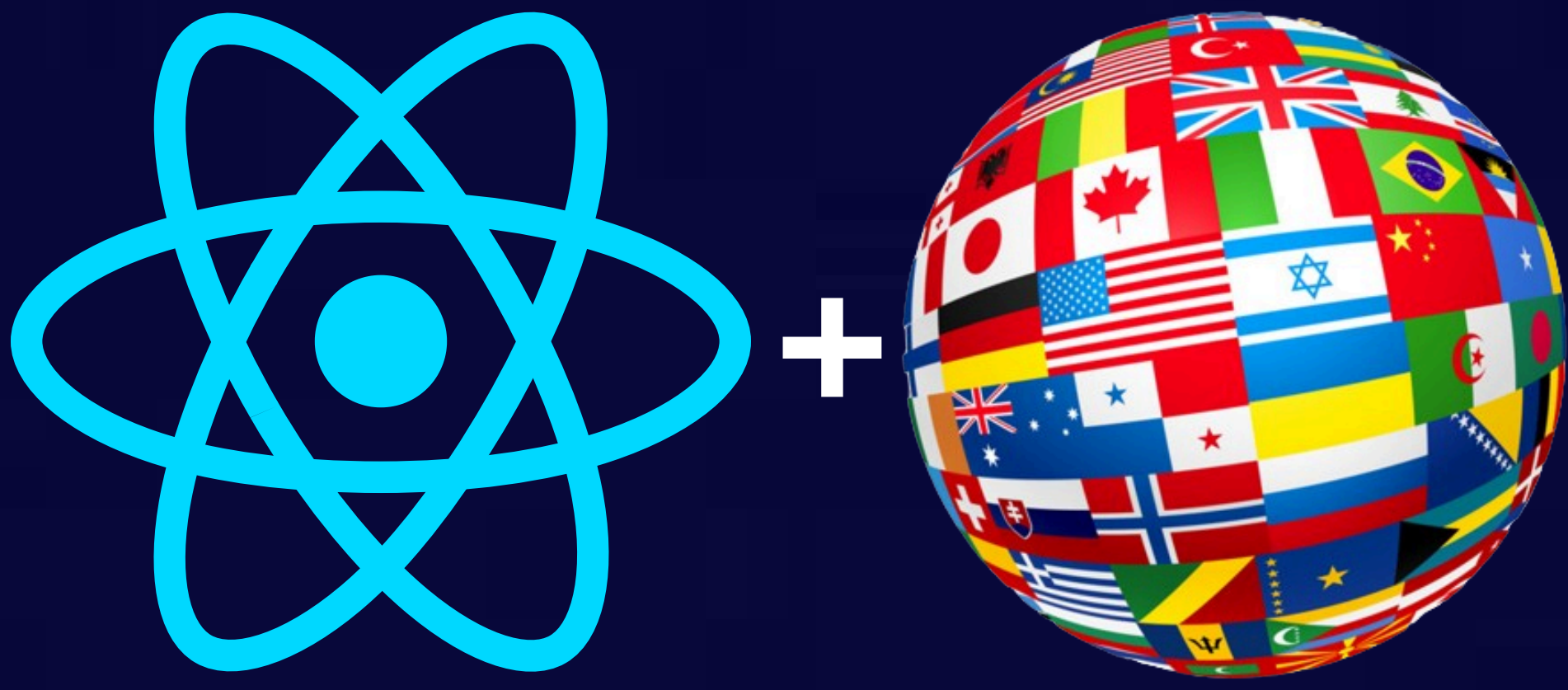
multilanguage



Internacionalização da sua aplicação React.js com testes e docs

Isaac Gomes

multilanguage

entendendo o multilanguage

uma situação comum ao trabalhar com aplicações voltadas a mais de um pais é o recurso multilanguage (suporte para múltiplos idiomas)

existe varias maneiras de construir esse recurso... para o exemplo farei todo esse controle no Front-end

Isaac Gomes

multilanguage

Ferramenta: i18next

para construir esse controle no Front faremos uso da lib i18next

Isaac Gomes

i18next - configuração

configuração base dos nosso idiomas

i18n.ts

```ts
import { availableLanguages } from "./availableLanguages"
import { RESOURCES } from "./resources"
import i18n from "i18next"
import { initReactI18next } from "react-i18next"
import LanguageDetector from "i18next-browser-languagedetector"

i18n
  .use(LanguageDetector)
  .use(initReactI18next)
  .init({
    resources: RESOURCES,
    fallbackLng: availableLanguages.es.code,
    interpolation: {
      escapeValue: false,
    },
  })

export default i18n
```

