ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Aplicación para Android de accesibilidad a las playas

AUTOR: Khalil Merzouki

DIRECTOR: Juan Carlos Sánchez Aarnoutse

Octubre / 2016

Autor	Khalil Merzouki
E-mail del Autor	mekhalil@hotmail.com
Director	Juan Carlos Sánchez Aarnoutse
E-mail del Director	Juanc.sanchez@upct.es
Codirector(es)	
Título del PFC	Aplicación para Android de accesibilidad a las playas
Descriptores	

Resumen

Desarrollar una aplicación para sistemas Android que integre toda la información relevante a la accesibilidad de las playas para personas con discapacidad Física. La aplicación debe operar sin necesidad de descargar datos pero debe permitir la actualización de la misma.

Titulación	Ingeniero de Telecomunicaciones
Intensificación	
Departamento	Tecnologías de la Información y comunicaciones
Fecha de Presentación	octubre - 2016

Índice General

Índice de figuras

Índice de Tablas

1 Introducción

La Dirección General de Turismo de la Región de Murcia puso en marcha el Plan Regional de Accesibilidad en Playas, desarrollado por la Oficina Técnica de Accesibilidad de FAMDIF, que tiene como objetivo ofrecer información actualizada y fiable sobre la dotación de equipamientos accesibles y de servicios de ayuda al baño en temporada alta en las 57 playas de la región, cuya divulgación se lleva a cabo a través de la red de Oficinas de Turismo y el portal web turístico regional y otras páginas web especializadas en el ámbito de la discapacidad a nivel nacional.

En los últimos años, el número de usuarios de aplicaciones móviles no para de crecer, a causa de las ventajas que ofrecen los dispositivos móviles sobre los ordenadores de mesa y los portátiles, como, por ejemplo: ubicuidad, eficiencia, tamaño, rapidez de ejecución de tareas y precio. Por lo tanto, las aplicaciones móviles son, hoy en día, un medio muy eficaz para divulgar todo tipo de información, sobre todo si está relacionada con la ubicación de lugares de interés público, como pueden ser las playas accesibles para personas con discapacidad física.

Aprovechando los datos sobre accesibilidad a las playas de la Región de Murcia, aportados por Plan Regional de Accesibilidad en Playas, y las bondades de los smartphones, este proyecto consiste en la creación de una aplicación para móviles que ofrece información actualizada sobre la accesibilidad a las 57 playas murcianas para personas con discapacidades.

1.1 Objetivos del proyecto

El proyecto consiste en el desarrollo de una aplicación cuyo objetivo es permitir a los usuarios encontrar las playas accesibles para las personas con discapacidad física en la región de Murcia, y que ofrece toda la información relacionada con dichas playas.

1.1.1 Requisitos

Los requisitos de la aplicación son los siguientes:

1. La aplicación es para sistemas de Android.
2. La aplicación debe operar sin la necesidad de descarga de datos.
3. La aplicación debe permitir la actualización de la información que contiene.

1.1.2 Información a mostrar

La aplicación debe proporcionar los siguientes datos sobre cada playa:

1. Plazas de estacionamiento reservados para las personas con discapacidad

2. Rampas de acceso a la playa
3. Pasarelas sobre la arena
4. Tramos enrollables
5. Vestuarios adaptados
6. Duchas interiores adaptadas
7. Aseos adaptados
8. Zonas de sombra adaptadas
9. Zona de baño adaptadas
10. Sillas anfibias
11. Muletas anfibias
12. Modalidad de ayuda al baño
13. Calendario y horario de ayuda al baño
14. Actividades adaptadas a personas con discapacidad
15. Valoración general

Además de toda la información anteriormente citada, debe proporcionar la ubicación exacta de la playa en el mapa e imágenes de los equipamientos accesibles.

1.2 Estructura del documento

Este documento ha sido dividido en varios capítulos con el fin de agilizar y facilitar al lector y su seguimiento.

La estructura del documento se detalla a continuación.

Capítulo 2:

Se detallan las tecnologías principales empleadas para el desarrollo de la aplicación.

Capítulo 3:

Se explican las fases de implementación seguidas para llevar a cabo el desarrollo de la aplicación y el aspecto técnico.

Capítulo 4:

Se explica de forma detallada el uso y funcionamiento de la aplicación.

Capítulo 5:

Se exponen las conclusiones finales del desarrollo de la aplicación, así como una serie de propuestas para la mejora de la aplicación objeto de este proyecto.

2 Tecnologías empleadas

Para llevar a cabo el desarrollo de la aplicación se ha optado por el uso de varias tecnologías, que se dividen en tres grupos principales:

1. Software de diseño
2. Herramientas de desarrollo
3. Herramientas de prueba

2.1 Software de diseño

Para el diseño del layout de la aplicación se ha hecho uso de dos aplicaciones:

1. Sketch
2. Invision

2.1.1 Sketch

Sketch es una de las mejores aplicaciones para el diseño web, cuya fama se debe a que está orientada al desarrollo de páginas web y de aplicaciones móviles, por lo que proporciona una serie de herramientas para facilitar las tareas de creación de componentes visuales y ofrece plantillas para dispositivos iOS, con un precio bastante asequible comparado con las herramientas de Adobe.

Además, existen muchos recursos gratuitos por la web, que facilitan el diseño de una aplicación en Android, como, por ejemplo, los proporcionados en sketchappsources.com. Dichos recursos permiten el diseño de un layout decente, sin tener conocimientos previos de diseño.

Sketch también se destaca por poder mostrar los diseños en forma de páginas, de esta forma se facilita la comparación entre varias páginas para seguir un mismo prototipo y permite reutilizar componentes visuales de forma más sencilla.

2.1.2 Invision

Invision es una aplicación web que permite compartir la vista de diseños web entre varias personas, mediante la generación de una dirección web privada relacionada con el diseño que se desea compartir. Los usuarios pueden dejar comentarios sobre partes del diseño y navegar entre las imágenes de cada página, como si fuera una aplicación real, y permite también mostrar el diseño en un móvil iPhone virtual, adaptando el diseño a las pantallas de los varios dispositivos de la marca, lo que facilita ver cómo se va a dibujar la aplicación en un móvil. La aplicación es gratuita para un número limitado de proyectos.

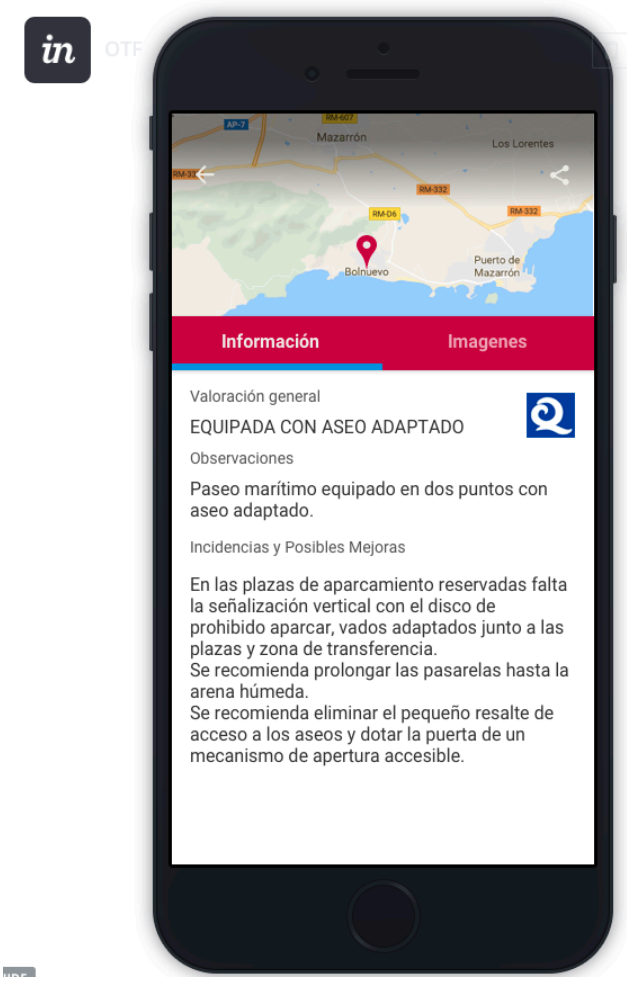

FIGURA 2-1 INVISION DISEÑO EN FORMATO MÓVIL

2.2 Herramientas de desarrollo

Las herramientas de desarrollo se dividen en 7 categorías:

1. Lenguajes de programación
2. Librerías
3. Base de datos
4. Entornos de desarrollo integrados (IDE)
5. Gestión y construcción
6. Control de versiones
7. Repositorio

2.2.1 Lenguajes de programación

2.2.1.1 JavaScript

A pesar de sus orígenes humildes como un lenguaje de programación para páginas web sencillas, JavaScript domina ahora la programación web y en estos últimos años se ha convertido en la fuerza principal que impulsa el cambio y la innovación en la Web, permitiendo la creación de avanzadas aplicaciones.

El avance de JavaScript se debe a las innovaciones técnicas en el lenguaje, a las nuevas herramientas y librerías, y a una próspera comunidad de desarrolladores entusiastas. Además, la aparición de nuevas librerías, como PhoneGap, permite a los desarrolladores usar JavaScript para crear aplicaciones para una amplia gama de dispositivos móviles. Incluso Microsoft, a partir de su sistema operativo Windows 8, ofrece herramientas de desarrollo de aplicaciones en HTML y JavaScript. Mozilla tiene la intención de crear su sistema operativo para dispositivos móviles, y también el Chrome book de Google.

Los avances en la velocidad de ejecución de JavaScript y el avance de HTML5 significan que estas aplicaciones son capaces de ofrecer niveles sin precedentes de sofisticación funcional y en la capacidad de respuesta de interfaz de usuario. Además, destaca la creación del servidor NodeJS para usar JavaScript en el lado servidor. JavaScript goza de su máxima popularidad, siendo, con diferencia, el lenguaje de programación más popular en Github, donde representa aproximadamente el 21% de todos los proyectos alojados públicamente, superando con diferencia el resto de los lenguajes.

Se puede decir que es la época dorada de JavaScript en el desarrollo de aplicaciones web, de aplicaciones de escritorio y móviles. Sin embargo, la programación de grandes y complejas aplicaciones en JavaScript no es una tarea fácil, por su legado como un simple lenguaje de programación web, su falta de organización y mantenimiento de código (a diferencia de la que se encuentra, por ejemplo, en los proyectos Java). Las herramientas no tan desarrolladas y el aumento de las nuevas librerías complican la correcta elección para cada aplicación.

Teniendo en cuenta todas las limitaciones anteriores, JavaScript tiende a enredarse en un código desorganizado. Por lo tanto, un proyecto en JavaScript tiene que superar estos desafíos para conseguir un código escalable y fácil de mantener. Para ello, es importante conseguir una arquitectura eficiente y optimizar el código.

Para el desarrollo de nuestra aplicación se ha utilizado una nueva versión de JavaScript, es la versión ECMAScript 2015.

ECMAScript 2015 es la versión de junio 2015 de la especificación de lenguaje de programación de tipo script, en la cual pertenece JavaScript. La versión de junio 2015, conocida como ES6, integra nuevas funcionalidades, que ha ido solicitando la comunidad de desarrolladores de este lenguaje a lo largo de estos últimos años, como clases, módulos, iteradores for/of, promisas y una serie de funciones para tratamiento de Arrays.

La mayoría de los navegadores no soportan todas las nuevas funcionalidades del nuevo estándar, pero existen procesadores que convierten el código en su equivalente en ES5 como BabelJs.

2.2.1.2 Java

Java es uno de los lenguajes más populares, particularmente para aplicaciones de cliente-servidor de web. Fue originalmente desarrollado por James Goslind de Sun Microsystems (compañía adquirida por Oracle) y publicado en 1995 como un componente fundamental de la plataforma de Java de Sun Microsystems. Las aplicaciones de Java son generalmente compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente ni el sistema operativo de la misma. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos.

Java usa el paradigma de la programación orientada a objetos, incluye por defecto soporte para trabajo en red, ejecuta código en sistemas remotos de forma segura y, además, es fácil de usar.

Una de las ventajas del lenguaje de programación Java es la recolección de basura, que solventa el problema de fugas de memoria. El programador determina cuándo se crean los objetos, mientras que el entorno en tiempo de ejecución de java (Java Runtime) es el responsable de gestionar el ciclo de vida de los objetos. De esta forma, se permite una fácil creación y eliminación de objetos y mayor seguridad.

2.2.1.3 SQL

SQL es un lenguaje declarativo de acceso a la base de datos relacionales. Sus principales funciones son insertar, recuperar, actualizar y borrar datos de la base de datos.

2.2.2 Librerías

2.2.2.1 React

React es un framework de JavaScript, que fue creado originalmente por ingenieros de Facebook para resolver los desafíos implicados en el desarrollo de interfaces de usuario complejas con conjuntos de datos que cambian con frecuencia. Además, requiere una solución escalable al nivel de la escala que tiene Facebook o Instagram.

React nació en el departamento encargado del desarrollo de las interfaces anuncios de Facebook, en el cual habían estado utilizando el enfoque tradicional Modelo-Vista-Controlador en la capa del cliente, ya las aplicaciones como la de los anuncios de Facebook consisten normalmente en renderizar la plantilla de datos bidireccionales entre la vista y el controlador.

