Integration of Toponyms Mobile data Collection in PostgreSQL Database
(A mobile application for toponyms of Kenya, Django REST API open source approach)

Daniel O. Nyangweso[1] – Mátyás Gede[2]

[1] Eötvös Loránd University, Budapest, Hungary, Department of Cartography and Geo-informatics, danielorongo@gmail.com
[2] Eötvös Loránd University, Budapest, Hungary, Department of Cartography and Geo-informatics, gedematyas@gmail.com

Abstract: Data integration involves getting data from different sources and viewing it on a platform. Data synchronization involves connecting the data entry and archiving in PostgreSQL database either in a server or mobile device in a two way mechanism. The aim of this paper is to show how the toponyms (place names) data can be synched in a local PostgreSQL database as mobile data collection exercise progresses using Django REST framework, Python libraries and other software. A toponyms project was created containing repositories builds on Bit Bucket cloud and endpoint generated for the toponyms file. A web client was then created to expose the endpoint that receives data from the android application hence connecting the mobile app. The mobile app also has OSM base map and data is collected and archived in real time as it is collected, in a PostgreSQL/PostGIS database backend. The process involves maintaining different file formats on each platform without the users notice. The online data back up by the devise enables secure back-up due to syncing with hosting server which renders the data both on an online cloud and a local computer storage. Similarly the OSM data can either be imported using QGIS quick map services plugin or be manually built and synched in a local disk of a computer. The mobile data collected, OpenStreetMap and validated data may then be running side by side or be displayed as an overlay. The endpoint, app and PostgreSQL communicate through the model. Data is sent through the endpoint which triggers data processing and storage into PostgreSQL throughout as long as there is internet connection. The integration eliminates data entry, minimizes editing and provides up to date fieldwork data. The mobile APK generates data for personal use and also for mapping organization(s) or agents of governments handling geographical names (after customization). Also, the app can be used to update maps data, after validation, without actual fieldwork; hence cutting cost since all software and tools used are all open source including free data collection using VGI. Further probes and analysis needs to be done on the two sources of VGI data to determine their characteristics.

Introduction

Good framework of Internet of things communication must be employed to check quality data flow and efficient validate. One may question whether the application environment is an html based or web based and how different data conversion challenges involved when data doesn't reach the backend. One way to solve include exporting the SQLite data from a toponyms mobile app is pushed to a website or manually converted into GeoJSON format using QGIS. The file is then added back to be viewed as an html document. Similarly, formal data can be reformatted to GeoJSON data and also be added as a GeoJSON layer as an overlay. The two data

sources can then be compared in terms of content, visibility and coverage to identify salient features of names depicted on each of the three sources of data for further analysis. In the Django administration JSON and XML data types can be displayed. In some instances data interchange for some devices may face challenges. A solution is sought such that you collect data, you check its availability on the device collected, on the web service and also on the local computer storage archive. In this case no data is lost, there is faster mobile data collection. The two data sources; mobile APK data (collected continuously) and official gazetteer (initially imported) are displayed separately with options for additional layers to overlay with on check boxes

Updating official gazetteer data may be simplified though synchronizing data collection to data storage through an online platform. The process not only hastens the data collection process but also enables monitoring the field work, data capture and synchronization and aids analysis validation procedures and minimizing or eliminating costly actual field visits which are budget dependent yearly.

Open source software and libraries

Manual building of map tiles (OpenStreetMap Contributors 2018) provides a platform to build own customized maps with OSM updates from all over world according to the user needs and also locate places where similar or comparable toponyms are used. Various users have been applied use of Mapnik rendered tiles such as David Rumsey (Rumsey 2018), whereby historical maps were scanned and can be compared with OSM and Google maps. In rendering the files Mapnik tool (Wiki 2019) and Osmosis (OpenStreetMap Wiki 2018) java command line tools are used. A toponyms mobile application was developed using android studio (Nyangweso – Gede 2018) published in play store, specifically to handle the toponyms data collection framework. Opens source tools and libraries exist such as Bitbucket (Attlassian 2019), Django REST API Framework (Django 2019) and python (Foundation 2018) among others.

Materials and Methods

Mobile data collection Workflow

Data collection on the server and serving on the website

Figure 1 shows the mobile data collection and synchronization workflow. It enables data storage to the local and or online PostgreSQL database.

Tools, data and resource mobilization

Software and platform used

Table 1 shows the tools and open source software used in the project.

