INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Map Services Management

ANDRÉ PINHAL GONÇALVES
outubro de 2023

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Map Services Management

André Pinhal

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Computer Systems

Supervisor: Dr. Alexandre Bragança, Professor, DEI/ISEP

Porto, October 14, 2023

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, October 14, 2023

Abstract

About 20 years ago, Google and other companies introduced the tiled maps, and nowadays, it is possible to produce similar work using open data and open source software.

Web Map Service and Tile Map Service are a set of open standards to provide ways for users to access and visualize maps by interacting with geospacial data, over the internet. Most of the solutions to provide maps, make use of geospacial databases like PostgreSQL/PostGIS or MBTiles/PMTiles. Dedicated servers follows the standards specified by organizations such as Open Geospatial Consortium.

The main goal of this work is to create a centralized and scalable solution that publish basemaps for a predefined set of geographic regions. These basemaps are displayed as part of a desktop or mobile applications with internet access.

In order to fulfill this purpose, the best approach is, for each geographic region, to generate a MBTile database using raw data extract of the OpenStreetMap packed by Geofabrik. The raw data are also combined with a second data source, Natural Earth, to complete the map information at smaller scales. The final result goes through a process of cartographic generalization to be able to access only the relevant geospatial data at a given map scale or zoom level.

The data are published as vector tiles, using a tile server, and for legacy applications there's also the possibility to display the basemaps as raster tiles. Another available option is to use PMTiles files, which are similar to MBTiles but cloud optimized and suitable for serverless solutions.

In the interest of ensuring good performance and stability, it is possible to keep everything together behind a reverse proxy, using as an example a Nginx server. Taking advantage of HTTP range requests functionality, also available in Nginx, it is possible to consider the serverless option of PMTiles and the standard tile server under the same umbrella.

Finally, two points were considered and explored as opportunities for improvement, however not fully implemented. The first is the ability to cache vector/raster tiles requests, and the second is the ability to deploy the solution supported by a Content Delivery Network.

Keywords: DevOps, WebMaps, TileServer, VectorTiles, OpenStreetMap

Resumo

Google e outros serviços introduziram o *tiled maps* há cerca de 20 anos. Actualmente, é possível produzir trabalhos semelhantes usando dados e software de código abertos.

Web Map Service e *Tile Map Service* são um conjunto de protocolos padrão abertos que fornecem aos utilizadores uma forma de acederem e visualizarem mapas interagindo com dados geoespaciais, através da Internet. A maioria das soluções que fornecem mapas fazem uso de bases de dados geoespaciais PostgreSQL/PostGIS ou MBTiles/PMTiles. Os servidores são dedicados conforme normas padrão especificadas por instituções como a Open Geospatial Consortium.

O principal objetivo deste trabalho é criar uma solução centralizada e escalável que publique mapas de base para um conjunto predefinido de regiões geográficas. Estes mapas de base devem ser mostrados numa aplicação desktop ou mobile com acesso à internet.

De forma a atingir este propósito, a melhor abordagem é, para cada região geográfica, gerar uma base de dados MBTile, usando extratos de dados em bruto do OpenStreetMap disponibilizados pela Geofabrik. Os dados em bruto são também combinados com uma segunda fonte de dados, o Natural Earth, para completar a informação do mapa nas escalas menores. O resultado final passa por um processo de generalização cartográfica de forma a disponibilizar os dados geoespaciais relevantes para uma determinada escala ou um determinado nível de zoom do mapa.

Os dados são publicados como *vector tiles*, usando um *tile server*, e para aplicações *legacy* também existe a possibilidade de disponibilizar os mapas em formato raster. Existe uma outra opção que consiste na utilização de ficheiros PMTile, que são ficheiros similares aos MBTiles mas otimizados para a *cloud* e disponibilizados num princípio *serverless*.

De forma a garantir um bom desempenho e estabilidade, é possível agregar toda a solução atrás de uma *reverse proxy* usando por exemplo um servidor Nginx. Tirando partido da funcionalidade *HTTP range requests*, disponível também no Nginx, torna-se possível servir PMTiles (*serverless*) e *Tile servers* sob a mesma infraestrutura.

Por fim, mais dois pontos foram considerados e explorados como oportunidades de melhoria, mas não foram totalmente implementados. O primeiro é a capacidade de armazenar em cache pedidos de *Tiles* vector/raster e o segundo é a capacidade de disponibilizar a solução apoiada num *Content Delivery Network*.

Palavras-chave: DevOps, WebMaps, TileServer, VectorTiles, OpenStreetMap

Contents

List of Figures

List of Tables

List of Source Code

List of Symbols

Data Storage

kB	kilobyte
MB	megabyte
GB	gigabyte

Time

ms	millisecond

List of Acronyms

AHP	Analytical Hierarchy Process.
API	Application Programming Interface.
CDN	Content Delivery Network.
CRS	Coordinate Reference System.
DB	Data Base.
EPSG	European Petroleum Survey Group.
ESRI	Environmental Systems Research Institute.
GCS	Geographic Coordinate System.
GDAL	Geospatial Data Abstraction Library.
GIS	Geographic Information System.
GL	Graphics Library.
GNSS	Global Navigation Satellite Systems.
GPS	Global Positioning System.
HTTP	Hypertext Transfer Protocol.
IaC	Infrastructure as Code.
ISO	International Organization for Standardization.
JS	JavaScript.
JSON	JavaScript Object Notation.
MVT	Mapbox Vector Tile.
NCD	New Concept Development.
OGC	Open Geospatial Consortium.
OSGeo	Open Source Geospatial Foundation.
OSM	OpenStreetMap.
PBF	Protocolbuffer Binary Format.
PCS	Projected Coordinate System.
PNG	Portable Network Graphics.
REST	REpresentational State Transfer.
SRS	Spatial Reference System.

SSD	System Sequence Diagram.
TMS	Tile Map Services.
UML	Unified Modeling Language.
URI	Uniform Resource Identifier.
URL	Uniform Resource Locator.
VGI	Volunteered Geographical Information.
W3C	World Wide Web Consortium.
WMS	Web Map Service.
WMTS	Web Map Tile Service.
XML	eXtensible Markup Language.

Chapter 1

Introduction

This chapter aims to provide an overall context about the work developed within the scope of this dissertation with the support of DevScope company. To this end, this document pretends to explain briefly the problem under study, the main motivation, objectives and contributions. The chapter ends by explaining the structure of the document.

1.1 Context

DevScope provides a set of solutions where it regularly has to provide basemaps. For each client it was decided to develop its own mechanism to access them. In order to make the maps available to the client, the company tested two approaches:

1. The first approach consisted of using `openstreetmap-tile-server` Overvoorde (2023), to serve raster tiles, based on a PostgreSQL/PostGIS database. It revealed low performance due to the process of converting the raw OpenStreetMap (OSM) into a PostgreSQL/PostGIS database, so this approach was discarded.
2. The second approach, is based in the TileServer GL (MapTiler 2023a) to publish raster/vector tiles from MBTiles database. The MBTile file is a SQLite and was built using the Tilemaker (Fairhurst 2023). The process is straightforward, using a OSM raw/base data files in the Protocolbuffer Binary Format (PBF) as input.

Currently the second approach is the solution used in production, and have the following deployment steps:

1. Most of the geospatial base data source comes from OSM provided by Geofabrik and maintained as `.osm.pbf` files.
2. Post-processed with Tilemaker into a ready-to-use multi zoom/scale MBTile file.
3. Served with TileServer GL as raster tiles in Portable Network Graphics (PNG) format.

1.2 Problem

Besides the current approach requiring some manual work, the way maps are provided to client is bringing some risks such as:

- Use maps not licensed for this purpose.
- Use of non-scalable resources.
- The inability to manage the access to the maps.

Regarding scalable resources, some products made available to clients are spread across several countries, and there is a need to quickly and without any inconvenience, instantiate new map regions for one or several products.

1.3 Motivation/Objectives

Considering the problems presented, with this project it is intended to:

- Create a centralized solution that publish maps using standardized formats for a predefined set of geographic regions.
- The proposed solution should make use of only one map provider that allows this type of use and can run on-premise or in the cloud.
- Solution to be portable and scalable based on infrastructure as code approach, making use of technologies like containers.

1.4 Contribution

As mentioned previously, the main goal is to implement a map service that is easy to maintain.

The first important step is to correctly use open source geospatial data in accordance with the OSM license. By properly using open data, it is expected to reduce the medium and long-term cost of dependence on third-party services.

Exploring different post-processing tools and geospatial data services is a relevant contribution to this work.

Additionally, empower company with skills in providing map-based services and understanding geospatial data types and sources.

1.5 Document Structure

Chapter 2. State of the Art presents the literature on important concepts for understanding the study and work developed, as well as the technologies, tools and data sources consulted and used to implement it.

Chapter 3. Value Analysis describes and evaluate the opportunities, and the value of the solution proposed using structured group decision technique.

Chapter 4. Design shows the architectural concepts as well as the functional and non-functional requirements of the developed solution.

Chapter 5. Implementation details the development and implementation process from study to solution.

Chapter 6. Evaluation formulates the hypothesis, the indicators, the sources of information and specify the evaluation methodology employed.

Chapter 7.Conclusions final remarks.

Chapter 2

State of the Art

This Chapter begins by presenting the concepts necessary to understand the work developed, and then talks about the technologies used during implementation.

As main references for the geospatial concepts the document follows these sources:

- A free book in Spanish about Geographic Information System (GIS) called *Sistemas de Información Geográfica* by Víctor Olaya (Olaya 2020).
- The article *Best Practices for Publishing, Retrieving, and Using Spatial Data on the Web* (Brink et al. 2019), and its project website, a collaboration between World Wide Web Consortium (W3C) and Open Geospatial Consortium (OGC) (W3C 2016).

Most of the figures used are print screens other than, diagrams or illustrations non produced by the author have references/credits to the sources.

2.1 Best Practices for Publishing Spatial Data on the Web

W3C/OGC Working Group on Spatial Data on the Web identifies 14 best practices for publishing spatial data on the Web (Brink et al. 2019).

Some examples are transcribed below:

Best Practice 1: Use globally unique persistent HTTP URIs for Spatial Things

Best Practice 5: Provide geometries on the Web in a usable way

Best Practice 6: Provide geometries at the right level of accuracy, precision, and size

Best Practice 7: Choose coordinate reference systems to suit your user's applications

Best Practice 14: Support requesting and returning geometries in a specific CRS

Best Practice 15: Include spatial metadata in dataset metadata

The web resources are available using Hypertext Transfer Protocol (HTTP) and identified by a Uniform Resource Identifier (URI) using a specific Coordinate Reference System (CRS) or Spatial Reference System (SRS). The endpoints are to be exposed as RESTful Web services (W3C 2016).

2.2 Web Maps Services

Web Map Service (WMS) is the oldest and most popular standard for web mapping (Doyle 1997). Generically speaking, a WMS is protocol developed by OGC to serve geospatial data over the internet using HTTP. It allows users to access and visualize geospatial data from a variety of sources, such as maps, satellite images, and geospatial databases, through web-based applications (La Beaujardiere 2006).

2.2.1 Web Maps Services and Tile Map Services

WMS is a protocol but also a generic terminology to serve maps on the web.

Using WMS as a map service is not the best approach when quick response times are crucial, because it relies on on-the-fly map rendering responses (La Beaujardiere 2006). The solution is to use pre-rendered data structured as map tiles. There are two specification approaches, the Web Map Tile Service (WMTS) from OGC (Joan Masó and Julià 2010) and Tile Map Services (TMS) from Open Source Geospatial Foundation (OSGeo) (OSGeo 2006).

From the specification(OSGeo 2006), it attempts to fulfill REpresentational State Transfer (REST) principles and also supports alternative SRS aligned with the best practices described in Section 2.1.

The TMS standard is older and simpler than WMTS and is the most commonly used specification on the web for providing map data in a tiled form. Google Maps was one of the first to publish TMS as a ZXY scheme (MapTiler 2023b). ZXY is the most popular, there are no advantages over TMS, the difference between the two is the **y** coordinate, it is inverted (MacWright 2013).

Most of the tiled map servers implement the ZXY scheme, but for the sake of simplicity, the terms TMS/ZXY are used interchangeably and will be further explored in Section 2.4.

2.3 Geographical Information Systems

Today, the vast majority of the information used is georeferenced. All data from geospatial servers are processed using GIS software.

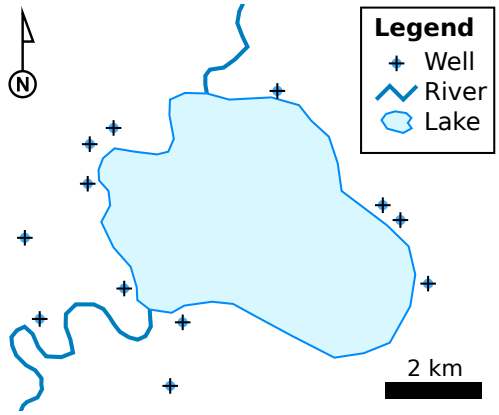
A standard GIS system enables the following operations (Olaya 2020):

- Read, edit, store and, generally speaking, manage spatial data.
- Analyze those data. This includes everything from simple queries to complex models, which can be performed using the spatial component of the data (the location of each value or element), the thematic component of the data (the value or element itself), or both.
- Generate documents such as maps, reports, plots, etc.

2.3.1 Types of GIS data

A geospatial feature is an abstraction and representation of a real-world entity (W3C 2016). Using geometric primitives such as, points, lines and polygons it is possible to represent most of the world's entities on a map. In the Figure 2.1 is an illustration of a map that uses

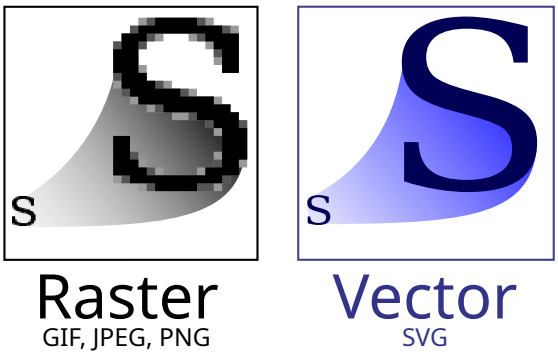
vector data as a representation model, where the points are the wells, the line is the river a the polygon is the lake.



Source: Simple_vector_map.svg (Wikimedia 2007)

Figure 2.1: Illustration of a simple vector map represented by points, lines and polygons.

Other representation model is the raster data, such as a satellite image map. The following Figure 2.2, illustrates raster and vector model concepts using image file formats. A raster image is composed by a grid of values called pixels.



Source: Bitmap_VS_SVG.svg (Wikimedia 2006)

Figure 2.2: Representation of a raster image based on pixels, on the left side, and vector base image, on the right side.

Raster and vector models are usually displayed together by overlapping vector data over raster data, where each level of data is called a layer (Olaya 2020).

2.3.2 Desktop GIS Software

ArcGIS and QGIS are the most used desktop GIS software, and both are used for the same purpose, which is visualize and analyze geospatial data.

ArcGIS

ArcGIS is commercial software from the Environmental Systems Research Institute (ESRI) company. It is mainly used by professionals and is responsible for developing the beginnings of GIS.

QGIS

Initially released in 2002, the QGIS is open-source and more accessible to a wide range of users, including students, academics and non-profit organizations. QGIS opens all OGC standards through the use of Geospatial Data Abstraction Library (GDAL).

2.3.3 Desktop GIS file formats

ESRI Shapefile

(`.shp`, `.shx`, `.dbf`)

ESRI Shapefile is vector file format and the most popular GIS format of all time. Developed by ESRI, the format was first introduced as part of ESRI's ArcView GIS software in the early 1990s.

According to an article by Shapefile format creator, Jeff Jenness, the format was created in response to the need for a standardized file format for exchanging spatial data between different GIS software packages. At the time, there were a variety of different file formats being used, but no single format that could be easily used across different platforms (Esri 1998).

There are some alternatives to Shapefile and there is a growing movement to replace it permanently, but it is still an ongoing process. For example, a best practices site: switchfromshapefile.org with reference to better alternatives.

GeoPackage

(`.gpkg`)

Such an alternative is the GeoPackage, an OGC initiative started in 2012, an open standard to replace the limited Shapefile format.

GeoPackage is designed to store all types of geospatial data, including multi-layer vector and raster data, in a single SQLite database container that can be easily transferred and accessed by different GIS software.

GeoJSON

(`.json`)

GeoJSON is based on JavaScript Object Notation (JSON), capable of representing geographical features such as points, lines and polygons, along with their properties, generally used for sharing geospatial data on the web.

Other GIS formats relevant for map server applications are introduced in Section 2.6.

2.3.4 GIS data sources

The main data sources that provide data for GIS (Olaya 2020):

Remote sensing and Photogrammetry : raster data based mainly on aerial platforms such as satellites aeroplanes and drone.

Digitization of printed cartography or already digital maps is a source of vector data.

Global Positioning System (GPS) or Global Navigation Satellite Systems (GNSS) allows the user to know the exact location of its position with an accuracy of a few meters. Using professional GNSS receivers it is possible to achieve a centimeter precision. This positional data is a source of vector data.