React cambió la forma en que la estas aplicaciones se crean, aportando un nuevo concepto al desarrollo web, por lo que después de su lanzamiento en 2013 la comunidad de desarrolladores web le prestó mucho interés, pero recibió muchas críticas sobre su nuevo enfoque, que desafía a las convenciones convertidas en los estándares de facto para las mejores prácticas de los framework de JavaScript. React introduce nuevos paradigmas y cambios en el status quo de lo que se necesita para crear aplicaciones escalables y mantenibles en JavaScript e interfaces de usuario, tales como el DOM virtual, JSX y los conceptos de flujo de datos.

Hoy en día existen varios framework en JavaScript para el desarrollo de aplicaciones de forma más ordenada y mantenible, siguiendo las pautas de diseño o arquitecturas como Angular 1, que implementa tanto el concepto Modelo Vista Vista-Modelo (MVVM) como el de Modelo Vista Controlador (MVC). Pero el equipo de ingenieros de Facebook tiene otro tipo de desafío que estos Framework no cubren: implementar interfaces de usuario a gran escala y con cambios de datos con mayor frecuencia. Para cubrir esta necesidad se ha decidido crear un nuevo Framework que hace frente a la visualización de datos en una interfaz de usuario de gran escala y, por lo tanto, facilitar la construcción de interfaces de usuario complejas como la del propio Facebook o Instagram.

Para entender el concepto que implementa React es importante saber la arquitectura de MVVM y los estándares de buena práctica que se están siguiendo en la mayoría de los más usados Framework en la actualidad y comparar con el concepto seguido por React.

2.2.2.1.1 MVVM

El Modelo Vista VistaModelo es una arquitectura en la que se basan la mayoría de los Frameworks de JavaScript en la actualidad y tiene como objetivo separar el desarrollo de

la interfaz de usuario y la lógica de negocio en una aplicación. Para conseguir este objetivo, la mayoría de las implementaciones de esta arquitectura utiliza el enlazado de datos (data binding) para obtener la separación entre la vista y el resto de las capas.

La implementación de esta arquitectura en JavaScript se lleva a cabo mediante librerías estructurales como KnockoutJs o AngularJs, aunque a este último se le considera MV* porque puede ser MVVM o MVC según como se quiere utilizar.

2.2.2.1.2 El Modelo

El Modelo en MVVM representa los datos o la información que maneja la aplicación. Un ejemplo del modelo es una cuenta de usuario que contiene como información su nombre, avatar y cuenta de correo electrónico.

```
var userAcount = {
      name: 'John Doe',
      avatar: 'johndoeAvatar.jpg'
      eMail: 'johndoe@example.com'
}
```

FIGURA 2-2 EJEMPLO DE UN MODELO EN JAVASCRIPT

El Modelo contiene información, pero no la gestiona. El Modelo no es responsable de tratar el formato ni de mostrar la información en el navegador. Mientras que mostrar la información es tarea de la Vista, el tratado de la información es responsabilidad de la lógica de negocio. La única excepción permitida para validar los datos en el Modelo es para poder definir o actualizar un modelo existente.

En KnockoutJs, los modelos se rigen por la definición anterior, aunque se suele usar las llamadas Ajax, tanto para leer como para escribir los datos del Modelo.

2.2.2.1.3 La Vista

La Vista es la parte de la aplicación donde interactúan los usuarios. Es la representación del estado del VistaModelo en la interfaz de usuario, por lo tanto la Vista se considera una parte activa, aunque en el MVVM puede ser también pasiva, cuando sólo muestra datos y no acepta entradas del usuario.

La Vista no tiene por qué tener conocimiento del modelo, pero es necesario el conocimiento de la VistaModelo para poder interactuar con el modelo, mediante el data-

binding que le permite ejecutar eventos y comportamientos, pero no gestionar estados que es labor del VistaModelo.

La Vista en KnockoutJs es simplemente un documento HTML con enlazado declarativo (Declarative Binding) para comunicarse con la VistaModelo. La vista en KnockoutJs muestra la información desde la VistaModelo, le pasa comandos (por ejemplo, si usuario hace clic en un elemento) y se actualiza cada vez que cambia el estado de la VistaModelo.

2.2.2.1.4 La VistaModelo

La VistaModelo puede considerarse un controlador especializado que actúa como un convertidor de datos, que transforma la información del Modelo en información de la Vista mediante el paso de comandos desde la Vista al Modelo.

Un ejemplo de la función de la VistaModelo es la conversión de una fecha en el modelo que se recibe del servidor en formato UNIX, por ejemplo 1333832407, donde es necesario convertir la fecha para mostrarla en Vista. Esta conversión se lleva a cabo por la VistaModelo al formato del 07/04/2012 17:00:00.

Además de encargarse de darle formato a los datos enviados a la Vista, la VistaModelo también actualiza el modelo con las acciones del usuario en la Vista mediante el lanzamiento de eventos.

2.2.2.2 Estándares de buenas prácticas

Para las grandes aplicaciones con JavaScript se considera un requisito indispensable para el buen funcionamiento, mantenimiento y testeo de la aplicación el respeto de los estándares de buena práctica.

Los estándares de buena práctica engloban una serie de aspectos, como comentarios, sangrado o las convenciones de nombres. Existen varias reglas, la mayoría de ellas parecidas a las de otros lenguajes.

JavaScript es un lenguaje interpretado, por lo que no tiene restricciones de código. El desarrollador debe tener cuidado para no convertir el código en una pesadilla a la hora de hacer correcciones o cambios.

Los problemas más comunes que ha de evitar a la hora de desarrollar un aplicación basada en el lenguaje JavaScript son: combinar código de los tres lenguajes (HTML, CSS y JavaScript) y separar los tres lenguajes en la interfaz de usuario.

La interfaz de usuario está formada por tres capas (JavaScript, HTML y CSS). Para facilitar el mantenimiento y testeo del código hay que separar al máximo la capa de JavaScript del resto. El objetivo de la separación es conseguir identificar los puntos de cambio de manera más intuitiva. Por ejemplo, si hay que cambiar la estructura, hay que ir al HTML; si desea cambiar la estética de la página web, al CSS: y si tiene que modificar el comportamiento de la página web, al JavaScript.

2.2.2.2.1 Mantener CSS fuera de JavaScript

A partir de la versión 8 del navegador Internet Explorer, se permite incrustar código de JavaScript en CSS mediante la función “expression”.

```
.box{
        width: expression(document.body.offsetWidth + “px”);
}
```

FIGURA 2-3 EJEMPLO DE USO DE JAVASCRIPT EN CSS

Es una práctica desaconsejable por el simple hecho de que CSS es para los estilos de la página y no debe tener funciones complejas que nos pueden confundir al fallar y también por el hecho de ser exclusiva para el navegador IE (*Internet Explorer* de *Microsoft*), por lo que no se adapta al resto de los navegadores.

2.2.2.2.2 Mantener CSS fuera de JavaScript

En JavaScript existe la posibilidad de cambiar los estilos del DOM, mediante el uso de las propiedades del objeto style.

```
element.style.color = “red”;

element.style.left = “10px”;

element.style.top=”100px”;
```

FIGURA 2-4 EJEMPLO DE USO DE CSS EN JAVASCRIPT

Es una solución poco efectiva, por el hecho de mezclar estilos y comportamiento, además de aumentar el coste computacional debido a que cada cambio es una llamada al DOM.

La solución más sencilla y eficiente es añadir una clase al elemento en el DOM que

tiene un conjunto de estilos deseados configurados en CSS.

```
element.className += “reveal”;
```

FIGURA 2-5 EJEMPLO DE ALTERNATIVA AL USO DE CSS EN JAVASCRIPT

2.2.2.2.3 Mantener JavaScript fuera de HTML

HTML permite la inclusión de código JavaScript. Es muy común utilizar el “onclick” en un botón o la función eval() para ejecutar un código HTML. Se considera una mala práctica, porque, por ejemplo, usar un onclick que llama a alguna función que todavía no existe implicaría un fallo en la llamada. Además, dificulta la tarea de testing y el mantenimiento.

Como solución es preferible usar eventos que escuchan la acción requerida y se encarguen de ejecutar las funciones deseadas para cada una.

Por ejemplo, al hacer clic en un elemento de la DOM con un identificador igual a actionBtn, el evento se puede escuchar mediante la función addEventListener.

```
function doSomething(){

    console.log(this);

}

var btn = document.getElementById("actionBtn");

btn.addEventListener("click",doSomething, false);
```

FIGURA 2-6 EJEMPLO DE ALTERNATIVA AL USO DE JAVASCRIPT EN HTML

2.2.2.2.4 Mantener HTML fuera de JavaScript

La función innerHTML de JavaScript permite añadir fracciones en forma de cadena de texto de un código HTML al documento. Es una mala práctica por el coste computacional y por la dificultad de mantener el código.

```
var div = document.getElementById("my-div");

div.innerHTML = "<h3>Error</h3><p>Invalid e-mail address.</p>";
```

FIGURA 2-7 EJEMPLO USO DE HTML EN JAVASCRIPT

La solución más sencilla es la recuperación del texto desde el servidor y añadirlo a un elemento recién creado dentro del document.

```
var div = document.createElement("div");

div.innerHTML = result;
```

FIGURA 2-8 EJEMPLO ALTERNATIVA AL USO DE HTML EN JAVASCRIPT

En la actualidad existen varias librerías para añadir textos o elementos al DOM de forma más dinámica y organizada, mediante métodos como el Binding o la creación de plantillas, que permiten separar las dos capas mejorando su mantenimiento y testeo.

2.2.2.3 El concepto de React

React no es un MVC Framework, y a menudo se le considera el V de MVC por la comunidad de desarrolladores Front-end. React es una librería para construir interfaces de usuario componibles y reutilizables.

React no utiliza plantillas de HTML, si no que construye las interfaces de usuario dividiéndolas en varios componentes, lo que quiere decir que usa JavaScript para dibujar las vistas. En las aplicaciones creadas con los Frameworks tradicionales tiene que estar escuchando continuamente los datos para que, cuando cambian, extender estos cambios al DOM. Mientras que en React, cuando el componente se inicializa, se llama al método render generando una ligera representación de la vista, una serie de marcado se produce y se inyecta al documento. Cuando los datos cambian, el método render vuelve a ser llamado, con el fin de realizar las actualizaciones de la forma más eficiente posible, comparando la llamada anterior con la nueva. De esta forma, genera un conjunto mínimo de cambios que se aplicarán al DOM. El resultado devuelto por el método render no es una cadena de texto ni un nodo del DOM, sino una ligera descripción de cómo debe estar el DOM. Así, se consigue optimizar el tiempo de actualizar los datos de una página. Este proceso se denomina *reconciliation*.

Por ejemplo, el código de arriba se puede ver que se llama a ReactDOM.render cada medio segundo con todo el componente para pintarlo en el DOM. Con este ejemplo se puede verificar que realmente el único dato que se actualiza es el tiempo, mientras el resto de los datos incluidos el introducido en la entrada de texto "<input" siguen siendo los mismos.

```
var HelloWorld = React.createClass({
  render: function() {
    return (
      <p>
        Hello, <input type="text" placeholder="Your name here" />!
        It is {this.props.date.toTimeString()}
      </p>
    );
  }
});


setInterval(function() {
  ReactDOM.render(
    <HelloWorld date={new Date()} />,
    document.getElementById('example')
  );
}, 500);
```

FIGURA 2-9 EJEMPLO DE UN COMPONENTE CON REACT

En el ejemplo anterior se puede apreciar que el método render devuelve un código HTML. Para que esto sea posible, se utiliza una sintaxis de JavaScript denominada JSX, cuya función es facilitar el desarrollo de un componente de fácil lectura y, por lo tanto, fácil de mantener. El uso de JSX no es obligatorio con React, pero la diferencia entre crear vistas sin usarlo deja claro que es una herramienta muy útil a la hora de crear componentes en React.

2.2.3 React Native

React Native es una librería creada por Facebook, que permite crear aplicaciones nativas para dispositivos iOS y Android, usando JavaScript y React.

Cuando hablamos de aplicaciones móviles se debe tener en cuenta que el entorno nativo es más potente y eficiente que el entorno web. En el entorno nativo se puede acceder a todas las utilidades y mejoras que se agregan con cada actualización, como el control de trabajos en multihilos, el reconocimiento de gestos, la decodificación de imágenes y el acceso a los componentes visuales proporcionados por la plataforma (mapas, calendarios, etcétera).

Las aplicaciones nativas tienen mejor calidad con animaciones más fluidas, imágenes más nítidas y control avanzado de los gestos de usuario, así como componentes visuales más fluidas, lo que se traduce a una mejor experiencia de usuario.

El entorno nativo es definitivamente mucho mejor que el entorno web para aplicaciones móviles, pero es también más hostil hacia los desarrolladores. Existen varias razones por las que el desarrollo de aplicaciones nativas para dispositivos móviles es mucho más complicado que el desarrollo web: el Layout, que a menudo hay que calcular de forma manual; el tamaño y la position de todos los componentes de la vista; la carencia de librerías que permitan agilizar el proceso de desarrollo y de mejorar su mantenibilidad, Además, con cada cambio se requiere una nueva compilación, mientras que en el entorno web se puede ver el resultado de un código con solo salvar el fichero y ejecutarlo en el navegador. Esto hace que el desarrollo sea mucho más lento que las nuevas funcionalidades sean más difíciles de probar. En definitiva, el desarrollo de aplicaciones nativas es mucho más lento y complejo que el web.

