University of Minho
Department of Informatics

Master in Informatics Engineering

Abstract Map Representation for Web Publishing

Tiago Correia

Dissertation for the degree of
Master of Informatics Engineering

Supervised by:
Prof. Dr. Jorge Gustavo Rocha

Braga, November 2, 2010

A language that doesn't affect the way you think about programming, is not worth knowing.

Alan Perlis

Resumo

O desenvolvimento de um Sistema de Informação Geográfica (SIG) é dispendioso e consome muito tempo. Assim, neste projecto propõe-se uma solução para diminuir os custos e permitir que qualquer utilizador com poucos conhecimentos de informação geográfica, possa construir a sua própria aplicação Web SIG.
A aplicação Web SIG é dividida em três camadas. O armazenamento da informação geográfica (IG), a camada intermédia, entre a base de dados e a apresentação, que contém todas a operações de negócio, e a camada de apresentação Web. Em todas as camadas, existem vários desafios, uma vez que estamos a lidar com informação geográfica, e com centenas de mapas a mostrar.
O objectivo final do projecto é gerar todos os controladores e interfaces necessárias para apresentar a aplicação Web SIG, a partir de uma especificação de uma linguagem de domínio específico. A linguagem é baseada em XML e descreve os principais componentes da aplicação, os mapas a serem apresentados e um conjunto de widgets para analisar e manipular os dados.
Uma framework SIG designada MapFish Studio foi alterada neste projecto, a fim de processar a linguagem baseada em XML e criar uma aplicação Web SIG baseada na especificação da linguagem. Além disso, foi desenvolvido um plugin QGIS que permite gerar a linguagem XML apartir de uma interface gráfica.

Palavras-chave: *Sistemas de Informação Geográfica (SIG), Linguagem de Domínio Específico, Base de Dados Espaciais, Web Publishing, Framework, MapFish, Plugin.*

Abstract

The development of a Geographic Information System (GIS) is costly and consumes a lot of time. Therefore, in this project it is proposed a solution to decrease costs and to allow anyone, with few knowledges of geographic information, to build their own Web GIS application.
A Web GIS application is divided into three tiers. The geographic information (GI) storage, the middle tier, between store and presentation, which is the business tier with all logic operations, and the Web presentation tier. In all tiers, several challenges are presented, since we are dealing with geographic information, and with hundreds of map layers to show.
The final goal of this project is to generate all controllers and interfaces, necessary to present the Web GIS application, from a specification in a domain specific language. The language is XML based and describes the main application components, the layers to be shown and a set of widgets to analyze and manipulate data.
A GIS framework named MapFish Studio was modified in this project, in order to process the XML based language and create a Web GIS application based on the abstract map representation XML. Furthermore, it was developed a QGIS plugin that allows the generation of the XML based language through a user-friendly graphical interface.

Key Words: *Geographic Information System (GIS), Domain Specific Language, Spatial Databases, Web Publishing, Framework, MapFish, Plugin.*

Acknowledgments

I am grateful to many people for help, both direct and indirect, in developing this project and writing this master theses.

I would like to thank to Prof. Dr. Jorge Gustavo Rocha for his dedication and attendance to this project, and for reviewing this dissertation. Also, for his encouragement to participate in conferences related to GIS.

To my friends that have helped me from the beginning of my journey and gave me all the academic support during the development of this project, here is my thanks, especially for my friends (alphabetical order) Fabio, Liliana, Miguel and Ricardo. Also, an individual thanks to Miguel for reviewing the entire text of the dissertation.

A special thanks to my family: to my sister Carolina and my brother Nuno for the friendship and support they have given me throughout my life, and my parents Goretti and Joao, who always struggle to give the best to all their family.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The progress on land surveying, aerial photography and remote sensing lead to a more demanding geographic information storage and dissemination. So it was important to have an Information System (IS) capable of meeting those needs. The collection, processing, storage and/or dissemination of information are the main objectives of an IS. Since the IS needed to work with geographic information, it was created a new category of IS designated Geographic Information System (`GIS`).

`GIS` is a special class of IS that keep track not only of events, activities, and things, but also the location of those phenomenons [LGMR05]. Since the first `GIS` developed in the mid-1960s which is the Canada Geographic Information System (CGIS), the `GIS` technology has evolved, influencing deeply the capabilities of geographic analysis, and in retrospect, it marked a turning point towards the strengthening of geography as an explicitly spatial discipline [Dra04]. `GIS` is now widely accepted as powerful and integrated tools for storing, manipulating, visualizing, disseminating and analyzing spatial data [Dra04, CGR+08].

Until recently the traditional atlas has been the main tool for collection, storage and dissemination of geographical knowledge about the earth [PSM01]. But the constant change of spatial data increased the need to create a system capable of producing maps with those changes by a very low cost. Therefore the development of `GIS` technology overcome the constraints of the traditional atlas.

The evolution of `GIS` technology has resulted in lower costs and improvements of its hardware and software components. These developments made so far led to the emergence of `GIS` applications that operate in areas like real estate, public health, crime mapping, national defense, sustainable development, natural resources, transportation, logistics,

etc.

Despite the considerable evolution that GIS had in recent years, its development costs remain high. Those costs includes training of staff and users, equipment (hardware and software), and other costs (services, data).

1.2 Problem

Nowadays, GIS has become indispensable to most of the businesses and organizations. Those corporations need a GIS to solve their problems. However, developing a GIS is very costly and cannot be afforded by most of the business and organizations.

The majority of those companies just need a system capable of resolving simple issues. Therefore, the GIS must have the capability of providing quick and easy access to large volumes of data and should be able to manipulate that spatial data. The difficulty is that the development process of a GIS is not automated. So, whenever a user or a company wants to publish Geographic Information (GI) on the Web it is necessary to develop a new GIS.

1.3 Objectives

The goal of this project is to design a solution to publish hundreds of map layers on the web easily. The process of publishing GI on the Web has to be simplified and automated, in order to ease the publication of spatial data in Web.

The system to be developed should allow users with few knowledges of GI to create their own Web GIS application. Also, any modification performed in the GIS must not imply a development of a new system. Finally, whenever the user wants to change some features of the application (layers, layout, etc), that must be done without much effort.

The resulting Web GIS application could be customized for each problem, accordingly to the user's needs. It is possible to add tools to the Web GIS application to analyze and manipulate spatial data.

1.4 Document Outline

In Chapter 2 it is given a definition of GIS and are defined some concepts of GI that are important to understand some GIS definitions. The same chapter describes the various architectures of GIS, specifying the Web and Desktop GIS platforms and technologies.

In the Chapter 3 it is shown the architecture of the project, a generic structure of the tools developed and the basic concepts of the `XML` based language.

The development of the domain specific language and its tools (generator and processor) are described in Chapter 4.

It is given in Chapter 5 two examples of a Web `GIS` application description in the language developed, and the results of using the language generator and language processor tools.

Finally, a conclusion of this thesis and a brief description of the future work that will be done is shown in Chapter 6.

Chapter 2

Geographic Information System

2.1 Definition

GIS is considered a special class of IS created to model aspects of the real world (located in a geographical area), often with different degrees of dependency and related issues arising in the field of human activity. These systems consist in integration of hardware, software, geographic data, procedures and people, enabling users to capture, store, retrieve, manipulate, analyze and visualize all kinds of information distributed geographically (spatial data).

2.1.1 GIS Components

A GIS is specified by six well-defined components that can be seen in the Figure 2.1.

These components are connected with each other by network component. The network is the most important component of GIS because without it no rapid communication or sharing of digital information could occur, except between a small group of people crowded around a computer monitor [LGMR05]. Furthermore, the Internet made possible the expansion and industrialization of GIS. The software and hardware of a user are two other components needed to make possible the interaction between the client (user) and the server. The hardware can be an office desktop, a laptop or a mobile device, that uses a web browser (Thin client), a desktop software (Thick client) bought from a GIS vendor, or even a mobile device software like the GPS software made for pda's. Database is another component, which consists of a digital representation of selected aspects of some specific area of the Earth's surface or near-surface, built to support a scientific purpose [LGMR05]. A GIS also requires management, so it is necessary to establish procedures, lines of reporting, control points, and other mechanisms for ensuring that

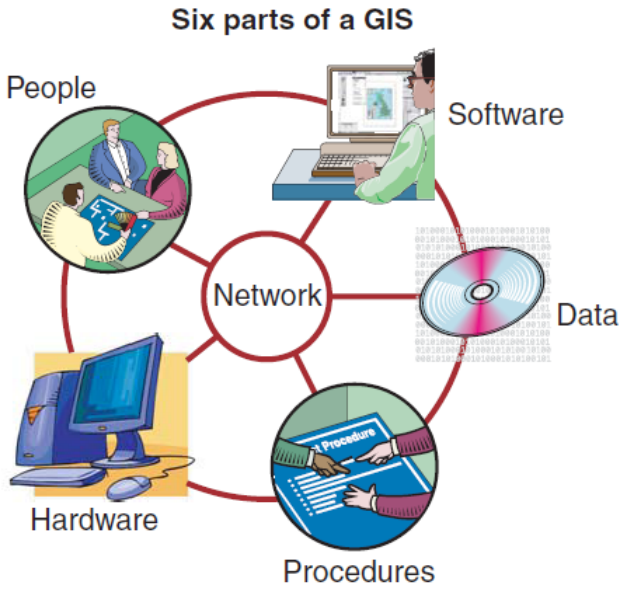

Figure 2.1: The six components parts of a GIS
[LGMR05]

GIS activities stay within budgets, maintain high quality, and generally meet the needs of the organization [LGMR05]. Ultimately, a GIS is useless without the people who design, program, and maintain it, supply it with data, and interpret its results [LGMR05].

2.2 Geographic Information

Geographic information is defined as one or more sets of data processed and organized, which show the location and shape of geographic features and may also include other attributes that characterize these same elements. Those geographic objects or entities are referenced by their location (latitude and longitude coordinates).

The growing importance of such information should be greatly to the development of Decision Support Systems, as well as widespread access to information by citizens, making the essential nature of spatial data.

Represent geographic information is not an easy task because the geographic world is infinitely complex, but computer systems are limited, therefore the representations must somehow limit the amount of detail captured [LGMR05]. The GIS separates the information into different map layers and store them independently, allowing the user to work with them quickly and easily. That way the user can manipulate the available information, the position and topology of objects in order to generate new information.

In the sub-section 2.2.1 it is described two forms of spatial data representation (vector

and raster data). The standards developed by OGC in order to achieve a standardization for geospatial content and services are defined in the sub-section 2.2.2.

2.2.1 Vector and Raster Data

Vector and raster data (spatial data) are the two common and extensively used GIS data formats in GISs [CGR+08]. In a raster representation space is divided into an array of rectangular (often square) cells where all geographic variation is then expressed by assigning properties or attributes to these cells (sometimes called pixels) [LGMR05]. In a vector representation, the basic units of spatial information are points, lines and polygons where lines are captured as points connected by precise straight lines and polygons are captured as a series of points or vertices connected by straight lines [LGMR05]. The raster data can be stored in many image formats like PNG, TIFF, JPEG, etc. The common format used to represent vector data is shapefile (.shp) which was developed by Environmental Systems Research Institute (ESRI). The Figure 2.2 shows the differences between vector and raster data in an real world example.

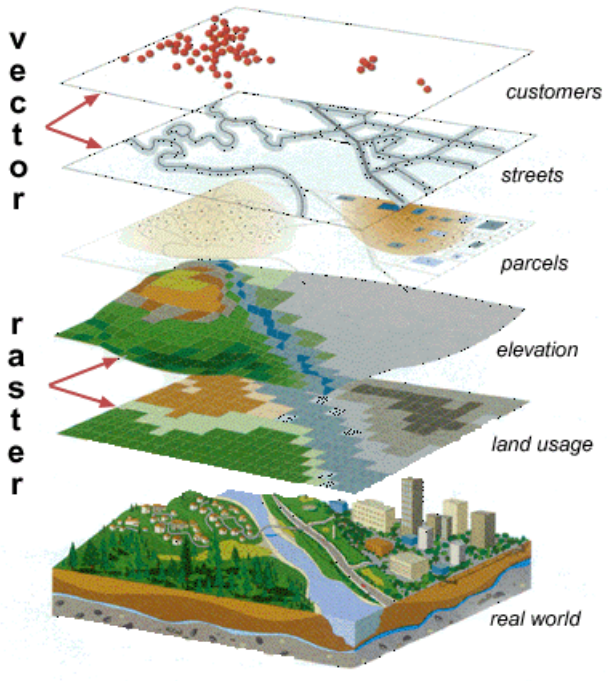

Figure 2.2: Vector and Raster Data
[CGR+08]

2.2.2 OGC Standards

In an attempt to arrive at a standardization for geospatial content and services, `GIS` data processing and data sharing it was created an international voluntary consensus standards organization named Open Geospatial Consortium (OGC), also designated OpenGIS. The most relevant of the 28 OGC standards developed are: Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS), Simple Features (SFS).

2.2.2.1 Simple Features (SFS)

The OpenGIS SFS provides a well-defined and common mechanism for applications to store and access feature data in relational or object-relational databases, so that the data can be used to support other applications through a common model, data store and information access interface. OpenGIS Simple Features are geospatial features described using vector data elements such as points, lines and polygons [ZZL+09]. These elements are in Well-Known Text (WKT) representation that are used to represent geometry objects in queries. Internally, the database stores geometry values in a format that is not identical to WKT format. An example of geometry objects represented in WTK format and its geometry values stored in a database can be seen in Figure 2.3. In order to convert a string in WTK format to a geometry type format (to be inserted in the database) it can be used the function *ST_ GeometryFromText(text WKT, SRID)*. The parameter *text* can be a point, linestring, polygon, etc, as shown in the left side of the equation on the Figure 2.3. The *SRID (Spatial Referencing System Identifier)* is a id (integer) that identifies the coordinate systems used to define the geometry columns. The right side of the equation on the Figure 2.3 is the result of the function *ST_ GeometryFromText(text WKT, SRID)* which is stored in the database on the geometry column.

POINT (10 10) = "01010000203801000000000000000024400..."
LINESTRING(10 10, 20 20, 30 40) = "0102000020380100000300..."
POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10)) = "010300002038010000010000..."

Figure 2.3: WTK format example

2.2.2.2 Web Map Service (WMS)

WMS is a OpenGIS standard for requesting a map from a GIS data store (vector or raster data), using a map server, and return an image according to the user's needs. It can also

supply meta data about the available layers, server capabilities, and contact/publisher information.

2.2.2.3 Web Feature Service (WFS)

WFS communicates geographic `feature` information. It was specified to allow the geographic `feature` data in maps to be queried, updated, created, or deleted by the user. This service uses `XML`-based GML format for `feature` data exchange between server and client. Data manipulation operations include the ability to create, delete or update a `feature` instance, and retrieve or query `feature`s based on spatial and non-spatial constraints.

2.2.2.4 Web Coverage Service (WCS)

The OpenGIS WCS defines a protocol-independent language for the extraction, processing, and analysis of multi-dimensional gridded coverages, representing sensor, image, or statistics data. The services that implement this language provide access to original or derived sets of geospatial coverage information, in useful ways for client-side rendering, input into scientific models, and other client applications [Bau09].

2.3 Architectures

Typically, a web based `GIS` is a client-server application, therefore its architecture is the same as the others web applications. A generic web application architecture is shown in the Figure 2.4.

The Figure 2.4 shows the three main components of a web application. The user interface component represents the client-side of the application, where the user (client) interacts with the system (server) using a specific platform. Database component contains the data of the system. The application is deployed to the application server component where it can guarantee security, performance, transaction support, data and code integrity. The application server connects to the user through HTTP using a web server to perform operations like request or response, and connects with the database via ODBC and using SQL, ensuring the full integration of the components.