Voluntary Geographical Information or Volunteered Geographical Information (VGI), being OSM the most relevant VGI project at the moment, a *collaborative project to create a free editable map of the world*.

2.3.5 Spatial Reference System

All SRS/CRS have a code to uniquely identify a set of spatial coordinates. These code are maintained by European Petroleum Survey Group (EPSG). The EPSG database is freely available for public use and its adoption has been endorsed by many standards organizations, including International Organization for Standardization (ISO) and OGC.

Using EPSG codes allows GIS software to identify the correct SRS and provide the appropriate coordinate transformations to overlay and analyze different data sets with different SRS.

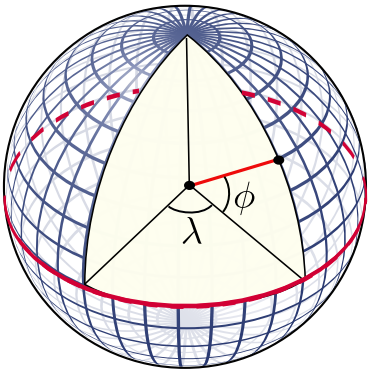
EPSG codes can identify different SRS, such as WGS84 (World Geodetic System 1984) or ETRS89 (European Terrestrial Reference System 1989), which are commonly used in geospatial data.

There are two important sources of CRS in GIS applications:

- Projected Coordinate System (PCS)
- Geographic/Geographic Coordinate System (GCS)

A PCS defines a map projection that converts geographic coordinates, latitude and longitude, to projected coordinates, x and y, in a specific unit of measurement, such as meters. A SRS based on the Universal Transverse Mercator (UTM) system is an example of a PCS.

A GCS is a coordinate system that uses a spherical or ellipsoidal model of the Earth's surface using latitude and longitude coordinates to define its locations. The Figure 2.3 shows the definition of latitude and longitude on a sphere.



Source: Latitude_and_longitude_graticule_on_a_sphere.svg (Wikimedia 2010)

Figure 2.3: Definition of latitude () and longitude () on a sphere

Unlike a PCS, a GCS does not flatten the Earth's surface and maintains the curvature of the surface. Usually PCS is used for local mapping, while GCS is used for global mapping.

WGS84 - EPSG:4326 - GCS

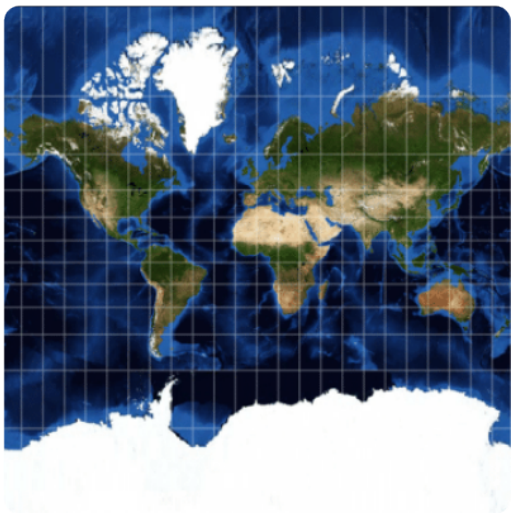
WGS84 is commonly used as the standard SRS for most of the geospatial data. The WGS84 uses an ellipsoid with a same name. This ellipsoid is a mathematical model of the shape of the Earth that is commonly used as the reference frame for the GPS in Figure 2.4.

Source: earth.png (MapTiler 2023b)

Figure 2.4: **Degrees** Geodetic coordinates WGS84 (EPSG:4326).

Web/Spherical Mercator - EPSG:3857 - PCS

The Web Mercator projection, is a SRS that is commonly used for web mapping applications, including Google Maps and OpenStreetMap, MapTiler or Mapbox. This CRS is based on a spherical model of the earth and uses a Mercator projection to represent the earth's surface on a 2D plane, like in figure 2.5. This projection is useful for displaying maps on a flat screen or a paper despite the big distortions in the polar regions. The coordinates origin (0,0) are at the intersection of the equator and the prime meridian.



Source: world.png (MapTiler 2023b)

Figure 2.5: **Meters** Projected coordinates Spherical Mercator (EPSG:3857).

The following equations show the simplification of the Mercator projection formula using a sphere as a reference, which is more computationally efficient.

$R = 6378137$ meters

$$x = R\,(\lambda - \lambda_0)$$

$$y = R \ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\left(\frac{1 - e\sin\varphi}{1 + e\sin\varphi}\right)^{\frac{e}{2}}\right] \quad \text{> e = 0 >} \quad y = R \ln\left[\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right]$$

Where x only depends on longitude input and y on latitude input: (`EPSG:4326 > Spherical Mercator Projection > EPSG:3857`).

eq. (2.1)

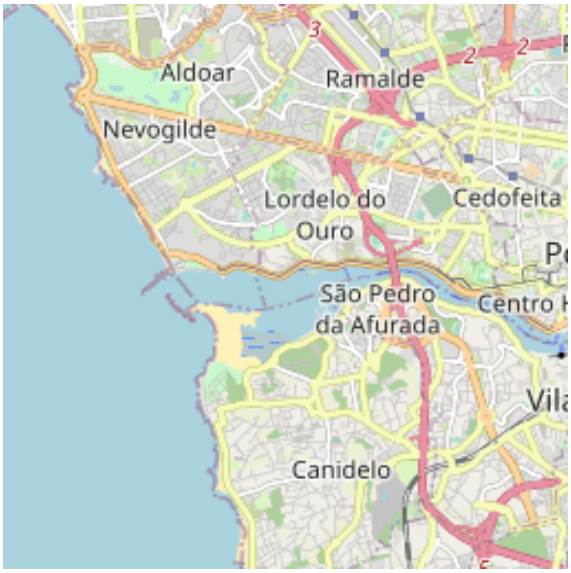
For the purpose of this project there are only two SRS to use as reference:

- WGS84 - EPSG:4326 - GCS
- Web Mercator - EPSG:3857 - PCS

Most of the original data like OSM data sources are in SRS WGS84 - EPSG:4326 and the data to published as web map is in the SRS Web Mercator - EPSG:3857.

2.4 Tiled Web Maps

The following image, Figure 2.6 shows the URI to a raster tlie (1533.png) at zoom level 12. This TMS follows ZXY scheme.



Source: tile.openstreetmap.org/12/1949/1533.png

Figure 2.6: Examples raster tile (1533.png) with a size of 256x256 pixels.

2.4.1 Map Tiles and Geospatial Datasets with Overviews/Pyramids

Geospatial tiles and overviews/pyramids are two techniques used to optimize the rendering and display of large-scale geospatial data.

Geospatial tiles are small, square images of a map or other geospatial data that are pre-rendered at different zoom levels and stored in database format to be served. This approach allows for fast and efficient rendering of large scale data, as only the tiles needed for the current view are requested and displayed. This is commonly used in web mapping applications, such as Google Maps or OpenStreetMap (MapTiler 2023b).

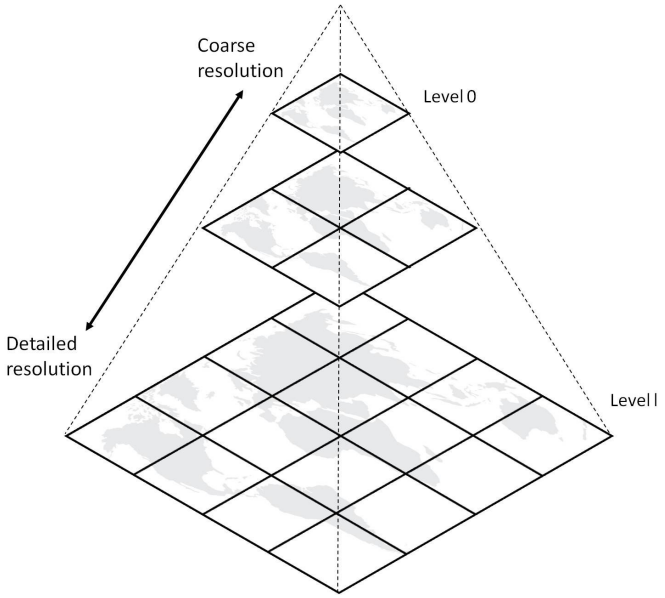
2.4.2 Web Mercator based Map Tiles

Web Mercator based tiles are the best choice for a global web mapping applications because they are optimized for fast and efficient rendering due to simplified mathematical projection formulas, equations 2.1 from Section 2.3.5. Using the geospatial data in Web Mercator (EPSG:3857) SRS as input source for a TMS server is the bast practice for an optimized online map service provider(Stefanakis 2017).

2.4.3 Introduction to the concept of Raster and Vector Tiles

To obtain a multiscale/zoom web map, the geospatial data goes through a cartographic generalization process (further explaned at Section 5.3.1). The result is a new set of multilayer or multilevel datastes ate gradual resolutions in a pyramidal system.

Starting with one tile on the top of the pyramid, then 4 tiles, 16 tiles, 64 tiles, etc, as shown in Figure 2.7 .

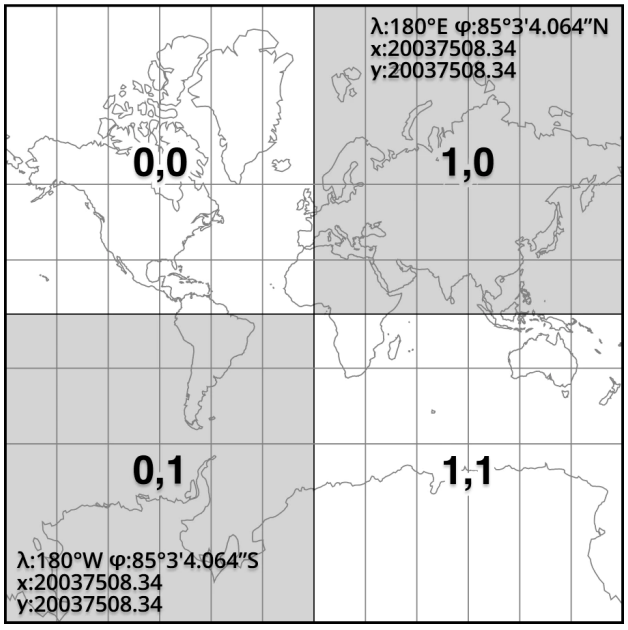


Source: tilePyramid.jpg (MapTiler 2023b)

Figure 2.7: Pyramid system with the representation of the zoom level 0, 1 and 2.

Typically, raster tiles are the size of 256x256 or 512x512 pixels depending on the screen resolution of the client device screen resolution. The use of vector tiles is independent because the map rendering happens client-side.

The number of tiles grows at a exponential rate by 4^z, where z is the zoom level. The following Figure 2.8 show the tiles number identification for the zoom levels 0, 1.



Source: mercator-epsg-img.png (MapTiler 2023b)

Figure 2.8: Pyramid tiling illustration at zoom levels 0, 1 with the tiles number identification.

" Only the relevant tiles are loaded and displayed for the area of interest/viewport." (MapTiler 2023b)

2.5 Open GIS Data Sources and Base Map Providers

MapTiler is a cloud-based mapping platform that offers tools for creating custom maps and visualizations based on OpenStreetMap data. It offers a range of features, including data hosting, styling, and custom map generation.

Mapbox is another cloud-based mapping platform that offers a suite of tools for creating and publishing custom maps. It provides a range of services, including mapping, geocoding, routing, and data analysis. Mapbox offers both free and paid plans, depending on the level of functionality required (Mallon 2015).

Maxar and Airbus are both companies that provide satellite imagery and geospatial solutions for a range of applications, including mapping, disaster response, and national security. They use high-resolution imagery and advanced processing techniques to provide accurate and up-to-date geographic data.

2.5.1 Open GIS Data Sources

OSM, is a collaborative VGI and open-source mapping platform that allows users to contribute and edit geographic data. It provides free and open data that can be used by anyone for any purpose, making it an important resource for a wide range of applications, including GIS, cartography, and navigation.

> The OSM project is a knowledge collective that provides user-generated street maps. OSM follows the peer production model that created Wikipedia; its aim is to create a set of map data that's free to use, editable, and licensed under new copyright schemes. A considerable number of contributors edit the world map collaboratively using the OSM technical infrastructure, and a core group, estimated at approximately 40 volunteers, dedicate their time to creating and improving OSM's infrastructure, including maintaining the server, writing the core software that handles the transactions with the server, and creating cartographical outputs. There's also a growing community of software developers who develop software tools to make OSM data available for further use across different application domains, software platforms, and hardware devices. The OSM project's hub is the main OSM Web site.

(Haklay and Weber 2008).

OSM Copyright and License

Transcribing OSM guidelines:

> OpenStreetMap is open data, licensed under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF).
>
> You are free to copy, distribute, transmit and adapt our data, as long as you credit OpenStreetMap and its contributors. If you alter or build upon our data, you may distribute the result only under the same licence. The full legal code explains your rights and responsibilities.

Our documentation is licensed under the Creative Commons Attribution-ShareAlike 2.0 license (CC BY-SA 2.0).

How to credit OpenStreetMap

Where you use OpenStreetMap data, you are required to do the following two things:

- Provide credit to OpenStreetMap by displaying our copyright notice 2.9.
- Make clear that the data is available under the Open Database License.

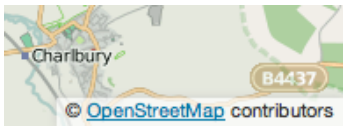


Figure 2.9: Ilustration of displaying OSM copyright notice.

(OpenStreetMap 2023a)

2.6 Geospatial Databases and Tile Map Services/Servers

With GIS data from all over the world, a desktop file like a Shapefile is not the right format for such dynamic information. Geospatial databases and map servers are important components of GIS system that enable users to store, manage, and access a big amount of geospatial data globaly. These databases include features such as spatial indexing and querying, which enable efficient and accurate processing of geospatial data. An example of a popular geospatial database is the PostGIS.

2.6.1 Geospatial Databases

PostgreSQL/PostGIS

PostgreSQL is an open source Relational Database Management System (RDBMS) that was originally developed at the University of California, Berkeley in the 1980s and it was first released in 1989 (Stonebraker and Rowe 1986).

PostGIS

PostGIS is also open source, and is and extension for the PostgreSQL that enables it to store and manage geospatial data.

The project was developed as part of the larger open source geospatial software community, which included projects like GeoServer and OpenLayers.

Since its initial release, PostGIS has become one of the most widely used geospatial database systems in the world. It is supported by a large and active community of developers and users.

SQLite

PostgreSQL and SQLite are both popular relational database. One of the main differences between PostgreSQL and SQLite is their approach to concurrency and locking. The PostgreSQL uses a multi version concurrency control system that allows multiple transactions to read and write to the same database at the same time, while still maintaining transactional integrity. SQLite, on the other hand, uses a locking mechanism that only allows one.

Mapbox Vector Tile (MVT) shared as PBF stores as MBTiles using SQLite Data Base (DB).

MBTiles

As previous stated, the SQLite is the database is used to store geospatial data in a MBTiles data format.

MBTiles can also be stored and served on the fly from a PostgreSQL/PostGIS database using the ST_TileEnvelope and ST_AsMVT functions that are included in PostGIS. These functions allow users to generate vector tiles from their geospatial data and serve them through a tile server like the GeoServer.

More informatio about the format and the way it is explored with OSM data in the next Section 2.6.2.

PMTiles

Are a Cloud-optimized version of a MBTile, more avalailabe in Section 5.2.1.

PostGIS vs MBTiles

Choosing Between PostgreSQL/PostGIS and MBTiles will depends on the specific needs of and application. If it is required advanced spatial analysis capabilities, and the need to manage large amounts of geospatial data, or required integration with other database systems, PostgreSQL/PostGIS may be the better choice. If the need is need a lightweight and efficient way to store and serve geospatial data for online mapping, MBTiles may be a better option.

2.6.2 Geospatial formats for retrieving OSM data

OSM data is stored in a structured format that can be accessed and processed using a variety of software tools and formats (OpenStreetMap 2023b).

OSM XML

The primary format for OSM data is the OSM eXtensible Markup Language (XML) format, which is a text-based format that uses a simple XML structure to represent the data. The OSM XML format includes tags for nodes, ways, and relations, which represent geographic features such as points, lines, and polygons. The data includes information about the geographic location of thefeature, as well as additional attributes such as name, type, and other metadata.

PBF - Protocolbuffer Binary Format

PBF is preferred over OSM XML and is the format used in the project.

PBF stands for Protocolbuffer Binary Format, which is a binary format for storing and exchanging geospatial data, including OpenStreetMap (OSM) data. PBF is an alternative to the OSM XML format, which is a text-based format.

PBF was developed by Google as a more efficient alternative to traditional XML based formats. PBF is a compact and fast binary format that uses Google's Protocol Buffers technology to serialize data. This results in smaller file sizes and faster processing times compared to XML based formats.