React Native fue anunciado en 2015 por Facebook, que lo creó con el propósito de exportar el concepto de React para crear aplicaciones nativas en iOS y Android. React Native utiliza JavaScript para llamar a las APIs nativas. Esta metodología se denomina script native. De esta forma, se puede aprovechar el potencial del entorno nativo, con la ventaja de reutilizar funcionalidades que ya tienen en JavaScript.

Como se ha visto en el punto 2.2.2.1, React no está ligado al DOM, por lo que puede utilizar otros sistemas de vista para mostrar sus componentes, como por ejemplo en el caso de UIKit de IOS.

La librería de React Native proporciona varios componentes pueden ser integrados para una pantalla de una aplicación nativa, haciendo fácil su maquetación con el uso de flexbox, similar al usado en CSS.

Actualmente existen varios componentes que se usan tanto para Android como para iOS, pero React Native no es totalmente multiplataforma. Primero, porque es sólo para Android e iOS. En segundo lugar, existen varios componentes que funcionan en una plataforma, pero no en la otra o representan el mismo componente, pero su uso es distinto entre las dos plataformas, como por ejemplo el caso de SwitchAndroid y SwitchIOS.

```
import React, { PropTypes, Component } from 'react';
import {
  StyleSheet,
  Text,
  TouchableHighlight,
} from 'react-native';

export default class NavButton extends Component {
  render() {
    return (
      <TouchableHighlight
        style={styles.button}
        underlayColor="#B5B5B5"
        onPress={this.props.onPress}>
        <Text style={styles.buttonText}>{this.props.text}</Text>
      </TouchableHighlight>
    );
  }
}

const styles = StyleSheet.create({
```

FIGURA 2-10 EJEMPLO DE UN COMPONENTE EN REACT NATIVE

En el ejemplo anterior se puede apreciar que se importan recursos tanto de react como de react-native, como por ejemplo Component (que es necesario para crear un componente) o el componente Text de la librería de react-native para mostrar texto en la pantalla.

Aunque el desarrollo en script native permite el uso de JavaScript para crear aplicaciones nativas, estas aplicaciones no pueden ser ejecutadas en un navegador, lo que quiere decir que seguimos con uno de los inconvenientes del desarrollo de aplicaciones nativas. Para resolver el problema React Native ha puesto un mecanismo de actualización automática de la aplicación ante cualquier cambio, además de la

depuración de código en la consola del navegador Chrome. De esta forma, se ahora un tiempo considerable al evitar compilar la aplicación cada vez que se haga un cambio.

2.2.4 OsmDroid

Es una librería para implementar mapas de Open Street Maps en Android.

Open Street Maps es un proyecto colaborativo para crear mapas gratuitos, por lo que a contrario de Google Maps y Mapbox las mapas se pueden descargar y usar si conexión a la red.

2.2.5 Base de datos.

2.2.5.1 SQLite

SQLite es un sistema de gestión de base de datos relacional. Se trata de una biblioteca escrita en C y es relativamente pequeña, no necesita servidor ni configuración previa.

SQLite es la base de datos SQL integrada en Android, donde la plataforma proporciona una serie de funciones para su gestión.

2.2.6 Entornos de desarrollo integrados (IDE).

2.2.6.1 Atom

Atom es un editor de texto gratuito y de código abierto. Uno de sus puntos fuertes es que permite la instalación de paquetes para añadir funcionalidades, convirtiéndolo en un popular entorno de desarrollo. Hoy en día existen miles de estos paquetes de gran utilidad en la red, gratuitos y de código libre.

En este proyecto se ha usado el paquete de Nuclide, que es un producto de Facebook para el desarrollo de aplicaciones en React y React Native, y se ha utilizado también el linter-eslint para detectar código erróneo o que no cumple con los estilos de desarrollo establecidos. Los estilos de desarrollo utilizados son los mismos empleados por Airbnb, una de las aplicaciones populares desarrollada en React.

2.2.7 Android Studio

Android Studio es un entorno de desarrollo integrado para la plataforma de Android basado en IntelliJ IDEA de JetBrains. Android Studio permite la edición de códigos, la depuración y contiene un sistema de compilación y herramientas de rendimiento.

Android Studio utiliza Gradle para empaquetar la aplicación y gestionar su dependencia.

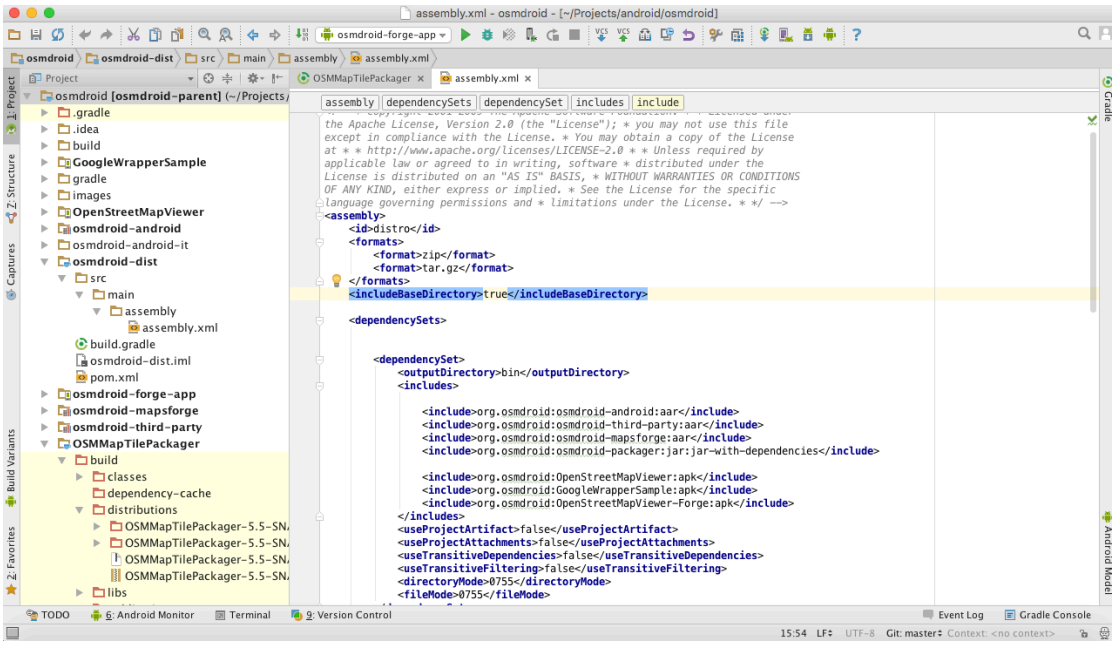

FIGURA 2-11 ANDROID STUDIO

2.3 Gestión y construcción

2.3.1 NPM

NPM es un gestor de paquetes para JavaScript, permite la gestión de dependencias y la descarga e instalación de las aplicaciones disponibles en su repositorio. En este proyecto NPM es necesario para poder descargar las dependencias con React y ejecutar la aplicación en el emulador; descargar paquete de React-native-sqlite-storage, que permite el uso de sqlite desde React Native; y React-native-vector-icons, que contiene una colección de iconos.

NPM se configura con un fichero JSON donde las dependencias se declaran de esta forma:

```
"devDependencies": {
  "babel-eslint": "^6.1.2",
  "eslint": "^3.3.0",
  "eslint-config-airbnb": "^10.0.1",
  "eslint-plugin-import": "^1.13.0",
  "eslint-plugin-jsx-a11y": "^2.1.0",
  "eslint-plugin-react": "^6.1.0"
}
```

FIGURA 2-12 EJEMPLO DE DEPENDENCIAS PARA NPM

Además de las dependencias de la aplicación existen los devDependencies, que son dependencias que se utilizan sólo en el entorno de desarrollo. En este caso, sólo se han utilizado las utilidades para el Lint del código. NPM se ejecuta mediante línea de comando.

2.3.2 Control de versiones.

2.3.2.1 Git

Git es un programa para la gestión de versiones. El control de versión es uno de los sistemas más importantes a la hora de programar aplicaciones, porque permite la gestión de los diversos cambios que se realizan sobre el código o la configuración del proyecto. Teniendo en cuenta que para crear una aplicación se hacen muchos cambios en muchos ficheros, el control manual es prácticamente imposible. El control de versiones permite ver el histórico de los cambios y recuperación en caso de que se rompa el código.

Git es uno más de los distintos softwares para control de versiones existentes, pero se diferencia del resto por la forma en la que modela sus datos. Mientras que la mayoría de los sistemas de control de versiones guardan la información como un conjunto de ficheros y los cambios sobre cada uno en cada momento. Git modela sus datos como un conjunto de instantáneas de una miniatura de los ficheros del sistema.

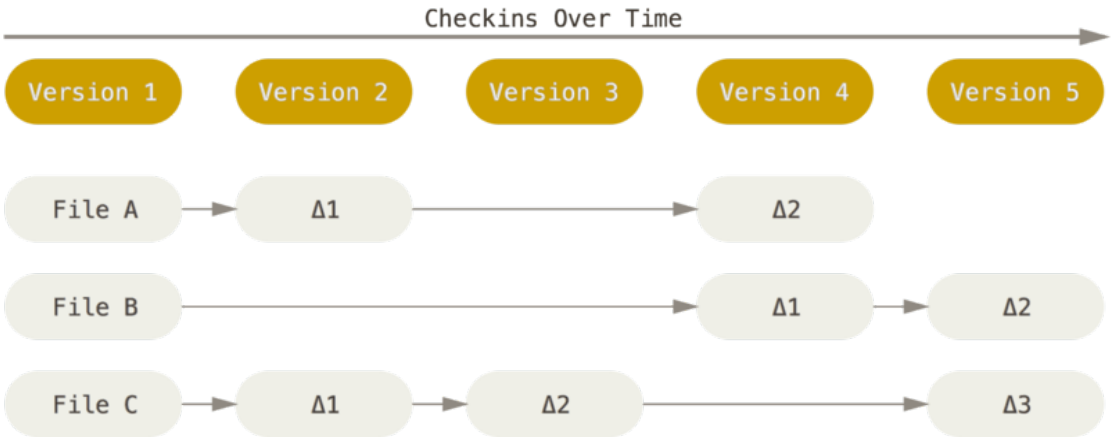

FIGURA 2-13 FUNCIONAMIENTO DE PROGRAMAS DE GESTIÓN DE VERSIONES

Cada vez que se haga un *commit,* básicamente se toma una instantánea de cómo están todos los ficheros en este momento y se guarda una referencia de la instantánea.

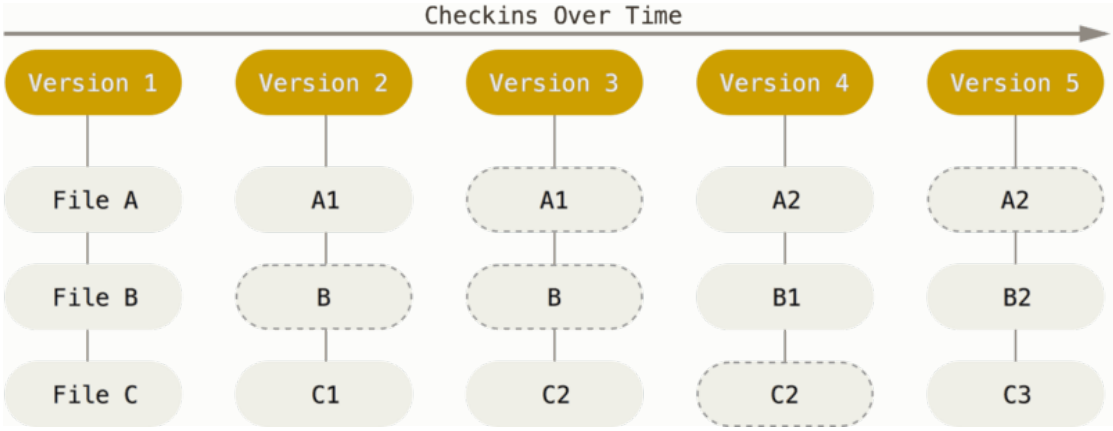

FIGURA 2-14 FUNCIONAMIENTO DE GIT

Otra de las ventajas de Git es la posibilidad de usarlo sin conexión a la red. Se puede trabajar en proyecto, hacer commit de los cambios y tenerlo todo en local hasta que se conecta a la red y se suben los cambios al repositorio remoto.

Para usar Git de una forma más sencilla, sin memorizar todos los comandos para la consola, se puede usar SourceTree que es software de escritorio gratuito que permite el uso de Git de una forma interactiva.

