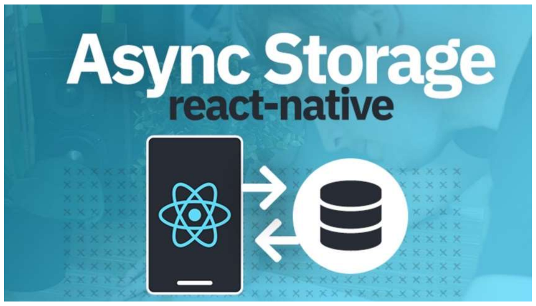
EIU
TRƯỜNG ĐẠI HỌC QUỐC TẾ MIỀN ĐÔNG
EASTERN INTERNATIONAL UNIVERSITY

Mobile Programming

Chapter 5: WORK WITH DATA

I. AsyncStorage

AsyncStorage asynchronous, unencrypted, persistent, key-value storage solution for your React Native application.
Multi-platform support
Data storage solution for **Android**, **iOS**, **Web**, **MacOS** and **Windows**.
Simple API
A handful of tools to simplify your storage flow. Easily save, read, merge and delete data at will!



I. AsyncStorage

Install

npm install @react-native-async-storage/async-storage

Import

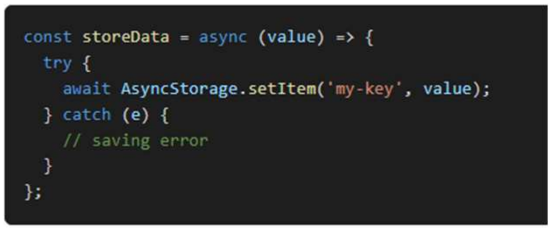
import AsyncStorage from '@react-native-async-storage/async-storage';

I. AsyncStorage

Storing data

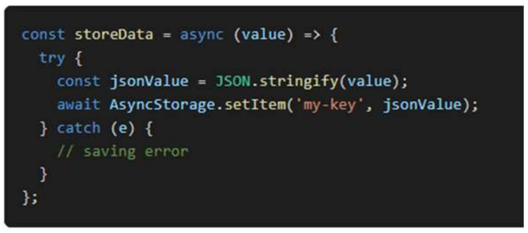
setItem() is used both to add new data item (when no data for given key exists), and to modify existing item (when previous data for given key exists).

Storing string value



```
const storeData = async (value) => {
  try {
    await AsyncStorage.setItem('my-key', value);
  } catch (e) {
    // saving error
  }
};
```

Storing object value



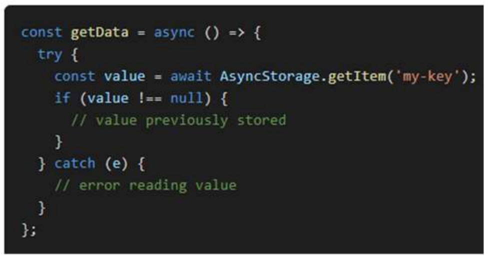
```
const storeData = async (value) => {
  try {
    const jsonValue = JSON.stringify(value);
    await AsyncStorage.setItem('my-key', jsonValue);
  } catch (e) {
    // saving error
  }
};
```

I. AsyncStorage

Reading data

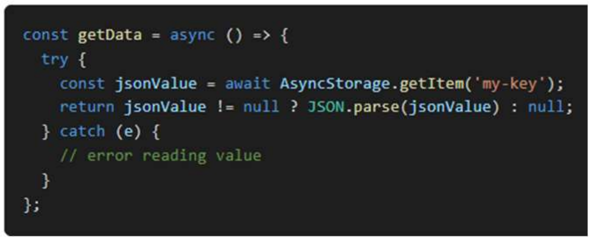
GetItem() returns a promise that either resolves to stored value when data is found for given key, or returns null otherwise.

