ARCHITECTING LOCATION INTELLIGENCE PLATFORMS USING OPEN-SOURCE COMPONENTS

A. BENAHMED DAHO [1], H. BEKHELIFI [1]

[1] TransformaTek, SARL (LLC), Ain Temouchent, Algeria - (ali.benahmeddaho, hatem.bekhelifi)@transformatek.dz

Commission IV, WG IV/6

KEY WORDS: GIS, Cloud Architecture, Location intelligence, Agile approach, GeoNode, Source control.

ABSTRACT:

Location Intelligence is an emerging application of geospatial industry. It allows professionals from different business domains (finance, disaster management, retail, health) and with minimum expertise in GIS technologies to use efficiently many spatial analysis tools and algorithms to solve their day-to-day problems. One of the major characteristics of Location Intelligence is the use of multisource and alternative data either user ingested or directly available in the platform (by geo-enrichment).
In this contribution we investigate the architectural challenges raised by the development of this kind of platforms, particularly in term of functionalities, use of open-source components, management practices, deployment on the Cloud and source code control. To validate our assumptions, we built a new platform, named "Geoweba", based on the GeoNode Project and having as objective to disrupt the Algerian consulting market. The resulting platform is deployed on the internet and freely available for users. In this contribution we demonstrate that our development approach based on Agile principles and open-source components gives a good result, mainly in the early stages when just a Minimum Viable Product (MVP) is needed for market validation of the idea.

1. INTRODUCTION

Location intelligence software, also called spatial intelligence software, is a business intelligence solution that provides location analytics to identify the relationship between certain objects based on their physical locations. Location intelligence tools enable users to see trends on maps and graphics to optimize certain business opportunities (G2, 2021). Noticing a lack in geospatial technologies adoption by businesses in our Country (Algeria), we decided to develop and deploy a cloud-based location intelligence platform to simplify the ingestion, the management, the visualization and the analysis of open and homemade spatial datasets.

To accelerate the speed of development of our platform we investigate the use of open-source components to build a Minimum Viable Product (MVP). The resulting Agile built platform named "Geoweba" was deployed into a cloud-based virtual machine and made freely available for users.

2. METHOD

2.1 Software modules roadmap

To architect the location intelligence platform, we used an Agile approach combining Lean Start-up process and Scrum rituals, which allows us to improve continually our solution using final user feedbacks. Also, this kind of approach give us the flexibility to update incrementally our TODO list of features (Backlog) and to build iteratively and as needed the required software modules in the form of a GeoApp (Geospatial Application) (Figure 1).

To keep on track, the World Bank Doing Business process (World Bank. 2020) has been used as a roadmap and each GeoApp is related to one of the generic business development cycle, nominally: opening a business, getting a location, accessing finance, dealing with day-to-day operations, operation in a secure business environment.

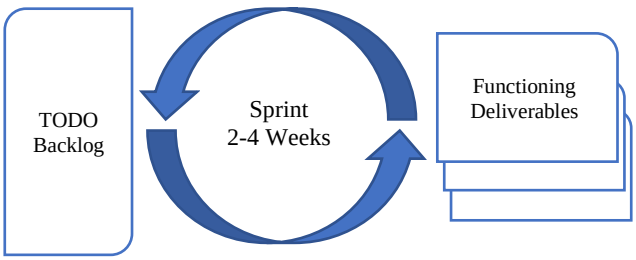

Figure 1. Scrum events and artefacts.

2.2 Functionalities identification

For the kick-off, we built a list of location intelligence platforms common features by analyzing all major commercial brands deployed worldwide (J. McCormick and E. Miller, 2019) and we compare them to the features already implemented in one of the most active open-source projects, namely GeoNode (Corti P, Bartoli F, Fabiani A, Giovando C, Kralidis AT, Tzotsos A. 2019).

This gap analysis (Table 1) was the genesis of the targeted platform backlog and serves as roadmap for the upcoming development activities. Non-functional requirements were also identified from the previous analysis, particularly mail confirmation and social login which are critical for user experience, and also Backup and restoration which are critical for the availability.

3. ARCHITECTURE CONSIDERATIONS

Location intelligence platforms are complex systems involving many interactions between heterogonous components both internal and external to the organization, particularly when they are deployed on a public cloud infrastructure which rises many privacy challenges.

