# Final Data Visualization Project

Karla Aniela Cepeda Zapata
Department of Informatic and Mathematics
Dundalk Institute of Technology
Co. Louth, Ireland
d00242569@student.dkit.ie

## I. Introduction

Bicycle-sharing service (BSS) is a self-service rental system in which bikes are available on a short-term basis for a price or free. Ireland's open data portal (data.gov.ie) and Smart Dublin (data.smartdublin.ie), as a way to promote innovation and transparency, have open bike-sharing data from Dublinbikes, Bleeperbikes, and Moby Bikes. For this project, the data to be used is from the business *Moby Bikes*.

For this project, we are required to create a multipage application using Dash Apps, a data visualization software that transforms data from custom and standard objects into an interactive system of clickable charts [4][5]. The application has the following structure:

- **Home**. This page shows a general description of the multipage application, source, link to code, and License statement from original data.
- **Dashboard**. This is where the most important information is shown as a summary. This information is real-time and cannot be filtered, however, to see data in more detail, it is important to access the order sections: Map, Bikes, and Rentals.
- **Map**. It shows the real-time location of bikes. This could be filtered by Location, Status of the Bike, and Battery Level.
- **Bikes**. This page shows the status of the bikes in real-time. It is shown the battery level and the physical status of the bike. This could be filtered by: date and/or hour.
- **Rentals**. This page shows the status of the bikes in real-time. It is shown the battery level and the physical status of the bike. This could be filtered by: date and/or hour.

The multipage was designed to monitor bikes' status and location. This is not aimed at the public, but *for the staff from Moby bikes* to keep track of the bikes.

The framework of this project was based on the CRISP-DM methodology. Figure 1 shows the whole life cycle of the project. In further sections, I will explain in more detail about the stages, especially the once filled in grey colour. The technology implemented in this project is enlisted in TABLE I.

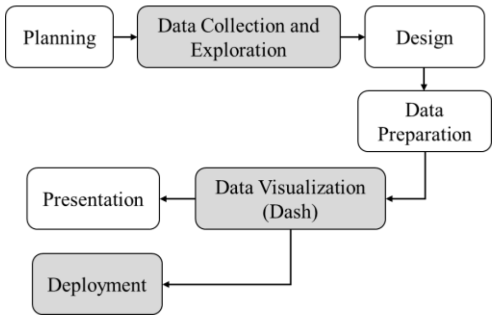


Figure 1. Life cycle of the project.

| **Technology** |
|---|
| For Database |
| MySQL |
| For Data Collection and Preparation |
| Programming language: Python (IDE Spyder)<br>Libraries:<br>• pandas<br>• numPy<br>• requests<br>• shapely<br>• ftplib<br>• mysql.connector<br>• sqlalchemy |
| For Data Visualization |
| Dash (including external scripts for Google Analytics and Net Promoter). |
| For Version Control System |
| Git |
| For Deployment |
| Heroku |

TABLE I. Description of final dataset

| **Dates** | **Tasks** |
|---|---|
| 26th of March | Decide on Data Source |
| 29th of March-15 of April | The Design process, Data Collection, Data Preparation, and Data Exploration |
| 22th - 28th of April | Data Visualization (Dash app) |
| 29th of April | Presentation |
| 2nd May | Deployment |
| 3rd May | Submit project on Moodle |

TABLE II. Timeline for workload

## II. DATA SORUCE

### A. *Description of the dataset*

The dataset and access to API were taken from the website Smart Dublin (also there is access on data.gov.ie, however, metadata was better explained on Smart Dublin). This dataset and API provide information about the location, battery levels, physical status of bikes from the business Moby bikes. The API provides current data of the e-bike bike-share scheme in operation in Dublin, and it is updated every 5 minutes. The designated area and bicycle stand data are available as static resources (geoJSON files) [1]. In TABLE III is shown a list to access the original data from Moby Bikes.

