Node.JS and PostgreSQL Example Application Guide

Abdülkadir Çakır

08/12/2019

Introduction

This guide will tell you about how to create sample application in Node.js. In this guide we will follow these steps.

1. Creating *Node.js* package
2. Creating *Node.js* environment
3. Connecting *PostgreSQL* database
4. Accessing database through *JavaScript*
5. Displaying and inserting data to database.

But first, what is JavaScript?

JavaScript is widely used dynamical programming language in Web applications.

Is it something like Java?

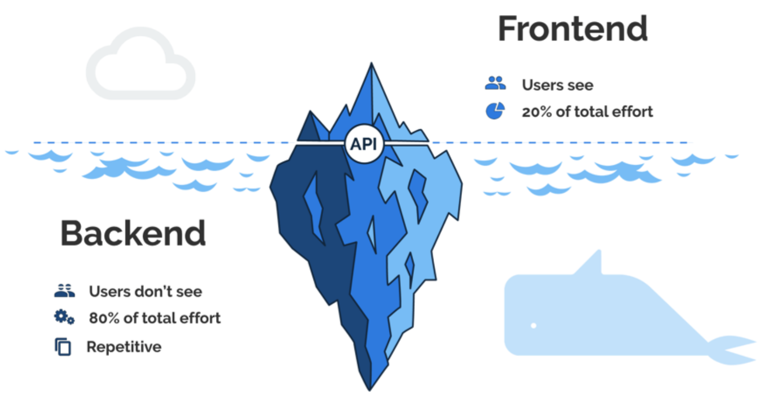
No. The relation between Java and JavaScript is like car and carpet. Their coding syntaxes and interpretation logic has lots of differences.

What is *node.js*?

Node.js (**Node**) is an open source development platform for executing JavaScript code server-side(backend).

In our project we will use node.js as our backend development platform. It makes project codes invisible to user. We will use this term “backend” in our project very often.

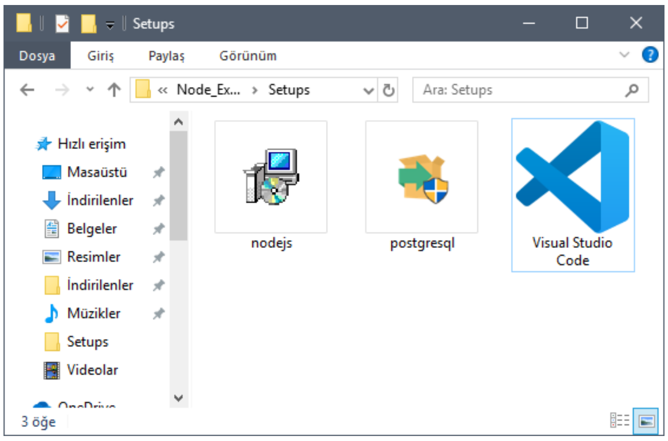
What is frontend and backend?



As illustrated in image, Frontend is visible part of that iceberg. Iceberg symbolizes an application. Users don't care about backend. They look for frontend. As a coder you should take care about both or you need to be master for one of them. If you choose to be frontend then you design how your application will look like. And you will be called as "frontend developer". Or else you will be called as "backend developer". Now we are ready to start to our journey in JavaScript and SQL queries.

This guide supposes that you use Windows in your PC. If you determined to use another operating system, hope you will make your own way. Because this guide will only tell you only what to do in Windows to make these codes come to life.

Step 1 : Installing Required Tools

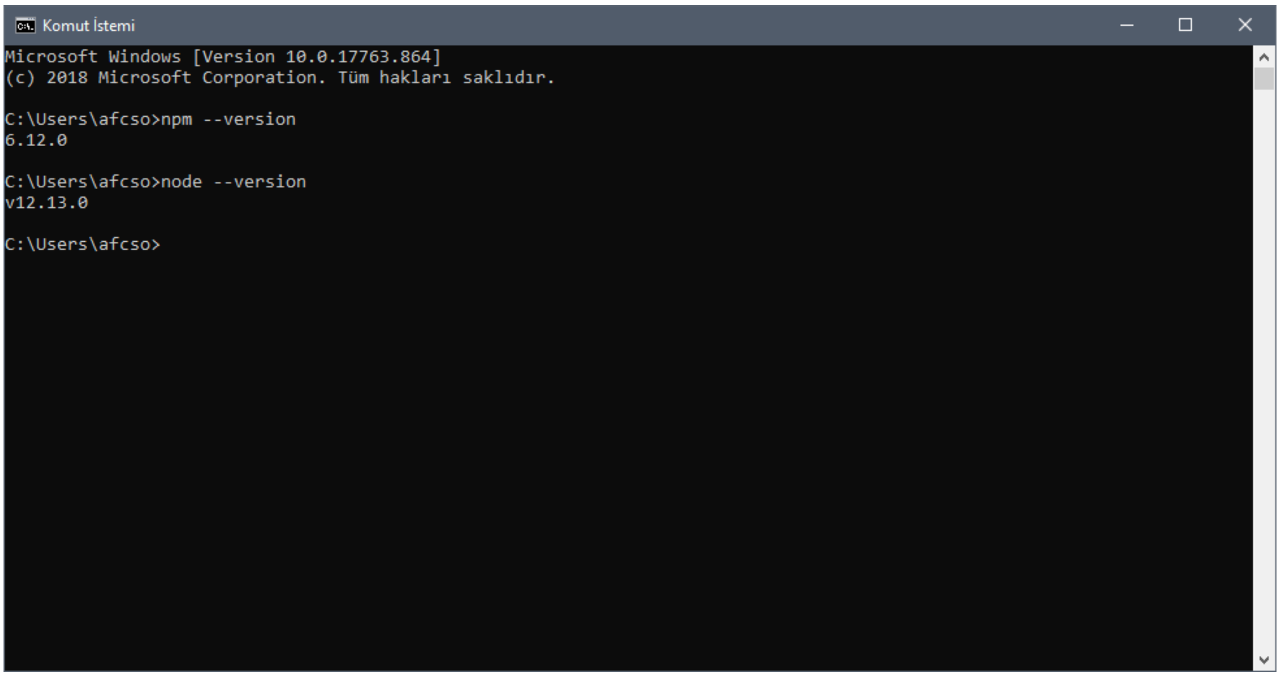


We need few tools to build our project.

- Node.js
- PostgreSQL
  - *Visual Studio Code

*: Visual Studio Code is recommended you can use your favorite text editor too.

