# Web Interface

This section describes how to visualize the data stored in the database on the web. First, the section provides a brief introduction on the required technologies, including NodeJS, Visual Studio Code and GitHub. Second, the code is presented that can be used to visualize the collected field data.

## 4.1. Technologies

**NodeJS**

NodeJS is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser. It can be downloaded from this [link](https://nodejs.org/en/).

**Visual Studio Code**

Visual Studio Code is a source code editor that can be used with a variety of programming languages. It is developed by Microsoft for Windows, Linux and macOS. It can be downloaded from this [link](https://code.visualstudio.com/). The version used for this tutorial is 1.40.1.

**GitHub**

GitHub is a software development version control system using Git. In order to effectively upload our project to Heroku, Git must be downloaded from this [link](https://git-scm.com/downloads).

## 4.2. Create App on Heroku

Collected field data are kept on online data stores. In the previous sections, we have relied on Heroku, which allows opening free account. Heroku also allows publishing web based projects, and in order to achieve this, a project must be created in Heroku.

Firstly, click the ‘new’ and ‘Create new app’ as shown in Figure 38.

|  |
| --- |
| Figure 38 Creating a new app in Heroku |

Then, enter the ‘App name’ and fill ‘Choose a region’. Press ‘create app’. You may choose ‘Europe’ as the region.

|  |
| --- |
|  |

Figure 39. Providing an appropriate app name

The app name should be acknowledged by Heroku. Once the provided app name is acknowledged, the statement **‘<app name> is available**’ would appear as shown in Figure 39.

Let’s begin the coding of the website. Create an empty folder on the computer. After that,open the Visual Studio Code. Go to File 🡪 Open folder.

|  |
| --- |
|  |

|  |
| --- |
|  |

Then go to Terminal 🡪 New Terminal.

It opens a terminal screen below the screen. Write ‘npm init’ to terminal screen end press enter. It asks some questions to create ‘package.json’. This file holds various metadata relevant to the project.

|  |
| --- |
|  |

Please press enter the all questions until the end. The metadata file package.json is created at the end of this process. Once all the default values are accepted, the ‘package.json’ would look like as shown in Figure 40:

|  |
| --- |
|  |

Figure 40 The metadata file: package.json

Some JavaScript packages need to be installed. These packages will be installed via npm (Node Package Manager). Packages to install are listed as follows:

* **Express Module**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications.

|  |
| --- |
|  |

Write ‘npm install express --save’ to terminal and press enter..

* **File System Module**

File system module allows you to work with the file system on your computer.

Write ‘npm install fs --save’ to terminal and press enter.

|  |
| --- |
|  |

* **Installing Bluebird Module**

Bluebird is a fully featured library with focus on innovative features and performance.

Write ‘npm install bluebird --save’ to terminal and press enter.

|  |
| --- |
|  |

* **Pg-Promise Module**

Pg-promise is a PostgreSQL interface for NodeJS.

Write ‘npm install pg-promise --save’ to terminal and press enter.

|  |
| --- |
|  |

The package.json will be updated once these packages are installed, and will look as shown in Figure 41.

|  |
| --- |
|  |

Figure 41 Updated package.json file

Project settings are ready. Firstly, the database connection can be established. Click the new file and add appConfig.js file as shown in Figure 42.

