# Tutorial on ‘Open geospatial data and software for UN SDG 16, Peace justice and open institutions’

# 1. Introduction

United Nations (UN) have identified 17 interconnected goals to achieve a better and more sustainable future for all until the end of 2030. These Sustainable Development Goals (SDGs) range from ending poverty to ensuring gender equality. In order to address most of these goals, it is imperative to have a better understanding of geographic information science and technology, as any ‘event’ has a particular geographic location. The aim of this educational tutorial is to investigate the use of Free and Open Source Software for GIS (FOSS4GIS) to address SDG 16: ‘Peace, Justice and Open Institutions’.

## 1.1. Scope

The following educational material has been drafted within the framework of the 2019 OSGeo UN Committee Educational Challenge entitled ‘Open geospatial data and software for UN SDG 16, Peace justice and open institutions’. The seventh target of SDG 16 is to ‘ensure responsive, inclusive, participatory and representative decision-making at all levels’. Consequently, UN officers and patrols might need to collect geospatial data in the field to provide quantitative measures to address this target. In that respect, we have investigated the use of QField, which is an open source mobile data collection platform that can be directly connected to QGIS. The investigated tasks include real-time editing of geographic features, recording photos and developing a web-interface to display the collected data. There are several cloud based application development platforms and this tutorial relied on ‘Heroku’ due to its support for PostgreSQL/PostGIS, as well as different web programming languages. Any user can open a free account to test their ideas on [Heroku](https://www.heroku.com).

## 1.2. Audience

We prepared this educational material for researchers, educators and professionals in local, regional, national or international agencies with minimal geospatial information knowledge. We assume our audience has basic knowledge of database management systems, and basic skills and understanding of how to work with geospatial and tabular data. However, the tutorial is intended to be self-contained meaning that anyone following the steps should be able to understand and implement what has been achieved in this tutorial.

## 1.3. License

This educational material was prepared by Berk Anbaroğlu under the mentorship of Maria Brovelli, Serena Coetzee and Timur Obukhov. In addition, İhsan Buğra Coşkun and Mohammad Almasri contributed to the development of this tutorial. We would like to thank Agit Oktay for his support regarding the development of the web interface. The material is distributed under Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license. This entitles the users of this tutorial to share and adapt this content as long as the attribution is provided and that the same license is used upon adaptations.

This tutorial has been kindly supported financially by the ‘2019 OSGeo UN Committee Educational Challenge’.

## 1.4. Software Used

* PostgreSQL (Postgres) 11.5 is used to create the database that stores the collected data. PostGIS 2.5.3 extension is used to handle spatial data.
* QGIS 3.6 is used to create the project based on the data stored in the Postgres database.
* QField 1.2.0 is used for mobile data collection and editing.
* Snycthing is used to synchronize collected photos.
* NodeJS 12.13.1 is used to visualize the collected data on the web.
* Visual Studio Code 1.41.0 is the Integrated Development Environment for developing the JavaScript code.
* Git is a version control system that is required to deploy the web based application.

## 1.5. Scenario

United Nations (UN) officers require mobile field data collection for various purposes, mainly of which is to provide reliable measures to the [indicators of targets](https://unstats.un.org/sdgs/metadata/). It is necessary to provide an easy-to-follow tutorial that will ease the process of mobile spatial data collection of UN officers. This tutorial addresses some of the practical requirements of mobile spatial data collection including offline data collection, attaching photos to recorded events, editing of spatial features on the field and displaying the collected data on the internet through a simple scenario.

The scenario in this tutorial is recording the location and thematic information of trees in a region. Specifically, a lecturer specifies polygons for which each student is responsible. Thereon, students go to the field to collect the information regarding trees in that region. Some of the regions might have limited broadband coverage, which requires students collect offline data, and some students may capture photos of trees. Finally, it would be interesting to visualize the collected data on the internet so that decision makers (lecturers) could have a better understanding of over/under mapped regions. [Other tutorials](https://livelihoods-and-landscapes.github.io/qgis_qfield_tutorials/index.html) focused on generating complex data collection forms.

The overview of this tutorial is illustrated in Figure 1:

|  |
| --- |
|  |

Figure 1. Overview of the tutorial

## 1.6. Acquired Knowledge

After going through the entire educational material, one will be able to understand the process to develop a mobile geospatial data collection platform using QField, QGIS, Postgres/PostGIS and Syncthing.

# 2. Setting Up the Environment

This section describes the steps required to set up the development environment. In order to realize the steps described in subsequent sections, it is recommended that the users:

* Open a Heroku account
* Install PostgreSQL and PostGIS on their local computer
* Connect the Heroku database from QGIS
* Set up the project in QGIS
* Transfer the project from desktop PC (QGIS) to mobile data collection platform (QField).

## 2.1. Opening a Heroku Account

In order to deploy a mobile geographic data collection project, a database stored in the web should be realized. There are several solutions that offer a free solution for this. One of the platforms supporting Postgres and PostGIS is Heroku, which is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. This tutorial relied on Heroku to set up the database on the cloud.

The sign up page of [Heroku](https://signup.heroku.com/login?redirect-url=https%3A%2F%2Fid.heroku.com%2Foauth%2Fauthorize%3Fclient_id%3Dd2ef2b24-e72c-4adf-8506-28db2218547d%26response_type%3Dcode%26scope%3Dglobal%252Cplatform%26state%3DSFMyNTY.g3QAAAACZAAEZGF0YW0AAAAxaHR0cHM6Ly9kYXNoYm9hcmQuaGVyb2t1LmNvbS9hdXRoL2hlcm9rdS9jYWxsYmFja2QABnNpZ25lZG4GANL7onluAQ.eDiBIjGpk8wBx82K2Ej2tBwAitNPBGNQoMMDLoySy78), as illustrated in Figure 2, requires some personal information such as name, surname and email address. Role is a compulsory field that can be, amongst others, ‘agency/partner developer’, ‘hobbyist’, ‘student’ or ‘other’. Select the role that best describes you.

|  |
| --- |
|  |

Figure 2 Sign up page of Heroku

Once the form is submitted, an email is sent to verify the account. These credentials is used to create a new app. The second step is to create the database as illustrated in Figure 3(a-h).

|  |
| --- |
|  |
| 1. Create new app |
|  |
| 1. App name should be acknowledged by Heroku |
|  |
| (c) Click on data |
|  |
| (d) Enter the name of your app |
|  |
| (e) Click on “Provision add-on” to agree to the terms of seerver |
|  |
| (f) Overall view of the project |
|  |
| (g) Click on “View Credentials ” |
|  |
| (h) Database credentials |

Figure 3 Deploying a Postgres database on Heroku

The credentials of the database, as illustrated in Figure 3h, include the necessary details of the database, such as the URL of the host and name of the database, user name, port and password. Since these details are private, they are censored within this tutorial with a grey rectangle.

## 2.2. Installing PostgreSQL / PostGIS

This step is optional. Specifically, a user interface that is used to connect a Postgres database such as [pgAdmin](https://www.pgadmin.org/download/pgadmin-4-windows/) or [DBeaver](https://dbeaver.io/) would be sufficient to achieve this step. The readers who may want to install Postgres on their local computers could follow this subsection.