Reading string value



```
const getData = async () => {
  try {
    const value = await AsyncStorage.getItem('my-key');
    if (value !== null) {
      // value previously stored
    }
  } catch (e) {
    // error reading value
  }
};
```

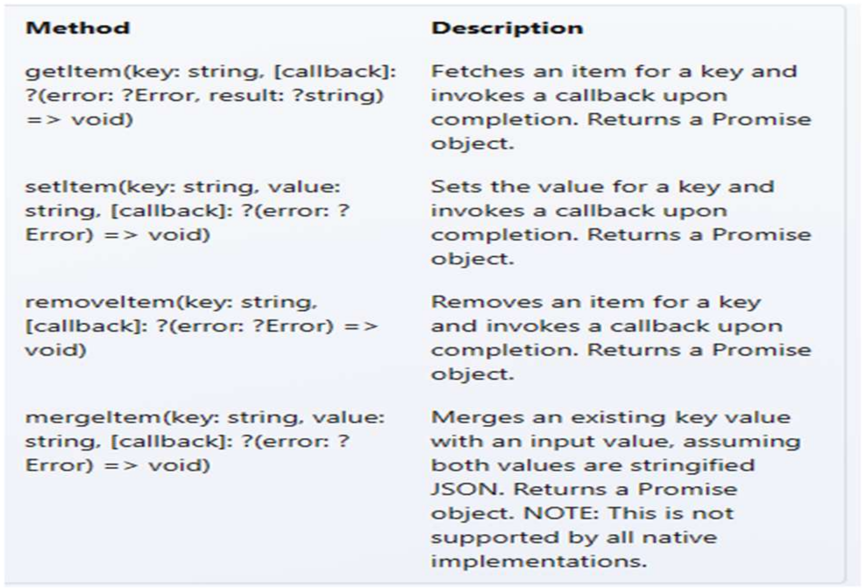
Reading object value



```
const getData = async () => {
  try {
    const jsonValue = await AsyncStorage.getItem('my-key');
    return jsonValue != null ? JSON.parse(jsonValue) : null;
  } catch (e) {
    // error reading value
  }
};
```

5

I. AsyncStorage

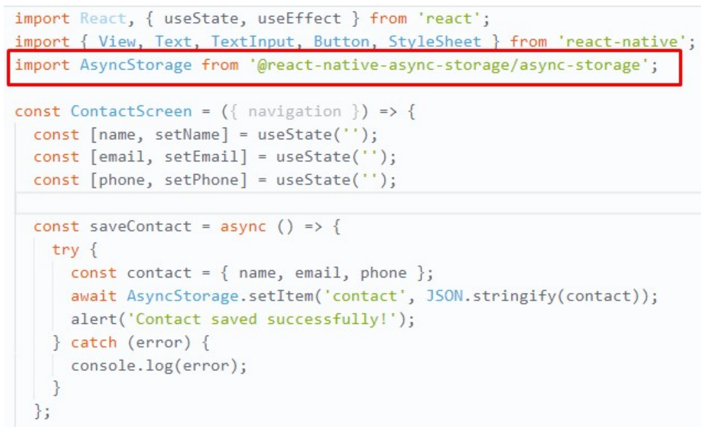


Method	Description
getItem(key: string, [callback]: ?(error: ?Error, result: ?string) => void)	Fetches an item for a key and invokes a callback upon completion. Returns a Promise object.
setItem(key: string, value: string, [callback]: ?(error: ?Error) => void)	Sets the value for a key and invokes a callback upon completion. Returns a Promise object.
removeItem(key: string, [callback]: ?(error: ?Error) => void)	Removes an item for a key and invokes a callback upon completion. Returns a Promise object.
mergeItem(key: string, value: string, [callback]: ?(error: ?Error) => void)	Merges an existing key value with an input value, assuming both values are stringified JSON. Returns a Promise object. NOTE: This is not supported by all native implementations.

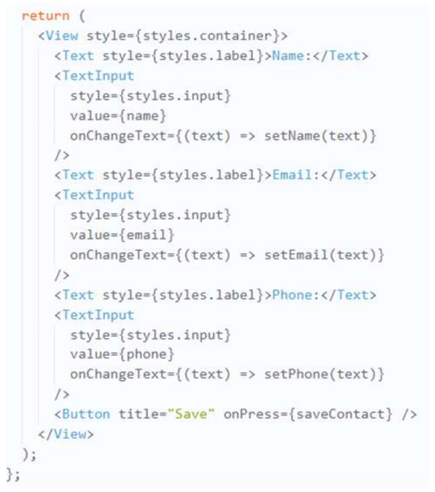
https://react-native-async-storage.github.io/async-storage/docs/api

