Fase 5: Internacionalización

Internacionalización

Partimos de...

- Árbol de componentes *presentacionales*
- Todo el contenido dinámico se carga desde JSON
- *Storybook* funcionando y *tests verdes*

Internacionalización

Queremos...

- Añadir soporte para *múltiples idiomas*
  - Inglés y castellano
- Añadir soporte para *múltiples monedas*

Internacionalización

react-intl

- Librería para i18n con React
  - Bastante completa
  - Soporte para diferentes casos de uso
- Textos, fechas y monedas

Internacionalización

3 pasos de preparación

1. Instalar *react-intl* con *npm*
2. Crear un fichero de inicialización
3. Envolver el árbol de componentes con *IntlProvider*

src/lib/setup-i18n.js

```
import { addLocaleData } from 'react-intl'
import esLocaleData from 'react-intl/locale-data/es'
import enLocaleData from 'react-intl/locale-data/en'
import esTransaltions from '../../config/translations/es.json'
import enTransaltions from '../../config/translations/en.json'

const locales = {
  es: { data: esLocaleData, translations: esTransaltions },
  en: { data: enLocaleData, translations: enTransaltions }
}

export default function setupI18n(locale) {
  const { data, translations } = locales[locale]
  addLocaleData(data)
  return translations
}
```

Internacionalización

Envolver el árbol de componentes con *IntlProvider*

- El concepto es *simple*:

  — Los componentes que utilicen i18n tienen que tener *IntlProvider* como ancestro

- Implica *adaptar* storybook y tests *por separado*

.storybook/config.js

```
import React from 'react'
import { configure, addDecorator  } from '@storybook/react'
import '../src/styles.css'
import { IntlProvider } from 'react-intl'
import setupI18n from '../src/lib/setup-i18n'

const locale = 'es'

addDecorator(story => (
  <IntlProvider messages={setupI18n(locale)} locale={locale}>
    {story()}
  </IntlProvider>
))

configure(() => {
  const req = require.context('../src/', true, /.stories.js$/)
  const files = req.keys()
  files.sort().forEach(filename => req(filename))
}, module)
```

tests/ui/views/ProductList.test.js

```
import { mount } from 'enzyme'
import { IntlProvider } from 'react-intl'
import setupI18n from '../../src/lib/setup-i18n'
/* resto de imports omitidos */

describe('<ProductList/>', () => {

  const wrapper = mount(
    <IntlProvider messages={setupI18n('en')} locale='en'>
      <ProductList data={data} shoppingCart={shoppingCart}/>
    </IntlProvider>
  )

  it('mounts correctly', () => {
    expect(wrapper.exists('.intro-title h1')).toBe(true)
  })

})
```

tests/ui/views/ProductList.test.js

```
import { intlMount } from '../../lib/intl-mount'
/* resto de imports omitidos */

describe('<ProductList/>', () => {

  const wrapper = intlMount(
    <ProductList data={data} shoppingCart={shoppingCart}/>
  )

  it('mounts correctly', () => {
    expect(wrapper.exists('.intro-title h1')).toBe(true)
  })

})
```

Internacionalización

Para *usar* la librería tenemos 2 opciones:

- Utilizar un *componente predefinido*
  - *FormattedMessage*
  - *FormattedNumber*
- Traducir el texto directamente
  - `intl.formatMessage({ id: '...' })`

```
import { FormattedMessage } from 'react-intl'
/* resto de imports omitidos */

const RelatedProducts = ({ products }) => (
  <div className="related-products">
    <BetaHeader>
      <FormattedMessage id="title:related-products"/>
    </BetaHeader>
    <ProductGrid products={products} />
  </div>
)
```

```
import { FormattedNumber } from 'react-intl'
/* resto de imports omitidos */

const Money = ({ price, quantity, total }) => (
  <span className={cx({
    'product-price': !total,
    'product-total': total
  })}>
    <FormattedNumber style="currency"
                     currency={price.currency}
                     value={(quantity * price.amount / 100)}
    />
  </span>
)
```

```
import { injectIntl } from 'react-intl'

const ProductList = ({ data, shoppingCart, intl }) => {
  const { results, totalPages, page } = data
  const title = intl.formatMessage({ id: 'title:products' })
  const subtitle = intl.formatMessage({ id: 'title:full-catalog' })
  return (
    <AppLayout shoppingCart={shoppingCart}>
      <IntroTitle title={title} subtitle={subtitle}/>
      <ProductGrid products={results} />
      <Pagination pages={totalPages} currentPage={page}/>
    </AppLayout>
  )
}

export default injectIntl(ProductList)
```

Ejercicio

Internacionaliza la aplicación

- Extrae todos los textos estáticos a los JSON de traducciones
- Modifica *Money* para que soporte diferentes monedas
- Adapta Storybook y los tests
  - crea la funcion *intlMount*