Isaac Gomes

i18next - configuração

aqui criamos um objeto para guardar os detalhes de cada idioma

i18n.ts

```ts
export const availableLanguages = {
  es: {
    code: "es",
    emoji: "ES Español",
    flagUrl: "img",
  },
  en: {
    code: "en",
    emoji: "US English",
    flagUrl: "img",
  },
  pt: {
    code: "pt",
    emoji: "BR Português",
    flagUrl: "img",
  },
} as const

export type LanguageCode = keyof typeof availableLanguages
```

Isaac Gomes

i18next - configuração

agora em nossos recursos vamos adicionar os idiomas possíveis... estou usando o objeto anteriormente criado para ser a chave

i18n.ts

```ts
import { availableLanguages } from "../availableLanguages"
import { EN_LOCATION } from "../location/EN_LOCATION"
import { ES_LOCATION } from "../location/ES_LOCATION"
import { PT_LOCATION } from "../location/PT_LOCATION"

export const RESOURCES = {
  [availableLanguages.en.code]: {
    translation: EN_LOCATION,
  },
  [availableLanguages.pt.code]: {
    translation: PT_LOCATION,
  },
  [availableLanguages.es.code]: {
    translation: ES_LOCATION,
  },
}
```

Isaac Gomes

i18next - configuração

agora em nossos recursos vamos adicionar os idiomas possíveis... estou usando o objeto anteriormente criado para ser a chave

i18n.ts

```ts
import { availableLanguages } from "../availableLanguages"
import { EN_LOCATION } from "../location/EN_LOCATION"
import { ES_LOCATION } from "../location/ES_LOCATION"
import { PT_LOCATION } from "../location/PT_LOCATION"

export const RESOURCES = {
  [availableLanguages.en.code]: {
    translation: EN_LOCATION,
  },
  [availableLanguages.pt.code]: {
    translation: PT_LOCATION,
  },
  [availableLanguages.es.code]: {
    translation: ES_LOCATION,
  },
}
```

Isaac Gomes

i18next - configuração

agora montamos a base do idioma

i18n.ts

```ts
import { EN_CARD } from "./card"

export const EN_LOCATION = {
  card: EN_CARD,
}
```

i18n.ts

```ts
export const EN_CARD = {
  title: "Title in English",
  content: "Content in English",
  buttonLabel: "Change Language",
  author: "Author",
  date: "Date",
}
```

o caso aqui para test farei a construção de um Card

Isaac Gomes

TS

i18next - configuração

criaremos uma função que recebe as opções de idiomas existente no objeto e faz essa mudança de idioma