This tutorial relied on [Postgres](https://www.postgresql.org/download/) 11.5 and PostGIS 2.5.3. Once Postgres is downloaded to the user’s local computer, it comes with an application called ‘Stack Builder’, which allows installing extensions to Postgres, including PostGIS. The extension has to be installed through Stack Builder as shown in Figure 4.

|  |
| --- |
| stack builder pgadmin 4 postgis ile ilgili görsel sonucu" |

Figure 4. Installing PostGIS through Stack Builder

[PgAdmin 4](https://www.pgadmin.org/download/pgadmin-4-windows/) provides an easy-to-use Graphical User Interface (GUI) to access and manipulate a Postgres database.

Once Postgres is installed, we can connect to our Heroku database with the following steps, which are illustrated in Figure 5(a-b).

* On pgAdmin right click on Servers 🡪 create server
* General 🡪 Give a name for your server (e.g. heroku).
* Go to the Connection tab as shown in Figure 5b and refer to the Heroku database credentials as shown in Figure 3h. Copy and paste the information to the following inputs:
  + Host name/address: Host
  + Maintenance database: Database
  + Username: User
  + Password: Password (check the ‘Save password’ box for ease of login in future)
* Finally go the ‘Advanced’ tab and state the name of the database as it appears on Heroku (i.e. the same value used for the ‘Maintenance database’), so that only the relevant database would be visible.

|  |
| --- |
|  |
| 1. General tab – Enter the name for your database server |
|  |
| (b) Connection tab – Enter the Heroku database credentials |

Figure 5 Connecting to the Heroku server from local computer

It is now possible to create the tables. This tutorial relies on pgAdmin as the Graphical User Interface (GUI) to execute the SQL queries. A query window will open once the database is right-clicked and the ‘Query Tool’ selection is made as shown in Figure 4.

|  |
| --- |
|  |

Figure 6. Opening up the query tool in pgAdmin 4

The scenario for this tutorial requires three tables: trees and regions are tables with a spatial aspect, whereas types\_of\_trees do not have a geometry and are used to ease the process of data collection as well as to provide a better way to support multiple languages.

First, the only table that does not include a spatial component is created: tree\_type. By creating a tree\_type table, it is easier for students to collect the type of tree as they can select the type of the tree from a drop-down list. In the example provided in Figure 7, the name of the tree attribute is created for English and Turkish. Second, two different tree types are inserted into this table. If there would be more trees in the investigated regions, then all of them should be specified.

|  |
| --- |
|  |

Figure 7. Creating the tree types and inserting two different tree types

The other two tables possess a spatial component. Therefore, the first step is to activate the extension ‘postgis’, which is achieved using the ‘create extension postgis’ statement. Next, spatial features including a spatial component can be created. Specifically, ‘trees’ table has a point geometry indicating the latitude and longitude of the observed tree as well as the tree type, height and observation time. The second table, ‘regions’ include a polygon geometry indicating the areas in which each student is expected to carry out data collection. The students’ names can be included in the ‘responsible’ attribute.

|  |
| --- |
|  |

Figure 8 Creating the tables possessing a spatial attribute

## 2.3. Connecting the Database with QGIS

The Postgres database created can be included directly into a QGIS project, thanks to the seamless linkage between QGIS and Postgres. This process is illustrated in Figure 9.

|  |
| --- |
|  |

Figure 9 Linking the Postgres database with QGIS

Once PostGIS is clicked in the ‘Browser’ panel of QGIS, a new window entitled ‘Create a New PostGIS Connection’ opens. By providing the details of the Heroku database that we have created, it is possible to access the tables stored in the database in QGIS. The name of the project could be ‘QField’ and service can be left empty. However, host and database information should be exactly the same as displayed in Figure 3c (i.e. credentials of the Heroku database). It is also important to note that the ‘SSL mode’ should be ‘allow’. Last, user name and password should be stored in order to ease the process of real time data collection. If all the credentials of the Heroku database are entered correctly, clicking the ‘Test Connection’ button pops up a window indicating the success of the connection. Finally, once the OK button is clicked, the connection appears on the Browser panel as shown in Figure 10 and the user can add all the relevant tables to the layer panel by double-clicking on the table.

|  |
| --- |
|  |

Figure 10 Adding the tables in Heroku to QGIS

## 2.4. Setting Up the ‘Value Relations’ in QGIS

In order to ease the process of data collection, it is useful to explicitly state the value relationships between the feature layers and to improve the user experience by hiding some of the details. For instance, a student could only select one of the pre-defined tree types. This can be achieved by adjusting the ‘Attribute Form’ specified under the ‘Properties’ of the ‘trees’ layer as shown in Figure 11.

|  |
| --- |
|  |

Figure 11 Relating the tree type in ‘trees’ layer with the pre-defined ‘tree\_types’

The tree type attribute ‘t\_type’ has a value relation to the ‘tree\_types’ table. In order to realize this relation. The following settings are specified as shown in Figure 11.

* Widget type 🡪 Value Relation
* Layer 🡪 trees\_types
* Key column 🡪 chose ‘name’
* Value column 🡪 chose ‘name’

In addition, some of the attributes might better be hidden from the users such as the ‘id’ attribute of the trees or regions as it is a serial number that auto increments as a tree/region is recorded. Therefore, this attribute shall not be editable. In addition, it can be hidden from the user to reduce clutter in the GUI of the QField. This process is illustrated in Figure 12.

|  |  |
| --- | --- |
|  | Actions:   * Uncheck ‘Editable’ * Widget Type: Hidden |

Figure 12 Removing clutter from the GUI by hiding some attributes

Once an attribute is ‘hidden’, it means that during data collection that field is not displayed to the user. However, a user could still update it under the ‘attribute table’. In order to prevent this as well, we have unchecked ‘Editable.

The last attribute of the ‘trees’ layer is the observation time. It might be important to record the observation time in order to facilitate analysis for further spatio-temporal analysis. At the moment, it is possible to record the height of the same tree at two or more distinct time intervals. By updating the default value of the observation\_time attribute to be ‘now()’ as shown in Figure 13, whenever a tree is recorded, its observation time is also recorded.

|  |  |
| --- | --- |
|  | Actions:   * Widget Type: Date/Time * Default value: now() |

Figure 13 Assigning default observation time to the corresponding attribute

OpenStreetMap can be included in the project to ease the data collection process as shown in Figure 14, where a region is added.

|  |
| --- |
|  |

Figure 14 Including OpenStreetMap in the project to ease data collection

Note that all of the edits made for the ‘trees’ and ‘regions’ layers are updated in the Heroku database stored in the cloud.

## 2.5. Transferring the Project to QField

Once all the layers are added, the project is saved as a [QGS file](https://docs.qgis.org/testing/en/docs/user_manual/introduction/project_files.html). This means that the file is a legible XML document. All of the details of the project are included in this document including the credentials of the Heroku database. Anyone possessing this QGS file can start collecting field data, which would have immediate effect on the Heroku database. In this way, multiple users can effectively collect field data into the same database. This process is described in Figure 15.

|  |  |
| --- | --- |
| SmartSelect_20191120-192242_QField | SmartSelect_20191120-192301_QField |
| (a) Editing a project – I | (b) Editing a project – II |
| C:\Users\ASD\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20191120-192348_QField.jpg | SmartSelect_20191120-192340_QField |
| (c) Selecting the location of a tree | (d) Filling the thematic attributes |

Figure 15 Data collection on QField

Once the QGS project file is transferred to the mobile device, it can be opened using QField as shown in Figure 15a. The layers appear just like they do in QGIS,as shown in Figure 15b. In order to start editing, the relevant layer is selected and then the pen icon is toggled in the upper-right corner of the screen. Once the horizontal lines button on the upper left corner is pressed, the user can start collecting data. The location could either be set by panning around the map and clicking on the ‘+’ button located at the bottom-right of the screen or by clicking the locate button at the left-hand side of the screen, which uses the GNSS receiver of the phone to locate the position of the user. New features are added just like they are added in QGIS with the functionality of hiding some attributes (e.g. id in trees) or by offering a drop-down list to the user (e.g. user can select one of the pre-defined tree types).