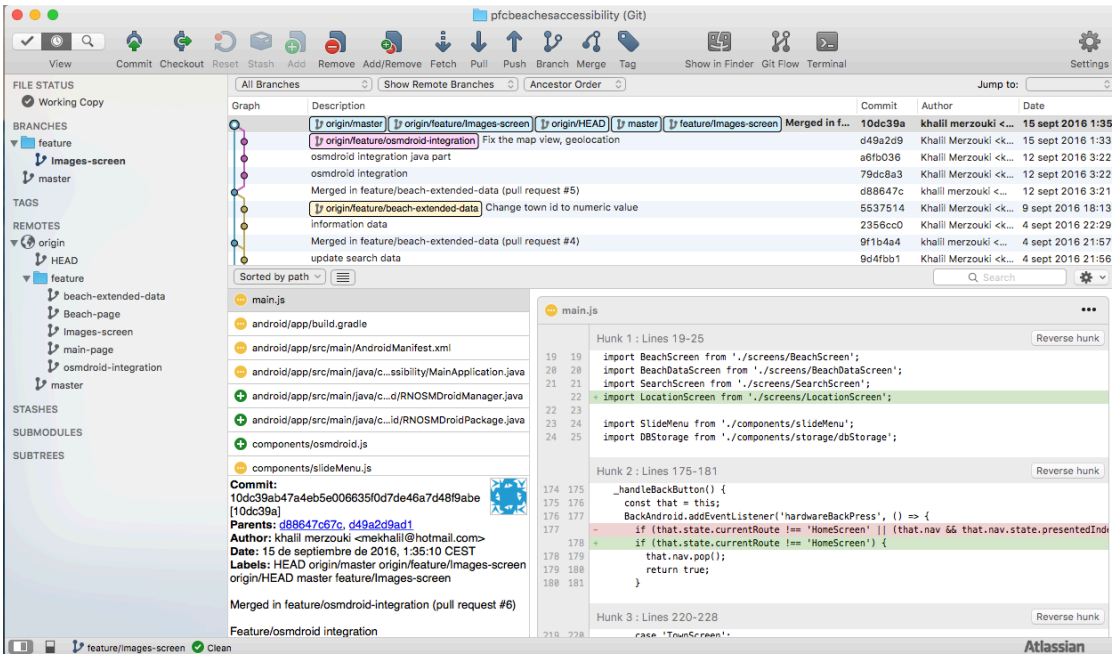

FIGURA 2-15 SOURCETREE

2.3.2.2 Bitbucket

Bitbucket es un servicio de alojamiento en la web de los proyectos que utilizan el sistema de control de versiones Git y Mercurial. Forma parte de los productos de Atlassian.

Bitbucket es gratuito para equipos hasta un máximo de 5 usuarios. Se destaca por su interfaz intuitiva y fácil de usar, además de la herramienta para gestionar los pull request.

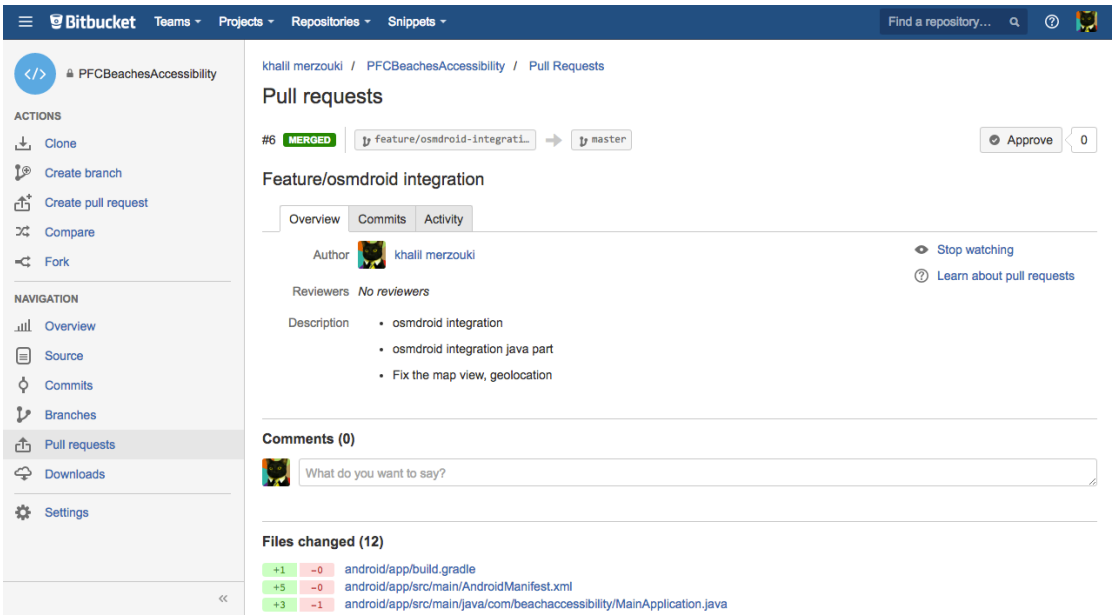

FIGURA 2-16 BITBUCKET

Las pull request son solicitudes para combinar una rama secundaria con la rama principal, permitiendo comparar, comentar en cada línea y al final aprobar el cambio o rechazarlo. De esta forma, se tiene un control sobre el código por parte de todos los miembros de un equipo.

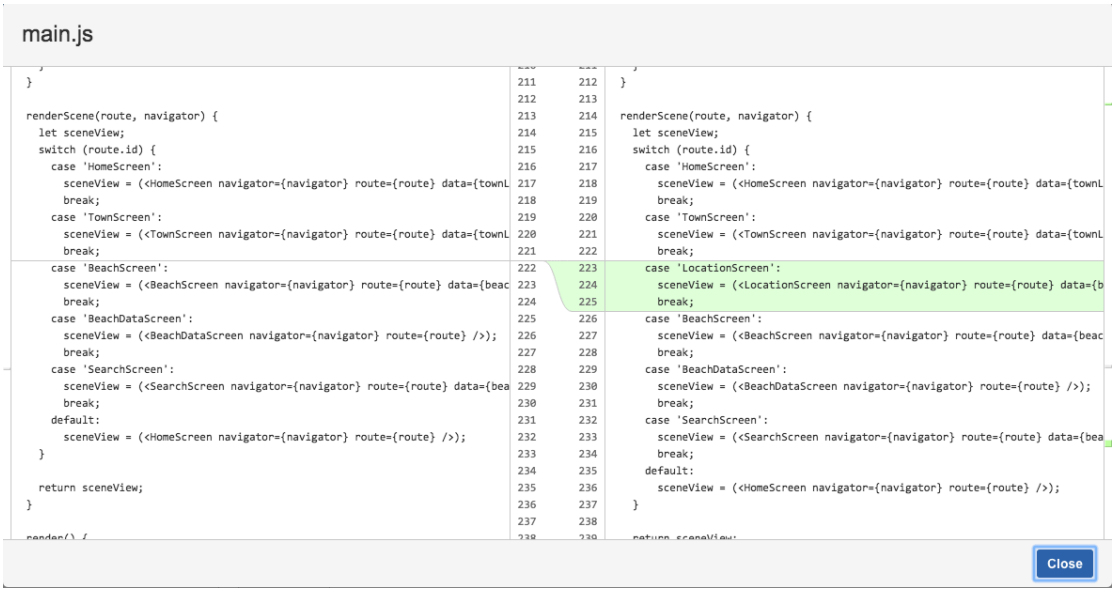

FIGURA 2-17 DIFERENCIAS EN UNA PULL REQUEST

2.4 Herramientas de prueba

1. Exponent
2. Emulador de Android

2.4.1 Exponent

Exponent es una herramienta para el desarrollo de aplicaciones móviles mediante React Native. Añade componentes nativos que no están incluidos en React, como, por ejemplo, el acceso a la cámara.

La mayor ventaja de Exponent es la facilidad con la que permite ejecutar la aplicación en dispositivos Android e iOS reales. Solamente hace falta instalar su aplicación en el escritorio e instalar su aplicación en el móvil desde play store o app store. De esta forma, al crear un proyecto, se puede compartir con el móvil usando la red local sin tener que usar cables.

Exponent ha sido muy útil en la fase de desarrollo de los componentes visuales de este proyecto, pero tiene la limitación de que no permite integrar nuevas funciones nativas, como el uso de SQLite o OSMDroid, que son necesarias para el desarrollo del proyecto.

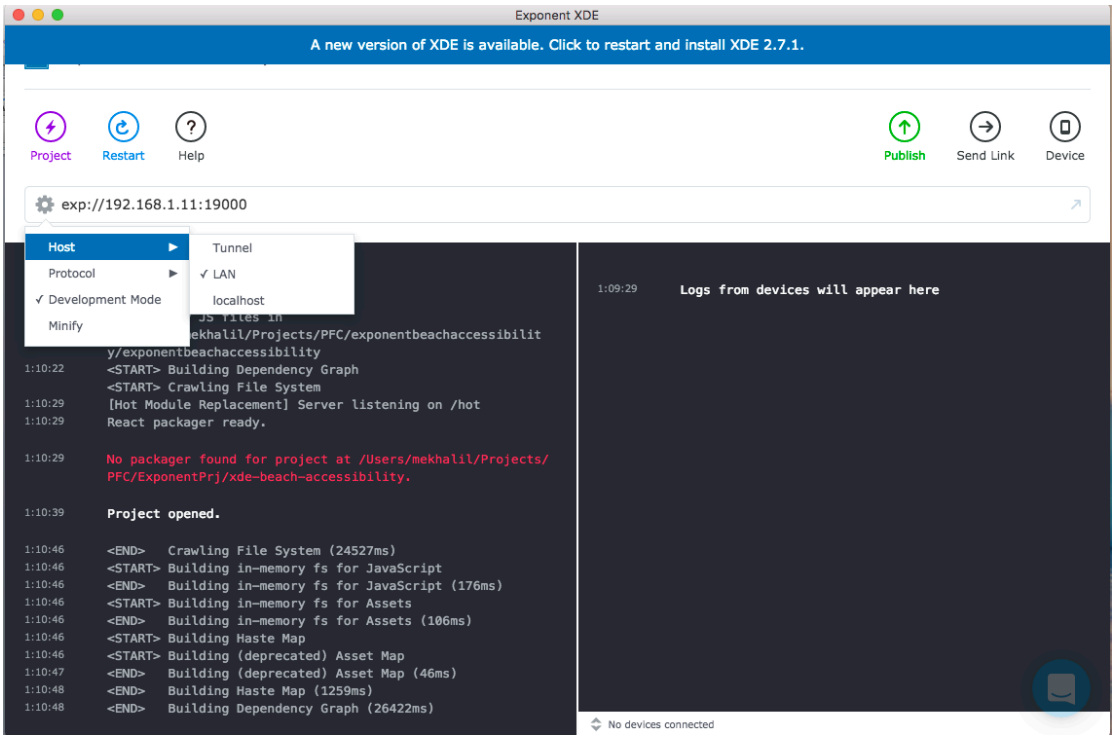

FIGURA 2-18 EXPONENT XDE

2.4.2 Emulador de Android

El emulador de Android es una herramienta importante a la hora de desarrollo de una aplicación en Android, ya que simula un dispositivo y permite probar la aplicación sin la necesidad de usar un dispositivo real. De esta forma, se puede probar la aplicación en varios tipos de dispositivos con tamaños de pantallas diferentes y versiones de Android diferentes.

FIGURA 2-19 ANDROID VIRTUAL DEVICE

3 Implementación de la aplicación

La implementación de una aplicación se consigue pasando por distintas fases, desde el estudio del caso de uso hasta su publicación.

3.1 Caso de uso

El caso de uso es un conjunto de descripciones de las interacciones entre el sistema y los actores. Para informarse sobre una playa con accesibilidad para las personas con discapacidad física, se contemplan varios casos donde el usuario puede acceder a la información. Los casos se han conseguido después de un estudio sobre las aplicaciones más populares en el mercado, aunque sean para localizar otro tipo de información, como un restaurante o una vivienda. Después del estudio de mercado, se han contemplado varios casos inspirados en varios ejemplos.

Todos los casos empiezan con una acción de un usuario con el propósito de encontrar la información sobre una playa deseada.

Caso de uso 1:

Acceso por lista de municipios.