The architecture of a `GIS` has the same principles of a web application, in other words, it is a standard 3-tier architecture where user interface, middleware (Web application server and `GIS` server), and database components are modularized [CGR+08]. Alongside with these components, the architecture also designs a file system for storing data files

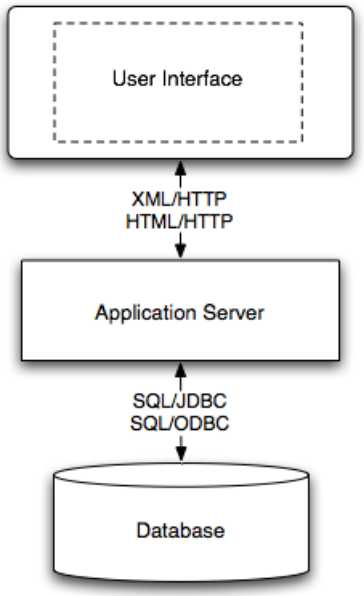

Figure 2.4: Generic Web Application Architecture

that can not be in database and web services that can be accessed over the World Wide Web (this component is optional). The generic GIS architecture is shown in Figure 2.5 which presents a web server not displayed in Figure 2.4.

As shown in Figure 2.5 a GIS application has different types of user interfaces: web browsers, desktop software and mobile devices.

The database contains spatial and non-spatial data [CGR+08]. The spatial DBMS (Database Management System) that was used in the development of this project is the PostGIS, which is a powerful open source DBMS that has proven reliability and respect, and is a OGC compliant [UCF+05]. PostGIS is an extension of the object-relational database PostgresSQL that supports geographic objects. PostGIS only supports vector data, so the raster data has to be stored in a file system.

Finally, the most important component is the GIS server which contains the application. This component provides visualization, spatial data analysis, mapping, and spatial data management services, and also supports complex workflow activities, including versioning [CGR+08, NOJGN99].

Map caches are used to help optimize performance of a GIS. A map cache is a collection of pre-rendered map tiles that can be used for displaying a map service. This allows a map service to quickly display maps because the map image does not have to be rendered on the fly, so the cost of rendering the image is paid only once when the cache is created.

The server map cache greatly improves performance because the map does not need

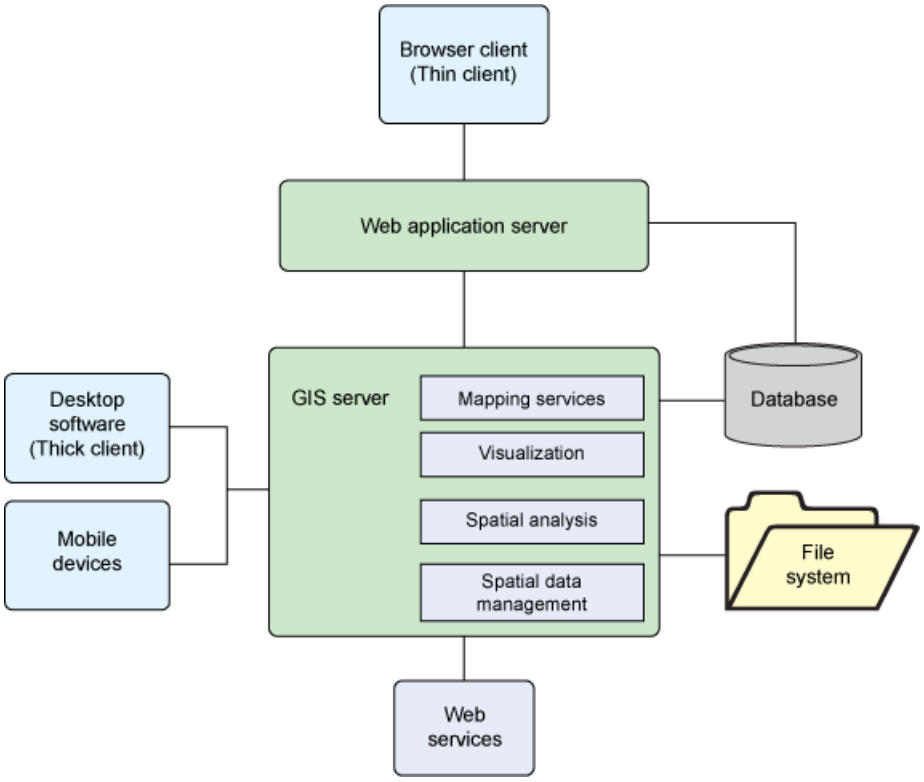

Figure 2.5: Generic GIS Architecture
[CGR+08]

to be computed each time the user requests a map to the server. Instead, the images are retrieved from the map cache for the requested map extent and resolution.

2.3.1 Desktop Model

A desktop GIS is a mapping software that is installed into and runs on a personal computer and allows users to display, query, update, and analyze data about geographic locations and the information linked to those locations [ESR10]. I.e., the software is not executed on a server and remotely accessed or controlled from or by a different computer.

This class of systems is used mainly in isolated projects, with the purpose of performing spatial analysis over regions of small and medium businesses. These systems emphasize the aspect of mapping and can also be characterized as project-oriented GIS.

This model has some characteristics such as:

- A GIS for every user
- A complete GIS, even when the user just wants some features
- Accessible only on the machine where it is installed

- Data is stored in the local machine
- The GIS operations speed depends on the capacity of the machine
- Learning difficulties

This model has proven inefficient for the reality of many GIS projects, which nowadays are often multi-platform and multi-user. Considering the technological advances, with increased rates of file transfer and web browsing responses more quickly and efficiently, it is easy to imagine that the GIS systems focused on the individual workstations, tend to be restricted to cases more specific and do not involve larger groups of users.

For those reasons this model was not chosen to apply onto the project. On the other hand, the Web GIS model has proven to be the most reliable approach to follow.

2.3.2 Web Model

Web GIS is a cheap and easy way of disseminating and manipulating spatial data. This model is the most used by many organizations that are interested on distributing maps and processing tools without time and location restrictions to users. Internet technology made possible the expansion of GIS systems, thus the Web GIS became widely used by many government organizations as well as numerous households [AHB02].

The architecture of a Web GIS is similar to the client/server typical three-tier architecture. A client typically is a Web browser (Firefox, Internet Explorer, Opera, etc). The server-side consists of a Web Server, Map Server, Web GIS software and Database (Figure 2.6) [AHB02].

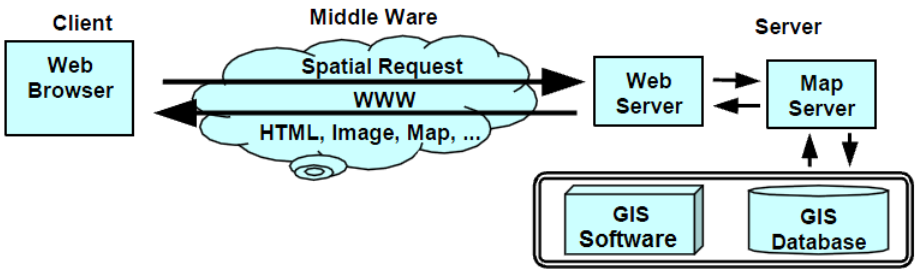

Figure 2.6: Generic Web GIS Architecture [AHB02]

Client/server architectures are widely used in enterprise applications. In this case the server provides an instance of a GIS running. The client has an interface. And the middleware provides the communication between the client and the GIS software [AHB02].

This model has the following characteristics:

- Multi-user GIS
- Component-based
- Users have access to data and its processing any time in any place with Internet access
- Data is stored in server and is accessed by a client user
- The GIS operations speed depends on the network bandwidth
- It is easier to learn, since nowadays users tend to have high experience with Web interfaces

The high cost of GIS system, the release of system specific databases, and the enormous software development efforts on upgrading the system are fading with the introduction of web-based GIS. Therefore, since the costs of developing a GIS is one of the motivations for this project, it was decided to use this type of model in the project [AHB02].

2.4 Technologies

There are many tools used in GIS, that can be either proprietary or open source software. The growth of open source technologies led to a wide acceptance of open source technology by the GIS community. Considering the purpose of this project, the open source technologies will be discussed in depth further in this document.

In the next two subsections (2.4.1 and 2.4.2) are discussed and analyzed some client-side and server-side open source technologies. Those technologies are the most accepted and used by the open source GIS community.

2.4.1 Client

In this section are approached some client-side open source technologies. Some of this technologies were used in this project, such as ExtJS and OpenLayers. The other two technologies illustrated (Geomajas and Chameleon) are very important to the GIS community. In the following sections those technologies are characterized.

2.4.1.1 ExtJS

ExtJS is a client-side JavaScript - or Ajax - Framework for building rich internet applications (RIAs). It is a large and comprehensive library, and also highly modular.

ExtJS library has all the general functionality that developers have come to expect from JavaScript libraries such as high-performing traversal of the DOM, manipulating CSS and HTML, simplified AJAX execution, handling of `XML` and JSON data, etc [OPKJ09].

This library includes:

- High performance, customizable UI widgets
- Well designed and extensible Component model
- An intuitive, easy to use `API`
- Commercial and Open Source licenses available

ExtJS library is focused on allowing developers to create great user interfaces in a web application, and has a collection of GUI-based form controls (or "widgets") for use within web applications as the following [Zam09]:

- Text, Date and Numeric fields
- List boxes and select boxes (combo)
- Option Button (Radio Button) and check boxes (box)
- HTML input areas
- List view
- Tree-structure
- Tabs display
- Menu Bar
- Dynamic space allocation of the controls
- Scroll Bars
- Adobe Flash Charts based on Adobe Flash

ExtJS is extremely powerful, providing, for the developers, windows (which can be dragged around, resized, maximized, and minimized) as well as tabbed interfaces, accordion widgets, grids, and more. ExtJS is designed to work alongside other well-playing frameworks. ExtJS has a good performance on small projects, and has excellent scalability for even the largest projects [Zam09].

2.4.1.2 OpenLayers

OpenLayers is a pure JavaScript library that makes it easy to put a dynamic map in any web page [YS07]. It can display map tiles and markers loaded from any source [YS07]. OpenLayers is completely free, Open Source JavaScript, released under a BSD-style License and initially developed by MetaCarta but made public to propagate the use of geographic information of all kinds [YS07].

OpenLayers has no server-side dependencies. OpenLayers implements a (still-developing) JavaScript `API` for building rich web-based geographic applications, similar to the Google Maps and Microsoft Virtual Earth `API`s, but unlike the others it is free.

Furthermore, OpenLayers can acess data provided through OGC WMS and WFS standards [WH09]. It is written in object-oriented JavaScript, using components from Prototype and Rico libraries.

As a framework, OpenLayers is intended to separate map tools from map data so that all the tools can operate on all the data sources. This separation breaks the proprietary frameworks that earlier GIS revolutions have taught civilization to avoid. The mapping revolution on the public Web should benefit from the experience of history.

The OpenLayers can be a useful tool to display maps in the project because it is free and is a JavaScript therefore it is a client-side technology which increases the performance of the system.

Figure 2.7: OpenLayers Interface
(Source: http://dev.openlayers.org/sandbox/ominiverdi/presentations/girona2008/)

An interface of a OpenLayers implementation map showing United States of America earthquakes in a determined time space is shown in the Figure 2.7.

In this figure it can be seen that OpenLayers JavaScript lib provide basic features such as *zoom*, *directional movements* and *layers customization*. Besides that, there can be seen

the base layers possible to use in OpenLayers like *Google Satelite*, *Google Hybrid*, *Satelite WMS (NASA)*, *OpenLayers WMS*, etc. In this specific map the only features available to be overlayed are the *Markers* and *Earthquakes (GeoRSS)*. In this Figure it is chosen the *OpenLayers WMS* base layer and the feature *Earthquakes (GeoRSS)* which shows the earthquakes occurrences in USA. OpenLayers has a very rich `API` that can be found in this website:
http://dev.openlayers.org/apidocs/files/OpenLayers-js.html

2.4.1.3 Geomajas

Geomajas is an innovative open source GIS application framework for building rich internet applications. It has sophisticated capabilities for displaying, managing and manipulating geospatial information. The modular architecture makes it easily extensible. The stateless client-server architecture guarantees a fully scalable Web GIS application [DG10].

The focus of Geomajas is to provide a platform for server-side integration of geospatial data, allowing multiple users to control, manage and manipulate the data with their own browsers. Using the Geomajas framework developers are able to build integrated GIS solutions for businesses and governments organizations. Basically, Geomajas framework provides a collection of powerful building blocks, from which the most complex GIS applications can easily be built. Key features include [DG10]:

- Modular architecture
- Clearly defined `API`
- Integrated client-server architecture
- Built-in security
- Advanced security
- Advanced geometry and attribute editing with validation
- Custom attribute definitions including object relations
- Advanced querying capabilities (searching, filters, style, ...)
- Integration of business logic and domain logic
- `GWT` face for the client side

- System configuration using Spring
- Services for rendering and feature handling
- Much detailed technical information

Geomajas is a fully open source web-based GIS framework with excellent performance and usability.

2.4.1.4 Chameleon

Chameleon is an open source environment for developing Web Mapping applications using the PHP programming language. It is integrated on MapServer (section 2.4.2.1) as the main mapping engine. Chameleon supports all MapServer data formats.

Chameleon has the ability to quickly build up new applications from a collection of widgets that can be placed in an HTML template file. It provides simplified access to many features, by providing pre-made scripts as re-usable components. Chameleon makes it possible for non-programmers to insert sophisticated mapping components into Web mapping applications. Chameleon is a collection of over 300 PHP scripts that provide access to mapping-related widgets and functions.

Chameleon is extendable, then developers can create their own custom widgets and then use them in mapping applications.

2.4.2 Server

In this section are approached some server-side open source technologies. Those technologies are MapServer (Section 2.4.2.1), GeoServer (Section 2.4.2.2), MapFish (Section 2.4.2.3) and PostGIS (Section 2.4.2.4).

2.4.2.1 MapServer

MapServer is an open source development environment for building spatially-enabled web mapping applications developed by the University of Minnesota, originally with NASA support. It renders GIS data sources into cartographic map products on-the-fly [Ram07]. MapServer is easily the most successful open source GIS project to date [Ram07]. It supports more input data sources than most proprietary products, has higher performance, and is simpler to install and set up [Ram07]. Its architecture is shown in Figure 2.8.

It creates map images from spatial information stored in digital format and can handle both vector and raster data [Kro05]. Also supports database formats like the PostgresSQL/PostGIS database described in the Chapter 3.

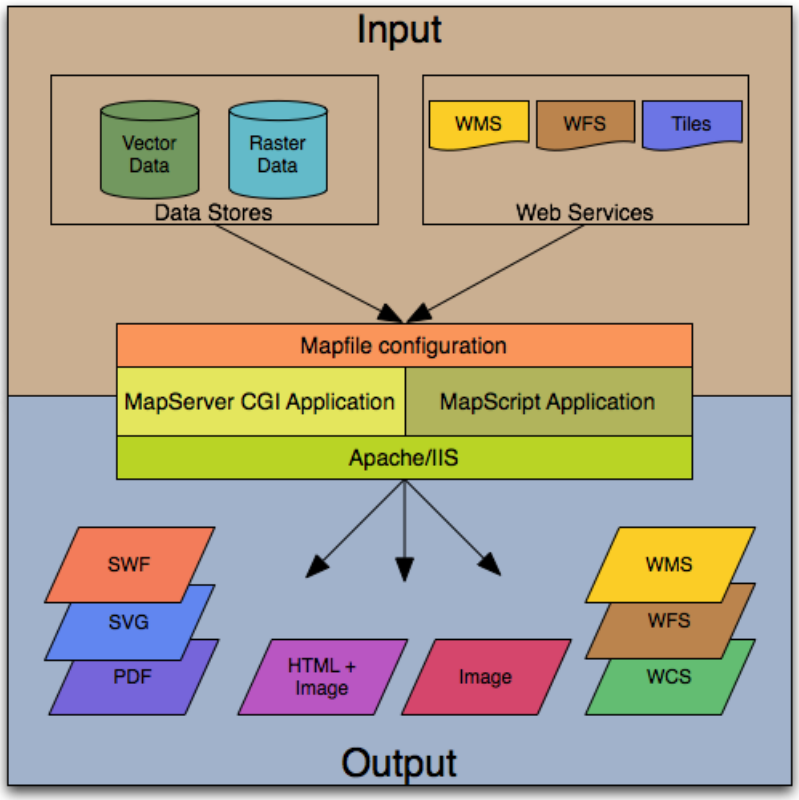

Figure 2.8: MapServer Architecture
(Source: Website of MapServer - http://mapserver.org/)

MapServer supports Open Geospatial Consortium (OGC) standards including WMS, WFS and WCS. These allow web clients to query and receive geographic information in the form of image, vector, or coverage data.