## 2.6. Offline Data Collection

QField allows its users to collect data when there is no internet connection. This requires the use of the ‘[QField Snyc](https://plugins.qgis.org/plugins/qfieldsync/)’ plugin, which can be installed from the QGIS plugin repository. The steps involved in offline data collection are as follows:

* In QGIS, select ‘Preferences’ as shown in Figure 16a. Clicking on this button opens the window presented in Figure 16b.
  + Set the ‘Default Import Directory’ to be the path of the project.
  + Set the ‘Default Export Directory’ to be the path of the files that are to be exported. This directory must be copied to the mobile device.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 16 QFieldSnyc plugin

* Select ‘Project Configuration’, which opens a window as shown in Figure 17, showing a list of the available layers and actions that can to be taken for each of them.
  + For OpenStreetMap, select ‘no action’.
  + The other layers should be selected to be in the ‘offline editing’ mode.
* Note that we need to change the properties of the layer (type of tree, id hidden and visible) as we did in Figure 11 and Figure 12.

|  |
| --- |
|  |

Figure 17. QField project synchronisation details

* Select ‘Package for QField’, which opens a window as shown in Figure 18. Select the export directory, which could be under the path identified in Figure 16b. Once the ‘Create’ button is clicked, the plugin creates the required Geopackage file and associates it with the QGS file. In this way, data are stored in Geopackage and each update to the project effectively update this Geopackage file.

|  |
| --- |
|  |

Figure 18. Creating the package for QField

* Once the project is transferred to the mobile device and the QField app is used to collect field data, the synchronisation must be made to ensure that the Heroku database is up-to-date. Select ‘Synchronize from QField’, which opens up a window as illustrated in Figure 19. Once the Synchronize button is clicked, all of the content of the Geopackage is read and the Heroku database is updated by appending the new records. In this way, it is possible for multiple users to collect offline field data.

|  |
| --- |
|  |

Figure 19. Synchronize project

# 3. Collecting Photos in the Field

Photos can be used in mobile data collection projects to better describe a spatial object. Storing photos in the database is not trivial, and most of the time the path to the photo is stored. For this reason, an additional platform is needed to share photos where multiple users collect data.

## 3.1. Setting Up the Snycthing Environment

[Syncthing](https://syncthing.net/) is a continuous file synchronization program. It synchronizes files between two or more devices. The location of the shared folder is specified on Syncthing. It requires internet to synchronize the files. The aim of using Syncthing in this project is synchronizing photo folders between mobile device and computer.

Firstly, Syncthing must be installed on all devices. It can be downloaded for Windows from [here](https://github.com/canton7/SyncTrayzor/releases/tag/v1.1.24) as illustrated in Figure 20.There are options as x64 and x86. The suitable option can be downloaded. Note that SyncTrayzor is the name of the software on the desktop PC

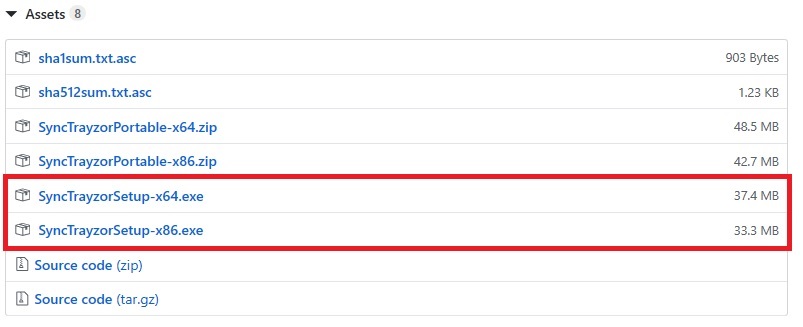


Figure 20 Installing Syncthing

Likewise, the app ‘Syncthing’ must be installed on the mobile device. It can be [downloaded](https://play.google.com/store/apps/details?id=com.nutomic.syncthingandroid) to android devices through Google Play. The logo of the app is illustrated in Figure 21.

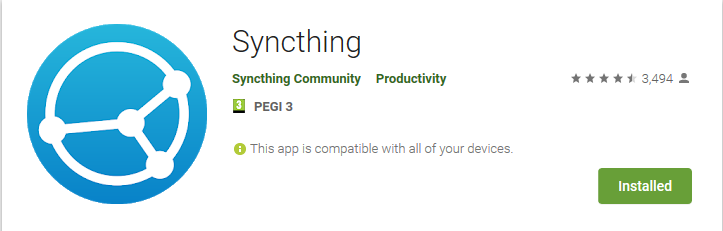


Figure 21. Logo of the Syncthing app

Having installed Syncthing on both the PC and mobile devices, they are required to be connected. Open SyncTrayzor. Go to “Actions” 🡪 Show ID as shown in Figure 22.

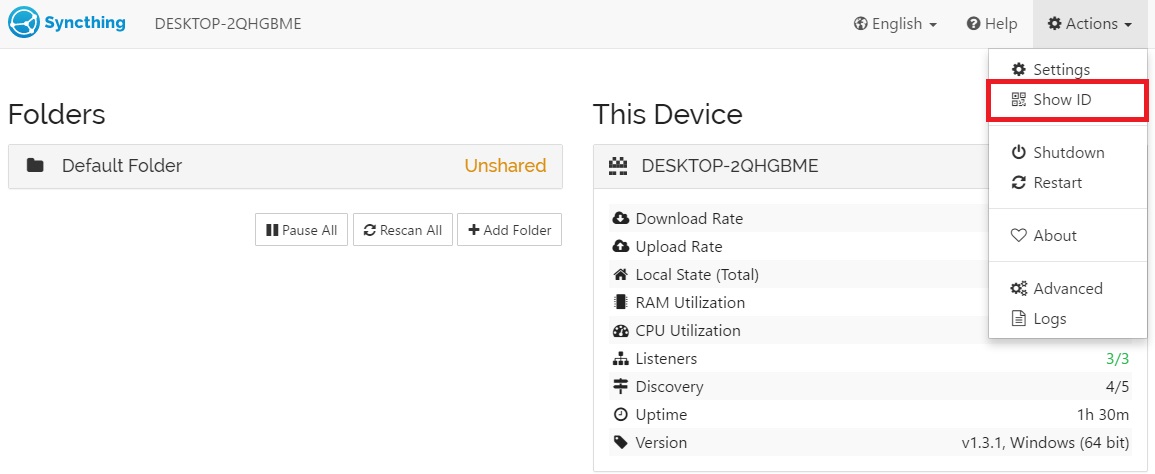


Figure 22. Getting the ID of the PC where each photo will be uploaded

The ID of a device is presented as a text and QR code as shown in Figure 23.



