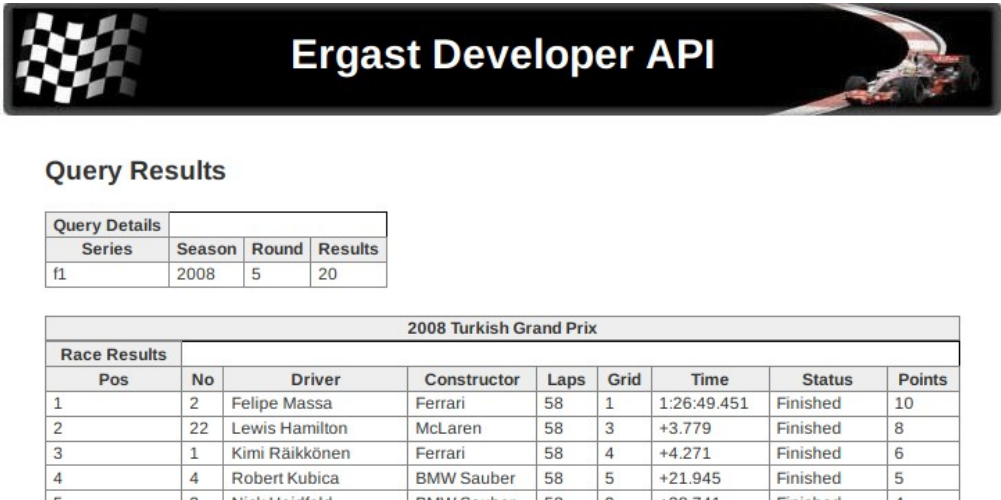


Grupa	PS3
Imię i nazwisko	Artur Tagisow
Zadanie	Projekt 3: API sieciowe.

Teoria

API Ergast pozwala na odczyt w dwojaki sposób:

poprzez odczyt pliku .html (<http://ergast.com/api/f1/2008/5/results>) (format przyjazny dla ludzkiego oka)

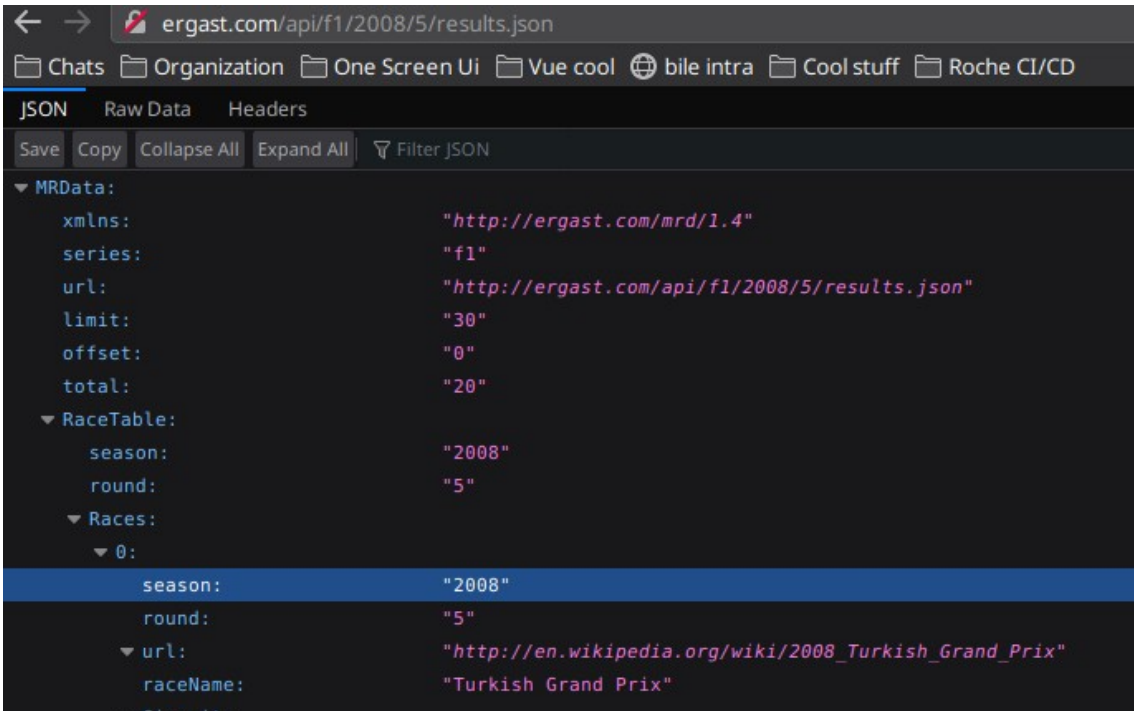


The screenshot shows the Ergast Developer API interface. At the top is a banner with a checkered flag and the text "Ergast Developer API". Below it, the "Query Results" section displays a table of query details and a table of race results for the 2008 Turkish Grand Prix.

Query Details			
Series	Season	Round	Results
f1	2008	5	20

2008 Turkish Grand Prix								
Race Results								
Pos	No	Driver	Constructor	Laps	Grid	Time	Status	Points
1	2	Felipe Massa	Ferrari	58	1	1:26:49.451	Finished	10
2	22	Lewis Hamilton	McLaren	58	3	+3.779	Finished	8
3	1	Kimi Räikkönen	Ferrari	58	4	+4.271	Finished	6
4	4	Robert Kubica	BMW Sauber	58	5	+21.945	Finished	5
5	3	Nick Heidfeld	BMW Sauber	58	9	+38.741	Finished	4

poprzez odczyt pliku .json: <http://ergast.com/api/f1/2008/5/results.json> (format przyjazny aplikacjom)



The screenshot shows a web browser displaying the JSON response from the Ergast API. The URL bar shows `ergast.com/api/f1/2008/5/results.json`. The JSON data is expanded, showing the following structure:

```

{
  "MRData": {
    "xmlns": "http://ergast.com/mrd/1.4",
    "series": "f1",
    "url": "http://ergast.com/api/f1/2008/5/results.json",
    "limit": "30",
    "offset": "0",
    "total": "20",
    "RaceTable": {
      "season": "2008",
      "round": "5",
      "Races": [
        {
          "season": "2008",
          "round": "5",
          "url": "http://en.wikipedia.org/wiki/2008_Turkish_Grand_Prix",
          "raceName": "Turkish Grand Prix",
          "Circuit": "Istanbul Park"
        }
      ]
    }
  }
}

```

Ergast pozwala na szczegółowe zawężenie danych poprzez parametry w URL:

<http://ergast.com/api/f1/drivers/alonso/constructors/renault/results>

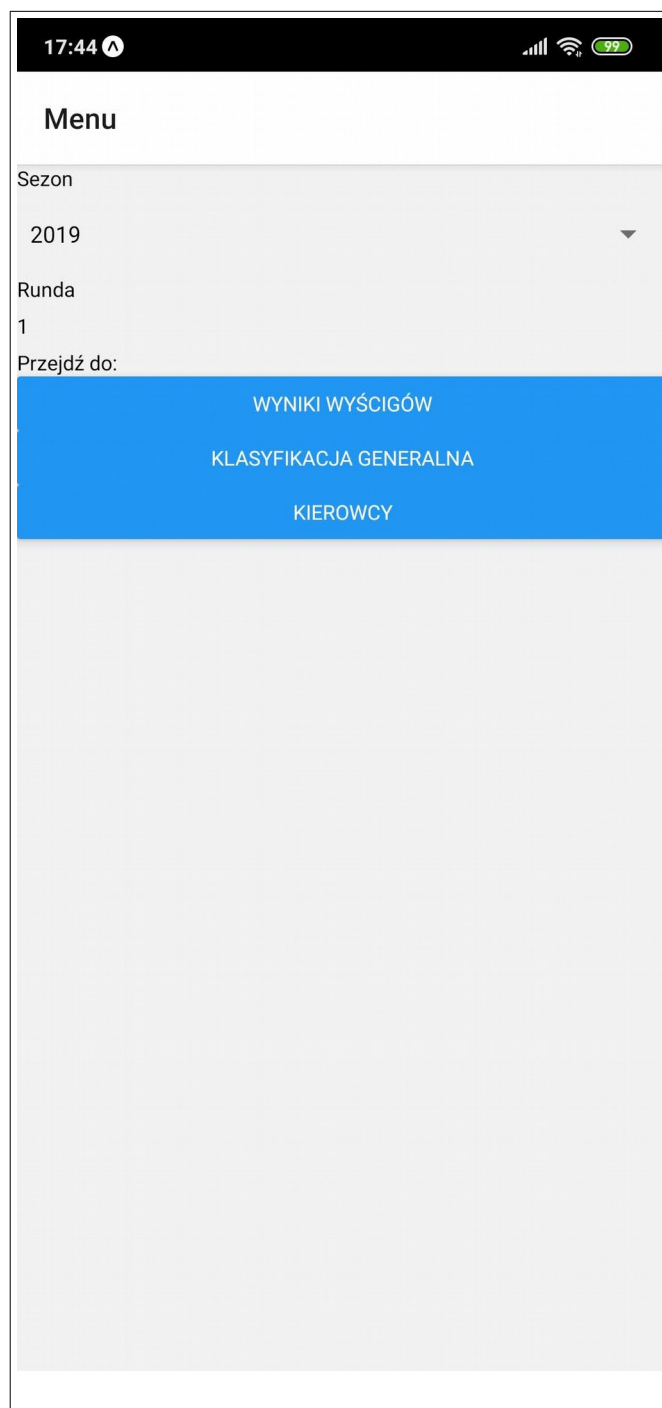
Do odczytu API użyto paczki NPM 'axios' - która jest nakładką na standardowe Fetch API.

Użyto komponentów FlatList, Picker, Picker.Item, TextInput, Text, Button oraz Touchable Opacity. Wszystkie są standardowymi komponentami dostępnymi w paczce 'react-native'.

Poza tym, wszystkie komponenty są wykonane przy użyciu React Hooks.

Zaimplementowano także wyszukiwanie elementów w FlatList (zadanie na dodatkowy punkt)

Listingi



Pierwszy widok aplikacji.

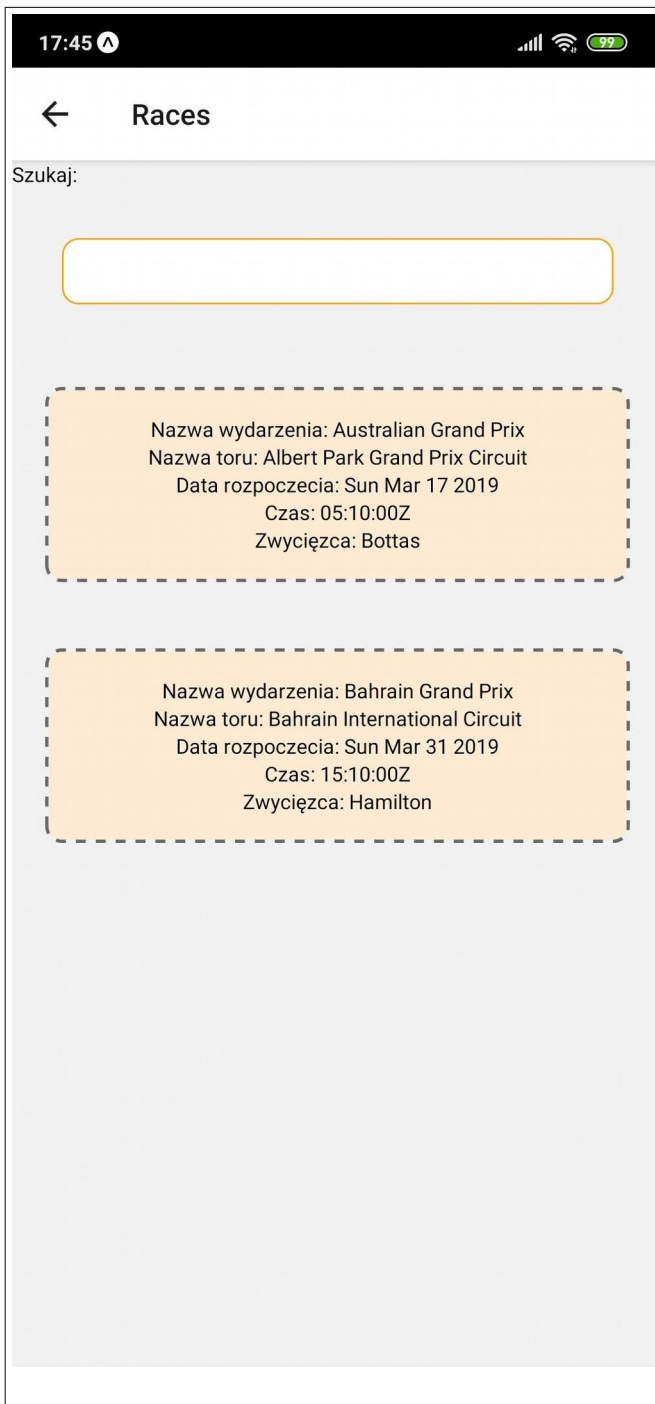
Sezony są zbudowane na bazie komponentu `<Picker>` - możliwe jest wybranie wartości od 1950 do 2019.

```
const [season, setSeason] =
  useState("2019");
const [round, setRound] = useState("1");
let seasons = [];
for (let i = 1950; i ≤ 2019; i++) {
  seasons.push(String(i));
}
```

```
<Picker
  selectedValue={season}
  onValueChange={setSeason}
>
  {seasons.map(season ⇒ (
    <Picker.Item
      label={season}
      value={season}
    />
  ))}
</Picker>
<Text>Runda</Text>
<TextInput
  onChangeText={setRound}
  value={round}
  keyboardType="number-pad"
/>
```



Wartości w <Picker>



Widok wyścigów w danej rundzie danego sezonu

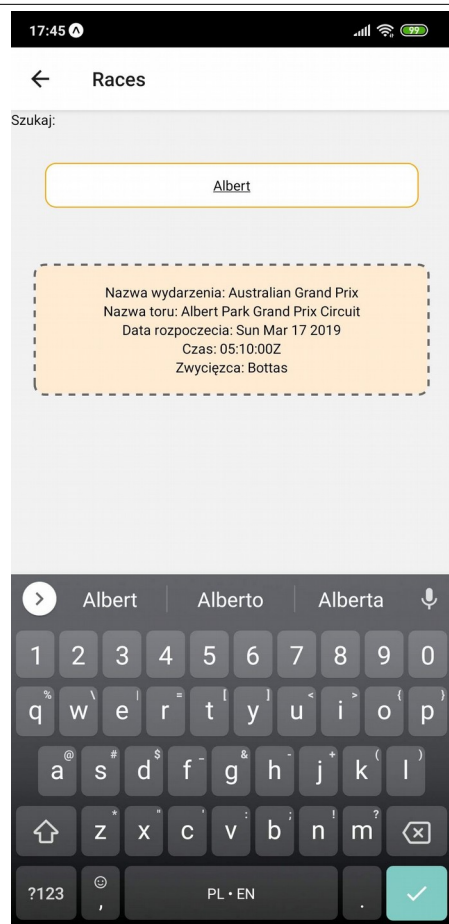
Funkcjonalność podstawowa

Dane są pobierane z poniższego endpointu Ergast:

```
return this.api
  .get(`${season}/results/${round}.json`)
  .then(({ data }) =>
    data.MRData.RaceTable.Races);
```

Główne komponenty (Item oraz funkcja wyszukująca “updateFilteredRaces” jest opisana na kolejnej stronie)

```
return (
  <View>
    <Text>Szukaj:</Text>
    <TextInput
      style={globalStyles.searchbox}
      value={search}
      onChangeText={text => {
        setSearch(text);
        updateFilteredRaces(text);
      }}
    />
    {races.length > 0 && (
      <FlatList
        data={racesFiltered}
        renderItem={
          data => Item({ data, navigation })
        }
        keyExtractor={({ round }) => round}
      />
    )}
  </View>
);
```

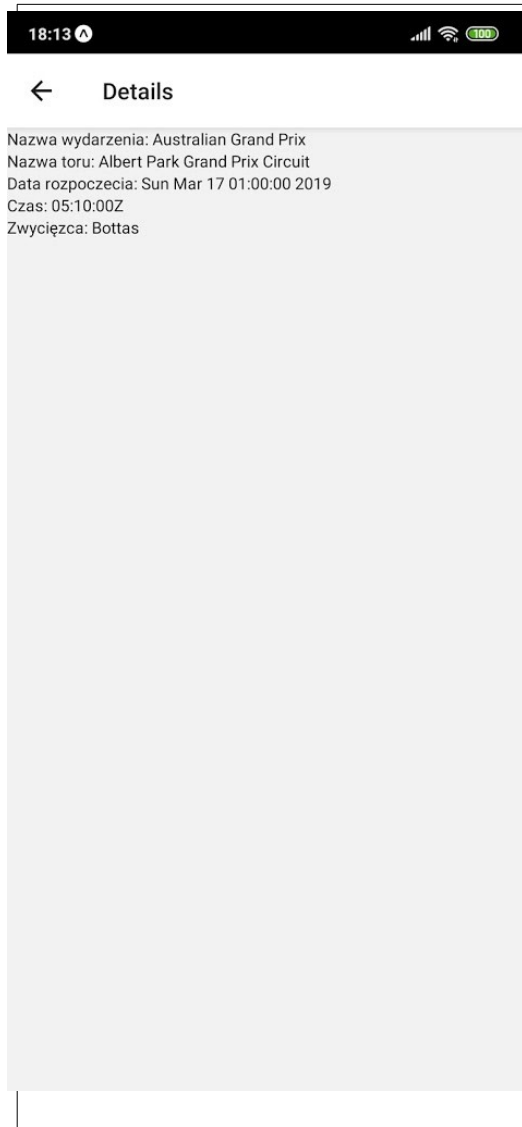


Widok wyścigów c.d Prezentacja działającego wyszukiwania

```
const updateFilteredRaces = searchTerm => {
const filtered = races.filter(race => {
//destrukuryzacja ze standardu ES6
const {
  raceName,
  Circuit: { circuitName }
} = race;
//wyszukiwanie bierze pod uwagę tylko nazwę trasy oraz
nazwę wydarzenia wyścigowego
if ([raceName, circuitName]
  .join("").includes(searchTerm))
  return true;
return false;
});
// gdy searchTerm === "" wyświetl wszystko
if (!searchTerm) {
  setRacesFiltered(races);
} else setRacesFiltered(filtered);
};
```

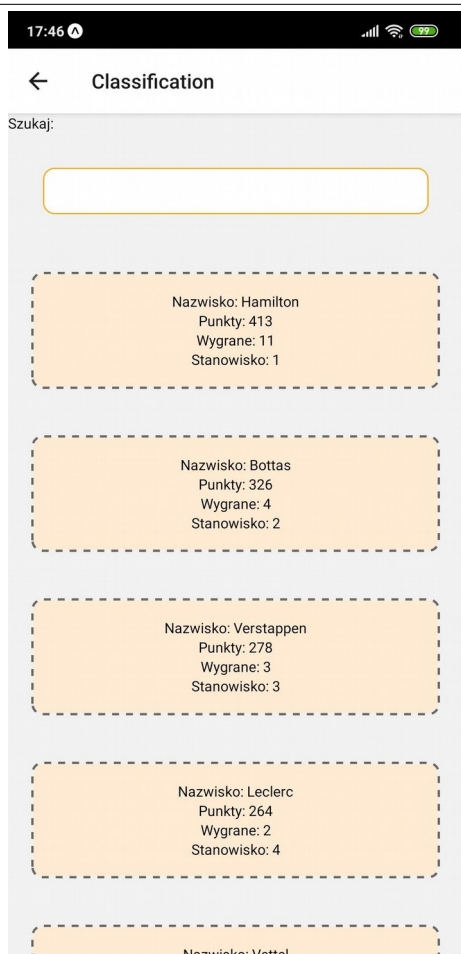
Definicja komponentu <Item>

```
return (
<TouchableOpacity
  style={globalStyles.TouchableOpacity}
  onPress={() => navigation.push("Details", { data })}
>
  <Text>Nazwa wydarzenia: {raceName}</Text>
  <Text>Nazwa toru: {circuitName}</Text>
  <Text>Data rozpoczęcia: {date}</Text>
  <Text>Czas: {time}</Text>
  <Text>Zwycięzca: {winner}</Text>
</TouchableOpacity>
);
```



Widok szczegółów dla wyniku wyścigów:

```
export const Details = ({
  route: {
    params: { data }
  }
}) => {
  console.log("details", data);
  const {
    raceName,
    Circuit: { circuitName },
    time,
    Results
  } = data.item;
  const date =
    new Date(data.item.date).toLocaleString();
  const winner = Results[0].Driver.familyName;
  return (
    <View>
      <Text>Nazwa wydarzenia:
        {raceName}
      </Text>
      <Text>
        Nazwa toru: {circuitName}
      </Text>
      <Text>
        Data rozpoczęcia: {date}
      </Text>
      <Text>Czas: {time}</Text>
      <Text>Zwycięzca: {winner}</Text>
    </View>
  );
};
```



Widok wyników końcowych danego sezonu wyścigowego (tj. Ranking kierowców)

Funkcjonalność dodatkowa (1 z 2)

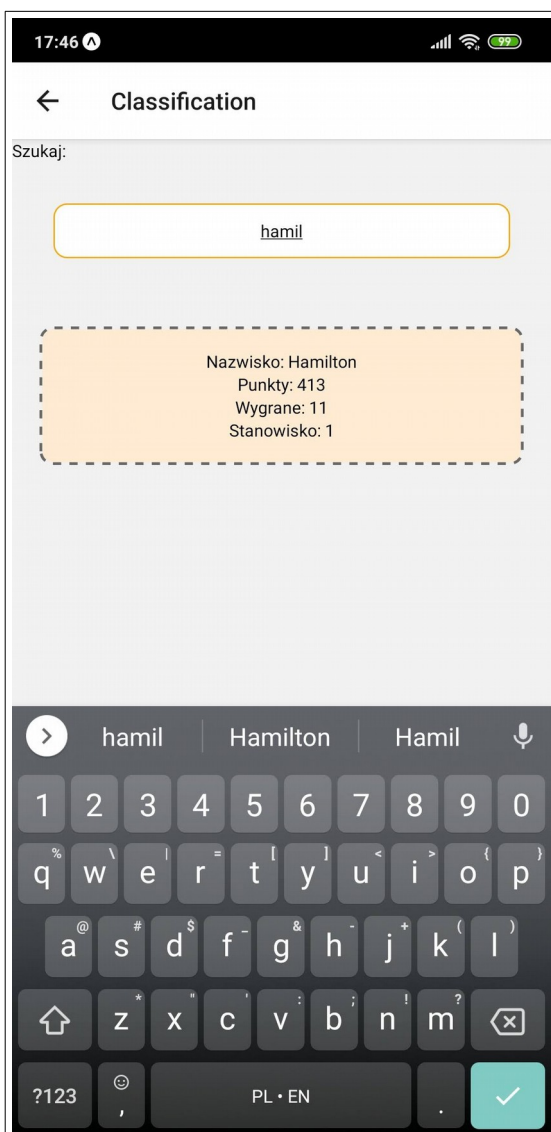
Dane są pobierane z API w następujący sposób:

```
getStandings(season = "current") {  
  return this.api.get(`  
    ${season}/driverStandings.json  
  `)  
  .then(({ data }) => {  
    console.log(data);  
  });  
}
```

//dosć głębokie zagnieżdzenie danych po stronie Ergast

```
return data.MRData  
  .StandingsTable.StandingsLists[0]  
  .DriverStandings;  
});  
}
```

Implementacja jest identyczna jak w przypadku poprzedniego widoku. Zmieniły się jedynie nazwy pól obiektów, które są używane w obiektach oraz nazwy pól które są używane podczas wyszukiwania.



Kod wyszukiwania kierowców:

```
const updateFilteredRaces = searchTerm => {
  console.log("ranking", races);
  const filtered = races.filter(driver => {
    const { familyName, givenName } = driver;
    // .toLowerCase() aby wyszukiwanie nie brało
    // pod uwagę wielkości liter (tj. 'hamilton'
    // vs 'Hamilton')
    if ([familyName, givenName]
      .join("")
      .toLowerCase()
      .includes(searchTerm)) return true;
    return false;
  });

  if (!searchTerm) {
    setRacesFiltered(races);
  } else setRacesFiltered(filtered);
};
```



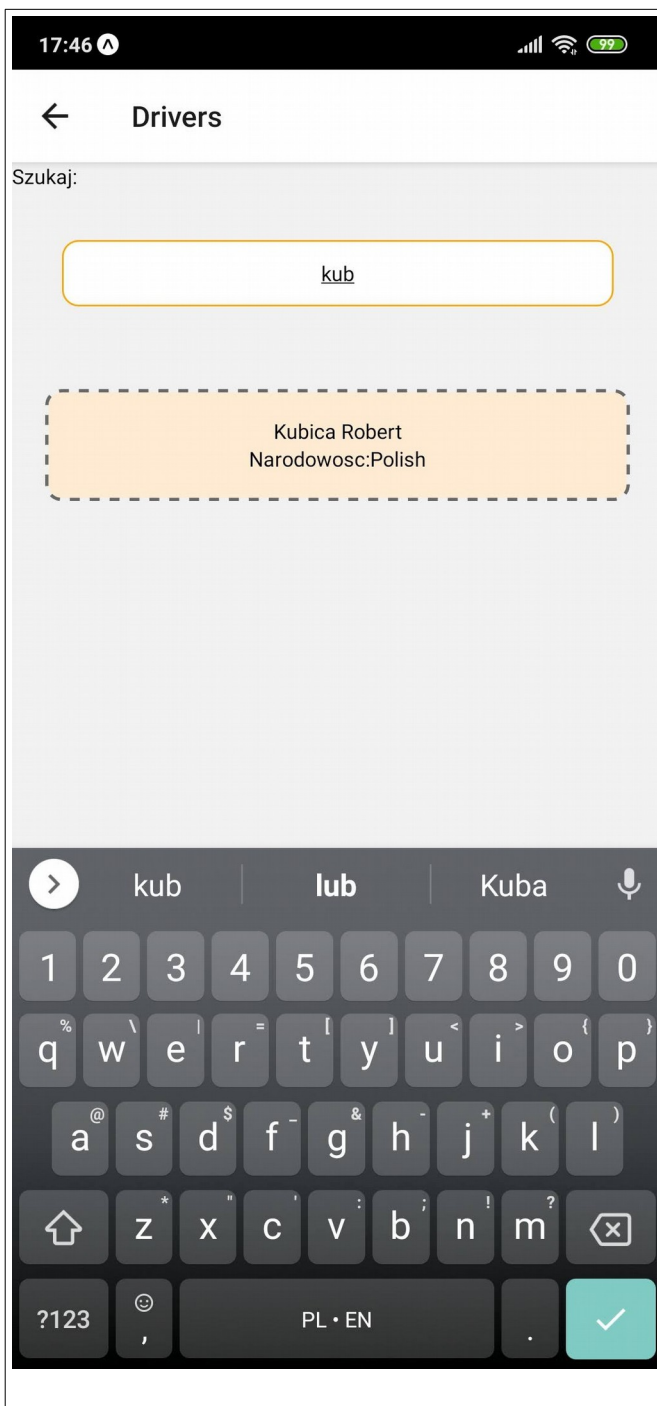
Ogólny widok wszystkich kierowców w danym sezonie Formuły 1.

Funkcjonalność dodatkowa (2 z 2)

Implementacja jest identyczna jak w przypadku poprzednich widoków. Zmienia się jedynie funkcja szukająca.

```
if ([familyName, givenName]
    .join("")
    .toLowerCase()
    .includes(searchTerm)
) return true;
```

Funkcja bierze pod uwagę imię oraz nazwisko kierowcy.



Funkcja szukająca w przypadku widoku kierowców

```

import axios from "axios";
class API {
  api = axios.create({
    baseUrl: "http://ergast.com/api/f1/"
  });

  /**
   *
   */
  getRaces(season = "current", round = "") {
    if (!round)
      return this.api.get(`${season}/results.json`).then(({
        data
      }) => {
        return data.MRData.RaceTable.Races;
      });
    // can also do /results/1.json
    else
      return this.api
        .get(`${season}/results/${round}.json`)
        .then(({
          data
        }) => data.MRData.RaceTable.Races);
  }

  getStandings(season = "current") {
    return this.api.get(`${season}/driverStandings.json`).then(({
      data
    }) => {
      console.log(data);
      return data.MRData.StandingsTable.StandingsLists[0].DriverStandings;
    });
  }

  getDrivers(season = "current") {
    return this.api.get(`${season}/drivers.json`).then(({
      data
    }) => {
      console.log(data);
      return data.MRData.DriverTable.Drivers;
    });
  }
}

export const ErgastAPI = new API();

```