6

I. AsyncStorage

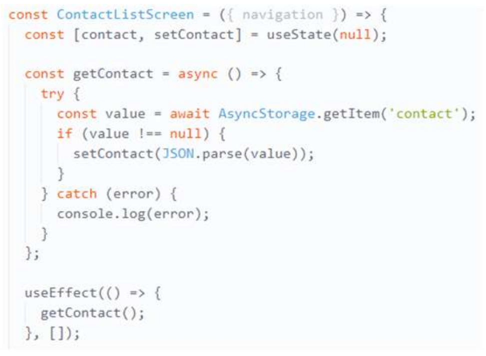


```
import React, { useState, useEffect } from 'react';
import { View, Text, TextInput, Button, StyleSheet } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const ContactScreen = ({ navigation }) => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [phone, setPhone] = useState('');

  const saveContact = async () => {
    try {
      const contact = { name, email, phone };
      await AsyncStorage.setItem('contact', JSON.stringify(contact));
      alert('Contact saved successfully!');
    } catch (error) {
      console.log(error);
    }
  };
```

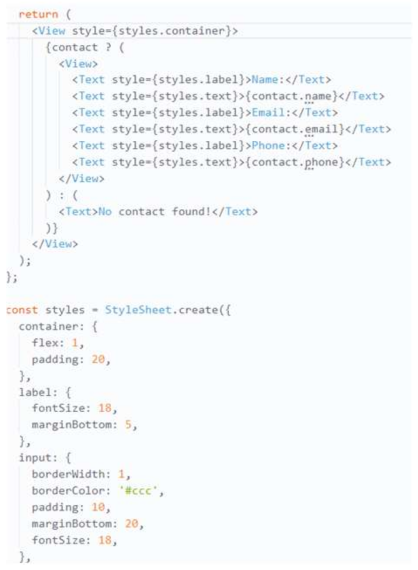


```
  return (
    <View style={styles.container}>
      <Text style={styles.label}>Name:</Text>
      <TextInput
        style={styles.input}
        value={name}
        onChangeText={(text) => setName(text)}
      />
      <Text style={styles.label}>Email:</Text>
      <TextInput
        style={styles.input}
        value={email}
        onChangeText={(text) => setEmail(text)}
      />
      <Text style={styles.label}>Phone:</Text>
      <TextInput
        style={styles.input}
        value={phone}
        onChangeText={(text) => setPhone(text)}
      />
      <Button title="Save" onPress={saveContact} />
    </View>
  );
};
```

I. AsyncStorage

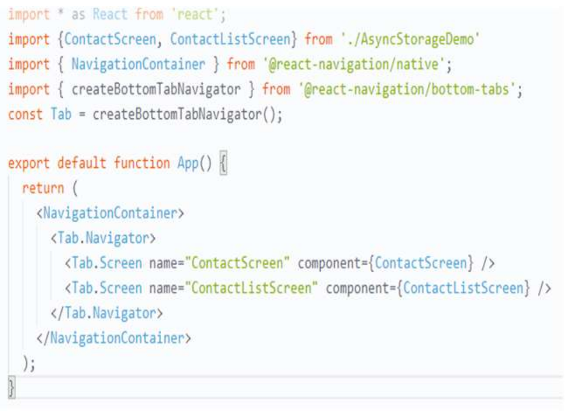


```
const ContactListScreen = ({ navigation }) => {
  const [contact, setContact] = useState(null);

  const getContact = async () => {
    try {
      const value = await AsyncStorage.getItem('contact');
      if (value !== null) {
        setContact(JSON.parse(value));
      }
    } catch (error) {
      console.log(error);
    }
  };

  useEffect(() => {
    getContact();
  }, []);
```



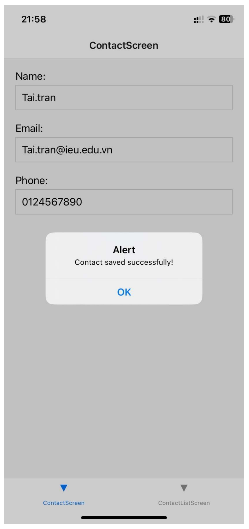
```
  return (
    <View style={styles.container}>
      {contact ? (
        <View>
          <Text style={styles.label}>Name:</Text>
          <Text style={styles.text}>{contact.name}</Text>
          <Text style={styles.label}>Email:</Text>
          <Text style={styles.text}>{contact.email}</Text>
          <Text style={styles.label}>Phone:</Text>
          <Text style={styles.text}>{contact.phone}</Text>
        </View>
      ) : (
        <Text>No contact found!</Text>
      )}
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
  },
  label: {
    fontSize: 18,
    marginBottom: 5,
  },
  input: {
    borderWidth: 1,
    borderColor: '#ccc',
    padding: 10,
    marginBottom: 20,
    fontSize: 18,
  },
```

