Curso de

Internacionalización con Next.js

@jonalvarezz


Internacionalización
y localización

En este curso

- Arquitectura de i18n con soporte de 2 locales.
- Detectar locale del usuario y permitir cambiarla.
- Rutas y contenido internacionalizado.

“

i18n y L10n

Numerónimos

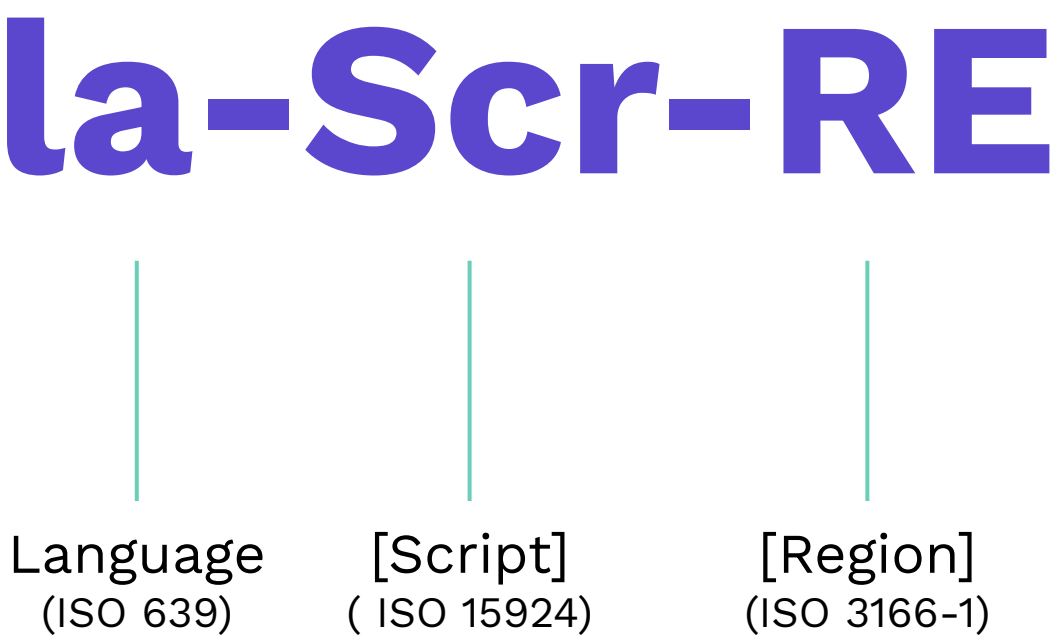
”

“

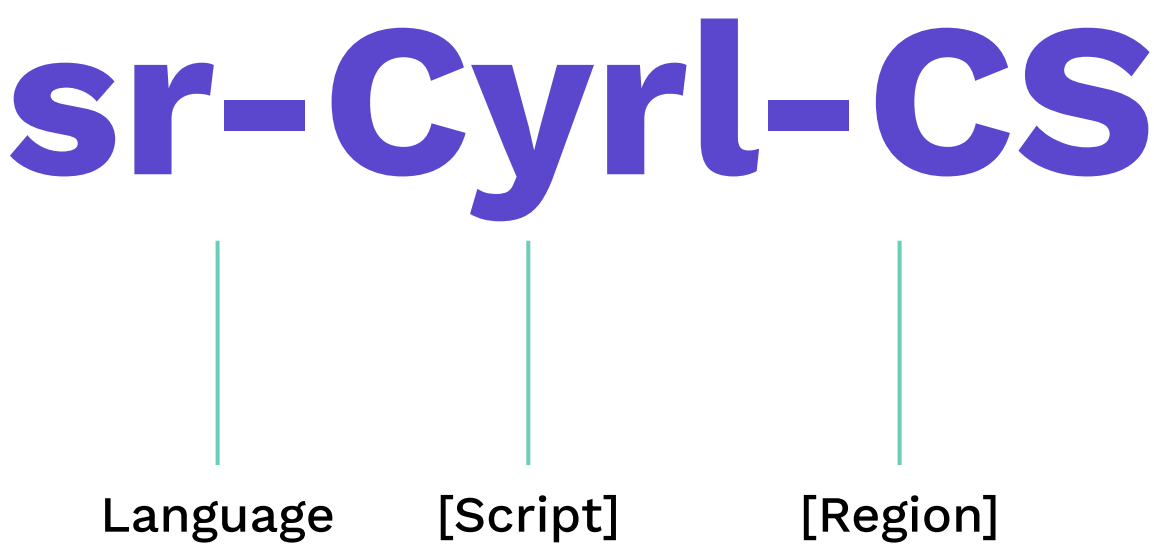
Un locale es un identificador para describir un lenguaje y región.

”

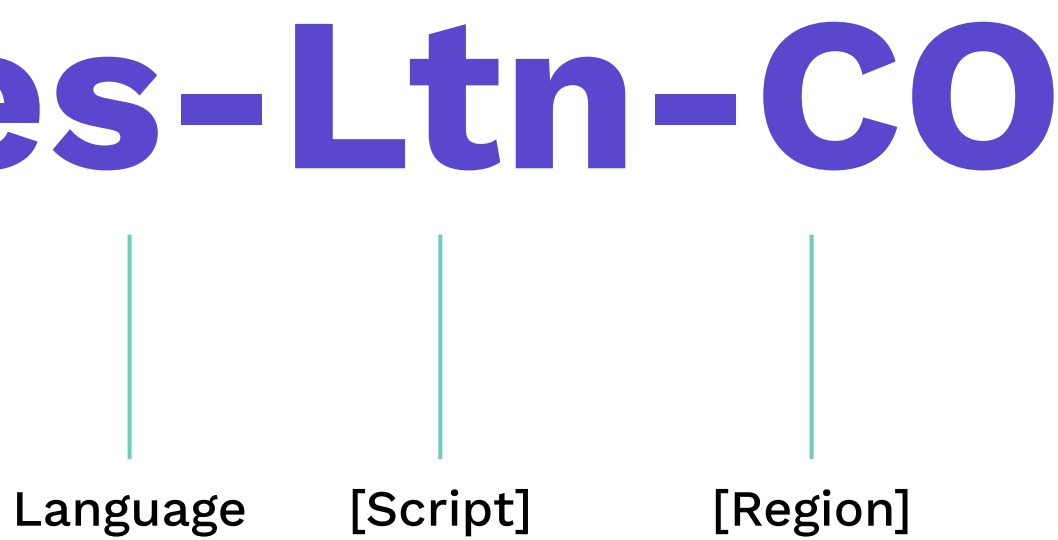
Anatomía de un Locale



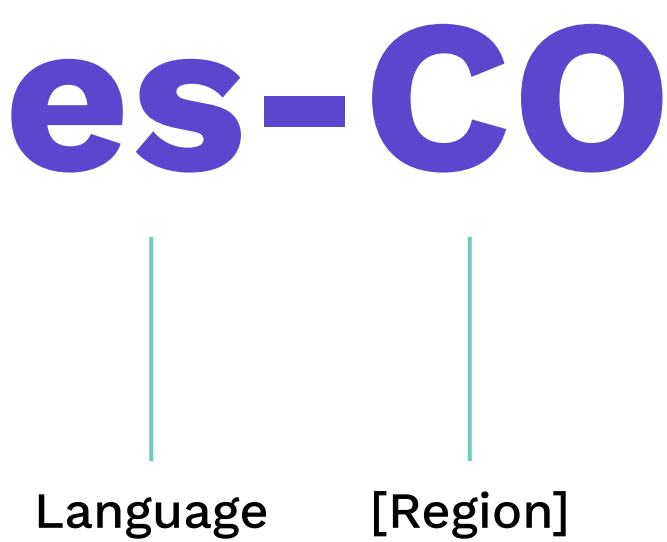
Anatomía de un Locale



Anatomía de un Locale



Anatomía de un Locale



Internacionalización

Locale

Internacionalización

Locale	Locale	Locale
Locale	Locale	Locale

Internacionalización

Locale	Locale	Locale
Locale	Locale	Locale

“

Internacionalización va más allá de traducir un sitio.