<table>
<tr><td>CU-01</td><td>Acceso por lista de municipios</td></tr>
<tr><td>Descripción</td><td>Encontrar la playa deseada a través de la lista de municipios.</td></tr>
<tr><td>Actores</td><td>Usuario</td></tr>
<tr><td>Secuencia de pasos</td><td>
<table>
<tr><th>Pasos</th><th>Descripción</th></tr>
<tr><td>P1</td><td>El actor selecciona la lista de municipios</td></tr>
<tr><td>P2</td><td>El sistema abre una nueva ventana con la lista de los municipios disponibles y el número de playas en cada una</td></tr>
<tr><td>P3</td><td>El actor selecciona un municipio de la lista</td></tr>
<tr><td>P4</td><td>El sistema abre una nueva ventana con la lista de las playas del municipio seleccionado</td></tr>
<tr><td>P5</td><td>El actor selecciona una playa de la lista</td></tr>
</table>
</td></tr>
<tr><td>Poscondición</td><td>Se accede a la información sobre la playa seleccionada</td></tr>
</table>

TABLA 3-1 CASO DE USO ACCESO A LA LISTA DE MUNICIPIOS

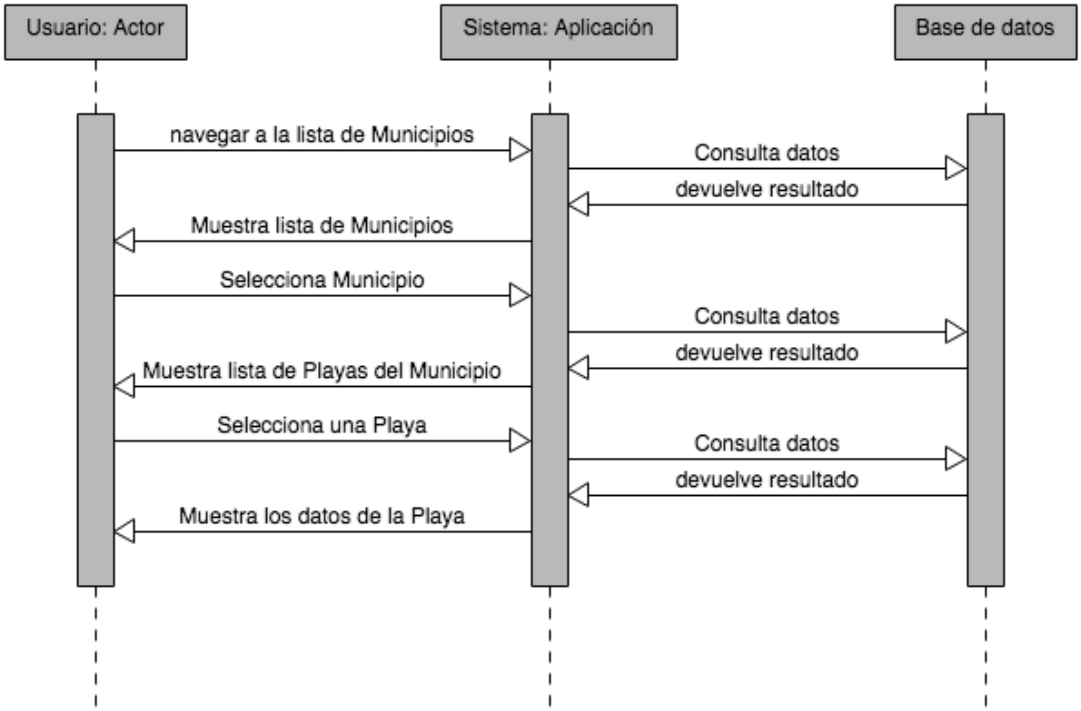

FIGURA 3-1 DIAGRAMA DE FLUJO PARA EL CASO DE USO 1

Caso de uso 2:

Búsqueda de playa.

<table>
<tr><td>CU-02</td><td>Búsqueda de playa.</td></tr>
<tr><td>Descripción</td><td>Encontrar la playa deseada a través de la búsqueda rápida.</td></tr>
<tr><td>Actores</td><td>Usuario</td></tr>
<tr><td>Secuencia de pasos</td><td>
<table>
<tr><th>Pasos</th><th>Descripción</th></tr>
<tr><td>P1</td><td>El actor selecciona la el icono de búsqueda</td></tr>
<tr><td>P2</td><td>El sistema abre una nueva ventana con un campo de búsqueda</td></tr>
<tr><td>P3</td><td>El actor introduce el texto de búsqueda</td></tr>
<tr><td>P4</td><td>Conforme el usuario está escribiendo le sale una lista con los resultados que coinciden con los caracteres introducidos</td></tr>
<tr><td>P5</td><td>El actor selecciona un resultado de la lista</td></tr>
</table>
</td></tr>
<tr><td>Secuencia alternativa</td><td>El actor introduce el nombre de un municipio. El resultado va a ser el municipio, por lo tanto, seguirá con los pasos del CU-01.</td></tr>
<tr><td>Poscondición</td><td>Se accede a la información sobre la playa seleccionada</td></tr>
</table>

TABLA 3-2 CASO DE USO BUSQUEDA DE PLAYAS

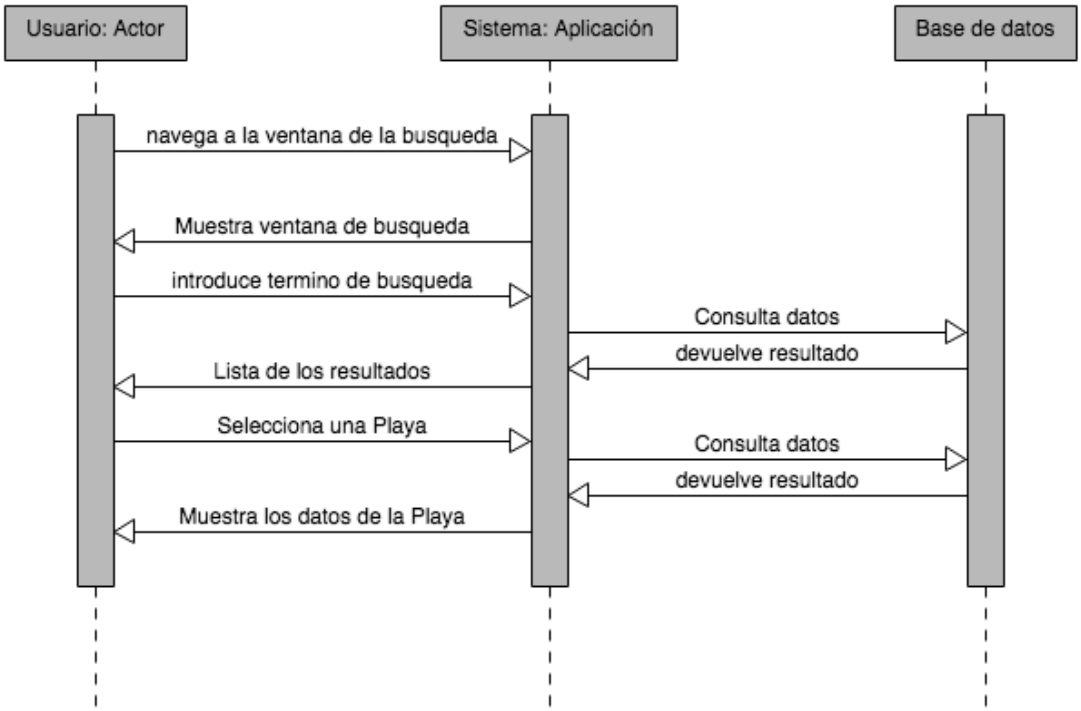

FIGURA 3-2 DIAGRAMA DE FLUJO PARA CASO DE USO 2

Caso de uso 3:

Geo-localización.

<table>
<tr><td>CU-03</td><td>Geo-localización.</td></tr>
<tr><td>Descripción</td><td>Encontrar las playas por cercanía a la localización del usuario</td></tr>
<tr><td>Actores</td><td>Usuario</td></tr>
<tr><td>Secuencia de pasos</td><td>
<table>
<tr><th>Pasos</th><th>Descripción</th></tr>
<tr><td>P1</td><td>El actor selecciona la opción de geo-localización</td></tr>
<tr><td>P2</td><td>El sistema abre una nueva ventana con la lista de las playas ordenadas de menor a mayor distancia de la localización del actor</td></tr>
<tr><td>P3</td><td>El actor selecciona una playa de la lista</td></tr>
</table>
</td></tr>
<tr><td>Secuencia alternativa</td><td>Si el usuario no tiene habilitado la geo-localización en su dispositivo, se abrirá una alerta para pedirle que la active antes</td></tr>
<tr><td>Poscondición</td><td>Se accede a la información sobre la playa seleccionada</td></tr>
</table>

TABLA 3-3 CASO DE USO GEOLOCALIZACIÓN

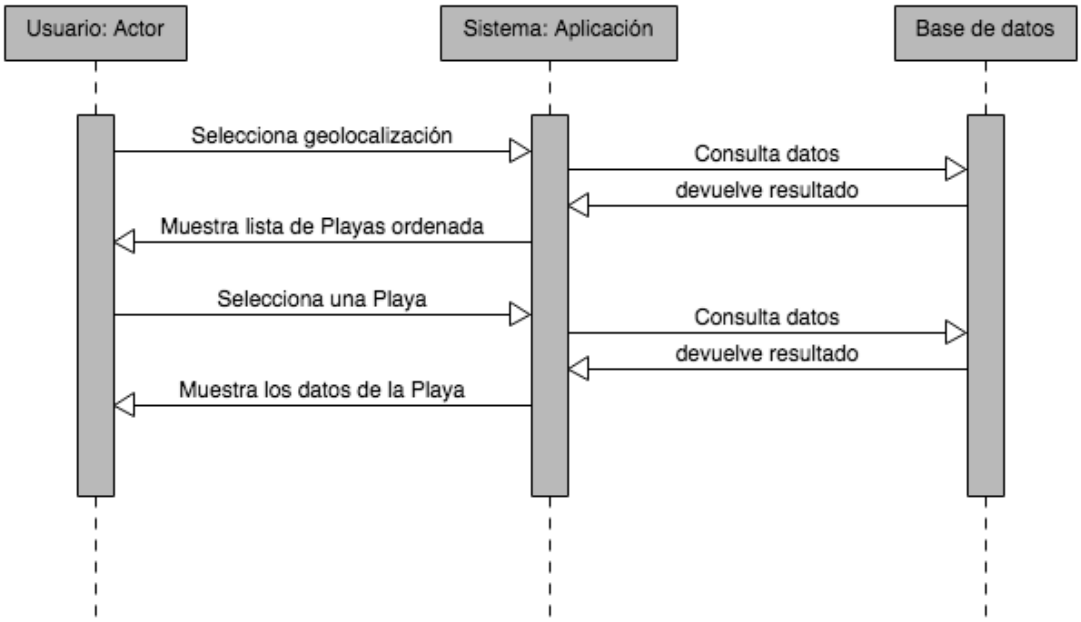

FIGURA 3-3 DIAGRAMA DE FLUJO PARA CASO DE USO 3

3.2 Diseño

El diseño de una aplicación es una parte importante y fundamental para crear una interfaz de usuario amigable e intuitiva para una mejor experiencia de usuario.

El equipo de diseño de Android ha puesto a disposición de los diseñadores y desarrolladores una serie de directrices y concejos para una mejor experiencia de usuario en Android. Algunas de estas directrices son la separación con los bordes, la anchura y altura mínima de un elemento seleccionable, animaciones y otras muchas para que la aplicación sea adaptable a los dispositivos Android.

El diseño de las Interfaces de Usuario se ha implementado partiendo de los casos de uso y respectando las directrices del equipo de diseño de Android.

Como se ha comentado en el apartado de los casos de uso el usuario tiene tres formas para acceder a la información sobre una Playa. La primera es filtrando por playas de cada Municipio de la Región, por búsqueda y por geolocalización.

Estas tres opciones se implementan para que estén siempre a disposición del usuario hasta que consiga localizar la información deseada, por lo tanto el acceso a estas opciones debe de estar en la ventana inicial.

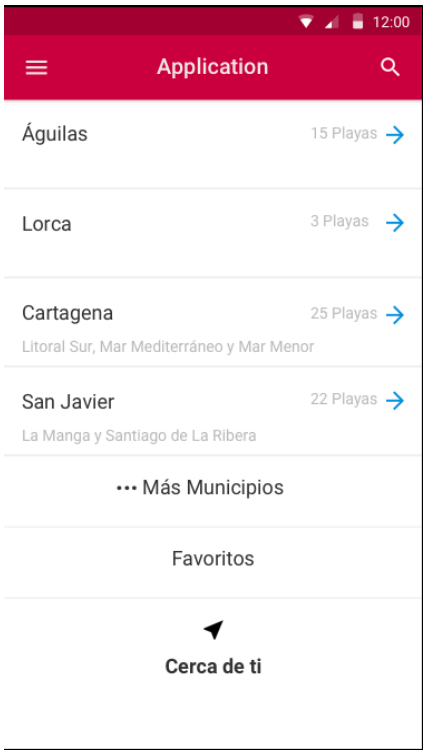

FIGURA 3-4 DISEÑO DE LA PANTALLA DE INICIO DE LA APLICACIÓN

En la figura anterior se muestra el diseño de la ventana principal, donde se ve una lista con algunos municipios y luego “Más Municipios”, el propósito de poner algunos municipios es para que el usuario sepa de antemano como va a ser la lista de Municipios y la información que representa, ya que es la entrada principal para una búsqueda detallada filtrando los datos. Además ya sabe que se informa sobre el número de Playas que tiene cada Municipio.

Después de la lista de Municipios está “Cerca de ti” con el icono distintivo de la geolocalización.

En la parte superior se encuentra en la parte derecha el icono de búsqueda, mientras que a la derecha está el icono del menú. El menú y la búsqueda van a ser accesibles desde todas las ventanas de navegación, por lo tanto en menú debería tener un enlace a la lista de Municipios y geolocalización.

