# WORKSHOP

# CREATE A REACT-NATIVE APP USING LEAGUE OF LEGENDS API

# Prerequisites :

React-native
npm
yarn
expo : `npm install expo-cli --global`
A league of legends account
an api key league of legends https://developer.riotgames.com/
expo android/ios app
Postman https://www.postman.com/downloads/

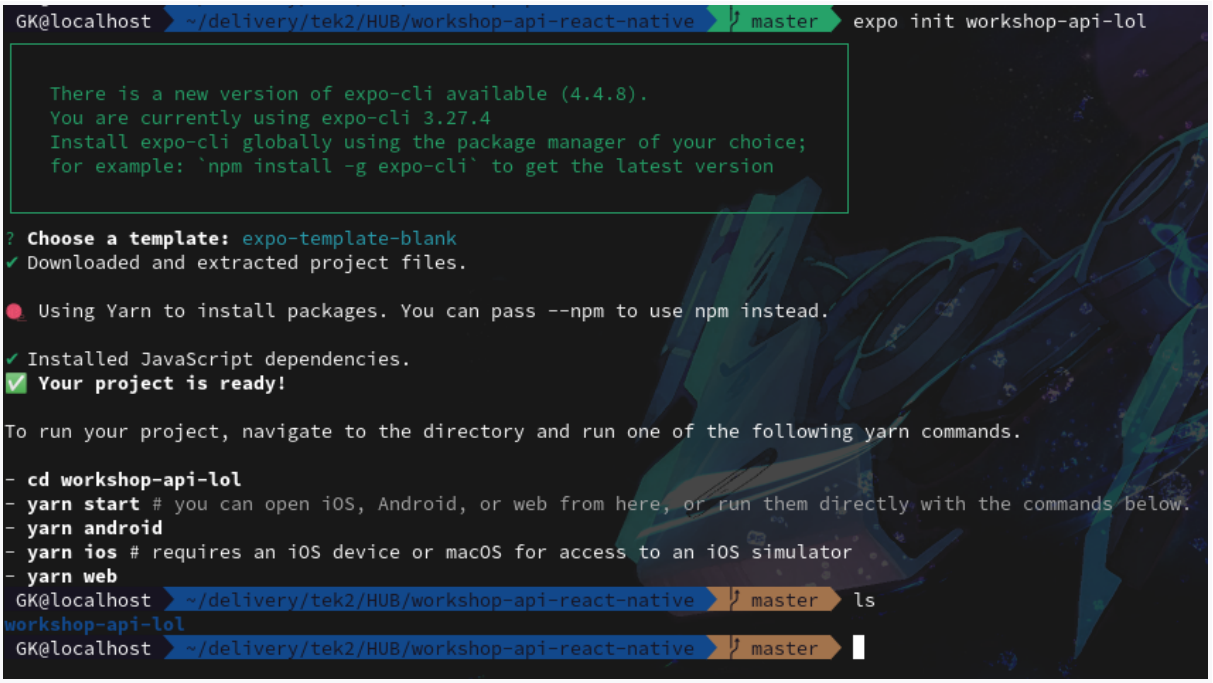
# Introduction

In this workshop you'll create an app for both IOS and Android using react-native and the language JavaScript.

We will step by step create an app that will use league of legends API to display some informations about a user, such as his summoner icon, his rank, the result of his last 10 games and more (A little like OP.GG)

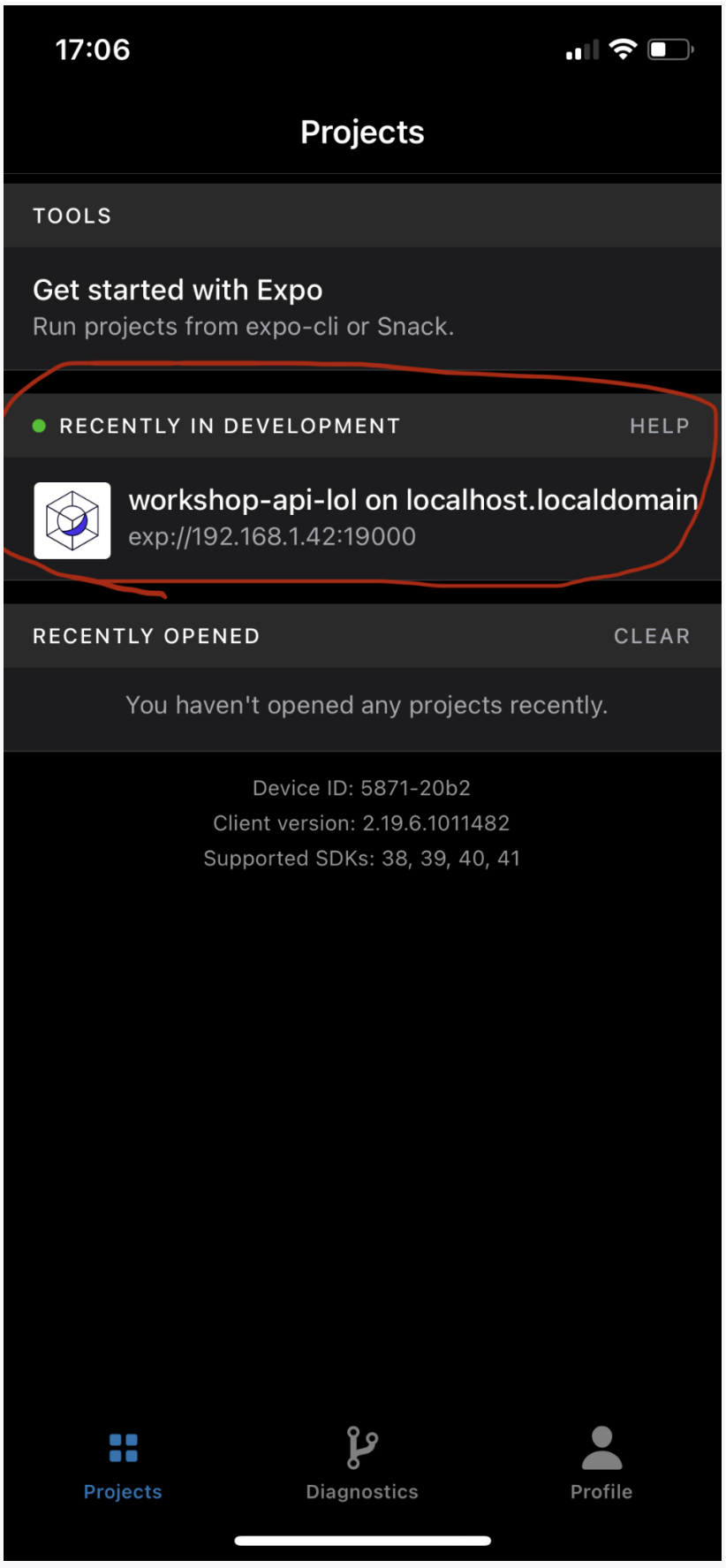
# Step 1 : Create the app

create a new repository and type : `expo init <project name>`
Select option 1 ( blank)