Step 2: Check for installs



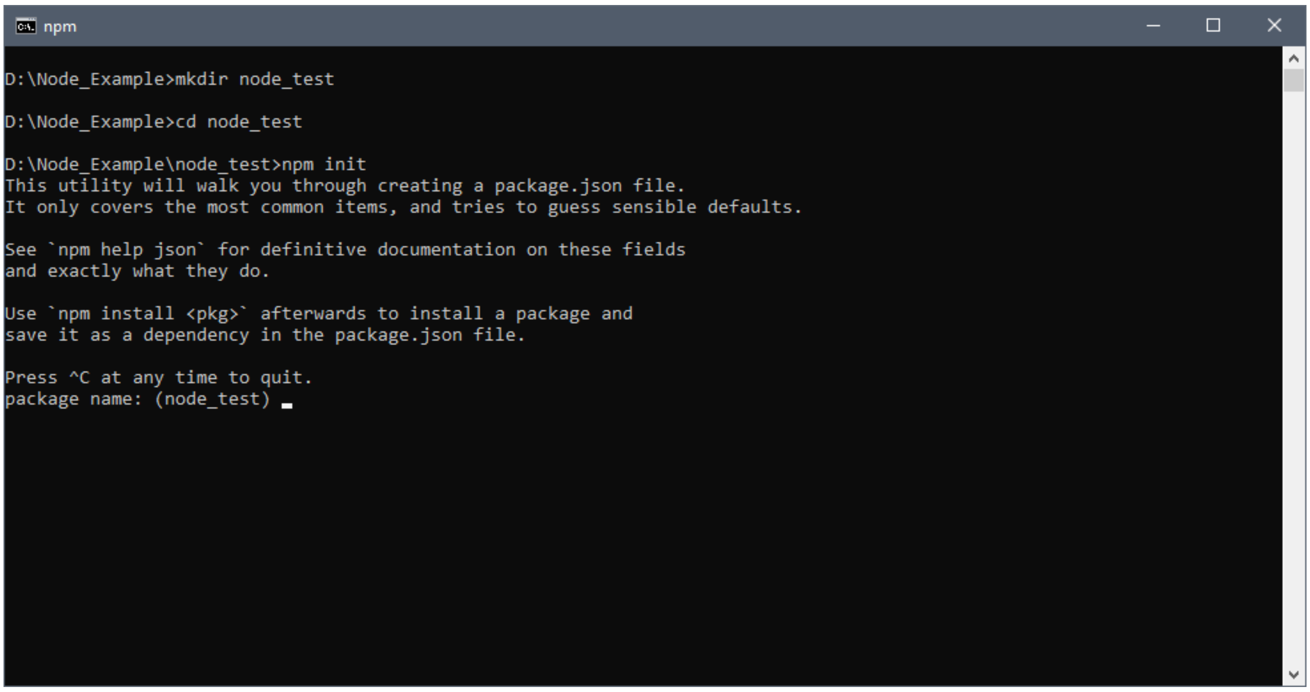
```
Microsoft Windows [Version 10.0.17763.864]
(c) 2018 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\afcso>npm --version
6.12.0

C:\Users\afcso>node --version
v12.13.0

C:\Users\afcso>
```

You need make sure that you have successful node.js installation before going further. You can simply check it using these two commands:

- npm --version
- node --version

If they both output their versions, then it is OK to go.

Step 3: Creating Package


npm

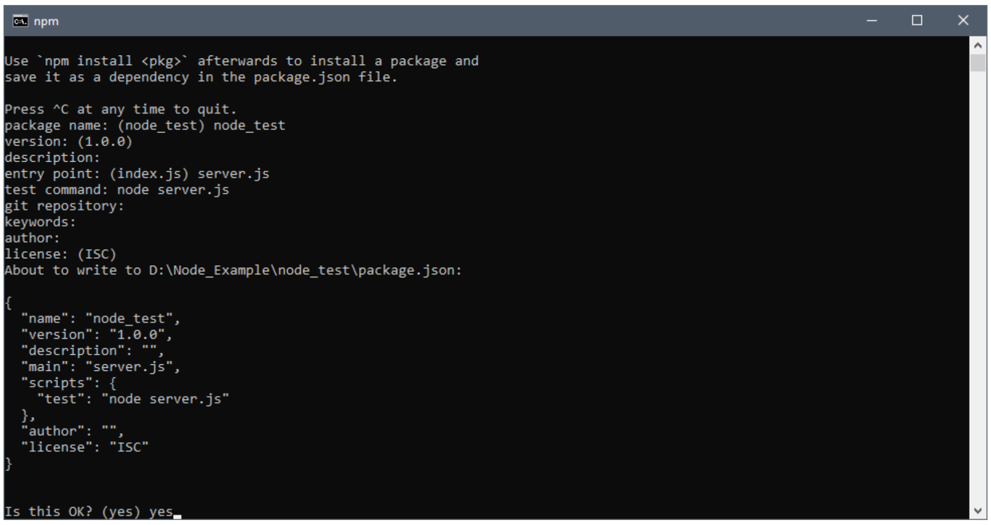
```
D:\Node_Example>mkdir node_test

D:\Node_Example>cd node_test

D:\Node_Example\node_test>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (node_test)
```

Now we need to create our application (package). We open a command line (Start Menu > type "**cmd**"). Direct the path of command line to where you want your project in (using "**cd**" command). You can crate directory using "**mkdir**".

Now we can run actual command that creates our package.