GitHub -protocolbuffers/protobuf

MBTiles

MBTiles is a format for storing geospatial data in a SQLite database, which can be used for offline map viewing and storage. MBTiles was developed by MapBox and is widely used in mobile and web mapping applications.

MBTiles can store geospatial data in a variety of formats, including PBF. When PBF is used with MBTiles, the resulting file is called an MBTiles PBF file, in his case, file contains a more compressed and optimized geospatial data (Netek et al. 2020).

2.6.3 Downloading and using OSM Data

There are many ways to use OpenStreetMap data. It is possible to download raw data `.osm.pbf` for a certain area, entire countries or regions, or certain features such as roads or buildings. There are also many ways to use use existing sets or of OpenStreetMap data, such as map images, routing software and more.

The easiest way to download raw data from OSM is through Geofabrik, this topic is developed/explained in the implementation Section 5.1.

2.6.4 TMS server options

In few words, the purpose of map servers is to provide maps over the Internet (web and mobile applications). TileServer GL and Martin Tile Server are open-source servers both used in this project.

TileServer GL

TileServer GL is designed to serve vector tiles and is able to render into raster tiles with MapLibre GL Native engine on the server side (MapTiler 2022).

Martin

Martin is a tile server that is able to dynamically combine multiple tile sources into one. It can generate and serve vector tiles on the fly from large PostGIS databases or SQLite databases (MBTiles/PMTiles files). Another important feature is that it is optimized for speed and heavy traffic. To use it in production environment, is recommended to launched it behind Nginx or Apache servers (MapTiler 2022).

2.7 Infrastructure as Code

Infrastructure as Code (IaC) is a practice in which infrastructure is managed and provisioned using code and automation tools. This approach provides numerous benefits, such as repeatability, consistency, and version control, and it has gained popularity in the DevOps and cloud computing communities.

2.7.1 Docker

While not traditionally considered part of IaC, Docker plays a significant role in modern infrastructure and application deployment. It complements IaC by providing a standardized way to package applications and their dependencies into lightweight, portable containers. These containers can be easily deployed, managed, and scaled across different environments, making it a valuable tool in the IaC ecosystem (Miell and Sayers 2019).

It fits into Infrastructure as Code in several way:

Containerization: containerize applications and services along with their dependencies. These containers encapsulate everything an application needs to run, from code and libraries to configurations. This makes your applications highly portable and consistent across different environments, a core principle of IaC.

Version Control: Docker containers and their associated images can be version-controlled. Container images (like Docker Hub or your private registry) can be stored and manage different versions for applications, ensuring reproducibility.

Orchestration: Docker Compose and Kubernetes enable to define and manage complex multi-container applications and their interactions. It allows to specify the network, storage, and scaling aspects of applications, providing infrastructure-like automation for containerized services.

2.8 HTTP Server

HTTP servers are fundamental components of the World Wide Web, allowing users to access websites and web applications. In simple words, an HTTP server, or web server, is a service that receives and responds to Hypertext Transfer Protocol (HTTP) requests from clients, typically web browsers, and serves web content.

2.8.1 Nginx

Nginx is a powerful and popular web server software that is often used as a reverse proxy and supports HTTP range requests. It's known for its efficiency, speed, and scalability, making it a popular choice for serving web content and handling various web-related tasks. Nginx is often used to deliver web pages and serve web applications, and it can also function as a reverse proxy, load balancer, and HTTP cache (Nginx 2016).

One of its main characteristics is high performance due to the handling of HTTP range requests. It is designed to handle a large number of simultaneous connections efficiently. It uses an event-driven asynchronous architecture to optimize resource utilization.

It has gained popularity in the web hosting and DevOps communities due to its speed and versatility. It's available under an open-source license and has both community and commercial support options.

Reverse Proxy

A reverse proxy is a server or software component that sits between client devices and a web server. It acts as an intermediary, receiving client requests and forwarding them to the appropriate backend server or application (Nginx 2023).

Reverse proxies have various functions, including:

- **Load Balancing:** a reverse proxy can distribute client requests across multiple backend servers to balance the load, ensuring optimal resource utilization and improving the system's availability.
- **Security:** providing security features such as firewall protection, web application firewall (WAF) services to protect backend servers from malicious traffic.
- **Caching:** cache static content, reducing the load on backend servers and improving the response time for frequently accessed resources.
- **URL Rewriting and Redirection:** rewriting URLs or perform URL redirection to ensure that clients are directed to the correct resources on the backend servers.

HTTP range requests

HTTP range requests allow clients to request a specific range of data from a resource hosted on a web server. This feature is especially useful for large files, streaming, or resuming interrupted downloads (Nginx 2016).

2.9 Content Delivery Network

A Content Delivery Network (CDN) is a system of geographically dispersed servers and data centers created to better effectively transmit web contents, including text, images, and videos, to users. Websites and web applications can be made faster, more accessible, and more effective with the help of a CDN, which also lighten the burden on the origin server (Cloudflare 2023).

Websites like video streaming platforms or frequently employ the use of a CDN to guarantee that their content is delivered effectively and consistently to users all over the world.

Tile map services are also a platforms that greatly improves their responsiveness when deployed with a CDN.

2.10 Web Map Clients

In this work it is important to render vector tiles, at least for testing results, and this is done using the client side as opposed to raster tiles that are rendered on the server side.

At the moment the Leaflet, MapLibre GL, OpenLayers are the most popular Graphics Library (GL) libraries for displaying interactive maps on a web page.

2.10.1 MapLibre GL

MapLibre GL JS is a TypeScript open source library that renders interactive maps from vector tiles in a browser and it is part of the MapLibre ecosystem. It is a fork of Mapbox GL that became closed source recently.

Web Map Styles

Is a very important topic for results testing, that follows the Mapbox Style Specification using a JSON file `Style.json`. This tematic is developed/explained in the implementation Section 5.3.1.

TileJSON

" TileJSON is a JSON format used for describing tilesets. It keeps track of where to request the tileset, the name of the tileset, and any attribution that's necessary when using the tileset. " (Mapbox 2023)

Chapter 3

Value Analysis

In this chapter it is pretended to understand what value analysis means and describe the concepts related with New Concept Development (NCD) and Analytical Hierarchy Process (AHP). Following these two models, it will allow to verify if the opportunity denoted by the problem is valid and assess if the selected solution is the the most adequate to solve the problem.

Starting with the definition of **value analysis**, it is an approach that gathers a set of techniques, knowledge and skills, in order to improve the value of an product thought the elimination of unnecessary costs or by the development and enhancement of its functions without compromise its quality, performance and reliability. It could be achieved by using a combination of creative and analytical techniques to identify alternative ways to accomplish the purpose of unnecessary costs reduction (Zeidan and Murtaza 1999).

The combination of value analysis and **analytic hierarchy process techniques** simplifies the cost assessment of a product, service or process and significantly enriches the process of identification, measurement, analysis, interpretation and diagnosis of the accountability and value of complex manufacturing and engineering systems. The information provided by these tools combined will be a great help with planning, controlling and decision-making activities, while provides clear insights on value and performance improvement (Zeidan and Murtaza 1999).

3.1 New Concept Development

Fuzzy Front End (FFE) and New Product Development (NPD) are both important stages in the development of new products.

The FFE is the initial stage of NPD and refers to the period of time when a company explores new product opportunities and makes decisions about which ideas to pursue. This stage involves market research, brainstorming, and idea generation (Koen et al. 2002).

The **New Concept Development (NCD)** is the complete process of bringing a new product from ideation to launch. It is used to evaluate and select the most promising concepts which includes prototyping, concept testing and market research activities (Koen et al. 2002).

During the NCD process, several tools could be used to support decision-making and product development, ensuring that the final product meets customer needs and expectations. Between then, Analytical Hierarchy Process (AHP) method, which is used to evaluate and prioritize the concepts based on specific criteria, such as customer needs, market potential,

and feasibility. It is a structured method that uses mathematical models and a hierarchy of criteria to evaluate alternatives and make decisions (Vaidya and Kumar 2006).

The selected concept is then developed into a final product through the NPD process. This stage includes product design, testing, and manufacturing.

Finally the product is launched and brought to market. This stage includes marketing, sales, and distribution efforts.

The New Concept Development (NCD) Model is a five-step process. The activities involved in each step of the NCD Model are as follows:

3.1.1 Opportunity Identification

Opportunity identification goal is to identify a problem or need that can be addressed through the development of a new concept (Mital et al. 2014).

During this stage, the focus is on:

- Market research: This involves gathering information about the target market, such as customer needs, trends, and preferences. This information can be used to identify new opportunities for products or services that meet unmet customer needs.
- Ideation and brainstorming: This involves bringing together a team of individuals from different departments and encouraging them to generate new ideas and concepts.
- Trend analysis: This involves identifying and tracking trends in the market, such as technological advancements, changes in consumer behavior, and shifts in regulations. This information can be used to identify new opportunities for products or services that align with these trends.
- Competitor analysis: This involves examining the products, services, and strategies of competitors in the market. This can help to identify new opportunities for products or services that differentiate from what is already available in the market.

The output of this stage is a list of potential opportunities that can be evaluated and screened in the next stage of the NCD Model (Mital et al. 2014).

Taking into consideration the scope of **this project**, it was **identified the** following **opportunities**:

- There is a growing number of company customers who needs of location-based services.
- Monitor the market practices in the availability of geospatial information on web.

3.1.2 Opportunity Analysis

This step consists on evaluating the potential of new concepts identified previously in the opportunity identification. The goal of opportunity analysis is to determine which concepts have the greatest potential for success in the market and should be further developed.

Referring to the theme of this project, it was **analysed** the following **opportunities**:

- Given the customer needs, the company is taking advantage of the moment to gain knowledge about market practices and offerings in the availability of geospatial information on web.
- The use of open source geospatial data is expected to reduce the medium and long term cost of dependence on third party services.
- Empowering the company's employees with skills in providing map-based services.

3.1.3 Idea Generation and Enrichment

It involves the generation and refining of new product and service concepts. The goal of idea generation and enrichment is to turn raw ideas into well-defined and valuable concepts that have the potential to meet the needs of the target market.

During the process of **idea generation** of this project, it came up with possible ideas:

- Centralize the mapping service instead of having one option per customer.
- Development of the map service based on APIs with the potential to become the market reference.

3.2 Analytical Hierarchy Process

Analytical Hierarchy Process, also known as AHP, is a structured group decision technique for organizing and analyzing complex decisions. This method was developed by Thomas L. Saaty in the 1970s and has been refined since then. AHP provides a rational framework for a decision that has to be made by quantifying its criteria and alternative options and connecting those elements with the overall purpose.

AHP method follows a simple methodology of data collection that is founded by the following phases:

1. **Factors and criteria:** after the breaking down in hierarchical smaller problems, factors and criteria that will be used to judge and decide must be defined. During this phase, criteria are quantified, in order to judge the alternative options and to relate the factors to the overall goal.
2. **Decision making process:** AHP uses a pairwise comparison to create a matrix and stablish priority. This quantifying capability is the key factor that distinguishes AHP from other decision-making methodologies.
3. **Alternatives:** on this last phase, AHP defines that numerical priorities are calculated for each of the alternative options. The numbers represent the most desired solutions, based on all users values.

Applying the AHP method to this work, first of all, the hierarchical decision tree is drawn by representing the problem, the criteria and the alternatives, as it is shown in figure 3.1.

The problem to solve, represented in first level, is **what is the best solution for serving and manage multiple maps over the internet**. In the second level are identified the **criteria** for evaluating the solutions, which are:

- **Performance**: web server performance refers to how efficiently and effectively it can respond to incoming requests. A fast response time is important for providing a good user experience.
- **Reliability**: reliability refers to the ability to consistently and continuously serve web pages and respond to client requests without any disruptions or downtime. It is up and running and able to serve pages for a large portion of the time.
- **Cost**: costs are mainly related with level of resources needed.
- **Customizability**: with customizability it is intended to configure and fine-tune several settings and configurations.

The third level represents the **alternatives** which were selected based on the web mapping service literature (Scharl and Tochtermann 2009). The combination of server types (WMS and TileServer) and data source (PgSQL and MBTiles) resulted in the following alternatives:

- **PgMS**: PostgreSQL/PostGIS with WMS
- **PgTS**: PostgreSQL/PostGIS with TileServer
- **MbMS**: MBTiles with WMS
- **MbTS**: MBTiles with TileServer

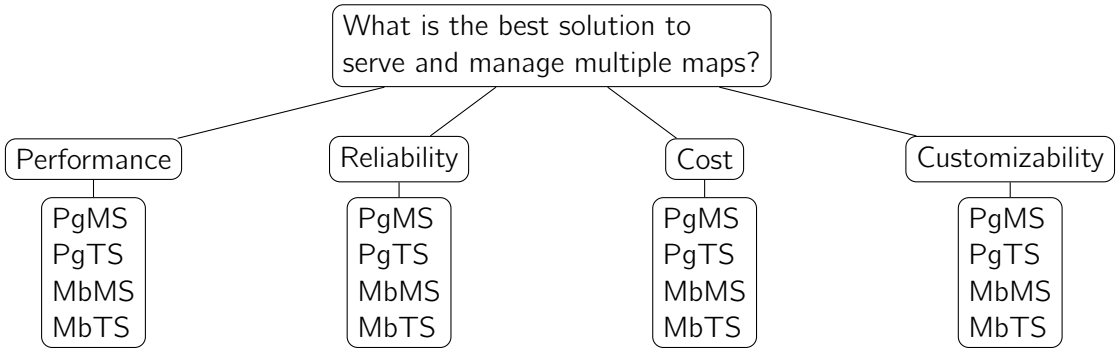


Figure 3.1: Hierarchical Decision Tree

The **second step** is to define the priorities for each criteria according with the numerical classification described in table 3.1.

Numerical Classification	Definition
1	Equal importance
3	Weak importance
5	Important
7	Very Important
9	Crucial

Table 3.1: Scale of importance AHP.

Based on this classification, for each pair of criteria, their relative importance was compared as it is shown in matrix represented in table 3.2.

	Performance	Reliability	Cost	Customizability
Performance	**1**	2	3	5
Reliability	1/2	**1**	2	3
Cost	1/3	1/2	**1**	2
Customizability	1/5	1/3	1/2	**1**

Table 3.2: Criteria pairwise comparison

Third step, from the criteria pairwise comparison, it is necessary to obtain the relative priority of each criteria, this is done by normalizing the comparison matrix. The values are represented on table 3.3.

Criteria	Relative Priority
Performance	0.482
Reliability	0.272
Cost	0.158
Customizability	0.089

Table 3.3: Relative priority for each criteria.

Analysing table 3.3, **performance** is the **criteria** with **highest priority** followed by reliability, cost and customizability. Being that performance is six times more preferred than the customizability criteria, three times more than cost and two times more than reliability. It is well known that performance is one of key factors of network services quality which consequently depends on the performance of servers (J. M. Almeida, V. Almeida, and Yates 1997). So, the performance relative priority value reflects the importance of service provides a map to a customer quickly.

In **step four**, it is important to evaluate the consistency of the relative priorities from the calculation of **Consistency Ratio (CR)**. When the CR is lower than 0.1 then the relative priorities are consistent. In this case the obtained CR was of **.0054**, assuring the consistency of the comparison matrix.

It gives enough confidence to move forward to **step five** where the **pairwise comparison matrix for each criteria** is built considering the four proposed alternatives (**PgMS**, **PgTS**, **MbMS**, **MbMS**).

The table 3.4 represents the prioritization of the alternatives by performance criteria where **MbMS** has the highest relative priority followed by **PgTS** > MbMS and with less priority **PgMS**.

	PgMS	PgTS	MbMS	MbTS	Relative Priority
PgMS	**1**	1/3	1/2	1/4	0.096
PgTS	3	**1**	2	1/2	0.277
MbMS	2	1/2	**1**	1/3	0.161
MbTS	4	2	3	**1**	0.466

Table 3.4: Alternatives relative priority for **performance** criteria

Regarding performance criteria:

- GIS DB servers: The values shows that WMS's alternatives are less priority than TileServer's alternatives. In fact, in the context of this project, TileServer is enhanced for caching and allows the display of native vectorial GIS data.
- Map server: since the focus of this project is on regional datasets, these results are expected. MBTiles is simpler and better suited for smaller and well structured amounts of GIS data, while PostgreSQL/PostGIS is prepared ot handle large and complex GIS datasets.

Thereafter it is time to analyse the relative priorities of each alternative based on the reliability criteria, table 3.5.

	PgMS	**PgTS**	**MbMS**	**MbTS**	**Relative Priority**
PgMS	**1**	1/3	2	1/2	0.161
PgTS	3	**1**	4	2	0.466
MbMS	1/2	1/4	**1**	1/3	0.096
MbTS	2	1/2	3	**1**	0.277

Table 3.5: Alternatives relative priority for **reliability** criteria

About reliability criteria:

- The relative order of priority of each of the alternatives is: **PgTS** > **MbMS** > **PgMS** > **MbMS**.
- GIS DB servers: PostgreSQL is better suited for large and customized geospatial datasets, and offers a greater flexibility and control over the management and manipulation compared to MBTiles (Stothers and Nguyen 2020).