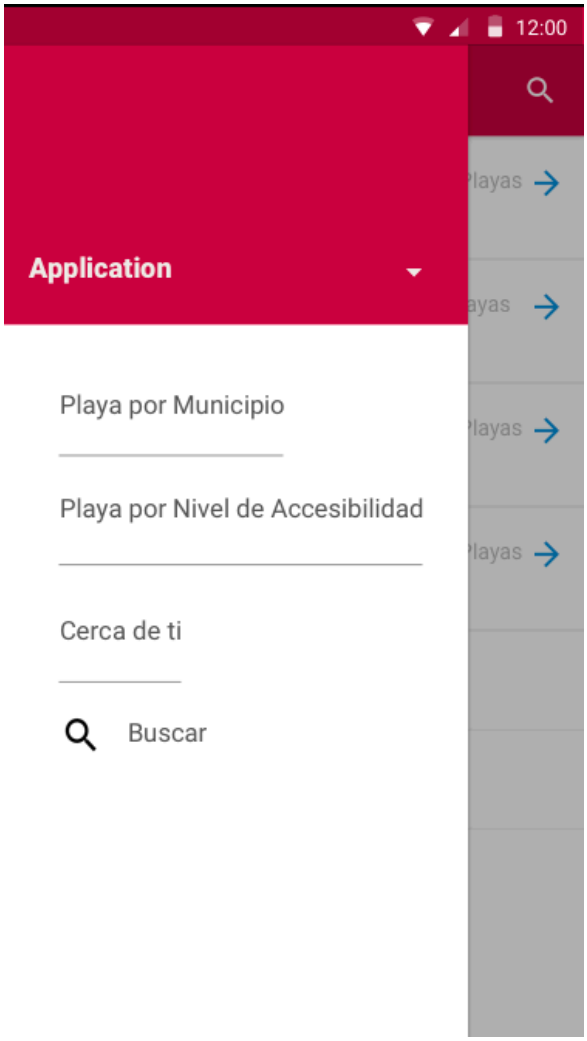

FIGURA 3-5 DISEÑO DEL MENÚ

Es importante tener en cuenta que los diseños suelen ser orientativos, y que en ocasiones pueden verse alterados a la hora del desarrollo. En general, esto suele ser debido al coste de desarrollo de algún componente o funcionalidad. En otras ocasiones puede ser que las herramientas usadas no lo soportan, por lo tanto se busca una alternativa que aporte la misma funcionalidad deseada. En el caso de la ventana principal se ve un elemento en la lista “Favoritos”, es un elemento que se ha agregado para dejar claro que cualquier elemento que se añade en la lista debe de ser antes de la

geolocalización, además esta última debe tener un alto mayor que el resto para destacar.

3.3 Base de datos

Los terminales con Sistema Operativo Android usan por defecto el sistema de gestión de bases de datos SQLite, donde podemos almacenar los datos necesarios para el uso de la aplicación sin la necesidad de conexión a la red, las bases de datos se caracterizan por estructurar los datos permitiendo el acceso o modificación de los datos de una forma más sencilla y controlada.

La mejor forma para estructurar la base de datos relacional es usar el Diagrama Modelo entidad-relación, para conseguirlo se ha hecho uso de la herramienta MySQLWorkbench.

Partiendo de los requerimientos de la aplicación el diagrama del Modelo entidad-relación en la figura 3-6.

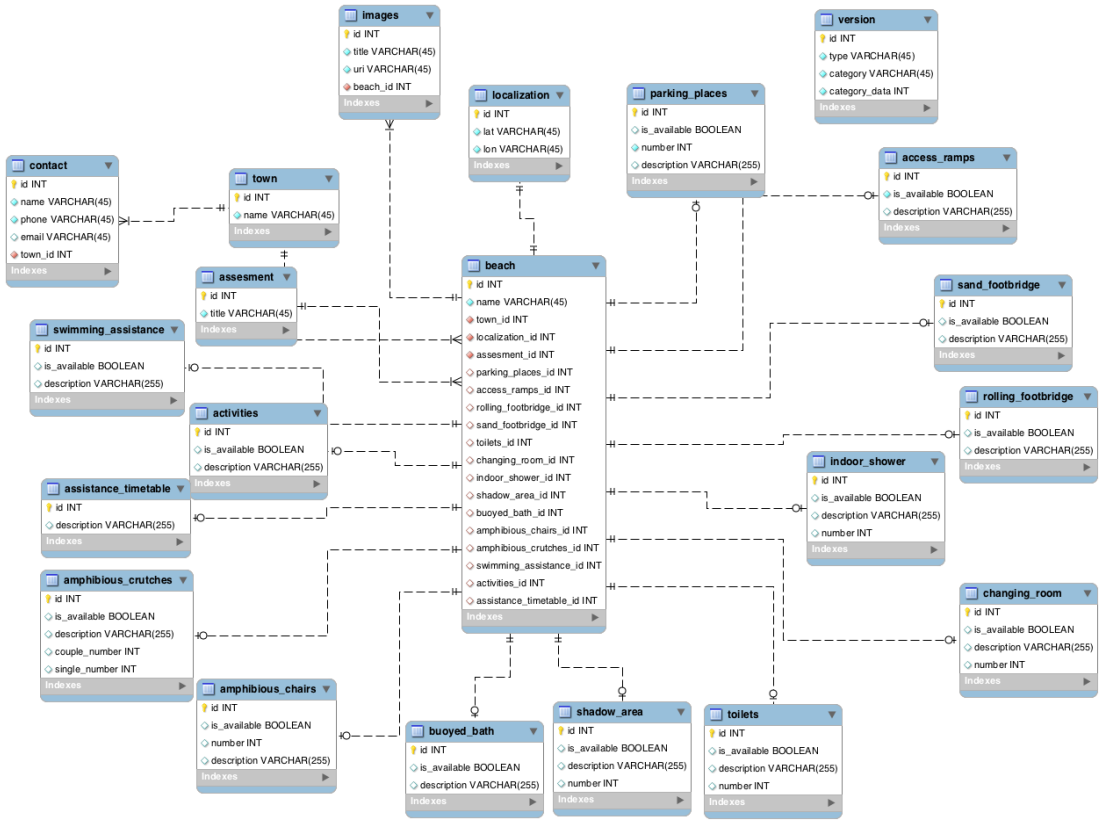

FIGURA 3-6 DIAGRAMA DEL MODELO ENTIDAD-RELACIÓN

Cada playa es una entidad y tiene una serie de datos que pueden ser atributos (por ejemplo, su nombre) u otras entidades (por ejemplo, el Municipio al que pertenece), donde cada entidad se define en otra tabla y se relaciona con la entidad de la playa por

una clave externa que apunta al identificador de la entidad. Para facilitar la lectura del Modelo, Las claves externas se componen por el nombre de la tabla más "_id" por ejemplo "town_id".

Cada Playa (tabla beach) debe tener un identificador único (id), donde también va a ser su clave primaria, un nombre (name), los datos del Municipio al que pertenece (town), localización geográfica (localization), valoración general relacionada con el nivel de accesibilidad para personas con discapacidad física (assesment) y una serie de instalaciones o servicios.

La relación entre la entidad de la Playa tanto con las instalaciones y los servicios como su localización geográfica es de 1 a 1, dado que son características relacionadas únicamente con la playa, mientras la relación con el Municipio, valoración general es de n a 1, por el hecho de que varias playas pueden pertenecer al mismo municipio, y tenga el mismo nivel de accesibilidad que otras playas.

Por otra parte, cada Municipio tiene 1 o varias oficinas de información, por lo que la relación entre town y contact es de 1 a n, y por ultimo cada playa puede tener varias imágenes, por lo tanto, la relación entre beach e images es de 1 a n.

En el diagrama existe otra tabla que no presenta ninguna relación con el resto, es la de versión, que se utiliza para el control de versiones.

3.4 Estructura del proyecto

El esqueleto para el desarrollo de la aplicación se basa en el que se ofrece por defecto para las aplicaciones en React Native, siendo este último ideal para las aplicaciones tanto de sistemas Android como IOS, donde el segundo no es objetivo de este proyecto.

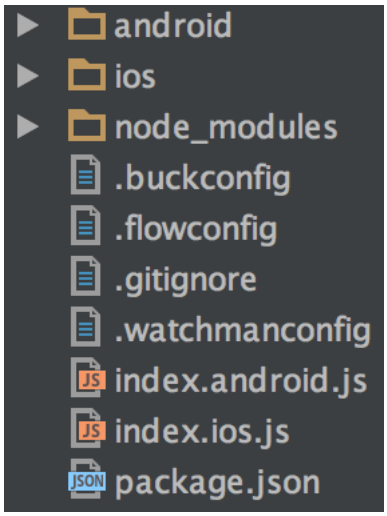

FIGURA 3-7 ESTRUCTURA DE ARCHIVOS POR DEFECTO DE REACT NATIVE

El fichero index.android.js es el primero que se ejecuta y es el encargado de llamar a otros componentes si surge el caso. Para facilitar una posible adaptación futura de la aplicación para dispositivos IOS, se ha creado un fichero main.js que acoge las funcionalidades comunes, por lo tanto, para su uso en IOS solo implica mover las funcionalidades específicas para Android al fichero index.android.js y hacer su equivalente para IOS en index.ios.js sin tener que crear otra aplicación totalmente nueva. Tanto el index.android.js como index.ios.js únicamente llaman al main.js siendo éste el encargado de gestionar la aplicación.

El fichero main.js es el encargado de registrar la aplicación, gestión de la navegación, y mostrar los componentes visuales comunes que en este caso son la barra de navegación y el menú, siendo todo el contenido central dependiente de la navegación.

La gestión de la navegación se hace mediante llamadas al componente navigator que a su vez llama a la función renderScene donde identifica el identificador de la ruta y llama a la pantalla que se pinta en el contenido central, siendo la pantalla inicio la mostrada defecto.

```
renderScene(route, navigator) {
  let sceneView;
  switch (route.id) {
    case 'HomeScreen':
      sceneView = (<HomeScreen navigator={navigator} route={route} data={townList} />);
      break;
    case 'TownScreen':
      sceneView = (<TownScreen navigator={navigator} route={route} data={townList} />);
      break;
    case 'LocationScreen':
      sceneView = (<LocationScreen navigator={navigator} route={route} data={beachList} />);
      break;
    case 'BeachScreen':
      sceneView = (<BeachScreen navigator={navigator} route={route} data={beachList} />);
      break;
    case 'BeachDataScreen':
      sceneView = (<BeachDataScreen navigator={navigator} route={route} />);
      break;
    case 'SearchScreen':
      sceneView = (<SearchScreen navigator={navigator} route={route} data={beachList} />);
      break;
    default:
      sceneView = (<HomeScreen navigator={navigator} route={route} />);
  }

  return sceneView;
}
```

FIGURA 3-8 GESTIÓN DE NAVEGACIÓN EN MAIN.JS

En la figura anterior se ve la función renderScene, donde para cada route.id se devuelve una vista en forma de código JSX, cada vista en este proyecto se denomina un Screen y se encuentran en la carpeta screens.

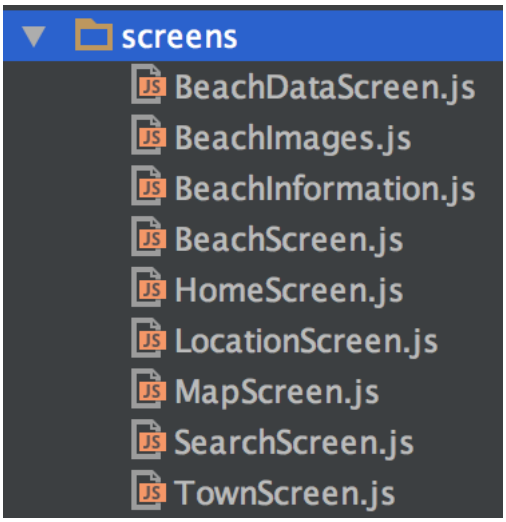

FIGURA 3-9 LA CARPETA SCREENS CON TODAS LAS VISTAS

A cada pantalla que es llamada desde el renderScene se le pasan dos parámetros:

1. Navigator.
2. Data.

Navigator

El navigator que tiene como objetivo facilitar la llamada al componente navigator en el main.js desde cada pantalla.

Data

En data están los datos requeridos por la pantalla. Estos datos se consiguen desde la base de datos donde estarán disponibles al inicio de la aplicación, de esta forma se evita llamar a la base de datos cada vez que se quiere mostrar una pantalla por lo tanto se evitan muchas llamadas a los hilos de Android que ralentizan la navegación y pueden provocar errores de funcionamiento.

Aparte de las pantallas (screens) existen otros componentes visuales que se encuentran en la carpeta componentes.

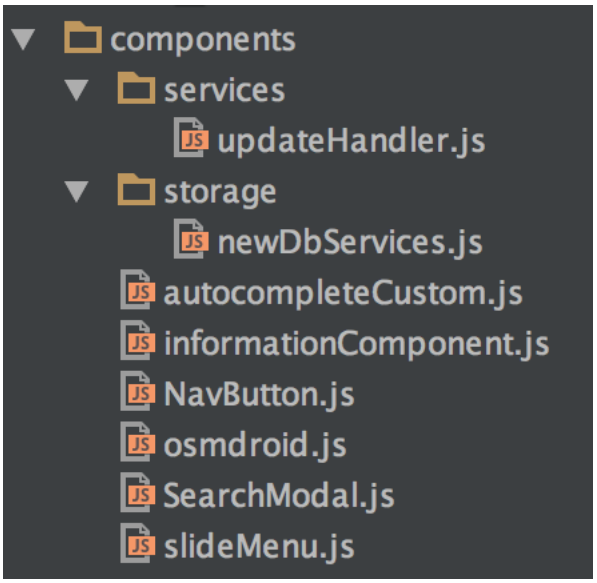

FIGURA 3-10 CARPETA COMPONENTS

Estos componentes visuales se consideran independientes de las pantallas, por lo tanto, es conveniente externalizarlos para poder reutilizarles. Además de los componentes visuales, la carpeta components tiene dos ficheros:

1. updateHandler.js
2. newDbServices.js

updateHandler.js es el encargado de buscar las actualizaciones, descargar los nuevos datos y guardarlos en la base de datos, mientras que newDbServices.js es donde están implementadas las funciones para leer, crear, modificar o borrar datos de la base de datos.