Figure 23. ID of a Snycthing device

This code must also be entered to the mobile device. Press the “+” button in the “Devices” menu as shown in Figure 24.



Figure 24. Synching the mobile device with PC

Device options will be shown. Enter the connection ID and connection name. Also, QR code can be used to set up the connection as illustrated in Figure 25.

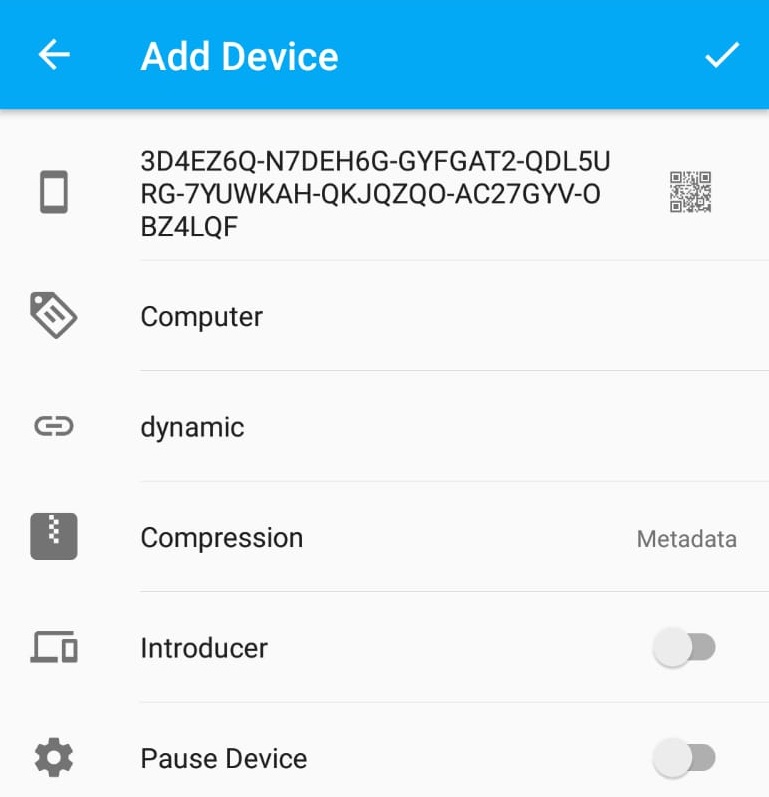


Figure 25 . Adding a device from mobile phone

Once the device is added, this action sends a request to the PC stating the details of the connection. Press the ‘Add Device’ as shown in Figure 26.

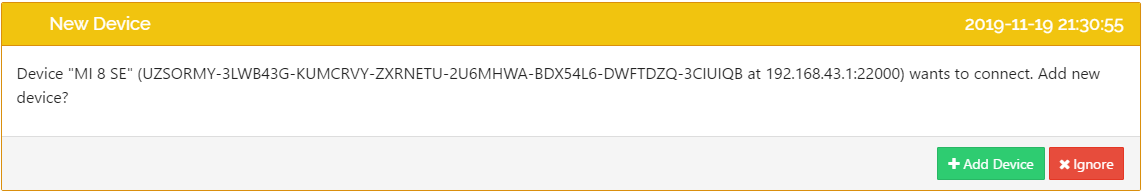


Figure 26. Accepting the request sent from the mobile device

After acceptance, connections will be shown in the SyncTrayzor as illustrated in Figure 27. The smart phone added in this tutorial is MI 8 SE.

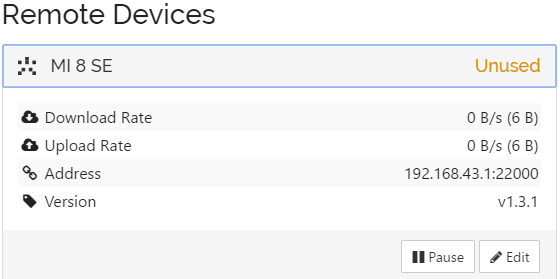


Figure 27. Connected devices – PC view

Also, Syncthing on the mobile device displays the connection information as illustrated in Figure 28.

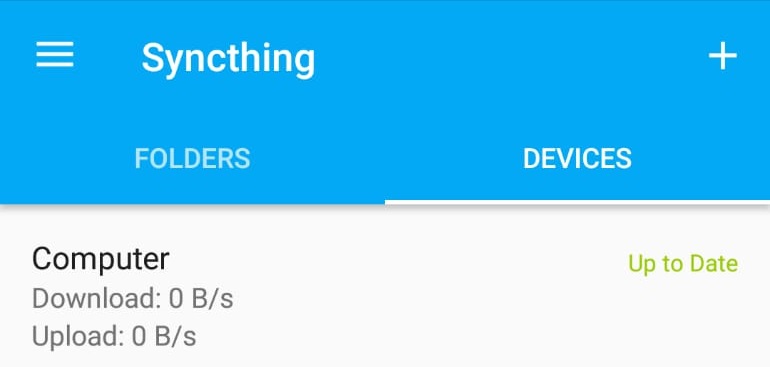


Figure 28. Connected devices – mobile device view

After having reached this step, the mobile device and PC are connected. One may add further mobile devices. Once the connection is established between the devices, the details regarding file synchronization must be provided.

First of all, both Syncthing and Synctrayzor needs to know that photos are saved in a folder named ‘DCIM’ by default. Therefore, a folder named ‘DCIM’ opens in the project folder. It is important to remember transferring the ‘DCIM’ folder when transferring the project from the PC to the mobile device.

## 3.2. Updating the QGIS Project

The path of the captured photos must also be stored in the database. There are two different ways to add a ‘photo’ attribute to the ‘trees’ layer. First, the attribute can be added when creating the table in the database. The data type must be chosen as text. Second, it can be added from the QGIS project. This tutorial relies on the latter option.

Open the attributes table. Right click the ‘trees’ layer and press open attributes table. Firstly, press the ‘Toggle Editing Mode’ which is shown as 1 in Figure 29. Secondly, press the ‘New Field’ button which is shown as 2 in the same figure.

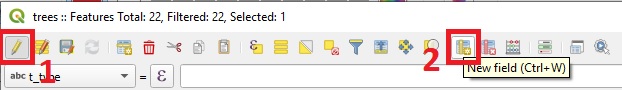


Figure 29. Adding a new attribute from QGIS

After that, enter the column name and select the type. Select type as ‘Text, unlimited length (text)’. Then, press OK as shown in Figure 30.

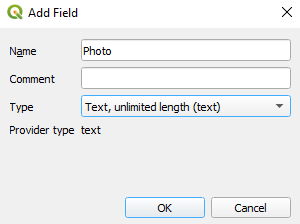


Figure 30. Adding a new field in QGIS

Close the ‘Toggle Editing Mode’ and press save.

The following adjustments have to be satisfied to display photos on QGIS:

1. Right click to ‘trees’ layer and press ‘properties’
2. Go to Attributes Form
3. Select the ‘Photo’ column
4. Select ‘Widget Type’ as ‘Attachment’
5. Enter default path of DCIM folder
6. Select ‘Relative paths’

These steps are shown in Figure 31.