Features	Available
Data Management	
Import layers (local)	Yes*
Import layers (from the cloud: Dropbox, Google Drive, …)	No
Automatic data importing from open data sources using automated scripts	No
Export data	Yes**
Explore geospatial data catalogue	Yes
Pre-existing data library	No
Map creation and editing	
Create maps	Yes***
Share maps	Yes
Data search:	
− By name	Yes
− By coordinates	Yes
− Using spatial query	Yes
− Using attributes query	Yes
Attributes table	Yes
Data filtering	Yes
Add widgets (table, graphs, text)	Yes****
Distance measuring tool	Yes
Edit layer style	Yes
Add drawings/annotations on top of the map	No
3D visualization	No
Layer operations	
Add columns to layers	No
Merge layers	No
Join features	No
Layer intersection	No
Create polygons from points	No
Limit merging	No
Algorithms and Analysis	
Summarize center and dispersion	No
Find centroids	No
Find similar locations	No
Find outliers	No
Clustering	No
Create buffers	No
Generate tessellations	No
Calculate time or distance of path	No
Create drive-time areas	No
Others	
Jupyter notebooks integration	No
Geocoding	No
Google Places integration	No
Plan routes	No
Live tracking	No
Aggregated datasets patterns	No

Table 1. GeoNode gap analysis table.

* Formats: KML,ESRI Shapefile, GeoJSON, GeoTiff
** Formats : GeoJSON, ESRI Shapefile, PNG, PDF, Excel, CSV, GML, ZIP
*** MapStore2 is used for web mapping
**** There are only three types of graphs (bar, pie and line chart)

3.1 IT Management practices

For operations purposes, best practices in the IT services management should be used to keep the development-testing-production pipeline under control (ESRI, 2020). It is also highly recommended to use DevOps principles like version control and containerized environment for development purposes. For testing, a stagging environment should be setup in a local virtual machine and updated with the latest software modules and data to avoid any production deployment problems. Production environment should be monitored closely and any incident must be documented and reviewed periodically during the Sprint Retrospective.

3.2 Programming languages

Programming languages are also critical concern, because not all languages give the same access to the frequently used frameworks and analysis tools, particularly artificial intelligence and machine learning packages. Languages like Python, Java and JavaScript allows developers to simply access a huge amount of high-quality battletested algorithms that can make the difference from a customer point of view.

Table 2 highlights some of the geospatial open-source projects maintained by the most active communities in this domain, namely: OSGeo (osgeo.org), LocationTech (eclipse.org) and Apache Software Foundation (apache.org). All those libraries are useful for spatial analysis and consequently for Location Intelligence applications.

Python	Java	Javascript
GeoPandas	GeoGig	Leaflet
PySAL	GeoMesa	OpenLayers
GDAL/OGR	GeoPeril	Proj4js
RSGISLib	GeoTrellis	Turf.js
PyProj	GeoWave	TerriaJS
GeoNode	JTS Topology Suite	MapStore2
Orfeo ToolBox	Proj4J	CesiumJS
GEOS	RasterFrames™	GeoMoose
PyCSW	SFCurve	
PyWPS	Spatial4j	
Open Data Cube	GeoTools	
GRASS	GeoNetwork	
SAGA GIS	Deegree	
OSMnx	Open route service	
	Apache Spatial Information System (SIS)	
	Apache Sedona	
	GeoOrchestra	

Table 2. List of open-source geospatial project.

3.3 Functional requirements

The functional requirements to implement include: Ingestion, Management, Visualization and Analysis of geospatially enabled datasets both homemade and imported/reformatted from external sources. To achieve that, many software components must be configured to work together efficiently as a SaaS (Software as a Service) solution in a cloud environment (Figure 2). The components needed are: Load Balancer, Web Server, Web Mapping server (with caching functionalities), Spatial Database and as needed Applications engines (Tomcat/Jetty, Django, …). In term of Cloud component, we need: Domain Name Service, Storage, Compute instances, Containers and virtual cloud networks.