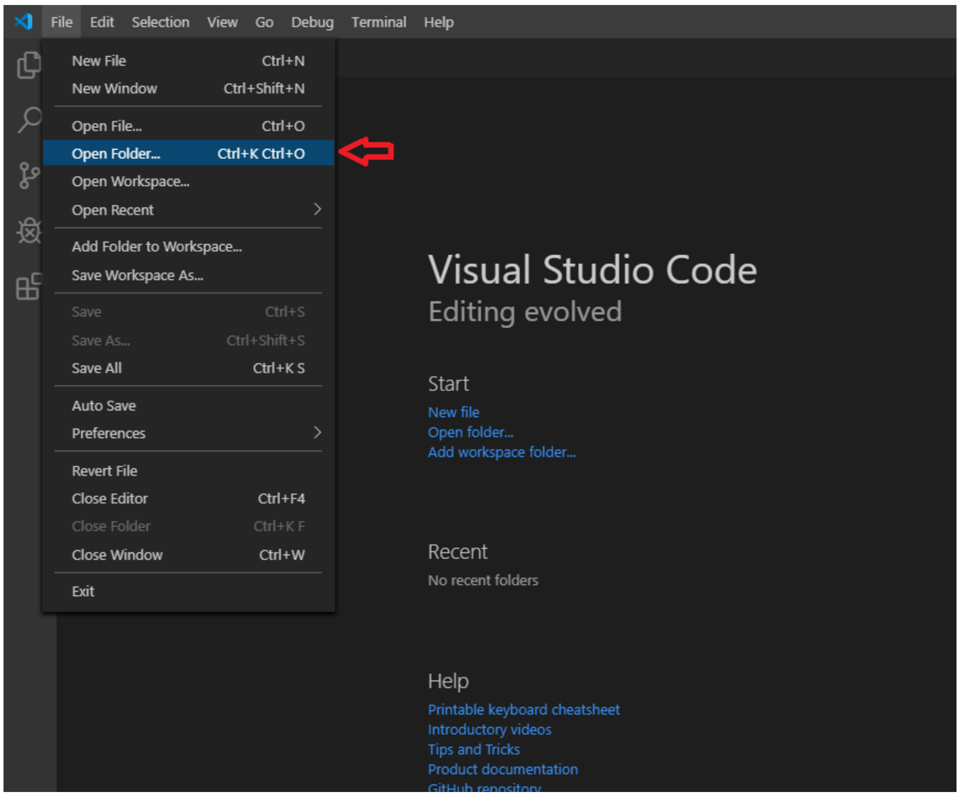
npm init

This command utility will ask you questions through the process. Like package name version etc. You can skip questions by simply pressing enter button. But there is an important question asked by npm. Entry Point. This is important for future use. We will type that "**server.js**" to this question.


npm

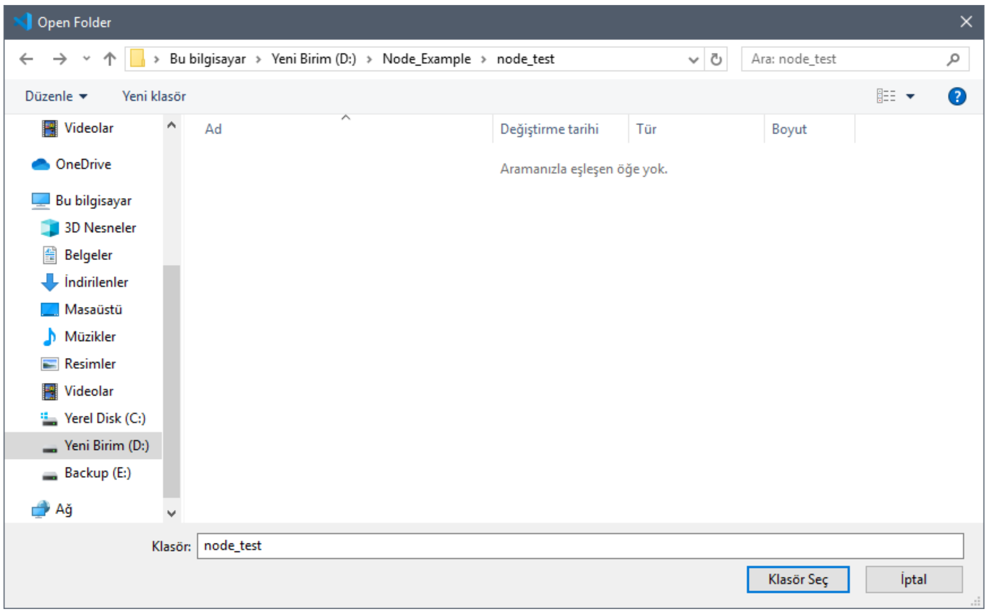
```
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (node_test) node_test
version: (1.0.0)
description:
entry point: (index.js) server.js
test command: node server.js
git repository:
keywords:
author:
license: (ISC)
About to write to D:\Node_Example\node_test\package.json:

{
  "name": "node_test",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "node server.js"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