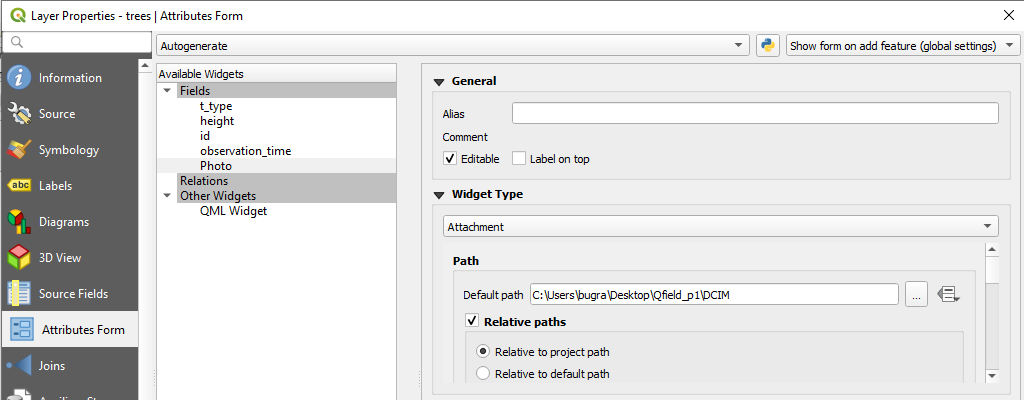


Figure 31. Configuring the photos field – part 1

Continue with the following steps:

1. Storage Mode select ‘file paths’
2. Display resources path option is selected
3. Also, ‘Display button to open file dialog’ option is selected as shown in Figure 32.

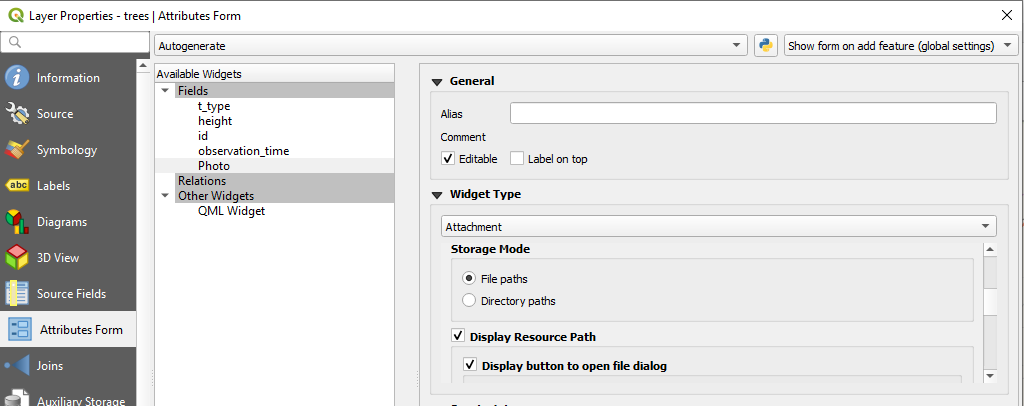


Figure 32. Configuring the photos field – part 2

The last thing to do is:

1. Selecting the ‘Integrated Document Viewer’ as image

After that press the apply button. The project is now ready to take photos.

Last step on photo synchronizing is that of sharing the photo folders. To synchronize the photo folder, press ‘Add Folder’ in the SyncTrayzor as shown in Figure 33.

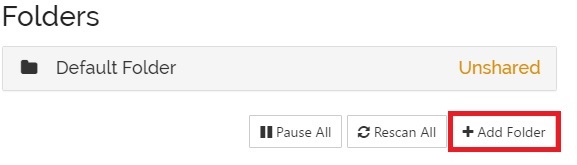


Figure 33. Adding the folder in SyncTrayzor for sharing photos with multiple users

Enter the folder name and folder path. It is important to enter the file location correctly as shown in Figure 34 to make sure that photos are displayed on QGIS.

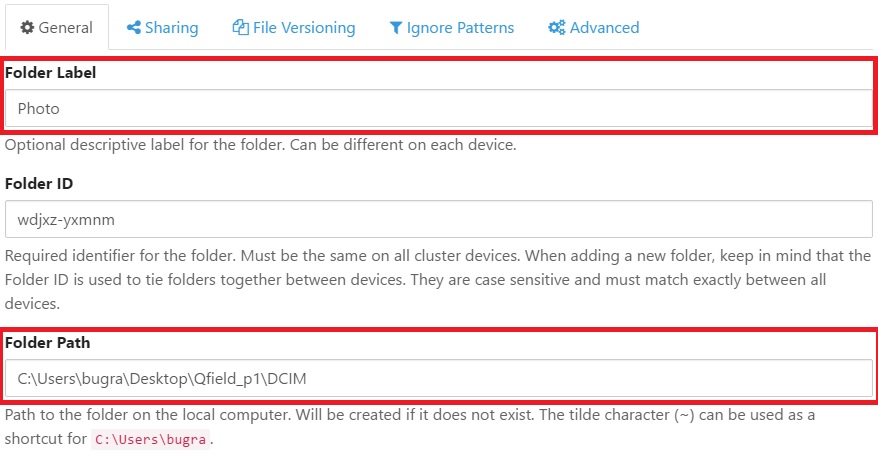


Figure 34. Setting the file synchronisation folder.

Select the mobile devices you want to synchronize the folder that contains photos in the ‘Sharing’ option and press ‘Save’ button as illustrated in Figure 35.

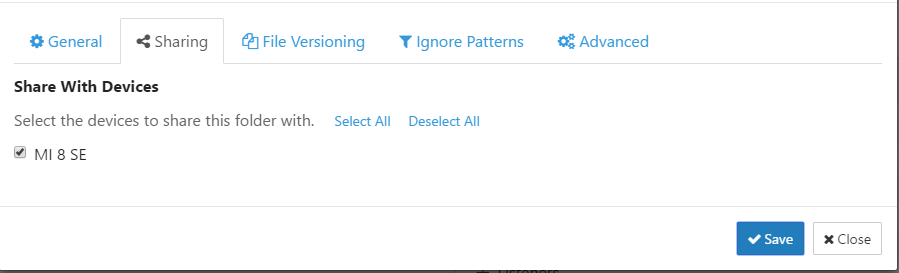


Figure 35. Selecting the devices to share the directory containing photos

This sends a request to the mobile device(s) selected to receive the corresponding acknowledgment from the mobile device user as illustrated in Figure 36.

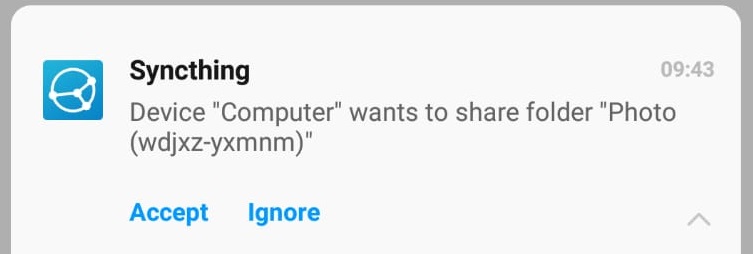


Figure 36.Share request sent to mobile device

Having accepted the request, the location of the 'DCIM' folder in the project must also be selected on the phone as illustrated in Figure 37.