Moby is an innovative Irish start-up company focused on developing and bringing to market a range of electric mobility solutions for cities and individuals [2].

| Resource | Website | URL | Note |
|---|---|---|---|
| Datasets | SmartDublin | https://bit.ly/3aVyVZY | CSV files contain data from Sep-2020 to May-2020. |
| API | SmartDublin | https://bit.ly/3aX04Md | There are two get methods: • *Historical:* same data available as CSV files). • *last_reading:* current data from bikes, updated every 5 mins. |

TABLE III. Resources used for final project.

**NOTE:** CSV files were not used for this project, however, historical data was collected by using the get method historical from API.

**NOTE 2:** Beyond this point, when talking about this API, I will refer to it as Moby API.

### B. *Dimentions and Type of variables*

As the data is real-time, the number of observations is changing every 5 minutes. From the historical data, there are more than 818,000 bike data from 2020-09-23 17:00:02 up to now. There are 15 features from the original dataset, which are shown in TABLE IV.

| Column | Type | Description |
|---|---|---|
| HarvestTime | timestamp | Data retrieval timestamp |
| BikeID | numeric | Use for rent bike |
| Battery | numeric | Bike max distance in km |
| BikeIdentifier | numeric | Generally, contains only digits, might contains chars |
| BikeTypeName | text | Every bike has to be of some Bike Type |
| EBikeProfileID | numeric | Every ebike profile has defined Geofence (allowed riding areas) |
| EBikeStateID | numeric | Indicates: {1:'Warning - is in move and not rented',2:'Normal',3:'Switched Off',4:'Firmware Upgrade',5:'Laying on the ground'} |
| IsEBike | text | Bike sends messages to Backend (bike is equipped with electronic, bluetooth etc.) |
| IsMotor | text | Electric engine might be used for ride |
| IsSmartLock | text | Bike has smart lock |
| LastGPSTime | timestamp | Last time bike connected with GPS |
| LastRentalStart | timestamp | Last time bike was rented |
| Latitude | numeric | Bike coordinates if bike is locked out of station |
| Longitude | numeric | Bike coordinates if bike is locked out of station |
| SpikeID | numeric | Might be used for rent bike instead of BikeID |

TABLE IV. Description of dataset

Not all features were used. This would be explained better in section Data Collection and Preparation.

## III. STRUCTURE OF THE PROJECT

The project has been designed as a real-time application. The architecture of the app in shown on Figure 2.

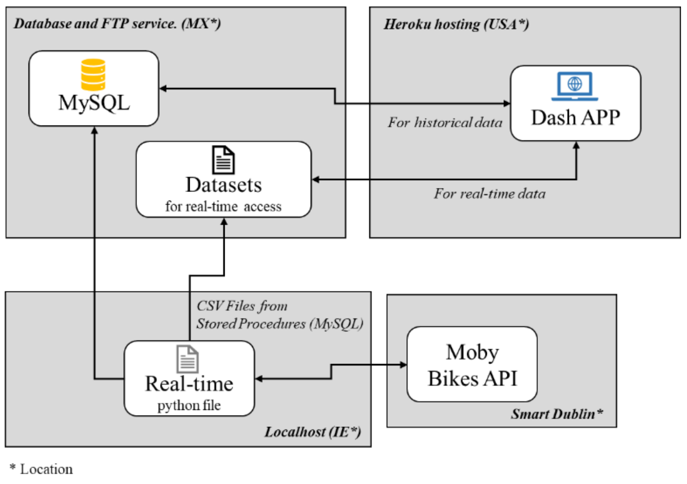


Figure 2. Structure of Project.

It is important to point out the following:

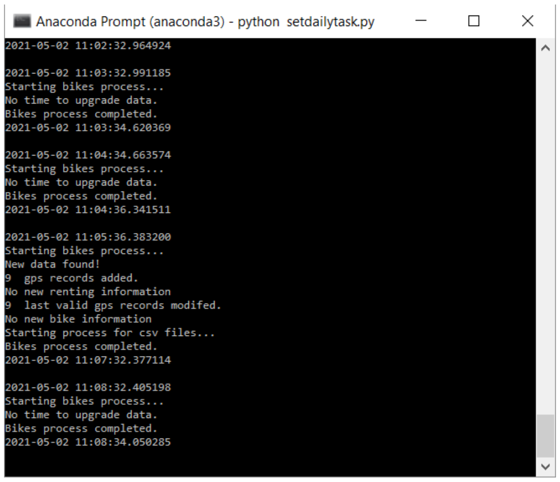
- Database and FTP file service are located in Mexico, as I have a personal hosting services provider. The domain is called "tanniestudio.com". This hosting has never been used, therefore I decided to use it for this project.
- When deploying my app into Heroku, by default the region selected in USA. When I realized this and found out there is another region called "Europe", I tried to deploy it into Europe by creating a new app. However, during the push process with git, it says that "*Heroku*" wasn't a correct command. When trying again with region USA, it worked perfectly (and I was doing the same process), so I decided to leave region USA.
- As I had to set a task to run every minute, I was not sure how to deploy these python files into Heroku, therefore I decided to keep the real-time upgrading

process running into my computer (bad practice, but I did not have time to look into it).

- Additionally, I had to set the time zone as “Europe/Dublin”.

This process is based on the fundament that the Moby API upgrades every 5 minutes bikes’ information. Therefore, this application was designed to retrieve and refresh the information every minute. The process follows these steps:

1. Python file *set_dailytask.py* collects all historical information and prepares other tables into MySQL database. Then, it sets a task that will be executed every minute to ask for new bike information to the Moby API.



```
2021-05-02 11:02:32.964924

2021-05-02 11:03:32.991185
Starting bikes process...
No time to upgrade data.
Bikes process completed.
2021-05-02 11:03:34.620369

2021-05-02 11:04:34.663574
Starting bikes process...
No time to upgrade data.
Bikes process completed.
2021-05-02 11:04:36.341511

2021-05-02 11:05:36.383200
Starting bikes process...
New data found!
9  gps records added.
No new renting information
9  last valid gps records modifed.
No new bike information
Starting process for csv files...
Bikes process completed.
2021-05-02 11:07:32.377114

2021-05-02 11:08:32.405198
Starting bikes process...
No time to upgrade data.
Bikes process completed.
2021-05-02 11:08:34.050285
```

Figure 3. Example of set_dailytask.py

2. When new information has been found, *getDailyBikes.py* file is executed and it collects the last reading, preparing information (cleaning process) to save into the database. Then, *preparingcsvfiles.py* file is executed to save specific CSV files into a public repository for real-time access.

**NOTE:** I have tried to connect to MySQL straight away every minute, however, it shown a bad performance.

## IV. Design Stage

An initial collection of the data has been carried out to perform an exploration of the data, as I have already seen the structure on the website. The exploration allows me to start with the design process. For the design process, there were three important tasks I had to carry out before coding the app using Dash App:

### A. *Architecture design.*

In this section, I decided on how to set all python files and the database for the real-time app. The outcome of this task is the structure of the project, which has been discussed in section III. STRUCTURE OF THE PROJECT.

### B. *Database design*

After the exploration, it was critical to normalize the dataset in a way in which will be suitable for the database. Therefore, the architecture chosen for the database is as shown in Figure 4. TABLE V shows the description for each table shown in Figure 4.

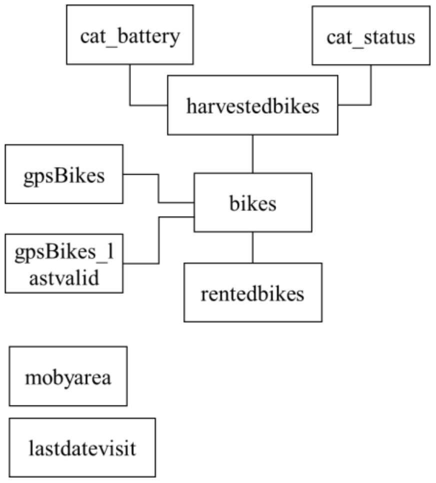


Figure 4. Structure of database in MySQL.