For the prioritization analysis based on cost, on table 3.6,the most priority alternative is **PgMS**, two time more than **MbMS**. Then, with less relative priority **PgTS** followed by **MbMS**.

	PgMS	**PgTS**	**MbMS**	**MbTS**	**Relative Priority**
PgMS	**1**	3	2	4	0.466
PgTS	1/3	**1**	1/2	2	0.161
MbMS	1/2	2	**1**	3	0.277
MbTS	1/4	1/2	1/3	**1**	0.096

Table 3.6: Alternatives relative priority for **cost** criteria

In terms of cost, the most expensive infrastructure to hold a GIS DB is a PostgresSQL server. It is expected, that a WMS to serve a generic GIS datasets, like OSM, needs more computacional power than TileServer. Since MBTiles is optimized for OSM data the integrations of this GIS DB and TileServer is apparently the most cost effective.

To conclude this analyse,

	PgMS	PgTS	MbMS	MbTS	Relative Priority
PgMS	**1**	1/3	1/2	1/4	0.096
PgTS	3	**1**	2	1/2	0.277
MbMS	2	1/2	**1**	1/3	0.161
MbTS	4	2	3	**1**	0.466

Table 3.7: Alternatives relative priority for **customizability** criteria

The TileServer have the ability to display multiples output customized in the client side on the other hand the WMS only displays one style output possible. This flexibility is mostly because TileServer is able o delivery vectorial and raster data and WMS only raster (Netek et al. 2020).

Considering the analysis in phase five, in the last step, all the conditions are met to identify the best alternative which will solve the problem of this work: What is the best solution to serve and manage multiple maps? To archive that, it is computed the alternatives priority composite as a result of the priority vectors of each criterias comparison matrix multiplied with the criterias relative priority. The results are shown in 3.8.

	Performance	Reliability	Cost	Customiz.	Rel. Priority
PgMS	0.096	0.161	0.466	0.096	0.172
PgTS	0.277	0.466	0.161	0.277	0.310
MbMS	0.161	0.096	0.277	0.161	0.162
MbTS	0.466	0.277	0.096	0.466	**0.356**
Rel. Priority	0.482	0.272	0.158	0.088	1.000

Table 3.8: Alternatives composite priority

According with the defined criteria and their relative priority values, the alternative using MBTiles GIS DB with TileServer is the best solution with a relative priority of **0.356**. Followed closely with by the a solution using the same map server with PostgreSQL/PostGIS as GIS DB with a relative priority of **0.310**. Analysing the data in more detail is important to take note of the following remarks:

- **PgMS** and **MbMS** relative priority values are closer from each other but are lower than other two alternatives;
- **PgTS** and **MbMS** relative priority values are higher and closer from each other too;
- In terms of server type, TileServer, is for sure the best alternative instead of WMS;
- In terms of data source, in this work the original raw data information comes from a PBF file, and for each region of interest to serve the map would be downloaded and converted to a database structure PostgreSQL/PostGIS or MBTiles. Since MBTiles stores map data in a single SQLite database file, it will be more versatile to launch a service for a region of interest.

3.3 Value of the Solution

To determine the value of a solution, project managers may conduct a business case analysis, which includes identifying the costs and benefits of the solution, estimating the return on

investment (ROI), and considering any risks or uncertainties. The business case analysis helps stakeholders understand the potential value of the solution and make informed decisions about whether to proceed with the project.

3.3.1 Customer Value

According to American Marketing Association, customer value is the difference between the benefits a customer perceives from a market offering and the costs of obtaining those benefits (Kurian 2013). To deliver customer value, companies must understand their customers' needs and preferences, and strive to provide products and services that meet or exceed those needs.

3.3.2 Perceived Value

Perceived value refers to the customer's overall assessment of the utility of a product based on perceptions of what is received and what is given (Kurian 2013). Perceived value is a key factor in consumer decision-making and can influence a customer's willingness to pay for a product or service. To increase perceived value, companies can focus on improving the quality, features, and benefits of their products or services, and also on communicating the value proposition to customers through marketing and advertising.

3.3.3 Analysis of the Benefits and Sacrifices

In the context of this project, it is intended to create a solution that brings value to the company and consequently to the customer. Although in practice the **client** will not notice any perceptible benefit or sacrifice, as long as service availability is guaranteed, as well as the response time to get the map is similar to external commercial services like Google maps.

For **company**, the major **benefits** would be:

- Follow current practices in the provision of geospatial information via web maps.
- Use open and reliable data sources like OSM.
- Have a scalable solution.

And the major **sacrifices** would be:

- responsibility in the development of the infrastructure of the map service and maintenance of updated geospatial information.
- guarantee the availability and speed of access to the map service.

Chapter 4

Design

In this chapter will be presented and described the functional and non functional requirements and the solutions high level architecture followed by a brief explanation of each component.

4.1 Requirements

Requirements analysis is a process that allows to specify the customer's needs and expectations regarding a product. This section aims to present two types of requirements:

- Functional requirements: described through use cases,
- Non-functional requirements: described through FURPS+ specification.

4.1.1 Functional Requirements

Functional requirements are a set of statements that define what a software system must do in order to satisfy the needs of its users. These requirements describe the specific features and functions of the software that are required to achieve the desired outcomes (ISO/IEC/IEEE 2011).

In Figure 4.1, the functional requirements are represented trough a use case diagram using Unified Modeling Language (UML) notation.

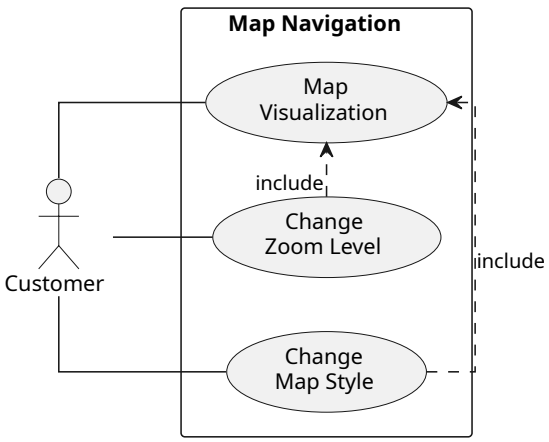


Figure 4.1: Project Use Case Diagram.

Each use case will be described and represented by a System Sequence Diagram (SSD) using UML notation.

Use Case 1: A customer accesses to web map service by using the preferred web browser or mobile app and then the map is displayed. Figure 4.2 represents the SSD for this use case.

Opening the service, the system fetches the vector tile data from geographic database and make them available to be displayed on the application.

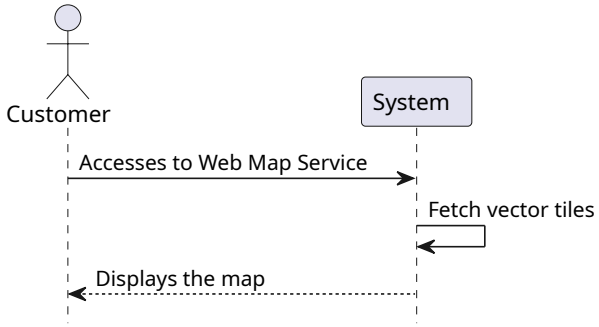


Figure 4.2: **UC1** - Customer has access to the web map.

Use Case 2: A customer changes the map zoom level. After map be displayed, the customer is able to zoom in and out a specific region. The system fetches the vector tile data by zoom level and tile coordinates *(x, y)* and make them available to be displayed on the application. Figure 4.3 represents the SSD for this use case.

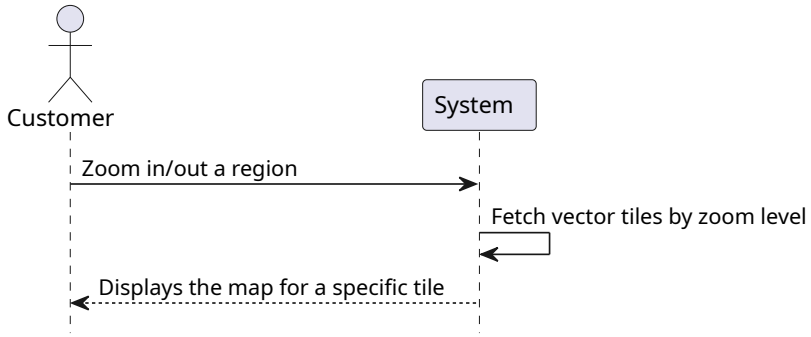


Figure 4.3: **UC2** - Customer changes the map zoom level.

Use Case 3: A customer requests a different map style, for example, road map; satellite or terrain. The available map styles are archived in a `Style.json` file which is applied to the application. Figure 4.4 represents the SSD for this use case.

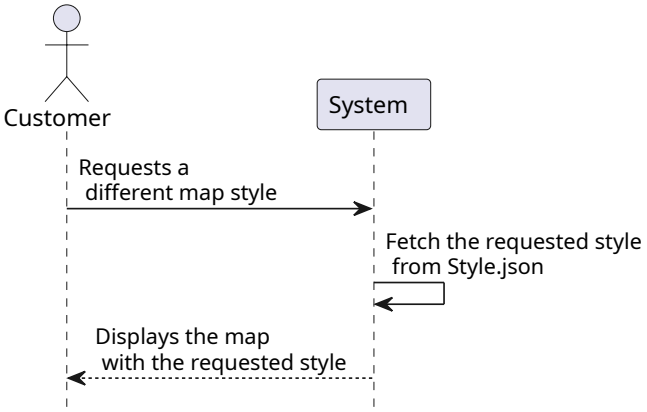


Figure 4.4: **UC3** - Customer requests a different map style.

4.1.2 Non Functional Requirements

Non-functional requirements are a set of criteria that define the qualities of a software system. These criteria describe how the software system should behave and operate, rather than what it should do (ISO/IEC/IEEE 2011).

As a way to organize and prioritize non-functional requirements, in software engineering is commonly use the FURPS+ classification. FURPS+ stands for Functionality, Usability, Reliability, Performance, and Supportability, with the plus sign indicating other non-functional requirements that do not fit into one of these categories like security (Grady 1992).

Focusing on non-functional acronym letters, here is a brief explanation of each one:

- **Usability**: The system must be easy to use, with a user interface that is intuitive and responsive.
- **Reliability**: The system must operate reliably, with a low rate of failures and errors, and be able to recover quickly from any failures that do occur.
- **Performance**: The system must perform at an acceptable level, with fast response times and high throughput, even under heavy loads.
- **Supportability**: The system must be easy to maintain and support, with clear documentation.
- **+**: Evaluate other non-functional requirements like security, scalability, compatibility, and regulatory compliance.

In the context of this project it was identified the above use cases associated to performance and reliability non-functional requirements.

Performance:

- The system should display the requested map on demand. Using as a reference the performance testing carried out by (Netek et al. 2020), the initial map loading took about 1200 milliseconds. When using any web service, the customer expects the page to load in 0.5 to 1 second. So, these references values will be used in this work to evaluate the system.

Reliability:

- The system should work with as few failures as possible during the time it is operating. It means that the server should be up and running and data should be available each time the customer makes a request.
- The geospatial data is updated every time there is new changes from data source (OSM).
- The system should display the most recent geospatial data.

4.2 Architecture

The **architecture** of the proposed solution is described here, presenting the structure of the system that will be developed and its components and relations. The UML deployment diagram shown in Figure 4.5 represents the development perspective of proposed solution.

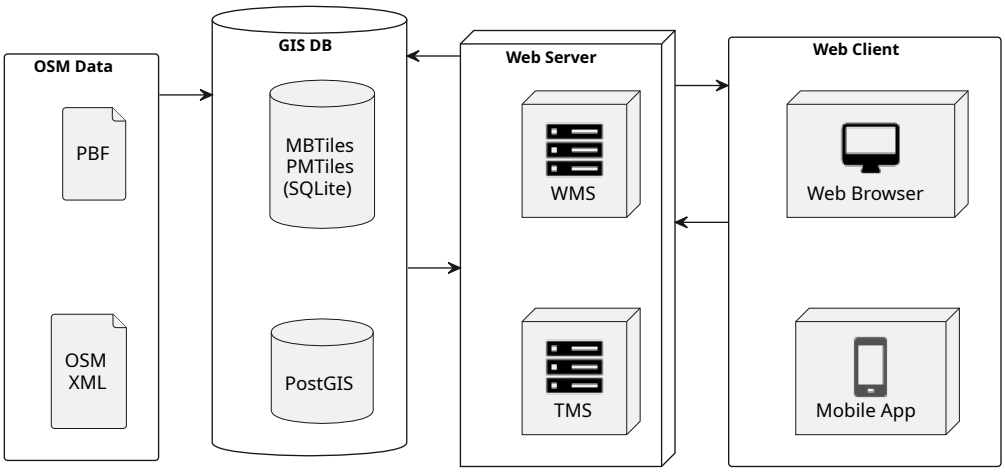


Figure 4.5: Deployment diagram.

4.3 Internal Components

The proposed solution is composed by the following elements:

- **OSM data** from two different data formats type/sources:
  - OSM XML file from OSM Application Programming Interface (API).
  - PBF file from data sources like GeoFabric (preferred file format).
- **GIS DB** from two different DB:
  - MBTiles is a static file SQLite DB.
  - PostGIS an extention of PostgreSQL DB Server.
- **Web Server**
  - WMS an OGC standard map service.
  - TMS a map service based on tiles more suited for web.
- **Web Client** desktop or mobile application.

Chapter 5

Implementation

During implementation, Docker was widely used as simple as `docker run` or in a `docker compose` to speed up the integration process. There were conducted alternative processing and server solution, some with useful results and some not, but only the meaningfully steps were documented here.

As mentioned initially in the introduction, Chapter 1, there is currently a solution in production based on TileServer GL (MapTiler 2023a) using an MBTile built with Tilemaker (**github :tile creator**).

The following overview diagram, Figure 5.1, conceptually presents the current and proposed solution, the differences are in the implementation details. There is no better solution than using a TMS for publishing maps on the web. The use of a WMS is not an option here, as introduced/explained in the State of the Art, Chapter 2.

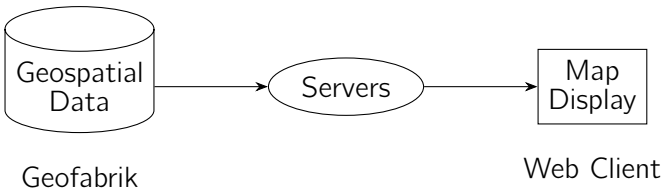


Figure 5.1: Overview implementation diagram

The proposed solution is presented in the following order, with a short introduction:

Geospatial Data is built using the tool `Planetiler` (Barry 2023), writen in java, that generates Mapbox Vector Tiles (MBTiles/PMTiles) ready to be served.

Servers: The TMS is the `Martin` (MapLibre 2023b) with the support of the Nginx web server.

Map Display: The client-side and the map rendering happens mostly in a web browser, but there was used the `QGIS` (QGIS 2023), an open-source GIS software, to verify the geospatial data during the process.

5.1 Geospatial Data

The Planetiler was the tool used and proposed to create a MVT from a OSM geographic region using a Geofabrik raw `.osm.pbf` file. The result is a multi-zoom vector tile GIS DB archive in the MBTile or PMTile file format. Figure 5.2 represents the SSD of this process.

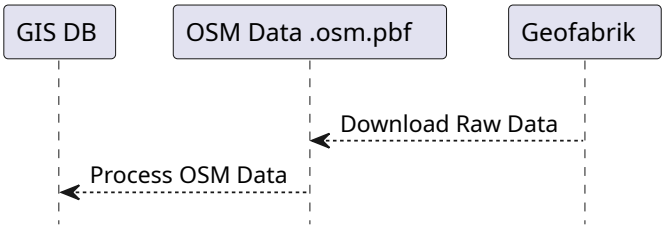


Figure 5.2: SSD representing the download and processing of OSM data into a GIS DB archive.

The following Docker command, Listing 5.1, builds a MBTile of Portugal in less then 10 minutes using 5GB of RAM with a Intel®Core™i7-6500U. Using the zoom level 15, the resulting MBTile becomes larger than the original `.osm.pbf` base file. Currently the `.osm.pbf` Geofabrik file of Portugal is approximately 300MB and after being processed by the Planetiler it is transformed into an MBTile file of approximately 400 MB.

```
1  # Portugal
2
3  docker run --rm \
4  -e JAVA_TOOL_OPTIONS="-Xmx5g" \
5  -v "$(pwd)/planetiler-data":/data \
6  ghcr.io/onthegomap/planetiler:latest \
7  --download --area=portugal \
8  --maxzoom 15 \
9  && \
10 cp planetiler-data/output.mbtiles portugal-mbtiles.mbtiles
```

Listing 5.1: Using Planetiler to build a MBTile of Portugal with Docker

Compared to Tilemaker, the Planetiler takes care of all geospatial data dependencies including the main `.osm.pbf` source data. There is typically about 1GB of extra data downloaded with coastlines and water regions to clip/crop and complete the OSM basemap. These auxiliary data files are required to properly build the MBTile either using the Tilemaker or the Planetiler.