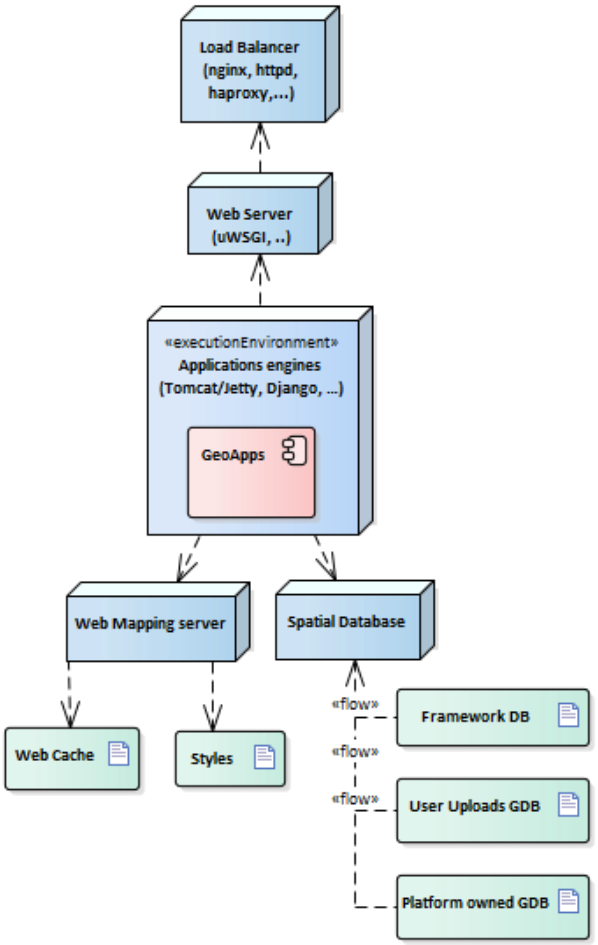

Figure 2. Location Intelligence platform components.

4. APPLICATION

4.1 Geoweba platform

Applying all the principles described before, we built and deployed on the internet an SaaS Location Intelligence platform named "Geoweba" (Geospatial Web for Algeria), based on GeoNode Project (GeoNode Project Contributors, 2021) and available at the following address: https://geoweba.transformatek.dz.

Geoweba targets entrepreneurs with limited resources and who lack market Insights. It simplifies the business analysis of the locally targeted value streams and unlike traditional consulting methods, Geoweba is built on cutting edge spatial data science technologies merging user ingested data with a large collection of Geo-Enabled datasets collected from various open and alternative sources. It includes three main components: Data Layers, Maps and GeoApps.

4.2 Dataset production process

An important component of any Location Intelligence Platform is its owned datasets (layers) that can be used directly in spatial analysis operations or combined with user ingested data by geo-enrichment. To produce those datasets of points of interests (POI), covering the Algerian territory and required for Geoweba to be fully operational, a Data Science inspired process was implemented including: Data engineering, Geocoding, Quality Check, Storage Management and Sharing at the end (Table 3).

Those Datasets, containing location data, are collected from different sources, mainly: public online sources (census, official gazette, official websites, ...), international providers (openstreetmap, UN agencies, NASA, USGS, NOAA, JAXA, ...), local partners (Fintech, VTC, Phone book, ...) and homemade by our company by direct surveying.

Process Steps
Source Selection
Identify and evaluate available sources (website, documents, OpenStreetMap, Phone books, gazetteers, ...) Validate most suitable source
Data engineering
Add a branch to the dataset repository in GitHub Collect raw data in a specific directory Explore and format data into a CSV file (Comma Separated Values) using Python and Jupyter Notebook
Geocoding
If the data has no coordinates, use a geocoder (Nominatim, or others) to locate their positions Identify the unresolved locations
Automatic check for incorrect locations using Python Jupyter Notebook
Check if locations are in their respective administrative units (provinces) Check if locations are near to buildings and roads extracted from OpenStreetMap (OSM) Compare data content with nearest OSM features tags Identify the suspicious locations and export to GeoJSON
Fix manually the incorrect locations using QGIS
For each location: Investigate the problem and search online navigation platforms (OpenStreetMap, Google Maps, HERE, TomTom) for the right address Check for duplicates Export final dataset to CSV
Upload the dataset to platform owned database
Create an SSH Tunnel to the remote database Connect with QGIS and upload the dataset Check if all geometries has been loaded correctly
Publish the dataset in the webmapping server
Create a connection to the platform owned database Define the data source properties Fill the metadata Create a dedicated style Set caching parameters Benchmark the published geo-service and fine tune its parameters if necessary
Publish the data layer in the platform portal
Update and Synchronize layers definitions Assign the corresponding owner Add Labels in the most suitable languages Create a thumbnail
Create maps to highlight layers content
Analyze existing layers and select those with interesting relationships, mainly those involved in a specific process (value stream) Create a new map with selected layers Update symbols / scales visibility if necessary Add widgets/charts (Pie, Bar, Line, …) to highlight content usefulness and spatial relationships with the area of interest