| Table | Description |
|---|---|
| bike | It contains list of all the bikes recorded. |
| harvestedbikes | It contains data from battery levels and physical status of bikes. This data is upgraded every 5 minutes. |
| gpsBikes | It contains the last location read from the GPS devices on each bike (this is just an assumption, as it is not stated on the metadata, however, *LastGPSTime* is not the same as *HarvestTime* when comparing an observation). |
| gpsBikes_lastvalid | It contains the last valid latitude and longitude data, as bikes might get lost (probably due to GPS lost sign, or battery level death. Again, this is just an assumption). |
| rentedbikes | Date and time in which a specific bike has been rented. |
| cat_battery | Catalogue to bin values on Battery feature. Options: Perfect (100-60), Good (59-20), Warning (19-1) and Dead (0). |
| cat_status | Catalogue for the physical status of the bikes. Options: in move – no rented, normal, lying in ground, switch off, and firmware upgrade. |
| mobyarea | Geofence for the valid area. |
| lastdatevisit | Last time information has been updated on (my) database. Used to ask for new bike data on the Moby API. |

TABLE V. Description of the tables in database.

### C. *Dashboard design*

The design of the App was based on the Dublinbikes online dashboard. Figure 5 shows a screenshot of this. To see more about this dashboard, follow the link https://www.schemestats.bike/schemes/dublinbikes/dashboard .

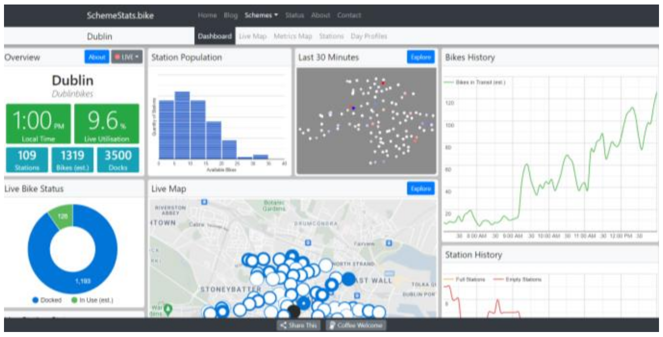


Figure 5. Dashboard from Dublinbikes.

The layout for the app was taken from the example sources from Dash Bootstrap Components. You can find more information in [https://dash-bootstrap-components.opensource.faculty.ai/examples/](https://dash-bootstrap-components.opensource.faculty.ai/examples/).

## V. Discussion on App

### *A. Strengths and Weaknesses*

- Strength:
  - The map functionality.
  - Filters well performed.
  - Color Palette chosen.
  - Real-time structure.
  - The general design of the website.
  - Net Promoter Score and Google Analytics (with Privacy Policy) included
- Weaknesses
  - When plotting maps, not an easy way to show more information when hovering.
  - No historical data available as a whole.
  - The structure of the project. I wish I could deploy the 5-minute task online not to keep my laptop on all the time.
  - I should have used a database located in a different region (not Mexico).

### *B. Regarding plots*

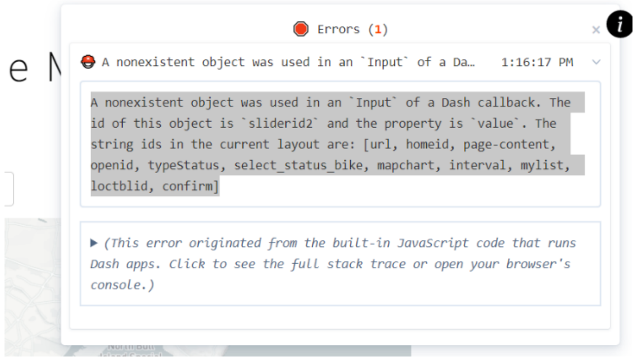
- Plotly has been a really easy to use like Matplotlib and Seaborn. I feel the learning curve was the same as when learning Matplotlib and Seaborn, and the documentation was really helpful.

- The map functionally was really interesting to look into as this feature was not shown in the lectures.

### *C. Regarding dash*

- Interesting to work with Dash App, as it is the first time, I have work with this technology. Learning and understanding Dash App was a bit complicated than Plotly especially for the callback function when adding filters to graphs. However, once you get the basic structure of a Dash App, it becomes easy to code.