Usually these auxiliary files are three, for example:

- `water-polygons-split-3857.zip`
- `lake_centerline.shp.zip`
- `natural_earth_vector.sqlite.zip`

Two Shapefiles with characteristics of lines and polygons coastlines and water regions. A sqlite with a geospatial database with points, lines and polygons of low resolution information such as toponymy of countries and continents and their borders.

The Planetiler wraps everything in a single command line and also offers the possibility to build the MVT as a PMTile file, that will be useful as option in the serverless implementation, in the Section 5.2.1.

The OSM data provider Geofabrik makes available a GeoJSON with all regions and sub-regions available to download. There is a special region, which is not provided by Geofabrik, which contains all the data to generate the world map and is called Planet. At the time at the time of writing it has a size of approximately 70 GB. Processing the Planet's MBTile using a Docker command similar to, Listing 5.1, resulted in a +150GB file after 7 hours of processing on a dedicated computer with plenty of CPU cores and +100GB of RAM.

Using the QGIS it is possible to analyse the GeoJSON, perform queries and filter the results as in Figure 5.3.

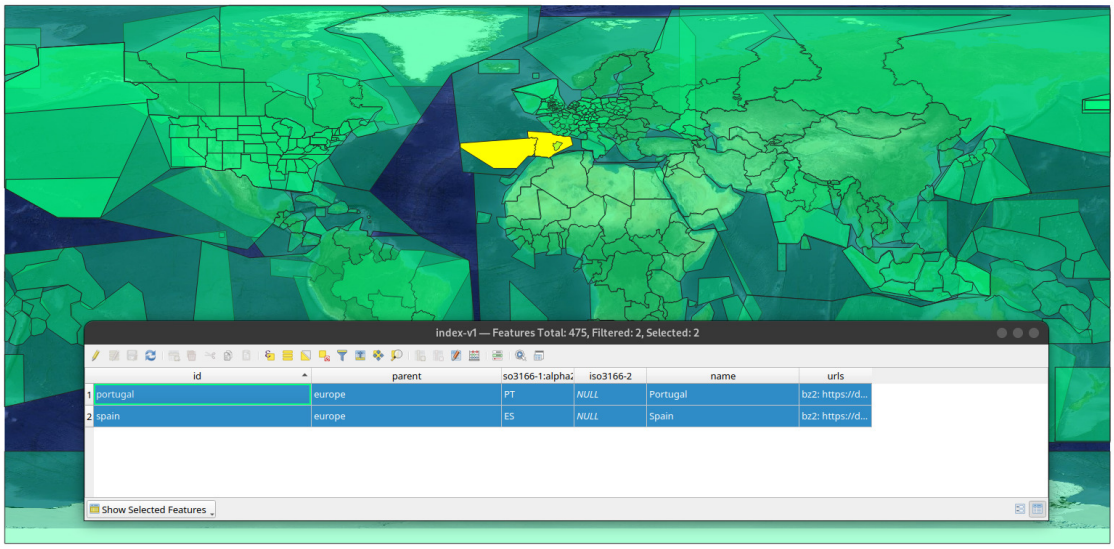


Figure 5.3: Using QGIS to query Geofabrik's GeoJSON to select the regions of Portugal and Spain

GDAL's `ogrinfo` command-line tool allows the same kind of functionalities as QGIS. This is easily integrated into a shell script or throw GDAL API with Java, Python or C/C++, etc.

The `ogrinfo` is also able to read MBTiles, and so, useful for data verification, for example reading the `france-guyane-mbtiles.mbtiles` file in the terminal. It is illustrated in Listing 5.2.

```
$ ogrinfo -so -al france-guyane-mbtiles.mbtiles

INFO: Open of 'france-guyane-mbtiles.mbtiles'
      using driver 'MBTiles' successful.

Metadata:
  ZOOM_LEVEL=15
  format=pbf
  center=-53.05215,4.23798,7
  bounds=-54.61269,2.10568,-51.49161,6.37029
  ...
  minzoom=0
  maxzoom=15
  ...
  planetiler:buildtime=2023-05-26T10:21:12.246Z
  planetiler:osm:osmosisreplicationurl=http://download.geofabrik.de/europe
    /france/guyane-updates
  ...
```

Listing 5.2: Using `ogrinfo` to verify a MBTile file.

Opening the same MBTile from the French Guiana in QGIS allows the possibility to inspect the data graphically, examples in Figures 5.4 and 5.5.

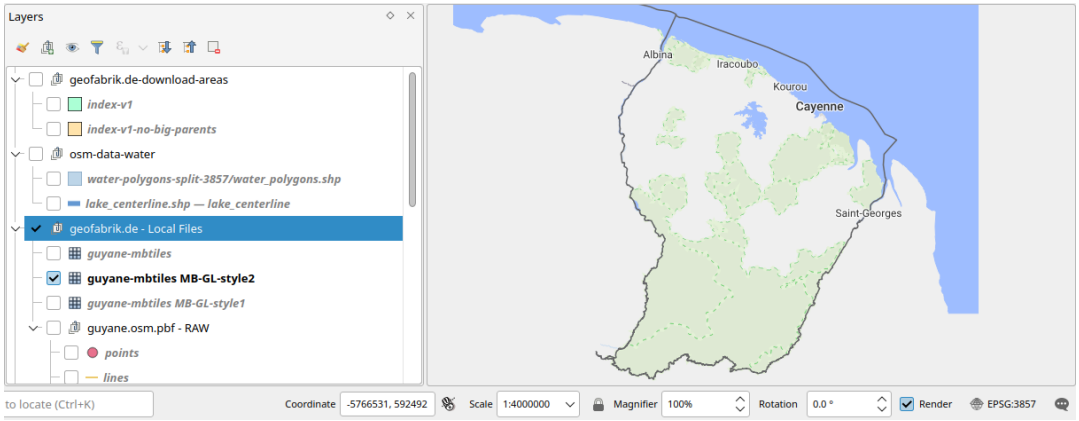


Figure 5.4: Local MBTile from the French Guiana in QGIS with zoom out.

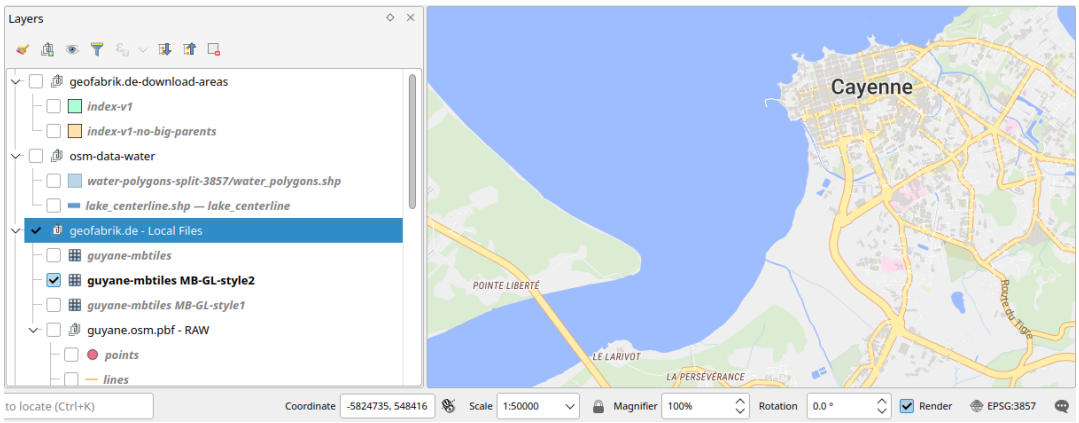


Figure 5.5: Same MBTile with zoom in over Cayenne.

Build a planet size MBTile is not a very practical approach mainly due to maintenance of a data file of more than 100GB.

In an attempt to reproduce a real situation, an MBTile file was created for the major regions of France. The identification of regions was achieved by querying the Geofabrik GeoJSON using the `ogrinfo` tool, presented in the following terminal example, Listing 5.3.

```
# ogrinfo -ro -al -geom=NO -where "parent = 'france'"  https://download.
    geofabrik.de/index-v1.json

# or using SQL

$ ogrinfo -ro -geom=NO -sql "SELECT id FROM \"index-v1\" WHERE parent = '
    france'" https://download.geofabrik.de/index-v1.json

INFO: Open of `https://download.geofabrik.de/index-v1.json'
      using driver `GeoJSON' successful.

Layer name: index-v1
Geometry: Multi Polygon
Feature Count: 27
Extent: (-62.017190, -21.729220) - (56.119990, 51.281210)
Layer SRS WKT:
GEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]]],
...
    ID["EPSG",4326]]
...
  id (String) = guadeloupe
  id (String) = guyane
  id (String) = martinique
  id (String) = mayotte
  id (String) = reunion
...
```

Listing 5.3: Using the `ogrinfo` to query Geofabrik GeoJSON.

Besides the Metropolitan France, there were a few more regions chosen to build the MBTile files. Using a shell script function called `build-planetiler-area`, Listing 5.4, simplifies the process.

```
# Shell script function - build-planetiler-area

$ build-planetiler-area () {

mkdir -p $AREA && cd $AREA \
&& \
docker run --rm \
-e JAVA_TOOL_OPTIONS="-Xmx30g" \
-v "${PWD}/${AREA}-planetiler-data":/data \
ghcr.io/onthegomap/planetiler:latest \
--download --area=${AREA} \
--maxzoom 15 --force \
&& \
cp ${AREA}-planetiler-data/output.mbtiles ${AREA}-mbtiles.mbtiles
}
```

Listing 5.4: Shell script function to run multiple Dockers with Planetiler.

The major areas from France, outside Metropolitan France or European France are the following: Guadeloupe, Guyane, Martinique, Mayotte, Reunion.

After running the `build-planetiler-area` shell script function, Listing 5.5, the MBTiles for the areas previously listed were built.

```
# run for each region

$ AREA=guadeloupe
$ build-planetiler-area

$ AREA=guyane
$ build-planetiler-area

$ AREA=martinique
$ build-planetiler-area

$ AREA=mayotte
$ buil-planetiler-area

$ AREA=reunion
$ build-planetiler-area
```

Listing 5.5: Running the shell script function `build-planetiler-area`.

The geospatial data is now collected to explore the server possibilities.

5.2 Server

The SSD in Figure 5.6 illustrates, in a simplified way, the process of the map server accessing the stored geospatial databases.

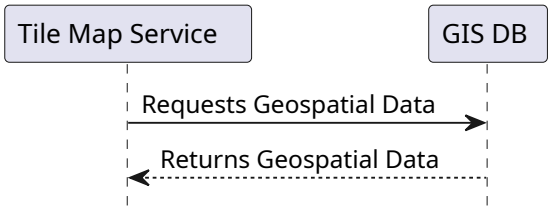


Figure 5.6: SSD representing the TMS accessing the GIS DB.

In the context of this work, a total of eight MBTiles were generated to be used during the server implementation. The following list shows the names of the files and their sizes:

```
- france-mbtiles.mbtiles (Europe)      7 GB

- france-guadeloupe-mbtiles.mbtiles   26 MB
- france-guyane-mbtiles.mbtiles       43 MB
- france-martinique-mbtiles.mbtiles   20 MB
- france-mayotte-mbtiles.mbtiles      10 MB
- france-reunion-mbtiles.mbtiles      36 MB

- portugal-mbtiles.mbtiles           374 MB

- planet-mbtiles.mbtiles             154 GB
```

The following code, Listing 5.6, shows a Docker Compose with server configuration where the TileServer GL is also available but the main focus is the Martin and the Nginx.

The Nginx became important later, not only because it ties everything together with a Reverse Proxy and Cache capabilities, but also because of its HTTP range request functionality.

The HTTP range request or partial requests will come in handy later using PMTiles files in a serverless implementation of a map service, introduced in Section 5.3.

The TileServer GL were kept in the Docker Compose file to compare results a check performance differences. TileServer is only serving information from the planet `-mbtiles.mbtiles` database, as can be seen in line 24 of the configuration file, Listing 5.6.

```yaml
version: "3"
services:

  webserver:
    image: nginx
    container_name: webserver
    hostname: webserver
    ports:
      - "80:80"
    volumes:
      - ./nginx/conf.d/:/etc/nginx/conf.d/:ro
      - ./data-dir:/usr/share/nginx/html:ro
    restart: unless-stopped

  tileserver-gl:
    image: maptiler/tileserver-gl
    container_name: tileserver-gl
    hostname: tileserver-gl
    ports:
      - "8080:8080"
    volumes:
      - ./data-dir/data/gis/osm/mbtiles:/data:ro
      - ./tileserver/styles:/usr/src/app/node_modules/tileserver-gl-styles
    /styles:ro
    command: -V --mbtiles planet-mbtiles.mbtiles
    restart: unless-stopped

  martin:
    image: ghcr.io/maplibre/martin
    container_name: martin
    hostname: martin
    ports:
      - "3000:3000"
    volumes:
      - ./data-dir/data/gis/osm/mbtiles:/data:ro
    command: /data
    restart: unless-stopped
```

Listing 5.6: Docker compose with Martin and Nginx.

The Martin Tile Server is very versatile solution, not only accepts a diversity of geospatial database inputs but also introduces a key functionality called Composite Sources. The Composite Sources allows the combination of multiple geospatial databases into one, example in Figure 5.8. This is extremely useful, as it allows to keep data distributed across multiple files, and therefore, keeps the maintenance of the different databases independent. In the next Figure 5.7 it is possible to verify the eight MBTiles available in the Martin's catalog.

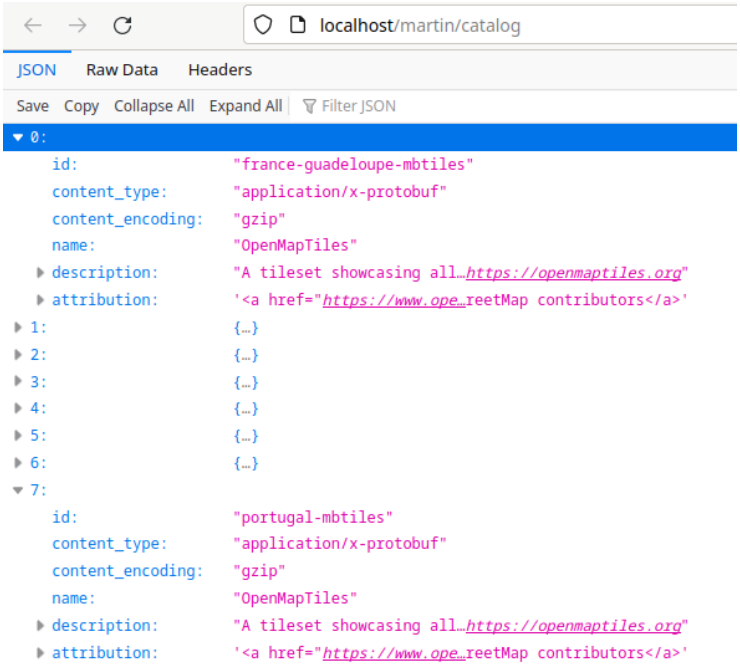


Figure 5.7: Martin catalog

Using Martin's Composite Sources is as simple as writing the multiple DB names in the endpoint Uniform Resource Locator (URL), as shown in Figure 5.8 where Portugal and France DB are being used.

The tiles are available under the following ZXY scheme:

- `localhost/martin/source1,source2,sourceN/{z}/{x}/{y}`

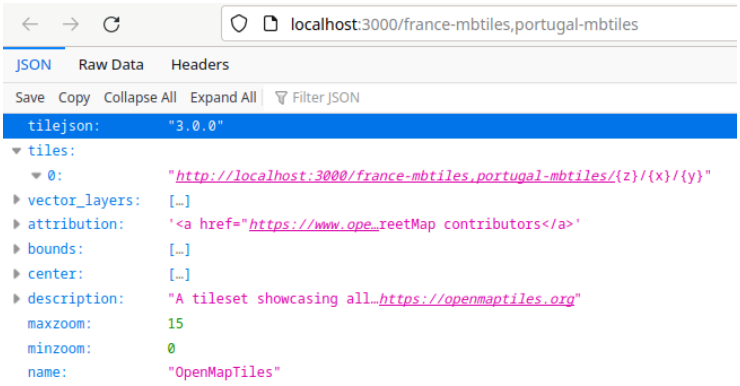


Figure 5.8: Martin Composite Source using France and Portugal MBTiles.

The Martin's URL structure allows the use of a Reverse Proxy to change the HTTP path, and this is where Nginx comes into play, by rewriting the path as:

- `localhost/martin/pt-fr/{z}/{x}/{y}` (example)

To configure Nginx as a reverse proxy server and pass requests to an HTTP proxied server like Martin, the `proxy_pass` command is used for each new `location`, following the example used in the Listing 5.7.

```
server {

listen 80 default_server;
listen [::]:80 default_server;
server_name _;

location / {

  root   /usr/share/nginx/html;
  index  index.html index.htm;
  autoindex on;
}

location /martin {

  proxy_set_header   X-Real-IP         $remote_addr;
  proxy_set_header   X-Forwarded-For   $proxy_add_x_forwarded_for;
  proxy_set_header   Host              $host;

  proxy_pass http://martin:3000/;
}

location /martin/france {

  proxy_set_header   X-Real-IP         $remote_addr;
  proxy_set_header   X-Forwarded-For   $proxy_add_x_forwarded_for;
  proxy_set_header   Host              $host;

  proxy_pass http://martin:3000/france-mbtiles,france-guadeloupe-mbtiles,
    france-martinique-mbtiles,france-guyane-mbtiles,france-mayotte-mbtiles,
    france-reunion-mbtiles,portugal-mbtiles/;
}
}
```

Listing 5.7: Nginx Reverse Proxy configuration with Martin.