MapServer consists of three components: the map file, the template files and the CGI program [BM02].

Mapfile (.map file) is the most important component of the MapServer because it is the basic configuration file for data access and styling. It defines a collection of mapping objects that together determine the appearance and behavior of the map as displayed in the web browser [Kro05].

A mapfile is an ASCII text file where a number of different objects are defined hierarchically [Kro05]. Each object has a variety of parameters available for it. An important object is LAYER. A layer is the combination of data plus styling. Data, in the form of attributes plus geometry, is combined with a given styling using CLASS and STYLE directives.

The template files are a common HTML pages provided with MapServer specific parameters and variables [BM02]. The template files are the files that are used to create an

HTML page ready to present maps, cartographic objects, queries and all other information which the web GIS designer wants to offer to the user [BM02].

The simplest form of a MapServer runs as an executable CGI application (normally called mapserv.exe) on a web server [Mit05]. MapServer is considered an HTTP-based stateless process, which means that it processes a request and then stops running [Mit05]. When a request is sent to MapServer, it uses information passed in the request URL and in the Mapfile (configuration file of MapServer) to create an image of the requested map. The request may also return images for legends, scale bars, reference maps, and values passed as CGI variables. An example of a request URL of a map from MapServer is: *http://localhost/cgi-bin/mapserv.exe?mode=map&map=C:/maps/portugal.map*.

In order to present its results, MapServer needs to format the map and associated elements as a web page, because the program itself does not create the HTML, rather it scans an HTML template for substitution strings [Kro05]. The strings can be file references, details of map geometry, layer specifications, zoom factors, and also current values of CGI variables such as image size, mapfile names, map extent, etc [Kro05]. MapServer replaces the substitution strings with the appropriate values and returns the modified HTML [Kro05].

MapServer can return SWF, SVG or PDF files with the HTML, and guarantees the use of OGC standards in the resulting map.

2.4.2.2 GeoServer

GeoServer is an open source web mapping server written in Java, maintained by The Open Planning Project (main maintainer), that allows geospatial data to be served over the web. GeoServer uses the WMS, WFS and WCS protocols (OGC standards) for serving data in image and `XML` formats from a variety of sources that include shapefiles, raster data files (PNG, TIFF, JPEG, etc) and postGIS-enabled databases [PTM07].

Geoserver integrates diverse spatial data repositories with simplicity and high performance. The main objective of GeoServer is to simplify the use of open standards in order to enable anyone to quickly share their geospatial information.

The main features of Geoserver are:

- Fully compatible with the specifications WMS, WCS and WFS.
- Easy to use through web-based administration tool.
- Support PostGIS, Shapefile, ArcSDE and Oracle formats.
- VFP, MySQL, Cascading WFS and MapInfo formats are also supported.

- Output of the WMS as JPEG, GIF, PNG, SVG and GML files.
- Pictures with anti-aliasing filtering.
- Full support the `SLD`
- Support for database transactions through the atomic standard protocol WFS-T, available for all data formats.
- Based on Java servlets (JEE), can run on any servlet container.
- Designed for extensions.
- Easy to write new data formats with the data storage interface GeoTools.
- Designed for interoperability.

The GeoServer consists of a set of modules, each with a very specific function. The Figure 2.9 describes the components that define the architecture of GeoServer.

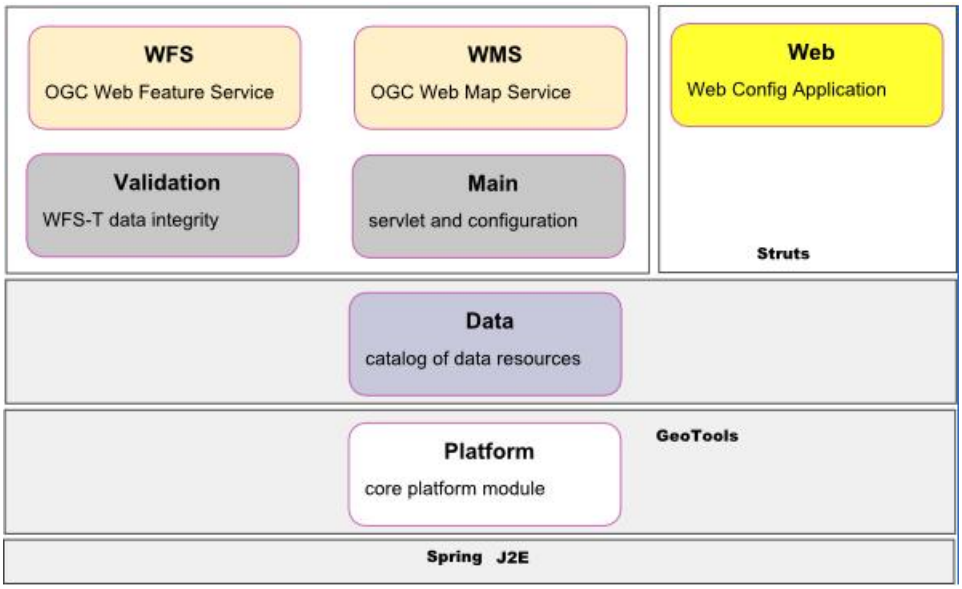

Figure 2.9: GeoServer Architecture
(Source: Website of GeoServer - http://geoserver.org/)

The module that manages the other components is called Spring. The platform on which applications are developed is the GeoTools. There is also a module responsible for the storage of the data itself. The set of standardized OGC services WMS and WFS are another of the modules of GeoServer. Finnally, it is provided an interface for interaction with users [PTM07].

2.4.2.3 MapFish

Mapfish is a open source framework for building Internet Rich Applications (RIAs). It is based on the `Pylons` Python framework. MapFish extends Pylons with geospatial-specific functionality.

MapFish is composed of two components: MapFish Client and MapFish Server. In the Figure 2.10 it is described a generic architecture of a mapfish application.

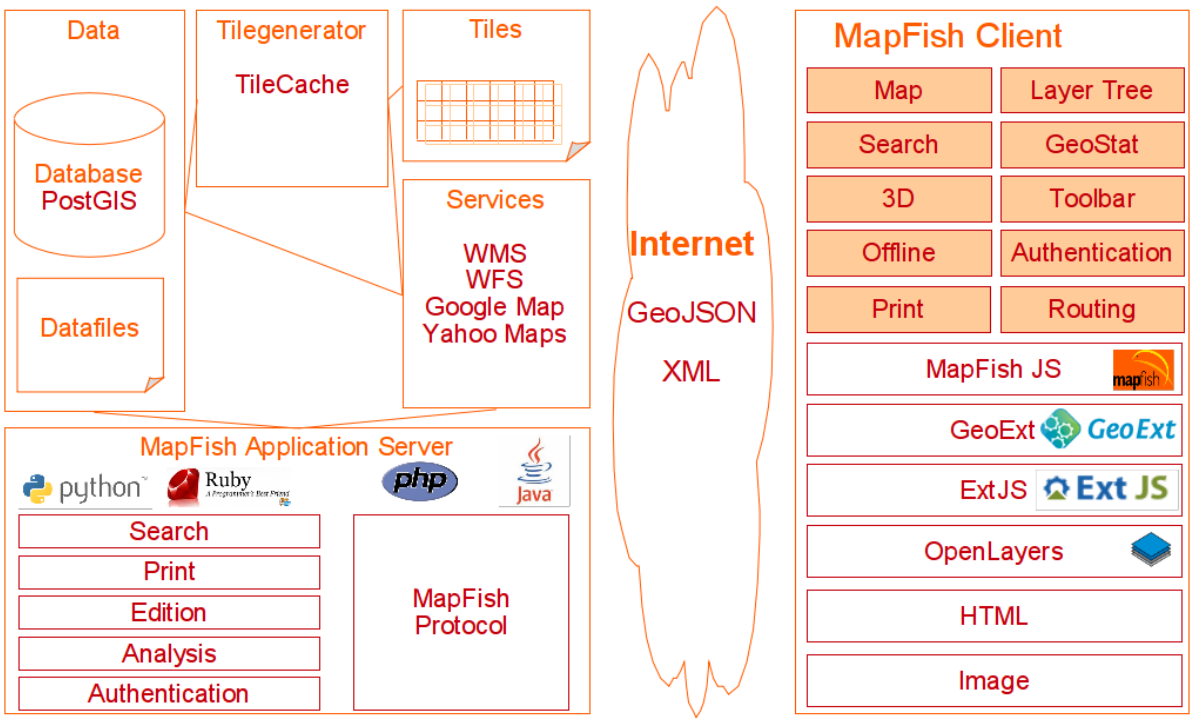

Figure 2.10: MapFish Applications Architecture
(Source: Website of MapFish - http://mapfish.org/)

On the MapFish Client, is provided a complete RIA-oriented JavaScript toolbox based on OpenLayers for the mapping part, on ExtJS and GeoExt for the GUI part. On the MapFish Server, several languages can be used, for instance Python, Java, Ruby and PHP.

MapFish is cartographic server (map server) agnostic and supports a range of mapping servers (should be able to communicate with open protocols like WMS or WFS) and web server languages.

MapFish is compliant with the Open Geospatial Consortium standards. This is achieved through OpenLayers or GeoExt supporting several OGC norms, like WMS, WFS, WMC, KML, GML etc.

2.4.2.4 PostGIS

PostGIS is a powerful open source DBMS that has proven reliable and respectful, and is a OGC compliant [UCF+05]. PostGIS is an extension of the object-relational database PostgresSQL that supports geographic objects. PostGIS only supports vector data, so the raster data need to be stored in a file system.

PostGIS includes a large number of functions for spatial/topology analysis that extends PostgreSQL itself [UCF+05]. PostGIS supports all the objects and functions specified in the OGC "Simple Features for SQL" specification (SFS) and has been certified as compliant with the "Types and Functions" profile [Ram09].

Chapter 3

Architecture

In this chapter it is described the architecture of the project. The first part of this chapter describes the architectural model and its components. The second part presents, in detail, all the components and the information flow between components.

The Figure 3.1 represents the model of the system architecture. The architecture consists of four essential components: a language generator, the language itself, a language processor and the resulting Web GIS application.

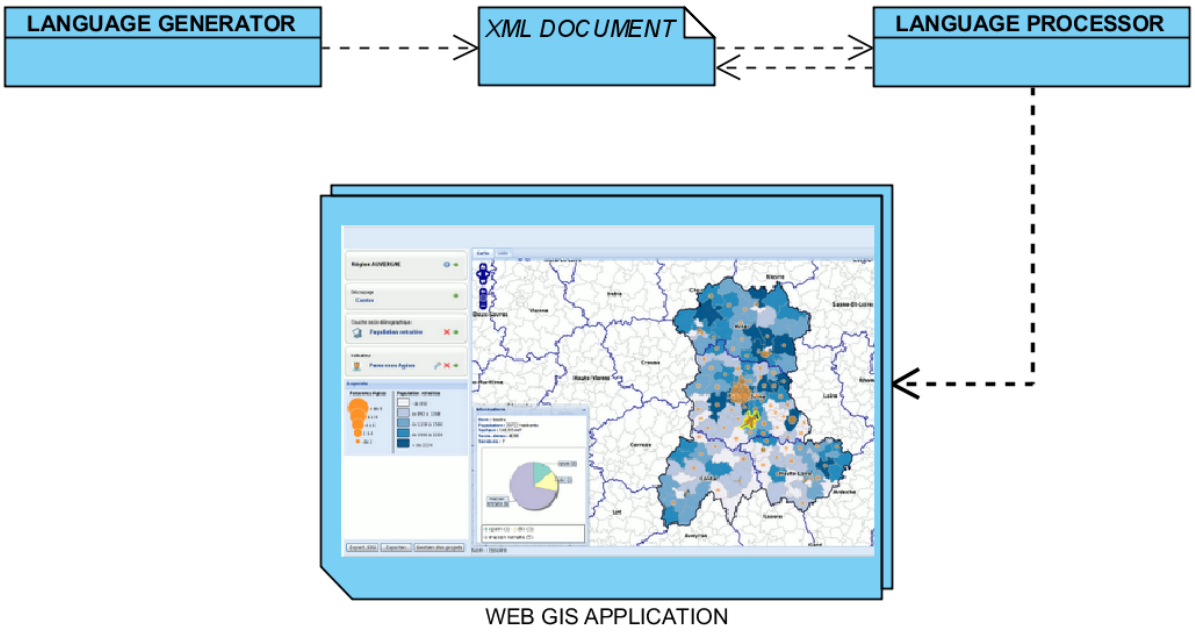

Figure 3.1: Project Architecture

The language generator is a desktop application that creates a `XML` document based on specified attributes. This `XML` document describes a Web GIS application. Then, the language processor (Web application) imports the `XML` document and creates a Web GIS

application. It is possible to do the inverse, i.e., generate a `XML` document of a Web GIS application through language processor. Thus, it is ensured the transferability of the language.

In the Figure 3.2 it is described an activity diagram designed in UML that represents the cycle of generating a Web GIS application. That generation process workflow is divided in three swimlanes (roles) which are *User*, *Language Generator* and *Language Processor*.

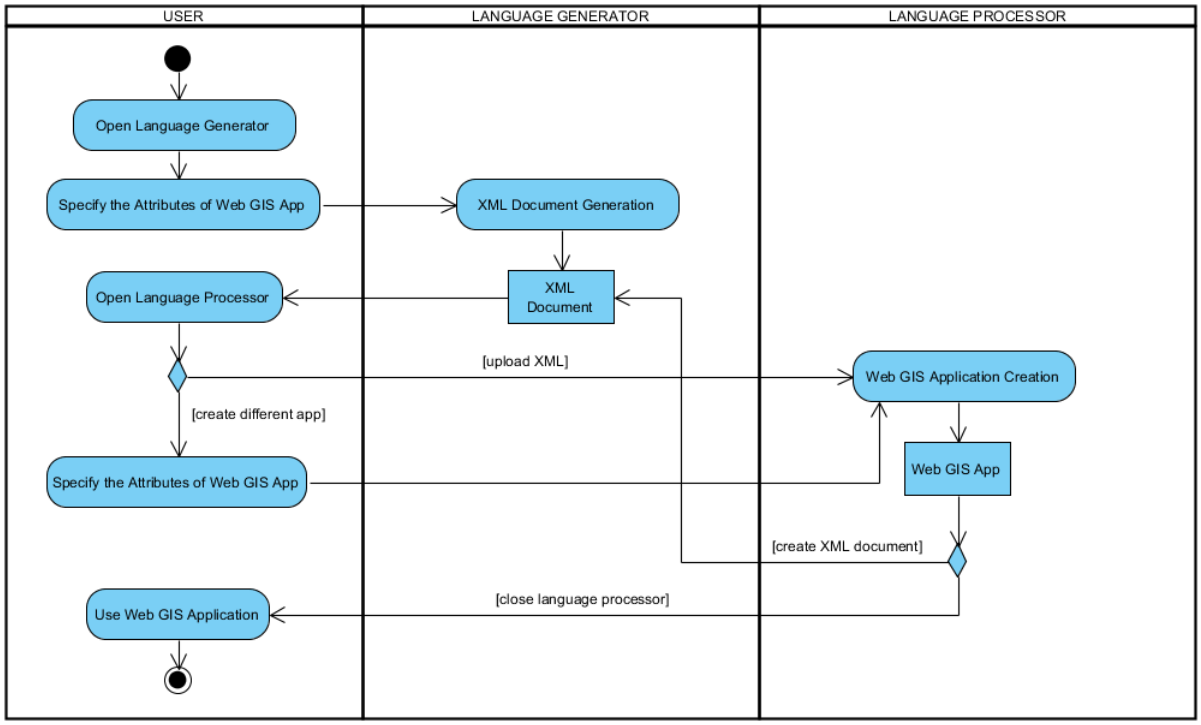

Figure 3.2: Activity Diagram

First, is necessary to open the language generator and define the characteristics of the Web GIS application. Then, the language generator creates a `XML` document with the specifications defined previously.

The resulting `XML` document can be uploaded to the language processor in order to generate the Web GIS application. Also, it is possible to create a different application in the language processor and generate the corresponding Web GIS application.

Moreover, a `XML` document can be created through language processor using the specifications defined. The cycle ends with the use of the resulting Web GIS application.

