Fase 7: Servicios y Rutas

Servicios y Rutas

Partimos de...

- Árbol de componentes *completo*
- Las *vistas* reciben *datos* y *acciones* como *props*

Servicios y Rutas

Queremos...

- Programar la *lógica* de la aplicación
  - Gestión del *estado*
  - Comunicación con el *servidor*
- Capa de servicios *desacoplados* y *testeables*
- Montar las *rutas* y conectar las *vistas* con los *servicios*

Servicios y Rutas

Servicios

- Sin depedencias
  - ni *internas* (acoplamiento)
  - ni *externas* (infrastructura)
- *Reciben* todo lo que necesitan

Servicios y Rutas

Servicios

- Cada servicio gestiona su propio *estado independiente*
- Métodos para satisfacer dos *responsabilidades*:
  - *consultas*: acceder a partes del estado
  - *comandos*: modificar el estado

Servicios y Rutas

Servicios

- En la carpeta */src/services*
- Una *consulta* por cada *dato* que reciban las *vistas*
- Un *comando* por cada *callback* que reciban las *vistas*

```
const initState = { catalog: {}, details: {} }

export default (infra, state = initState, setState) => ({
  // queries
  getProducts: () => state.catalog,
  getProduct: () => state.details,

  // commands
  async fetchProducts(page) {
    setState({ ...state, loading: true })
    const catalog = await infra.xhr.get('/products', { params: { page } })
    setState({ ...state, catalog, loading: false})
  },
  async fetchProduct(productId) {
    setState({ ...state, loading: true })
    const details = await infra.xhr.get(`/products/${productId}`)
    setState({ ...state, details, loading: false})
  }
})
```

```
import { identity } from 'lodash'
import productsService from '../../src/services/products-service'

describe('Products Service', () => {
  describe('Queries', () => {
    test('getProducts', () => {
      const catalog = { test: true }
      const service = productsService({}, { catalog }, identity)
      expect(service.getProducts()).toBe(catalog)
    })
    test('getProduct', () => {
      const details = { test: true }
      const service = productsService({}, { details }, identity)
      expect(service.getProduct()).toBe(details)
    })
  })
})
```

```
describe('Commands', () => {
  let state = {}
  const setState = (newState) => { state = newState }
  let promise, xhr, service
  const testPayload = { test: true }

  beforeEach(() => {
    setState({})
    promise = makeDeferred()
    xhr = { get: jest.fn(() => promise) }
    service = productService({ xhr }, state, setState)
  })

  test('fetchProducts', async () => {
    const done = service.fetchProducts(1)
    expect(state).toEqual({ loading: true })
    expect(xhr.get).toHaveBeenCalledWith('/products', { params: { page: 1 } })
    promise.resolve(testPayload)
    await done
    expect(state).toEqual({ catalog: testPayload, loading: false })
  })
})
```

```
PASS tests/services/products-service.test.js
  Products Service
    Queries
      ✓ getProducts (3ms)
      ✓ getProduct
    Commands
      ✓ fetchProducts (4ms)
      ✓ fetchProducts (1ms)
      ✓ failed request (1ms)

-------------------------|----------|----------|----------|----------|-------------------|
File                     |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
-------------------------|----------|----------|----------|----------|-------------------|
All files                |      100 |      100 |      100 |      100 |                   |
 src/services            |      100 |      100 |      100 |      100 |                   |
  products-service.js    |      100 |      100 |      100 |      100 |                   |
 src/services/lib        |      100 |      100 |      100 |      100 |                   |
  update-with-loading.js |      100 |      100 |      100 |      100 |                   |
 tests                   |      100 |      100 |      100 |      100 |                   |
  setup.js               |      100 |      100 |      100 |      100 |                   |
 tests/lib               |      100 |      100 |      100 |      100 |                   |
  make-deferred.js       |      100 |      100 |      100 |      100 |                   |
-------------------------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        1.709s, estimated 2s
```

```
const initState = { id: null, items: [] }

export default ({ xhr }, state = initState, setState) => ({
  // queries
  getShoppingCart: () => state,

  // commands
  async fetchShoppingCart() {
    // if the cart exists: refresh it
    // else: create a new cart
  },
  async createShoppingCart() {
    // crate a new cart
  },
  async addProduct(id, quantity) {
    // add a product to the cart
  },
  async removeItem(id) {
    // optimistically update the state before the request
    // make the request and update the state
  },
  async modifyItemQuantity(id, quantity) {
    // optimistically update the state before the request
    // make the request and update the state
  },
  async checkout(data) {
    // make the request
  }
})
```