”

“

Internacionalización no se limita al texto.

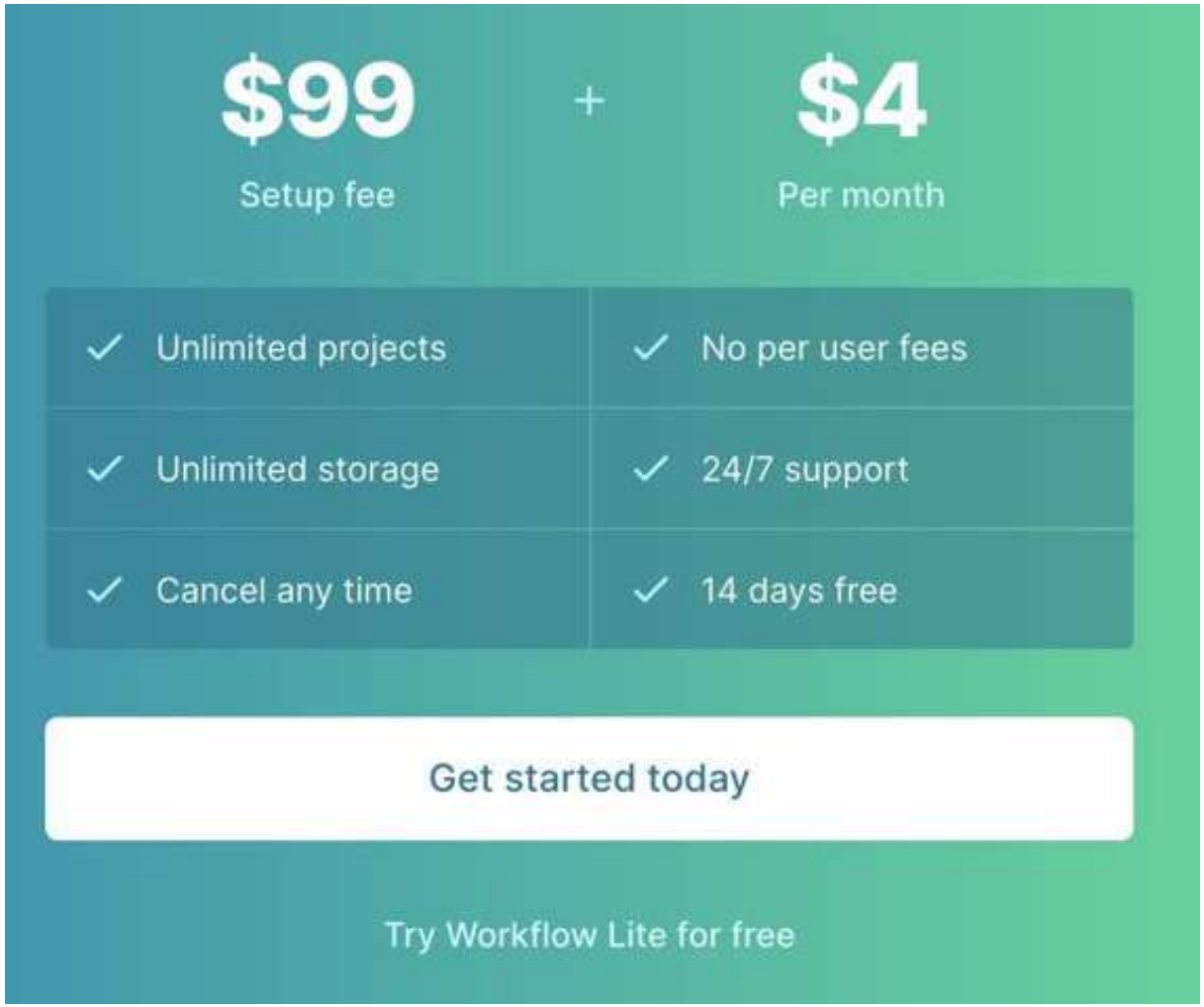
”

i18n

- Formato de dinero. **e.j.: $, €, £**
- Formato de fecha. **e.j.: 01.12.2021, 12.01.2021**
- Formato de número.
- Zona horaria (timezone).

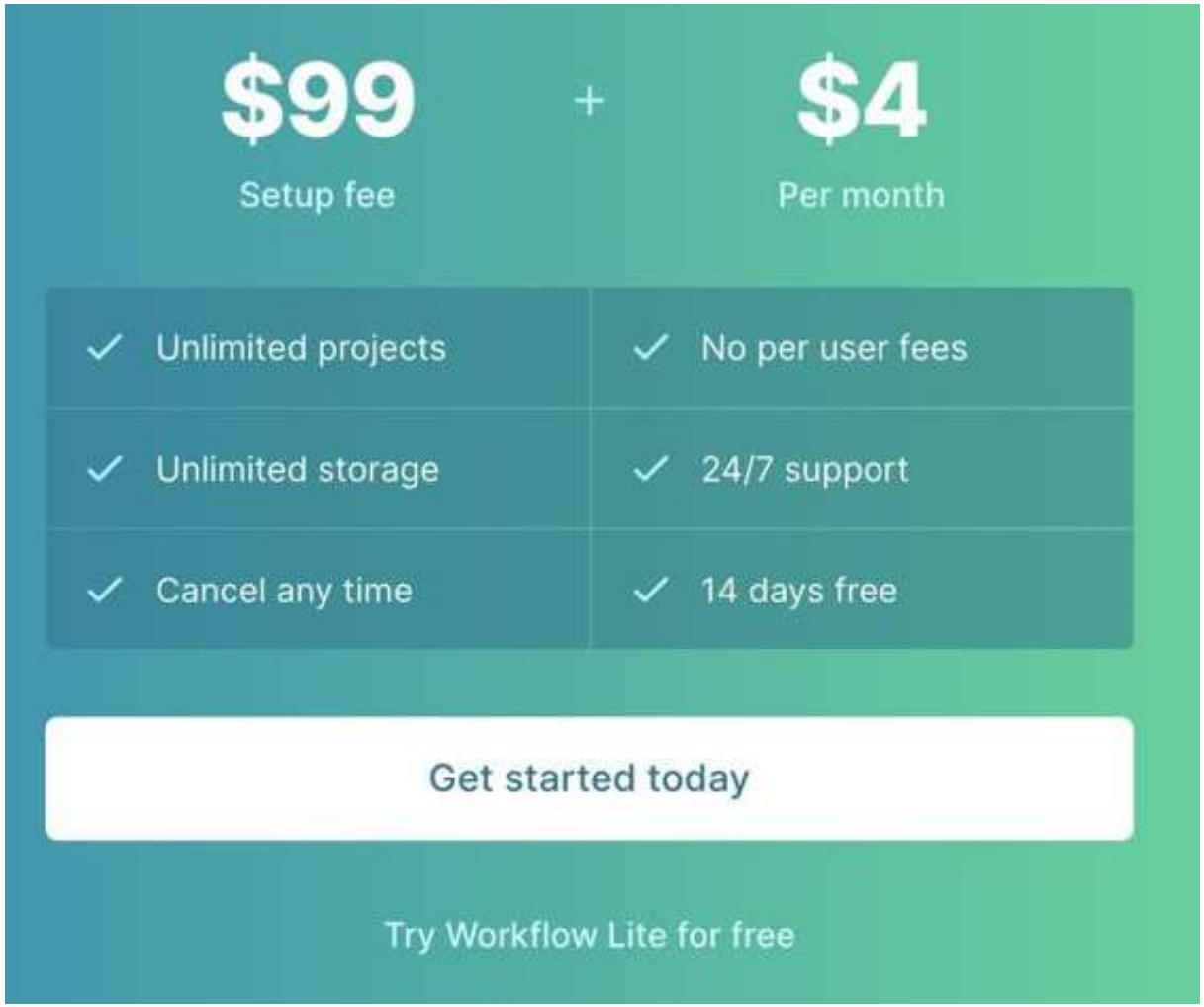
Arquitectura

Arquitectura de Internacionalización para Apps de JavaScript parte 1


$99
Setup fee
+
$4
Per month
Unlimited projects
No per user fees
Unlimited storage
24/7 support
Cancel any time
14 days free
Get started today
Try Workflow Lite for free

