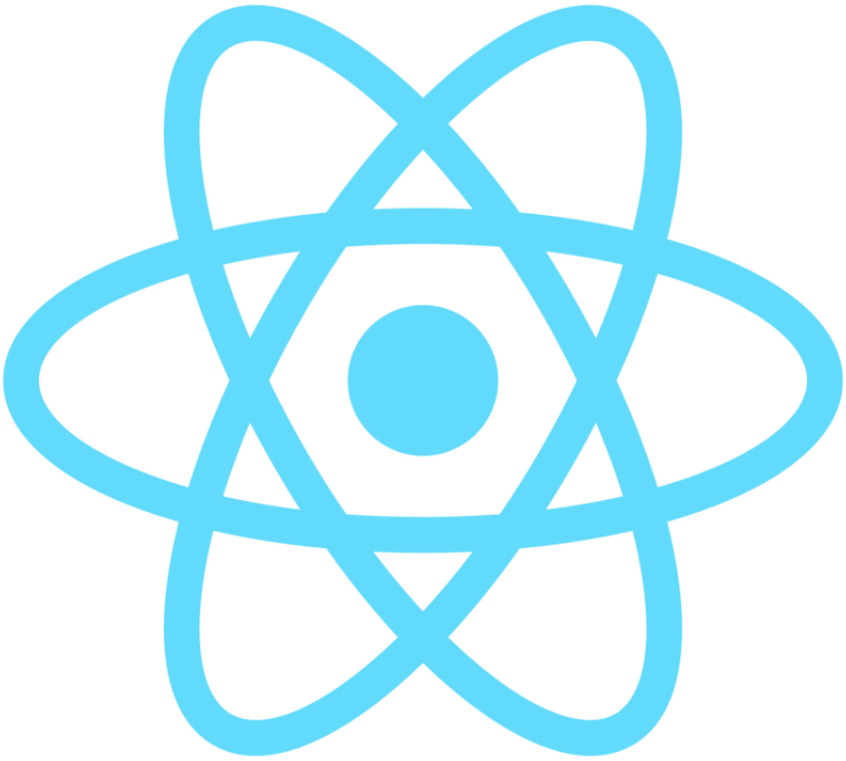
REACT NATIVE



INDICE

INTRODUCCIÓN

El desarrollo de aplicaciones nativas ha sido siempre complicado debido a que hay que desarrollar para cada plataforma de forma independiente. Una de las soluciones más utilizadas en los últimos años han sido los *WebViews*, que nos permiten encapsular una aplicación web común en una aplicación instalable.

Hay varias alternativas para unificar el desarrollo de estas aplicaciones, siendo una de ellas React Native. Con React Native podemos desarrollar en un único proyecto una aplicación nativa tanto para iOS, como Android o UWP.

OBJETIVO DEL PROYECTO

Este proyecto se ha enfocado en aprender a utilizar React Native para desarrollar aplicaciones nativas para Android sin necesidad de aprender un nuevo lenguaje o profundizar en Java y pudiendo utilizar técnicas y estilos propios de la programación web.

El proyecto consiste en una aplicación Android para compartir y crear documentos de texto que puedan ser organizados en una estructura de capítulos. A través del proyecto se explicará la estructura y organización del mismo, así como los tipos de componente fundamentales en React y React Native.

TECNOLOGÍAS Y LIBRERÍAS UTILIZADAS

Para este proyecto se han utilizado diversas tecnologías, destacando principalmente **React** + **React Native**, **React Navigarion** y **Redux**.

Por detrás de React Native se encuentran Gradle y Java para *traducir* la aplicación a un lenguaje nativo, ya sea Android, iOS o UWP (este caso con C# en vez de Java).

Menciones:

También se ha utilizado Ramda para realizar operaciones con listas y objetos, y Native Base como librería de componentes con estilos.

REACT + REACT NATIVE

React es un framework web desarrollado y mantenido por Facebook. Después del gran éxito que ha tenido, la compañía ha ido desarrollando un framework en conjunto para desarrollar aplicaciones nativas. Se trata de un desarrollo conjunto ya que React Native se basa por completo en *React*. Utiliza su misma sintaxis (JSX), sus mismos tipos de componente (*Stateless function* y Clases), además del mismo ciclo de vida para cada componente.