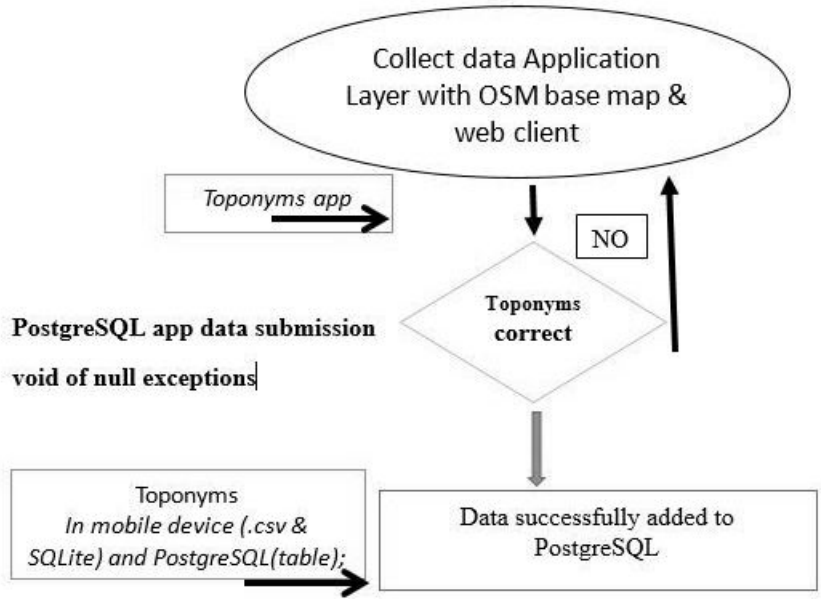


Figure 1. Data collection and posting Workflow

Table 1. Tools. Software and Platforms used

	Platform/tool/software	**purpose**
1.	Visuo Studio Code	Code editing
2.	QGIS	Data processing, preparation and display
3.	Django 2.15.0/Python 3 and Pip 8.1	Development of API
4.	OSGEO	Provision of geometry abstraction of spatial data
5.	Leaflet	JavaScript libraries
6.	Postgres/Postgis	Provision of back end for data storage
7.	Nginx web server	Web server service

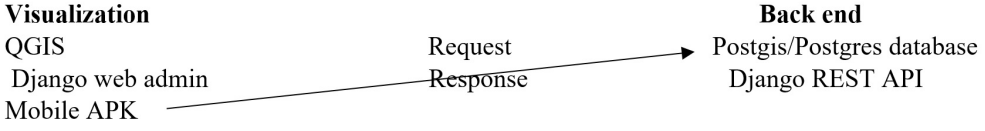
Data to be integrated

1. Validated gazetteer (only names from Survey of Kenya Records; only for a few places.

2. OpenStreetMap data view using locally build OSM tiles or using Qgis quick map service

3. Mobile APK data collection using smart phones with background OpenStreetMap display.

Data integration Implementation strategy



Resources needed:

- Internet connections in the APK and web services.
- Toponyms mobile APK integrated with Postgres through Django REST API
- Web server and Web Client, powered by Nginx Webserver with all associated software such as Python, OSGEO, GDAL
- QGIS quick maps plugin for rendering the OSM map or use Mapnik open source toolkit for rendering OSM maps and cache them locally.

Sample mobile data collection involved:

1. Testing the data collection by the app.
2. Data type rendered in the web platform and downloadable data.
3. Data storage and editing via the Bitbucket repository hub.
4. Filling the data collect form for all entries.
5. Adding the data to the device database.
6. Submitting the data to the PostgreSQL database and visualizing it in the postgis interface through QGIS.

Data Validation: Django REST Framework and PostgreSQL/Postgis backend

Editing can be on the Django administration site or in the PostgreSQL backend.

a. SQLite data

The required area is extracted using bounding box as map tile extract for a specific region or country, imported and build manually as a local server of the OpenStreetMap files which is sequentially updated. This is done by first launching QGIS and opening the layer with spatial locations and in CSV file format and then saving it as GeoJSON. The file is then copied to a pre-created leaflet folder in Apache, under grouped layer control open source tool.

b. OSM data

Initial data is build using Mapnik map tiler and assembled, served and updated locally as long as there is OSM update. Building a Local OSM Tile server involved use of available open source material and software to render and serve the OpenStreetMap and display the tile layers as synced and rendered using Mapnik carto software found online.Alternatively, one can use the quick map service plug in in Qgis to render the OSM layer as base map

c. Validation data

The OSM2psql data imported is used as base layer. The validation data comes from the mobile APK collected and data extraction and editing is done in Postgres database contained all the data from the three sources. The PostgreSQL data table to be edited has the validation column to ascertain editing has been done and verification made on the OSM and APP data sources. After verification of the data, it is displayed on the website for visualization and analysis.

d. Web Client

A web client was developed to enable visualization of server and web information contained in the toponyms database and overall remote monitoring of the mobile data collection process.