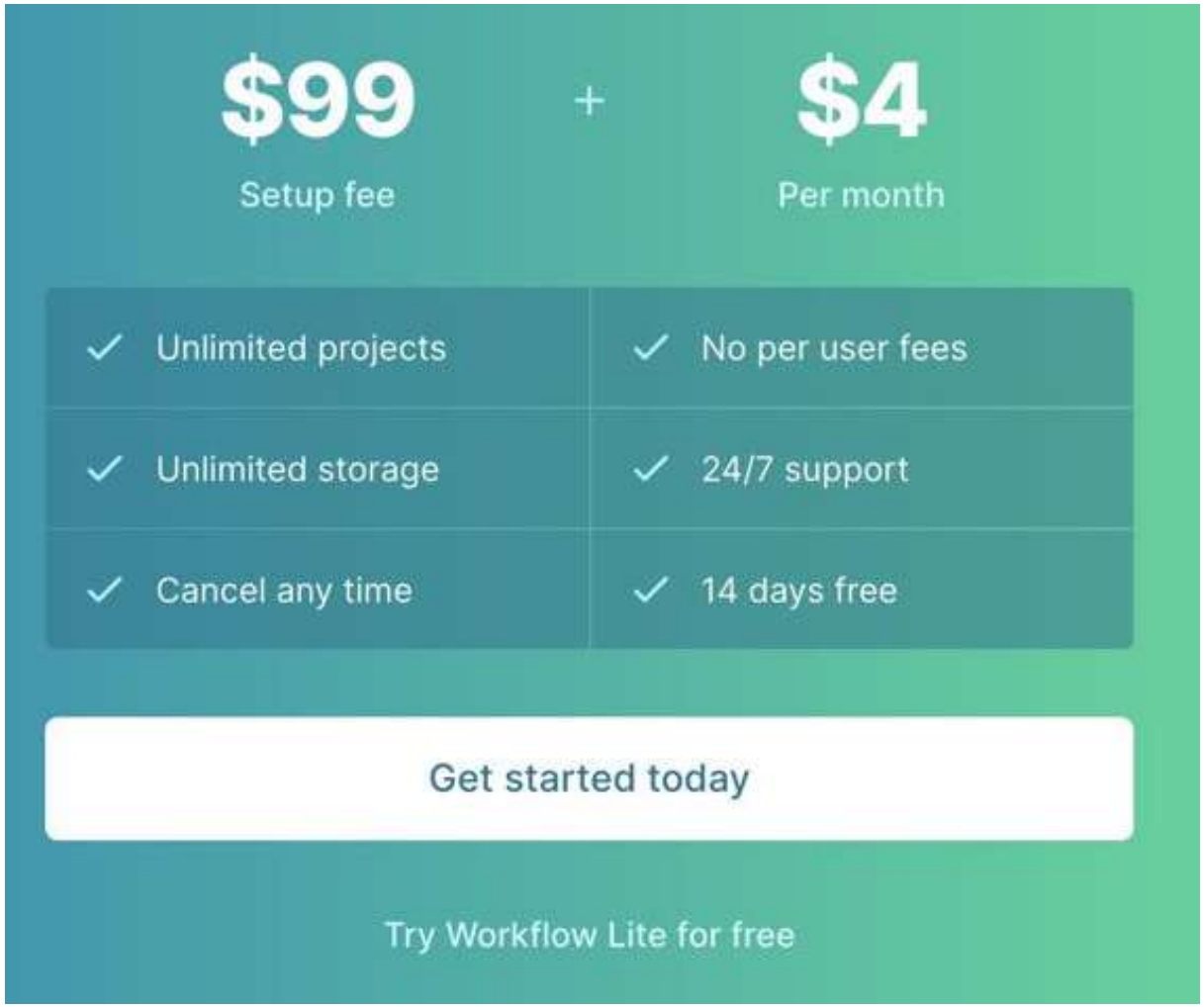
1

2


$99
Setup fee
+
$4
Per month
Unlimited projects
No per user fees
Unlimited storage
24/7 support
Cancel any time
14 days free
Get started today
Try Workflow Lite for free