Table 3. POIs dataset production process.

4.3 GeoApps Development

Developing specific spatial analysis tools (GeoApps) for each business need is a challenging task, particularly when undertaken using frequently updated open-source components. In our case we use GeoNode Project, which is mainly based on the python Django framework for the backend and the frontend. But with the integration of MapStore2 Client in GeoNode 3.x, a big part of the front end is now based on the React JavaScript Framework with its corresponding backend API (Application Programming Interface) implemented with Django REST Framework.

Therefore, Geoweba GeoApps are developed by inheriting the GeoNode GeoApp class (Figure 3) and a specific serialization mechanism is implemented to allow REST API calls by the frontend developed using ReactJS and Material UI.

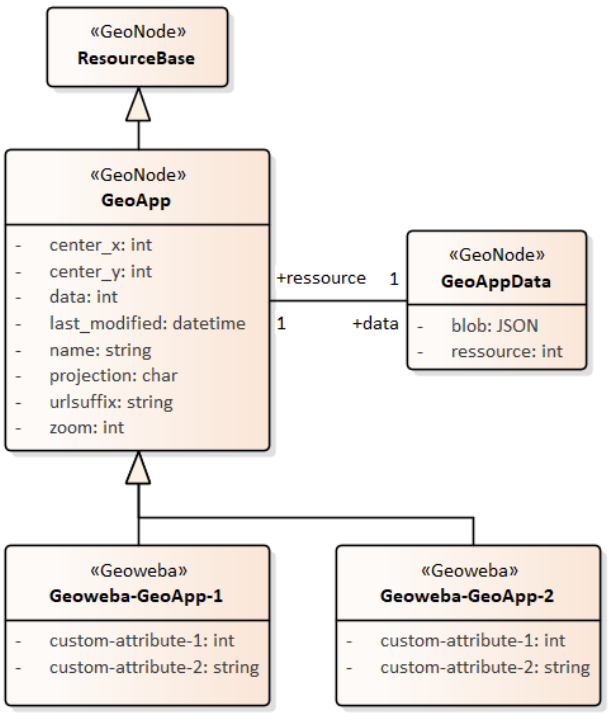

Figure 3. Class diagram for GeoApps.

The big advantage of this design is the freedom given to the GeoApp developer to store any kind of user related parameters in one JSON object (GeoAppData class), stored as a blob in the database and easily consumable by frontend JavaScript part of the application.

4.4 Source control

When working with open-source components, source code control became a huge challenge, especially with upstream active projects having new commits every day. It's become more complicated when our built components inherit their behavior from those externally managed software packages. If we don't strictly respect the above-mentioned best practice of having a stagging environment, applying untested changes can be harmful to the production environment and may cause service interruption and data loss.

To avoid that, the strict branching system shown in Figure 4 has been be put in place to keep production source code under control.

Firstly, the two (02) upstream repositories with the HEAD branches of GeoNode and Mapstore2-client has been forked to an organization owned account on Github and cloned to the local development environment with the upstream set to the GeoNode organization repositories.

Assuming that the HEAD of the forked repositories must keep in sync with the upstream, new branches has been created for our future contributions to the GeoNode project and a specific branch for our production environment with all necessary customizations and rebranding modifications has been created.

