**Documentación del proceso de implementación de multilenguaje:**

**Introducción:**

Existen varias formas implementar el multilenguaje en una aplicación, a través de complementos o de forma nativa por medio i18n.

En el caso de EducaDroid, se implementó el multilenguaje de forma dinámica utilizando ngx-translate. (https://github.com/ngx-translate/core)

La principal ventaja de este complemento es poder configurar el path de los archivos .json diccionario y la capacidad de seleccionar el lenguaje de forma dinámica.

**Procedimientos:**

1. *Instalación de la librería ngx-translate:*

Ejecutar desde la consola el siguiente comando:

npm install @ngx-translate/core --save

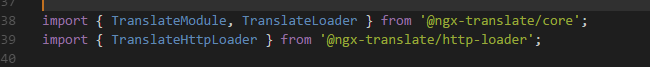
1. *Instalación del Loader de ngx-traslate:*

Esta librería para poder seleccionar el path donde se encuentran los diccionarios. En el caso de aplicaciones creadas con ionic es necesario mapear los diccionarios a la carpeta assetts. El loader se instala con el siguiente comando:

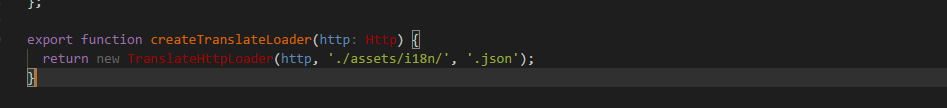
npm install @ngx-translate/http-loader --save

1. *Importar y configurar librerias al modulo principal:*

En el **app.module.ts** se deben importar las librerías instaladas



Luego, para poder setear el path donde se levantaran los diccionarios es necesario crear un método que devuelva una instancia del TranslateHttpLoader con el path del archivo.



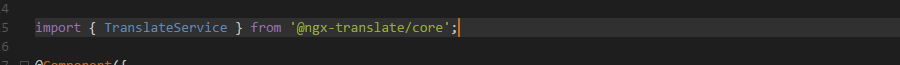
En nuestro caso, utilizaremos el path “src/assets/i18n/”, todo archivo con extensión .json

Por último, en el @NgModule hay que importar el TranslateModule, configurando el loader como se indica en la foto:



1. *Setear el idioma por defecto y los disponibles:*

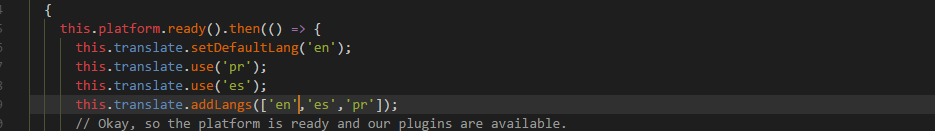
En el **app.component.ts**  importamos el TranslateService



Lo agregamos al constructor



Seleccionamos el idioma default y los disponibles, en la promesa de platform.ready()

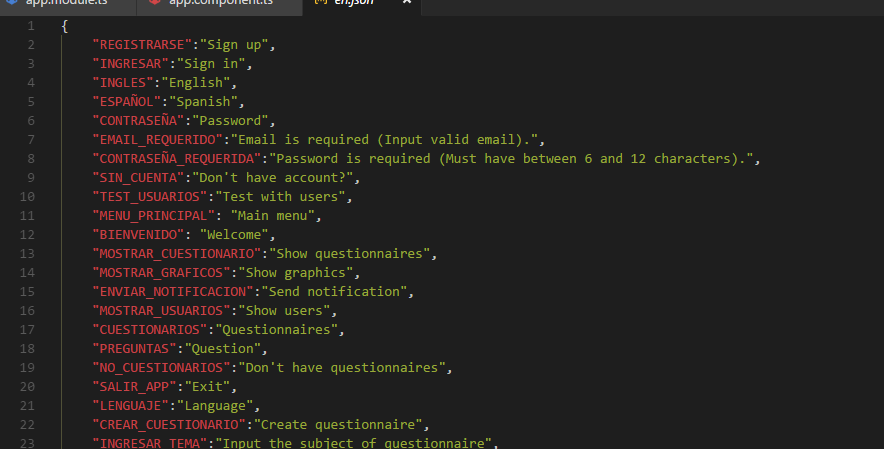


En nuestro caso, hemos agregado Inglés, Español y Portugués. Se recomienda hacer un “use” de todos los lenguajes para que se cacheen los diccionarios o bien, subcribirse al observable que retorna “use”.

El método “use” selecciona a través del TranslateHttpLoader el lenguaje que deseamos utilizar.

1. *Agregar diccionarios:*

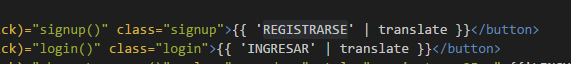
Los diccionarios son archivos con extensión .json. En nuestro caso, nuestros archivos son en.json, es.json y pr.json. El formato de los archivos es el siguiente:



**Importante:** Las claves de entre los distintos .json sean iguales.

1. *Uso de los diccionarios:*

Para hacer referencia a las palabras del diccionario desde el HTML, se debe escribir la KEY del diccionario junto con el pipe ‘translate’ de la siguiente forma:

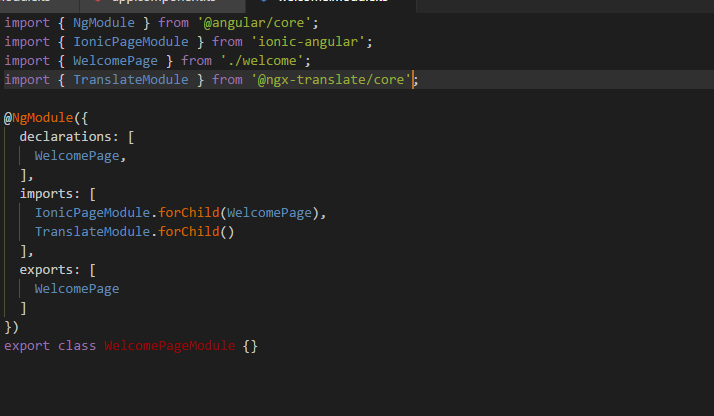


Si se desea hacer referencia a las KEYS de los diccionarios desde los .ts podemos utilizar el método “instant” del TranslateService de la siguiente forma:



1. *Importar TranslateService en pages LazyLoading (SOLO LazyLoading):*

En el caso de utilizar pages LazyLoading, deberá importarse a cada modulo que desee traducirse (xxxxxx.module.ts) la librería TranslateModule para poder utilizar el pipe de la siguiente forma:



También se puede importar a un SharedModule y traerlo a todos módulos de cada pages.