I. AsyncStorage



```
import * as React from 'react';
import {ContactScreen, ContactListScreen} from './AsyncStorageDemo'
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="ContactScreen" component={ContactScreen} />
        <Tab.Screen name="ContactListScreen" component={ContactListScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

https://snack.expo.dev/@taitv/asyncstorage



21:58

ContactScreen

Name: Tai.tran

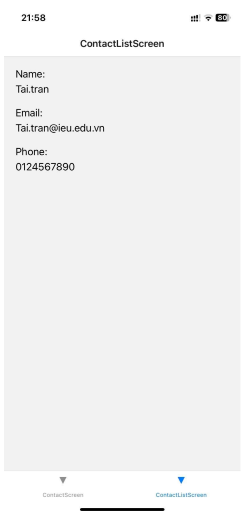
Email: Tai.tran@ieu.edu.vn

Phone: 0124567890

Alert

Contact saved successfully!

OK



21:58

ContactListScreen

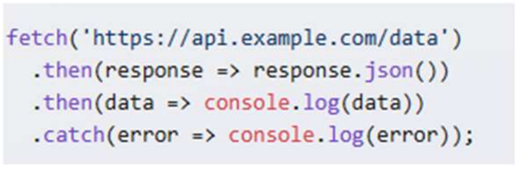
Name:
Tai.tran

Email:
Tai.tran@ieu.edu.vn

Phone:
0124567890

Promise

In React Native, Promise is an asynchronous processing mechanism that handles asynchronous operations such as fetching data from APIs. To use Promise in your application, you can use the then() and catch() functions to process the results returned from the promise.



```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log(error));
```

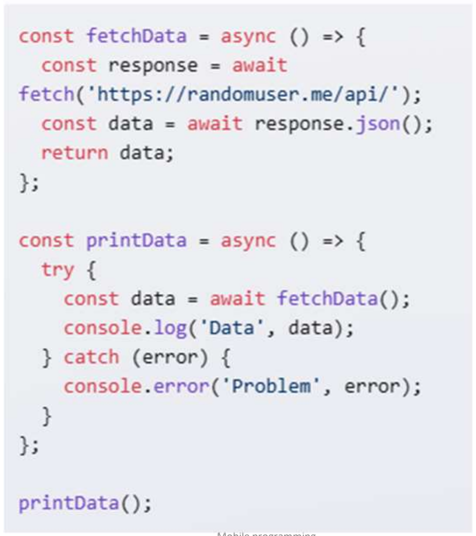
Async/Await

Async / Await is a feature of JavaScript that helps us work with asynchronous functions in a way that is more interesting and easier to understand. It is built on Promises and is compatible with all API based Promises

Async - declares an asynchronous function
Automatically transforms a regular function into a Promise.
When called to the async function, it processes everything and returns the result in its function.
Async allows the use of Await.

Await - pause the execution of async functions.
When placed before a promise, it will wait until the promise ends and returns the result.
Await only works with Promises, it doesn't work with callbacks.
Await can only be used inside async functions.

Async/Await



```
const fetchData = async () => {
  const response = await
fetch('https://randomuser.me/api/');
  const data = await response.json();
  return data;
};

const printData = async () => {
  try {
    const data = await fetchData();
    console.log('Data', data);
  } catch (error) {
    console.error('Problem', error);
  }
};

printData();
```

FETCH API

In React Native, you can use the Fetch API to make network requests. The Fetch API is similar to the XMLHttpRequest API and allows you to make GET and POST requests to a remote server or API.

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.log(error));
```

```
const fetchData = async () => {
  try {
    const response = await
fetch('https://api.example.com/data');
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.log(error);
  }
};

fetchData();
```

