Guía internacionalización

Para la internacionalización en React se empleará el framework i18next. Las librerías mínimas necesarias para emplear este framework son dos:

- i18next – framework de localización en JavaScript
- react-18next – La implementación para React de i18next

---

npm install i18next react-i18next --save

Para su creación habrá un fichero inicial **i18n.js** que controlará las traducciones, es decir, contendrá directa o indirectamente todas las traducciones. Será el fichero raíz. Aquí será donde se puedan hacer configuraciones referentes a i18next. Para el correcto mantenimiento y tener un orden este fichero irá dentro de una carpeta con el nombre del framework (i18n).

src > i18n > JS i18n.js > ...

```
1  import i18n from 'i18next';
2  import Vocabulary_en from './locales/en/Vocabulary_en.json';
3  import Vocabulary_es from './locales/es/Vocabulary_es.json';
4
5  const languages_avaliable = [ 'en', 'es'];
6
7  i18n.init({
8
9      interpolation:{escapeValue: false},
10     fallbackLng: languages_avaliable.includes(navigator.language) ? navigator.language : 'es', //Language by default
11
12     resources: {
13         en: {
14             Vocabulary: Vocabulary_en,
15         },
16         es: {
17             Vocabulary : Vocabulary_es,
18         }
19     }
20 });
21
22 export default i18n;
```

Ilustración 1. Archivo i18n.js, configuración de la traducción

Este fichero deberá importarse en el fichero principal (**index.js**) de la aplicación pasándose como prop del componente *I18nextProvider*.

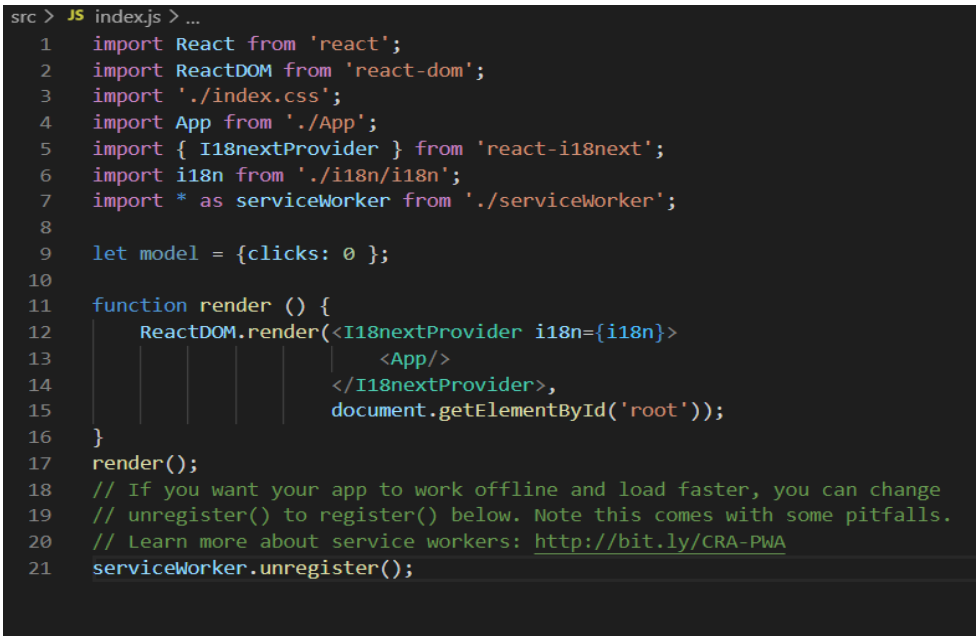

Ilustración 2. Importación fichero de traducciones sobre index del proyecto.

Para realizar traducciones separadas para cada idioma y por cada fichero se ha instalado otro paquete, que nos permitirá controlar las traducciones. Estas irán en una carpeta llamada locales dentro de la carpeta del framework. Se crearan subcarpetas que tendrán como nombre la clave del idioma que tendrán los archivos.

---

npm install --save i18next-xhr-backend

Puede existir un único fichero para cada idioma. Esté deberá tener el mismo título o uno por cada componente de la aplicación. Estos serán archivos de formato json donde vendrá identificado lo que se va a traducir seguido de su traducción.

Ilustración 3. Organización ficheros traducción

Para añadir la traducción en los componentes, se recogerá del props t. Para utilizarlo se añadirá el texto haciendo uso de su clave y este irá rodeado entre paréntesis y t. Finalmente unas llaves cerraran esta traducción.

```
class About extends Component {
    render () {
        const { classes, t } = this.props;

        return (
            <div>
                <div className={classes.top}>
                    <Avatar className={classes.avatar} alt="IsabelDiaz" src={avatar} className={classes.large} />
                    <Typography variant="subtitle1" className={classes.name} >{t('Name')}</Typography>
                </div>
```

Ilustración 4. Ejemplo traducción en componente.

Por último, sobre el componente se importará la librería de react-i18next *withTranslation.* Está irá sobre el componente en la exportación para indicar el uso de la traducción en este caso empleando el fichero con el mismo nombre que el componente.

```
import { withTranslation } from 'react-i18next';

export default withStyles(styles)((withTranslation('Vocabulary'))(About));
```

Ilustración 5. Importación y agregación del uso del traductor en el componente.