App

Texto estático
hardcodeado

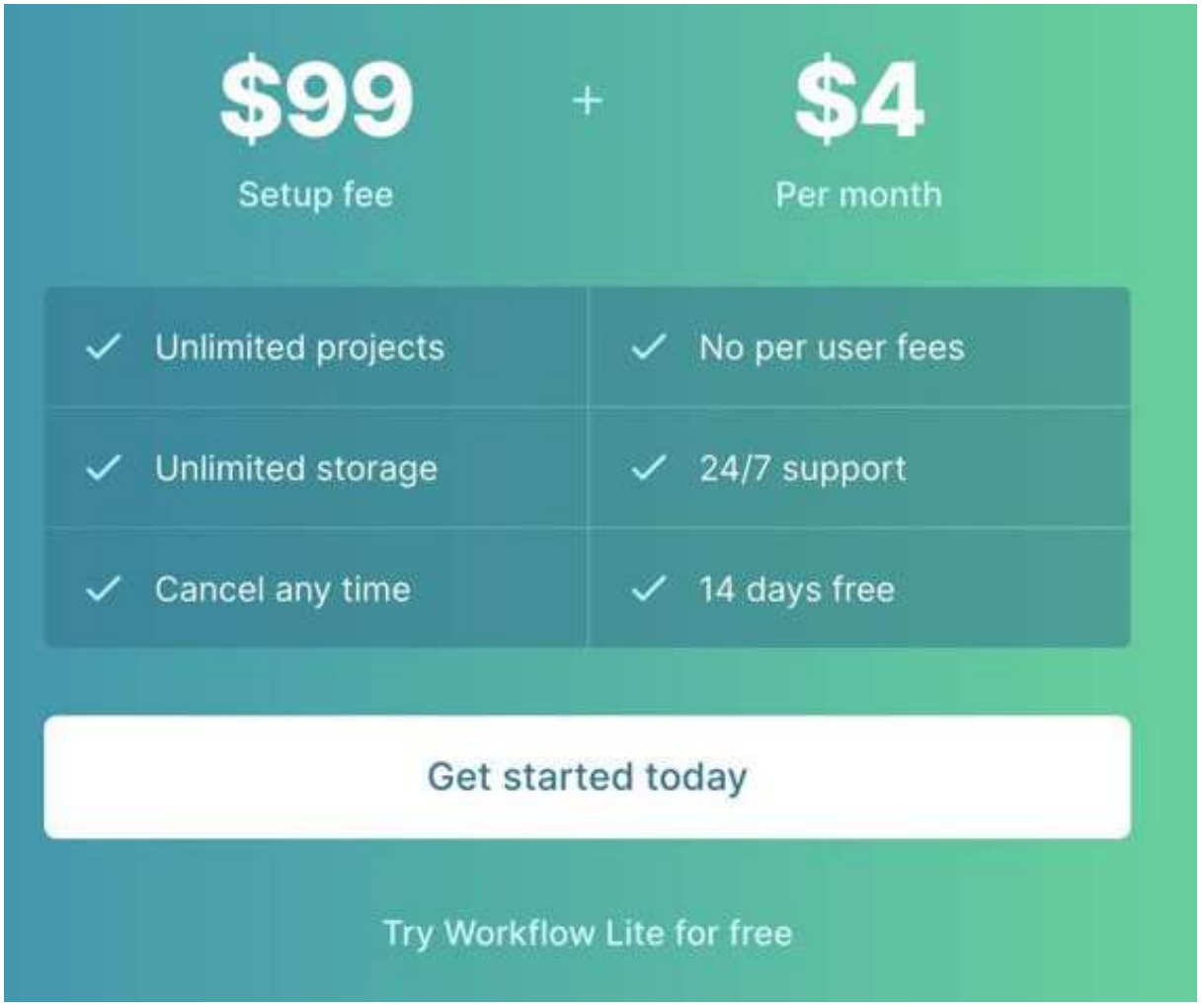


API

Información dinámica que obtenemos

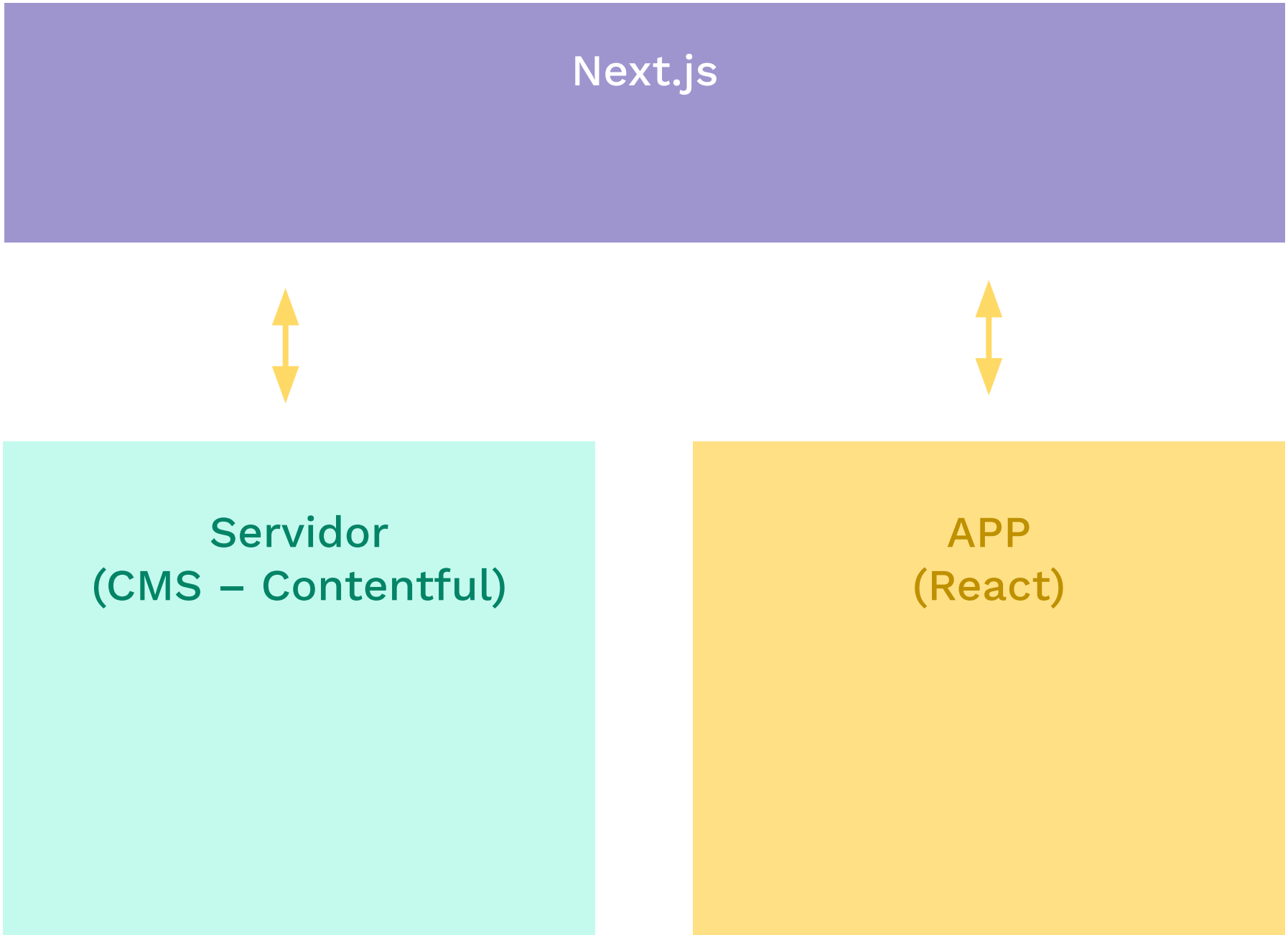
App

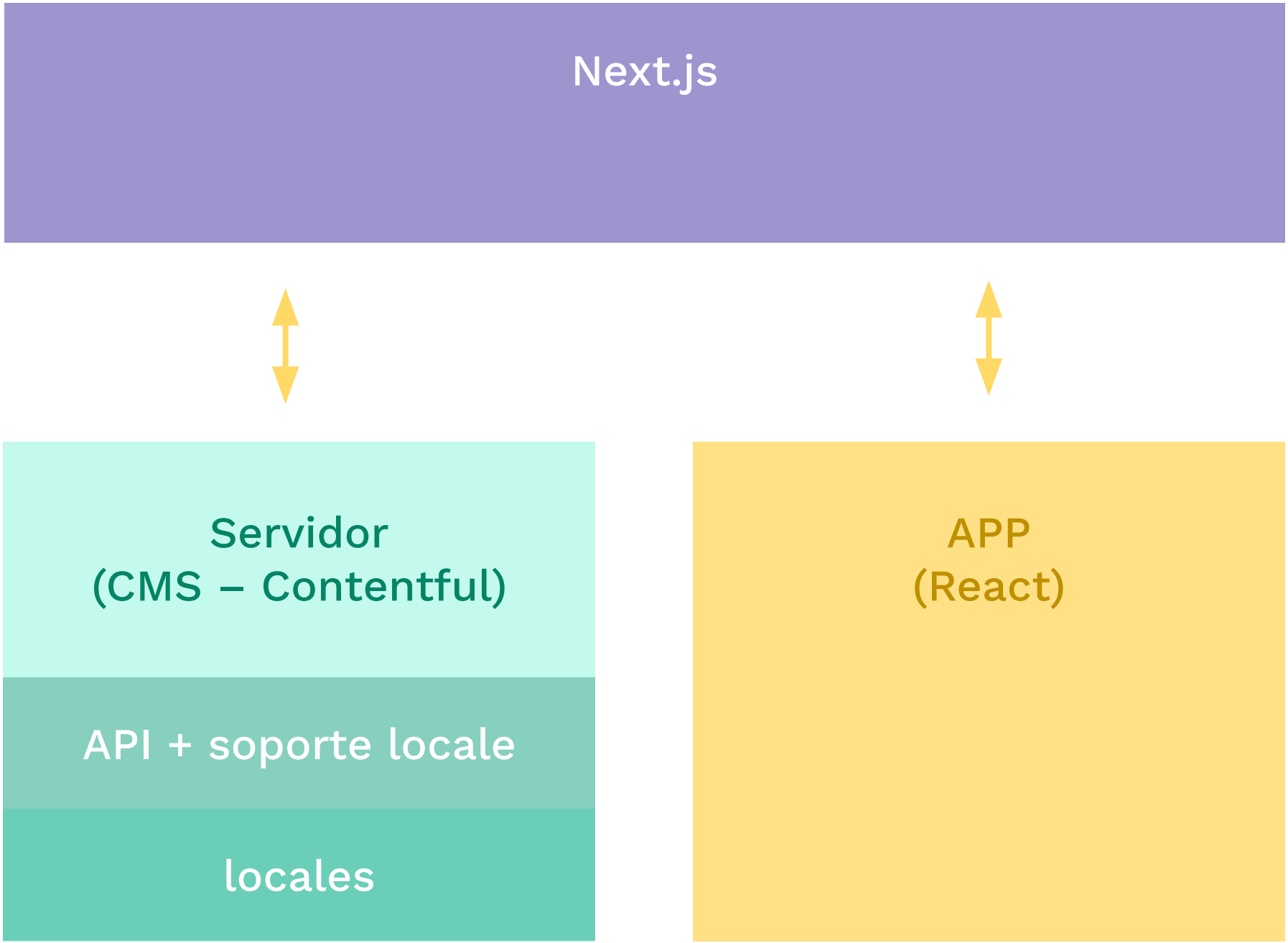
Texto estático *hardcodeado*

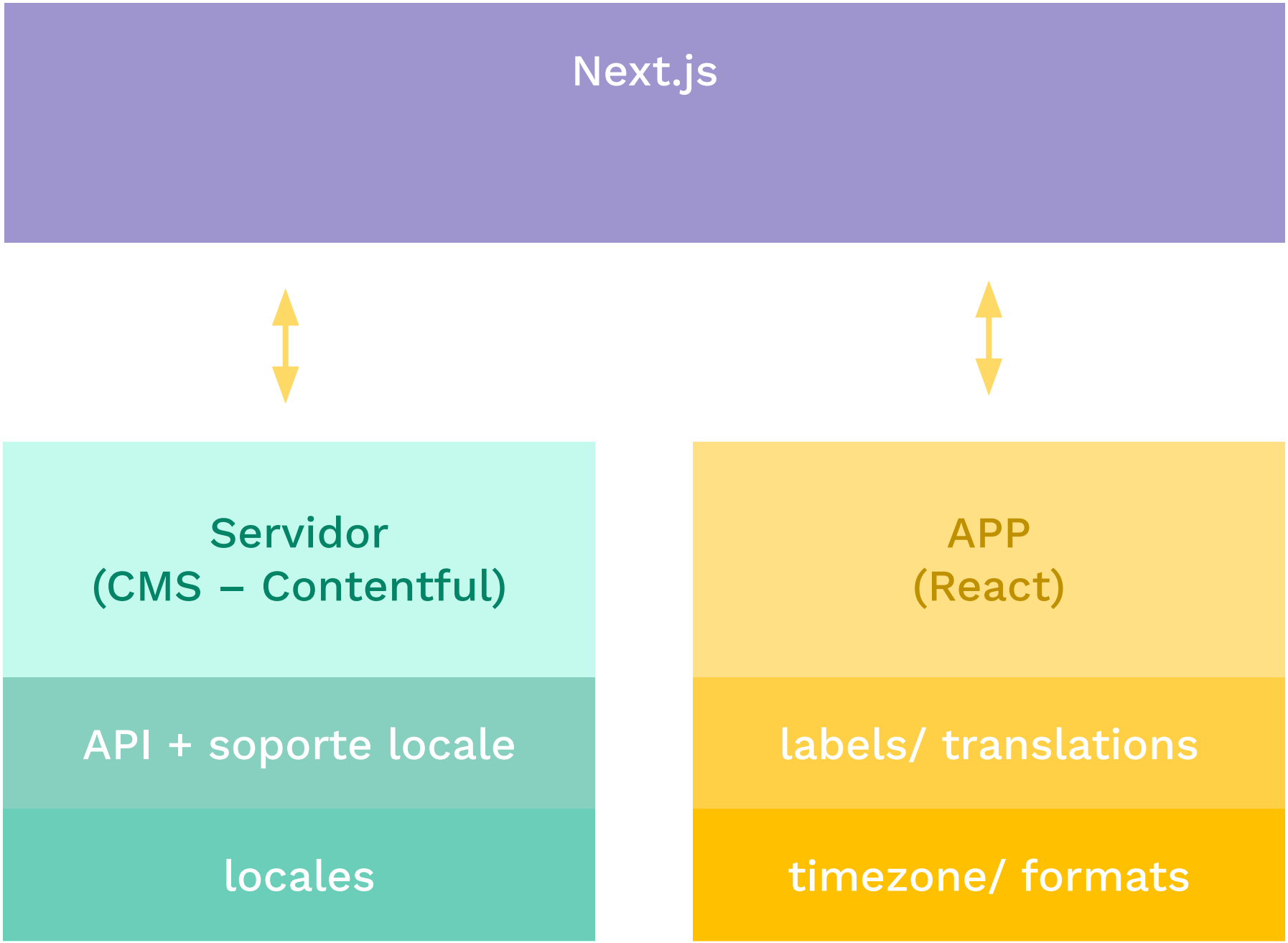


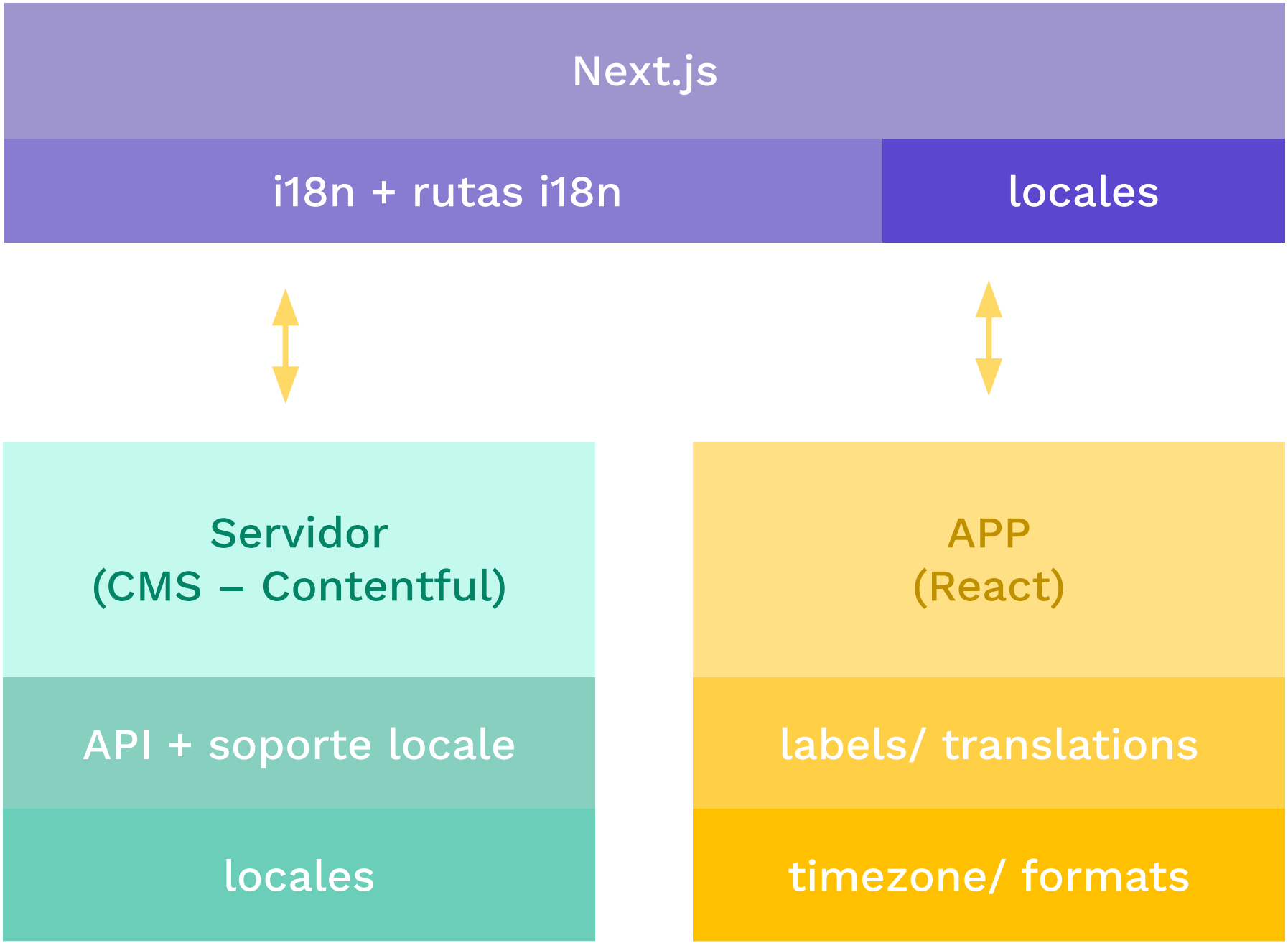
misitio.com/productos

misitio.com/es/productos


Next.js
Servidor
(CMS – Contentful)
APP
(React)


Next.js
Servidor
(CMS – Contentful)
API + soporte locale
locales
APP
(React)