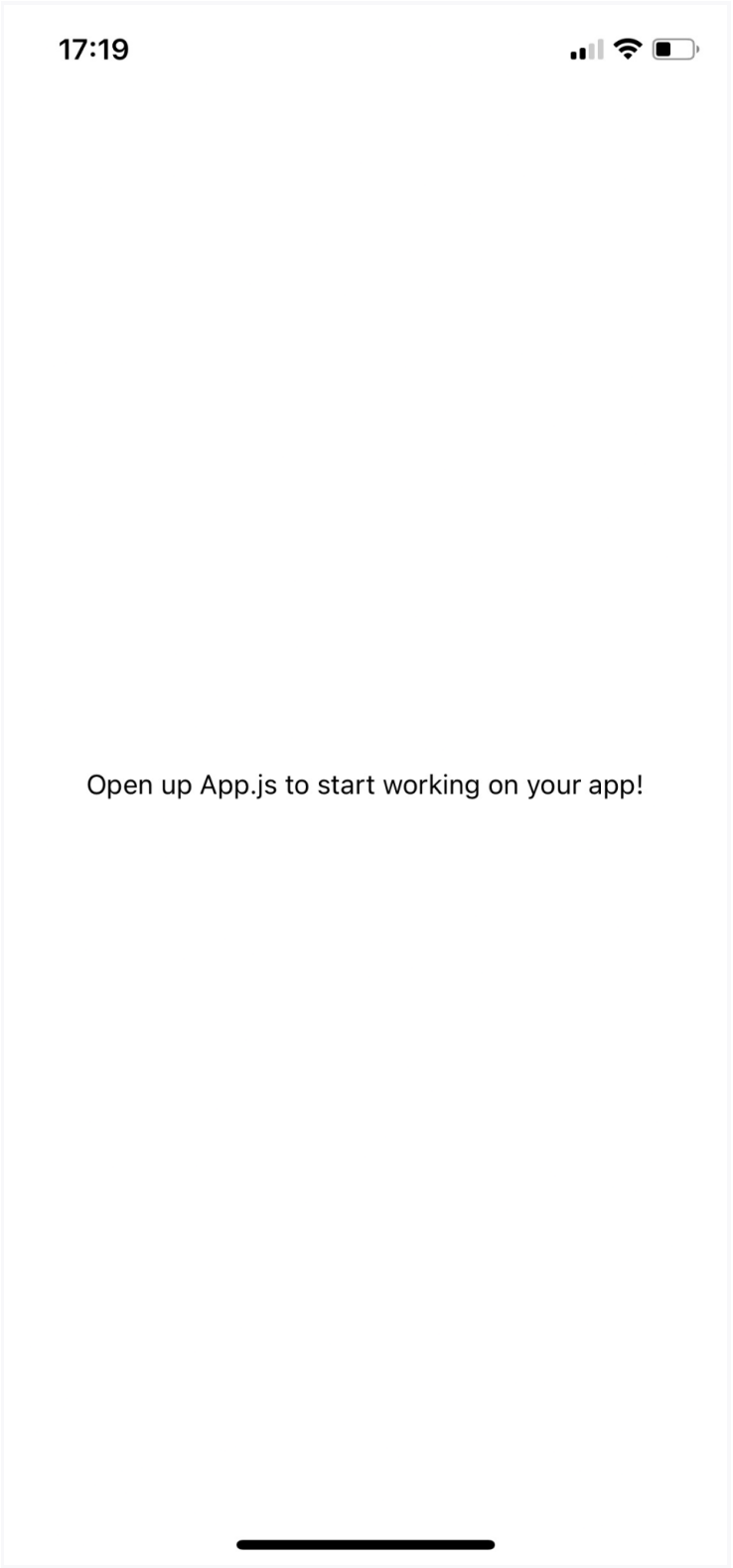



now go to the repository and type : `expo start`

go to your phone, open expo then open your app


17:06
Projects
TOOLS
Get started with Expo
Run projects from expo-cli or Snack.
RECENTLY IN DEVELOPMENT
HELP
workshop-api-lol on localhost.localdomain
exp://192.168.1.42:19000
RECENTLY OPENED
CLEAR
You haven't opened any projects recently.
Device ID: 5871-20b2
Client version: 2.19.6.1011482
Supported SDKs: 38, 39, 40, 41
Projects
Diagnostics
Profile

You should have something like that :



# Step 2 :

Open your vscode or whatever text editor you use.

Go to App.js and start modifying some code.

First we will remove the text you just saw.