La sintaxis JSX

JSX es el aspecto más destacado de React. Consiste en una sintaxis que introduce código HTML en el propio JavaScript, al contrario que otros frameworks, que inyectan JavaScript en el template HTML (Vue, Angular). Esto nos permite que cada parte de un componente sea completamente programable, ya que se trata por completo de código JavaScript.

Esta sintaxis supone reaprender algunos conceptos básicos del desarrollo web, por ejemplo, todo debe ir en *camel-case* y evitando algunas palabras reservadas por el lenguaje utilizado. El ejemplo más claro es el nombre de las clases, *class* en HTML y *className* en JSX.

```
const Ejemplo = (props) => (
  <div className={props.class}>
    {props.text}
  </div>
)
```

En **React Native** esta sintaxis no supone un cambio tan grande. Para poder convertir el código web en código entendible por el sistema, se han creado nuevos componentes equivalentes. En RN no tenemos *divs*, tenemos *Views*, y así con multitud de elementos. Cada componente es una representación de su equivalente nativo (View => View en Android o View => UIViewController en iOS).

Tipos de componentes

En *React* nos encontramos con dos tipos de componentes fundamentales, las Clases y las llamadas *Stateless functions.* Vamos a empezar explicando éstas últimas.

Las *Stateless functions* o *Funciones sin estado*, son un tipo de componente cuya función es únicamente realizar una representación basada en unos datos (propiedades), que reciben de un componente padre. Este tipo de componentes no tienen ningún control sobre su propio estado y no realizan ningún tipo de operación lógica de forma directa que afecte al resto de la aplicación. En React este es el tipo de componente predilecto, ya que es muy versátil e independiente del resto, no se encuentra acoplado.

```
const VistaEjemplo = props => (
  <View>
    <Text>Texto de ejemplo</Text>
  </View>
)
```

En algunas situaciones necesitamos que los componentes realicen alguna operaciones en función de su propio estado o para mutar/modificar elementos del resto de la aplicación. Para esto utilizamos las Clases, ya que disponen de su propio contexto y estado. Otro punto a favor de las Clases es que nos permiten trabajar con el ciclo de vida de React y así controlar los cambios en el propio componente en función del punto en el que se encuentre.

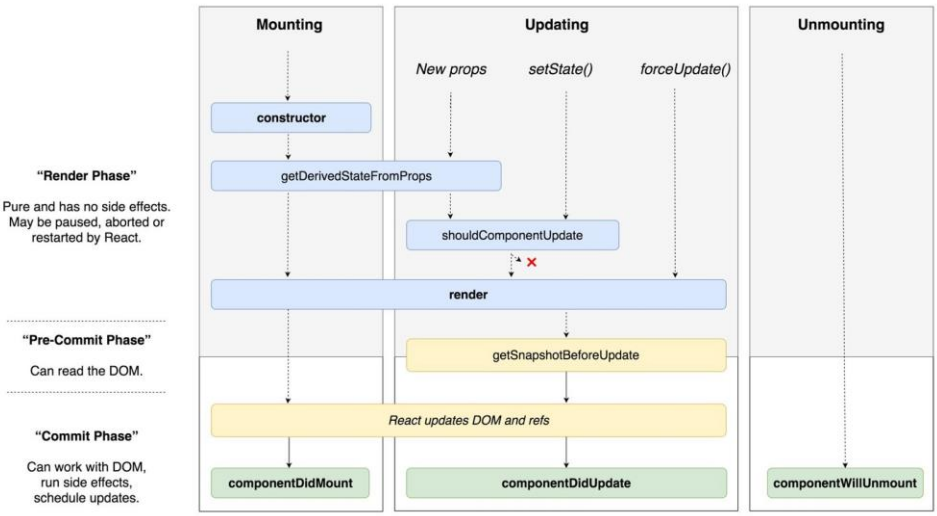
Las clases necesitan una función render, la cual debe devolver un componente válido para ser mostrado en la vista. Esta función es llamada al montar el componente y con cada cambio en el estado del mismo o en las propiedades que recibe de otro componente.

```
class VistaEjemplo extends React.Component {
  unaFuncionUtil = () => console.log('Algo')

  render () {
    const { ejemplo } = this.props

    return (
      <View>
        <Text>Texto de {ejemplo}</Text>
      </View>

    )

  }
}
```