Step 4: Creating Environment



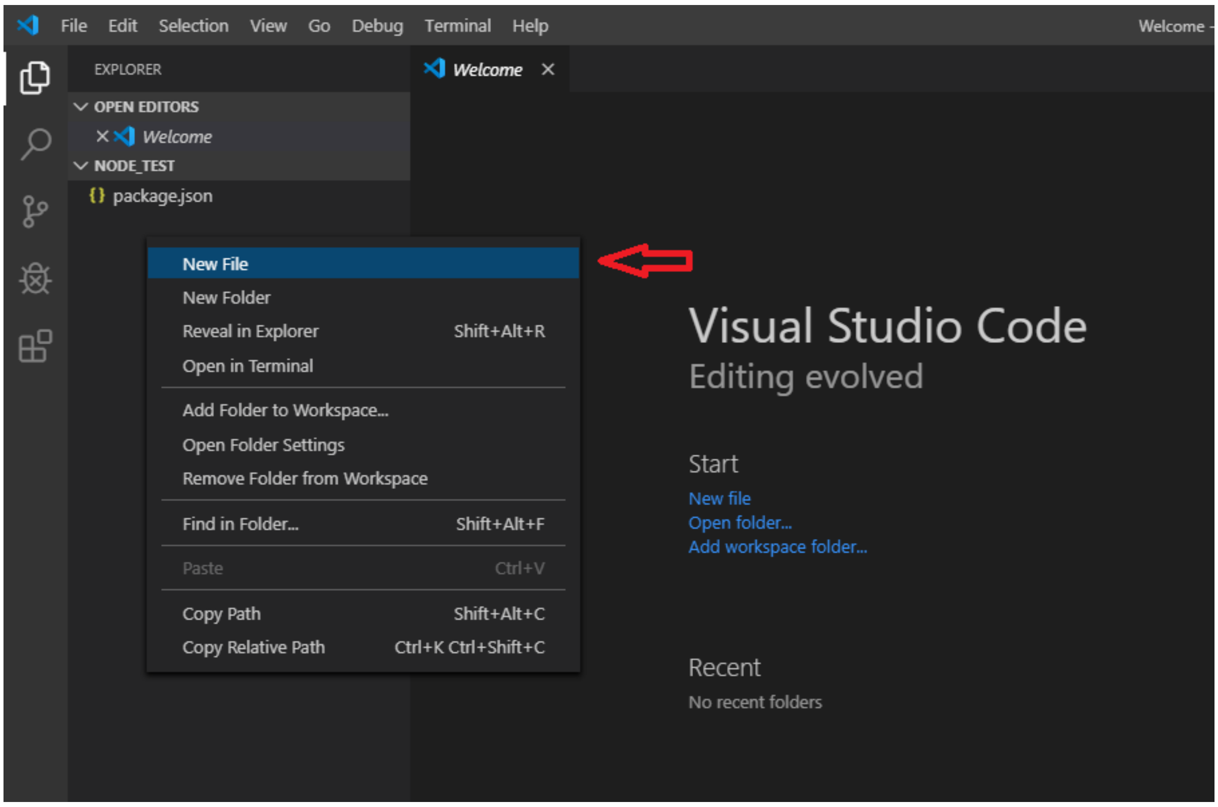
Open Visual Studio Code and follow these steps:

File > Open Folder



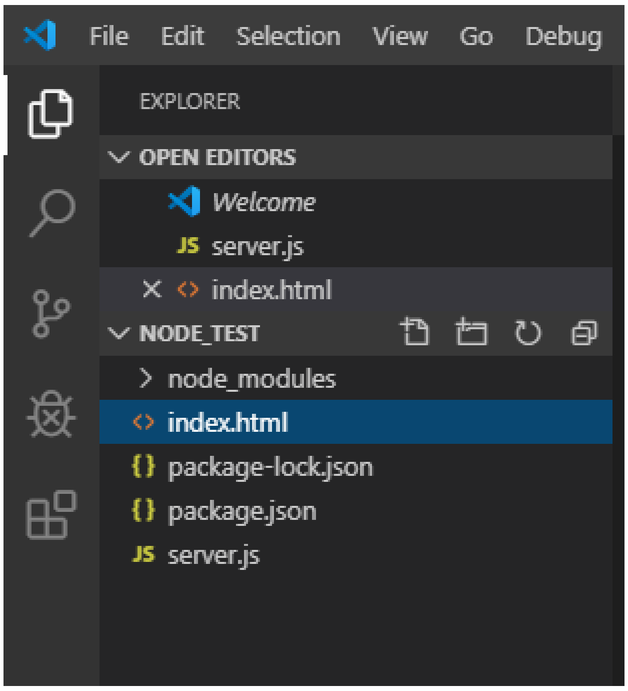
Then select the directory where your package.json stays.

Step 5: Creating Code File



Right click on the left panel and click to “New File”. And type “server.js” as file name.

Repeat this procedure for “index.html” file.

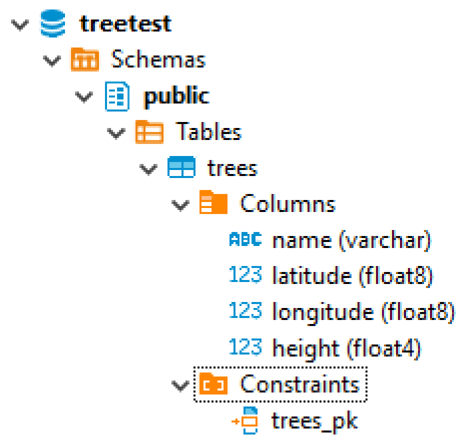


Your workspace should look like this.

(Except node_modules and package-lock.json. They should not appear now.)

Step 6 : Designing Database

**You can do the same process in pgAdmin or another software. I used DBEaver for design.



