Diseño de Aplicaciones Móviles

Andrés Muñoz Órdenes
andmunoz@dcc.uchile.cl

dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Menú de Hoy

- Revisión Actividad 7
- Más sobre React Native
- Actividad 8: Ejemplo de uso de API

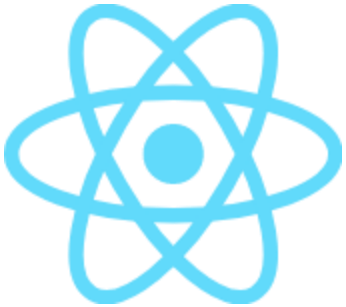
Revisión Actividades

5 min

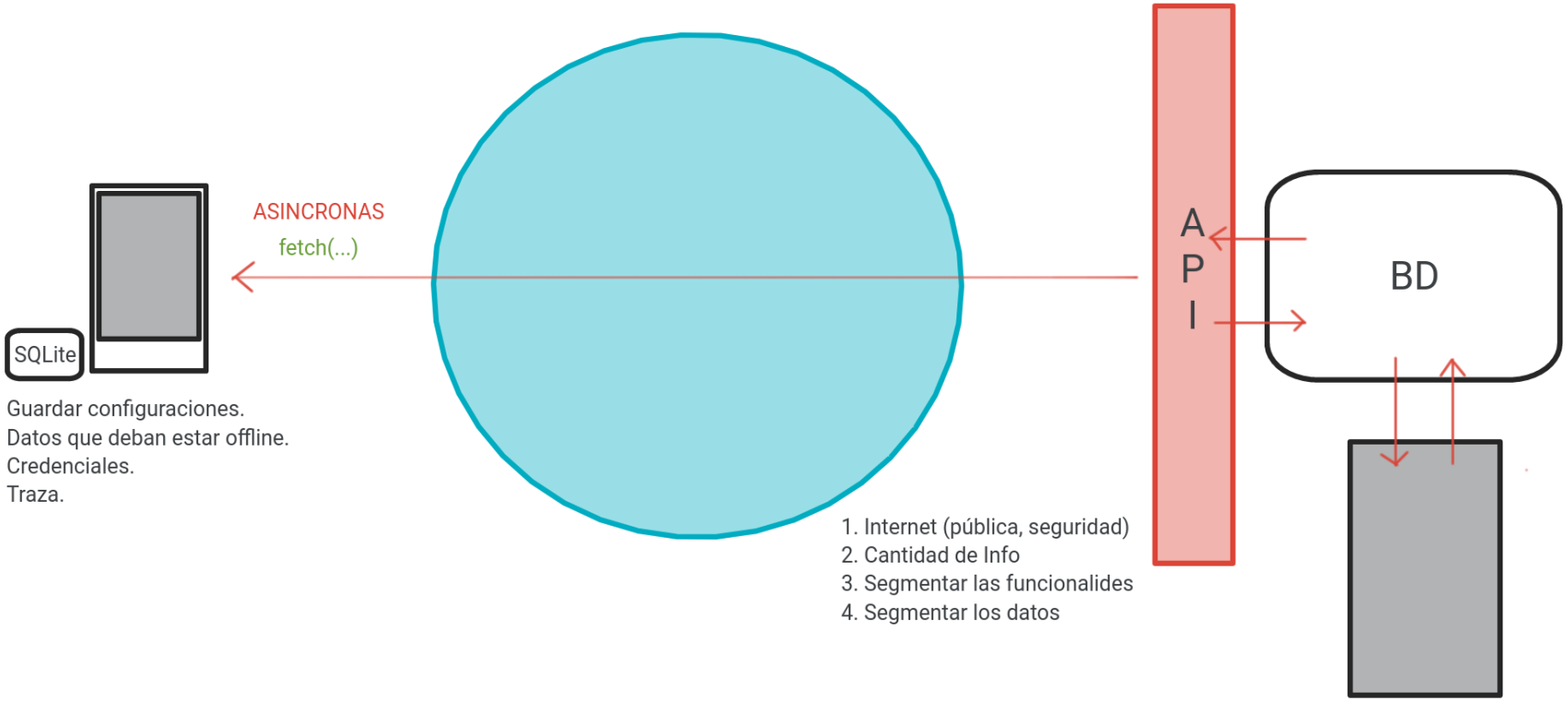
Revisión Actividad 7

- Utilizando lo que ya hemos construido, debemos comenzar a confeccionar dos elementos:
  - La navegación (utilizando las transiciones)
  - El poblamiento de datos en las listas y pantallas (dinámicamente)
- Para el segundo punto, por ahora, podemos usar mapas javascript con datos que podemos cambiar a mano, ya que la próxima sesión utilizaremos un servicio para su poblamiento.