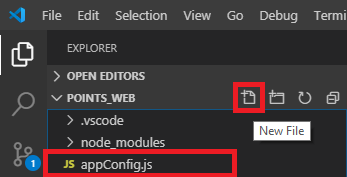


Figure 42 Adding the appConfig.js file

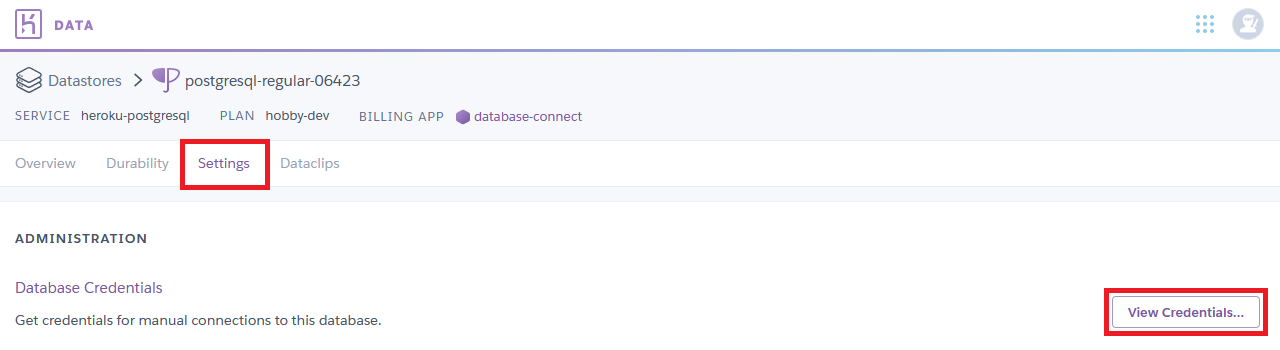
The appConfig.js would include the database connection settings (for now it is the localhost) and the entire content of the file is illustrated in Figure 43.



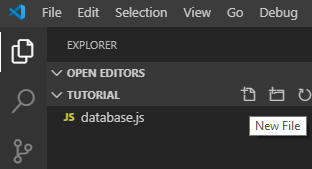
Figure 43. appConfig.js file

The only statement that requires to be changed regarding the credentials of the Heroku database is the connectionString.

Credentials of the Heroku database can be found through ‘Settings’ 🡪 ‘View Credentials’.



After that, data stored in the database must be retrieved. For this, we need to create a new file ‘database.js:



The connection is applied from appCongig.js and the query is applied in this script. The important thing is the query at this point. Check that your table name and queries correspond to those in your database. The content of the ‘database.js’ is shown in Figure 44.



Figure 44 database.js file

Finally, the web site can be created. Press ‘new file’ and create the Index.html file.

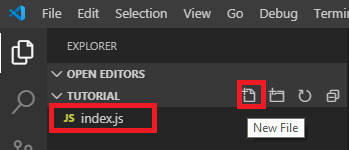
|  |
| --- |
|  |

The map is obtained from OpenLayers. The query response returns a GeoJSON file. Therefore, Jquery library will be used for the return query response. Also, the security policy is added to run the web site page as HTTPS protocol. Copy the code as an HTML file as shown in Figure 45.



Figure 45 index.html file

Finally, the website needs to be published through NodeJS. Click the new file and create index.js file.

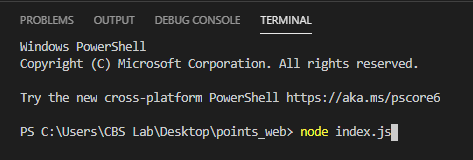


This file provides code to publish the website. It reads the HTML page and it sends a response.



Figure 46. index.js file

It can be checked whether the website is working or not. Go to terminal and write ‘node index.js’, then press enter.



It starts the project in ‘http://localhost:4000/’. Go to a web browser and click the <http://localhost:4000/>. Once the browser opens, the data stored on the database would be retrieved and displayed on the map as shown in Figure 47.