Servicios y Rutas

Endpoints de ShoppingCart:

POST /cart ⇒ Crea un nuevo carrito

GET /cart/:id ⇒ Recupera el carrito :id

POST /cart/:id/items { id, quantity } ⇒ Añade un item al carrito

DELETE /cart/:id/items/:itemId ⇒ Quita el item :itemId

PUT /cart/:id/items/:itemId { quantity } ⇒ Modificar el item :itemId

POST /cart/:id/checkout { ...formFields } ⇒ Confirmar el carrito

Ejercicio

Implementa los servicios de la aplicacion

- `products-service.js`
- `shopping-cart-service.js`
- `100% de coverage!`

Servicios y Rutas

Montar los servicios

- Inicializar la *infrastructura*
- Inicializar los servicios con *infra*, *state* y *setState*
- Utilizar *Context API* para *inyectar* los servicios

/src/config/index.js

```
const configs = {
  develop: {
    apiUrl: 'http://localhost:3001'
  }
}

const env = process.env.NODE_ENV
const activeConfig = (configs[env] || configs.develop)
export default activeConfig
```

/infrastructure/xhr.js

```
import axios from 'axios'

export default ({ apiUrl }) => ({
  async get(path, params) {
    const { data } = await axios.get(apiUrl.concat(path), params)
    return data
  },
  async post(path, body) {
    const { data } = await axios.post(apiUrl.concat(path), body)
    return data
  },
  async put(path, body) {
    const { data } = await axios.put(apiUrl.concat(path), body)
    return data
  },
  async delete(path) {
    const { data } = await axios.delete(apiUrl.concat(path))
    return data
  }
})
```

/app/lib/use-services.js

```
import { useState } from 'react'
import productsService from '../../services/products-service'
import shoppingCartService from '../../services/shopping-cart-service'

export default (infra) => ({
  productsService: productsService(infra, ...useState()),
  shoppingCartService: shoppingCartService(infra, ...useState())
})
```

/app/lib/StateContext.js

```
import { createContext } from 'react'

const StateContext = createContext()
export default StateContext
```

/src/index.js

```
/* resto de imports omitidos */
import StateContext from './app/lib/StateContext'
// infrastructure
import config from './config'
import xhr from './infrastructure/xhr.js'
// services
import useServices from './app/lib/use-services'

const infra = { xhr: xhr(config) }

const Wrapper = () => {
  const services = useServices(infra)
  return (
    <StateContext.Provider value={services}>
      <App />
    </StateContext.Provider>
  )
}

ReactDOM.render(<Wrapper />, document.getElementById('root'));
```

/src/app/ProductList.js

```
import React, { useEffect } from 'react'
import StateContext from './lib/StateContext.js'
import ProductList from '../ui/views/ProductList'

const Wrapper = ({ services }) => {
  const { productsService, shoppingCartService } = services
  const data = productsService.getProducts()
  const shoppingCart = shoppingCartService.getShoppingCart()
  const page = 1
  useEffect(() => productsService.fetchProducts(page), [page])
  return (
    <ProductList data={data} shoppingCart={shoppingCart}/>
  )
}

export default () => (
  <StateContext.Consumer>
    {services => <Wrapper services={services}/>}
  </StateContext.Consumer>
)
```

/src/App.js

```
import React from 'react'
import { BrowserRouter as Router, Route, Redirect, Switch } from 'react-router-dom'
// i18n
import { IntlProvider } from 'react-intl'
import setupI18n from './lib/setup-i18n'
// components
import ProductList from './app/ProductList'

const locale = 'es-ES'

const App = () => (
  <IntlProvider messages={setupI18n(locale)} locale={locale}>
    <Router>
      <Switch>
        <Route path="/catalog/:page?" component={ProductList} />
      </Switch>
    </Router>
  </IntlProvider>
)

export default App
```

Ejercicio

Conecta las 4 vistas con los servicios

- Utiliza **StateContext** para acceder a los servicios
- Invoca a las **queries** de los servicios para obtener **data**
- Pasa los **comandos** de los servicios como **callbacks**
- Utiliza **efectos** para **cargar** la data desde el **servidor**