Chapter 4

Development

In this Chapter it is defined the development process and the three tools developed. The following sections describes the three main components developed (language, language generator and processor). Also, it is specified the technologies used in the development of the project. Those technologies were explained in the section 2.4.

In section 4.2 it is characterized the `XML` based language developed (named MapFishXML) and it is explained the schema of the `XML`. At section 4.1, it is given an introduction to the domain-specific languages (DSL), `XML` and XSD. The language generator (QGIS Plugin) and the language processor (MapFish Studio) are defined in the sections 4.4 and 4.3, respectively.

4.1 Background

In this section it is given a description of some of the technologies used for the development of the language. Those technologies are DSL (4.1.1), XML (4.1.2) and XSD (4.1.3).

4.1.1 Domain-specific Language (DSL)

A Domain-specific Language (DSL) is a language specification that pursuits a solution to a given problem of a specific particular domain. Logo, YACC grammars, SQL and DOT are some examples of DSLs. The opposite of DSLs are the General-purpose Languages (GPLs) such as C, Java or UML. In these languages (GPL) there is no specific area for which they were created, therefore the applications created using these languages, can belong to the most varied fields.

The advantages of using DSLs consist in the possibility of being able to express the solution in the language and level of abstraction of the problem domain. The DSL allows

the generation of commented code for the specified domain. Because of these characteristics DSLs increase the quality, productivity, reliability, portability and reusability, when applied to a specific field [vDKV00].

However the DSLs have some disadvantages, since sometimes the cost of developing and maintaining an DSL is very high. Also, the development requires both domain knowledge and language development expertise. Moreover, the language needs to be updated whenever a change occurs in the problem domain. Those languages can only be used in the specific domain. Finally it is complicated to do a debugging of DSL which can consume a large amount of time [MHS05, vDKV00].

4.1.2 XML

Extensible Markup Language (`XML`) is a meta-markup language that provides a format for describing structured data. This facilitates more precise declarations of content and more meaningful search results across multiple platforms. `XML` is designed to transport and store data, and is very easy to learn. `XML` is a World Wide Web Consortium (W3C) Recommendation. `XML` consists of a finite number of tags, or markup language structures that represents brief instructions, with a start tag and end tag [Tit03, Ray03].

XML provides a structured representation of data that has proven to be widely implementable and easy to be developed. `XML` is considered of great importance in the Internet and intranets because of its interoperability and portability. Being a open standard, flexible and device-independent, it can be delivered via HTTP without changes in existing networks [Tit03, Ray03].

Since `XML` is a verbose language, it is totally dependent on who is writing it. A verbose language may pose problems for other users. However for this project that problem was overcame because the `XML` based language is not to be analyzed and written by users. The language developed is generated and processed by software.

The Figure 4.1 describes an example of a `XML` document, representing employees information.

4.1.3 XML Schema

An `XML` Schema language, published as a W3C recommendation, is a formalization of the constraints that apply to a class of `XML` documents. Those constraints are the permissible names for elements and attributes, and permissible structures and values of elements and attributes. An `XML` Schema can be defined as a collection of validation rules for `XML`

```
<?xml version="1.0" ?>
<Employees>
  <Employee>
    <SSN>737333333</SSN>
    <Name>ED HARRIS</Name>
    <DateOfBirth>1960-01-01</DateOfBirth>
    <EmployeeType>FULLTIME</EmployeeType>
    <Salary>4000</Salary>
  </Employee>
</Employees>
```

Figure 4.1: XML Example

documents. This language is an alternative to `DTD`, whose syntax is not `XML`-based format [Vli02, MLM00].

The Figure 4.2 describes the Schema of the `XML` example specified in the Figure 4.1.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Employee" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SSN="xsd:string>
        <xsd:element name="Name" type="xsd:string"/>
        <xsd:element name="DateOfBirth" type="xsd:date"/>
        <xsd:element name="EmployeeType" type="xsd:string"/>
        <xsd:element name="Salary" type="xsd:long"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Figure 4.2: XML Schema Example

4.2 Language

This section gives a description of the language developed, named MapfishXML, based on the properties necessary to characterize a Web GIS application. The language is defined by the schema represented in the Figure 4.3.

For this project, the following properties were defined to characterize a Web GIS:

- Name of the application.
- Main projection.

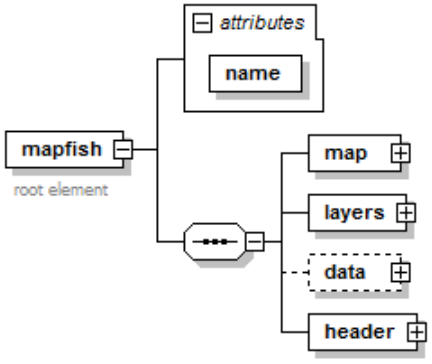

Figure 4.3: MapfishXML Schema

- Maximum extent and its units.
- Layers.
- Widgets for data manipulation and analyses.

Those properties were specified in the schema. Obviously, other properties were defined and considered important for the mapfish framework used in this project. Below it is described each element of the `XML` language defined by its schema, where the root element is *mapfish*. The *mapfish* element has an attribute identified as *name* which determines the name of the application. Additionally, the root element is a collection of four elements: *map*, *layers*, *data* (optional) and *header*. Those four elements are described in the following subsections.

4.2.1 Map

In the Figure 4.4 it is represented the *map* element and its child elements. This element contains a sequence of five elements: *projection*, *scales*, *units*, *maxExtent* and *dpi*.

The projection of the application (for example "EPSG:4326") is defined in the element *projection*. At the element *scales* it can be specified an unlimited set of zoom scales. The element *scale* has the type integer. The units, which can be "dd" (degrees), "miles", "meters", "kilometers", "inches" or "feet", are defined in the *units* element. This represents the units of the spatial extent of the map. In the element *maxExtent* is specified the maximum spatial extent of the map. All the *leftX*, *bottomY*, *rightX* and *topY* elements have the type double and should be accordingly the projection of the map. Finally, the element *dpi* (dots per inch) has the type integer and is used for a tool named print in the mapfish framework. This element has a default value of "72".

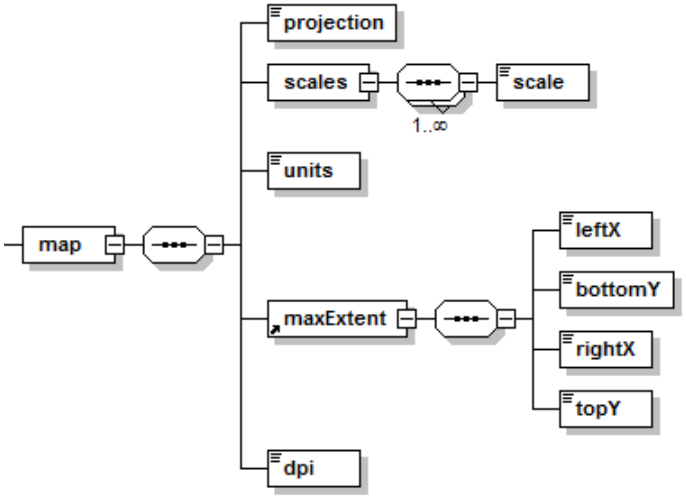

Figure 4.4: Map Element of the MapfishXML Schema

4.2.2 Layers

The Figure 4.5 describes the element *layers* and its child elements. Layers is a collection of *baselayer* and/or *overlayer*. Those two elements have the same type named **layer_options**.

The element *baselayer* has an optional attribute named *transparent* of type boolean (true or false), which determines if a layer is transparent or not. If this attribute is not defined it is assumed that the value is false. The layer label is specified in the element *name*.

To request a layer it is necessary to provide an url of the data. The url could be a request from MapServer, GeoServer or PostGIS. Examples of a request from those three data providers:

- MapServer - http://alfarrabio.di.uminho.pt/cgi-bin/mapserv?map=/home/test.map
- GeoServer - http://ogi.us/geoserver/ows?service=WMS&request=GetCapabilities
- PostGIS - PG:host=bombordo.pt dbname=mygisdb user=mike password=qwerty

Those requests should be represented in the element *url*. The type of the layer request (WMS, WFS or WCS) is defined at the element *type*. In the element *maxExtent* it is specified the maximum spatial extent of the data (layer). An unlimited set of layers are defined in the elements *layer*. Those elements are a selection of layers from the given data storage url. Optionally, an icon can be assigned to the layer label by a given url in the *icon_url* element.

4.2.3 Data

This element (optional) is used to determine the widgets to be presented in the application. It is only permitted to use widgets to analyze or manipulate data that are in a database PostGIS. Therefore, the attributes *host*, *dbname*, *user* and *password* are used to specify the PostGIS connection. The Figure 4.6 describes this element.

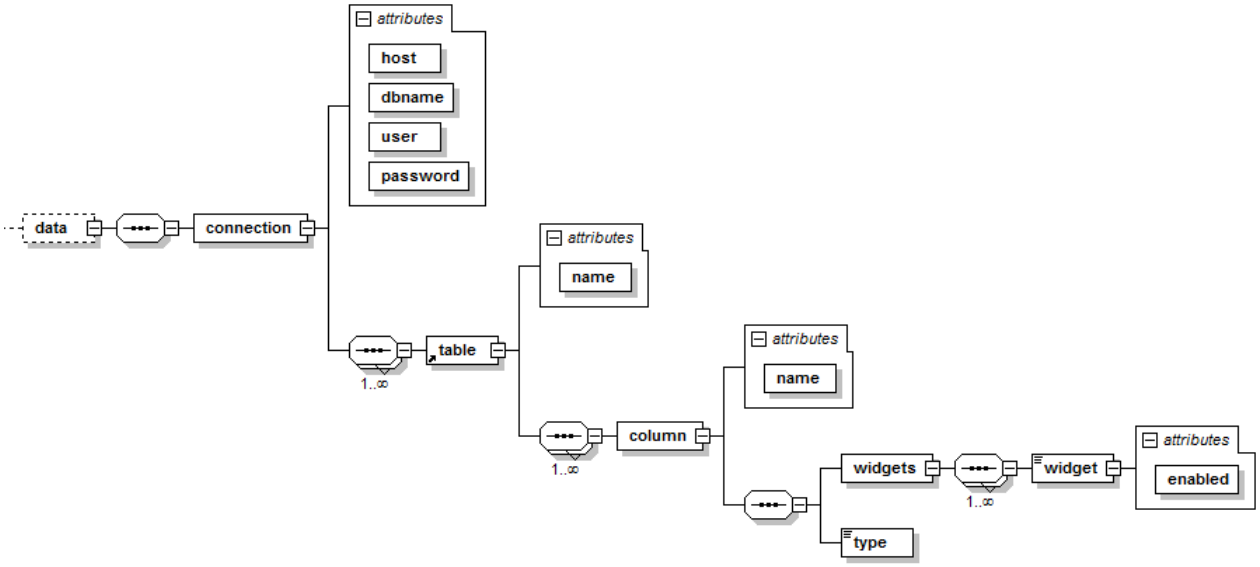

Figure 4.6: Data Element of the MapfishXML Schema

For this version, the element *data* has only one *connection*, because the mapfish framework allows only one database connection. For the PostGIS connection it is possibly to specify an unlimited number of tables and its columns. For each table or column, the attribute *name* determines the name of the table or column. A set of Widgets should be associated to a column of a table. The available Widgets are:

- search - With this Widget the user could search for any value of the associated column.
- tooltip - This allows users to mouse over spatial data elements and expose information (column value) in the form of descriptive text displayed in a pop-up window.
- editable - The edit Widget is designed to make the end user editing experience simple and intuitive. It allows users to edit the value of the associated column.

Those Widgets have the attribute *enabled* to determine if the Widget is enabled or disabled. This attribute is of boolean type (true or false). Finally, for each set of Widgets it is necessary to associate a *type* that could be "string" or "numeric".

4.2.4 Header

Finally, the Figure 4.7 represents the *header* element. This element specifies the top bar layout of the application generated. This element was defined only for the mapfish framework.

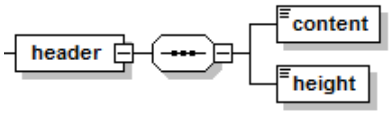

Figure 4.7: Header Element of the MapfishXML Schema

The *content* element has type string and can be an url for an image or a simple text. The height of the bar is defined in the element *height*.

4.3 Language Processor

It was required to test the implementation of the language developed in a real case study. Therefore, was necessary to develop a tool to process the language and generate a Web GIS application based on the MapfishXML language. Developing such a tool consumes a lot of time and is very complex. Since there was no time to develop a tool of this magnitude during the project, it was decided to modify an existing tool. In this project was used the MapFish framework. As can be read in Section 2.4.2.3, this framework allows a simple development of a Web GIS application. Furthermore, due to the fact that this framework is open source, it is possible to modify it accordingly to the developer needs.

This framework contains a web software, named MapFish Studio (Figure 4.8), used to administrate web mapping applications based on MapFish framework. This software allows the web administrator to manage the geodata, to apply styles with the wysiwyg editor MapFile (for MapServer) and to create a custom web mapping application.

This tool (MapFish Studio) was modified to process the MapfishXML developed. After clicking the *Create new MapFish* button, a window is opened. In that window a button **Upload XML** was introduced (Figure 4.9). When it is clicked, a pop-up window (Figure 4.10) is opened where it is possible to enter the URL of the XML. After clicking the **Upload** button, Studio converts the MapfishXML to JSON, which is the language that stores all the information of the Web GIS applications. The transformation of MapfishXML to JSON is done through an XSL built for that purpose. Finally, the framework

generates the Web GIS application based on the resulting JSON. The application generated can now be executed in the MapFish Studio server. Alternatively, the application package can be downloaded and deployed into another application server.

4.4 Language Generator

4.4.1 Quantum GIS (QGIS)

Quantum GIS (QGIS) is a user friendly geographic data viewer with a set of analytical capabilities (Figure 4.11), licensed under the GNU General Public License. QGIS is an official project of the Open Source Geospatial Foundation (OSGeo), and runs on Linux, Unix, Mac OSX, and Windows [NM08].

It supports numerous vector, raster, and database formats and functionalities. Some of the formats supported are the ESRI's proprietary shapefile, MapInfo and PostGIS. Also, provides access to data from external sources by using the WMS and WFS standards [NM08].

QGIS provides a continously growing number of capabilities provided by core functions and plugins. It is possible to visualize, manage, edit, analyse data, and compose printable maps [NM08].

4.4.2 QGIS Plugin

The QGIS plugin was developed with the objective of facilitate the writing of the XML language. Thus, it is possible to generate the language, interactively, without requiring any prior knowledge of the language structure (schema).

The plugin was developed in Python. A template was used to generate all the basic code needed. The plugin interface was developed through the program QT4 designer. This application is a tool for designing and building graphical user interfaces (GUIs) from Qt components. It generates the Python code of the GUI designed.

The GUI of this plugin is represented by a window that contains several forms, where a user can fill all the information to generate the language. In the Figure 4.12 it is represented the QGIS plugin.

Some of these forms are filled automatically through the settings made in QGIS such as:

- Application Name
- Projection

- Layers
- Data

The application name is defined in the "project title" on the QGIS project properties. The projection is specified in the properties of the QGIS project. The form of the layers displays a tree with all the layers added to the QGIS project. Those layers have the same projection of the project. Therefore, the layers that have a different projection are ignored. Only the layers that are stored in a PostGIS database or in a WMS server can be added to the project. It is possible to create sublayers by drag and drop. Finally, the form *data* is a representation of a database table. For each layer from PostGIS, the attributes of that layer (columns of the table) are added to the form data. For each column it is possible to enable the widgets editable, tooltip and/or search (are disabled by default).

apart from these, QGIS plugin also provides other forms not filled automatically by QGIS. Therefore, these forms have default values that can be modified. These forms are:

- Units
- DPI
- Scales
- Maximum Extent
- Content
- Height

These forms allow the users to define the elements of XML that are described in Section 4.2, such as *units*, *dpi*, *scales*, *maxExtent*, *content* and *height*.

When the *OK* button is pressed (after all the forms are filled in) the plugin creates a XML file with the given specifications.

