

## Instalación y Configuración:

(<https://next-intl-docs.vercel.app/docs/getting-started/app-router/with-i18n-routing>)

1) Abre example-i18n con Visual Studio Code y en la terminal ejecuta el siguiente comando:

```
npm install next-intl
```

2) Crear carpeta en la raíz del proyecto:

```
/messages
```

3) Dentro de la carpeta /messages crear los archivos:

```
en.json  
es.json
```

4) Dentro de en.json:

```
{  
  "Index": {  
    "title": "Hello world!"  
  }  
}
```

Dentro de es.json:

```
{  
  "Index": {  
    "title": "Hola mundo!"  
  }  
}
```

5) Configuración en next.config.mjs:

Sustituir todo el contenido existente por este otro:

```
import createNextIntlPlugin from 'next-intl/plugin';  
  
const withNextIntl = createNextIntlPlugin();  
  
/** @type {import('next').NextConfig} */  
const nextConfig = {};  
  
export default withNextIntl(nextConfig);
```

6) Crear archivo i18n.ts dentro de la carpeta /src

7) Dentro del archivo i18n.ts poner este código:

```
import {notFound} from 'next/navigation';
import {getRequestConfig} from 'next-intl/server';

// Can be imported from a shared config
const locales = ['en', 'de'];

export default getRequestConfig(async ({locale}) => {
  // Validate that the incoming `locale` parameter is valid
  if (!locales.includes(locale as any)) notFound();

  return {
    messages: (await import(`../messages/${locale}.json`)).default
  };
});
```

IMPORTANTE:

En esta línea: `const locales = ['en', 'de'];` sustituir 'de' por 'es'

8) Crear archivo `middleware.ts` dentro de la carpeta `/src`

9) Dentro del archivo `middleware.ts` poner este código:

```
import createMiddleware from 'next-intl/middleware';

export default createMiddleware({
  // A list of all locales that are supported
  locales: ['en', 'de'],

  // Used when no locale matches
  defaultLocale: 'en'
});

export const config = {
  // Match only internationalized pathnames
  matcher: ['/', '/(de|en)/:path*']
};
```

IMPORTANTE:

En esta línea: `locales: ['en', 'de'],` sustituir 'de' por 'es'

En esta línea: `defaultLocale: 'en'` sustituir 'en' por 'es'

En esta línea: `matcher: ['/', '/(de|en)/:path*']` sustituir 'de' por 'es'

10) Crear carpeta `[locale]` dentro de la carpeta `/src/app`

11) Arrastrar los archivos 'layout.tsx' y 'page.tsx' de la carpeta /src/app dentro de la carpeta /src/app/[locale]

12) En app/[locale]/layout.tsx:

1- Agregar este parametro y su tipo:

```
params: {locale}
```

```
params: {locale: string};
```

2- Modificar este: import "../globals.css"; por este otro import "@app/globals.css";

3- Modificar el atributo lang del documento html: <html lang='en'> por <html lang={locale}>

El archivo 'layout.tsx' debe quedar de esta forma:

```
import { ReactNode } from "react";
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import "@app/globals.css";

const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
  title: "Example i18n width Next.js 14 and TS",
  description: "Example i18n width Next.js 14 and TS",
};

export default function RootLayout({
  children,
  params: {locale}
}: Readonly<{
  children: ReactNode;
  params: {locale: string};
}>) {
  return (
    <html lang={locale}>
      <body className={inter.className}>{children}</body>
    </html>
  );
}
```

13) En app/[locale]/page.tsx:

1. importar: import {useTranslations} from 'next-intl';

2. crear constante dentro del cuerpo del componente: const t = useTranslations('Index'); // accede a

nuestro en.json 'Index' y es.json 'Index'

3. traemos el 'title' de nuestro .json y lo mostramos de esta forma <h1>{t('title')}

El archivo 'page.tsx' debe quedar de esta forma:

```
import {useTranslations} from 'next-intl';

export default function Home() {

  const t = useTranslations('Index');

  return (
    <div className="min-h-screen flex justify-center items-center text-2xl">
      <h1>{t('title')}
```

#### 14) REALIZAR CAMBIOS:

Si deseas modificar los nombres de los objetos y propiedades de los archivos .json debes de hacerlo de esta forma:

Por ejemplo, cambiamos el nombre del objeto 'Index' por 'Home' y la propiedad 'title' por 'text'.