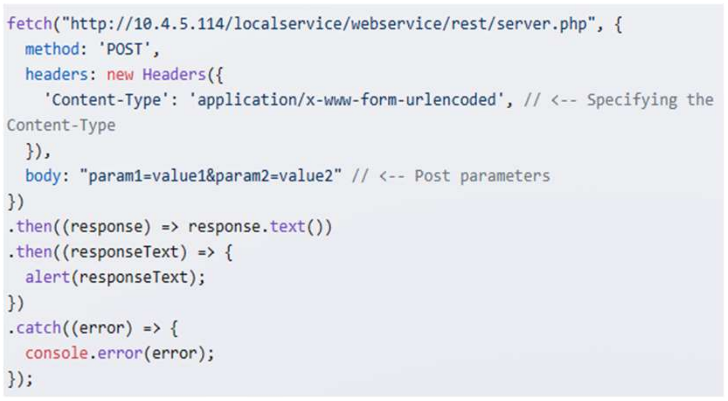
FETCH API

Fetch API with Get Method

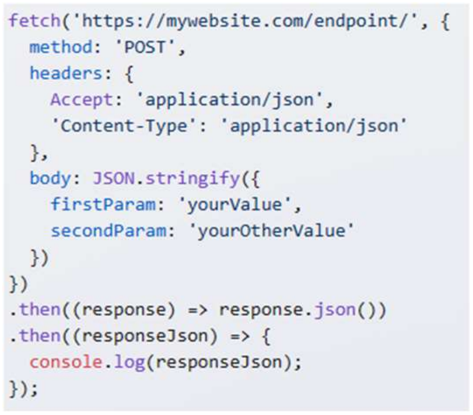
```
fetch('https://example.com/data?
param1=value1&param2=value2', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json'
  }
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error(error));
```

FETCH API

Fetch API with Post Method



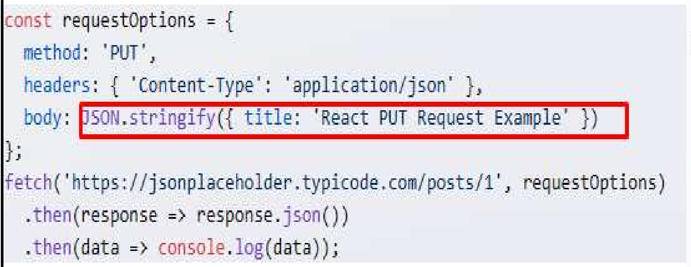
```
fetch("http://10.4.5.114/localservice/webservice/rest/server.php", {
  method: 'POST',
  headers: new Headers({
    'Content-Type': 'application/x-www-form-urlencoded', // <-- Specifying the
Content-Type
  }),
  body: "param1=value1&param2=value2" // <-- Post parameters
})
.then((response) => response.text())
.then((responseText) => {
  alert(responseText);
})
.catch((error) => {
  console.error(error);
});
```



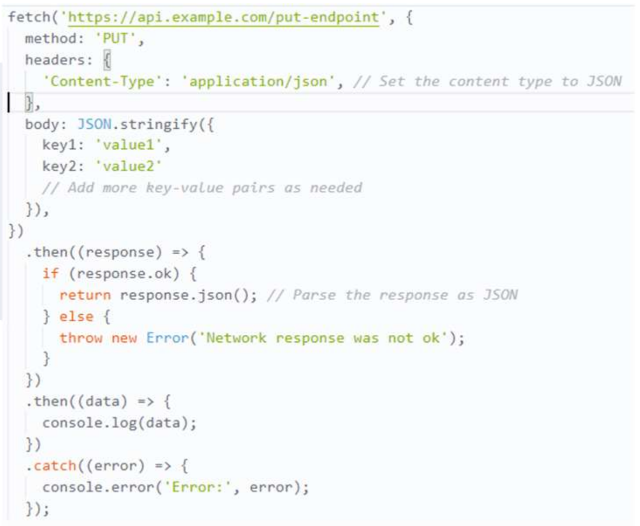
```
fetch('https://mywebsite.com/endpoint/', {
  method: 'POST',
  headers: {
    Accept: 'application/json',
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    firstParam: 'yourValue',
    secondParam: 'yourOtherValue'
  })
})
.then((response) => response.json())
.then((responseJson) => {
  console.log(responseJson);
});
```