5.2.1 Serverless Implementation with PMTiles

A serverless implementation with PMTiles was made available as a proof of concept using the Nginx HTTP Range Request capabilities. A standard configuration of Nginx was installed in small dual-core server and made available publicly under the domain: `srvr.apinhal.pt`.

To convert MBTiles to PMTiles archives, there is a dedicated command line `pmtiles` from the project Protomaps (Protomaps 2023). Using this tool is not only possible to convert to PMTiles, but also inspect the archives and upload to compatible cloud storage.

Converting it is a simple as: `pmtiles convert in.mbtiles out.pmtiles`

The Protomaps promotes this serverless approach to deploying as TMS from single Cloud-optimized static file. This approach was discovered at the end of the semester and allows to test this solution with the help of a CDN. This is possible because the server goes public through a Cloudflare network that relies on its own CDN.

5.3 Map Display

When the customer accesses the TMS through the QGIS desktop application or the web client, the requested map is displayed. The map displayed is the result of the integration of three contributions, the map display with front-end, server with back-end and the data with GIS DB.

Publishing map data from Martin is available for both QGIS and to the Web client. On the other hand, accessing directly to PMTiles in serverless mode is only available for the web client, as schematized in Figure 5.9.

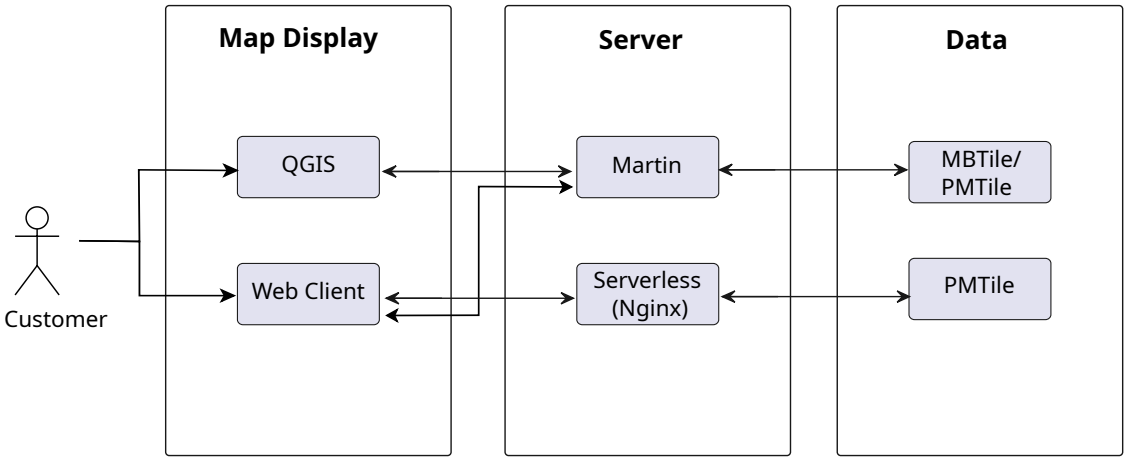


Figure 5.9: The map displayed is the result of the integration of geospatial data, web servers and front-end application components.

QGIS proved to be the right companion during the project development, from the raw data inspection to server testing, including the introduction of map styles rendering. This partnership did not exclude some front-end development to test the PMTiles performance using a web browser as client.

During the implementation process a big QGIS project grown with a lot of layers that the following, Figure 5.10, shows.

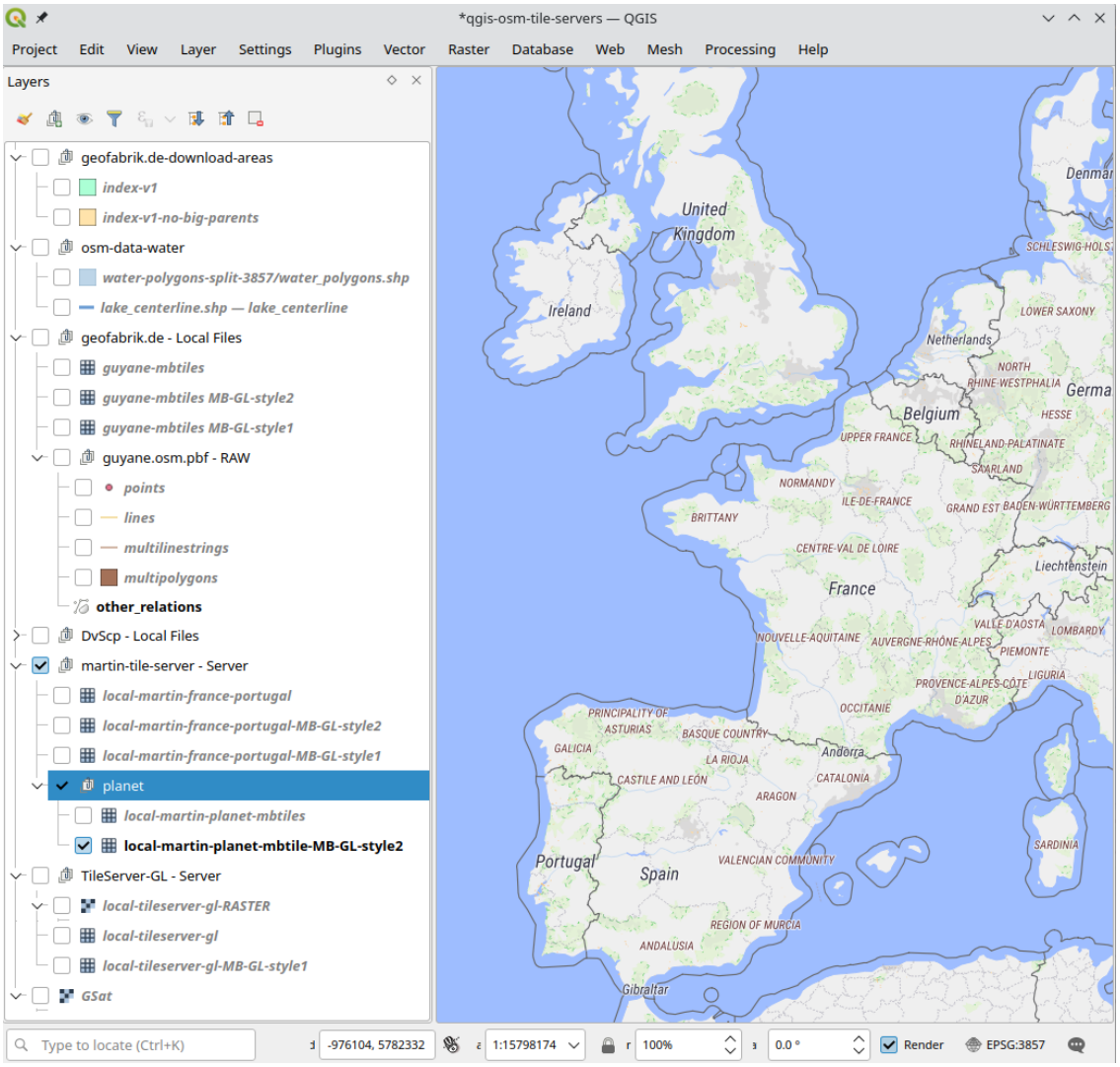


Figure 5.10: QGIS project with the list of most geospatial layers used.

The map displayed in the QGIS project is from the Martin server using the Planet MBTtile with a map style applied. Without a map style, all the lines, the points and the polygons have the same style as it is shown in Figure 5.11.

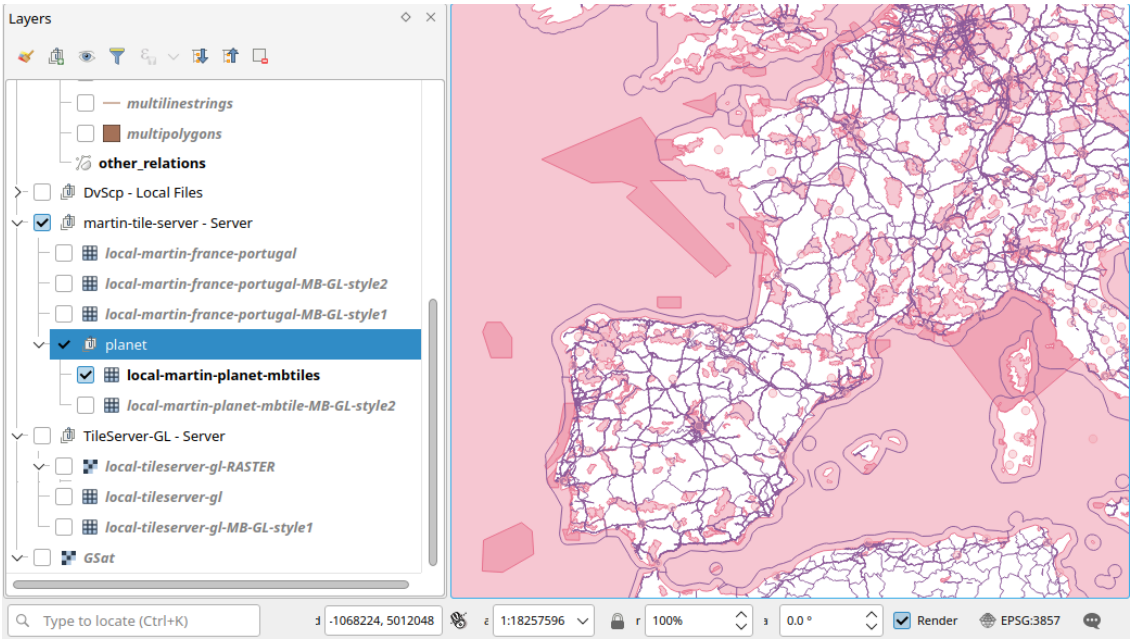


Figure 5.11: QGIS project with the list of most geospatial layers used.

Without a map style there is no toponymy as well, which means there is no place names, road names, no labels.

5.3.1 Map Styles

The map style is a JSON file identified as `style.json`, which follows the Mapbox Style Specification, establishing a hierarchical relationship between the zoom level and the layers styles. This means that based on the style, a layer may only appear after a certain zoom level, like the buildings, or to decide the toponymy detail labels to display by country, by district or at street level.

Styling a map is more than just colors a lines thickness, there's tools dedicated to this purpose such as the open-souce maputnik and the open style free to use, the OSM Liberty from the same source (Maputnik 2023).

During the transformation from the raw base file `.osm.pbf` to the MBTile/PMTile ready-to-be served the geospatial information goes throw a map/cartografic generalization process. The purpose of this process is to maintain only the appropriate level of detail for each zoom level. This means deleting features or simplifying their geometries while respecting the zoom's geospatial precision level.

The following Figure 5.12 shows the QGIS symbology after importing a `style.json` to represent a MBTile served by Martin, and at the end of the print screen, there is a rule that buildings only appear after zoom level 13.

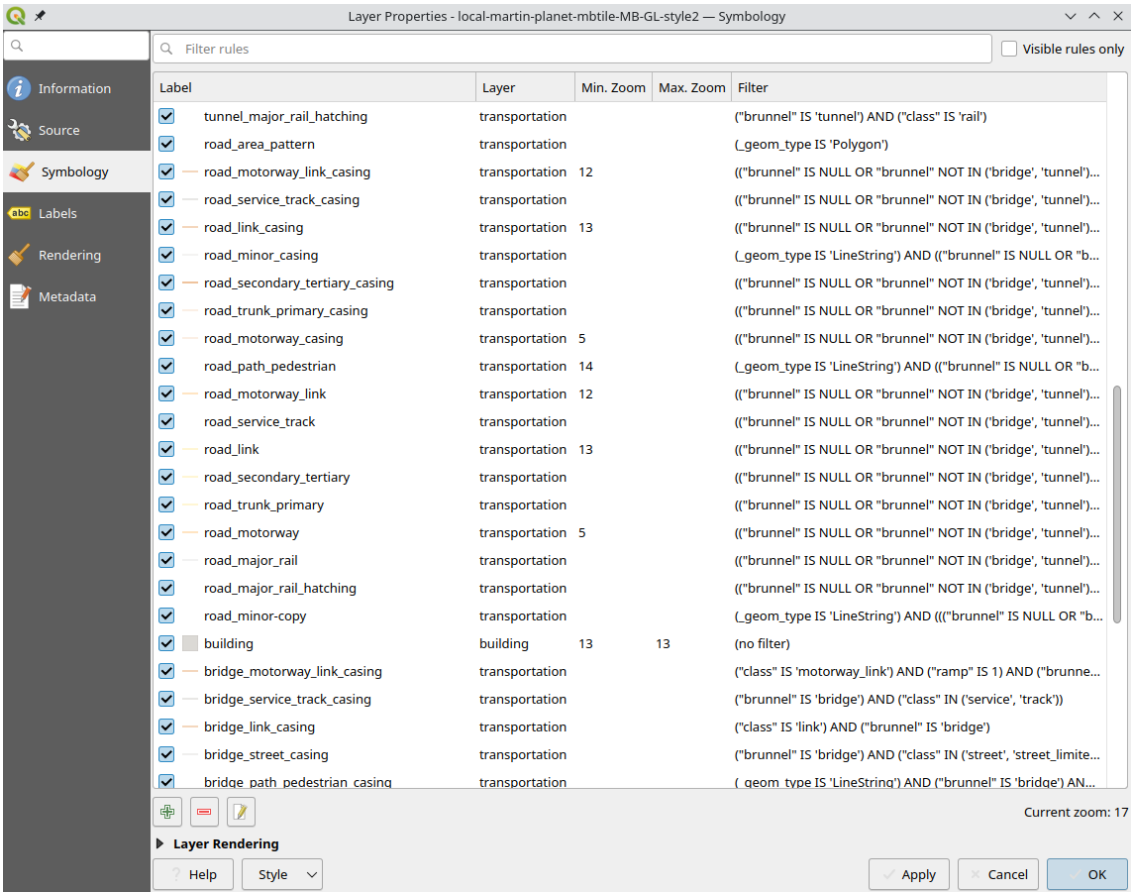


Figure 5.12: Symbology style in Qgis after importing a `style.json` file.

5.3.2 Serverless Map Display with PMTiles

On a server running Nginx and hosting the PMTiles archives a small web page was made available throw the following code, Listing 5.8, to publish the map using the JavaScript (JS) MapLibre GL (MapLibre 2023a). MapLibre is also responsible for the development of Martin.

```
1  <html>
2
3  <head>
4
5  <title></title>
6  <meta charset="utf-8"/>
7
8  <link rel="stylesheet" href="https://unpkg.com/maplibre-gl@2.4.0/dist/
       maplibre-gl.css" crossorigin="anonymous">
9  <script src="https://unpkg.com/maplibre-gl@2.4.0/dist/maplibre-gl.js"
       crossorigin="anonymous"></script>
10 <script src="https://unpkg.com/pmtiles@2.4.0/dist/index.js"></script>
11 <script src="https://unpkg.com/protomaps-themes-base@1.1.0/dist/index.js">
       </script>
12
13 <style>
14 body { margin: 0; }
15 #map { height:100%; width:100%; }
16 </style>
17
18 </head>
19
20
21 <body>
22
23
24 <div id="map"></div>
25
26 <script type="text/javascript">...</script>
27
28
29 </body>
30
31
32 </html>
```

Listing 5.8: Planet PMTile client: `index.html` with the javascript code in Listing 5.9.

The web browser understands the PMTiles and display the map using the MapLibre GL JS through the use of the PMTiles JS library from the Protomaps, using the code lines 3 and 4 of the JS code Listing 5.9, (Protomaps 2023).

```
<script type="text/javascript">

let protocol = new pmtiles.Protocol();
maplibregl.addProtocol("pmtiles",protocol.tile);

let URL = "https://srvr.apinhal.pt/data/gis/osm/pmtiles/planet-mbtiles.pmtiles";

  const map = new maplibregl.Map({
    maxZoom: 18,
    container: 'map',
    hash: true,
    style: {
      version:8,
      glyphs:'https://cdn.protomaps.com/fonts/pbf/{fontstack}/{range}.pbf',
      sources: {
        "protomaps": {
          type: "vector",
          url: "pmtiles://" + URL,
          attribution: 'Protomaps © <a href="https://openstreetmap.org">OpenStreetMap</a>'
        },
      },
      layers: protomaps_themes_base.default("protomaps", "light")
    }
  });

</script>
```

Listing 5.9: Planet PMTile client: `javascript`.

A print screen of the resulting web pages the the serverless map is available in Figure 5.13, displaying a region of Espinho, Portugal.