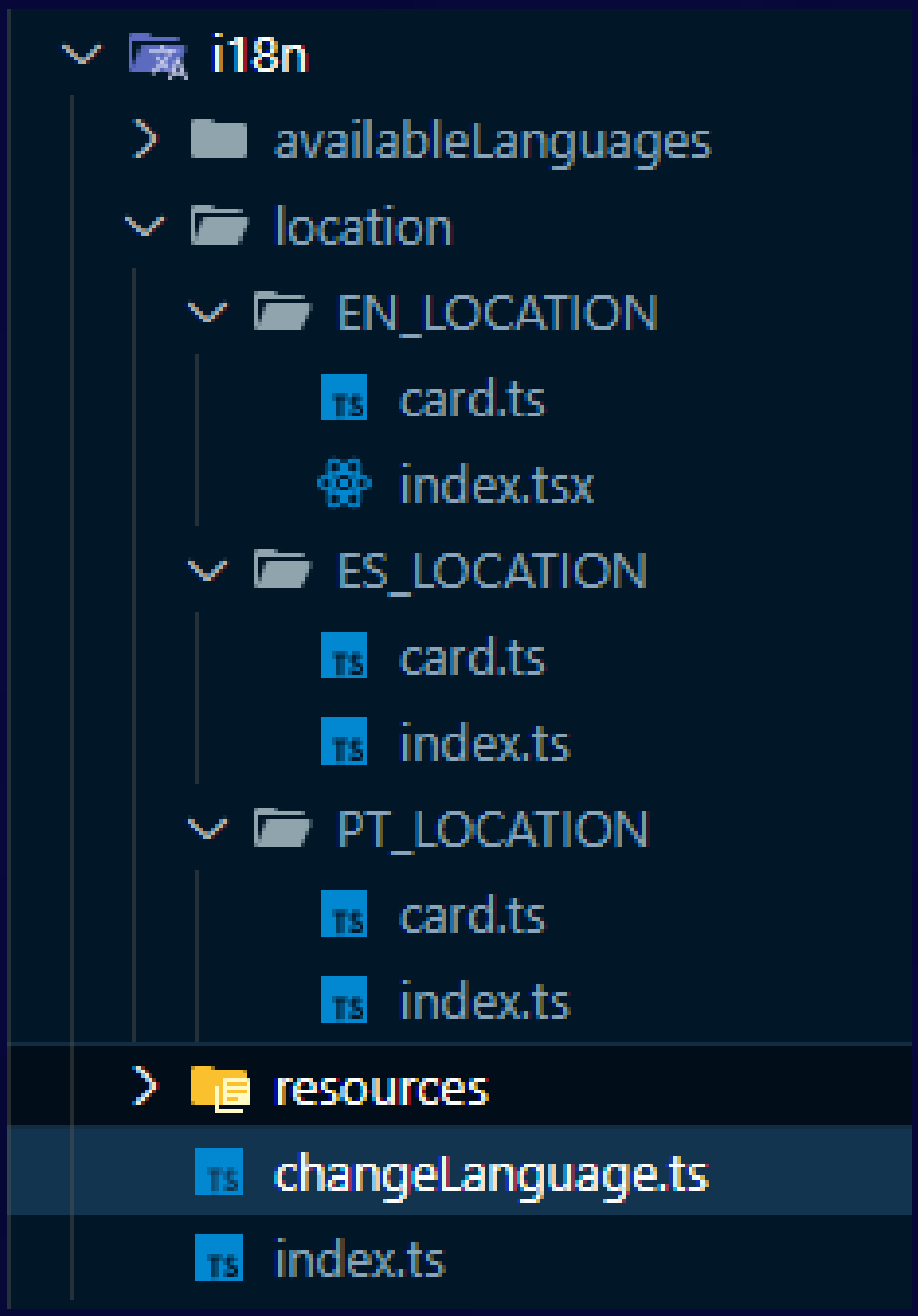
i18n.ts

```ts
import i18n from "i18next"
import { LanguageCode } from "./availableLanguages"

export const changeLanguage = (language: LanguageCode) => {
  i18n.changeLanguage(language)
}
```

Isaac Gomes

i18next - configuração

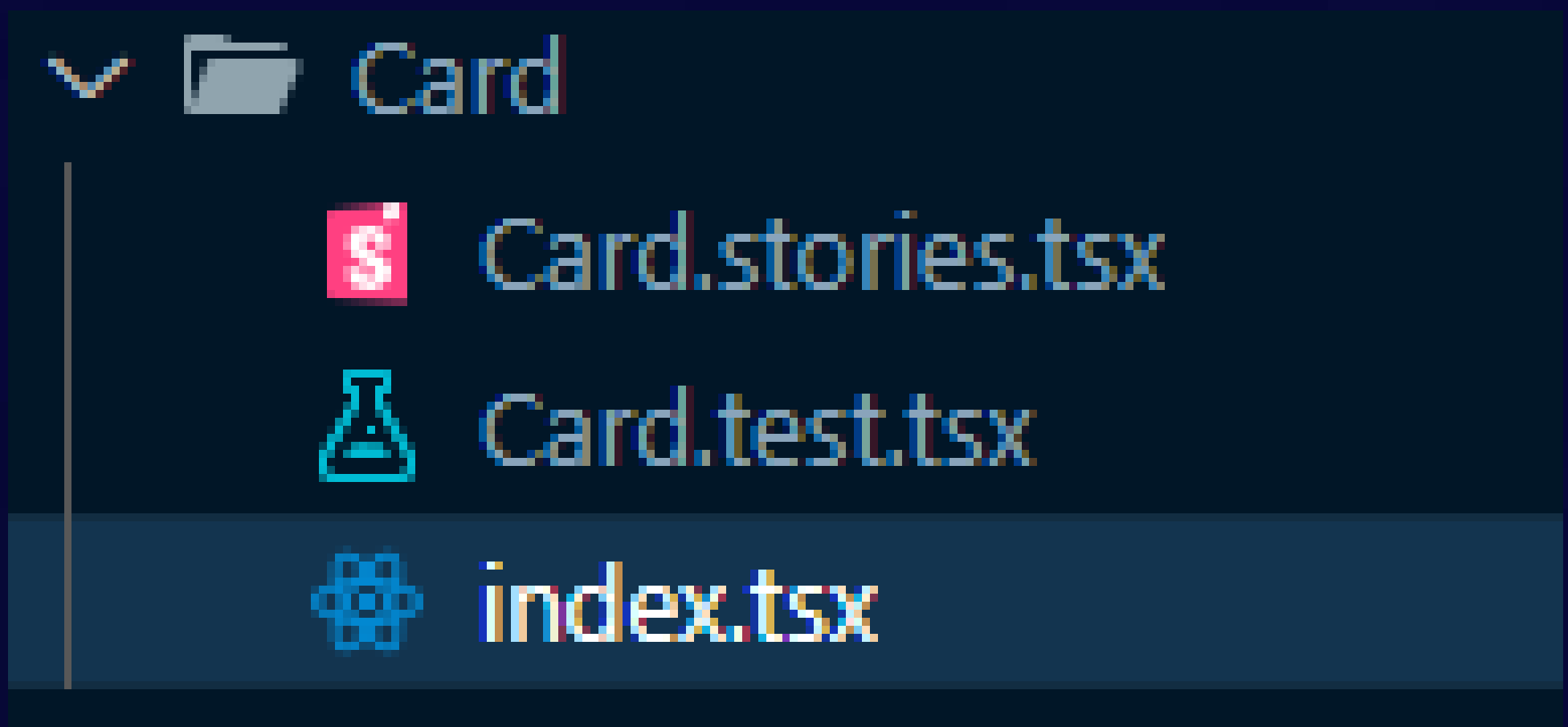
assim, a estrutura do nosso
multilanguage fica dessa maneira