Remove

```
<Text>Open up App.js to start working on your app!</Text>
```

Create a folder named screens and add a file names MainScreen.js. Add the following imports :

```
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
```

Then add this code :

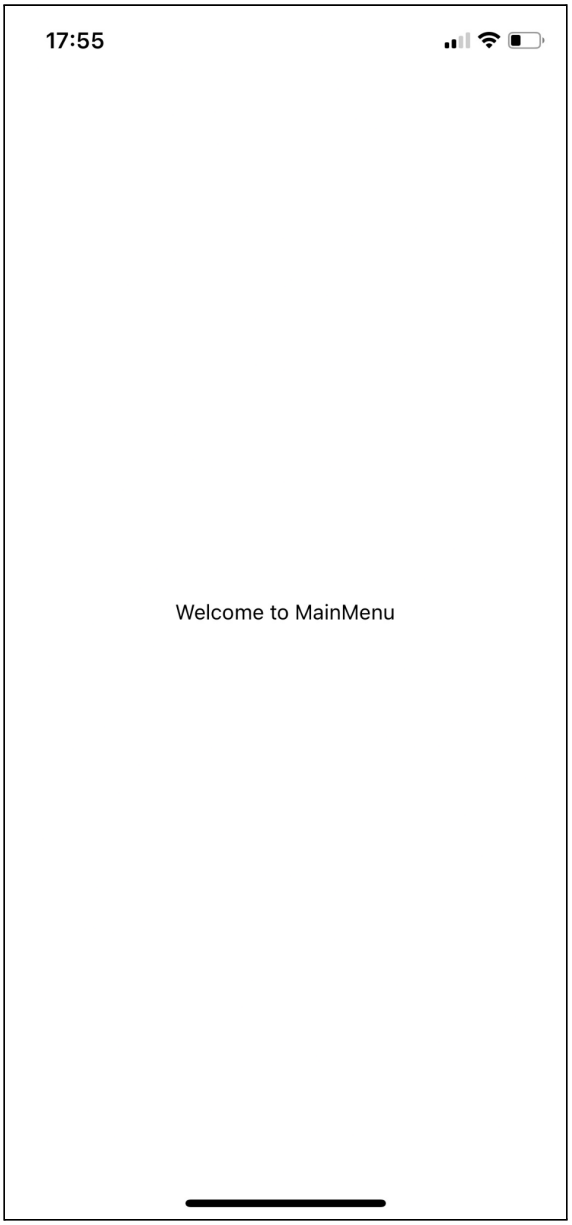
```
class MainMenu extends React.Component {
   constructor(props) {
     super(props);
     this.state = {

     };
   }

   render() {
       const { } = this.state;
       return (
           <>
               <Text>Welcome to MainMenu</Text>
           </>
       )
   }
}

export default MainMenu;
```

Go back to app.js, import the new .js and add it where the <Text> previously was.

You should have something like that :

```
import { StatusBar } from 'expo-status-bar';
import React from 'react';
import { StyleSheet, Text, View } from 'react-native';
import MainMenu from './screens/MainScreen';

export default function App() {
 return (
   <View style={styles.container}>
     <MainMenu></MainMenu>
     <StatusBar style="auto" />
   </View>
 );
}

const styles = StyleSheet.create({
 container: {
   flex: 1,
   backgroundColor: '#fff',
   alignItems: 'center',
   justifyContent: 'center',
 },
});
```

Your app should look like that :



# STEP 3

First we will add a TextInput (a place where a user can enter their username for example)

here's the documentation for textinput
https://docs.expo.io/versions/v41.0.0/react-native/textinput/

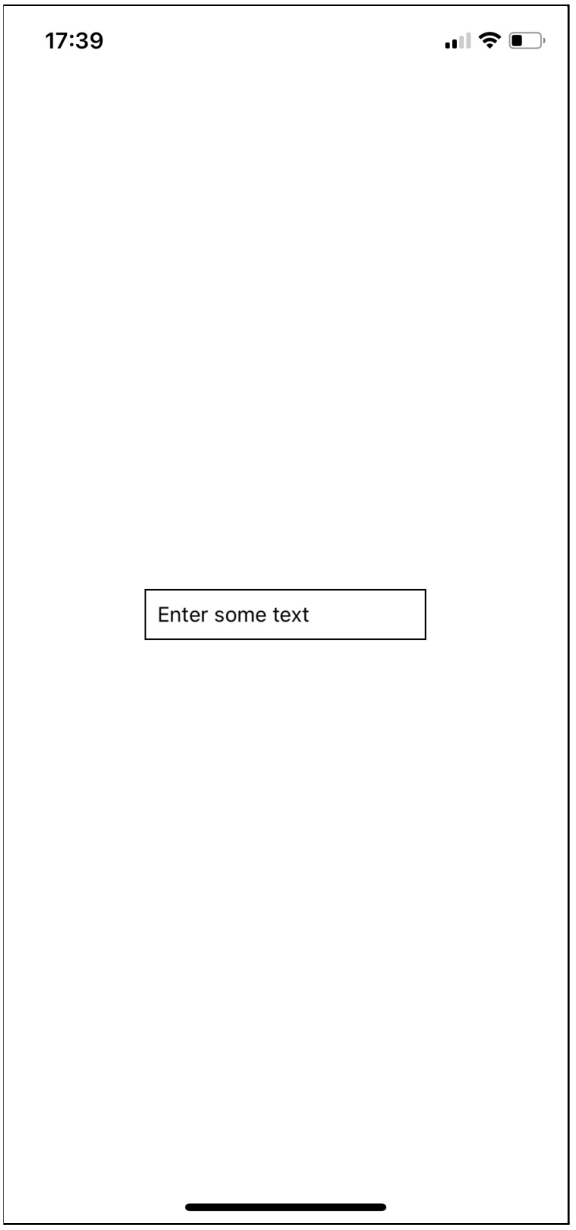
Add the import for TextInput :

```
import { TextInput } from 'react-native';
```

then add a textInput Containing a placeholder and attach some style to it.

You should have something like that :

```
import React from 'react';
import { StyleSheet, Text, View, TextInput } from 'react-native';

const styles = StyleSheet.create({
 firstInput: {
   borderColor: 'black',
   borderWidth: 1,
   padding: '2%',
   width: '50%'
 }
});

class MainMenu extends React.Component {
   constructor(props) {
     super(props);
     this.state = {

     };
   }

   render() {
       const { } = this.state;
       return (
           <>
             <TextInput
               style={styles.firstInput}
               placeholder="Enter some
text"

placeholderTextColor='black'
             />
           </>
       )
   }
}

export default MainMenu;
```



Currently, our TextInput doesn’t have any value and doesn’t save the text you added anywhere.

To do so, you will need to give your textInput the attribute value

first add a variable named inputValue with an empty string as a default value

```
    this.state = {
      inputValue: '',
    };
```

then give the attribute ‘value’ to the TextInput and make it be inputValue
Add a text with as value InputValue to see if what you did worked.