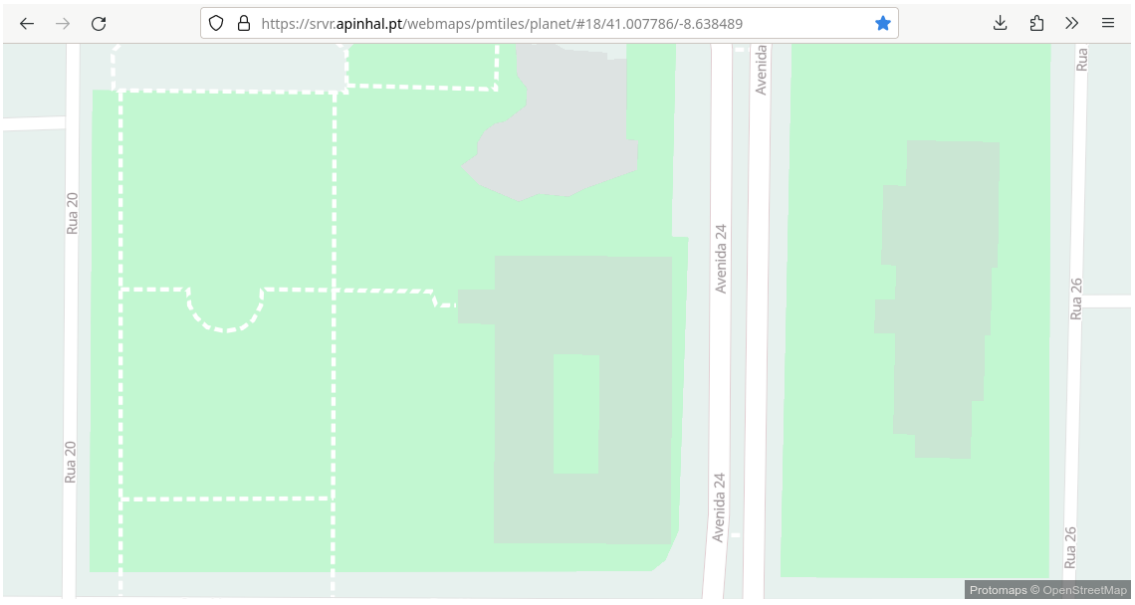


Figure 5.13: Web page displaying the serverless map from the Planet PMTile hosted in the Nginx server.

The Planet PMTile in use was converted from the previously generated Planet MBTiles using Planetiler, documented in Section 5.1, and hosted in the Nginx server under the URL defined at line 6 of the JS code in the Listing 5.9.

Access to a TMS/ZXY server uses the following scheme: `domain/{z}/{x}/{y}`. The URL accessed in the Figure 5.13 has the same logic and points to the following coordinates at zoom level 18: `#18/41.007786/-8.638489`.

In an attempt to test and display the map using different PMTiles with distinct zoom levels, two extra file were downloaded from the Protomaps demo data. A total of three PMTiles files were made available at `srvr.apinhal.pt/webmaps/pmtiles/`

A list with the maximum zoom and file sizes followed by the URL to access each PMTile map is listed below:

```
- planet-mbtiles.pmtiles                      (max. zoom 15)  +100 GB
        planet/#18/41.007786/-8.638489

- protomaps_vector_planet_odbl_z10.pmtiles    (max. zoom 10)    +2 GB
    planet-z10/#18/41.007786/-8.638489

- vn-gaia.pmtiles (samll region)              (max. zoom 14)     7 MB
          gaia/#18/41.007786/-8.638489
```

There is a big difference in detail from zoom level 10 to 15, but there is no graphical difference from 14 to 15, at least for these data examples. In fact, the zoom level 14 is Planetiler's default value for generating MBTiles/PMTiles.

Precision doubles with each new zoom level. The precision associated with zoom level 14 is around 0.5 meters, so the precision doubles to 0.25 metres at zoom 15 (Fischer 2023). These precision values are higher than the most of the OSM geospatial data, because using standard GNSS receiver like in a smartphone, the horizontal positional error is no better than 4 meters. Therefore, there is no major improvement in generating vector tiles at zoom level 15, for most of the maps displayed on the web.

For styles comparison, a QGIS image displaying the same area using a different map style from the PMTile serverless web page, Figures 5.14 and 5.13, respectively.

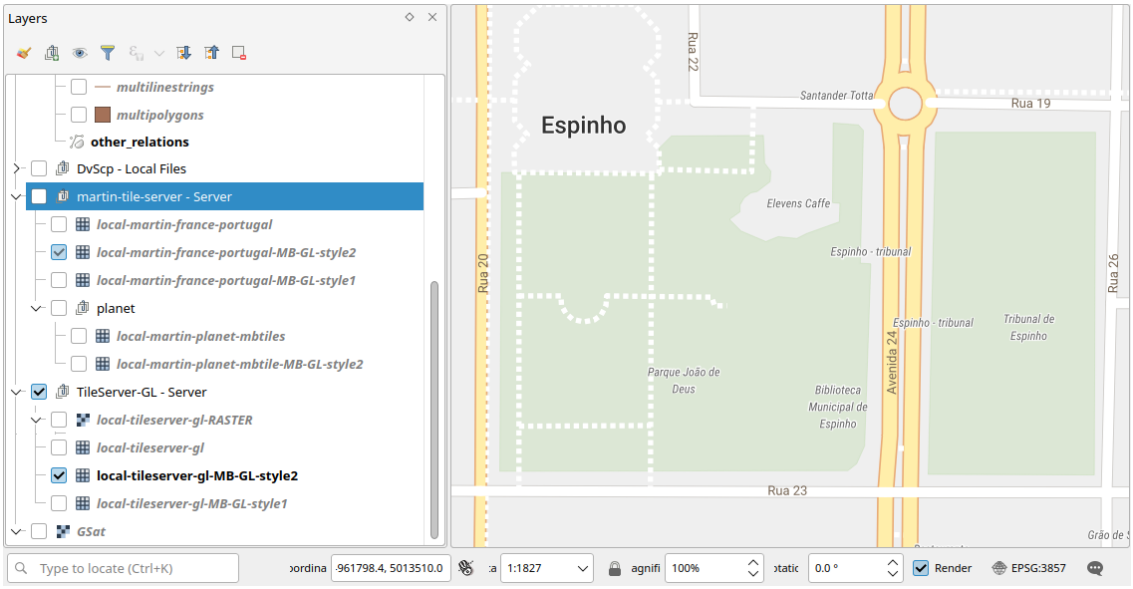


Figure 5.14: QGIS representing the same region as previous Figure 5.13 using a different style.

The style of the map is very important, even with the same database source the final result can be very different.

Chapter 6

Evaluation

This chapter shows the evaluation process of the proposed solution. In this sense, hypotheses will first be formulated, indicators and sources of information are defined, the pretended evaluation methodology will be described and, finally, the results of measured metrics.

6.1 Hypothesis

As defined in value analysis chapter, a problem is formulated in the form of a question serving as the basis from which a hypothesis is derived. A hypothesis is a suggested solution to a problem that should be tested and verified (Bulajic, Stamatovic, and Cvetanovic 2012).

In order to evaluate the developed solution, the following assumptions will be taken into consideration:

1. Creating/Updating a geographic database based on open geospatial data sources.
2. Availability of the map server from the geographic database.
3. The architecture allows the system to be scalable, supported by technologies such as Docker, Kubernetes, and a Content Delivery Network.

6.2 Indicators and Sources of Information

From the hypothesis previously defined, it is intended to evaluate two characteristics: **performance** and **reliability** of the system. Being a system composed of several components, it is important to evaluate each one as well as the interaction between them.

In terms of **performance**, the **response time** between obtaining the geospatial data sources, with different sizes, until they are published on a web platform, is a critical metric in this type of system. As well as the evaluation of the system's ability to process multiple requests in a period of time.

Regarding **reliability**, the intention is to evaluate whether the system **works flawlessly**, consistently performing its function over a period of time.

6.3 Evaluation Methodology

In order to **evaluate the system performance**, response time method will be used to measure the **system's responsiveness** in the following situations:

- Download the original geospatial data (raw data).
- Creating/Updating a geographic database with new geospatial data for a given region.
- Install and running the map server.
- Multiple requests to map server at same time.

For the **reliability evaluation**, given the geospatial data source is updated, it is intended to validate the database update only with the differences. Additionally, calculate the **Mean Time Between Failures (MTBF)** metric which represents the average time between failures of a system or component (Torell and Avelar 2004).

$$MTBF = \frac{Total\,operating\,time}{Number\,of\,failures} \tag{6.1}$$

It is calculated by dividing the total operational time of the system by the number of failures that occurred during that time and it is expressed in hours per failure, equation 6.1.

MTBF is correlated with reliabilty from the mathematical expression illustrated in equation 6.2

$$Reliability = e^{-(\frac{Time}{MTBF})} \tag{6.2}$$

The higher the MTBF number is, the higher the reliability of the product.

6.4 Evaluation Measured Metrics

At this stage of the work where the goal is to measure the quality of the proposed solution, it made sense to analyze the metrics associated with performance and reliability, namely, server response time and the integrity of the geographic data presented. The reliability evaluation as described in Section 6.3, equation 6.2, was not measured.

6.4.1 Servers repose time and errors

The performance evaluation was carried out following the same path using a dedicated web client for all the servers, the TileServer GL, the Martin and the Nginx in serverless mode with PMTiles. The zoom level of requests was kept high during the tests ensuring the highest resolution available (zoom 15) and the highest number of tiles fetched, as shown the example in Figure 6.1.

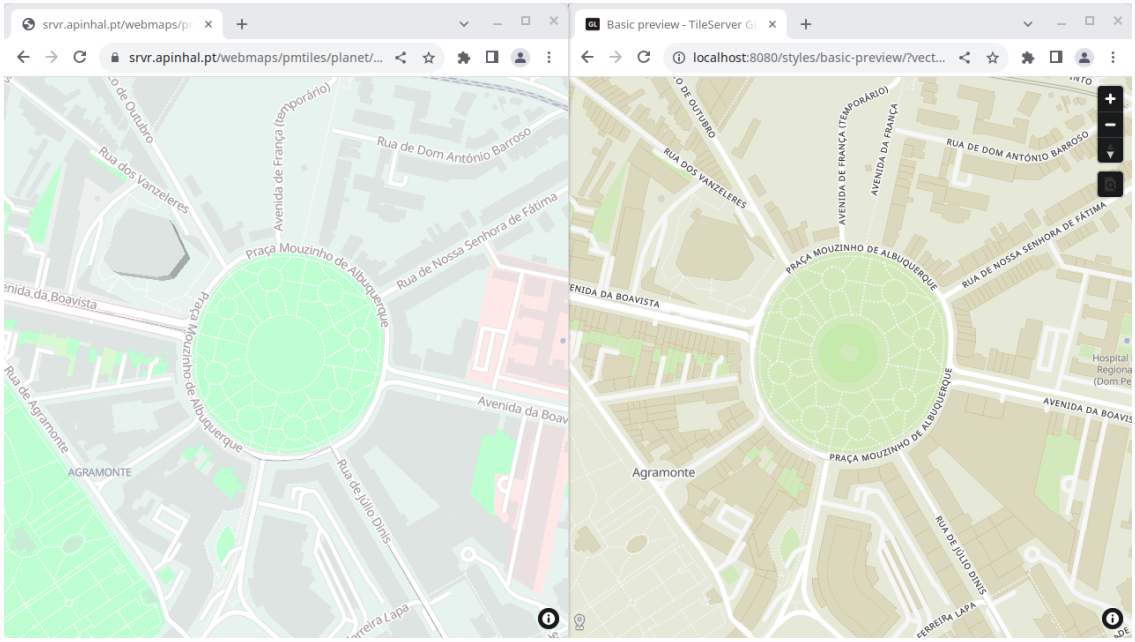


Figure 6.1: Nginx (serverless) on the left and TileServer GL on the right with each web clients displaying the same map overview during response time test.

The path started in Porto, city center, as shown in Figure 6.1, navigating south on the map along the Atlantic coast. The path ends in Aguda, Vila Nova de Gaia, as shown in Figure 6.2.

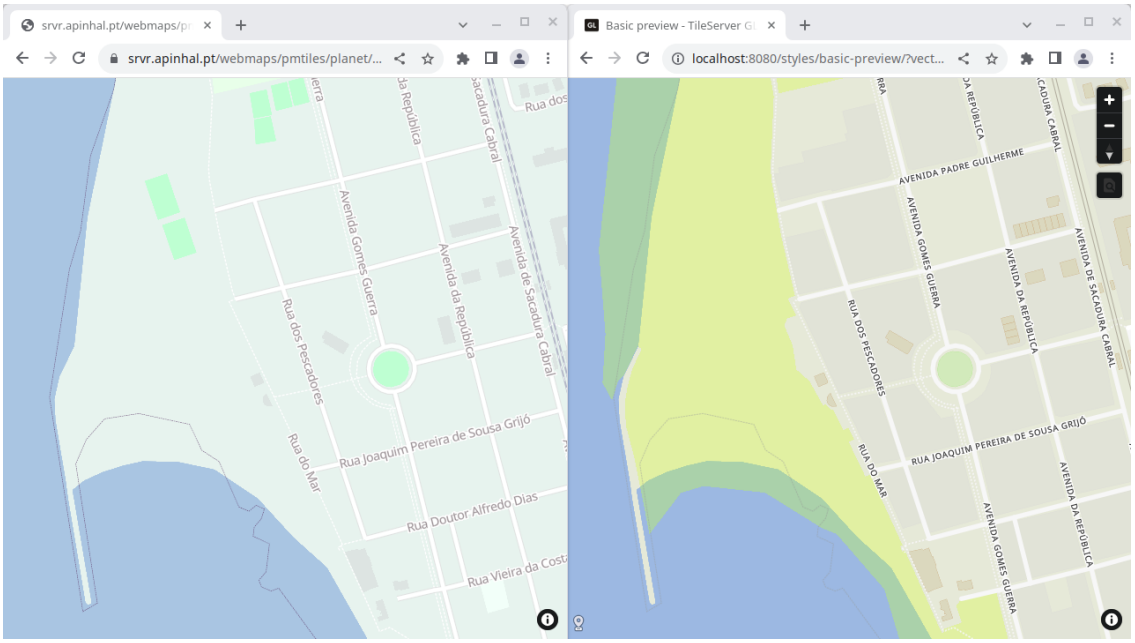


Figure 6.2: Path end.

Every tile requested during the servers test, Figure 6.3 are at zoom level 15, the maximum tile resolution available.

The average time request to fetch a tile at local server is bellow 100 ms and for the public server (Nginx serverless) is below 500 ms.

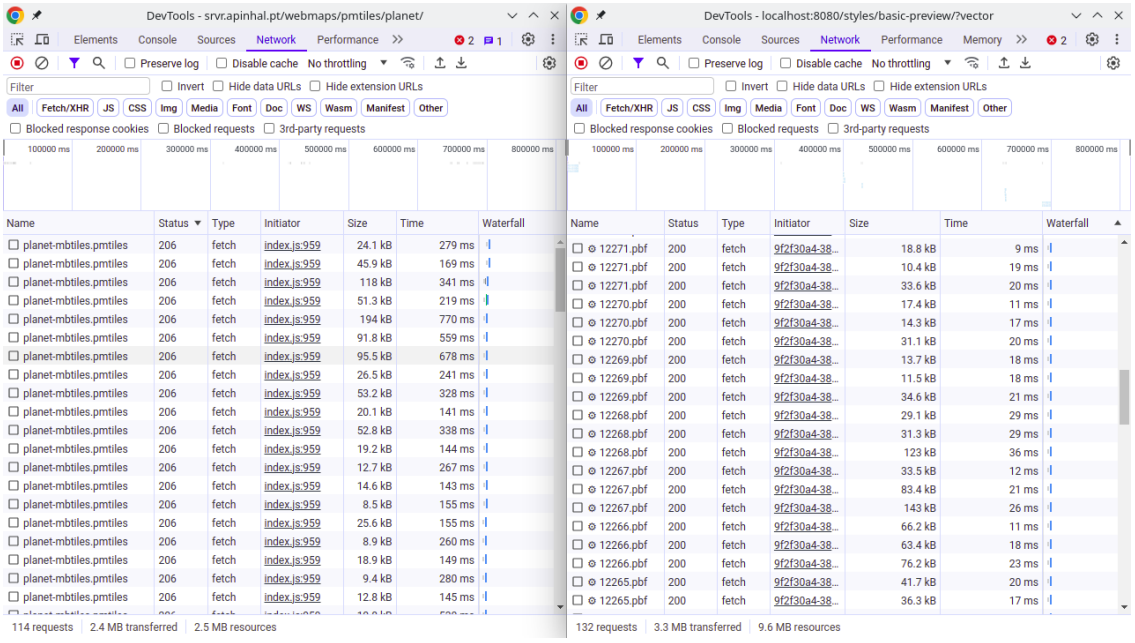


Figure 6.3: Servers response times of Nginx (serverless) on the left and TileServer GL on the right, delivering multiple MVT.