La estructura final de los archivos es la siguiente:

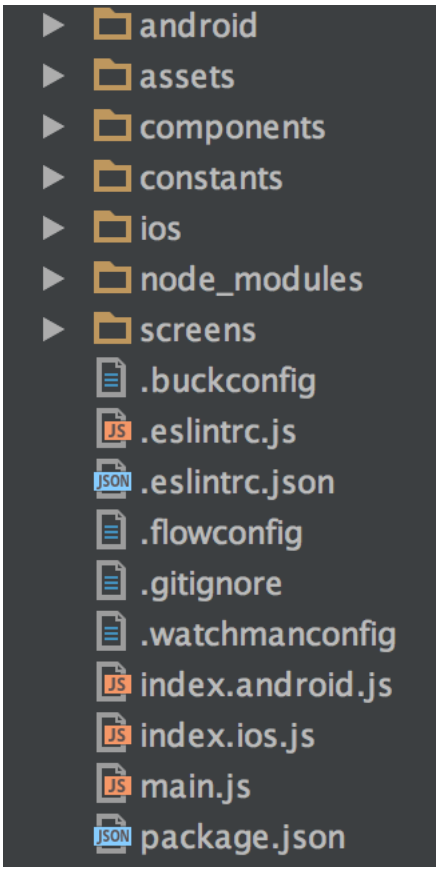

FIGURA 3-11 ESTRUCTURA DE LOS ARCHIVOS DE LA APLICACIÓN

Aunque React Native proporciona la mayoría de los componentes que interactúan con el código nativo de Android, pero faltan todavía algunas, donde en esta aplicación han sido necesarias, como por ejemplo el acceso a la base de datos SQLite.

React Native permite implementar funcionalidades nativas y acceder a ellas desde JavaScript mediante el componente nativeModules.

3.4.1 Módulos Nativos

En la aplicación se ha implementado un módulo nativo para la visualización del Mapa, ya que React Native no proporciona un módulo para mapas sin conexión a la red.

Para ello se ha usado la librería OSMDroid para la visualización de mapas open Street map en Android, para conectar el OSMDroid con JavaScript, se han creado dos clases Java RNOSMDroidManager.java y RNOSMDroidPackage.java, donde el Manager es el encargado de llamar al OSMDroid e implementa los atributos para que se comunique con JavaScript, mientras que el package es para declararlo en los paquetes necesarios para el funcionamiento de la aplicación, al final se llamar al módulo nativo en JavaScript pasándole los parámetros necesarios como las coordinadas de latitud y longitud.

Para la integración con la base de datos, se ha usado el módulo nativo react-native-sqlite-storage creada por Andrzej Porebski y distribuida en Github con código abierto y bajo la licencia MIT, el cual se ha añadido como dependencia en el fichero build.gradle y declararlo en el MainApplication.java.

3.5 Datos y Actualizaciones

Teniendo en cuenta que los datos de cada Playa se actualizan en un periodo de tiempo largo, y la cantidad de los datos es significativa, entonces se ha optado por distribuir los datos del año 2016 con la aplicación, de esta forma al descargar la aplicación desde Google Play ya tendrá la mayoría de información sobre la accesibilidad en las Playas de la región.

Los datos iniciales están encapsulados en una base de datos SQLite en forma de fichero que se guarda en la carpeta Assets de Android con la extensión ".db", donde posteriormente se llama desde la clase newDbServices.js para su manipulación.

Las actualizaciones se verifican en el inicio de la aplicación mediante llamadas al servidor, donde antes de llamar al servidor se verifica si el dispositivo está conectado a internet, en el caso contrario se llama a una función que escucha el evento que avisa cuando se conecta el dispositivo a internet. Una vez conectado a la red se descargan los datos almacenados en la tabla de versiones de la base de datos alojada en el servidor.

Con cada cambio en la base de datos del servidor se crea una nueva versión con el dato cambiado, por ejemplo, si se modifica una playa, se añade una nueva versión de tipo "update", categoría "beach" y el identificador de la playa modificada en el campo "category_data".

En la base de datos en el dispositivo cliente está la tabla de versiones, donde se compara con la base de datos del servidor. Si la nueva versión no está en el dispositivo, se descargan los datos nuevos o modificados de cada versión y luego se almacenan en la base de datos del dispositivo cliente. Si el tipo de cambio es "delete", entonces se borra el dato almacenado en el dispositivo cliente. Al finalizar la actualización en el dispositivo cliente se añade la nueva versión a la base de datos, de esta forma estará igual con la del servidor.

4 Manual de usuario

4.1 Instalación de la aplicación

Las aplicaciones en Android se pueden instalar de tres formas:

1. Google Play
2. Archivo APK
3. Mediante USB

4.1.1 Google Play

Google Play es el contenedor de aplicaciones de google para dispositivos Android, donde se permite descargar las aplicaciones y instalarlas de forma segura en el dispositivo. Cuando la aplicación se publica en Google Play solo será necesario buscarla y seleccionar el botón instalar.

4.1.2 Archivo APK

Los APKs son paquetes ejecutables en el sistema Android, para generar el APK de una aplicación en React Native hay que generar un keystore y configurar el acceso al fichero en build.gradle, luego acceder desde consola en la carpeta /android del proyecto y lanzar el comando ./gradlew assembleRelease. Cuando ya tenemos el fichero apk de la aplicación, lo copiamos en el almacenamiento del dispositivo y ejecutarlo, pero antes de ejecutarlo hay que habilitar que acepte orígenes desconocidos en la configuración del dispositivo.

4.1.3 Mediante USB

React Native proporciona un comando para ejecutar la aplicación en los dispositivos Android con solo conectarlos al ordenador mediante USB. El comando es ***react-native run-android***, pero esta opción solo instala la versión de depuración de la aplicación, además en algunos tipos de dispositivos requieren la instalación de controladores proporcionados por el fabricante.

4.2 Uso de la aplicación

Al ejecutar la aplicación se muestra la pantalla de inicio que contiene una serie de enlaces. Cada enlace ofrece una opción para encontrar una playa.

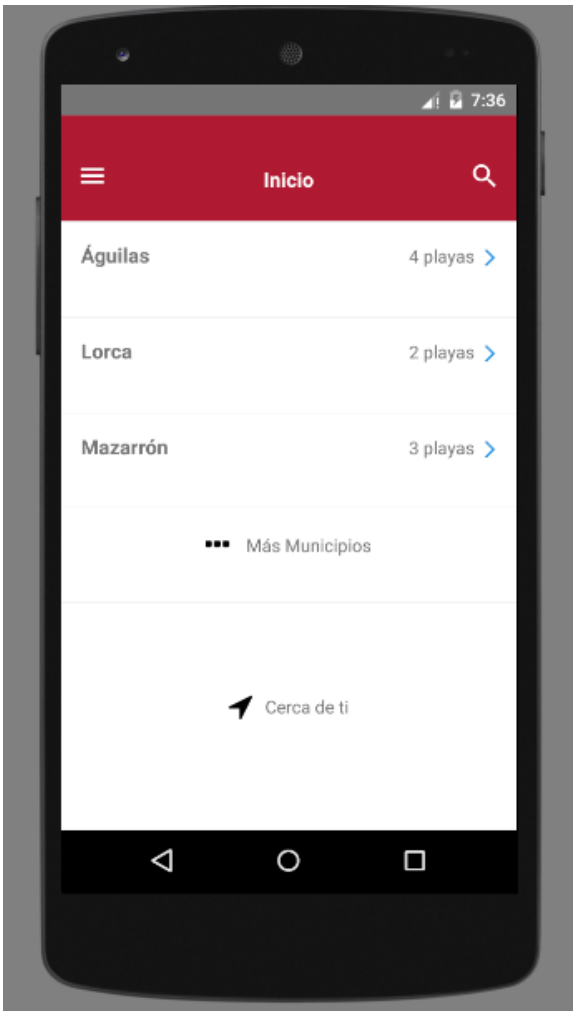

FIGURA 4-1 PANTALLA DE INICIO DE LA APLICACIÓN

4.2.1 Acceso por lista de Municipios

Al seleccionar la lista de Municipios se muestra cada uno de los municipios disponibles con el número de playas que contiene, y en la barra de navegación se muestra el nombre del Municipio seleccionado.

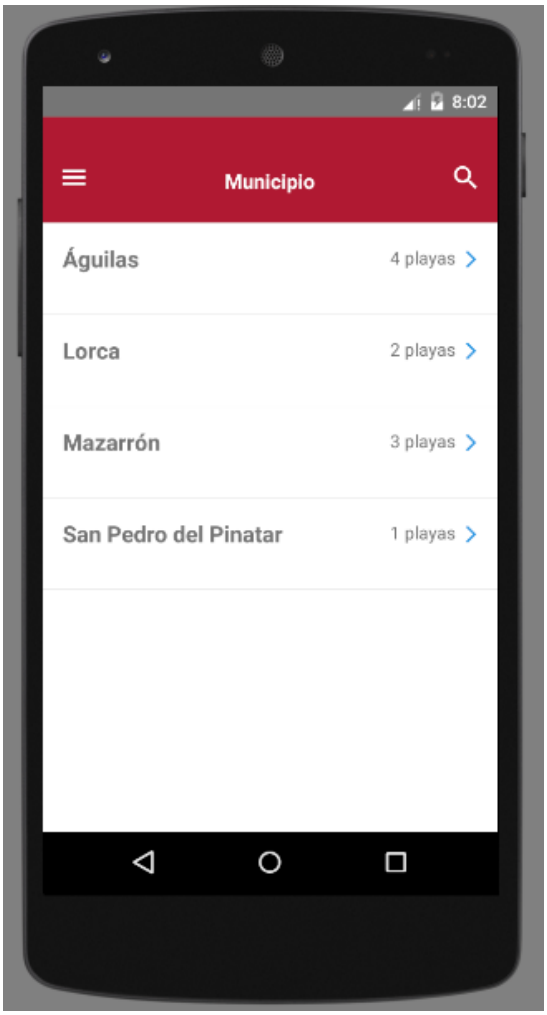

FIGURA 4-2 LISTA DE MUNICIPIOS

Cuando se selecciona un Municipio se muestra la lista de las playas que contiene, a su vez si se selecciona una playa se muestran la información de la playa.

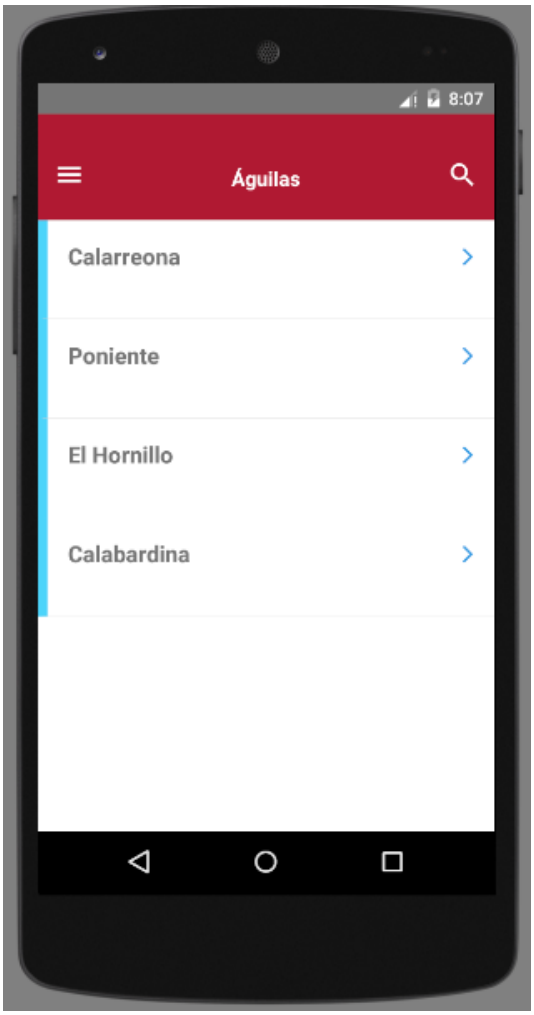

FIGURA 4-3 LISTA DE PLAYAS DE UN MUNICIPIO

4.2.2 Acceso por búsqueda

Para acceder a la búsqueda se tiene que seleccionar en la barra de navegación el icono de la Lupa. Una vez abierta la pantalla de la búsqueda se selecciona por defecto el campo de texto, por lo tanto, se abre el teclado. Conforme se empieza a escribir se van mostrando los resultados en forma de lista. El icono con forma de flecha a la izquierda permite cerrar la búsqueda, mientras que la cruz a la derecha limpia el campo de texto y la lista de resultados.