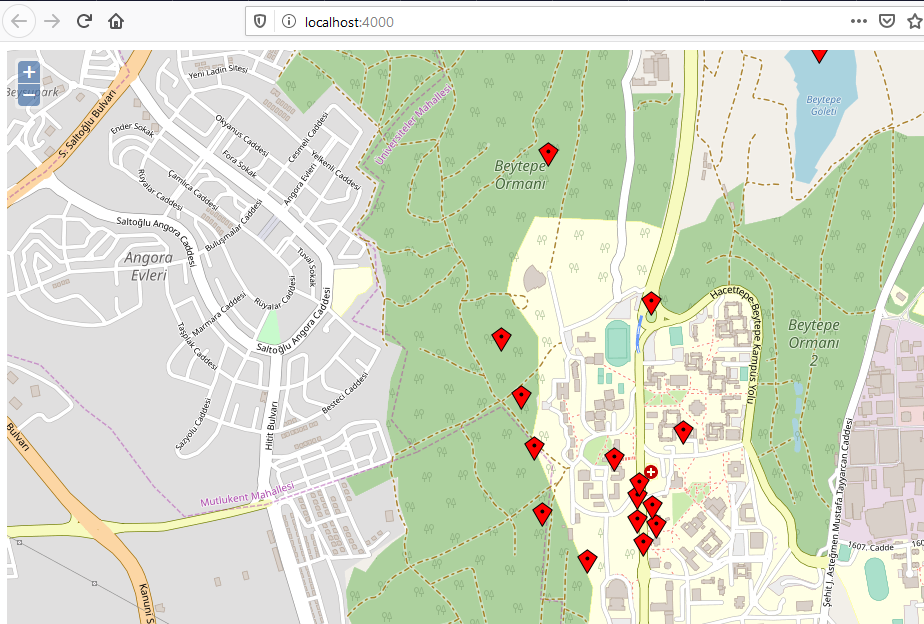
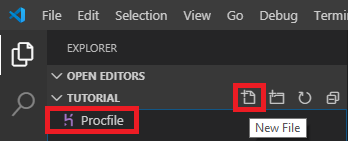


Figure 47. Displaying collected data on the browser

## 4.3. Deploying the Project on Heroku

Another file needs to be created before the website is published through Heroku. Go to new file and create ‘Procfile’ file. Heroku apps include a Procfile that specifies the commands that are executed by the app on startup.



Only the starting file of the project is specified.

There is one last thing needed before pub deploying the project on Heroku. The created ‘package.json’ file needs to be updated. It must be told by which file the website is started. Therefore, the start parameter must be added as shown in Figure 48.

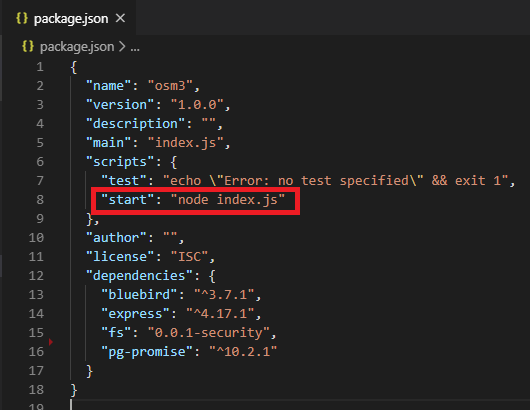
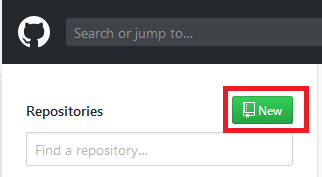
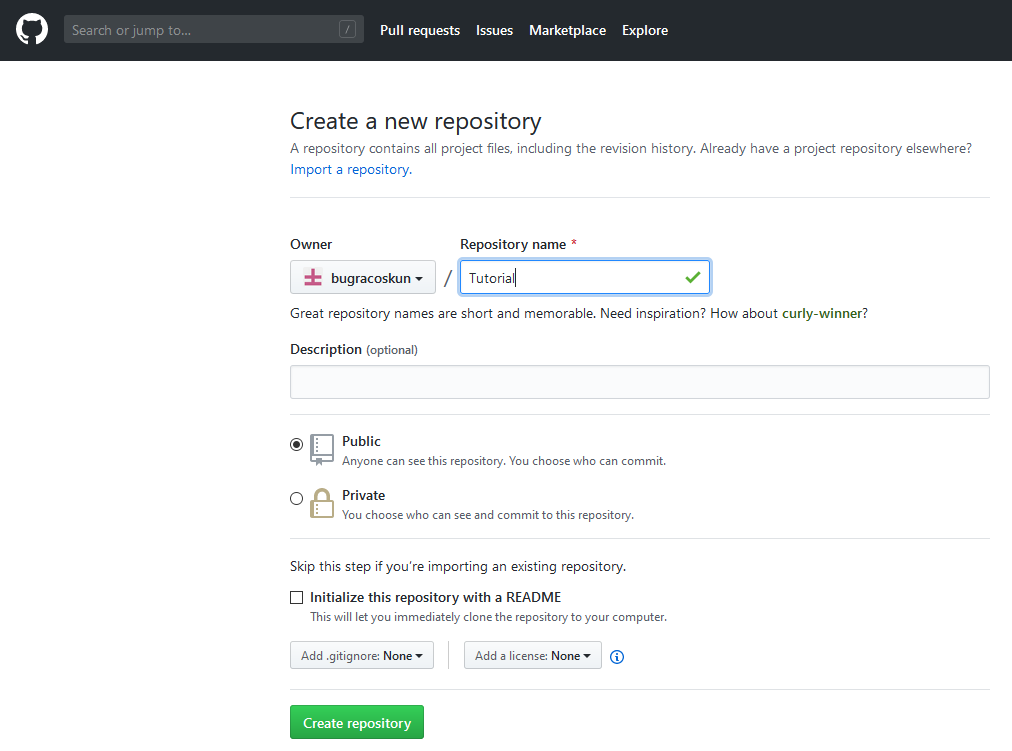