```
treetest
  Schemas
    public
      Tables
        trees
          Columns
            name (varchar)
            latitude (float8)
            longitude (float8)
            height (float4)
          Constraints
            trees_pk
```

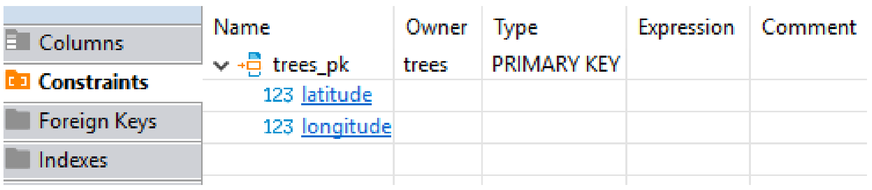
Create Database named “treetest”

Create table named “trees”.

Create Columns named with names:

- name
- latitude
- longitude
- height

Create Constraint named trees_pk and set primary key for latitude and longitude at same time.



Name	Owner	Type	Expression	Comment
trees_pk	trees	PRIMARY KEY		
123 latitude				
123 longitude				

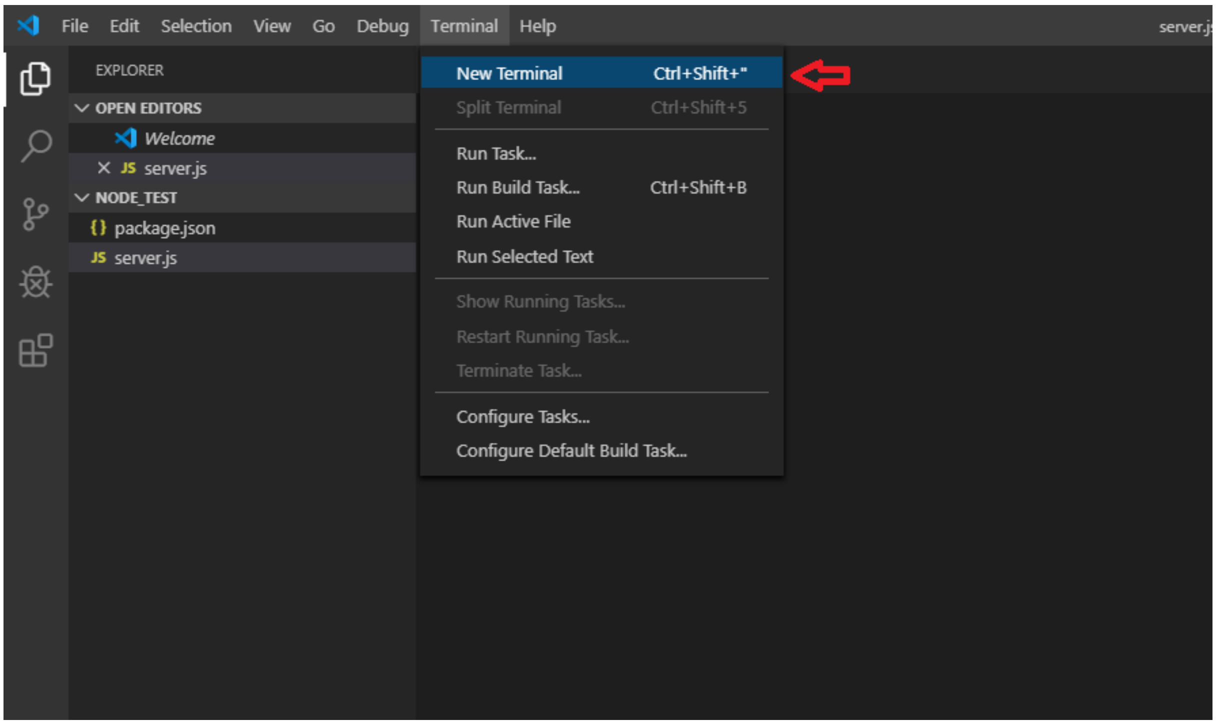
Step 7: Packages

Now switch back to Visual Studio Code and install some packages. Using:

npm install *package name*

- npm install pg
- npm install express
- npm install body-parser

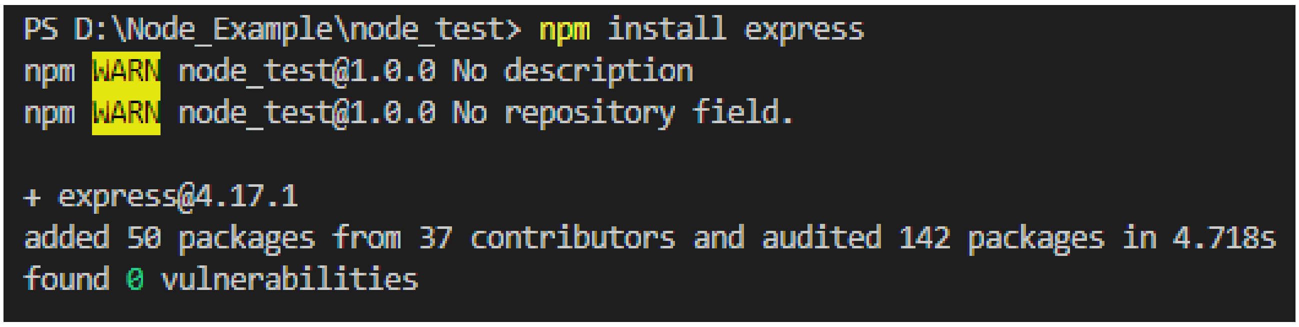
You need to open terminal :



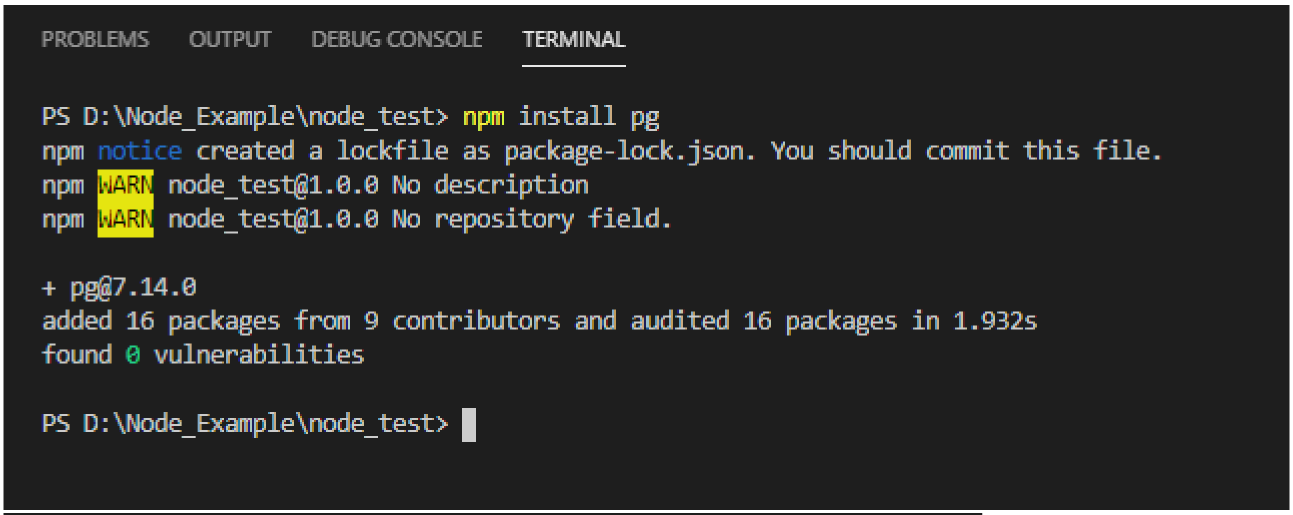
Terminal > New Terminal.

And now type your installations commands defined above in windows that appeared at bottom of Visual Studio Code.

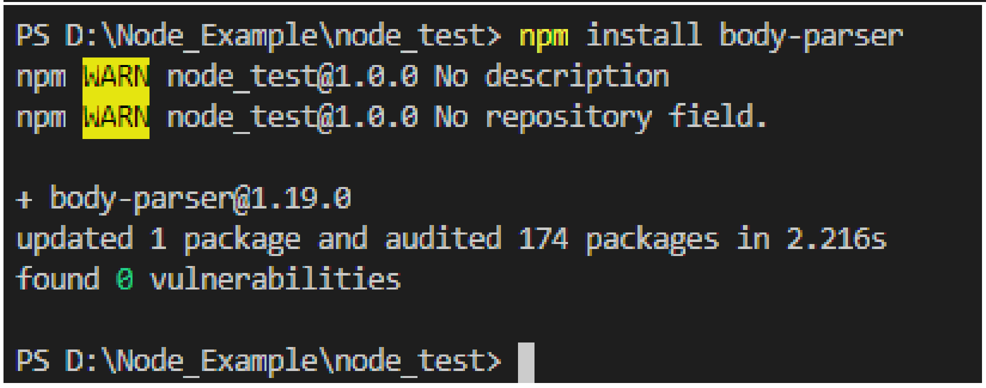
Commands:



```
PS D:\Node_Example\node_test> npm install express
npm WARN node_test@1.0.0 No description
npm WARN node_test@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 142 packages in 4.718s
found 0 vulnerabilities
```



```
PS D:\Node_Example\node_test> npm install pg
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN node_test@1.0.0 No description
npm WARN node_test@1.0.0 No repository field.

+ pg@7.14.0
added 16 packages from 9 contributors and audited 16 packages in 1.932s
found 0 vulnerabilities

PS D:\Node_Example\node_test>
```

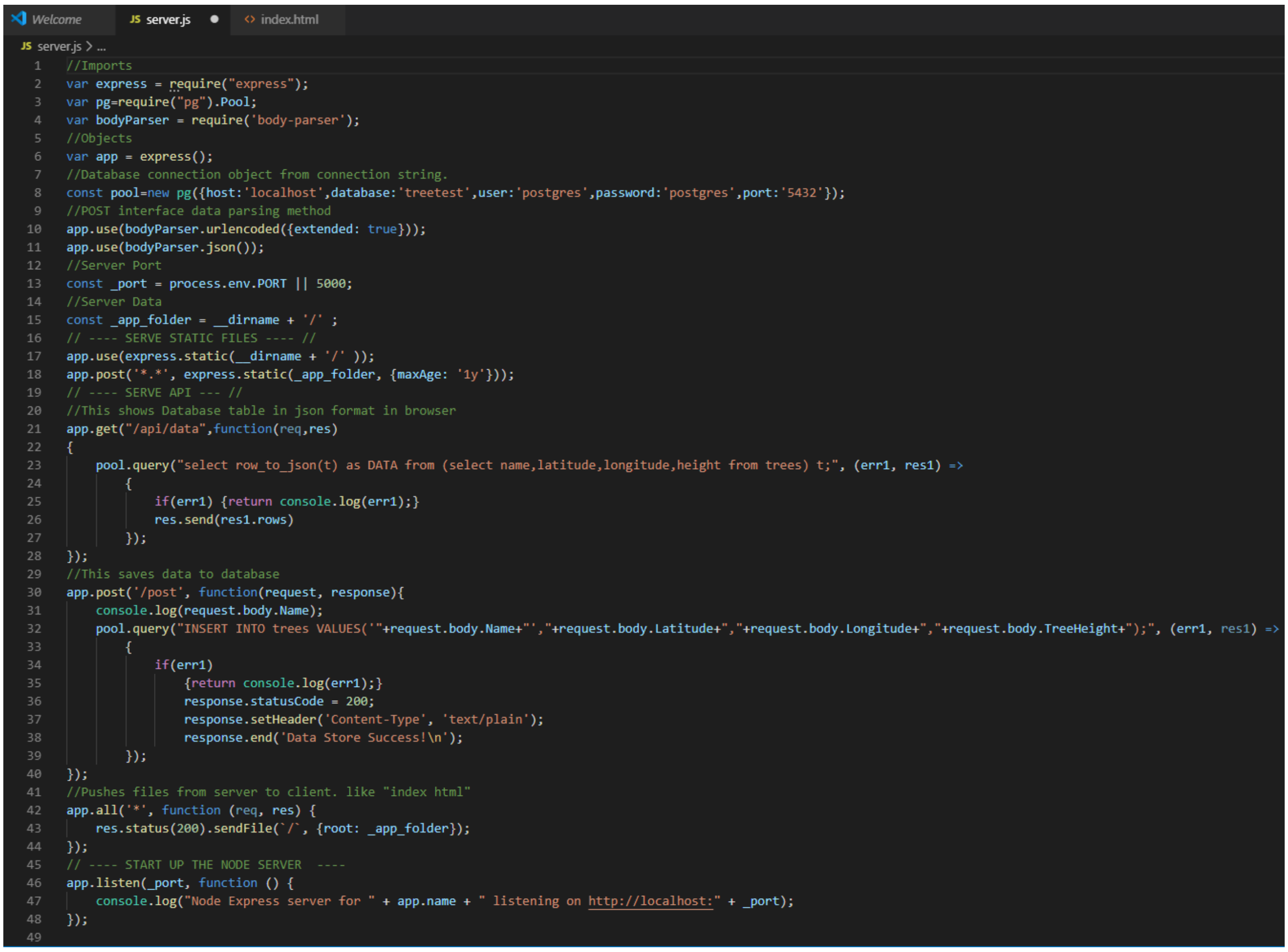


```
PS D:\Node_Example\node_test> npm install body-parser
npm WARN node_test@1.0.0 No description
npm WARN node_test@1.0.0 No repository field.

+ body-parser@1.19.0
updated 1 package and audited 174 packages in 2.216s
found 0 vulnerabilities

PS D:\Node_Example\node_test>
```

Step 8 : Coding

server.js:

https://github.com/afcsoft/nodeform_test/blob/master/server.js



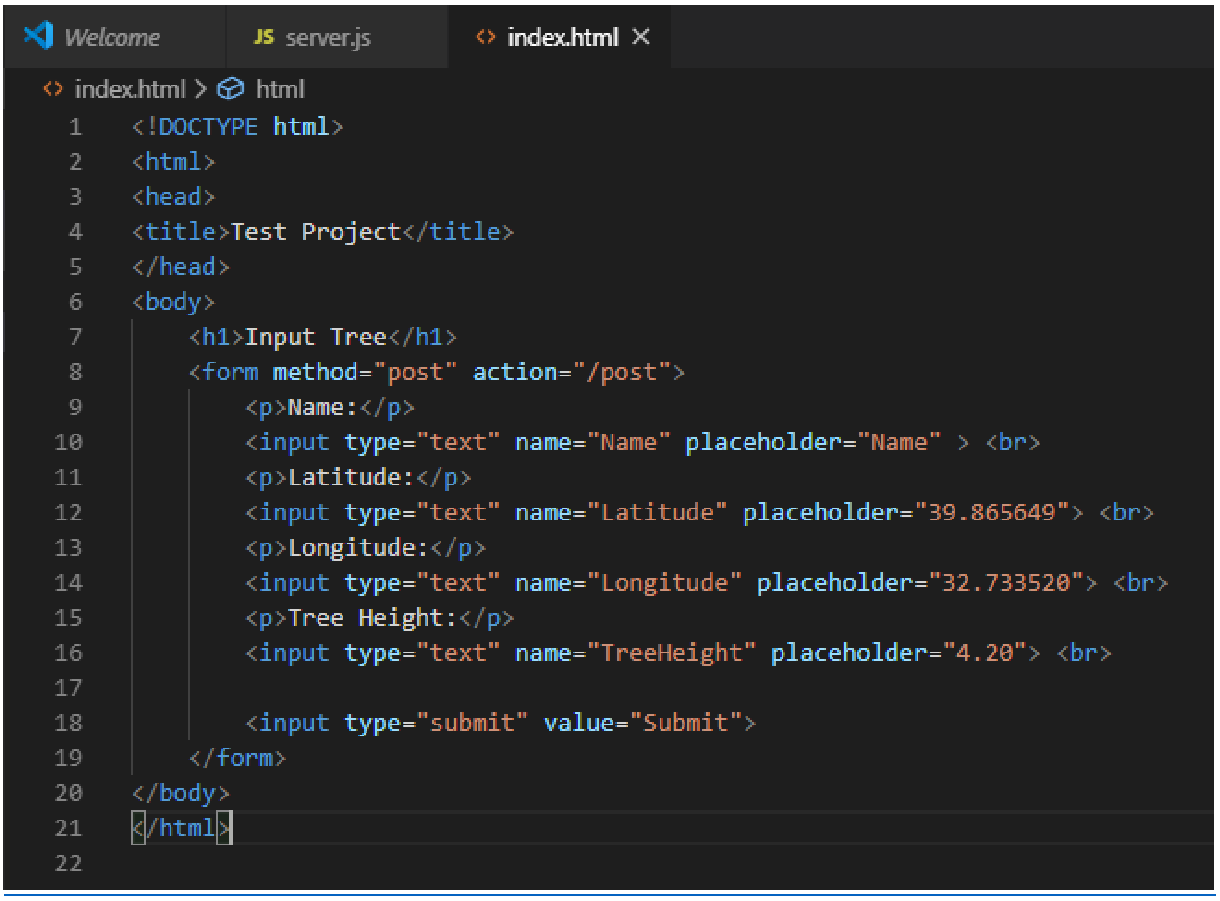
Welcome | server.js | index.html

server.js > ...