Next.js
Servidor
(CMS – Contentful)
API + soporte locale
locales
APP
(React)
labels/ translations
timezone/ formats


Next.js
i18n + rutas i18n
locales
Servidor
(CMS – Contentful)
API + soporte locale
locales
APP
(React)
labels/ translations
timezone/ formats

Agregando locales a nuestro contenido y Rutas i18n

“El soporte de locales para el contenido depende del CMS y su API.”

Rutas i18n: estrategias

1. sub-path routing

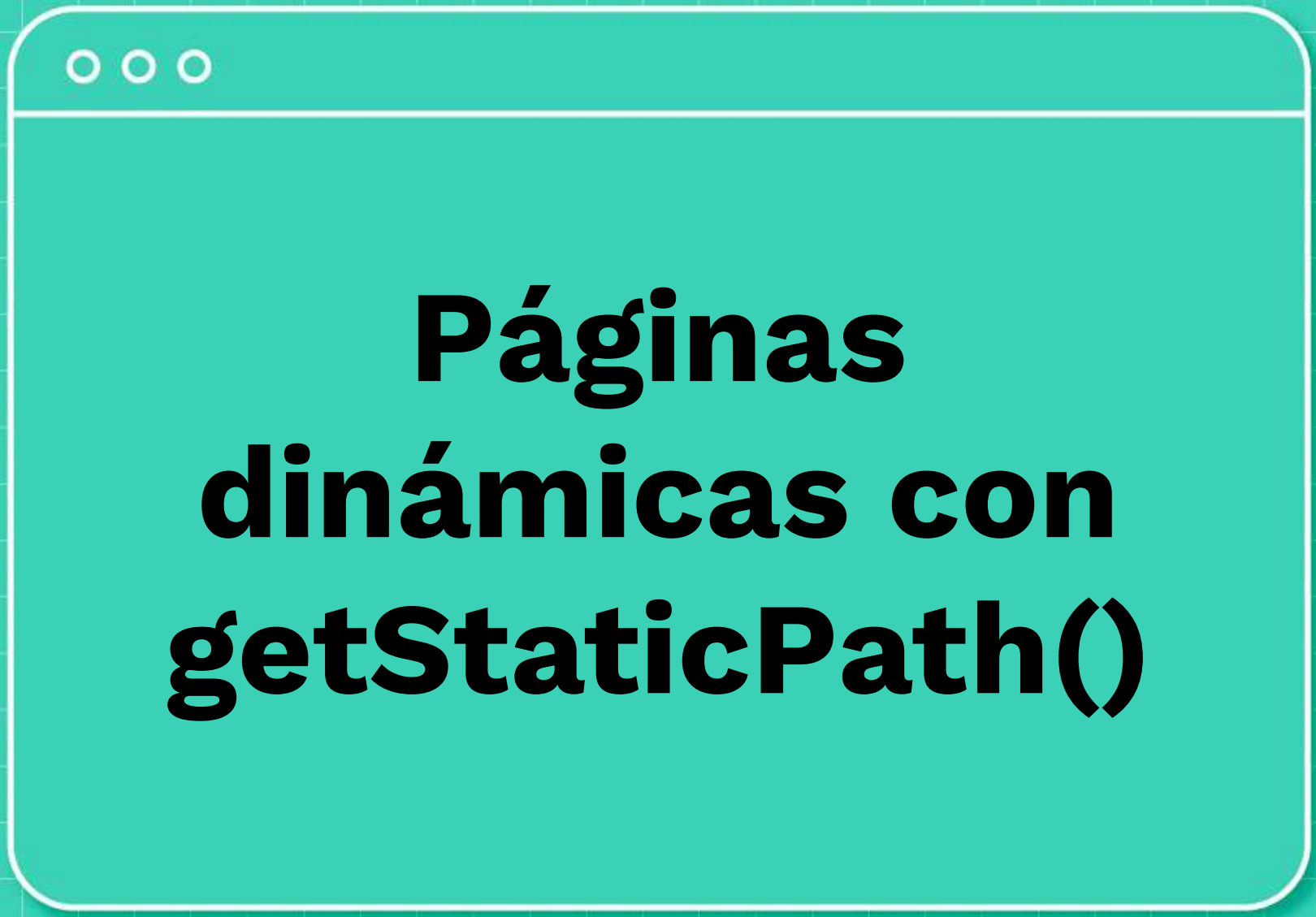
   misitio.com/es/productos

2. Domain routing

   misitio.es/productos


Rutas i18n

Páginas no-dinámicas (Home)


Páginas
dinámicas con
getStaticPath()

Experiencia de usuario

Detección de idioma automática y next/link

“

Next.js detecta el idioma del usuario de forma automática*

”

```
// next.config.js
module.exports = {
  i18n: {
    localeDetection: false,
  },
}
```

i18n y next/link

- next/link navegará al locale actualmente activo

```
router.push(
  url: string,
  as?: string,
  { locale: 'br' }
)
```