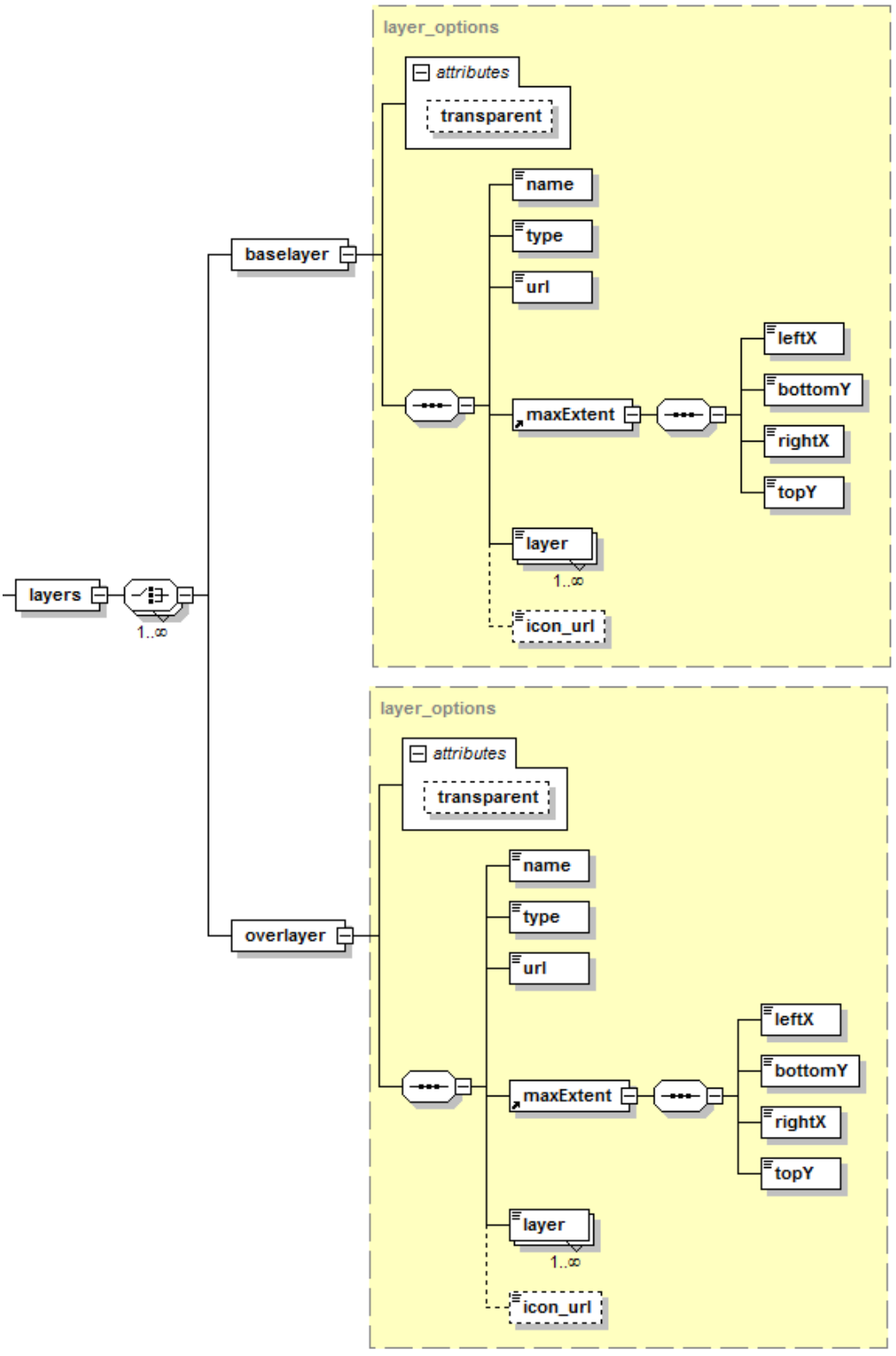

Figure 4.5: Layers Element of the MapfishXML Schema

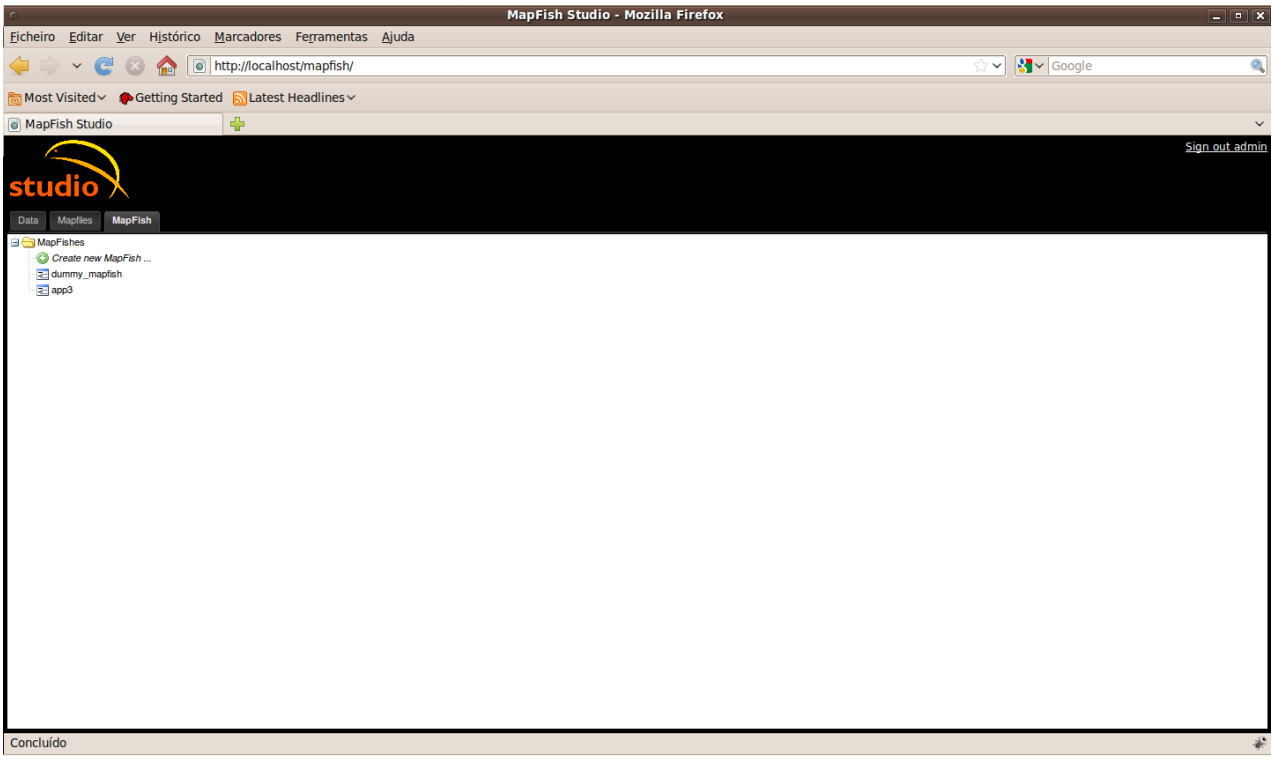

Figure 4.8: MapFish Studio

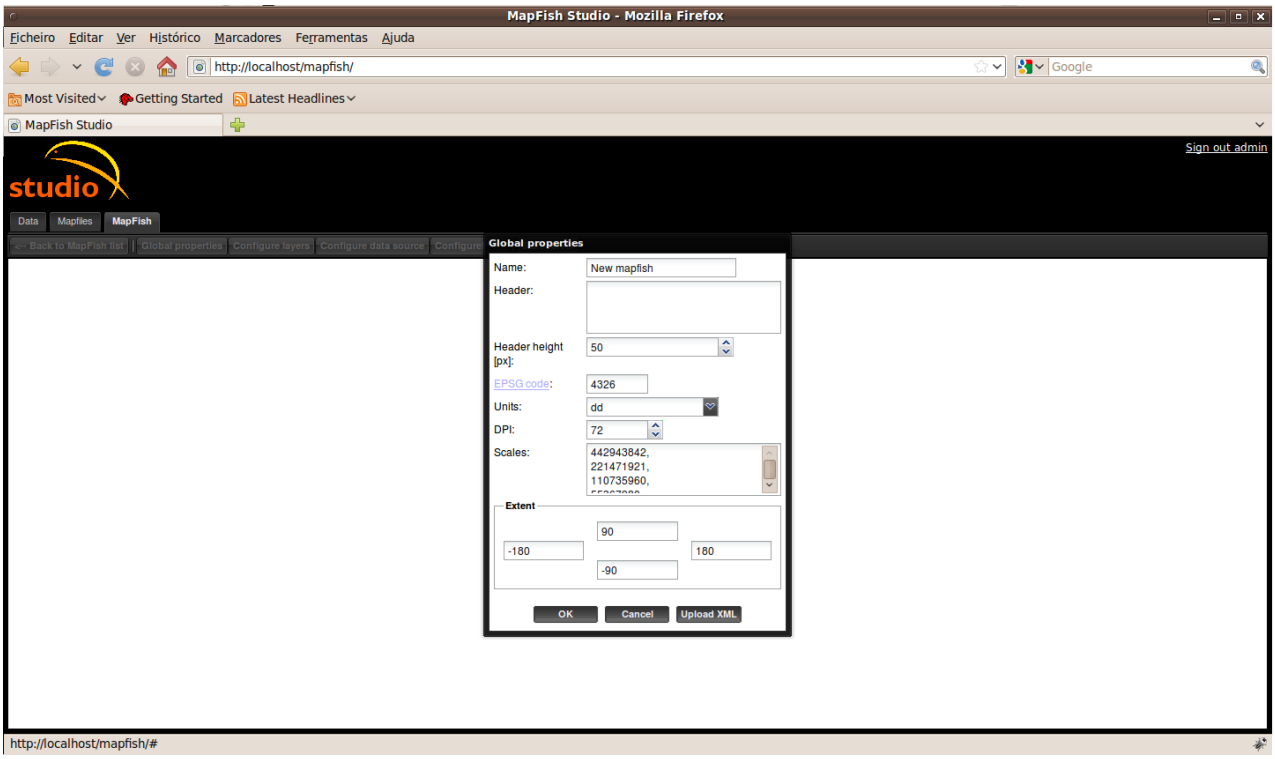

Figure 4.9: MapFish Studio - Upload XML

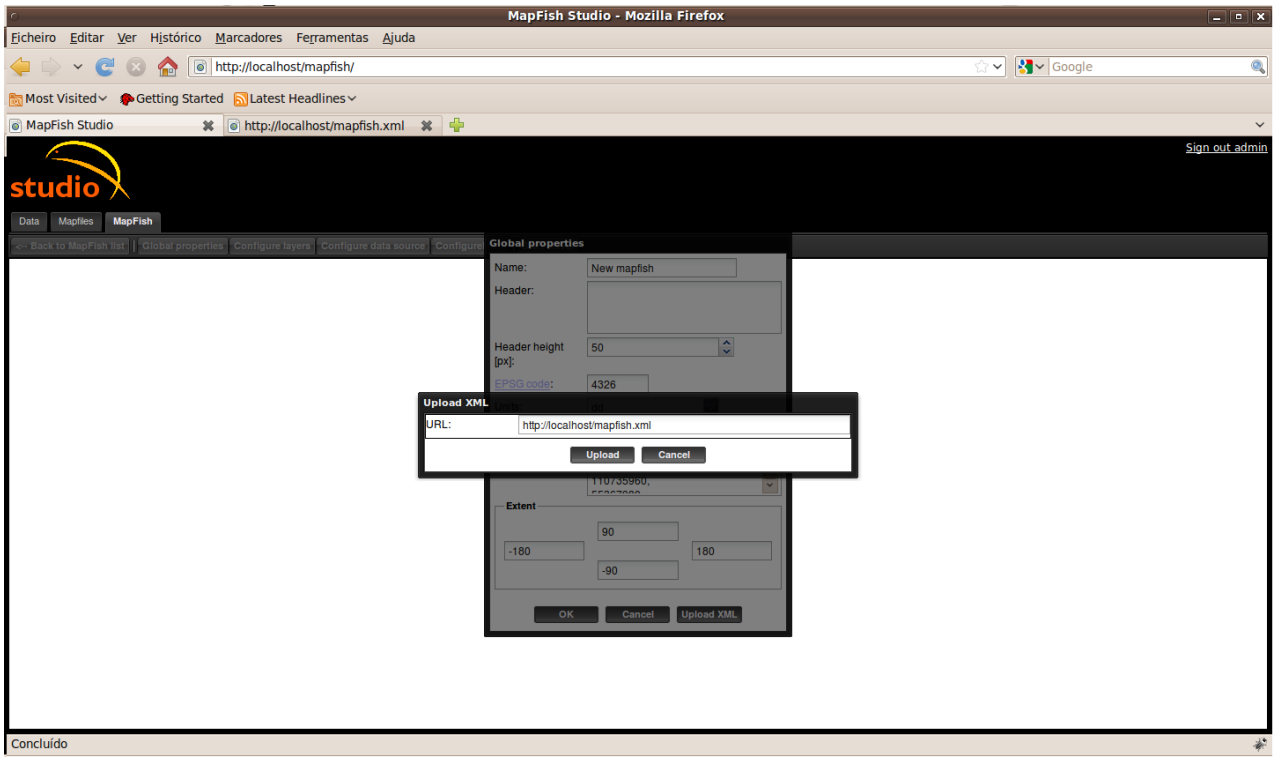

Figure 4.10: MapFish Studio - Window to Insert URL of the XML

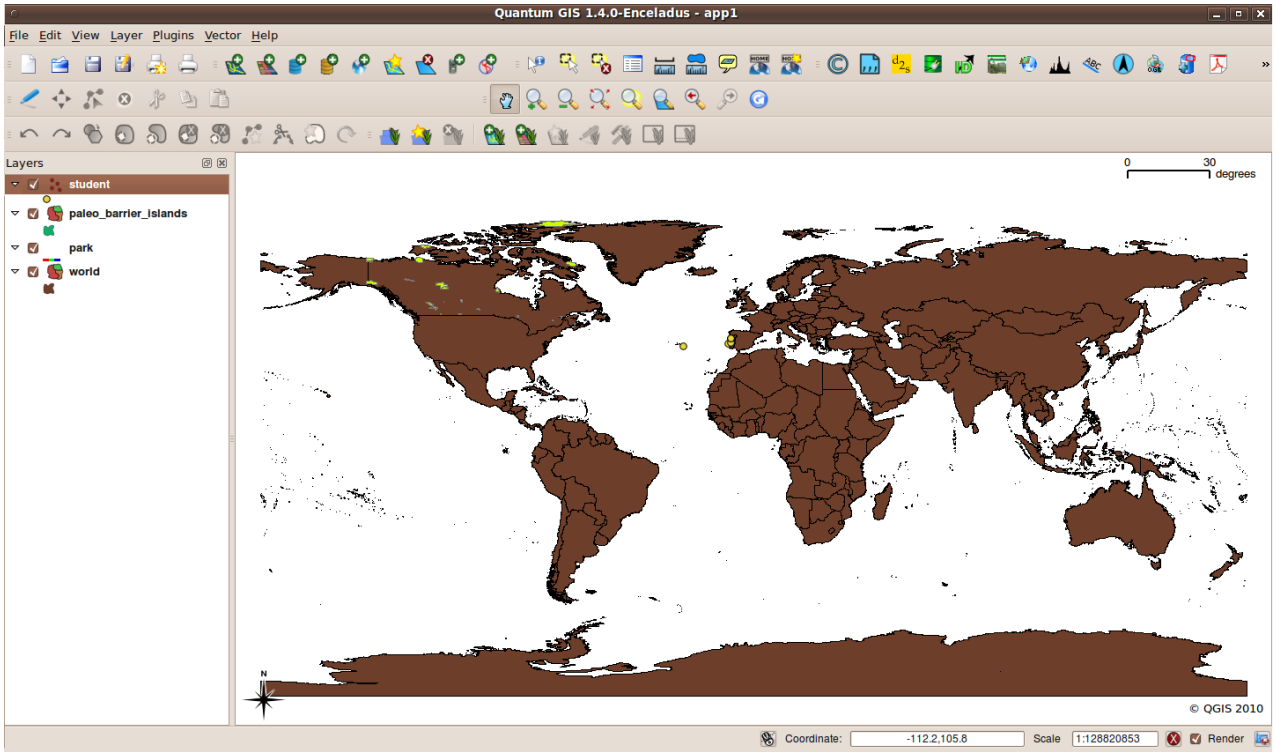

Figure 4.11: QGIS Interface

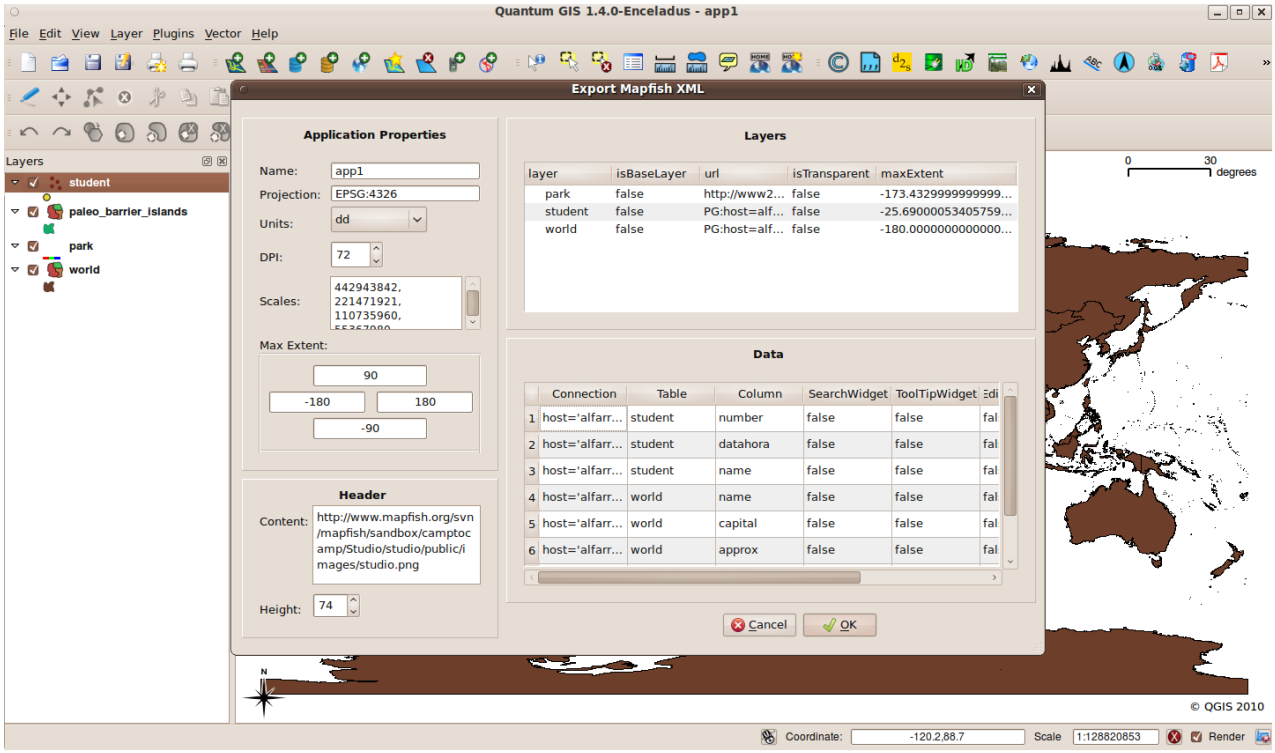

Figure 4.12: QGIS Plugin

Chapter 5

Results

This Chapter illustrates some examples of the generation of Web GIS applications and the use of the tools developed for that purpose. In each example it is described the required application, the XML document and it is demonstrated how the same document was created through QGIS plugin, and the resulting Web GIS application.

An example of a simple application is described in the section 5.1. In section 5.2 it is illustrated an example of a generation of an application that has widgets associated and more layers than the first example.

5.1 Example 1

For this section it is represented an example of a simple Web GIS application. This application has few layers and does not have widgets associated. The QGIS plugin was used to create the XML document. In this tool it is specified an application named *Hello World*, with the following layers:

- **world**, from PostGIS connection: dbname='mygisdb' host=alfarrabio.di.uminho.pt port=5432 user='mike'
- **roads**, from WMS: 'http://www2.dmsolutions.ca/cgi-bin/mswms_gmap'
- **rail**, from WMS: 'http://www2.dmsolutions.ca/cgi-bin/mswms_gmap'
- **world_cities**, from PostGIS connection: dbname='sig' host=alfarrabio.di.uminho.pt port=5432 user='mike'

The Figure 5.1 illustrates the four layers added to the QGIS plugin and its representation.

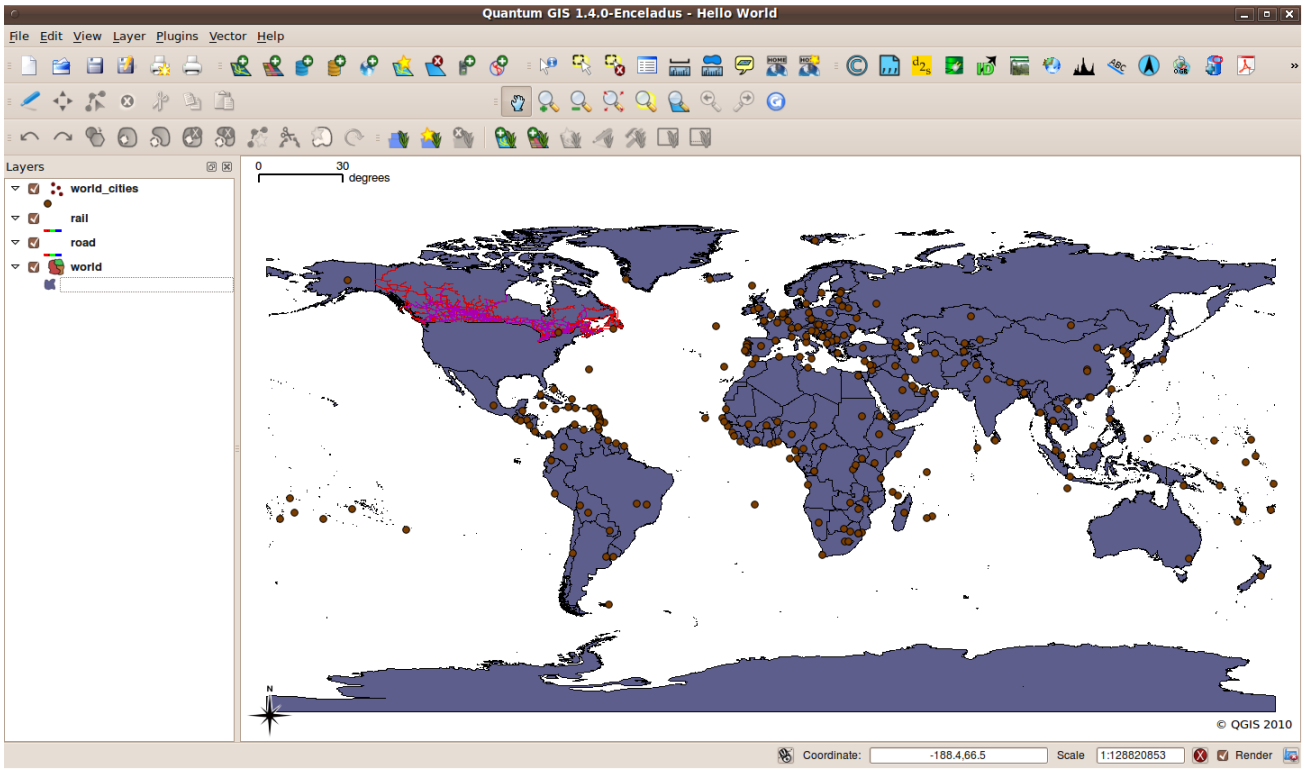

Figure 5.1: Layers added in QGIS

The application name was specified as *Hello World*. Also, were defined four layers. The layer **world** was defined as a base layer. Thereafter, it was opened the QGIS plugin MapfishXML. The plugin settings are described in the Figure 5.2.

In the layer **world** the column *isBaseLayer* was changed to *true* to define **world** as a base layer. In order to have overlap between layers, the layers that are not base layers need to have the attribute *isTransparent* defined as "true". As seen in the Figure 5.2 no widget has been activated. The other parameters in the plugin were conserved as default.