Isaac Gomes

i18next

vamos gerar um componente via CLI e iniciar o consumo do i18



Isaac Gomes

i18next

agora vamos escrever o componente

Card.tsx

```tsx
export const Card = () => {
  const { t } = useTranslation()
  return (
    <div>
      <div>
        <div>
          {t("card.title")}
        </div>
        <p>
          {t("card.content")}
        </p>
      </div>
      <div className="px-6 py-4">
        <span>
          {t("card.author")}
        </span>
        <span>
          {t("card.date")}
        </span>
      </div>
      <div>
        {Object.entries(availableLanguages).map(([code, language]) => (
          <button
            key={code}
            onClick={() => changeLanguage(language.code)}
          >
            <img src={language.flagUrl} alt={language.code} />
            <p>{language.emoji}</p>
          </button>
        ))}
      </div>
    </div>
  )
}
```

Isaac Gomes

TS

i18next

agora vamos escrever o Story

Card.tsx

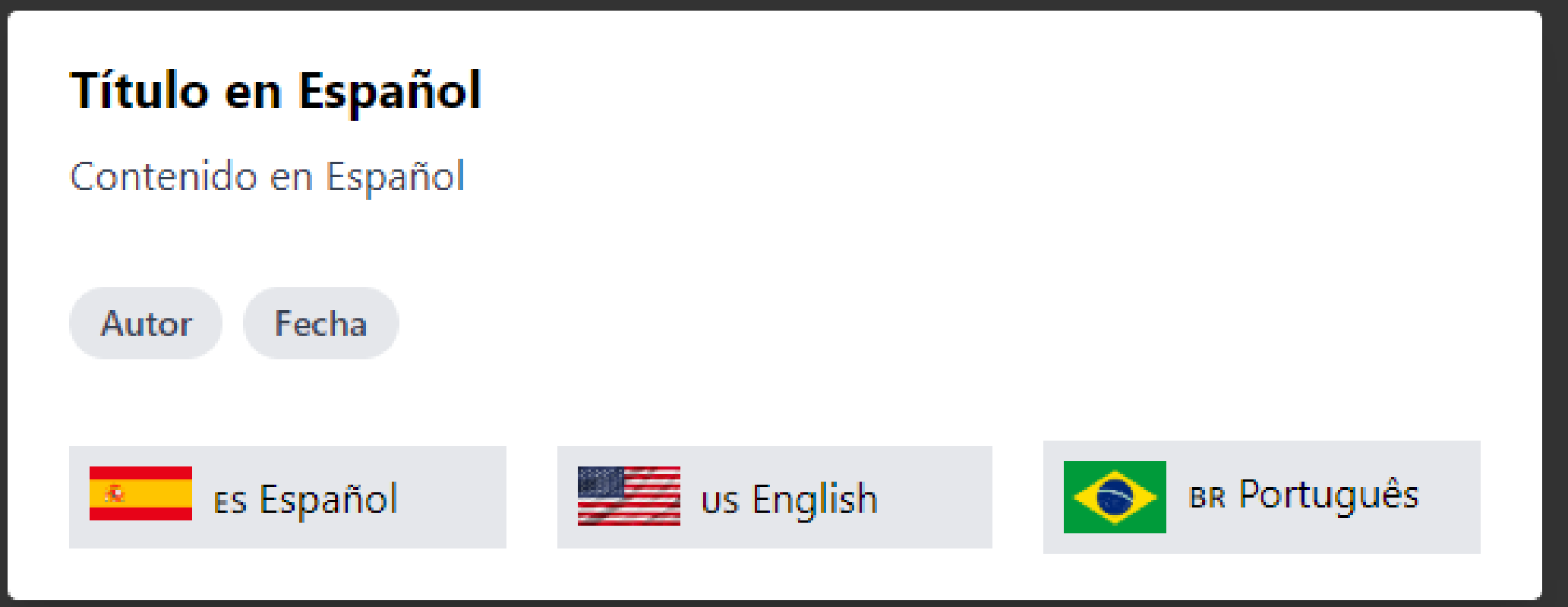
```tsx
import type { Meta, StoryObj } from "@storybook/react"
import { Card } from "."

type Story = StoryObj<typeof Card>

const meta: Meta<typeof Card> = {
  component: Card,
  parameters: {
    layout: "centered",
  },
}
export default meta

export const CardLocation: Story = {
  args: {},
}
```