```javascript
//Imports
var express = require("express");
var pg=require("pg").Pool;
var bodyParser = require('body-parser');
//Objects
var app = express();
//Database connection object from connection string.
const pool=new pg({host:'localhost',database:'treetest',user:'postgres',password:'postgres',port:'5432'});
//POST interface data parsing method
app.use(bodyParser.urlencoded({extended: true}));
app.use(bodyParser.json());
//Server Port
const _port = process.env.PORT || 5000;
//Server Data
const _app_folder = __dirname + '/' ;
// ---- SERVE STATIC FILES ---- //
app.use(express.static(__dirname + '/' ));
app.post('*.*', express.static(_app_folder, {maxAge: '1y'}));
// ---- SERVE API --- //
//This shows Database table in json format in browser
app.get("/api/data",function(req,res)
{
    pool.query("select row_to_json(t) as DATA from (select name,latitude,longitude,height from trees) t;", (err1, res1) =>
        {
            if(err1) {return console.log(err1);}
            res.send(res1.rows)
        });
});
//This saves data to database
app.post('/post', function(request, response){
    console.log(request.body.Name);
    pool.query("INSERT INTO trees VALUES('"+request.body.Name+"',"+request.body.Latitude+","+request.body.Longitude+","+request.body.TreeHeight+");", (err1, res1) =>
        {
            if(err1)
                {return console.log(err1);}
                response.statusCode = 200;
                response.setHeader('Content-Type', 'text/plain');
                response.end('Data Store Success!\n');
        });
});
//Pushes files from server to client. like "index html"
app.all('*', function (req, res) {
    res.status(200).sendFile(`/`, {root: _app_folder});
});
// ---- START UP THE NODE SERVER  ----
app.listen(_port, function () {
    console.log("Node Express server for " + app.name + " listening on http://localhost:" + _port);
});

```

Index.html:

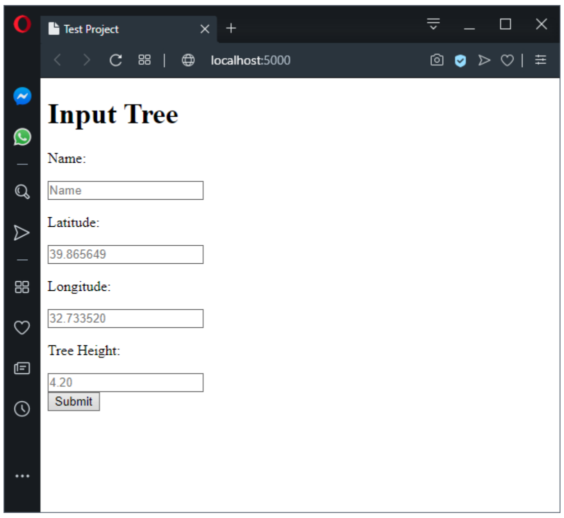
https://github.com/afcsoft/nodeform_test/blob/master/index.html



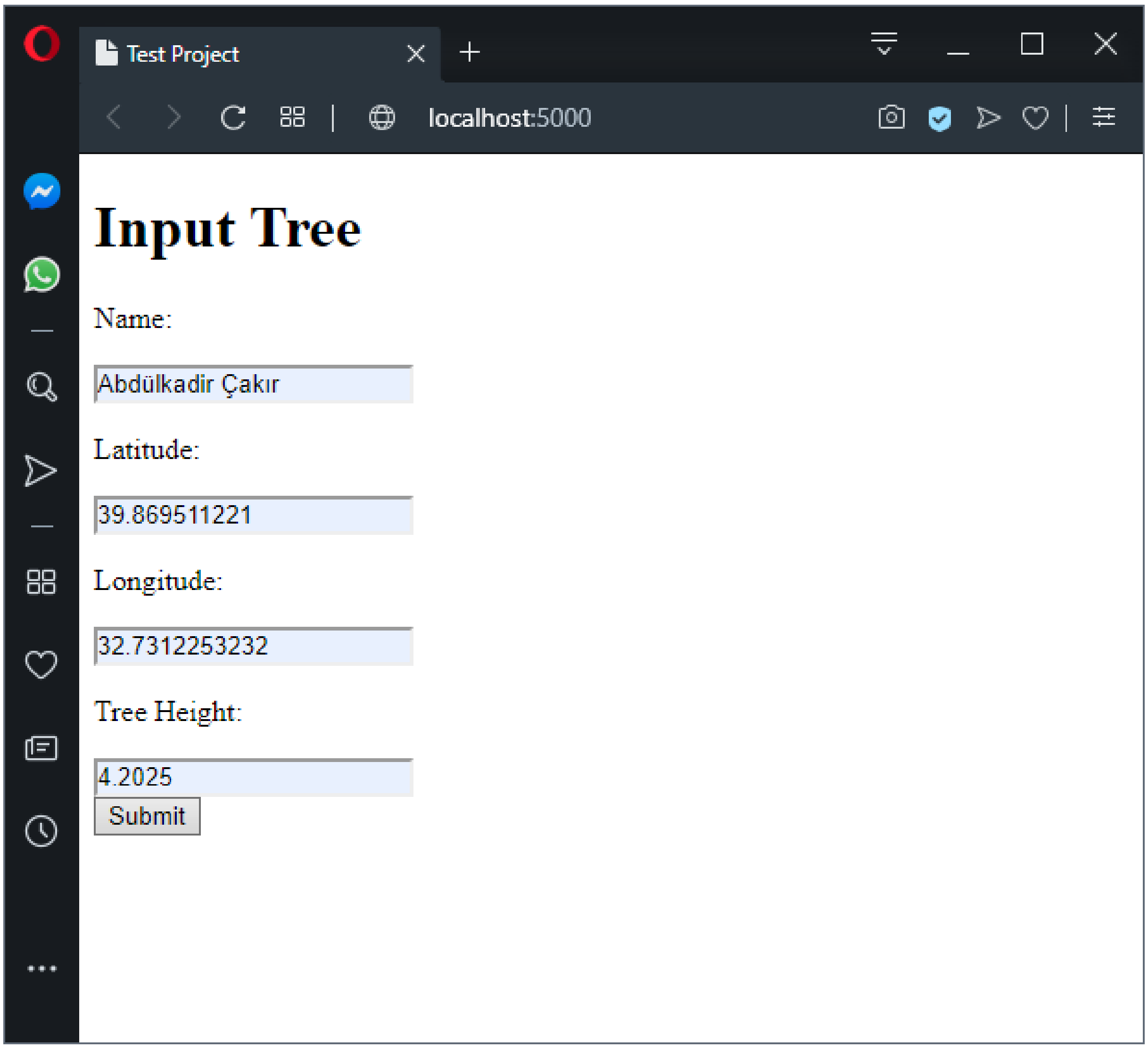
```html
<!DOCTYPE html>
<html>
<head>
<title>Test Project</title>
</head>
<body>
    <h1>Input Tree</h1>
    <form method="post" action="/post">
        <p>Name:</p>
        <input type="text" name="Name" placeholder="Name" > <br>
        <p>Latitude:</p>
        <input type="text" name="Latitude" placeholder="39.865649"> <br>
        <p>Longitude:</p>
        <input type="text" name="Longitude" placeholder="32.733520"> <br>
        <p>Tree Height:</p>
        <input type="text" name="TreeHeight" placeholder="4.20"> <br>