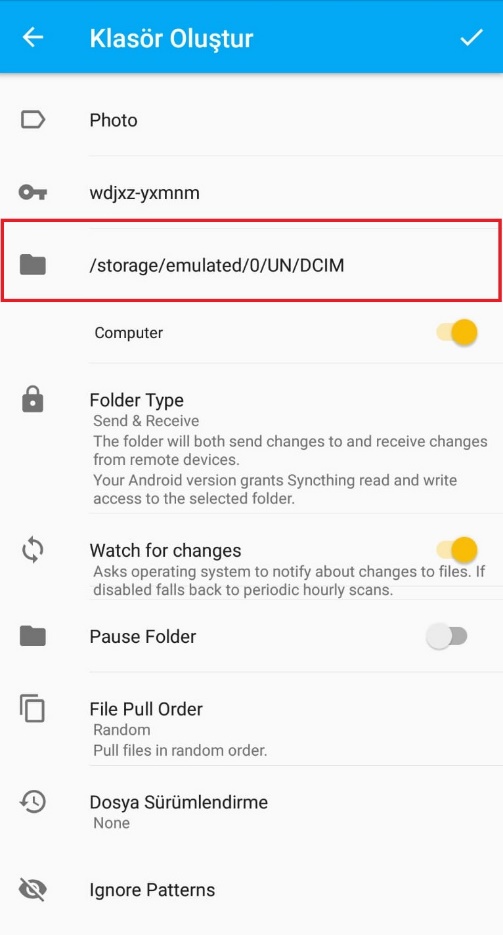


Figure 37. Setting up the mobile device to accommodate synched photos

Now the synched mobile devices are ready to take photos, which will be synched with the PC, as well as with other synched mobile devices.

# 4. 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.

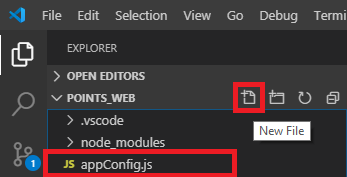


Figure 42 Adding the appConfig.js file

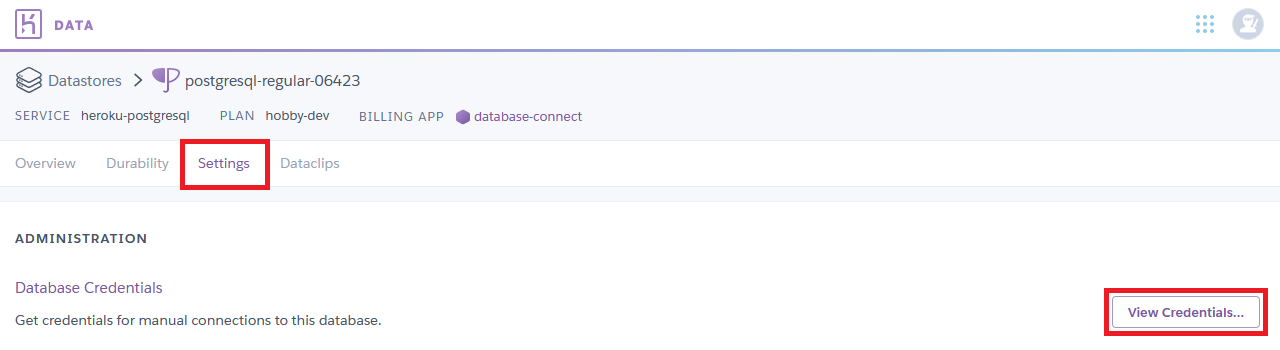
The appConfig.js would include the database connection settings (for now it is the localhost) and the entire content of the file is illustrated in Figure 43.



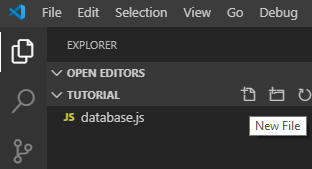
Figure 43. appConfig.js file

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.

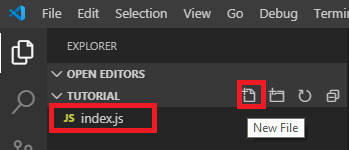
|  |
| --- |
|  |

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.

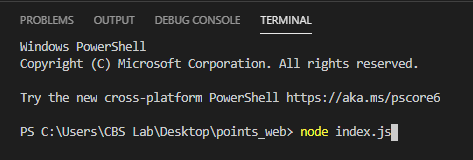


This file provides code to publish the website. It reads the HTML page and it sends a response.



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.

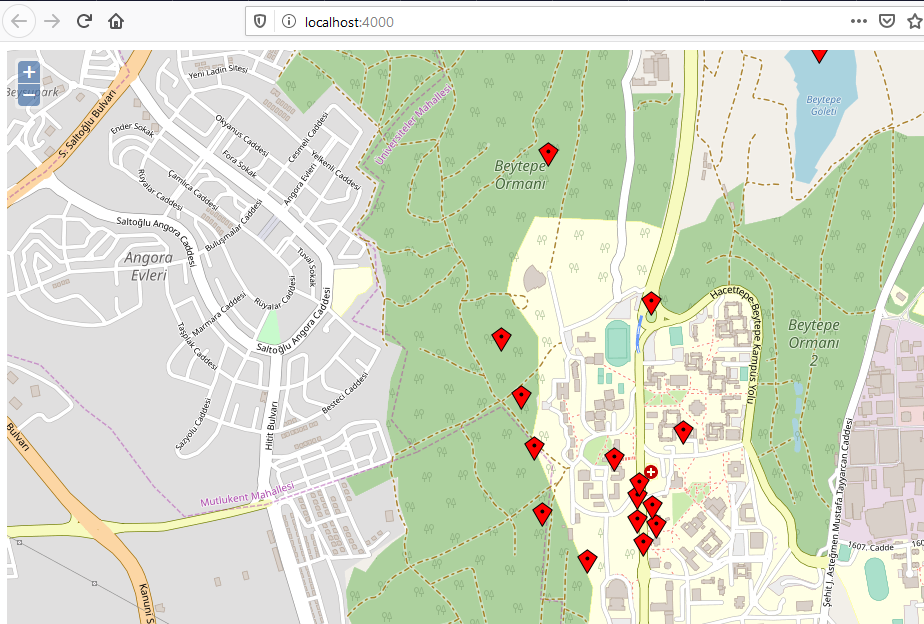
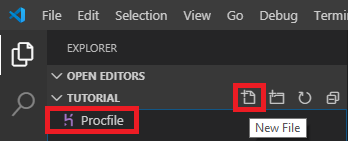