You should now have something like that :

```
class MainMenu extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      inputValue: '',
    };
  }

  render() {
      const { inputValue } = this.state;
      return (
          <>
            <TextInput
              style={styles.firstInput}
              placeholder="Enter some text"
              placeholderTextColor='black'
              value={inputValue}
            />
            <Text>{inputValue}</Text>
          </>
      )
  }
}
```

But if you test, you'll see that your text won't be changed.
To be able to do that, you'll need to add the attribute 'onChangeText'
You'll need to use this.setState to modify the value of the text.

```
class MainMenu extends React.Component {
   constructor(props) {
     super(props);
     this.state = {
       inputValue: '',
     };
   }

   render() {
       const { inputValue } = this.state;
       return (
           <>
             <TextInput
               style={styles.firstInput}
               placeholder="Enter some text"
               placeholderTextColor='black'
               value={inputValue}
               onChangeText={(value)=> this.setState({ inputValue: value })}
             />
             <Text>{inputValue}</Text>
           </>
       )
   }
}
```

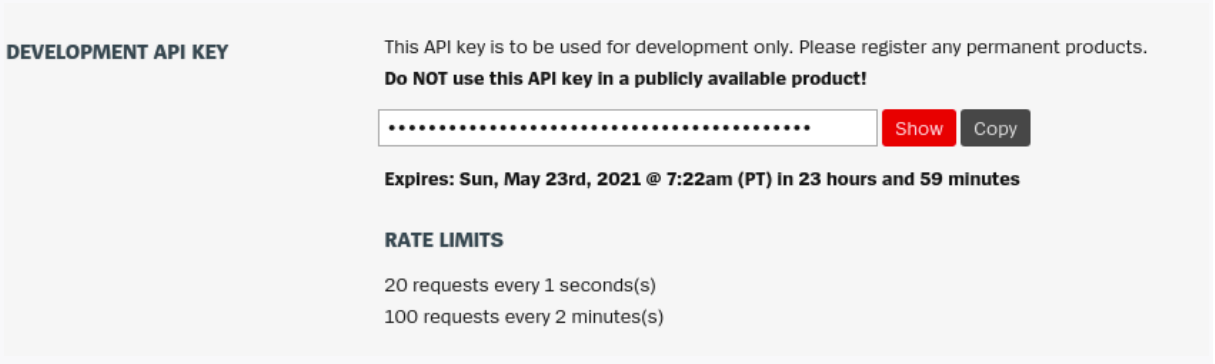
Now you'll see that the text will change and will be displayed under the TextInput.

# STEP 4

Now that we have some basis with react-native, we can start using league of legends API.

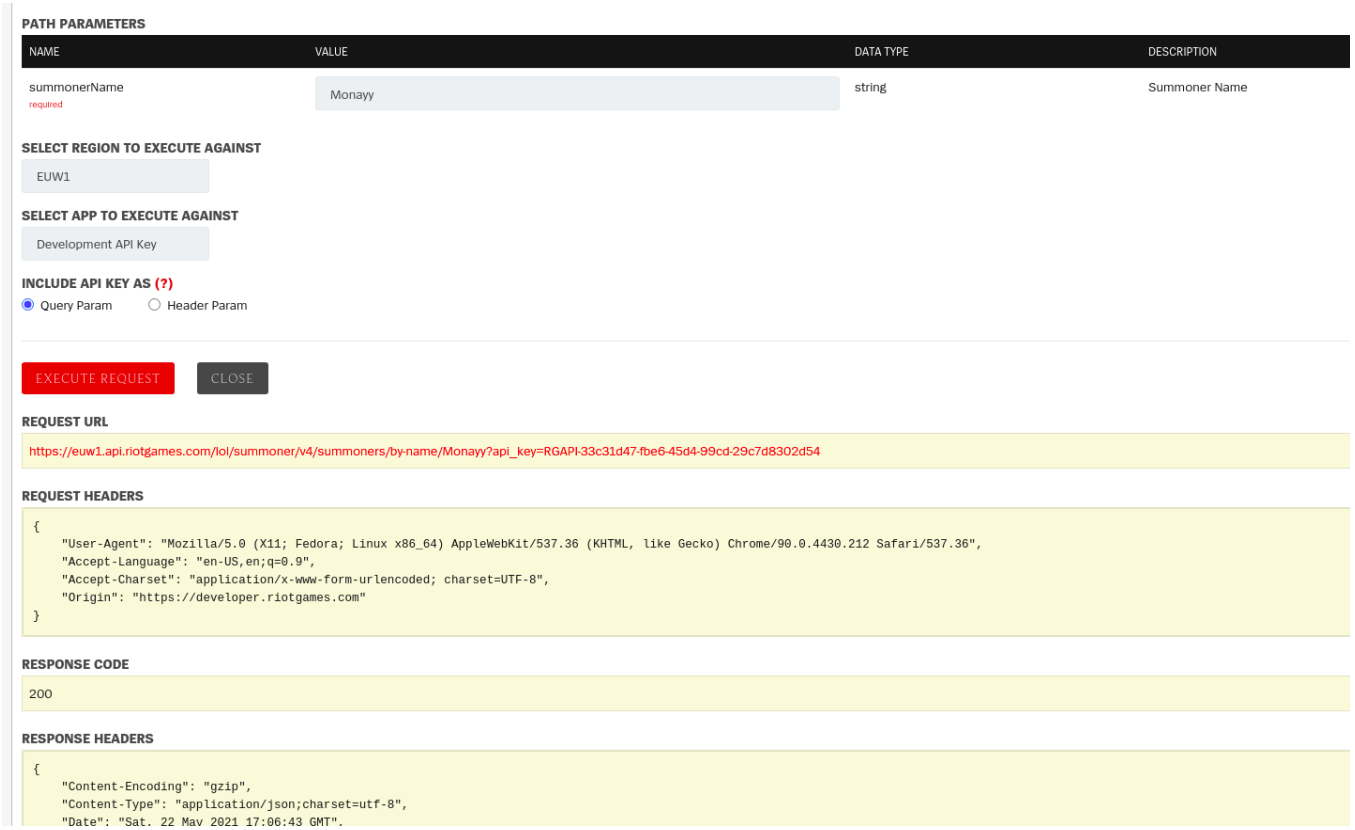
Open Postman and then go to https://developer.riotgames.com/

copy your API code




First we will try using the api with postman.

