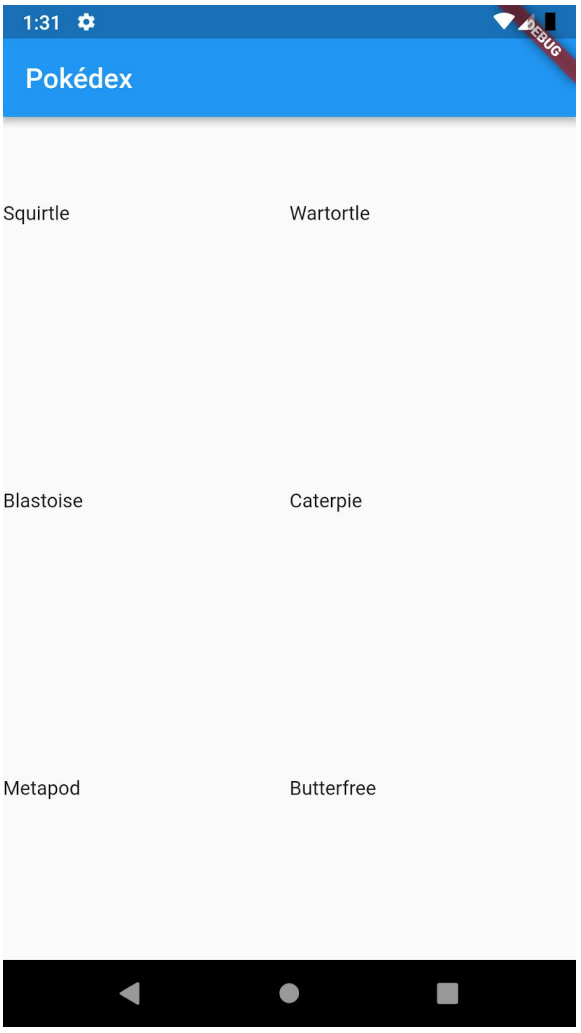
1) Using a GridView

Use a GridView to enable scrolling and show the name of the Pokémons in a grid.

Have a look at the GridView cookbook for inspiration.

You can also have a look at the API documentation, but it is quite extensive.

The final result should look something like what you see on the right.



IT MINDS

2) Your first widget

We want to show more than just the name of the Pokémon. Let's start by creating a widget for that.

Create a stateless widget called `PokemonGridItem`.

The widget should take a `Pokemon` as a constructor argument and save it in a property on the class.

Use the `Text` widget to show the name of the Pokémon.

Use your `PokemonGridItem` in the GridView.

An example of a basic stateless widget:

```
class MyWidget extends StatelessWidget {
  ...

  @override
  Widget build(BuildContext context) {
    return Text(...);
  }
}
```

3) Showing images

Use the `Column` and `CachedNetworkImage` widgets to show the Pokémon images.

Have a look at the Image cookbook for inspiration.

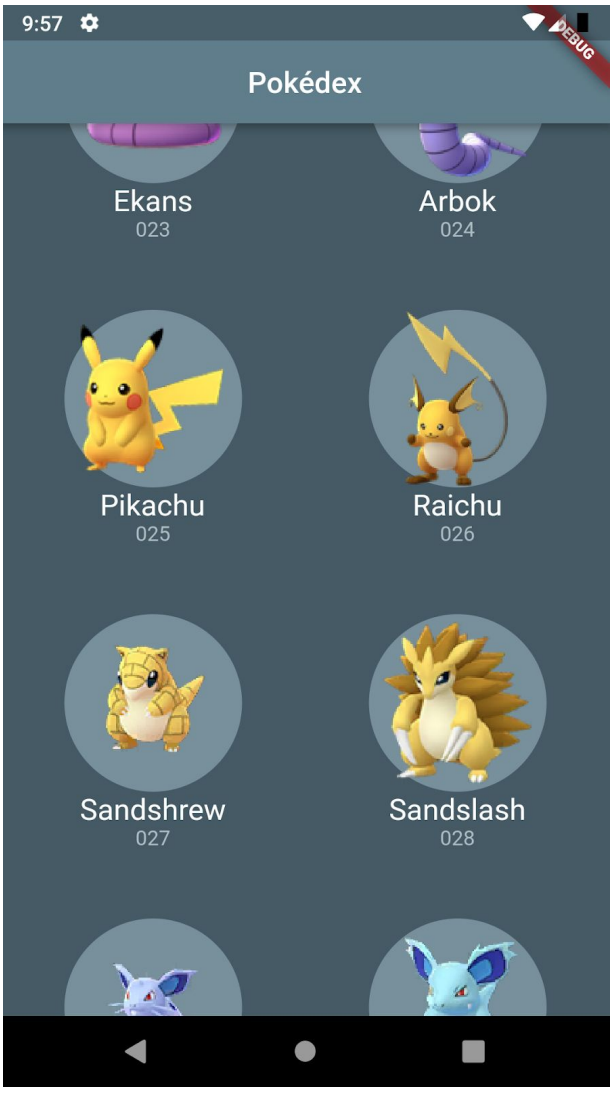
Hint: The `cached_network_image` package, is already installed in the project.