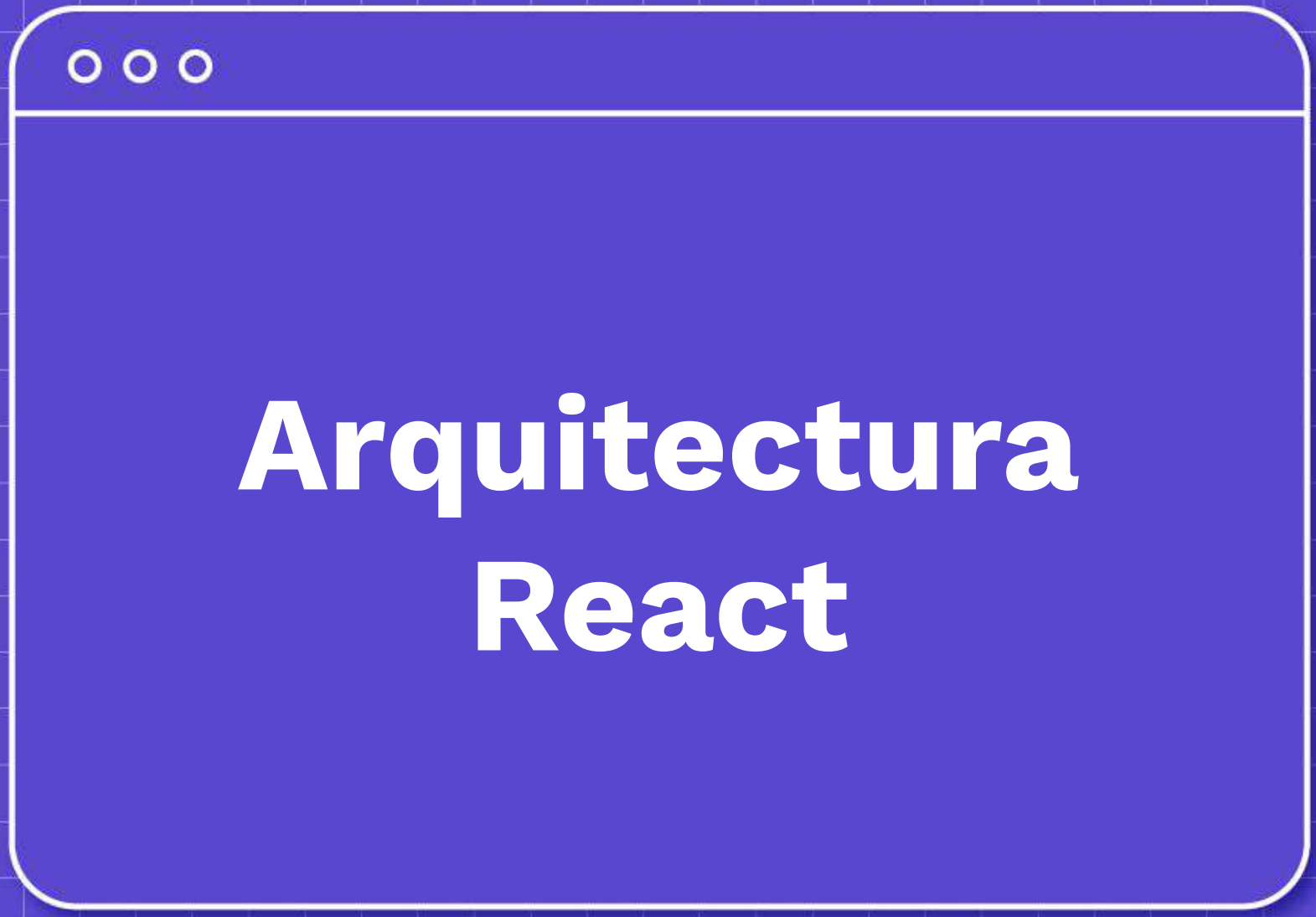
```
<Link
  href="/br/about-us"
  locale={ false }
/>
```

i18n Cookie

- NEXT_LOCALE

Permite controlar el locale a mostrar de un usuario.