FETCH API

Fetch API with Put Method



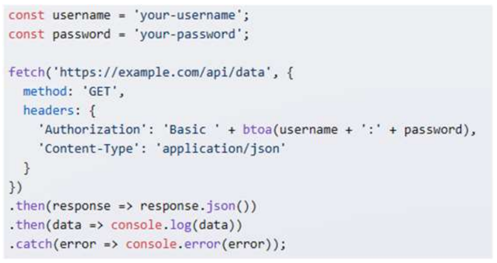
```
const requestOptions = {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ title: 'React PUT Request Example' })
};
fetch('https://jsonplaceholder.typicode.com/posts/1', requestOptions)
  .then(response => response.json())
  .then(data => console.log(data));
```



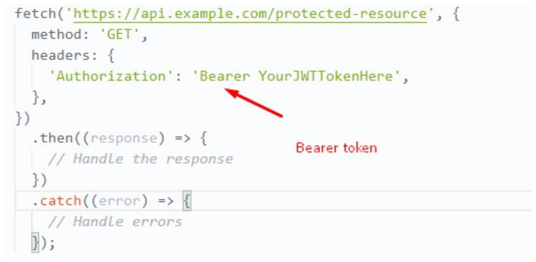
```
fetch('https://api.example.com/put-endpoint', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json', // Set the content type to JSON
  },
  body: JSON.stringify({
    key1: 'value1',
    key2: 'value2'
    // Add more key-value pairs as needed
  }),
})
  .then((response) => {
    if (response.ok) {
      return response.json(); // Parse the response as JSON
    } else {
      throw new Error('Network response was not ok');
    }
  })
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.error('Error:', error);
  });
```

FETCH API

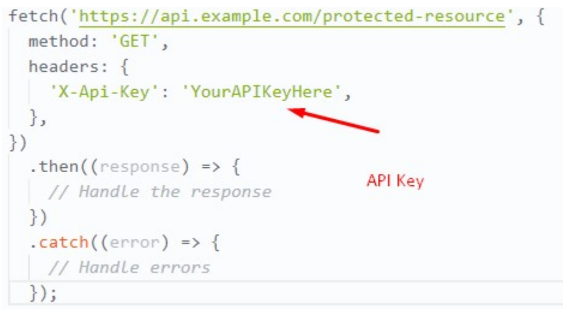
Authentication with Fetch API



```
const username = 'your-username';
const password = 'your-password';

fetch('https://example.com/api/data', {
  method: 'GET',
  headers: {
    'Authorization': 'Basic ' + btoa(username + ':' + password),
    'Content-Type': 'application/json'
  }
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error(error));
```



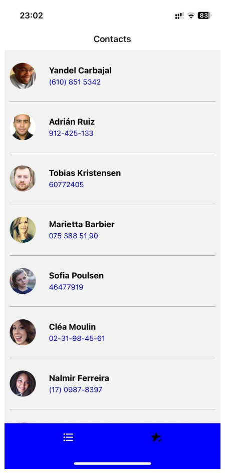
```
fetch('https://api.example.com/protected-resource', {
  method: 'GET',
  headers: {
    'Authorization': 'Bearer YourJWTTokenHere',
  },
})
  .then((response) => {
    // Handle the response
  })
  .catch((error) => {
    // Handle errors
  });
```



```
fetch('https://api.example.com/protected-resource', {
  method: 'GET',
  headers: {
    'X-Api-Key': 'YourAPIKeyHere',
  },
})
  .then((response) => {
    // Handle the response
  })
  .catch((error) => {
    // Handle errors
  });
```

Exercise

In Lab 4, contact information data is transferred from the Contacts screen to the favorites screen using Redux Toolkit. Please change Redux toolkit to AsyscStorage



23:02

Contacts

Yandel Carbajal
(610) 851 5342

Adrián Ruiz
912-425-133

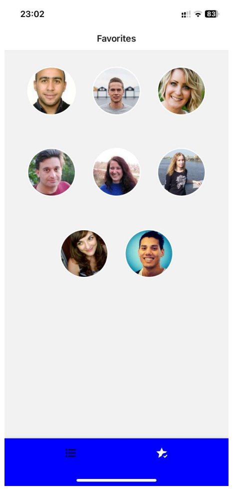
Tobias Kristensen
60772405

Marietta Barbier
075 388 51 90

Sofia Poulsen
46477919

Cléa Moulin
02-31-98-45-61

Nalmir Ferreira
(17) 0987-8397



Q&A

EIU
TRƯỜNG ĐẠI HỌC
QUỐC TẾ
MIỀN ĐÔNG
EASTERN
INTERNATIONAL
UNIVERSITY