- I had issues with a callback as strangely it says an object ID in an Input was found in different pages (same ID, and just used on page Bikes). Figure 6 shows the error I got. I did find the solution by creating a new page, and start adding plots and filter one by one. I copied everything the same, I did not understand what was the problem but the error is not shown anymore.



```
Errors (1)

A nonexistent object was used in an `Input` of a Da…    1:16:17 PM

A nonexistent object was used in an `Input` of a Dash callback. The
id of this object is `sliderid2` and the property is `value`. The
string ids in the current layout are: [url, homeid, page-content,
openid, typeStatus, select_status_bike, mapchart, interval, mylist,
loctblid, confirm]

▸ (This error originated from the built-in JavaScript code that runs
Dash apps. Click to see the full stack trace or open your browser's
console.)
```

Figure 6. Error when coding.

- Easy to incorporate a Net Promoter Score into the Dash App. The one used is [https://survicate.com/](https://survicate.com/).

- I found it difficult to figure out how to include Google Analytics script into Dash code. I found a solution by customizing Dash's HTML index template. For more information visit [https://www.programmersought.com/article/43425885776/](https://www.programmersought.com/article/43425885776/).

### *D. Regarding project structure*

- When the description of the project was available and we have the option to use real-time data, I wanted to face the challenge of building it. And THAT was a challenge as it was my first time coding a real-time app.
- Dash App makes it simple to refresh the page every minute for new information.
- By using the task library on Python, it was easy to collect new bike data from Moby API every 5 minutes.
- Another challenge was the structure of the database, as I had to normalize the original data.
- Additionally, finding the right way to stream real-time data took me time to come up with an idea, as connecting every minute into the database was ineffective. The solution: creating CSV files from relevant data (with format and data cleaned) for each page into an FTP repository. You can see all the files Dash App access here [https://mydata.tanniestudio.com/datasets/](https://mydata.tanniestudio.com/datasets/).
- Inconsistency when upgrading CSV files for real-time data and refreshing the pages. The process might not finish but in the website might shown different information as some CSV files have been upgraded. I did not have time to address this issue.

## VI. ONLINE APP

The app has been deployed using Heroku, a platform that enables developers to build, run and operate applications entirely in the cloud [3]. The app is online and available at the following link:

- Name of app: MobyBikes
- Link: https://mobybike-app.herokuapp.com/.
- Author: Karla Cepeda

We invite you to leave feedback!

## VII. SCREENCAST

- Link: https://bit.ly/3vE7kVj
- Author: Karla Cepeda

| **Time** | **Description** |
|---|---|
| 00:00-10:16 | App Demonstration |
| 10:16-11:13 | Google Analytics and Net Promoter Score |
| 11:13-13:49 | Explain Structure of App |

TABLE VI. Description of video

## VIII. ETHICAL CONSIDERATIONS

This data and metadata are associated with the Creative Commons Attribution (CC-BY) Licence. Under the CC-BY Licence, users must acknowledge the source of the Information in their product or application. Where the Information Provider does not provide a specific attribution statement user should include, or link to, this attribution statement: “Contains Irish Public Sector Data licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence [6].

## REFERENCES

[1] Smart Dublin. *Moby Bikes bikeshare*. Available from: https://data.smartdublin.ie/dataset/moby-bikes [*accessed 02-May-2021*].

[2] MOBY. *About us*. Available from: https://mobybikes.com/about-moby/ [*Accessed 02-May-2021*].

[3] Heroku. *Could Application Plataform.* Avaliable from: https://www.heroku.com/ [*accesed 02-May-2021*].

[4] Plotly. *Overview*. Available from: https://plotly.com/dash/ [*accessed 02-May-2021*].

[5] Financesonline. *Dash Review and Pricing*. Available from: https://reviews.financesonline.com/p/dash-app/ [*accessed 03-May-2021*].

[6] Data.gov.ie. *Open Data License*. https://data.gov.ie/pages/opendatalicence [*accessed 03-May-2021*].