El ciclo de vida

Todos los componentes de *React* (y por ende de *React Native*) disponen de un ciclo de vida de cuatro etapas, cada una de ellas con funciones específicas para realizar operaciones lógicas.

Todos los componentes pasan por al menos tres de las cuatro fases durante su vida útil: Montaje del componente, Actualización y Desmontaje del componente. Además hay esa mencionada cuarta fase, la cual se encarga de gestionar y tratar los errores de forma similar a un *Try-Catch.*



REACT NAVIGATION

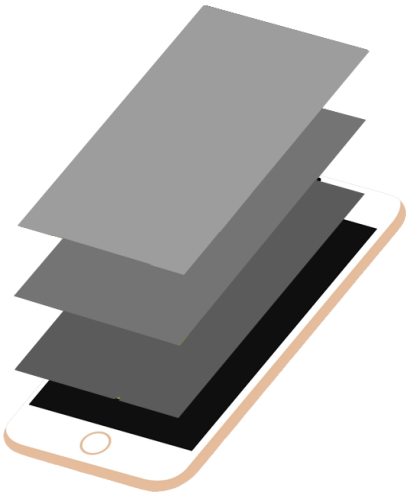
React Navigation es una libreria desarrollada para React Native para suplir la falta del soporte nativo para la navegación entre pantalla. Gracias a esta librería podemos movernos por la aplicación utilizando únicamente *JavaScript*.

Esta librería basa su funcionamiento en *screens*, componentes mayores de *React Native* que representan una vista de la aplicación. Estas vistas se almacenan en objetos de tipo *navigator*, que contienen todas las propiedades de navegación, como las acciones y parámetros de cada una de las pantallas.

En esta aplicación se usan dos tipos de *navigator*:

Stack navigator

Las pantallas se almacenan una sobre otra y la que se encuentra en la parte superior de la pila es la pantalla activa en el stack.



Drawer navigator

Este tipo de navigator funciona internamente de la misma manera que los *StackNavigators*, pero además incluye todo lo necesario para configurar un menú lateral desplegable para la aplicación, otorgándole toda la funcionalidad necesaria para la navegación desde el mismo.

REDUX

Redux es una librería para controlar el estado general de la aplicación. Aquí se almacenan desde datos generales cargados de forma dinámica como estados compartidos por varios componentes.

Esta librería basa su funcionamiento en el lanzamiento de acciones. Estas acciones son las que se encargan de realizar cambios en el estado y pueden ser tanto asíncronas como síncronas.

Redux y sus tres principios fundamentales

Todo el estado de la aplicación se guarda en una única *store*, el arbol de datos que establece el mismo. A esto se le conoce como la **Única fuente de verdad.**

El estado es de solo lectura, esto quiere decir que la única forma de modificarlo es mediante el lanzamiento de acciones, que cuentan con un objeto que describen el cambio realizado.

El estado se modifica solo con funciones puras. Las funciones que modifican el estado toman el estado anterior, el tipo de acción y devuelven el nuevo estado modificado.

La conexión con los componentes

Para conectar Redux a un componente utilizamos unos *helpers* que nos permiten mapear tanto el estado como las acciones disponibles (previamente definidas) y asociarlos a propiedades del componente.

```
import { connect } from 'react-redux'
import { NavigationActions } from 'react-navigation'

import { setBookmark } from '../actions/profile'
import BookmarksView from '../screens/BookmarksView'

const mapStateToProps = ({ book, profile }) => ({
  bookmarks: profile.aditionalInfo.bookmarks.map(bookmark => ({
    ...book.books.find(bk => bk.info.bookId === bookmark.bookId).info,
    chapterId: bookmark.chapterId,
    chapterTitle: bookmark.chapterTitle
  }))
})

const mapDispatchToProps = dispatch => ({
  handleDeleteBookmark: bookmark => dispatch(setBookmark(bookmark)),
  handleOnPress: bookId =>
    dispatch(NavigationActions.navigate({
      routeName: 'BookDetails',
      params: { bookId }
    }))
})

export default connect(mapStateToProps, mapDispatchToProps)(BookmarksView)
```