Elección de idioma para el usuario a través de API Next.js


Arquitectura
React

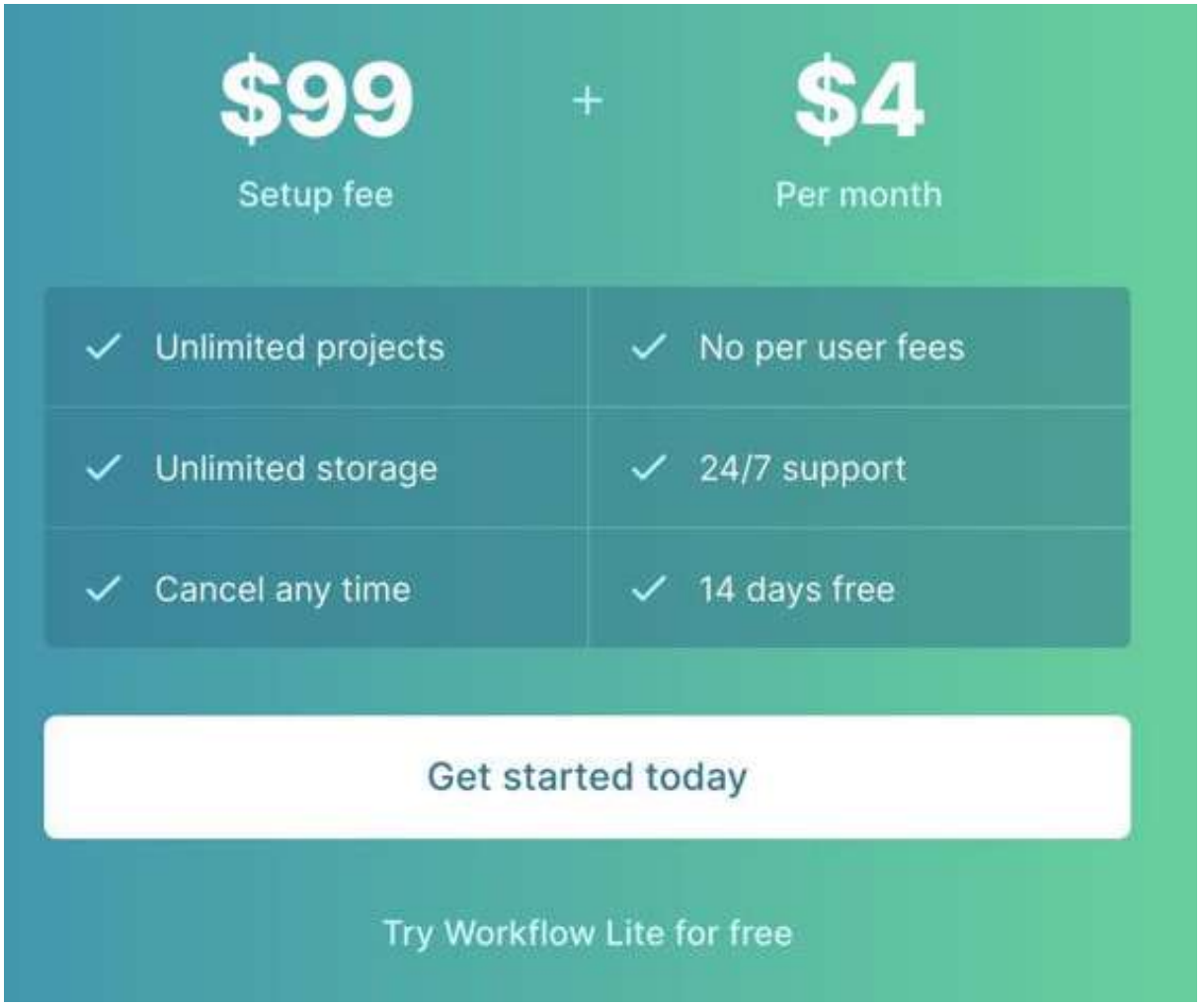

Arquitectura
Internacionalización
de labels en Apps
JavaScript

API

Información dinámica que obtenemos

App

Texto estático *hardcodeado*



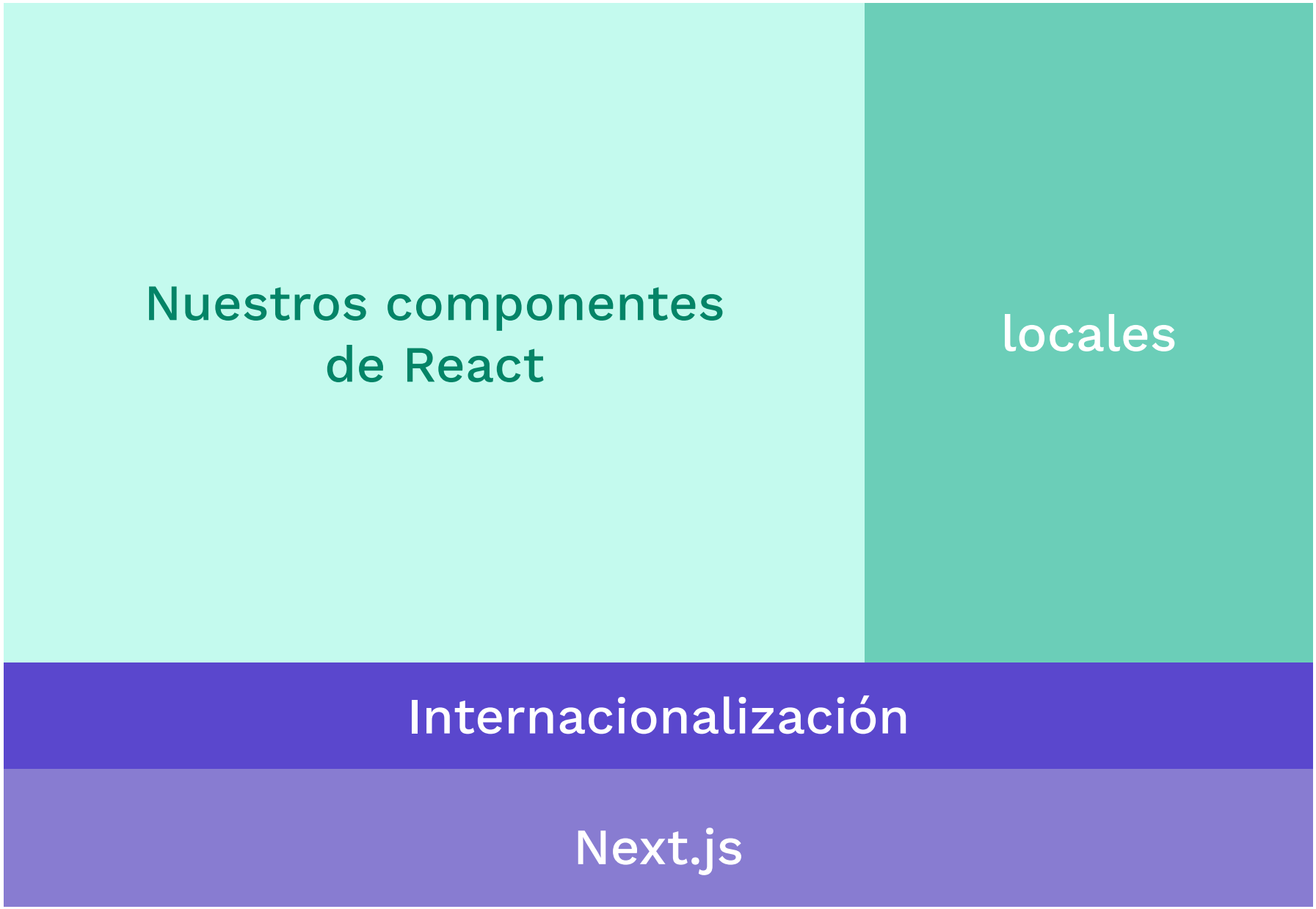
Estrategia

1. Todo texto se extrae del código y se mueve hacia un archivo JSON.
2. locales/es.json
3. Se crean los archivos JSON adicionales por locale. e.j.: locales/en-US.json
4. Se carga el archivo correcto según el locale.

Nuestros componentes
de React

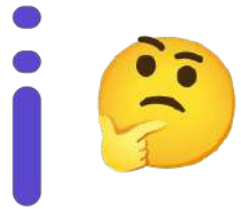
Internacionalización

Next.js


Nuestros componentes de React
locales
Internacionalización
Next.js

```
function MyComponent () {

  return (
    <Button>Comprar</Button>
  )

}
```

```
import labels from “locales/es.js”


function MyComponent () {

  return (
    <Button>{labels.comprar}</Button>
  )

}
```



1. ¿Cómo cargar el archivo locale con los labels correcto? ¿En el cliente o en el servidor?

2. ¿Cómo cargarlo impactando el tamaño del bundle lo menos posible?

3. ¿Una vez que lo cargue, cómo conectarlo con cada uno de los componentes?

¡🤔

1. ¿Una vez que lo cargue, cómo conectarlo con cada uno de los componentes?

2. ¿Cómo facilito el trabajo a nuestros editores no-desarrolladores?

Arquitectura Internacionalización en Componentes React

“

Analicemos la forma simple

”

```
// allLabels.json

{
   “es”: {
      “buy”: “Comprar”,
      “accept”: “Aceptar”,
      ...
   },

   “en-US”: {
      “buy”: “Buy”,
      “accept”: “Accept”,
      ...
   },
   ...
}
```

```
import allLabels from “locales/labels.json”

function getServerSideProps () { ... }

function MyComponent ({ locale }) {
   const labels = allLabels[locale]

   return (
      <Button>{labels.comprar}</Button>
   )
}
```

Desventajas

1. Entre más locales, más pesado el archivo. ¿Dividimos también por locales?

2. Entre más labels, más pesado el archivo. ¿Dividimos por componentes?

3. La responsabilidad está repetida en cada componente. ¿La extraemos?

```
// allLabels.json

{
   “es”: {
      “buy”: “Comprar”,
      “accept”: “Aceptar”,
      ...
   },

   “en-US”: {
      “buy”: “Buy”,
      “accept”: “Accept”,
      ...
   },
   ...
}
```

```
// locales/es.json

{
   “buy”: “Comprar”,
   “accept”: “Aceptar”,
   ...
}
```

```
// locales/es.json

{
   “buy”: “Comprar”,
   “accept”: “Aceptar”,
   ...
}

// locales/en-US.json

{
   “buy”: “Buy”,
   “accept”: “Accept”,
   ...
}
```

```
// locales/es/checkout.json

{
   “buy”: “Comprar”,
   “addToCart”: “Agregar al carro”,
}

// locales/es/about.json

{
   “about”: “Nosotros”,
   “phone”: “Teléfono”,
   ...
}
```

```
// locales/es/checkout.json

{
  “buy”: “Comprar”,
  “addToCart”: “Agregar al carro”,
  “contactUs”: “¿Interesado/a?”,
}

// locales/es/about.json

{
  “about”: “Nosotros”,
  “phone”: “Teléfono”,
  “contactUs”: “Conócenos”,
  ...
}
```

¡ Desventajas

1. Entre más labels, más pesado el archivo. ¿Dividimos por componentes?

2. Entre más locales, más pesado el archivo. ¿Dividimos también por locales?

3. La responsabilidad está repetida en cada componente. ¿La extraemos?

React

1. **React.Context:** para abstraer lógica en un solo lugar y permitir su acceso a componentes que la necesiten sin *prop drilling*.

2. **React Hooks:** para encapsular y re-usar lógica.

```
import useTranslations from “useTranslations”

function getServerSideProps () { ... }

function MyComponent () {
   const labels = useTranslations()

   return (
      <Button>{labels(‘comprar’)}</Button>
   )
}
```

```
function useTranslations () {
   const ctx = useContext(LocaleContext)
   const locale = ctx.getCurrentLocale()
   const labels = ctx.labels.get(locale)

   return labels
}
```

Librerías populares

- react-intl
- react-i18next
- lingui
- rosetta
- next-intl

Nuestra App completamente traducida

Librería elegida

- react-i18next

Retos y Trade-offs

Trade-offs

- Peso e impacto en performance vs. experiencia de usuario de una app internacionalizada.
- Peso e impacto en performance vs. características necesarias de la app e.j.: time zones, money formatting.

Retos

- Mitigar el impacto en performance.
- Aplicaciones que no son server o static rendered.
- Mantenibilidad: e.j.: chequeos en CI para asegurar que todos los labels estén extraídos o extraer automáticamente.

Conclusiones

Conclusiones

- Next.js nos proporciona los fundamentos en nuestra aplicación para internacionalizar una app.
- Nos brinda la flexibilidad para utilizar la API y los diferentes modos de rendering y complementar su solución.

Conclusiones

Las estrategias vistas las podemos replicar en otras apps de React y JavaScript.

Próximos pasos