11. AUGUST 2021

SIDEKICK

ADD LANGUAGES

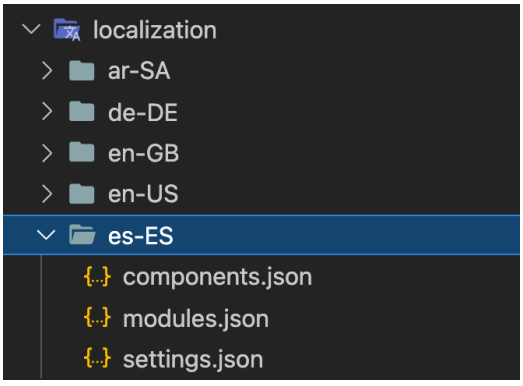
Rebar Ahmad
SIDEKICK CONTRIBUTOR

ADD A NEW LANGUAGE

To add a new language to sidekick, please follow the listed steps below.

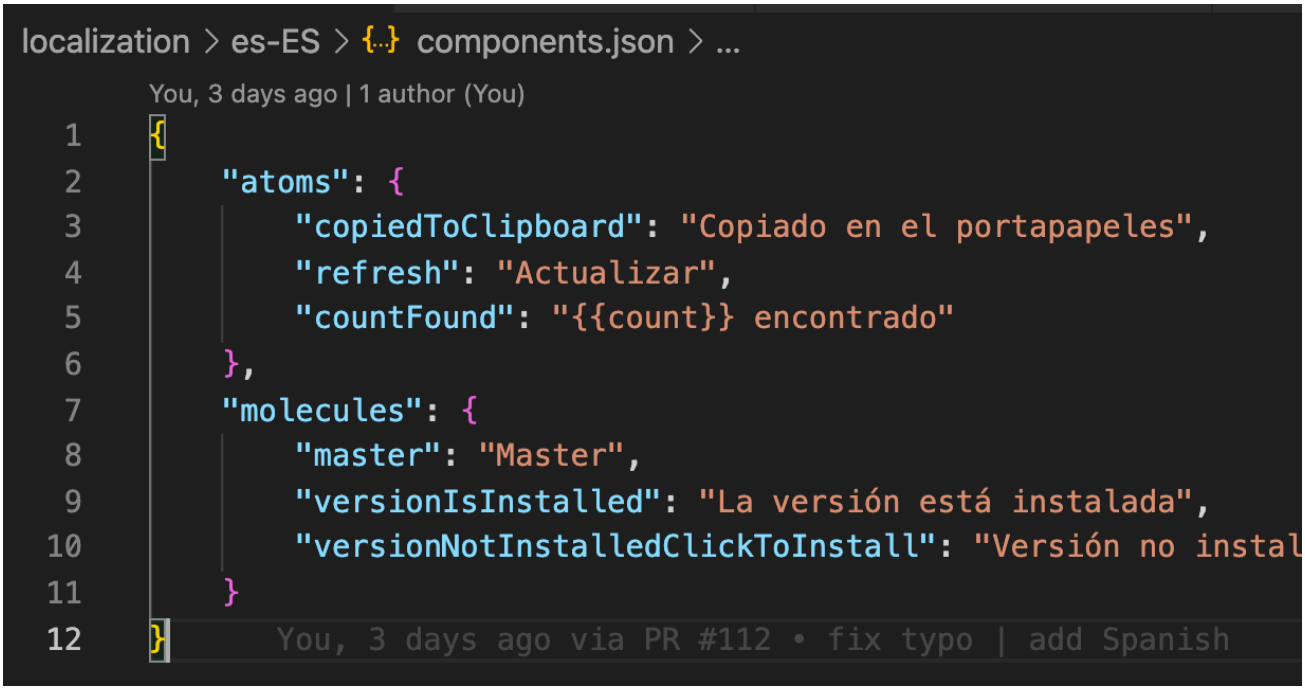
1. ADD YOUR NEW LANGUAGE

At first, before telling Sidekick, that there's a new language in the app, translate the existing files into your target language. For this you'll find a folder named **localization** in the root folder of sidekick:



Picture 1: Folder structure

In *Picture 1* you can see the **json** files. These are the actual texts which comes with a corresponding key. Sidekick only uses the keys to identify a translation.



Picture 2: Key-Value storage of translations

The actual translation can be a simple key that stands for a specific translation as String.

Example 1: **Simple translation**:

json:

{

„refresh“: „Actualizar“

}

Call in code (only for Devs):

```
I18Next.of(context).t('components:refresh')
```

Example 2: **Nested translation**

{

„atoms“: {

„refresh“: „Actualizar“

}

}

Call in code (only for Devs):

```
I18Next.of(context).t('components:atoms.refresh')
```

Example 3: **Passing variables**

{

„atoms“: {

"countFound": "{{count}} encontrado"

}

}

Call in code (only for Devs):

```
I18Next.of(context).t('components:atoms.countFound', variables: {
   count': count.toString(),
}),
```

To summarize this: You can specify the way translations will be called in the actual source code! we got the file: **components.json** where the name of the file **components** is also the namespace for the translation. After that a **:** is following to tell the **I18Next** package that now he can go and find the first root-key in the namespace. In *Example 3* you can see that **atoms** is one of the root keys. Now with every **.** you can speficy the nested key inside **atoms**. You can structure and nest as deep as your app needs, but be aware and try to keep it as simple as possible.

If you copied an existing language and renamed it like: **jp-JP** for example for Japanese, add all your translations to the existing keys, you can step forward and add the configurations for Flutter.

2. ADD FLUTTER CONFIGURATION

1. First you need to tell Flutter where to find the new folder of your translation. For that just add a new assets in the **pubspec.yaml** entry like shown:

```
flutter:
  uses-material-design: true
  assets:
    - assets/logo.png
    - localization/ar-SA/
    - localization/de-DE/
    - localization/en-GB/
    - localization/en-US/
    - localization/es-ES/
```

Picture 3: Add assets enrty

2. At the end you only need to add a new **Locale** to the Language-Manager which you can find in **lib > i18n > language_manager.dart**:

```
final List<Locale> _supportedLanguagesCodes = [
  const Locale('en', 'GB'),
  const Locale('en', 'US'),
  const Locale('ar', 'SA'),
  const Locale('de', 'DE'),
  const Locale('es', 'ES'),
];
```

Picture 4: Add Locale

That's it! That's all you need to do to add a new translation. Flutter will now automatically detect and read the new language and provide it in the **DropdownMenu**.