ESTRUCTURA DEL PROYECTO

El proyecto se divide fundamentalmente en dos carpetas: *android* y *app.*

Android

En la carpeta *android* nos encontramos toda la parte interna de la aplicación nativa. Aquí podemos configurar aspectos generales de Android, como el icono y el color de los elementos principales, así como los puntos de entrada de la aplicación y la configuración de la *build*.

Este directorio apenas se modifica durante el desarrollo, siendo utilizado sobretodo para el aspecto que se va a mostrar al instalar la aplicación (nombre e icono).

App

En este proyecto la estructura de carpetas está pensada para tener en cada una un tipo de componente.

App

En este directorio tenemos los componentes padres de toda la aplicación, utilizados para conectar Redux y React Navigation.

Components

Aquí guardamos todos aquellos componentes "tontos", que principalmente serán *Stateless functions*. Estos componentes son los elementos más pequeños de la aplicación.

Containers

Los containers son los componentes "listos" de la aplicación. Son los encargados de conctar el estado y las acciones con los componentes, además de realizar otro tipo de operaciones lógicas, como filtros.

Screens

Las *screens* son las vistas que luego utilizamos en React Navigation. Comunmente suelen llevar asociado un *container* que se encarga de gestionar su información.

Además nos encontramos otras carpetas, como las usadas para configurar Redux y React Navigation.

Actions

Aquí describimos las acciones de Redux, que serán observadas cuando se lancen e interpretadas para realizar los cambios.

Epics

Los Epics son aquellas acciones de Redux que se ejecutan generalmente de forma asíncrona, como pueden ser peticiones a base de datos o a apis.

Reducers

Los Reducers son aquellas acciones que modifican el estado de forma síncrona. Aquí encontramos las funciones puras mencionadas en los tres principios de Redux.

Config

En el directorio config tenemos distintas configuraciones de la aplicación, como es la configuración inicial de Redux y los distintos *navigators* que van a conformar la navegación entre las diferentes pantallas.

CONCLUSIÓN

React Native es una herramienta muy potente para el desarrollo de aplicaciones nativas. Gracias al framework desarrollado por Facebook podemos realizar aplicaciones para las dos plataformas móviles principales sin tener que aprender dos lenguajes distintos. También cuenta con soporte por la propia *Microsoft* para el desarrollo de aplicaciones nativas en *Windows*, lo que nos permite crear una única aplicación útil para los dos grandes entornos de trabajo en la actualidad.

El uso de tecnologías web permite un aprendizaje mucho más rápido que de otras maneras ya que el sistema de vistas es realmente intuitivo y facilmente adaptable desde una versión web. Al principio se para una fase de adaptación, sobretodo al venir de otro tipo de frameworks, que puede ser sufrida, pero en seguida se pueden ver los la aplicación en funcionamiento y el resultado es realmente satisfactorio. Además, al estar basado en *React* y *JavaScript*, muchas de las librerías son compatibles con cambios menores o sin cambios directamente, por lo que es muy sencillo extender su funcionalidad mediante terceros.

Como principal contra podemos remarcar que se necesita un entorno más potente y complejo para el desarrollo, ya que necesitamos un dispositivo real o emulado para probar las aplicaciones. Además algunos problemas que surgen durante el desarrollo están relacionados con la parte *Android* directamente, la cual es más compleja y desconocida al no haber trabajado tanto sobre ese apartado.

BIBLIOGRAFÍA

React: https://reactjs.org/

React Native: https://facebook.github.io/react-native/

React Navigation: https://reactnavigation.org/

Redux: https://github.com/reduxjs/redux