Figure 48. Adding the start parameter to the package.json

Finally, the project is ready to be published on Heroku. Go to Github page and press New.

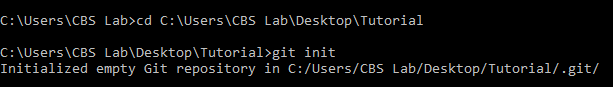


Enter the ‘Repository name’ and press ‘Create repository’.



Go to your file directory through cmd (Command Prompt). Write the command:

* git init



Git is initialized in the file. Then write this command to add all the contents of the folder:

* git add .



Files need to be committed with a command. Write the command:

* git commit –m “your message”



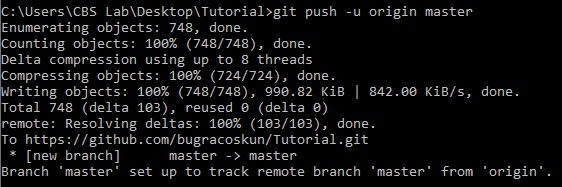
After that, establish a remote connection. Note that you need to write the project link instead of ‘ProjectLink’. It can be done with:

* git remote add origin ProjectLink

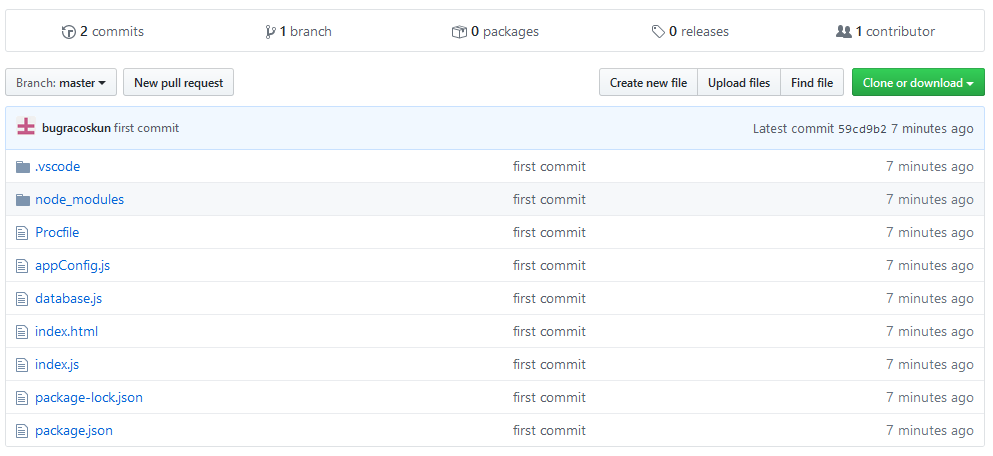


The last step is pushing the files to GitHub. Write this code:

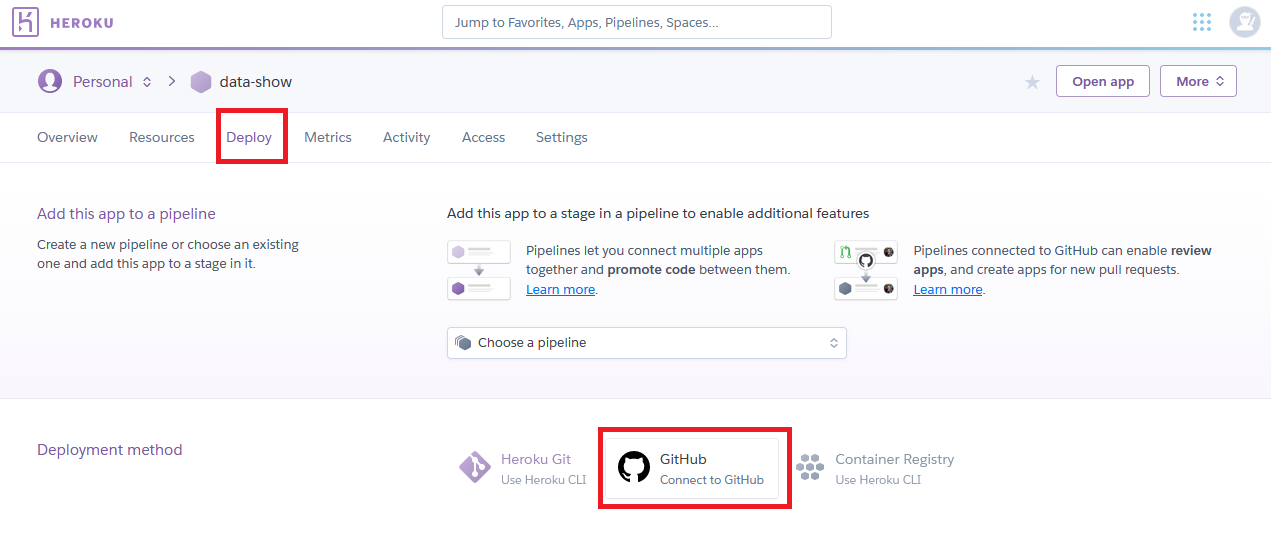
* git push -u origin master



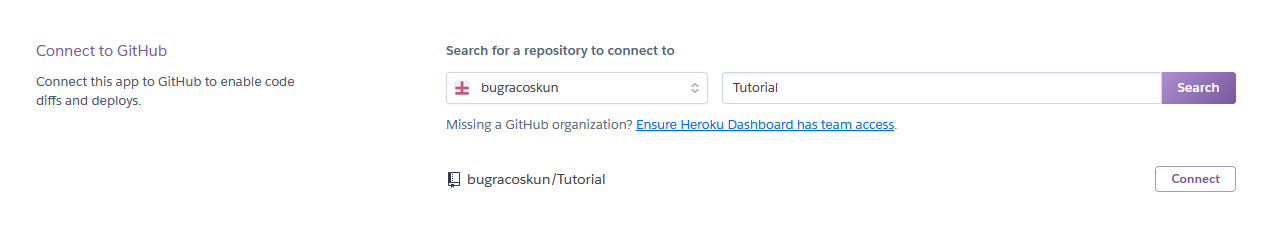
Our files can be seen in the GitHub page.