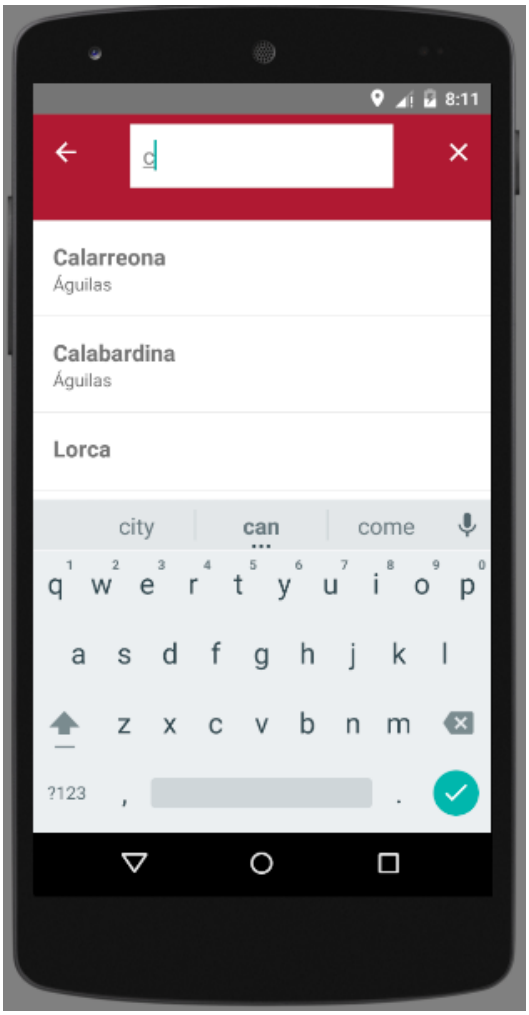

FIGURA 4-4 PANTALLA DE BÚSQUEDA

Los resultados pueden ser playas o municipios, por lo tanto, si se selecciona un municipio se muestra la lista de playas relacionadas, mientras si selecciona una playa se accederá directamente a la pantalla de información.

4.2.3 Acceso por Geolocalización

En la pantalla de inicio está el enlace para localizar las playas más próximas a la localización del usuario. Al seleccionar la opción de geolocalización se muestra la lista de las playas ordenadas por proximidad, y cada una tiene la distancia aproximada desde el punto en el que se encuentra el dispositivo.

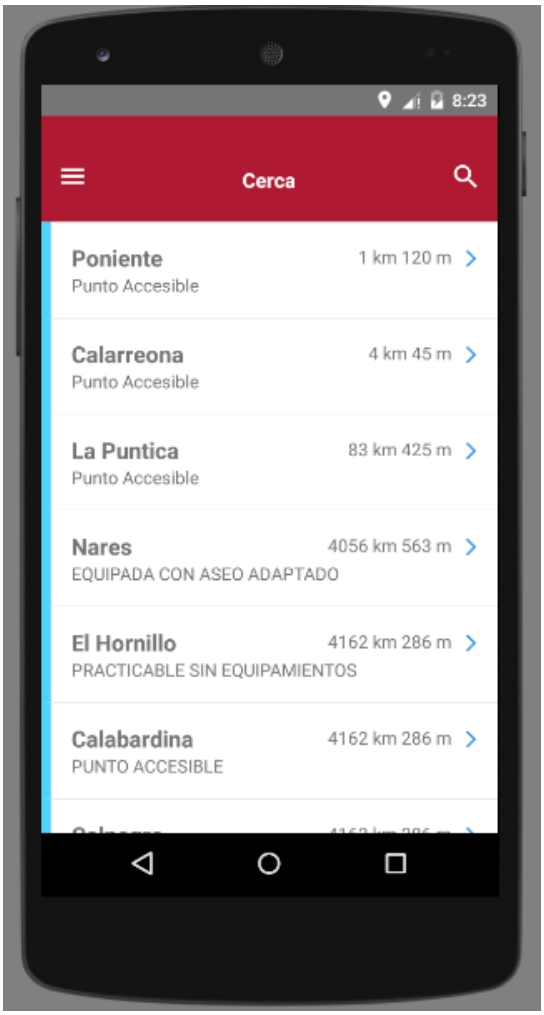

FIGURA 4-5 PANTALLA DE LA GEOLOCALIZACIÓN

4.2.4 Visualización de los datos de una Playa

Cuando se selecciona una playa se muestra la localización y la información de accesibilidad de la Play seleccionada.

La localización se muestra en un mapa de la región con un punto señalando la localización. Debajo del mapa se encuentran dos pestañas, la de información que es la seleccionada por defecto y la de Imágenes. La de información es la encargada de mostrar los datos de accesibilidad de la playa.

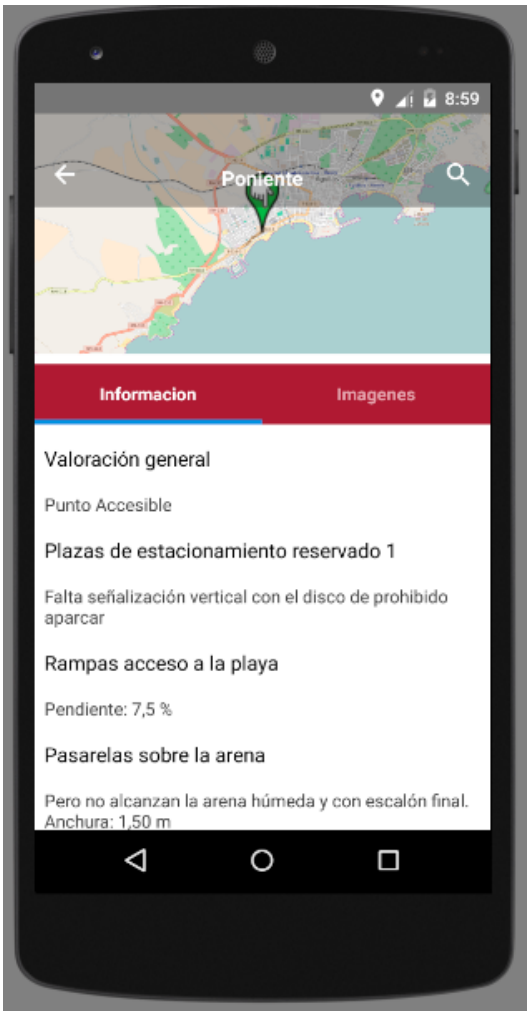

FIGURA 4-6 PANTALLA DE INFORMACIÓN DE LA PLAYA

La pestaña de Imágenes es la encargada de mostrar una lista de las imágenes de la Playa.

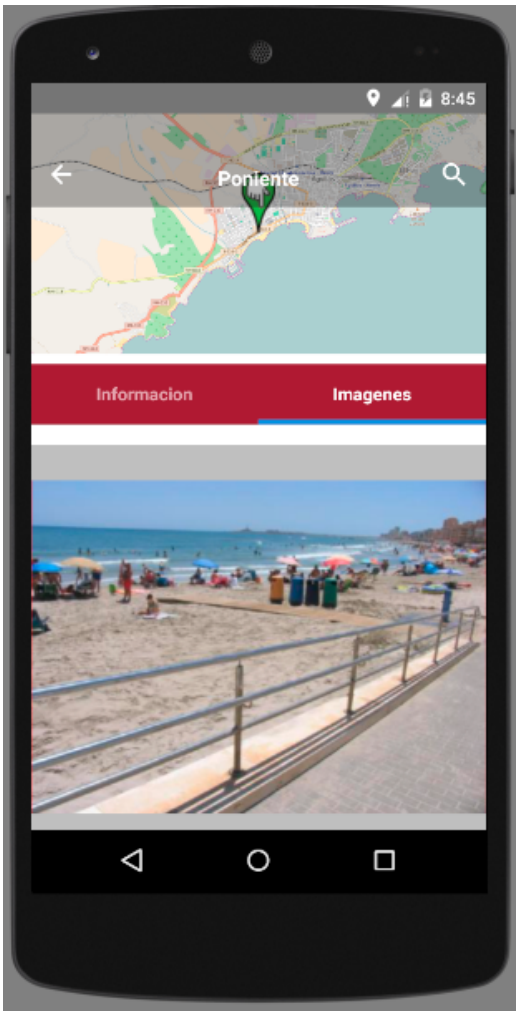

FIGURA 4-7 PANTALLA DE LAS IMÁGENES DE LA PLAYA

4.2.5 Menú

El menú de la aplicación tiene como objetivo facilitar la navegación por las diferentes opciones de la aplicación, por lo tanto, se encuentra en todas las pantallas excepto la de información de la Playa, ya que existen varias formas para acceder a la información y se debe dejar la posibilidad de volver a la opción seleccionada.

Para acceder al menú se tiene que seleccionar el icono que se encuentra en la parte izquierda de la barra de navegación.

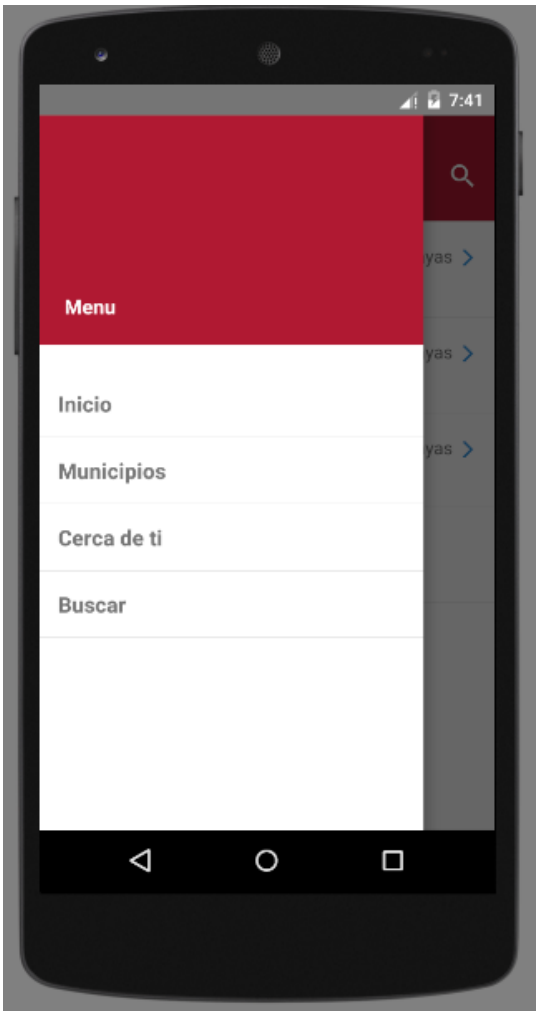

FIGURA 4-8 MENÚ DE LA APLICACIÓN

El menú contiene los enlaces para volver a la pantalla de Inicio, lista de Municipios, geolocalización y la búsqueda.

5 Conclusiones y líneas futuras

5.1 Conclusiones

En el proyecto se ha desarrollado una aplicación que contiene información sobre la accesibilidad para personas con discapacidad física en las playas de la región de Murcia. Se ha conseguido una aplicación con las siguientes características:

• Opera sin necesidad de descargar datos de la red.

• Ofrece varias opciones para encontrar de forma más rápida y sencilla la información que el usuario esté buscando.

• Los datos han sido almacenados en una base de datos estructurada permitiendo el acceso a ellos de forma rápida y controlada.

• Se ha implementado una aplicación web para la administración de las actualizaciones en el lado del servidor. De esta forma, la aplicación va a estar actualizada en todo momento ante cualquier tipo de cambio en los datos de la accesibilidad a las playas de la región.

• Permite la localización del usuario por GPS y muestra información sobre las playas próximas al usuario.

Además, la aplicación se ha desarrollado respectando las guías de buena práctica. De esta forma, se permite un mantenimiento fácil de la aplicación y fácil adaptación a los dispositivos basados en IOS.

5.2 Líneas futuras

Como se ha especificado en el último punto de la conclusión. La aplicación se ha desarrollado con una arquitectura que permite con un menor esfuerzo la adaptación a otros sistemas aparte de Android, y con añadir filtrado por provincia para que pueda ofrecer información sobre la accesibilidad en las playas a nivel Nacional.

En la aplicación se ha usado un mapa gratuito para facilitar la localización de las playas, pero tiene un funcionamiento limitado, mientras que google maps o mapbox ofrecen muchas funcionalidades que permiten la localización y navegación de una forma mucho más óptima. Además, ofrecen mapas vectoriales que no dependen de la resolución de cada pantalla, permitiendo una visualización con más detalles del mapa independientemente del tipo de dispositivo. Obviamente todas estas mejoras que ofrecen no son gratuitas, pero en muchos casos merece la pena.

Para la actualización de los datos, se puede crear una aplicación móvil para la administración que puede gestionar las estas actualizaciones en situ, y permitir al usuario señalar incidencias para que lo tengan en cuenta el resto de los usuarios.

6 Bibliografía

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

https://facebook.github.io/react/

https://developer.android.com/index.html

https://babeljs.io/

https://github.com/andpor/react-native-sqlite-storage

https://github.com/oblador/react-native-vector-icons

https://github.com/osmdroid/osmdroid/wiki

https://github.com/mevdschee/php-crud-api

https://en.wikipedia.org