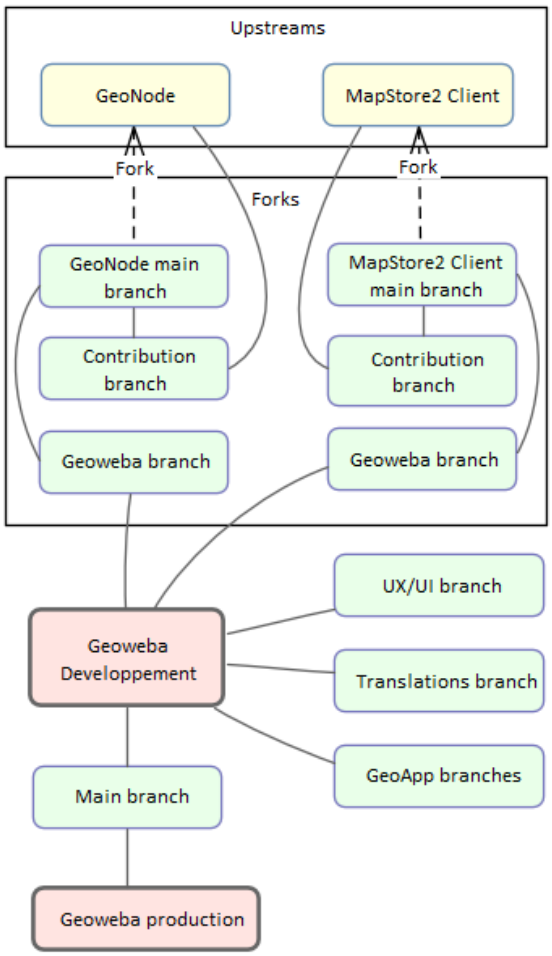

Figure 4. Source code management and control with git.

In the development environment, we can have as many branches as needed, like branches for User Interface /eXperience (UI/UX), translations, bug fixes and a branch for each GeoApp under development.

The most important branch in the development environment is of course the main (master/HEAD) branch, which has to be in sync with our production deployed binaries. Any modification in this branch must be done with a pull request which should be reviewed by as many team members as possible before its merge. Finally, a proper documentation and comments should be maintained by the team to ease the code understanding and reusability.

4.5 Containerization

To keep the development process in sync with the frequent updates in the source code it is recommended to use virtualization technologies whereby each virtual machine is providing a given service. This eliminates the need to install and administrate application dependent software packages and libraries on the physical processing nodes. It also avoids the problem of applications having conflicting library requirements. (Soille, P., Burger, A., De Marchi, D., Kempeneers, P., Rodriguez, D., Syrris, V., & Vasilev, V., 2018).

Key advantage of using GeoNode community driven project is the preconfigured docker compose containerized settings which are available in two flavors: development and production-like.

In the development setting (Figure 5), the Django debug flag is set True and the Paver Python-based build/distribution/deployment scripting tool is used to create the web development server. Hence, when building the solution, separate docker images are created for each software component of the stack, nominally: PostGIS database, Elasticsearch, RabbitMQ, Celery, Geoserver, Django and Nginx. Also, a specific network and volumes are created for intra container communications and data storage.

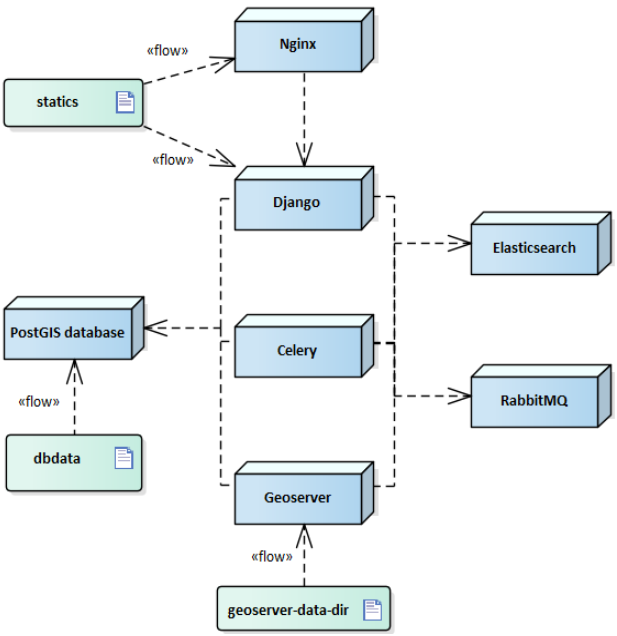

Figure 5. GeoNode Docker setting for Development.