Figure 47. Displaying collected data on the browser

## 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.

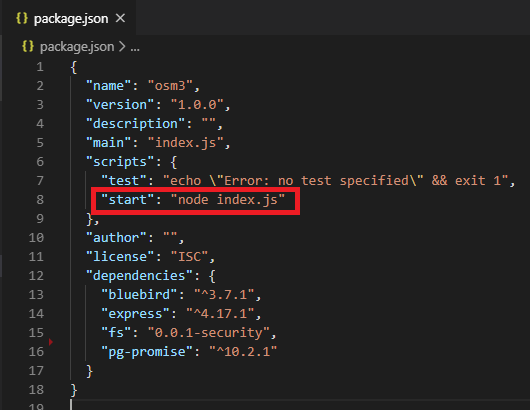
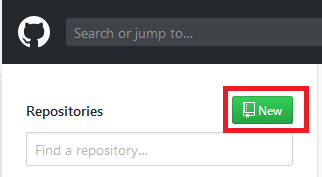
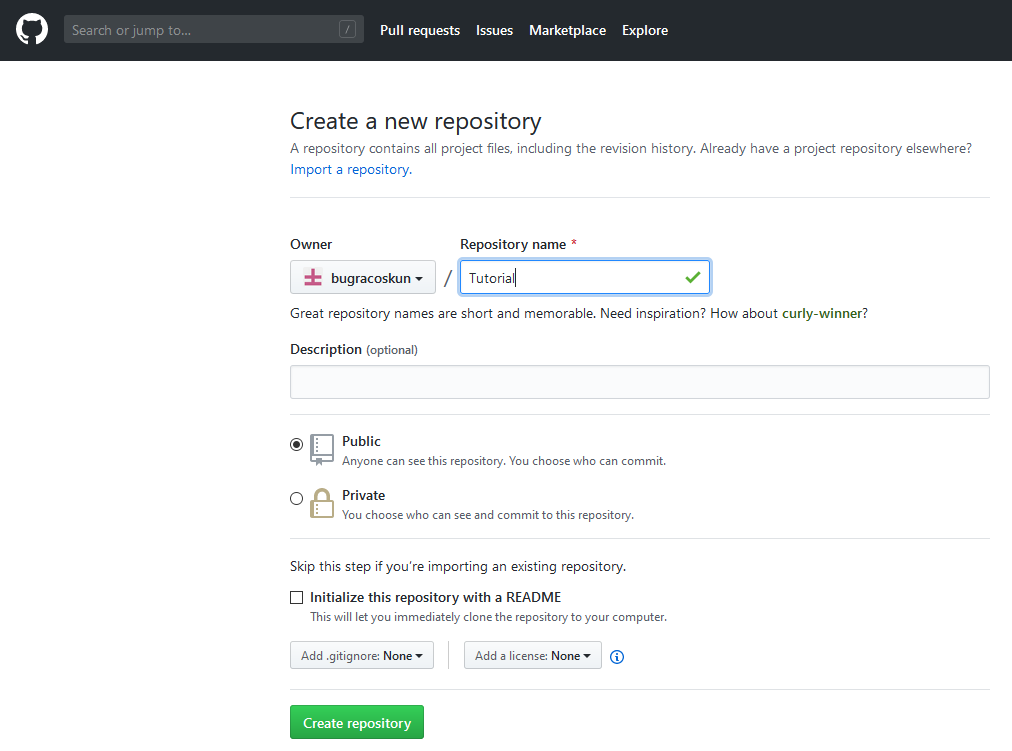


Figure 48. Adding the start parameter to the package.json

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

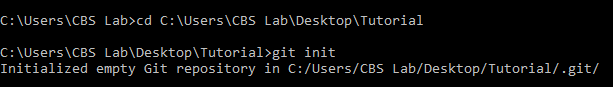


Enter the ‘Repository name’ and press ‘Create repository’.



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”



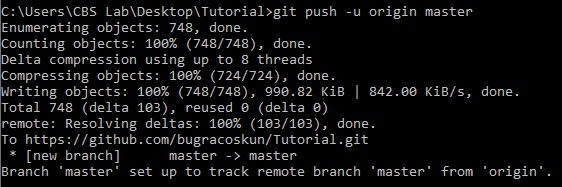
After that, establish a remote connection. Note that you need to write the project link instead of ‘ProjectLink’. It can be done with:

* git remote add origin ProjectLink

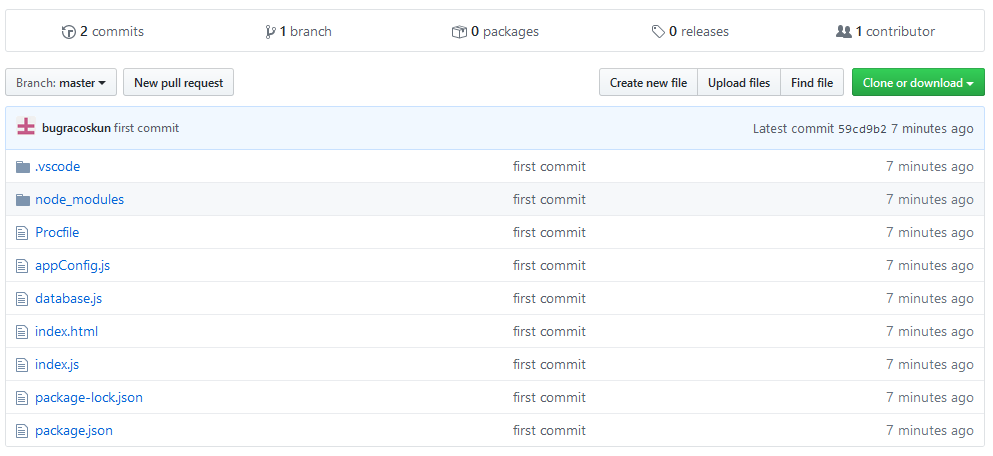


The last step is pushing the files to GitHub. Write this code:

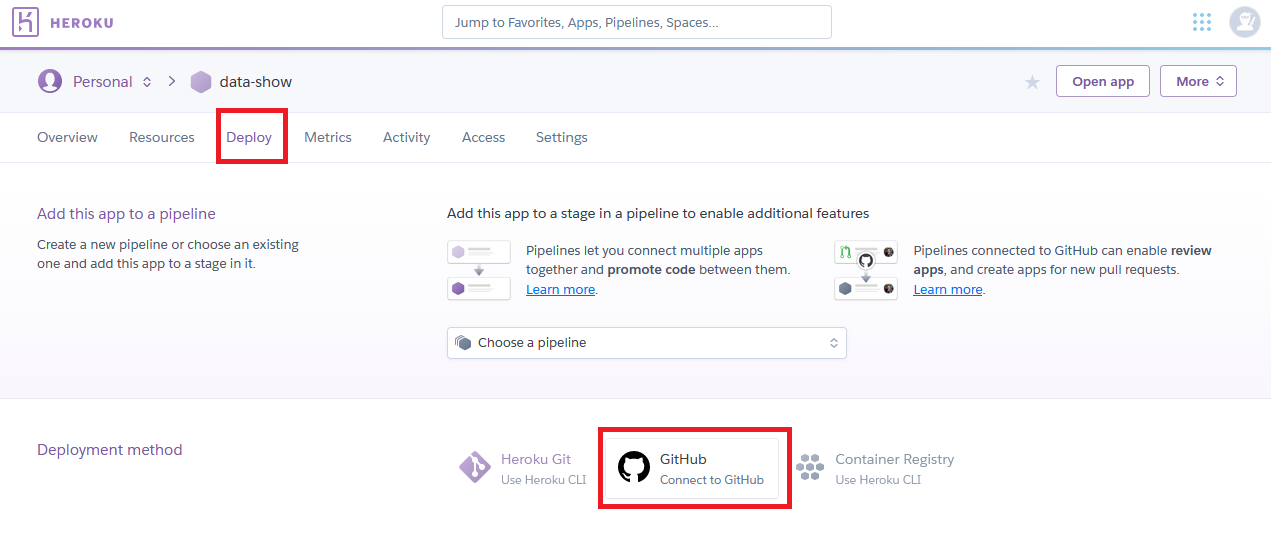
* git push -u origin master



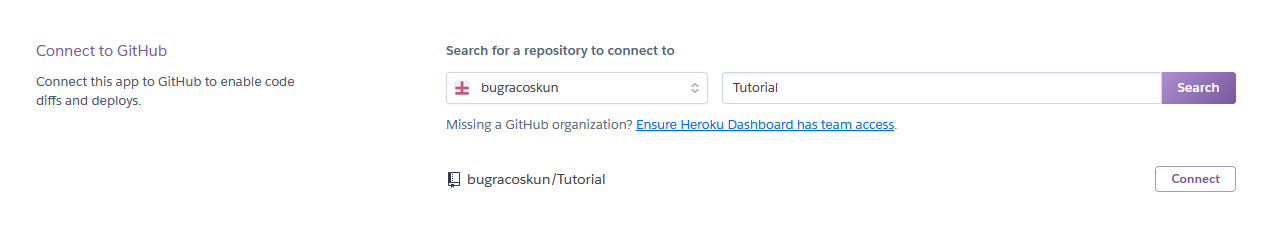
Our files can be seen in the GitHub page.