The generated XML document is described in the Appendix A. The XML document represents the application required. The resulting Web GIS application for this XML document (Appendix A) can be seen in the Figure 5.3.

The Figure 5.3 illustrate an application with one base layer and three overlayers. The overlayers can be set to not visible. The map shows the four layers selected. It uses the openlayers javascript library. That library provides tools like Zoom, DragFeature, OverviewMap, Permalink, etc. Finally, the header of the application is a default image of the symbol of MapFish Studio.

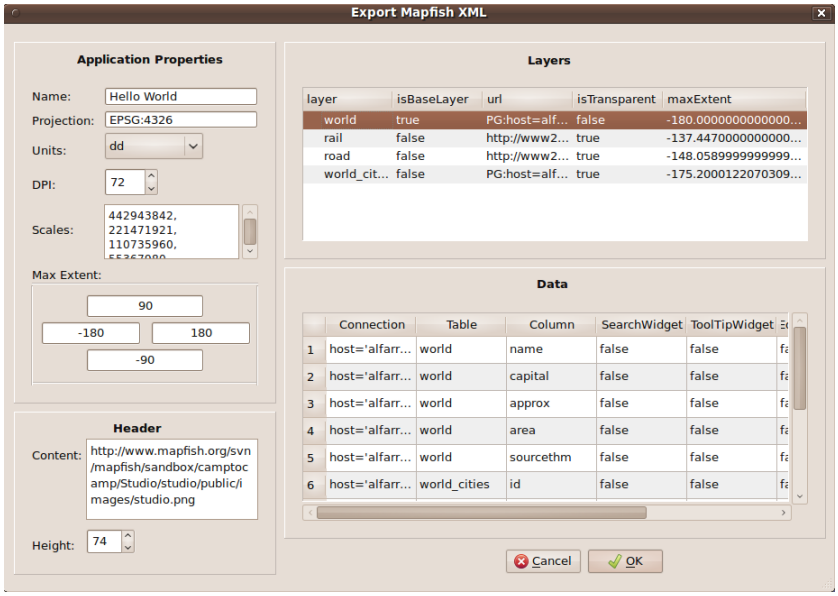

Figure 5.2: QGIS Plugin Specifications

5.2 Example 2

The second example is an Web GIS application with six layers. Some of those layers are sublayers. Also, the widgets editable, tooltip and search are used in this application. To create the XML document it is used the QGIS plugin. In this tool it is specified an application named *World Atlas*, with the following layers:

- **world_cities**, from PostGIS connection: dbname='mygisdb' host=alfarrabio.di.uminho.pt port=5432 user='mike'
- **river**, from PostGIS connection: dbname='mygisdb' host=alfarrabio.di.uminho.pt port=5432 user='mike'
- **ind-storage-p**, from PostGIS connection: dbname='mygisdb' host=alfarrabio.di.uminho.pt port=5432 user='mike'
- **sc500k**, from WMS: 'http://mapas.igeo.pt/wms/sc500k'
- **basic**, from WMS: 'http://labs.metacarta.com/wms/vmap0'
- **TM_WORLD_BORDERS_SIMPL-0.3**, from PostGIS connection: dbname='mygisdb' host=alfarrabio.di.uminho.pt port=5432 user='mike'

In the Figure 5.4 are illustrated the six layers added to the QGIS plugin and its representation.

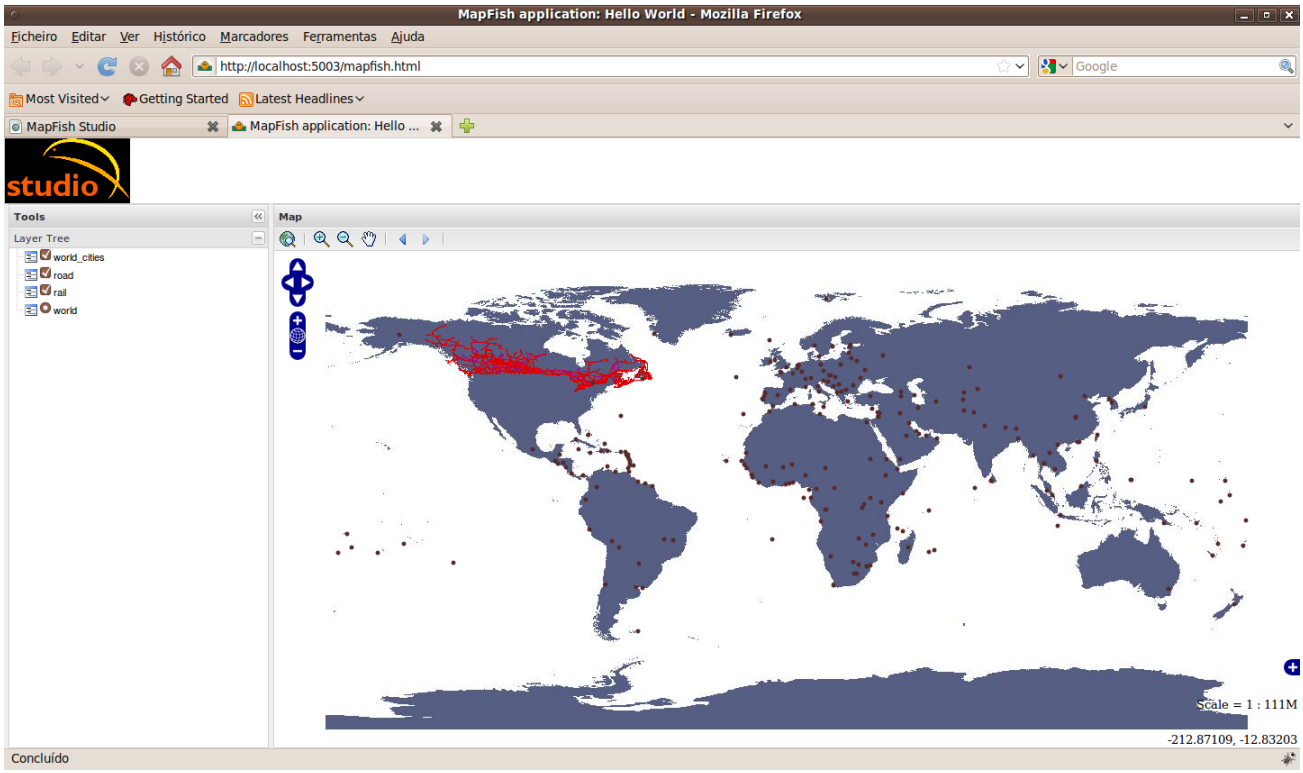

Figure 5.3: Web GIS Application

The application name was specified as *World Atlas*. Also, were defined six layers. The layers **basic** and **TM_WORLD_BORDERS_SIMPL-0.3** were defined as base layers. Thereafter, it was opened the QGIS plugin MapfishXML.

The layer **ind-storage-p** was specified as a sublayer of **world_cities**. This was done by drag and drop. Also, the following widgets were defined:

- Search Widget for the column 'name' in the table 'world_cities'
- Editable Widget for the column 'name' in the table 'river'
- Tooltip Widget for the columns 'ele_popu', 'inflation', 'pays' and 'population' in the table 'TM_WORLD_BORDERS_SIMPL-0.3'

The plugin settings are described in the Figure 5.5.

The Figure 5.5 shows that the widgets described above were activated. The content field was defined as none (empty). The other parameters in the plugin were conserved as default.