Results

OSM base map as visualization in the desktop and mobile app user interfaces

The added OSM layer provides the base map to assess the spatial display of the points in the desktop user interface.

Figure 2 show a sample Locally Build and Rendered Tiles which can also be added via QGIS quick map services plugin. *Figure 3* shows, the equivalent OSM map as incorporated in the mobile app interface.

Figure 4 shows the OSM base map with current location coordinates as the user starts to collect data.

One can also query the status of the toponyms online service status.

Figure 5 shows activated online Postgres RDMS service after successful login via command line and toponyms web client service as active, once the configurations are complete.

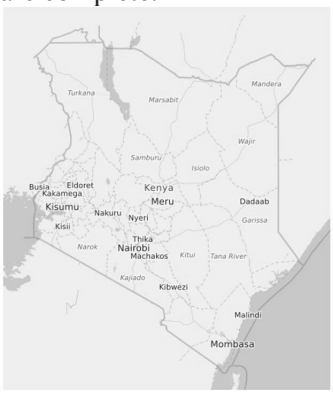


Figure 2. OSM base map, Mobile interface

Figure 3. OSM Base map, Desktop interface

Figure 4. Mobile App OSM

```
● toponyms.service - Toponym Web Client Daemon
   Loaded: loaded (/etc/systemd/system/toponyms.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-03-30 22:21:13 EAT; 52s ago
 Main PID: 5207 (gunicorn)
    Tasks: 4 (limit: 1997)
   CGroup: /system.slice/toponyms.service
           ├─5207 /srv/www/venv/bin/python3 /srv/www/venv/bin/gunicorn --access-logfile - --work
           ├─5229 /srv/www/venv/bin/python3 /srv/www/venv/bin/gunicorn --access-logfile - --work
           ├─5230 /srv/www/venv/bin/python3 /srv/www/venv/bin/gunicorn --access-logfile - --work
           └─5232 /srv/www/venv/bin/python3 /srv/www/venv/bin/gunicorn --access-logfile - --work
(venv) ...
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sat 2019-03-30 22:38:27 EAT; 10s ago
  Process: 5577 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 5577 (code=exited, status=0/SUCCESS)
(venv) ...
```

Figure 5. PostgreSQL and toponyms online service

Data administration and validation

Django administration Login Page

Figure 6 shows a graphic user interface to login to view or edit the online toponyms list or add new users into the system. The administrator ensures quality data collection, validation of the data and overall administration of the toponyms database.

Figure 7 shows data generated by the users of the APK in Django REST framework, where one edit if deemed necessary.

Figure 8 shows the online toponyms feature edit tool. It does not delete the original entry but creates a duplicate. The final edits are to be done in the Postgres server during validation.

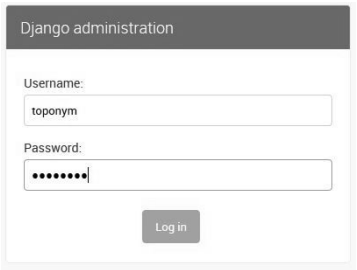


Figure 6. Django administration for logging into web client

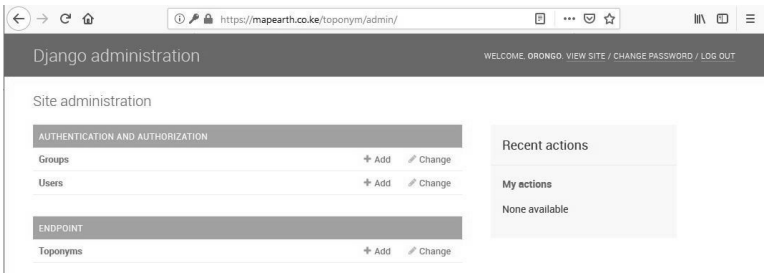


Figure 7. Django Administration

Figure 8. Toponyms feature online edit feature

Bitbucket Repository administration Login Page

Viewing and editing the code or committing a feature edit on the app requires an account with the providers of the Atlassian who are Bitbucket repository services. One can log in using google, any email address or a user name allocated during registration to the administration system.

Figure 9. Login interface

Figure 9 shows the log in interface used to access the repositories for all the project, web client and android studio toponyms files.

Toponyms Repositories

Figure 10 shows the Bitbucket repositories files such as those for the project, Django REST framework web client and android studio mobile application.4.2.3 Django REST framework data visualization.

Figure 11 show the Django REST framework for the toponyms already collected by the mobile application, with entries for id, type, geometry automatically generated after the user of the mobile application fills in the collect data and submits it with no null exceptions to the Postgres database.