Go to https://developer.riotgames.com/apis#summoner-v4
let's try the second one.



For this workshop we will need to use / call several routes of league of legend's API

Here's the routes we will need to use in order to get the informations about a user, a little like what you can see in op.gg

1 : https://developer.riotgames.com/apis#summoner-v4/GET_getBySummonerName

2 : https://developer.riotgames.com/apis#league-v4/GET_getLeagueEntriesForSummoner

3 : http://ddragon.leagueoflegends.com/cdn/10.13.1/img/profileicon/

You can read the documentation contained inside the given link to understand how to use it and test it.

# STEP 5

Now that you know a bit about the API, we will try to use it in our application.

To do so, we will use our TextInput and create a button that will start a function in which we will use the API.

To create a button, you need <TouchableOpacity>

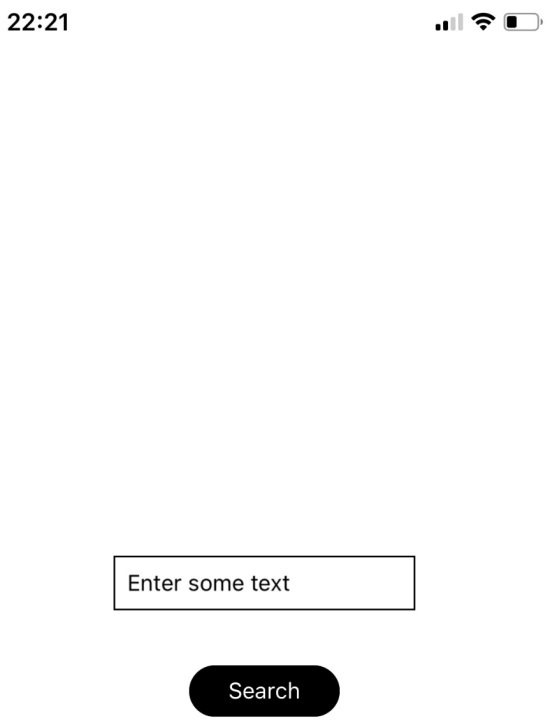
Don’t forget to add the import to your import list

Add some style to your button

For example :

```
<TouchableOpacity
  style={styles.searchButton}
>
  <Text style={{ color: 'white'
}}>Search</Text>
</TouchableOpacity>
```

```
searchButton: {
   borderWidth: 1,
   marginTop: 20,
   height: 32,
   width: '25%',
   justifyContent: 'center',
   alignItems: 'center',
   borderRadius: 360,
   backgroundColor: 'black',
   alignSelf: 'center',
   textAlign: 'center',
 }
```



After creating the button, let’s create our function that will use the API.

Inside the class under the constructor and above the render create a function.

for example :

```
handleRequest = () => {
  console.log(“here”);
}
```

Call it when the button is pressed

```
<TouchableOpacity
  style={styles.searchButton}
  onPress={() => this.handleRequest()}
>
```

Let’s use the API.

You will need to use fetch function. To know more about it you should see this website https://jsonplaceholder.typicode.com/

Let’s change our function handleRequest to make her do a request to this route : https://developer.riotgames.com/apis#summoner-v4/GET_getBySummonerName

The response you should get, should be something like that :


```
{
    "id": "Ia0_MCsL0I5A0kbybE1bWPiz1raun7I2CXBDtPUvJ41bFTBe",
    "accountId": "i73dZbsx6ZF4gGQUY9zuJguWdE-POQojuRJRPmVmtLyeYZs7sQq2DJWF",
    "puuid":
"6RCKKtq2veDyjhroXSBDxsFV9oEjgUsxcxJeO2dIjqru1OfuIk_4BoD_j4FCxpHVrImzNUPjnTSX0A",
    "name": "Hrothgorr",
    "profileIconId": 4814,
    "revisionDate": 1620939018000,
    "summonerLevel": 134
}
```


Of course it’s only if the given username correspond to a user.

If not consider checking the status code.

To make our request we will use the fetch function, give it the uri

https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/Monayy?api_key=

We will divide this link into several pieces.

Let’s add our API key to this.state :

```
  this.state = {
    inputValue: '',
    apiKey: 'your key'
  };
```

Now we have our api key which is this.state.apiKey and the name of the user : inputValue.

We will give our fetch something like that :

'https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/' + this.state.inputValue + '?api_key=' + this.state.apiKey

If you try to make a direct call to the API, you might get a CORS ORIGIN ERROR.
To fix that add `mode: 'no-cors'` to your fetch.

For now, we just want to look what will be the result of our api, so just print the response in the shell of your computer (console.log)

hint : .then / .json()

Answer :

```
 handleRequest = () => {

fetch('https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/' + this.state.inputValue + '?api_key=' + this.state.apiKey, {
    mode: 'no-cors'
  })
  .then((response) => {
    return response.json()
  })
  .then((result) => {
    console.log('result = ', result);
  })
 }
```

it will print something like that on your shell (assuming the username was valid) :

```
result =  Object {
  "accountId":
"jqcCzxT8ZuQUPoDj3E2NWT8LgcxqSyOmWOOKlt6iczon5gU",
  "id":
"HwJJ0pvAWnzTuuT6EahHrsp23Xq3Fm2wq_J4knpJYEWLoZU",
  "name": "Monayy",
  "profileIconId": 558,
  "puuid":
"htWcPF8skNzMHmcnaOp6StbYX9wgHm1ei5PKZgghuDYZyNSe10YX-QXYosSU7t2QJLp1NMnlwOFbzQ",
  "revisionDate": 1621618715000,
  "summonerLevel": 128,
}
```

if you want to use the results and make sure your app doesn't crash, consider checking if result.status exists.

# STEP 6

Now that we have our first result, the second step is to retrieve specific informations about our user.

https://developer.riotgames.com/apis#league-v4/GET_getLeagueEntriesForSummoner

As shown in the documentation of this route, we have to use the summoner id of the player to access these informations. Well, we just made our function retrieve this id.

Add a variable userId to this.state.