The generated XML document is described in the Appendix B. The XML document represents the application required. The label of the group of sublayers **ind-storage-p**

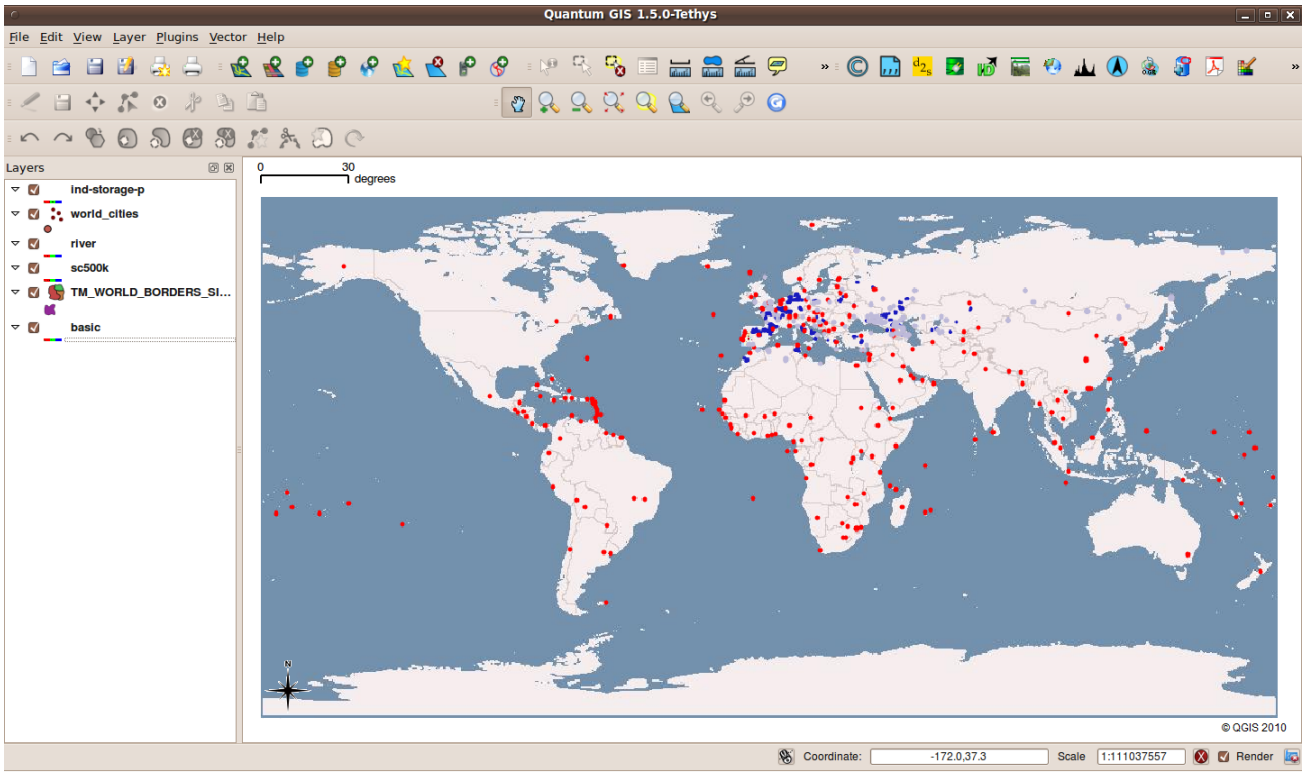

Figure 5.4: Layers added in QGIS

and **world_cities** was changed to *world* in the XML document. The resulting Web GIS application for this XML document (Appendix B) can be seen in the Figure 5.6.

The Figure 5.7 demonstrates the use of tooltip widget. After selecting the layer **TM_WORLD_BORDERS_SIMPL-0.3**, the country *Spain* was clicked over. Then a pop-up window opened with information about inflation, population, name of the country and elected population.

The testing of the widget search is illustrated in the Figure 5.8. For this demonstration the layers **basic**, **world_cities** and **sc500k** (Map of Mainland Portugal) were selected. In this example, when the capital of Portugal (Lisbon) was searched for, the application automatically zoomed to Lisbon city.

The edit widget is demonstrated in the Figures 5.9, 5.10, 5.11 and 5.12. The layer defined to be editable was **river**. First, it was selected an area of the map and the widget edit (Figure 5.9) was opened. In this widget, the drawing tool (*Draw Polygons*) was activated. Then, the river was created using clicks (Figure 5.10). After that, it was introduced a name 'Test_River' and clicked the button *OK* (Figure 5.11). Finally, the feature is inserted in the database by clicking in the button *commit* (Figure 5.12).

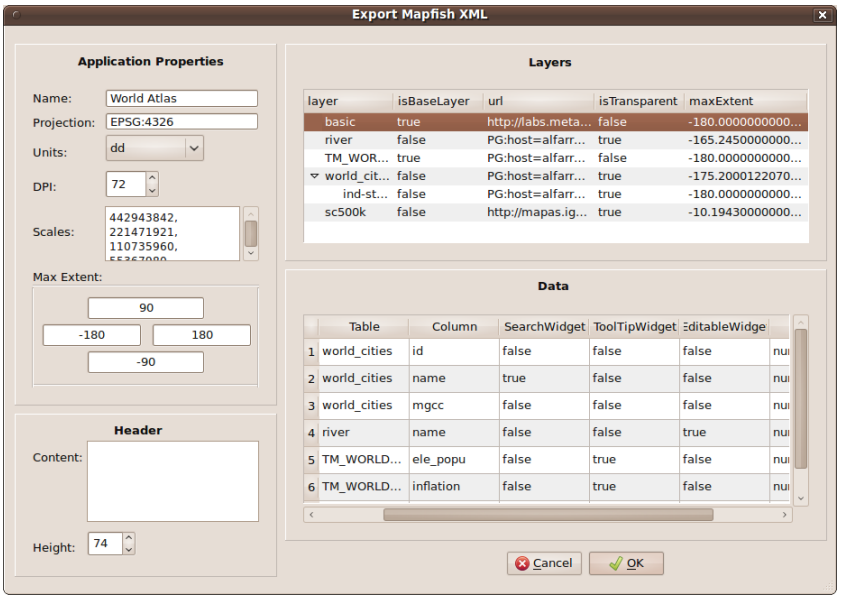

Figure 5.5: QGIS Plugin Specifications

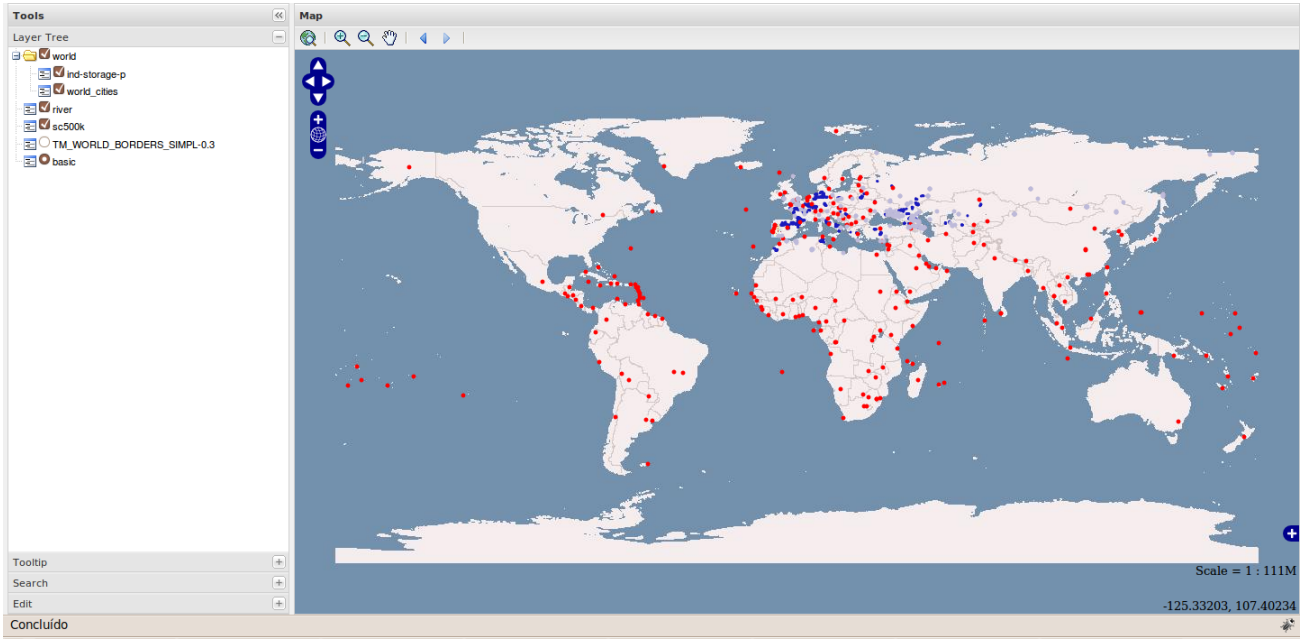

Figure 5.6: Web GIS Application

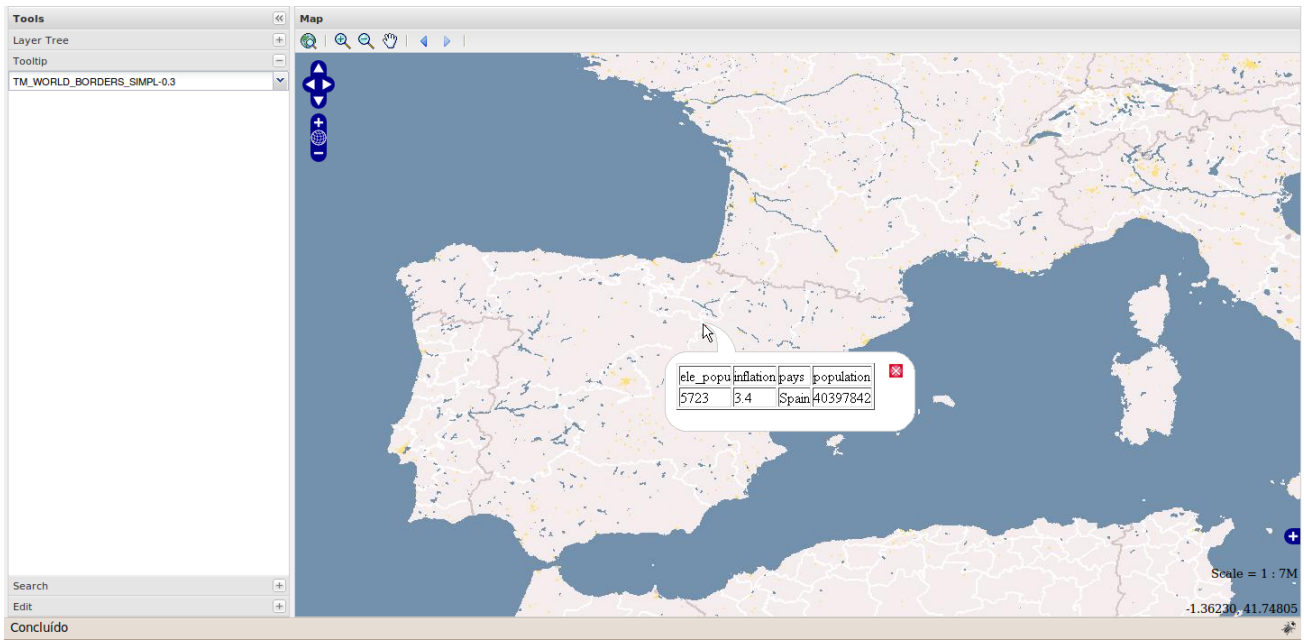

Figure 5.7: Web GIS Application - Tooltip Widget

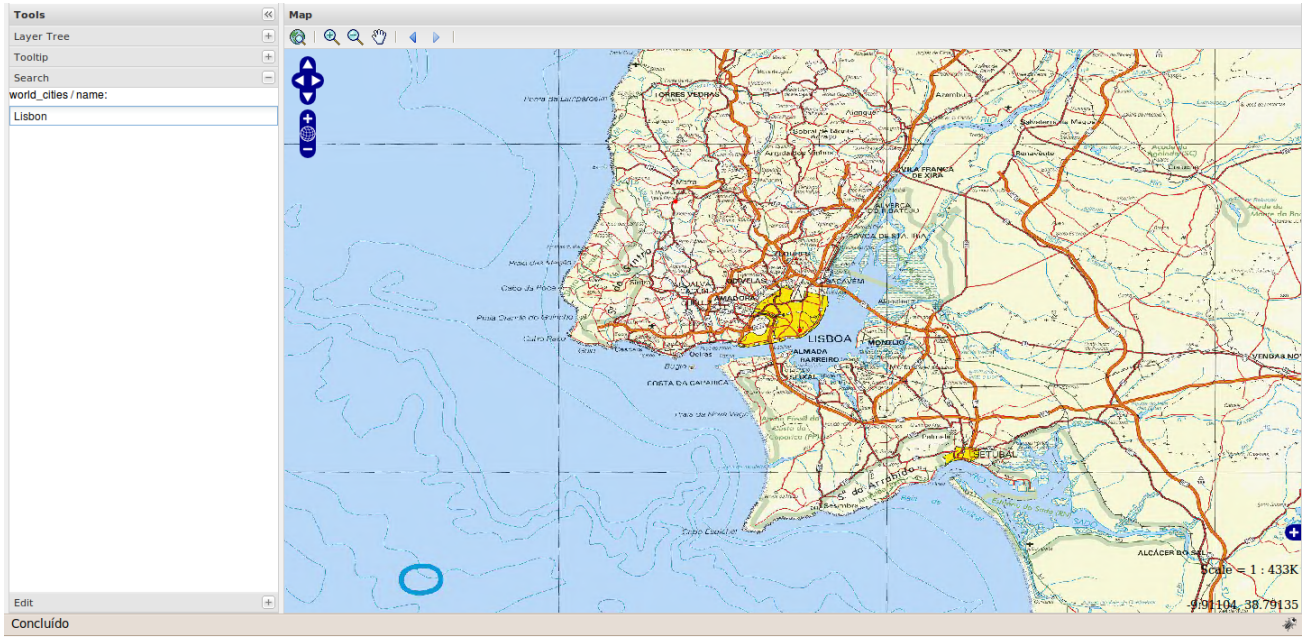

Figure 5.8: Web GIS Application - Search Widget

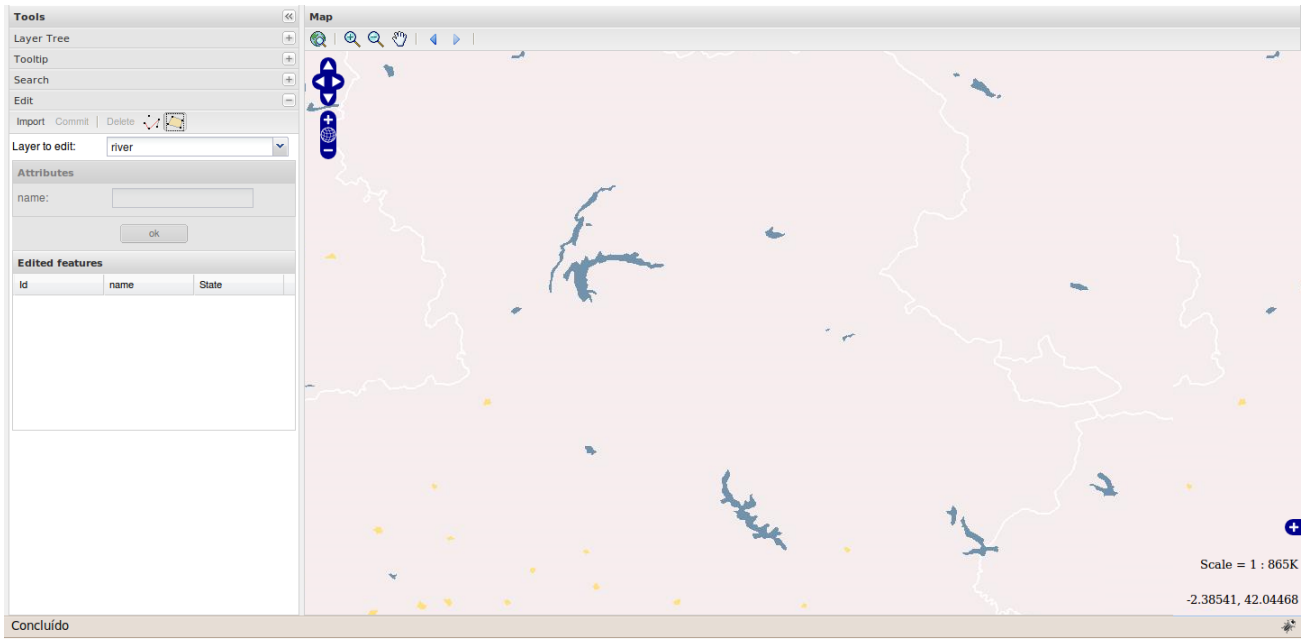

Figure 5.9: Web GIS Application - Editable Widget (1)