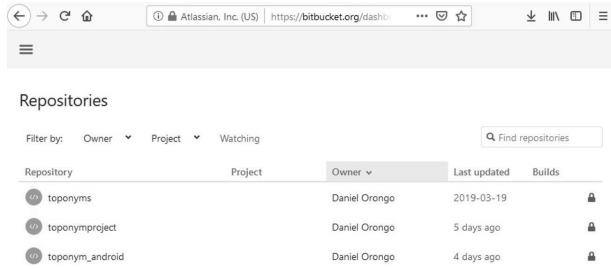


Figure 10. Bitbucket Repositories

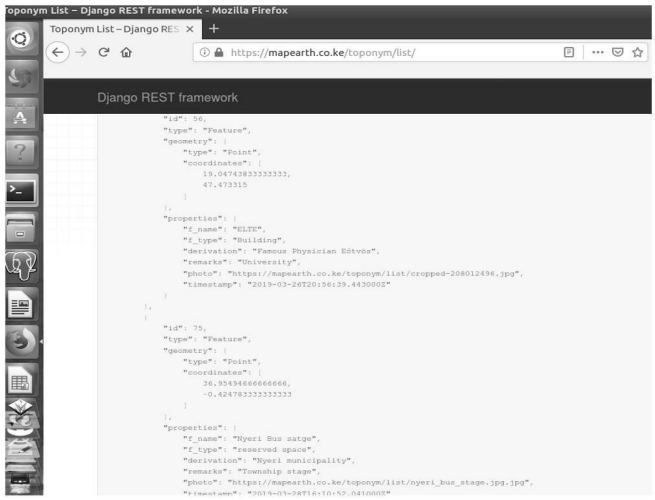


Figure 11. Django REST framework

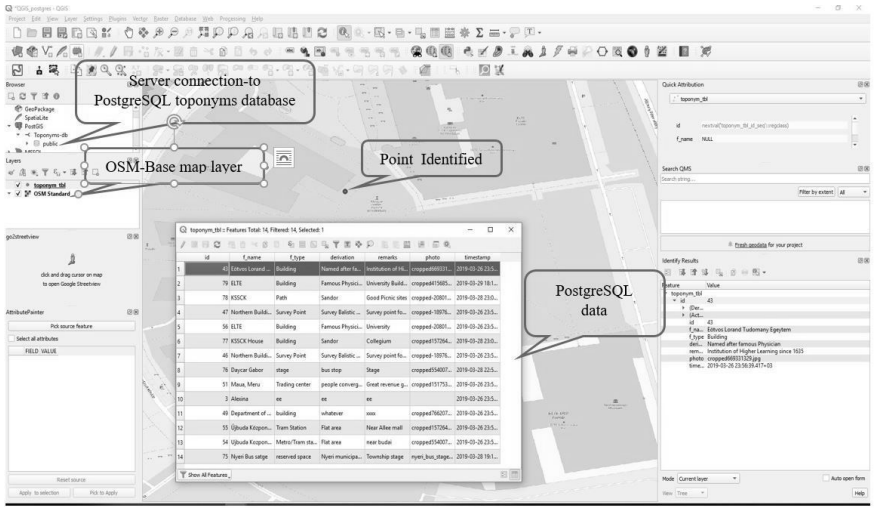


Figure 12. Postgres/PostGIS interface Toponyms table as visualized on QGIS

PostgreSQL/PostGIS server backend interface

Figure 12 shows the PostgreSQL spatial layer properties as visualized in the PostGIS user interface.

References

Attlassian, Bitbucket. Retrieved from Bitbuck Repostory: https://bitbucket.org, Download Time: January 2019.

Django. Retrieved from https://www.django-rest-framework.org, Download Time: February 2019.

Foundation, P. S., Python. Retrieved from Python: https://www.python.org. Download Time: December 2019.

Nyangweso, D. – Gede M. (2018): Toponyms Mobile APK for Volunteered Geographic Information. Lisbon: Geomundus2018. Retrieved from http://www.geomundus.org/2018/docs/papers/Danniel.pdf, Download Time: January 2019.

OpenStreetMap Contributors. Manually building a tile server (18.04 LTS). Retrieved from Switch2OSM: https://switch2osm.org/manually-building-a-tile-server-16-04-2-lts. Download Time: August 2018.

OpenStreetMap Wiki. Osmosis. Retrieved from WIKI: https://wiki.openstreetmap.org/wiki/Osmosis Download Time: August 2018.

Rumsey, D.: Georefrencer and Compare. Retrieved from David Rumsey Map collections: https://www.davidrumsey.com/view/georeferencer Download Time: December 2018.

Wiki, O. Mapnik. Retrieved from Wiki: https://wiki.openstreetmap.org/wiki/Mapnik Download Time: January 2019.