        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Now we can run our code by simply typing to terminal this:

```
node server.js
```



And we can go to http://localhost:5000 to view our project

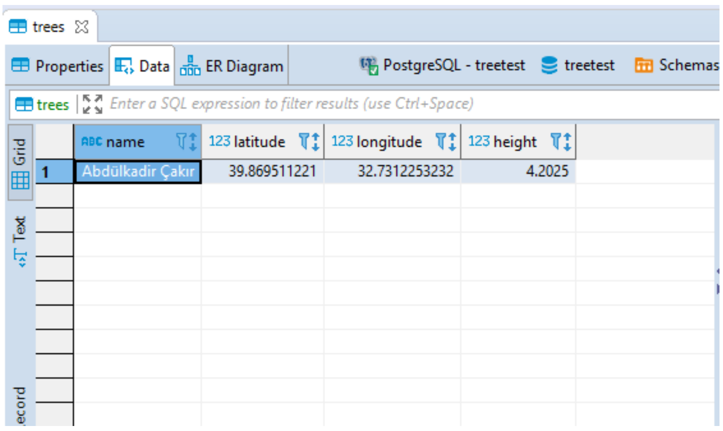

Test Project

localhost:5000

Input Tree

Name:	Abdülkadir Çakır
Latitude:	39.869511221
Longitude:	32.7312253232
Tree Height:	4.2025

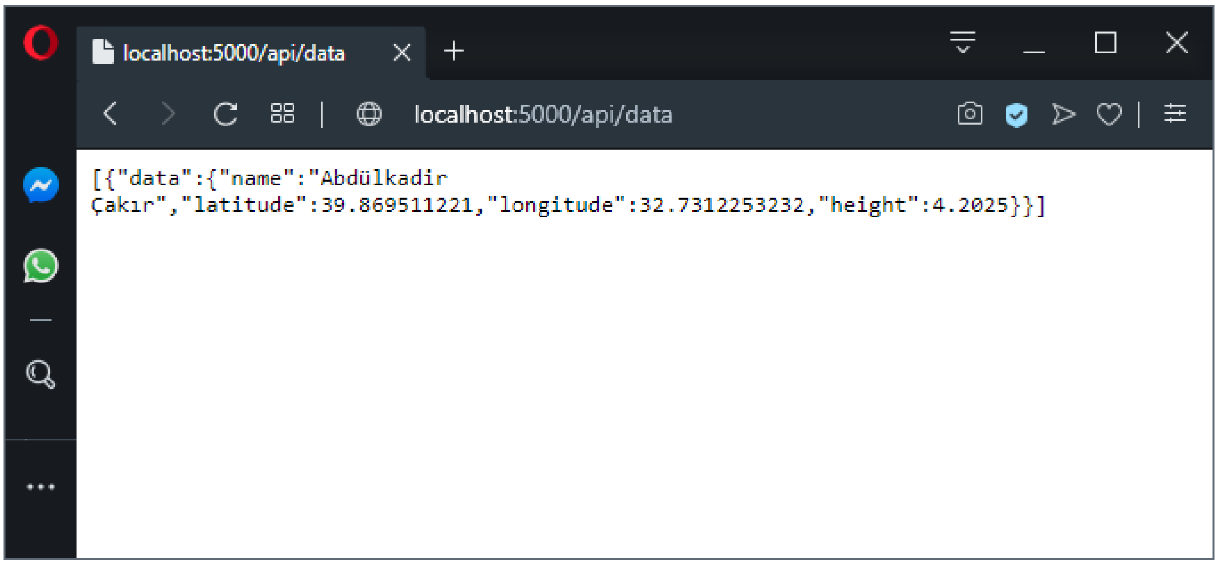
When you entered the data and hit submit button, backend code will take this data and will insert it to database.



	name	latitude	longitude	height
1	Abdülkadir Çakır	39.869511221	32.7312253232	4.2025

And you can view it on web site in JSON format using API that we designed in our code.

http://localhost:5000/api/data



```
[{"data":{"name":"Abdülkadir
Çakır","latitude":39.869511221,"longitude":32.7312253232,"height":4.2025}}]
```

All steps are done. That is all for this project.

END OF FILE