1. Archivo en.json:

```
{
  "Home": {
    "text": "Hello world!"
  }
}
```

2. Archivo es.json:

```
{
  "Home": {
    "text": "Hola mundo!"
  }
}
```

3. Archivo pages.tsx:

```
import {useTranslations} from 'next-intl';

export default function Home() {

  const t = useTranslations('Home');

  return (
```

```

<div className="min-h-screen flex justify-center items-center text-2xl">
  <h1>{t('text')}</h1>
</div>
);
}

```

## 15) AGRAGAR PAGE NOT-FOUND

1. crear archivo not-found.tsx dentro de /src/app/[locale]
2. crear el siguiente código dentro del archivo not-found.tsx:  
(<https://nextjs.org/docs/app/api-reference/file-conventions/not-found>)

```

import Link from 'next/link'

export default function NotFound() {
  return (
    <div className="min-h-screen flex gap-4 justify-center items-center text-xl">
      <h2>Page not found:</h2>
      <Link href="/" className='underline'>Return Home</Link>
    </div>
  )
}

```

## 16) CREAR COMPONENTE 'Header':

1. crear la carpeta /components dentro de la carpeta /src
2. dentro de la carpeta /components crear el archivo Header.tsx
3. dentro del archivo Header.tsx crear el código:

```

const Header = () => {
  return (
    <header>
      Texto del header
    </header>
  );
}

```

export default Header

4. tarer el componente 'Header' a nuestra page 'Home':

```

import Header from '@components/Header';
import {useTranslations} from 'next-intl';

```

```

export default function Home() {

  const t = useTranslations('Home');

  return (
    <div className="min-h-screen flex flex-col justify-center items-center text-2xl">

```

```

    <Header />
    <h1>{t('text')}

```

Pero el 'Texto del header' no se traduce.

## 17) TRADUCIR 'TEXTO DEL HEADER' (COMPONENTE):

### 1. Archivo en.json:

```

{
  "Home": {
    "text": "Hello world!"
  },
  "Header": {
    "text": "Header in English"
  }
}

```

### 2. Archivo es.json:

```

{
  "Home": {
    "text": "Hola mundo!"
  },
  "Header": {
    "text": "Header en Español"
  }
}

```

### 3. Archivo header.tsx:

```

"use client"

import {useTranslations} from 'next-intl';

const Header = () => {

  const t = useTranslations('Header');

  return (
    <header>
      <h1>{t('text')}

```

## IMPORTANTE:

Si hacemos este proceso igual que hicimos con 'Home', al ser un componente que esta del lado del cliente nos dará este error:

Error: Failed to call `useTranslations` because the context from `NextIntlClientProvider` was not found.

This can happen because:

- 1) You intended to render this component as a Server Component, the render failed, and therefore React attempted to render the component on the client instead. If this is the case, check the console for server errors.
- 2) You intended to render this component on the client side, but no context was found.

Learn more about this error here:

<https://next-intl-docs.vercel.app/docs/environments/server-client-components#missing-context>

Es porque 'useTranslations' solo se puede utilizar del lado del servidor.

Para solucionarlo debemos hacerlo de esta otra forma:

1. nos dirigimos a (<https://next-intl-docs.vercel.app/docs/environments/server-client-components>) y copiamos lo siguiente:

```
import pick from 'lodash/pick';
import {NextIntlClientProvider, useMessages} from 'next-intl';
```

2. agregamos las importaciones al archivo donde vayamos a utilizar el componente, en este caso en la page 'Home'.

3. debemos instalar lodash y los @types:

```
npm i lodash
npm i --save-dev @types/lodash
```

4. rodeamos nuestro componente con: NextIntlClientProvider y le pasamos los mensajes: `const messages = useMessages();`

La page 'Home' debe quedar de esta forma:

```
import pick from 'lodash/pick';
import {useTranslations, NextIntlClientProvider, useMessages} from 'next-intl';
import Header from '@components/header';

export default function Home() {

  const t = useTranslations('Home');
  const messages = useMessages();

  return (
    <div className="min-h-screen flex flex-col justify-center items-center text-2xl">
      <NextIntlClientProvider messages={pick(messages, 'Header')}>
        <Header />
      </NextIntlClientProvider>
      <h1>{t('text')}</h1>
    </div>
  );
}
```

```
}
```

Para pasar todos los mensajes, sustituir: `messages={pick(messages, 'Header')}` por `messages={messages}` y comententar `// import pick from 'lodash/pick';`

----- FIN -----