Go crazy with styling.

Try using the `Container` widget to add some padding and colors.

Have a look at the widget catalog for inspiration.

The final result could look something like what you see on the right.



4) Showing details

The `Pokemon` class contains more information than we can show in the grid view.

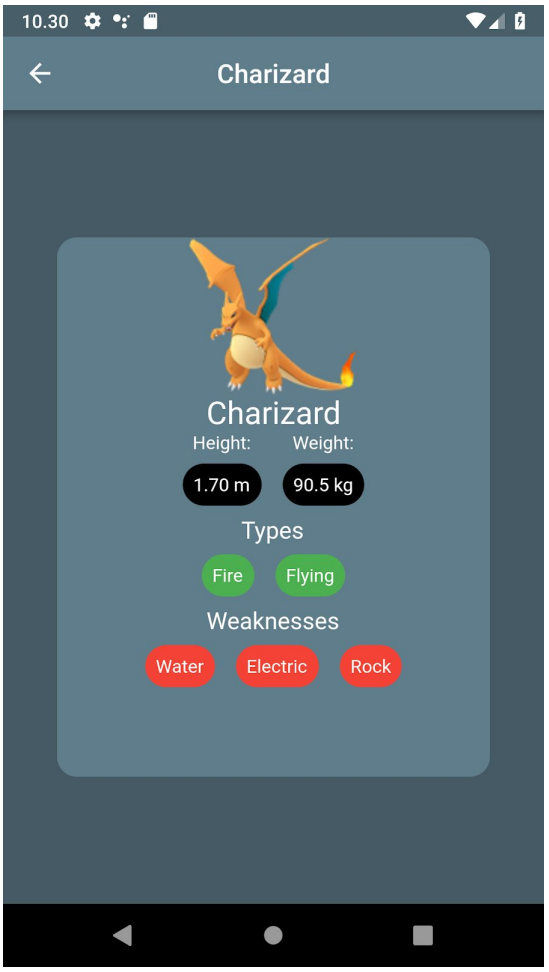
Let's create a screen where we can show more details such as the height, weight and weaknesses of the Pokémon.

Create a stateless widget called `DetailsScreen`. It should get a `Pokemon` just like the `PokemonGridItem` and show all its details.

The Pokédex should open this screen when you tap on a Pokémon in the grid view.

Use the `GestureDetector` widget for detecting taps. Have a look at the GestureDetector cookbook for inspiration.

Use the `Navigator` to navigate to the details screen on tap. Have a look at the Navigator cookbook step 2 for inspiration.



5) Animations

To add a hero animation to your pokémon images, wrap the images in a `Hero` widget.

Give the `Hero` widget a unique tag, like the pokémons name or number.

```
Hero(
  child: CachedNetworkImage(imageUrl: pokemon.image),
  tag: pokemon.number,
),
```

IT MINDS

Extra improvements

Show evolutions

The API also exposes evolutions. Update the `DetailsScreen` to support evolutions by showing both name and image of the evolutions. You will need to update your model.

An example from the API:

```
"prev_evolution": [
  {
    "number": "001",
    "name": "Bulbasaur"
  }
],
"next_evolution": [
  {
    "number": "003",
    "name": "Venusaur"
  }
]
```

Catch Pokémons

Make it possible to mark which pokémons you have caught.

You could implement this by adding a “onLongPress” callback to the `GestureDetector`, and use a stateful widget to store the list - or maybe a BLoC?

The `PokemonGridItem` should also be edited, so it reflects the changes by showing some kind of icon on the caught pokémons.

Search field

There is a lot of pokémons in the list. A useful feature would be to search for names. Implement a search feature on the homepage.

You could also implement filtering by type or sorting by height.