React Native



Llamadas Asíncronas desde la App



Uso de Fetch para llamar a un Servicio

fetch() es una función que permite utilizar la llamada asíncrona desde la aplicación. Es importante considerar que:

- Cada llamada a un servicio debe ser asíncrona, porque ésta requiere de que el dispositivo no se bloquee mientras obtiene la información.
- Las llamadas asíncronas utilizan unas funciones especiales de React que permiten al sistema operativo activar el rendering cuando el servicio termine de interactuar con la app.
- Éstas funciones son **useState()** y **useEffect()**.

Uso de Fetch para llamar a un Servicio

useState() permite configurar un "estado" a una variable, la cual puede cambiar de manera asíncrona:

```
const [isLoading, setLoading] = useState(true);
```

- En ejemplo se crea una constante ***isLoading*** que tomará el valor de estado ***true*** y se define una función ***setLoading()*** que permite cambiar el estado a uno específico.

Uso de Fetch para llamar a un Servicio

useEffect() es un *hook* que le indica al programa que el componente debe hacer algo después de renderizarse (*componentDid...*):

```
useEffect(() => { document.title = 'Hola'; }, []);
```

- En ejemplo prepara un cambio en el título de la pantalla una vez que es renderizada la primera vez o cada vez que se actualiza de alguna forma.

dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Uso de Fetch para llamar a un Servicio

De esta manera, usaremos los estados y los efectos para hacer el llamado de manera asíncrona:

```
const [isLoading, setLoading] = useState(true);
const [data, setData] = useState([]);

useEffect(() => {
  fetch('...', params)
    .then((response) => response.json())
    .then((json) => setData(json))
    .catch((error) => console.error(error))
    .finally(() => setLoading(false));
}, []);
```

Ejemplo

En este ejemplo, se llama al servicio de la CMF para obtener información específica.

Ojo que en este caso se utiliza en los parámetros la autenticación básica del servicio (user/pass) y además se indica que el resultado es de tipo JSON.

```
const [isLoading, setLoading] = useState(true);
const [data, setData] = useState([]);

useEffect(() => {
  fetch('https://www.cmfchile.cl/sitio/api/fmide/identificacion/', {
    headers: new Headers({
      'Authorization': 'Basic ' + base64.encode("USR_PRUEBA:API.prueba.2020"),
      'Content-Type': 'application/json'
    })
  })
  .then((response) => response.json())
  .then((json) => setData(json))
  .catch((error) => console.error(error))
  .finally(() => setLoading(false)
  );
}, []);
```

Ejemplo


```jsx
import React, { useEffect, useState } from 'react';
import { FlatList, Text, View } from 'react-native';

const base64 = require('base-64');

const App = () => {
  const [isLoading, setLoading] = useState(true);
  const [data, setData] = useState([]);

  useEffect(() => {
    fetch('https://www.cmfchile.cl/sitio/api/fmide/identificacion/', {
      headers: new Headers({
        'Authorization': 'Basic ' + base64.encode("USR_PRUEBA:API.prueba.2020"),
        'Content-Type': 'application/json'
      })
    })
    .then((response) => response.json())
    .then((json) => setData(json))
    .catch((error) => console.error(error))
    .finally(() => setLoading(false))
    );
  }, []);
  console.log(data);

  return (
    <View style={{ flex: 1, padding: 24 }}>
      <Text style={{ fontSize: 18, color: 'green', textAlign: 'center'}}>Regiones de Chile</Text>
      {isLoading ? <Text>Loading...</Text> :
      ( <View style={{ flex: 1, flexDirection: 'column', justifyContent:  'space-between'}}>
          <FlatList
            data={data.Data}
            keyExtractor={({ id }, NUMERO) => id}
            renderItem={({ item }) => (
              <Text>{item.NUMERO}. {item.NOMBRE_CORTO} ({item.NOMBRE_FONDO})</Text>
            )}
          />
        </View>
      )}
    </View>
  );
};

export default App;
```

40 min

Actividad 8: Desarrollo de Ejercicio

- Utilizando las API's Públicas de Chile que se mencionan en los links de interés, crear una aplicación que las consuma y que las muestre en pantalla.
- Para ello, debe considerar que el despliegue de la información lo haga en una FlatList tal como sale en el ejemplo visto.
- Lo revisaremos la próxima sesión.

Links de Interés

- Sitio Oficial de React Native (https://reactnative.dev/)
- Listado de API's Públicas en Chile (https://juanbrujo.github.io/listado-apis-publicas-en-chile/)
- Usando el Hook de Efecto (https://es.reactjs.org/docs/hooks-effect.html)

dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE