

EXP 2
8/7/24

Working with APIs

AIM:

To complete the,

- a) Guided projects in IBM skills network lab:
 - Building RESTful APIs with Express
 - Working with simple APIs
- b) Building of a RESTful API and testing it with the Postman tool.
- c) To consume a third party API in a webpage.

PROCEDURE:

- a) Guided projects in IBM skills network lab:
 - Building RESTful APIs with Express.

[Course](#)
[Progress](#)
[Dates](#)
[Discussion](#)

Working with Simple APIs

Pick up where you left off

Resume course

Course Tools

Bookmarks

Expand all

Working with Simple APIs: Random User and Fruitvice API Examples

Overview 1 min

Lab: Simple APIs 1 min + 1 activity

Enter ID (or leave blank t)

FETCH

Name
Country of Origin
Heat

ADD

Enter ID
Name
Country of Origin
Heat

UPDATE

Enter ID

REMOVE

Status :

- Working with simple APIs

Course Progress Dates Discussion

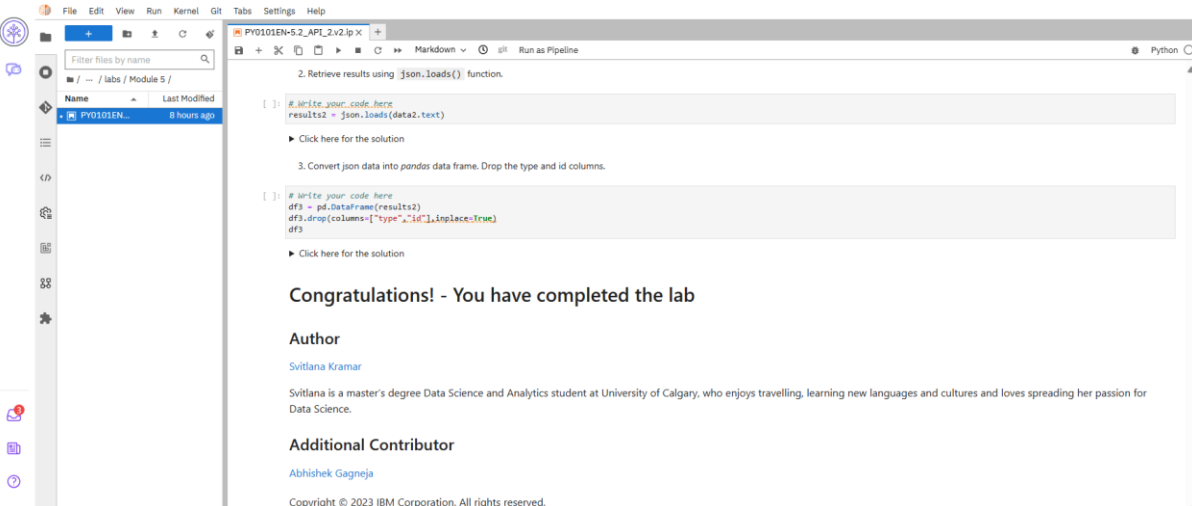
Working with Simple APIs

Pick up where you left off [Resume course](#) [Course Tools](#)
[Bookmarks](#)

[Expand all](#)

- ✓ Working with Simple APIs: Random User and Fruitvice API Examples
- ✓ Overview 1 min
- ✓ Lab: Simple APIs 1 min + 1 activity

COGNITIVE CLASS



2. Retrieve results using `json.loads()` function.

```
[ ]: # Write your code here
results2 = json.loads(data2.text)
```

▶ Click here for the solution

3. Convert json data into pandas data frame. Drop the type and id columns.

```
[ ]: # Write your code here
df3 = pd.DataFrame(results2)
df3.drop(columns=["type", "id"], inplace=True)
df3
```

▶ Click here for the solution

Congratulations! - You have completed the lab

Author
 Svitlana Kramar
 Svitlana is a master's degree Data Science and Analytics student at University of Calgary, who enjoys travelling, learning new languages and cultures and loves spreading her passion for Data Science.

Additional Contributor
 Abhishek Gagneja

Copyright © 2023 IBM Corporation. All rights reserved.

b) Building of a RESTful API and testing it with the Postman tool.

- Download Nodejs Installer: <https://nodejs.org/en/download/prebuilt-installer> and complete the setup.

node Learn About Download Blog Docs Certification

Download Node.js®

Download Node.js the way you want.

Package Manager **Prebuilt Installer** Prebuilt Binaries Source Code

I want the **v20.15.1 (LTS)** version of Node.js for **Windows** running **x64**

[Download Node.js v20.15.1](#)

Node.js includes **npm (10.7.0)**

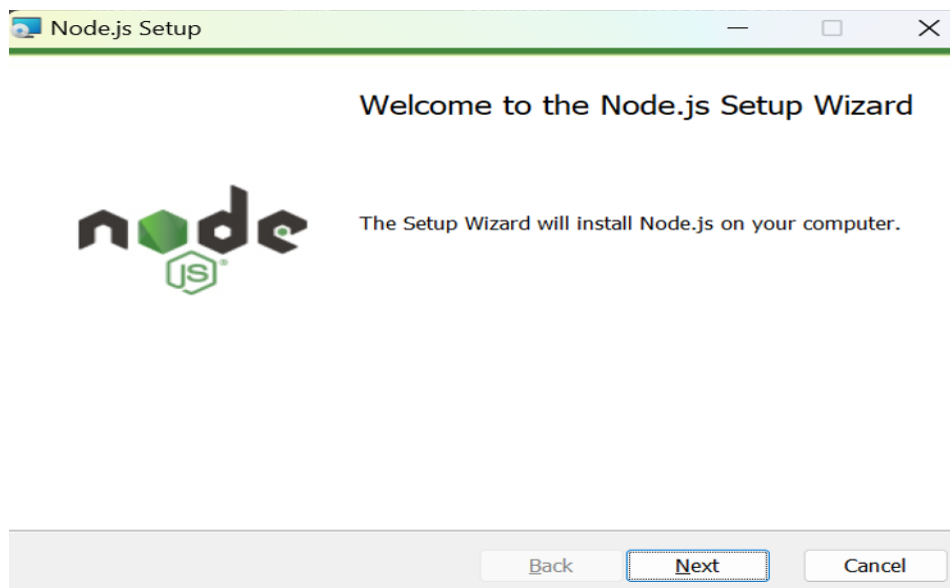
[Read the changelog for this version](#)

[Read the blog post for this version](#)

[Learn how to verify signed SHASUMS](#)

[Check out all available Node.js download options](#)

[Learn about Node.js Releases](#)

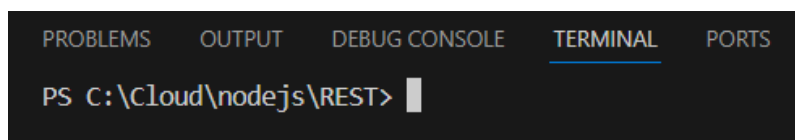


- Check node installation by using following commands:

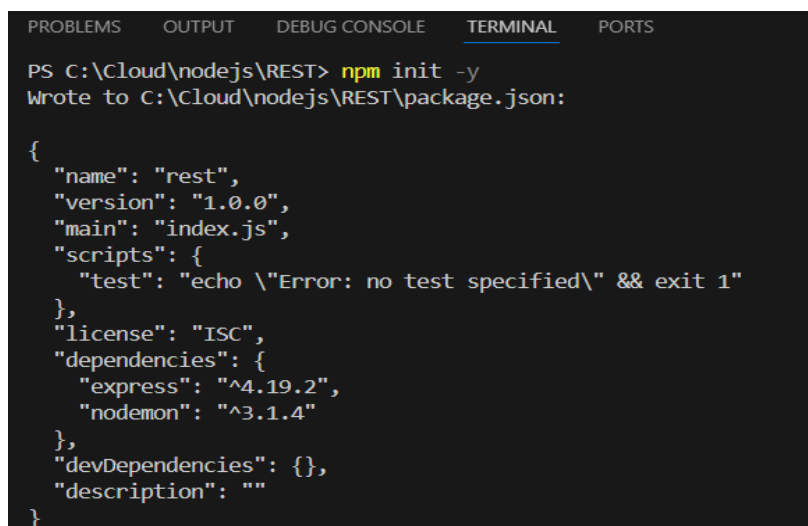
```
C:\Cloud>node -v
v20.15.1

C:\Cloud>npm -v
10.7.0
```

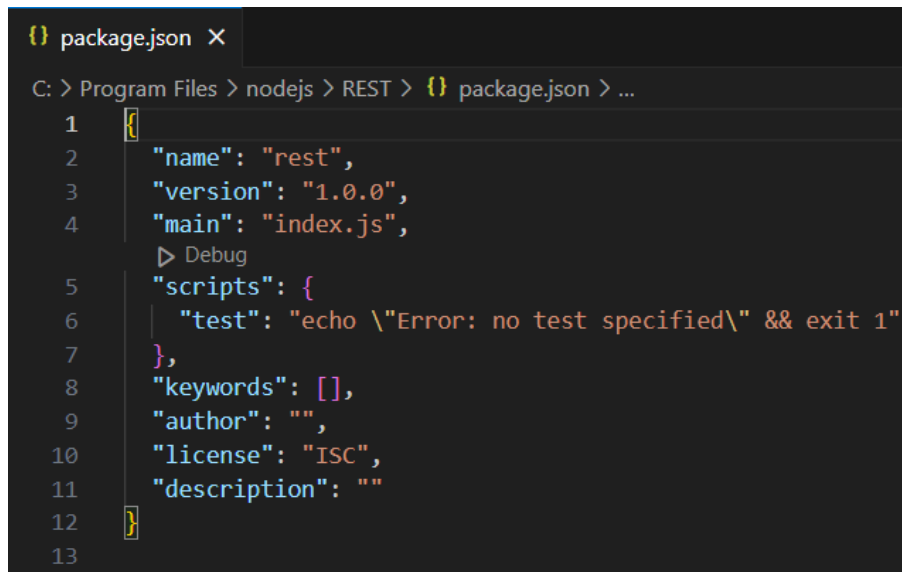
- Create a folder for your Project – REST and make it your current working directory.



- Initialize your project using the command : **npm init -y**



- This creates a package.json file in the project folder.

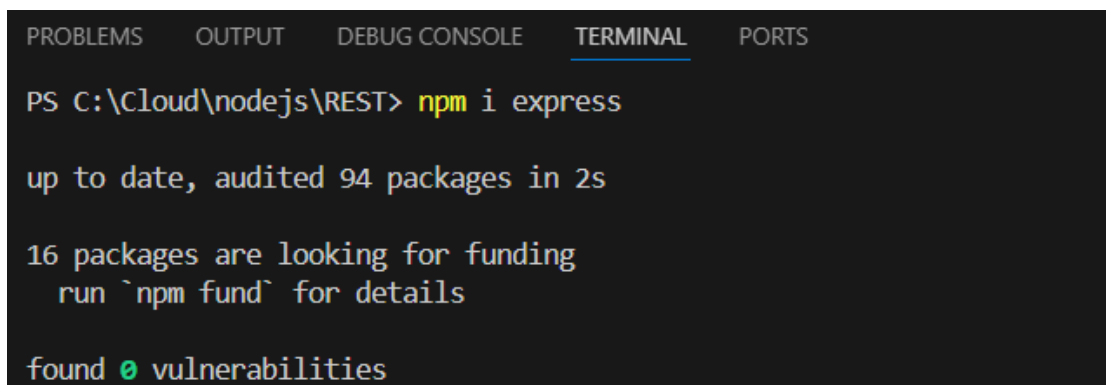


```

{} package.json X
C: > Program Files > nodejs > REST > {} package.json > ...
1  {
2    "name": "rest",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "keywords": [],
9    "author": "",
10   "license": "ISC",
11   "description": ""
12 }
13

```

- Install Express using the command : **npm install express**



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Cloud\nodejs\REST> npm i express

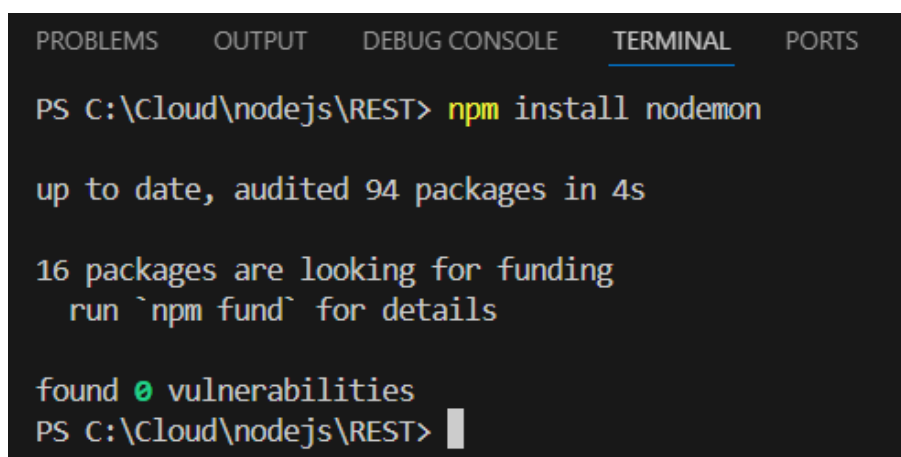
up to date, audited 94 packages in 2s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

- Install nodemon using the command : **npm install nodemon**



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Cloud\nodejs\REST> npm install nodemon

up to date, audited 94 packages in 4s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Cloud\nodejs\REST>

```

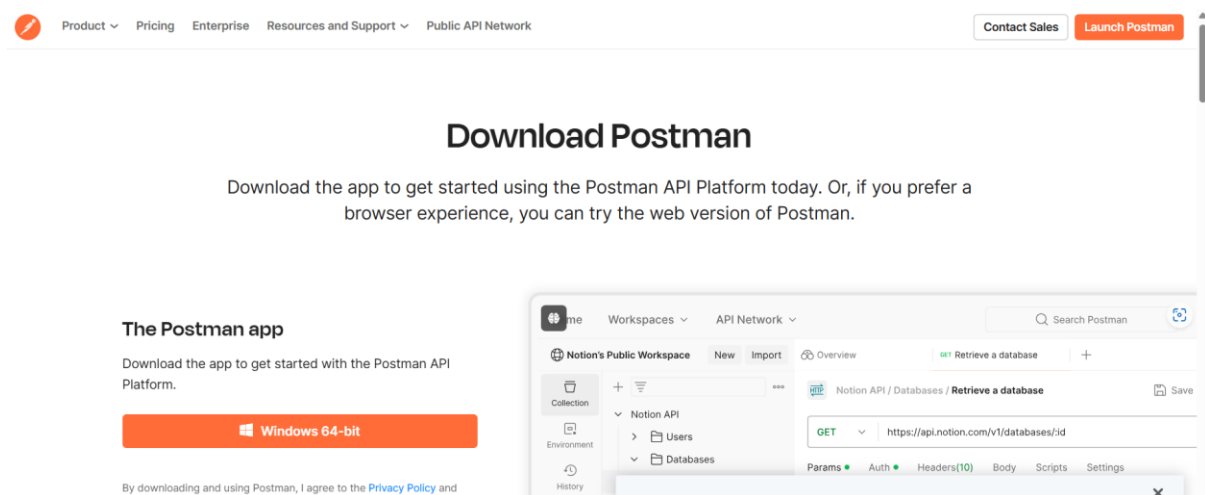
- Run a simple server.js code for testing.

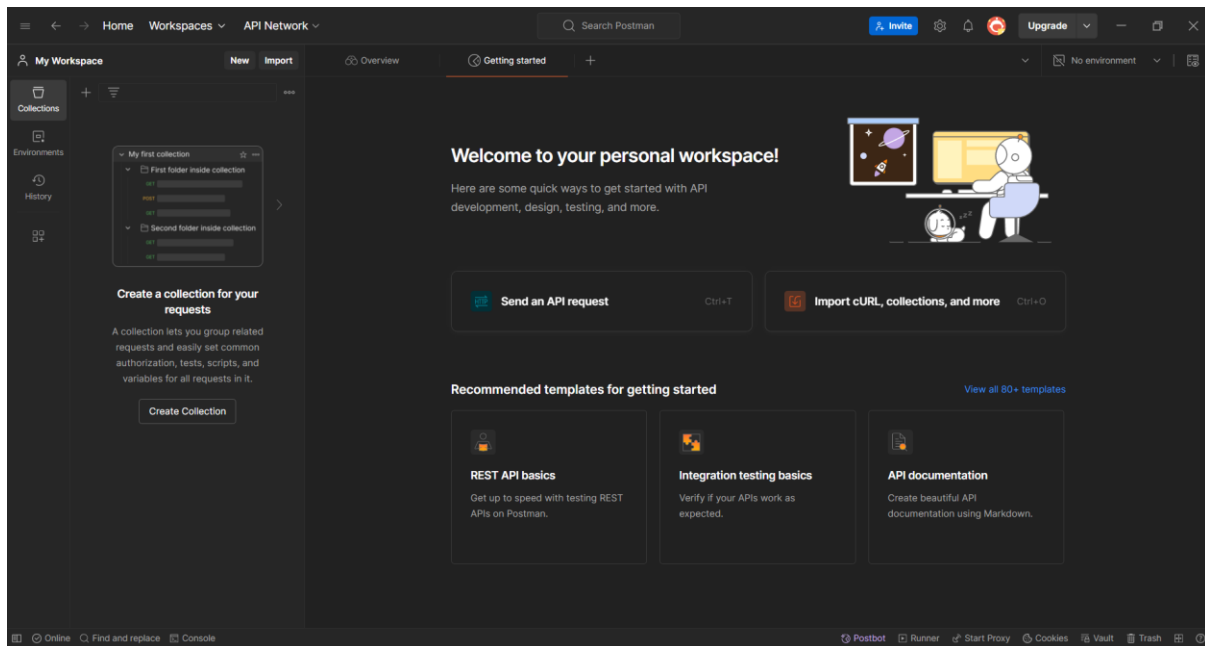
```
JS server.js > ...
1  const http = require('http');
2  const server = http.createServer((req, res) => {
3    res.statusCode = 200;
4    res.setHeader('Content-Type', 'text/plain');
5    res.end('Have a good day');
6  });
7  const port = 3000;
8  server.listen(port, () => {
9    console.log(`Server running at http://localhost:${port}/`);
10 });
11
```

```
PS C:\Cloud\nodejs\REST> node server
Server running at http://localhost:3000/
```

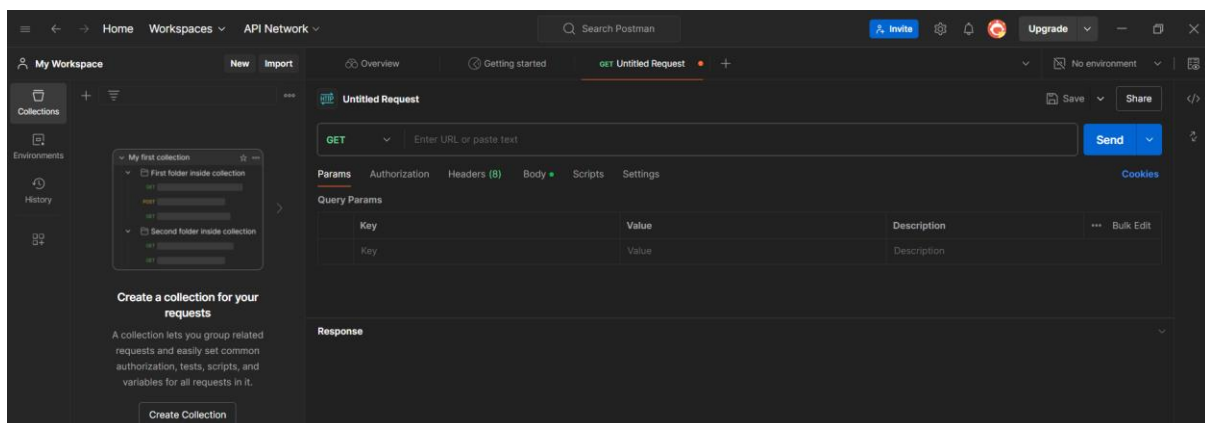


- Install Postman tool: <https://www.postman.com/downloads/> and open it.





- Click **Send an API request**.



- Do code for the Football players api and save it on your project directory.

CODE:

football.js

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 5000;

let players = [
  { id: 1, name: 'Lionel Messi', nationality: 'Argentina', position: 'Attacker', goals: 700 },
  { id: 2, name: 'Cristiano Ronaldo', nationality: 'Portugal', position: 'Attacker', goals: 800 },
  { id: 3, name: 'Neymar jr', nationality: 'Brazil', position: 'Attacker', goals: 400 },
  { id: 4, name: 'David Beckham', nationality: 'England', position: 'Midfielder', goals: 200 },
  { id: 5, name: 'Sergio ramos', nationality: 'Spain', position: 'Defender', goals: 800 }
];

app.use(bodyParser.json());
```

```

app.get('/players', (req, res) => {
  res.json(players);
});

app.get('/players/:id', (req, res) => {
  const playerId = parseInt(req.params.id);
  const player = players.find(player => player.id === playerId);
  if (player) {
    res.json(player);
  } else {
    res.status(404).send('Player not found');
  }
});

app.post('/players', (req, res) => {
  const newPlayer = req.body;
  if (!newPlayer || !newPlayer.name || !newPlayer.nationality || !newPlayer.position ||
!newPlayer.goals) {
    return res.status(400).send('Missing required fields');
  }

  newPlayer.id = Math.max(...players.map(player => player.id)) + 1;
  players.push(newPlayer);
  res.status(201).json(newPlayer);
});

app.put('/players/:id', (req, res) => {
  const playerId = parseInt(req.params.id);
  const updatedPlayer = req.body;
  const playerIndex = players.findIndex(player => player.id === playerId);

  if (playerIndex !== -1) {
    players[playerIndex] = { ...players[playerIndex], ...updatedPlayer };
    res.json(players[playerIndex]);
  } else {
    res.status(404).send('Player not found');
  }
});

app.delete('/players/:id', (req, res) => {
  const playerId = parseInt(req.params.id);
  const playerIndex = players.findIndex(player => player.id === playerId);

  if (playerIndex !== -1) {
    players.splice(playerIndex, 1);
    res.sendStatus(204);
  } else {
    res.status(404).send('Player not found');
  }
});

```

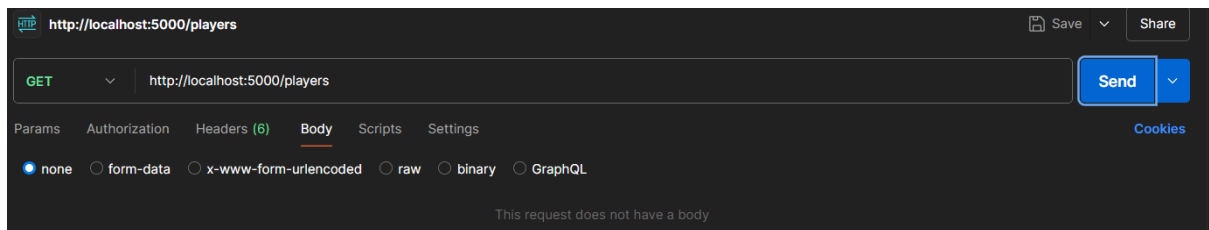
```
}
});
```

```
app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

- Run the code football.js (server)

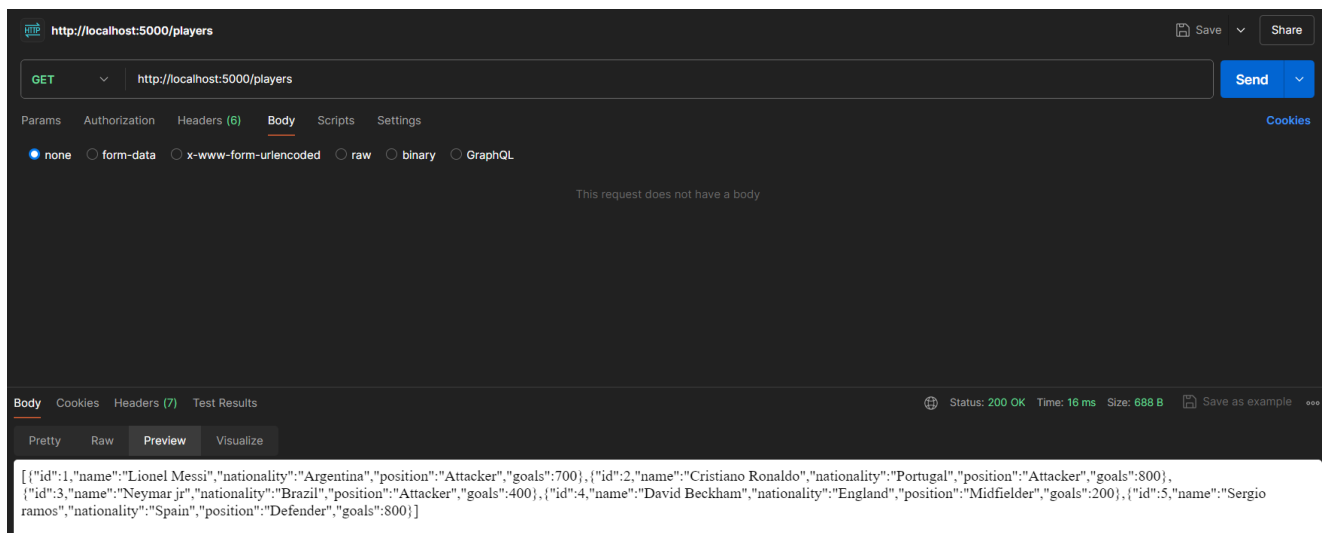
```
PS C:\Cloud\nodejs\REST> node football.js
Server listening on port 5000
```

- Enter the URL of your server **http://localhost:5000**, the port where your API code is running. And in the address bar and append **/players** to it.

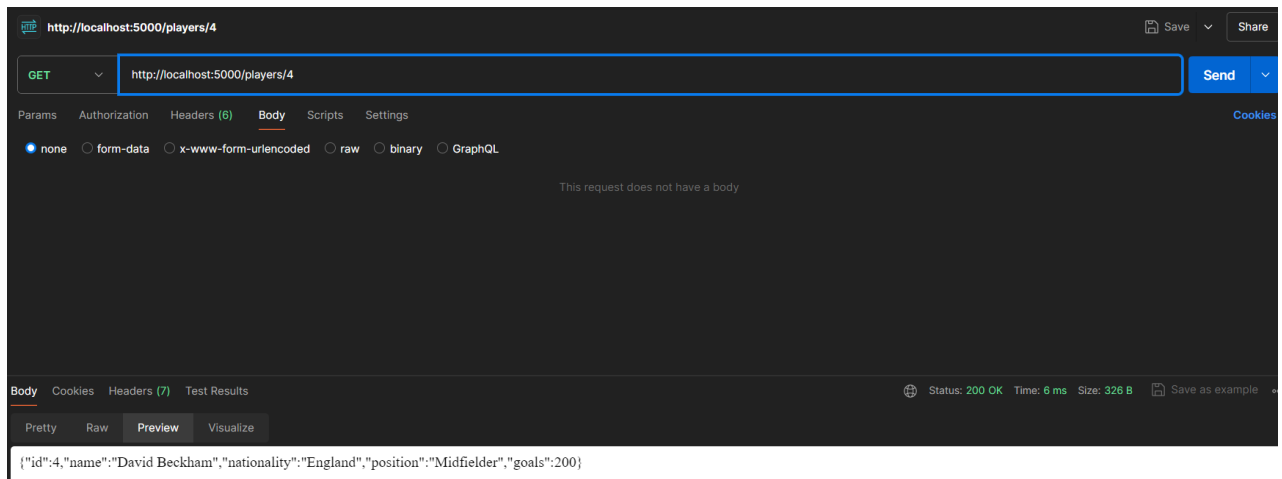


GET:

- Select request method as GET and click send. This gives details about all players.

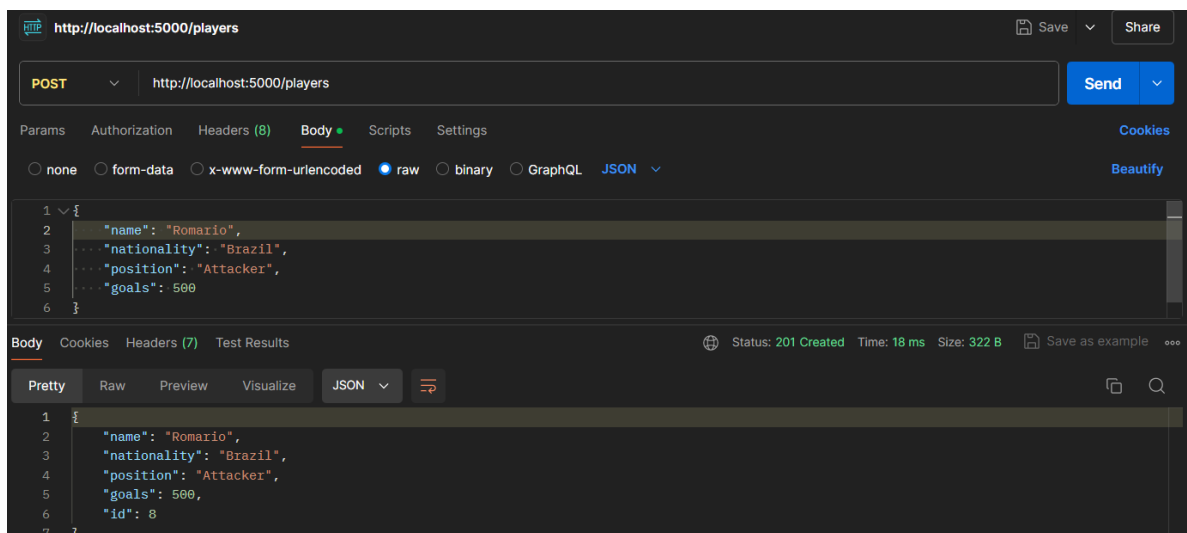


- Append **/id** to the url to get details about the player with particular id.



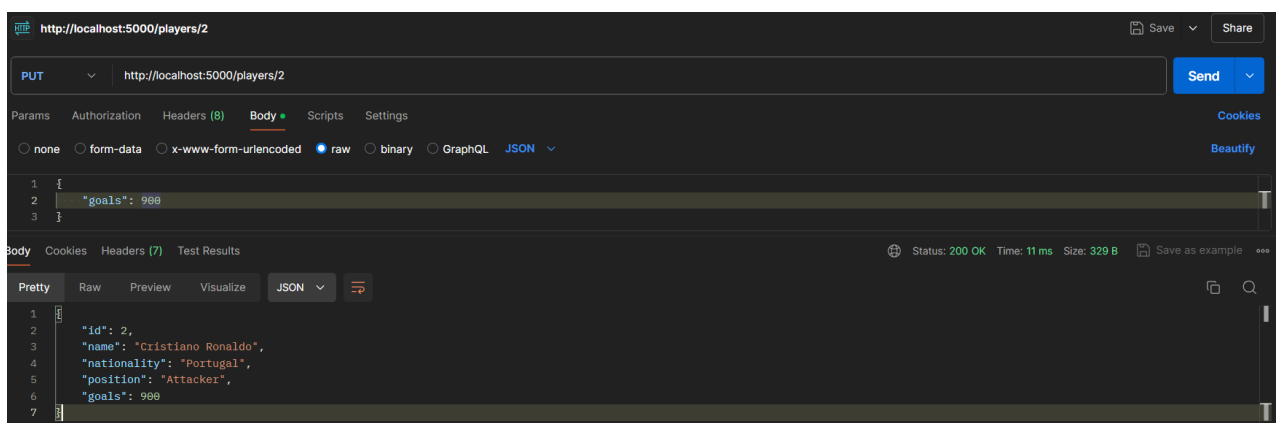
POST:

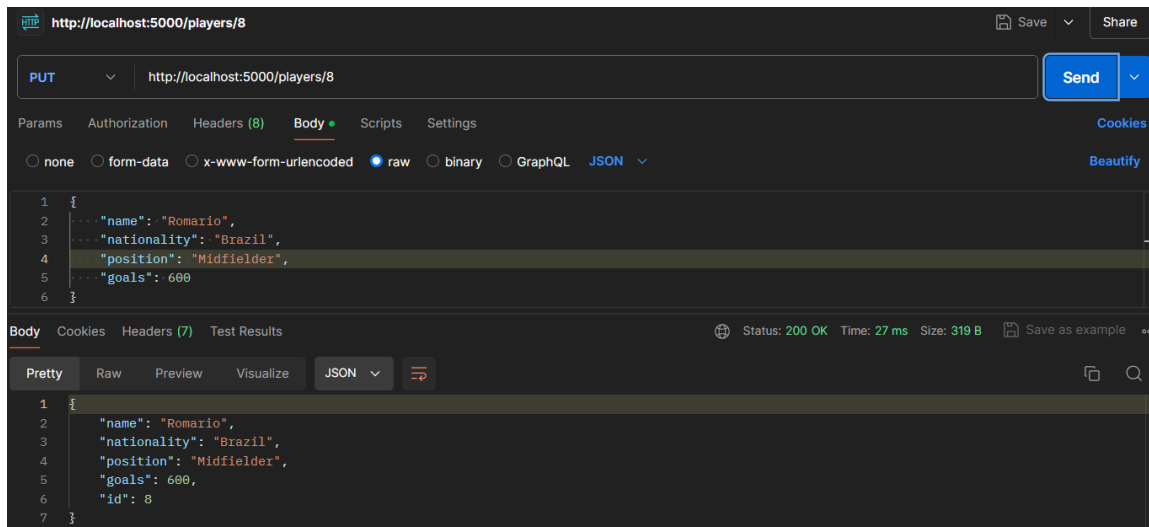
- Select request method as POST ,enter the details of new player in the body and click send. It adds the new user details to the server.



PUT:

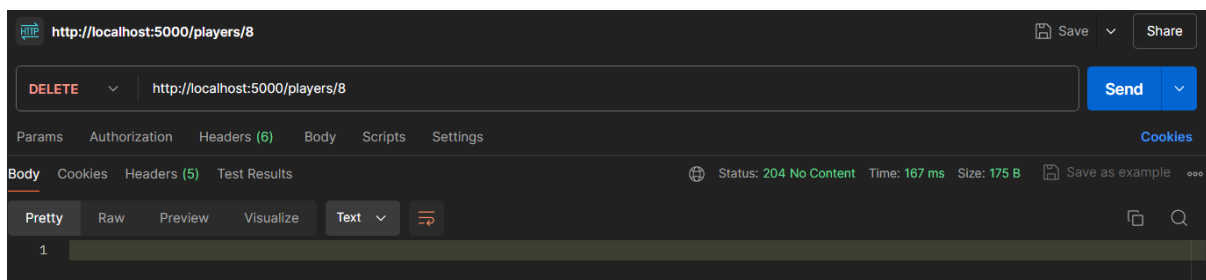
- Select request method as PUT ,specify the id to update on the URL, enter the details to update in the body and click send. It updates the details.





DELETE:

- Select request method as DELETE ,specify the id to delete on the URL and click send. It deletes the respective id details.



c) To consume a third party API in a webpage.

CODE: quotes.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Random Ron Swanson Quote</title>
</head>
<body>
  <h1>Ron Swanson Wisdom</h1>
  <button id="get-quote-button">Get Random Quote</button>
  <p id="quote-display"></p>
```

```
<script src="quotes.js">
</script>
</body>
</html>
```