What we want to do is to save the summoner id of the player into this variable when we receive the response of the first request, to be able to use it later.

Your function should now look like that :

```
handleRequest = () => {

fetch('https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/' + this.state.inputValue + '?api_key=' + this.state.apiKey, {
    mode: 'no-cors'
  })
  .then((response) => {
    return response.json()
  })
  .then((result) => {
    console.log('result = ', result);
    if (!result.status) {
      this.setState({ userId: result.id});
    }
  })
 }
```

Now that we have stored the summoner id, let's use it.

after storing, make another request, but to the new route this time. (It need to be inside the .then((result) =>)

Like we did before, we will need to divide the URI in several pieces.

'https://euw1.api.riotgames.com/lol/league/v4/entries/by-summoner/' + this.state.userId + '?api_key=' + this.state.apiKey

we will do the same as before and just print the result. (You can also check if the status exists here)

After doing so your function should look like that :

```
 handleRequest = () => {
   fetch('https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/'
+ this.state.inputValue + '?api_key=' + this.state.apiKey, {
     mode: 'no-cors'
   })
   .then((response) => {
     return response.json()
   })
   .then((result) => {
     console.log('result = ', result);
     if (!result.status) {
       this.setState({ userId: result.id});
     }

fetch('https://euw1.api.riotgames.com/lol/league/v4/entries/by-summoner/' +
this.state.userId + '?api_key=' + this.state.apiKey, {
       mode: 'no-cors'
     })
     .then((response) => {
         return response.json()
     })
     .then((result) => {
         console.log('res is : ', result)
     })
   })
 }
```

If you want to make a prettier function, you can search how to use async/await in javascript.

We won't be using this here as we don't want to complicate things.

The result received by the last request should look like that :


```
result =  Object {
  "accountId":
"i73dZbsx6ZF4gGQUY9zuJguWdE-POQojuRJRPmVmtLyeYZs7sQq2DJWF
",
  "id":
"Ia0_MCsL0I5A0kbybE1bWPiz1raun7I2CXBDtPUvJ41bFTBe",
  "name": "Hrothgorr",
  "profileIconId": 4814,
  "puuid":
"6RCKKtq2veDyjhroXSBDxsFV9oEjgUsxcxJeO2dIjqru1OfuIk_4BoD_
j4FCxpHVrImzNUPjnTSX0A",
  "revisionDate": 1620939018000,
  "summonerLevel": 134,
}
res is :  Array [
  Object {
    "freshBlood": false,
    "hotStreak": false,
    "inactive": false,
    "leagueId": "7d4d9b7c-3916-46f6-aef8-8d29a3ac63db",
    "leaguePoints": 53,
    "losses": 13,
    "queueType": "RANKED_SOLO_5x5",
    "rank": "II",
    "summonerId":
"Ia0_MCsL0I5A0kbybE1bWPiz1raun7I2CXBDtPUvJ41bFTBe",
    "summonerName": "Hrothgorr",
    "tier": "PLATINUM",
    "veteran": false,
```

```
    "wins": 6,
  },
]
```

As you can see we got an array. It’s because there is several queue in league of legends (solo queue and Flex queue).

Let’s store the array inside a ‘userInfos’ variable in this.state.

```
userInfos: [],
```

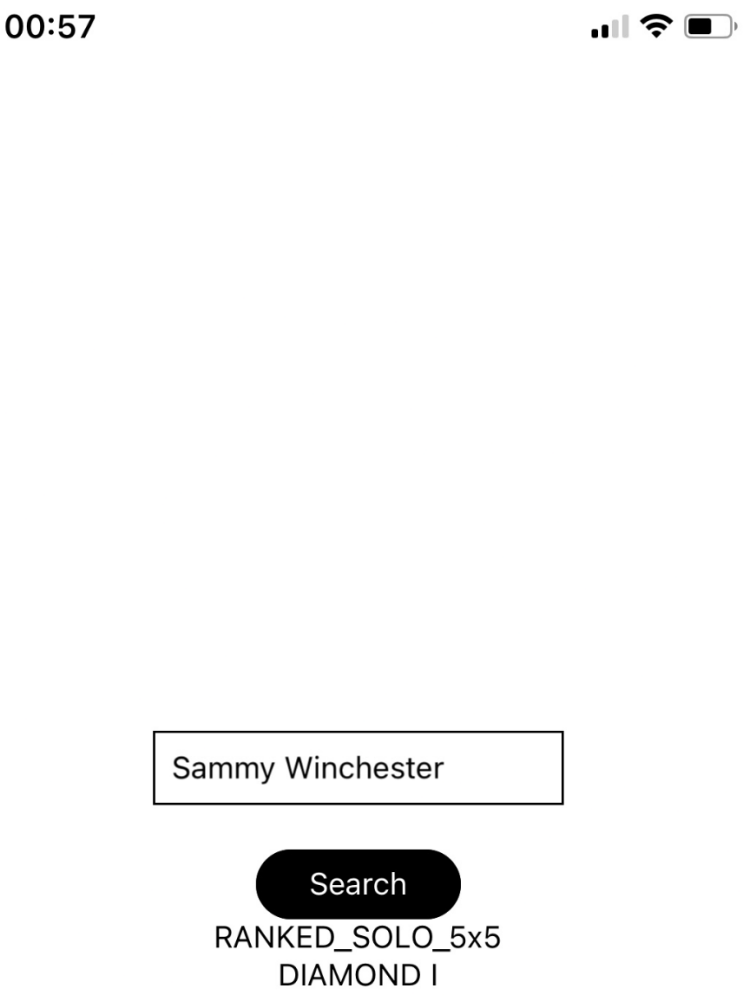
Then let’s display some of these informations in our app.