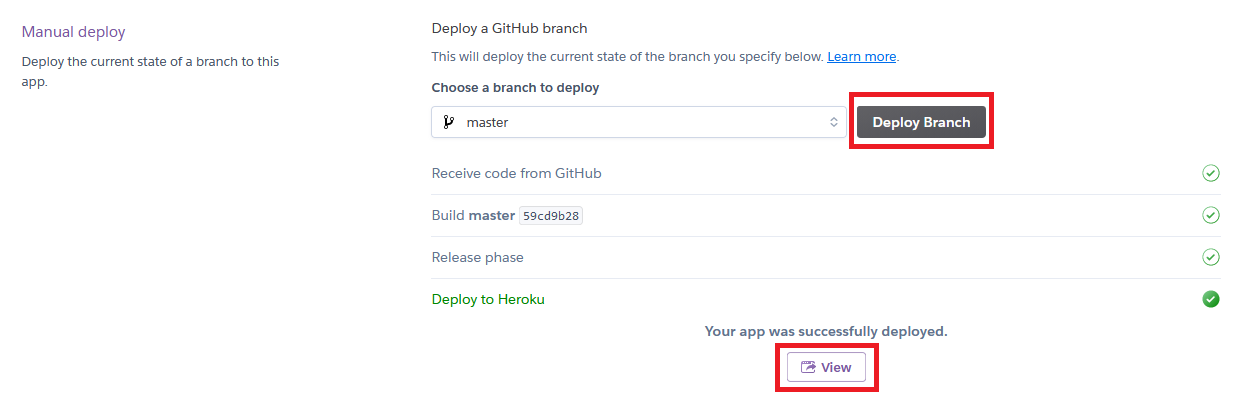
The last step to publish the website is deploying it to Heroku. Go to the Heroku project page and ‘Deploy’ 🡪 Github.



Find the repository on Heroku and press connect.



There are two options. Go to manual deploy and press ‘Deploy Branch’. After the development stage, press ‘View’.



The website is published in the given link. Whenever there is an update in QField, it is possible to observe the change on the web page as shown in Figure 49.

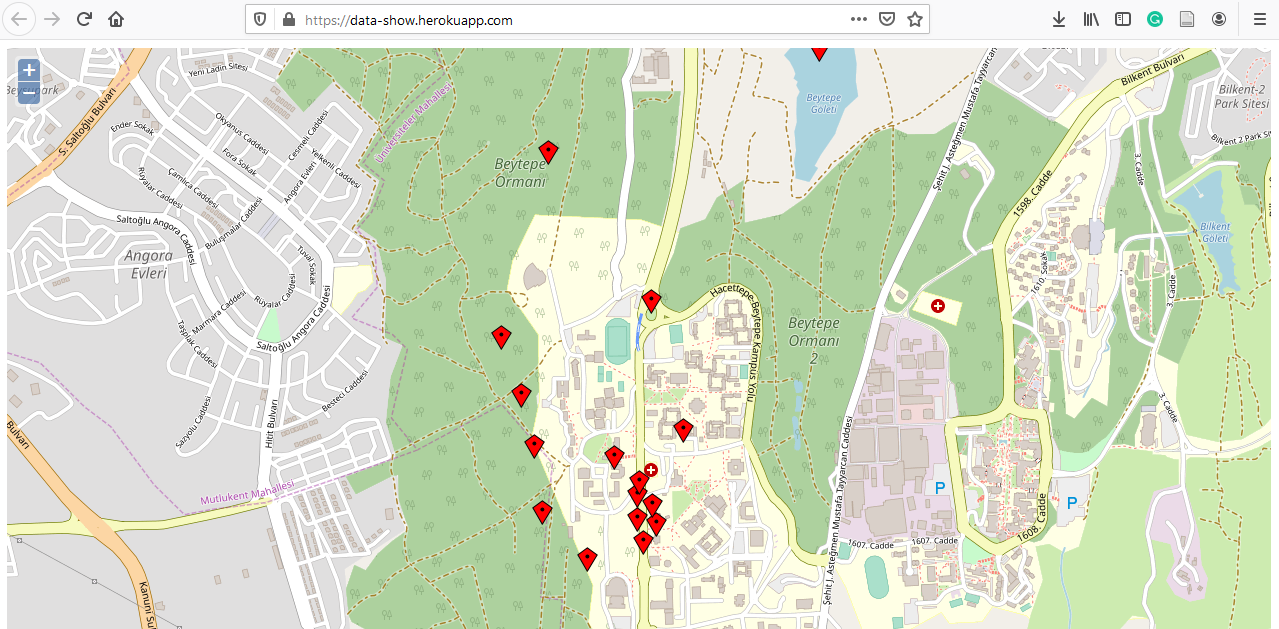


Figure 49 Displaying the collected data