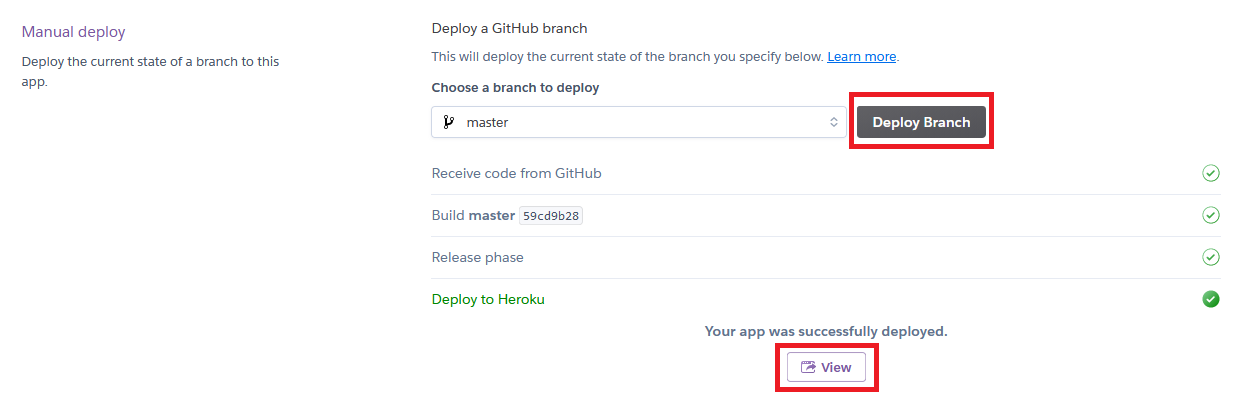
The last step to publish the website is deploying it to Heroku. Go to the Heroku project page and ‘Deploy’ 🡪 Github.



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.

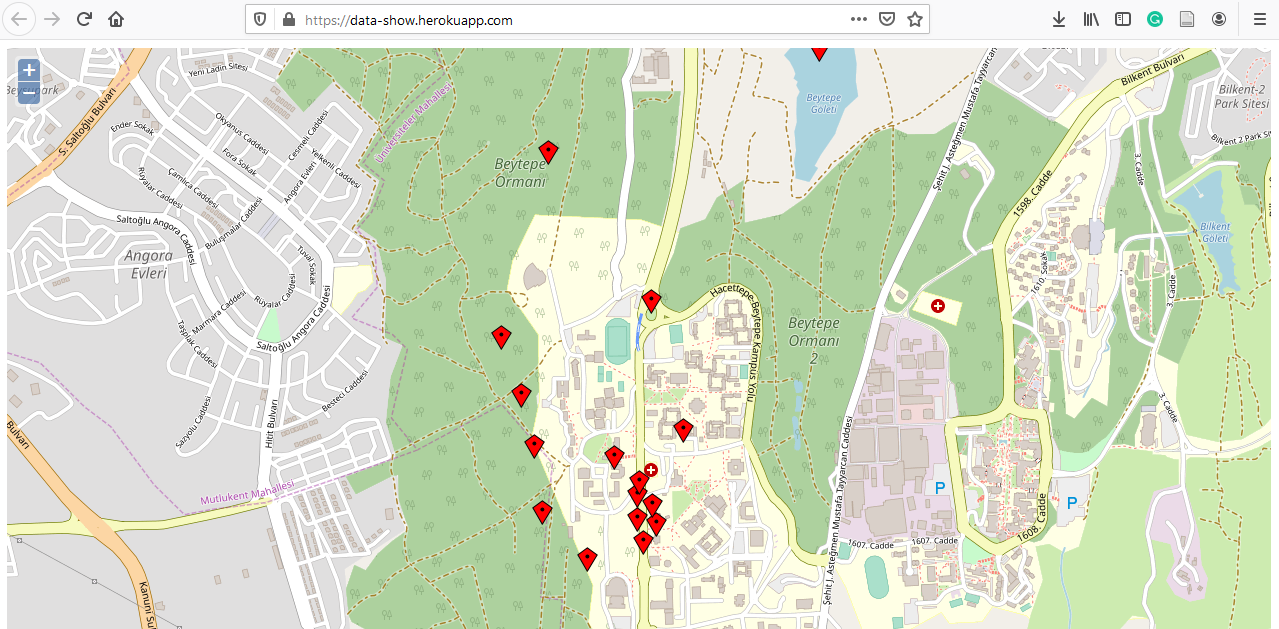


Figure 49 Displaying the collected data

# 5. Concluding Remarks

Mobile spatial data collection is one of the common tasks that UN officers and patrols carry out on a regular basis. Therefore, easy-to-use and flexible means to achieve this improves the effectiveness of day-to-day operations. Some of the following requirements must be met to facilitate mobile data collection:

* Offline data collection: it might be possible that UN officers might be working on sites where there is limited or no access to internet. Even under such circumstances, the data collection framework must allow offline data collection. Whenever internet becomes available, officers could upload the collected data.
* Attaching multimedia evidence such as photos (and possibly other forms including voice records and videos) to events in order to promote peaceful and inclusive societies for sustainable development. By recording multimedia data on complex scenarios, such as an interview with a victim, it might be possible to have a better understanding on the linkage between undesired ‘events’ and their respective ‘locations’.
* Editing of spatial features on the field. One of the critical requirements to ensure responsive, participatory and representative decision-making at all levels is to have up-to-date data. Therefore, updating currently available databases might be as important as collecting new data.
* Defining user roles is important step towards ensuring quality assurance and quality control (QA/QC); since different users of the same system would usually have different privileges.
* Security of collected, both in terms of server infrastructure as well as providing the means to achieve anonymity, would be important to ensure a smooth data collection process.

This tutorial addressed several of the aforementioned requirements. However, some of them remained unmet. For instance, it was not possible to define user roles due to the limitations of the Heroku platform providing free account. Specifically, it is restricted to execute the SQL command ‘create user’, which would be necessary to define the user privileges. Second, the credentials of the database connection is legible in the QGS file that is transmitted to the mobile devices possessing QField. Even though only confirmed officers would possess this file, it might still be a security breach.

The developments on providing cloud technologies offer the potential to satisfy the aforementioned requirements. The readers of this tutorial are encouraged to follow the developments on [QField Cloud](https://qfield.cloud/). Integrating such developments into the current tutorial would substantially improve the data collection and update operations in the field.