Isaac Gomes

TS

i18next

resultado no storybook



Isaac Gomes

i18next

agora vamos aos testes para garantir o comportamento

Card.tsx

```tsx
export const testCardWithLanguageContent = ({
  language,
  getByTestId,
}: Props) => {
  const title = getByTestId("title")
  expect(title.textContent).toEqual(language.title)

  const content = getByTestId("content")
  expect(content.textContent).toEqual(language.content)

  const author = getByTestId("author")
  expect(author.textContent).toEqual(language.author)

  const date = getByTestId("date")
  expect(date.textContent).toEqual(language.date)
}
```

primeiro passo vai ser criar um helper para verificar os textos do card

Isaac Gomes

TS

i18next

segundo passo vai ser criar um helper para clicar no button

Card.tsx

```tsx
export const simulateButtonClick = ({ getByTestId, code }: Props) => {
  const button = getByTestId(code)
  fireEvent.click(button)
}
```

Isaac Gomes

TS

i18next

agora é só montar os testes usando os helper e os objetos de cada language

Card.tsx

```tsx
describe("<Card />", () => {
  it("should render card with English content", () => {
    testCardWithLanguageContent({ language: EN_CARD, getByTestId })
  })

  it("should render card with Spanish content", () => {
    simulateButtonClick({
      getByTestId,
      code: availableLanguages.es.code,
    })
    testCardWithLanguageContent({ language: ES_CARD, getByTestId })
  })

  it("should render card with Portuguese content", () => {
    simulateButtonClick({
      getByTestId,
      code: availableLanguages.pt.code,
    })
    testCardWithLanguageContent({ language: PT_CARD, getByTestId })
  })
})
```

Isaac Gomes

TS

i18next

com isso garantimos que....

1 - o texto default é inglês

2 - que ao realizar o click os textos vão mudar para a língua selecionada

Isaac Gomes

i18next

cobertura de testes

File	% Stmts	% Branch	% Funcs	% Lines
All files	100	100	100	100
Helper	100	100	100	100
simulateButtonClick.ts	100	100	100	100
testCardWithLanguageContent.ts	100	100	100	100
components/Card	100	100	100	100
index.tsx	100	100	100	100
i18n	100	100	100	100
index.ts	100	100	100	100
i18n/availableLanguages	100	100	100	100
index.ts	100	100	100	100
i18n/location/EN_LOCATION	100	100	100	100
card.ts	100	100	100	100
index.tsx	100	100	100	100
i18n/location/ES_LOCATION	100	100	100	100
card.ts	100	100	100	100
index.ts	100	100	100	100
i18n/location/PT_LOCATION	100	100	100	100
card.ts	100	100	100	100
index.ts	100	100	100	100
i18n/resources	100	100	100	100
index.ts	100	100	100	100

Isaac Gomes

i18next

importante

esse é somente um dos jeitos
podemos criar esse recurso de varias maneiras exemplo podemos usar algo como BFF para ter textos dinâmicos e deixar no Front somente a língua selecionada... tudo depende da sua necessidade

Isaac Gomes

TS

Gostou?

Curta

Compartilhe

Salve

Isaac Gomes