In the production-like setting (Figure 6), the uWSGI Python web server is used instead of Paver and docker images are created for the following software components: PostGIS database, RabbitMQ, Celery, Geoserver, Django, Letsencrypt, Jenkins and Nginx.

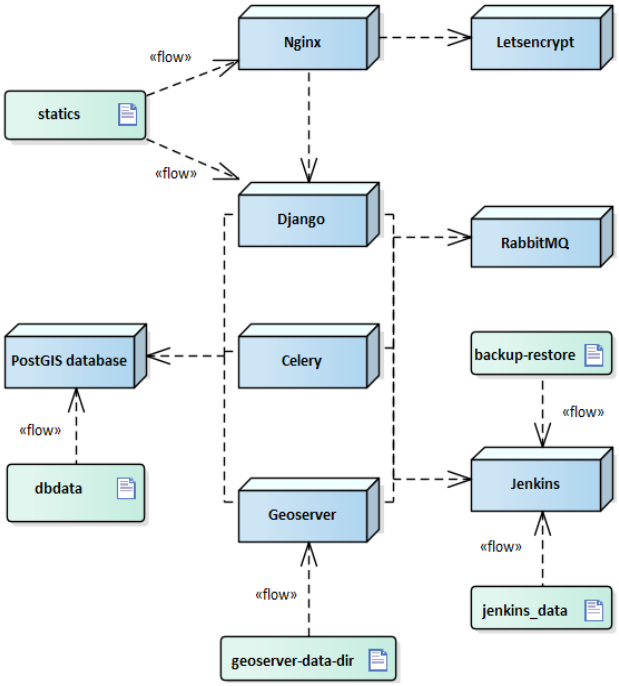

Figure 6. GeoNode Docker setting for Production.

5. RESULTS AND DISCUSSIONS

In this contribution we investigated the use of open-source components to architect a Location Intelligence platform. This approach allows us to quickly build an MVP and to bring it to the final user for feedback and continuous improvement. It has been proven that using best practices in IT service management and Agile approaches is well suited for this kind of projects.

Most important challenges of using open-source components have been highlighted and some solutions has been proposed for teams working in this kind of projects.

The final choice of GeoNode (GeoNode Contributors, 2021) as a starting point for the proposed location intelligence platform is also justified because the system complies to all non-functional and functional requirements defined herein.

The resulting Location Intelligence platform (Geoweba) is a demonstration of the suitability and relevance of the Agile/DevOps approach discussed in this contribution.

ACKNOWLEDGEMENTS

The authors thank the board of SARL TransformaTek for its continuous support and all the contributors to the GeoNode platform and GeoNode Project source code.

REFERENCES

Corti P, Bartoli F, Fabiani A, Giovando C, Kralidis AT, Tzotsos A. 2019. GeoNode: an open-source framework to build spatial data infrastructures. PeerJ Preprints 7:e27534v1 https://doi.org/10.7287/peerj.preprints.27534v1

ESRI, 2020. Architecting the ArcGIS System: Best Practices. ESRI External. December 2020.

G2, 2020: G2 Grid® Report for Location Intelligence | Winter 2021. https://www.g2.com/reports/grid-report-for-location-intelligence-winter-2021

GeoNode Contributors, 2021. Open-Source Geospatial Content Management System, Version 3.0. geonode.org (January 2021).

GeoNode Project Contributors, 2021. GeoNode template project: Generates a Django project with GeoNode support, Version 2.10. https://github.com/GeoNode/geonode-project (January 2021).

J. McCormick and E. Miller, 2019: Now Tech: Location Intelligence Technologies-Q3 2019, Forrester's Overview Of 34 Location Intelligence Technology Providers, July 18, 2019.

Soille, P., Burger, A., De Marchi, D., Kempeneers, P., Rodriguez, D., Syrris, V., & Vasilev, V. (2018). A versatile data-intensive computing platform for information retrieval from big geospatial data. Future Generation Computer Systems, 81, 30-40.

World Bank, 2020. Doing Business 2020: Comparing Business Regulation in 190 Economies. Washington, DC: World Bank. © World Bank. http://hdl.handle.net/10986/32436. License: CC BY 3.0 IGO.