6.4.2 Geospatial Data Integrity

The first tile at the zoom level 0 was requested from Martin to test the server response and the resulting geospatial data, using the following URL:

- `http://localhost/martin/planet-mbtiles/0/0/0`

This URL uses the Reverse Proxy configured in Nginx, at Linting 5.7, line 14.

At zoom level 0 there is only one tile available and Figure 6.4 shows the request analysis using the Chrome web browser DevTools.

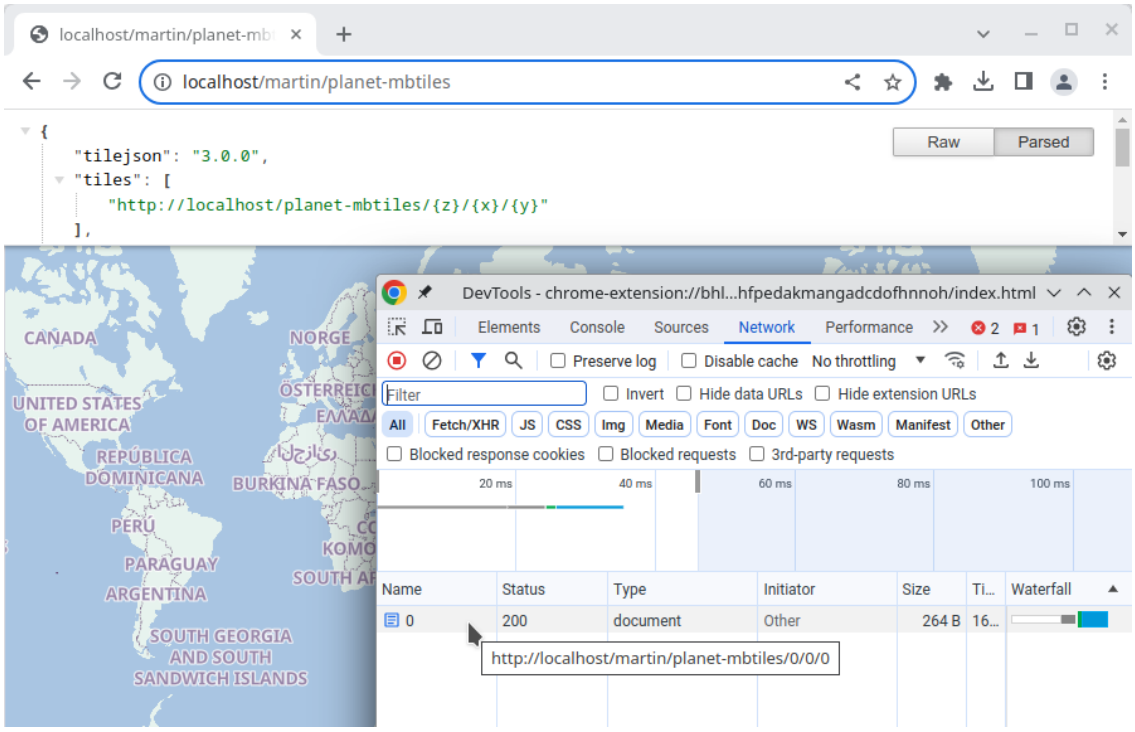


Figure 6.4: Analysing the lowest zoom tile (zoom level 0) requested from Martin serving the Planet MBTile.

The result is a 66 kB MVT in PBF format, composed by five layers verified with the `ogrinfo` terminal command in the next Listing 6.1.

```
1 $ ogrinfo -ro planet-0-0-0.pbf
2
3 INFO: Open of 'planet-0-0-0.pbf'
4       using driver 'MVT' successful.
5 1: water (Multi Polygon)
6 2: boundary (Multi Line String)
7 3: landcover (Multi Polygon)
8 4: place (Multi Point)
9 5: water_name (Multi Point)
```

Listing 6.1: Using `ogrinfo` to verify a tile (MVT) served from Martin.

Opening the file in QGIS, in Figure 6.5, confirms the five layers identified using `ogrinfo` but allows a more interactive geospatial data analysis.

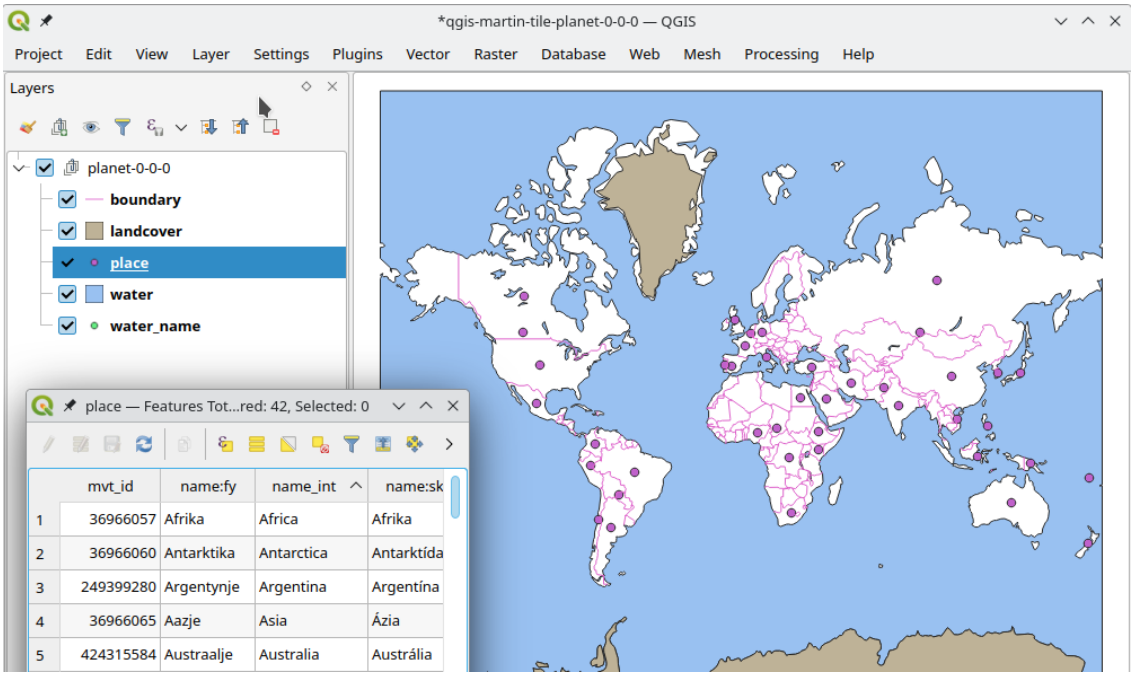


Figure 6.5: Zoom 0 Planet tile served from Martin and displayed in QGIS.

For example, using the layer **place** selected in Figure 6.5, it is possible to check the 42 point features and their attributes, such as the names used in the toponymic map labeling.

Chapter 7

Conclusions

The work developed throughout this academic year culminates in this document. There is a mixing of expectations versus the reality. Initially the Design 4 and Evaluation 6 chapters were written following a different direction and ended up being rewritten or added more information to make sense.

The same expectation applies to the resulting work being delivered, developing a new approach in a company with a solution already in production. This started with validating that at the time of writing this document there is no workaround, the company's production approach using a TMS/ZXY tile server is the way to go. At the beginning I realized that I could bring value by having specialized knowledge in geospatial data acquisition and cartographic production. Map servers for the web were not a new topic, but serving global geospatial data or base maps to the masses, with performance, where there is no time to wait, was new. I was facing the possibility of not being able to bring significant innovation to the project at hand, given the company current solution.

The purpose of this work was to create a centralized and scalable solution for publishing base maps using a set of geographic regions, deployable on-premises or in the cloud.

To produce an open map service, open data is needed and at the street level/resolution there is only OSM data. The important part of using this data is to give credit to its source and this is normally done by making the license attribution available on the map.

All processing involved dealing with geospatial databases and covers most of the web development spectrum, from back-end to front-end. The front-end development despite being small was inevitable because at the end of the day the best way to measure the service performance is by monitoring response time and if the request was fetched successfully. The back-end brings the innovation by introducing Martin server with the ability to use Composite Sources and adding the Planetlier to take care of geospatial databases. Another outcome is by using zoom level 14/15 is enough resolution to generate GIS database files to serve web maps for most uses.

Keeping 1 second mark as a response time reference, Section 6.4, the proposed solution with Martin met the expectations and the performance was in-line with the TileServer GL currently used in production.

The offer for Tile Map Services either open or proprietary is very wide and constantly evolving. Other options could easily bring an equally good answer, however this study focused on a solution, as the approach presented with Martin was the most interesting during the project.

In the end, I was very grateful to have had the opportunity to study and see the geospatial field from another perspective.

Bibliography

Almeida, Jussara M, Virgilio Almeida, and David J Yates (1997). "Measuring the behavior of a world-wide web server". In: *High Performance Networking VII: IFIP TC6 Seventh International Conference on High Performance Networks (HPN 97), 28th April–2nd May 1997, White Plains, New York, USA*. Springer, pp. 57–72 (cited on page 23).

Barry, Michael (2023). *GitHub - Planetiler*. url: https://github.com/onthegomap/planetiler (visited on 10/14/2023) (cited on page 31).

Brink, Linda van den et al. (2019). "Best practices for publishing, retrieving, and using spatial data on the web". In: *Semantic Web* 10.1, pp. 95–114 (cited on page 3).

Bulajic, Aleksandar, Milan Stamatovic, and Slobodan Cvetanovic (2012). "The importance of defining the hypothesis in scientific research". In: *International Journal of Education Administration and Policy Studies* 4.8, pp. 170–176 (cited on page 49).

Cloudflare (2023). *What is a content delivery network on Azure?* url: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/ (visited on 10/14/2023) (cited on page 17).

Doyle, Allan (1997). "Www mapping framework". In: *Open GIS Consortium* (cited on page 4).

Esri, E (1998). "Shapefile technical description". In: *An ESRI white paper* (cited on page 6).

Fairhurst, Richard (2023). *GitHub - Tilemaker*. url: https://github.com/systemed/tilemaker (visited on 10/14/2023) (cited on page 1).

Fischer, Erica (2023). *GitHub - tippecanoe - Felt*. url: https://github.com/felt/tippecanoe (visited on 10/14/2023) (cited on page 47).

Grady, Robert B (1992). *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc. (cited on page 29).

Haklay, Mordechai and Patrick Weber (2008). "Openstreetmap: User-generated street maps". In: *IEEE Pervasive computing* 7.4, pp. 12–18 (cited on page 12).

ISO/IEC/IEEE (2011). *Systems and software engineeringLife cycle processesRequirements engineering* (cited on pages 27, 29).

Joan Masó, Keith Pomakis and Núria Julià (2010). "OpenGISőWeb Map Tile Service Implementation Standard Version 1.0.0". In: (cited on page 4).

Koen, Peter A et al. (2002). "Fuzzy front end: effective methods, tools, and techniques". In: *The PDMA toolbook 1 for new product development* (cited on page 19).

Kurian, George (2013). *The AMA dictionary of business and management*. Amacom (cited on page 26).

La Beaujardiere, Jeff de (2006). "OpenGISő Web Map Server Implementation Specification. Version 1.3.0." In: (cited on page 4).

MacWright, Tom (2013). *The difference between XYZ and TMS tiles and how to convert between them*. url: https://gist.github.com/tmcw/4954720 (visited on 10/14/2023) (cited on page 4).

Mallon, Melissa (2015). "Data visualization". In: *Public Services Quarterly* 11.3, pp. 183–192 (cited on page 12).

Mapbox (2023). *Mapbox - TileJSON*. url: https://docs.mapbox.com/help/glossary/tilejson/ (visited on 10/14/2023) (cited on page 18).

MapLibre (2023a). *GitHub - MapLibre GL*. url: https://github.com/maplibre/maplibre-gl-js (visited on 10/14/2023) (cited on page 45).

– (2023b). *GitHub - Martin*. url: https://github.com/maplibre/martin (visited on 10/14/2023) (cited on page 31).

MapTiler (2022). *MapTiler Server and TileServer GL compared*. url: https://documentation.maptiler.com/hc/en-us/articles/8358453119249-MapTiler-Server-and-TileServer-GL-compared (visited on 10/14/2023) (cited on page 15).

– (2023a). *GitHub - MapTiler TileServer GL*. url: https://github.com/maptiler/tileserver-gl (visited on 10/14/2023) (cited on pages 1, 31).

– (2023b). *Tiles à la Google Maps*. url: https://www.maptiler.com/google-maps-coordinates-tile-bounds-projection/ (visited on 10/14/2023) (cited on pages 4, 8–12).

Maputnik (2023). *GitHub - OSM Liberty Style - Maputnik*. url: https://github.com/maputnik/osm-liberty (visited on 10/14/2023) (cited on page 43).

Miell, Ian and Aidan Sayers (2019). *Docker in practice*. Simon and Schuster (cited on page 16).

Mital, Anil et al. (2014). *Product development: a structured approach to consumer product development, design, and manufacture*. Elsevier (cited on page 20).

Netek, Rostislav et al. (2020). "Performance testing on vector vs. raster map tilescomparative study on load metrics". In: *ISPRS International Journal of Geo-Information* 9.2, p. 101 (cited on pages 15, 25, 29).

Nginx (2016). *Smart and Efficient Byte-Range Caching with NGINX & NGINX Plus*. url: https://www.nginx.com/blog/smart-efficient-byte-range-caching-nginx/ (visited on 10/14/2023) (cited on pages 16, 17).

– (2023). *What Is a Reverse Proxy Server?* url: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/ (visited on 10/14/2023) (cited on page 17).

Olaya, Víctor (2020). *Sistemas de Información Geográfica*. Lulu. isbn: 978-1-71677-766-0. url: https://github.com/volaya/libro-sig/releases/download/v3.0/Sistemas.de.Informacion.Geografica.pd (visited on 10/14/2023) (cited on pages 3–5, 7).

OpenStreetMap (2023a). *OpenStreetMap - Copyright and License*. url: https://www.openstreetmap.org/copyright/ (visited on 10/14/2023) (cited on page 13).

– (2023b). *OpenStreetMap - Formats*. url: https://wiki.openstreetmap.org/wiki/OSM_file_formats (visited on 10/14/2023) (cited on page 14).

OSGeo (2006). *Tile Map Service Specification 1.0*. url: https://wiki.osgeo.org/wiki/Tile_Map_Service_Specification (visited on 10/14/2023) (cited on page 4).

Overvoorde, Alexander (2023). *GitHub - openstreetmap-tile-server*. url: https://github.com/Overv/openstreetmap-tile-server (visited on 10/14/2023) (cited on page 1).

Protomaps (2023). *GitHub - PMTiles - Protomaps*. url: https://github.com/protomaps/PMTiles (visited on 10/14/2023) (cited on pages 40, 45).

QGIS (2023). *GitHub - QGIS*. url: https://github.com/qgis/QGIS (visited on 10/14/2023) (cited on page 31).

Scharl, Arno and Klaus Tochtermann (2009). *The geospatial web: how geobrowsers, social software and the Web 2.0 are shaping the network society*. Springer Science & Business Media (cited on page 22).

Stefanakis, Emmanuel (2017). "Web mercator and raster tile maps: two cornerstones of online map service providers". In: *Geomatica* 71.2, pp. 100–109 (cited on page 10).

Stonebraker, Michael and Lawrence A Rowe (1986). "The design of Postgres". In: *ACM Sigmod Record* 15.2, pp. 340–355 (cited on page 13).

Stothers, Jessica AM and Andrew Nguyen (2020). "Can Neo4j replace PostgreSQL in healthcare?" In: *AMIA Summits on Translational Science Proceedings* 2020, p. 646 (cited on page 24).

Torell, Wendy and Victor Avelar (2004). "Mean time between failure: Explanation and standards". In: *white paper* 78, pp. 6–7 (cited on page 50).

Vaidya, Omkarprasad S and Sushil Kumar (2006). "Analytic hierarchy process: An overview of applications". In: *European Journal of operational research* 169.1, pp. 1–29 (cited on page 20).

W3C (2016). *Spatial Data on the Web Best Practices*. url: https://www.w3.org/TR/sdw-bp/ (visited on 10/14/2023) (cited on pages 3, 4).

Wikimedia (2006). *Wikimedia Commons - Bitmap_VS_SVG.svg*. url: https://commons.wikimedia.org/wiki/File:Bitmap_VS_SVG.svg (visited on 10/14/2023) (cited on page 5).

– (2007). *Wikimedia Commons - Simple_vector_map.svg*. url: https://commons.wikimedia.org/wiki/File:Simple_vector_map.svg (visited on 10/14/2023) (cited on page 5).

– (2010). *Wikimedia Commons - Latitude_and_longitude_graticule_on_a_sphere.svg*. url: https://commons.wikimedia.org/wiki/File:Latitude_and_longitude_graticule_on_a_sphere.svg (visited on 10/14/2023) (cited on page 8).

Zeidan, Faisal and Mirza Murtaza (1999). "The analytic hierarchy process and value analysis applied to a petrochemical process". In: *Cost Engineering* 41.5, p. 34 (cited on page 19).