Add 2 <Text> like that :

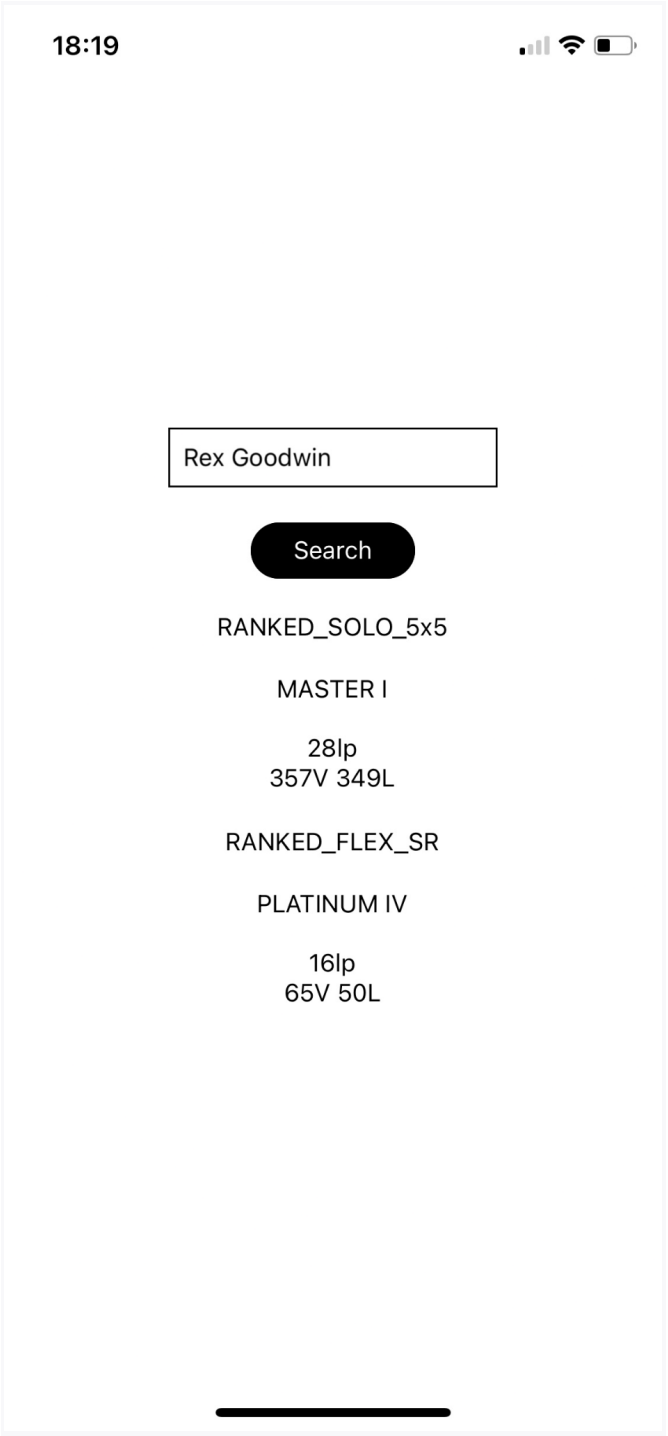
```
        <Text>
          {userInfos[0] ? userInfos[0].queueType : null}
        </Text>
        <Text>
          {userInfos[0] ? userInfos[0].tier : null} {userInfos[0]
? userInfos[0].rank : null}
        </Text>
```

As you can see if userInfos[0] exists we display .queueType, but if it doesn’t we will display null, which won’t display anything.

Your app should display something like that



If you add some display, for example all the informations of solo queue and flex queue.




As you can see we have access to the number of league points in each queue, its rank, the number of wins and losses. Look at the documentation to know more about the informations you have access to. Here is the code to get the same screen as above :

```
render() {
   const { inputValue, userInfos } = this.state;
   return (
     <>
       <TextInput
         style={styles.firstInput}
         placeholder="Enter some text"
         placeholderTextColor='black'
         value={inputValue}
         onChangeText={(value)=> this.setState({ inputValue: value })}
       />
       <TouchableOpacity
         style={styles.searchButton}
         onPress={() => this.handleRequest()}
       >
         <Text style={{ color: 'white' }}>Search</Text>
       </TouchableOpacity>
       <Text style={{ padding: '5%' }}>
         { userInfos[0] ? userInfos[0].queueType : null }
       </Text>
       <Text>
         { userInfos[0] ? userInfos[0].tier : null } { userInfos[0] ?
userInfos[0].rank + '\n' : null }
       </Text>
       <Text>
         { (userInfos[0] && userInfos[0].leaguePoints) ?
userInfos[0].leaguePoints + 'lp': null }
       </Text>
       <Text>
         { (userInfos[0] && userInfos[0].wins && userInfos[0].losses) ?
userInfos[0].wins + 'V ' + userInfos[0].losses + 'L' : null }
       </Text>
       <Text style={{ padding: '5%' }}>
         { userInfos[1] ? userInfos[1].queueType : null }
       </Text>
       <Text>
         { userInfos[1] ? userInfos[1].tier : null } { userInfos[1] ?
userInfos[1].rank + '\n' : null }
       </Text>
       <Text>
         { (userInfos[1] && userInfos[1].leaguePoints) ?
userInfos[1].leaguePoints + 'lp': null }
       </Text>
       <Text>
         { (userInfos[1] && userInfos[1].wins && userInfos[1].losses) ?
userInfos[1].wins + 'V ' + userInfos[1].losses + 'L' : null }
       </Text>
     </>
```

```
    )
  }
```

If you want to display the summoner icon you will have to use https://developer.riotgames.com/docs/lol#data-dragon_other and the component image : https://docs.expo.io/versions/v41.0.0/react-native/image/

I will let you try to display the icon of the player before giving the answer.