quotes.js

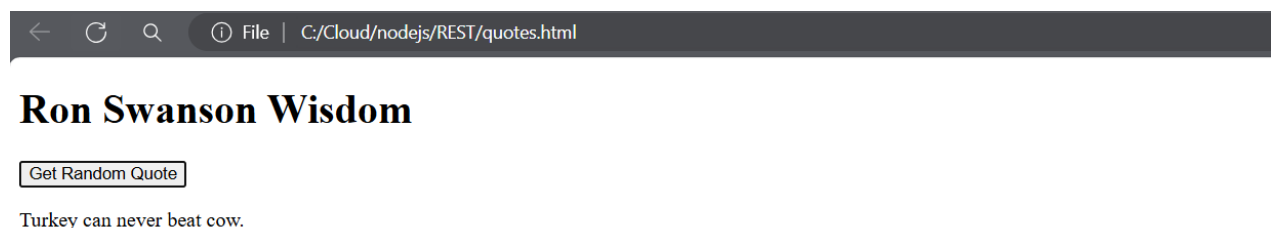
```
const quoteButton = document.getElementById('get-quote-button');
const quoteDisplay = document.getElementById('quote-display');

quoteButton.addEventListener('click', getQuote);

function getQuote() {
  fetch('https://ron-swanson-quotes.herokuapp.com/v2/quotes')
    .then(response => response.json())
    .then(data => {
      const quote = data[0];
      quoteDisplay.textContent = quote;
    })

    .catch(error => {
      console.error('Error fetching quote:', error);
      quoteDisplay.textContent = 'Error: Could not retrieve quote.';
    });
}
```

OUTPUT:



RESULT:

Thus, the guided projects in IBM skills network lab: Building RESTful APIs with Express and Working with simple APIs, Building of a RESTful API and testing it with the Postman tool and consuming a third party API in a webpage have been completed successfully and output have been verified.