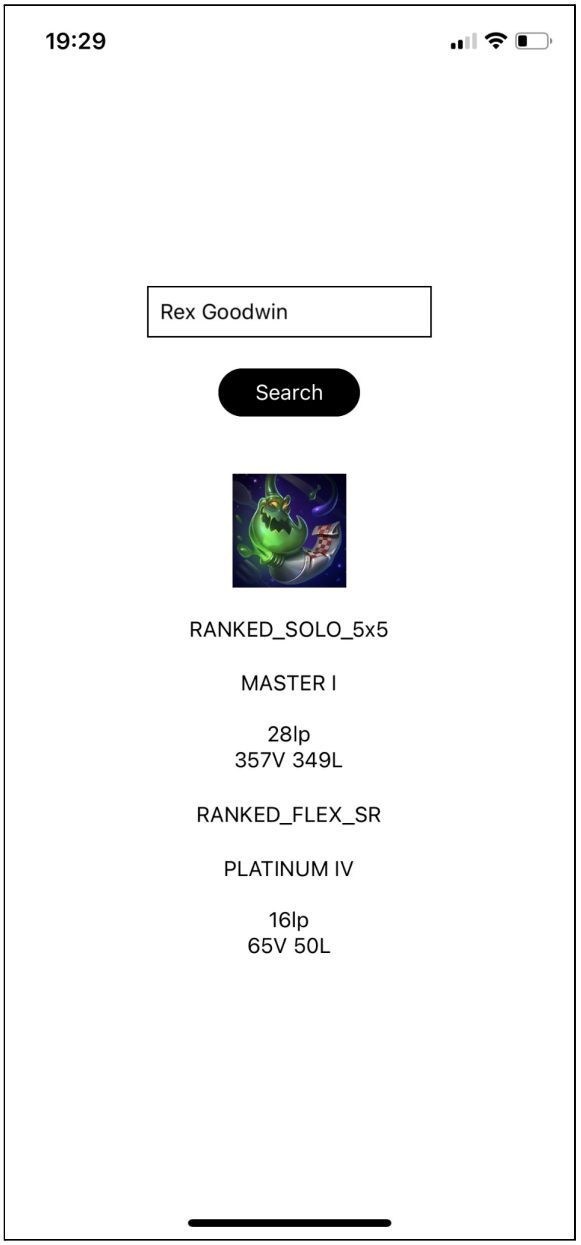
To display the icon, you will have to retrieve the profileIconId from the first request you made (the one with the username)

here's an example of the link you will have to get :

```
'http://ddragon.leagueoflegends.com/cdn/10.13.1/img/profileicon/' +
profileIconId + '.png'
```

```
<Image style={{ padding: '10%', marginTop: '10%' }} source={{ uri:
'http://ddragon.leagueoflegends.com/cdn/10.13.1/img/profileicon/' +
profileIconId + '.png' }}></Image>
```

Place this after the button.




Here is the final code :

```
import React from 'react';
import { StyleSheet, Text, View, TextInput, TouchableOpacity, Image } from
'react-native';

const styles = StyleSheet.create({
 firstInput: {
   borderColor: 'black',
   borderWidth: 1,
   padding: '2%',
   width: '50%'
 },
 searchButton: {
   borderWidth: 1,
   marginTop: 20,
   height: 32,
   width: '25%',
   justifyContent: 'center',
   alignItems: 'center',
   borderRadius: 360,
   backgroundColor: 'black',
   alignSelf: 'center',
   textAlign: 'center',
 }
});

class MainMenu extends React.Component {
 constructor(props) {
   super(props);
   this.state = {
     inputValue: '',
     apiKey: 'RGAPI-3df7a4cb-50b0-4582-98cf-2f803467a594',
     userId: '',
     userInfos: [],
     profileIconId: ''
   };
 }

 handleRequest = () => {
   fetch('https://euw1.api.riotgames.com/lol/summoner/v4/summoners/by-name/'
+ this.state.inputValue + '?api_key=' + this.state.apiKey, {
     mode: 'no-cors'
   })
   .then((response) => {
     return response.json()
   })
   .then((result) => {
     console.log('result = ', result);
     if (!result.status) {
```

```
        this.setState({ userId: result.id});
        this.setState({ profileIconId: result.profileIconId })
      }

fetch('https://euw1.api.riotgames.com/lol/league/v4/entries/by-summoner/' +
this.state.userId + '?api_key=' + this.state.apiKey, {
        mode: 'no-cors'
      })
      .then((response) => {
          return response.json()
      })
      .then((result) => {
          console.log('res is : ', result)
          this.setState({ userInfos: result });
      })
    })
  }

  render() {
    const { inputValue, userInfos, profileIconId } = this.state;
    return (
      <>
        <TextInput
          style={styles.firstInput}
          placeholder="Enter some text"
          placeholderTextColor='black'
          value={inputValue}
          onChangeText={(value)=> this.setState({ inputValue: value })}
        />
        <TouchableOpacity
          style={styles.searchButton}
          onPress={() => this.handleRequest()}
        >
          <Text style={{ color: 'white' }}>Search</Text>
        </TouchableOpacity>
        <Image style={{ padding: '10%', marginTop: '10%' }} source={{ uri:
'http://ddragon.leagueoflegends.com/cdn/10.13.1/img/profileicon/' +
profileIconId + '.png' }}></Image>
        <Text style={{ padding: '5%' }}>
          { userInfos[0] ? userInfos[0].queueType : null }
        </Text>
        <Text>
          { userInfos[0] ? userInfos[0].tier : null } { userInfos[0] ?
userInfos[0].rank + '\n' : null }
        </Text>
        <Text>
          { (userInfos[0] && userInfos[0].leaguePoints) ?
userInfos[0].leaguePoints + 'lp': null }
```

```
        </Text>
        <Text>
          { (userInfos[0] && userInfos[0].wins && userInfos[0].losses) ?
userInfos[0].wins + 'V ' + userInfos[0].losses + 'L' : null }
        </Text>
        <Text style={{ padding: '5%' }}>
          { userInfos[1] ? userInfos[1].queueType : null }
        </Text>
        <Text>
          { userInfos[1] ? userInfos[1].tier : null } { userInfos[1] ?
userInfos[1].rank + '\n' : null }
        </Text>
        <Text>
          { (userInfos[1] && userInfos[1].leaguePoints) ?
userInfos[1].leaguePoints + 'lp': null }
        </Text>
        <Text>
          { (userInfos[1] && userInfos[1].wins && userInfos[1].losses) ?
userInfos[1].wins + 'V ' + userInfos[1].losses + 'L' : null }
        </Text>
      </>
    )
  }
}

export default MainMenu;
```

# STEP 7 : YOUR TURN

Now that you know how to use the API, and you know a bit about react-native, you can try to display other informations, make the display prettier or use another API !

You could show the champion rotation
https://developer.riotgames.com/apis#champion-v3/GET_getChampionInfo

Get the list of the best players in specific queue
https://developer.riotgames.com/apis#league-exp-v4/GET_getLeagueEntries

Use TFT (Teamfight Tactics) API
https://developer.riotgames.com/apis#tft-league-v1