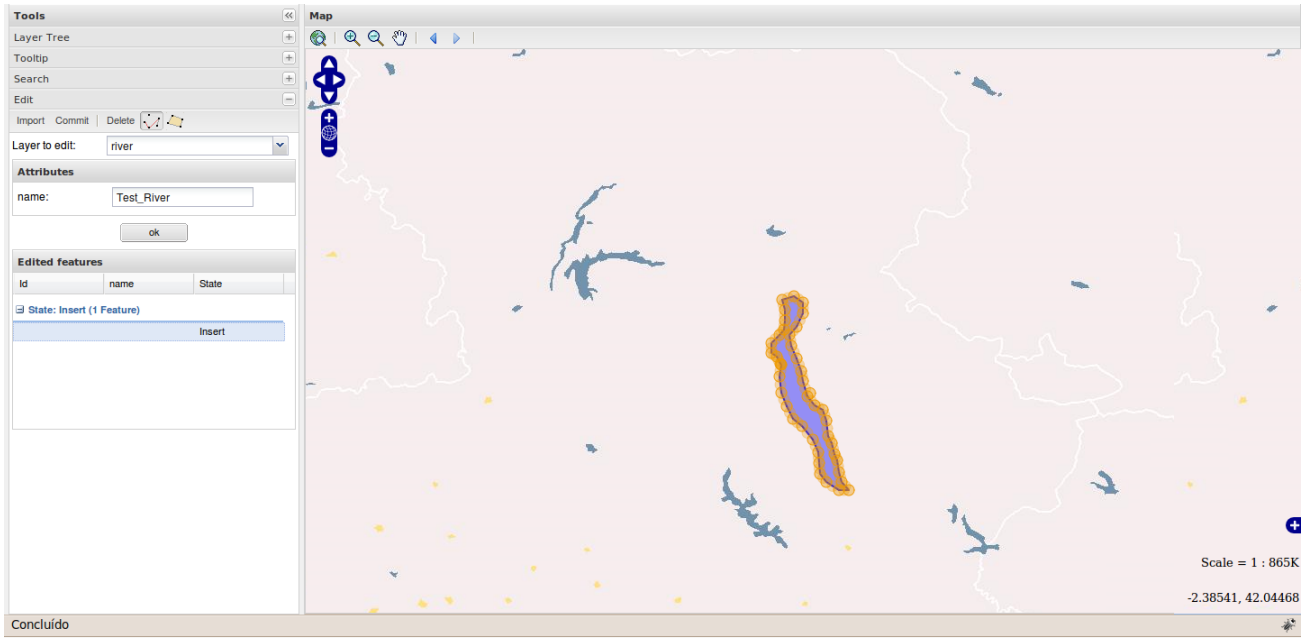

Figure 5.10: Web GIS Application - Editable Widget (2)

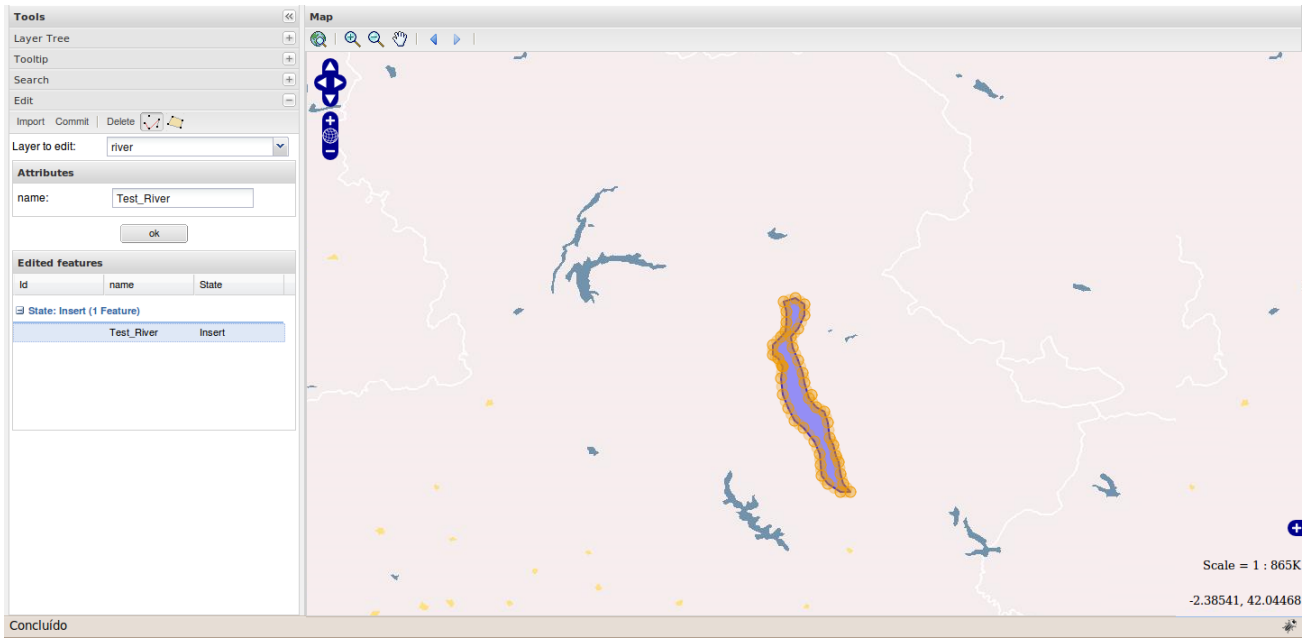

Figure 5.11: Web GIS Application - Editable Widget (3)

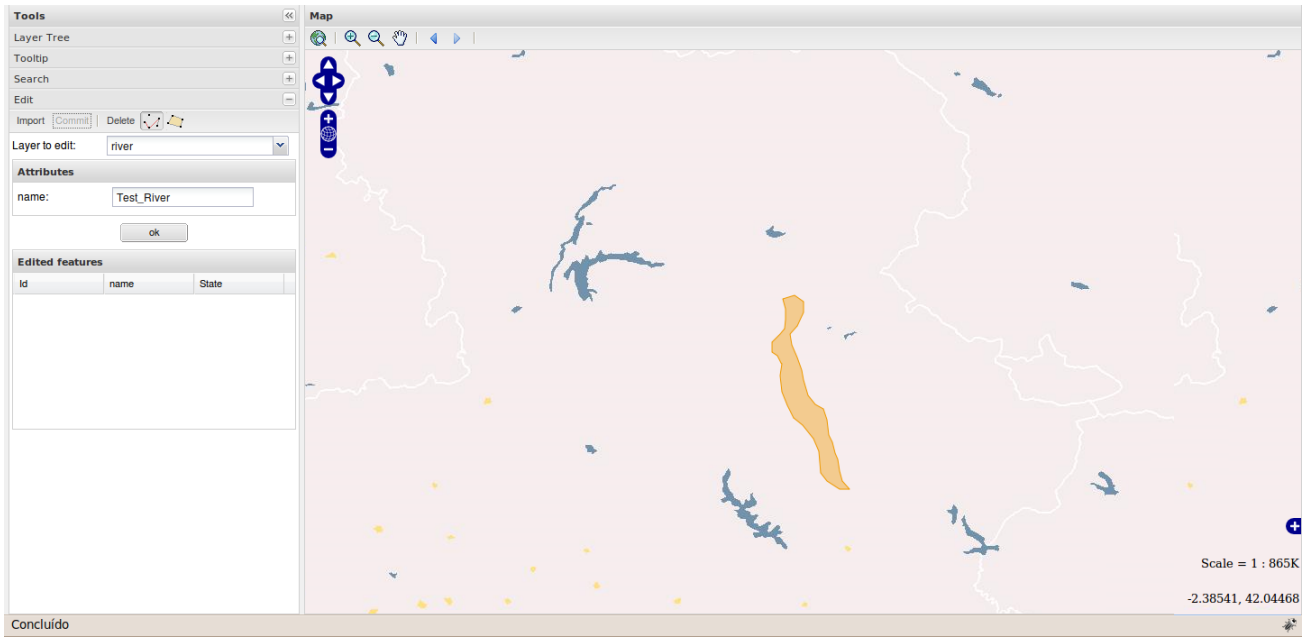

Figure 5.12: Web GIS Application - Editable Widget (4)

Chapter 6

Conclusions and Future Work

This chapter concludes the dissertation. It is divided into three sections: conclusion (Section 6.1), future work (Section 6.2) and personal opinion (Section 6.3). In the first section it is clarified if the objectives of the project were achieved. The second section identifies some future developments in the project that could make a more attractive and generic MapfishXML language, and efficient tools. Finally, in the third section some final considerations are given (by the author).

6.1 Conclusion

In this dissertation it was studied a solution for the problem (Section 1.2). The solution for the problem was to model and specify a domain specific language capable of representing hundreds or thousands of layers in a single map.

This work contributed to the development of a prototype that allows users to create their own Web GIS application with few knowledges of GI. We conclude that the objectives proposed in this dissertation were achieved. According to the prototype developed, the way developers create Web GIS applications can now be standardized. The prototype was based on three main components:

- A language based on `XML` which describes the map properties, the layers to be displayed and a set of `Widget`s to analyze and manipulate data, in other words, this language is able to represent GI on the Web.

- An application that analyses the language developed and generates a Web `GIS` application with fully functional tools to display, edit, analyze, and manipulate data.

- An application that creates a XML file with the Web GIS application description. With this application the user can choose which layers to be presented, specify the global properties of the Web GIS application and choose the widgets to be used.

Therefore, with this XML based language it is guaranteed a systematic publication of GI on Web with the support of the language generator and language processor tools developed.

The major difficulties in the development of this project were the lack of knowledge in the area of GI. Also, the use of MapFish Framework required an wide study of the Pylons Framework, which consumed a lot of time. However, it was learned the Python language (used in the server side of the MapFish Framework) and the JavaScript libraries ExtJS and Openlayers.

6.2 Future Work

Although the objectives were achieved, some improvements can be made in the future. The language can be enriched with more widgets, and with the possibility to choose the layout of the generated applications. Also, a SLD could be included in the language to be able to control the visual portrayal of the geospatial data.

For this project the MapFish Framework was the only that was modified to import the MapfishXML language. For the future, more Frameworks could be adapted to accept the language. The future goal is for this language to become a standard. But this is only possible if a great number of GIS corporations incorporated this language in their Frameworks.

Finally, the QGIS plugin should allow adding vector and raster layers from shapefiles (.shp) or images (PNG, JPEG, TIFF, etc). For the language to become widely know more applications (in various programming languages) should be developed to generate the MapfishXML.

6.3 Personal Opinion

This research project was very interesting, not only for being a challenging proposal but also for the knowledge achieved. The meetings with the professor facilitated the understanding and the achievement of the goals, making the orientation very helpful. Besides that, with this project I became more interested in this area of GIS.

Appendix A

HelloWorld.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mapfish name="Hello World" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://
    localhost/mapfish.xsd">
     <map>
          <projection>EPSG:4326</projection>
          <scales>
               <scale>442943842</scale>
               <scale>221471921</scale>
               <scale>110735960</scale>
               <scale>55367980</scale>
               <scale>27683990</scale>
               <scale>13841995</scale>
               <scale>6920997</scale>
               <scale>3460498</scale>
               <scale>1730249</scale>
               <scale>865124</scale>
               <scale>432562</scale>
               <scale>216281</scale>
               <scale>108140</scale>
               <scale>54070</scale>
               <scale>27035</scale>
               <scale>13517</scale>
          </scales>
```

```
        <units>dd</units>
        <maxExtent>
            <leftX>-180</leftX>
            <bottomY>-90</bottomY>
            <rightX>180</rightX>
            <topY>90</topY>
        </maxExtent>
        <dpi>72</dpi>
    </map>
    <layers>
        <baselayer transparent="false">
            <name>world</name>
            <type>WMS</type>
            <url>PG:host=alfarrabio.di.uminho.pt dbname=mygisdb
                user=mike password=qwerty</url>
            <maxExtent>
                <leftX>-180.0000000000000000</leftX>
                <bottomY>-76.2670822143554972</bottomY>
                <rightX>180.0000000000000000</rightX>
                <topY>83.6747360229492045</topY>
            </maxExtent>
            <layer>world</layer>
        </baselayer>
        <overlayer transparent="true">
            <name>rail</name>
            <type>WMS</type>
            <url>http://www2.dmsolutions.ca/cgi-bin/mswms_gmap</
                url>
            <maxExtent>
                <leftX>-137.4470000000000027</leftX>
                <bottomY>37.7145999999999972</bottomY>
                <rightX>-46.3200999999999965</rightX>
                <topY>66.7201000000000022</topY>
            </maxExtent>
            <layer>rail</layer>
        </overlayer>
```

```
        <overlayer transparent="true">
            <name>road</name>
            <type>WMS</type>
            <url>http://www2.dmsolutions.ca/cgi-bin/mswms_gmap</
                url>
            <maxExtent>
                <leftX>-148.0589999999999975</leftX>
                <bottomY>35.8819999999999979</bottomY>
                <rightX>-33.7745000000000033</rightX>
                <topY>72.5502999999999929</topY>
            </maxExtent>
            <layer>road</layer>
        </overlayer>
        <overlayer transparent="true">
            <name>world_cities</name>
            <type>WMS</type>
            <url>PG:host=alfarrabio.di.uminho.pt dbname=sig user
                =mike password=qwerty</url>
            <maxExtent>
                <leftX>-175.2000122070309942</leftX>
                <bottomY>-51.7000007629395029</bottomY>
                <rightX>179.2166748046880116</rightX>
                <topY>78.2166671752929972</topY>
            </maxExtent>
            <layer>world_cities</layer>
        </overlayer>
    </layers>
    <header>
        <content>http://www.mapfish.org/svn/mapfish/sandbox/
            camptocamp/Studio/studio/public/images/studio.png</
            content>
        <height>74</height>
    </header>
</mapfish>
```

Appendix B

WorldAtlas.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<mapfish name="World Atlas" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://
    localhost/mapfish.xsd">
     <map>
          <projection>EPSG:4326</projection>
          <scales>
               <scale>442943842</scale>
               <scale>221471921</scale>
               <scale>110735960</scale>
               <scale>55367980</scale>
               <scale>27683990</scale>
               <scale>13841995</scale>
               <scale>6920997</scale>
               <scale>3460498</scale>
               <scale>1730249</scale>
               <scale>865124</scale>
               <scale>432562</scale>
               <scale>216281</scale>
               <scale>108140</scale>
               <scale>54070</scale>
               <scale>27035</scale>
               <scale>13517</scale>
          </scales>
```

```
        <units>degrees</units>
        <maxExtent>
            <leftX>-180</leftX>
            <bottomY>-90</bottomY>
            <rightX>180</rightX>
            <topY>90</topY>
        </maxExtent>
        <dpi>72</dpi>
    </map>
    <layers>
        <baselayer transparent="false">
            <name>basic</name>
            <type>WMS</type>
            <url>http://labs.metacarta.com/wms/vmap0</url>
            <maxExtent>
                <leftX>-180</leftX>
                <bottomY>-90</bottomY>
                <rightX>180</rightX>
                <topY>90</topY>
            </maxExtent>
            <layer>basic</layer>
        </baselayer>
        <baselayer transparent="false">
            <name>TM_WORLD_BORDERS_SIMPL-0.3</name>
            <type>WMS</type>
            <url>PG:host=alfarrabio.di.uminho.pt dbname=mygisdb
                user=mike password=qwerty</url>
            <maxExtent>
                <leftX>-180</leftX>
                <bottomY>-90</bottomY>
                <rightX>180</rightX>
                <topY>83.5703</topY>
            </maxExtent>
            <layer>TM_WORLD_BORDERS_SIMPL-0.3</layer>
        </baselayer>
        <overlayer transparent="true">
```

```
        <name>river</name>
        <type>WMS</type>
        <url>PG:host=alfarrabio.di.uminho.pt dbname=mygisdb
            user=mike password=qwerty</url>
        <maxExtent>
            <leftX>-8.86922</leftX>
            <bottomY>31.476</bottomY>
            <rightX>72.0152</rightX>
            <topY>58.5495</topY>
        </maxExtent>
        <layer>hydro-aqueduct-canal-l</layer>
    </overlayer>
    <overlayer transparent="true">
        <name>world</name>
        <type>WMS</type>
        <url>PG:host=alfarrabio.di.uminho.pt dbname=mygisdb
            user=mike password=qwerty</url>
        <maxExtent>
            <leftX>-25000000</leftX>
            <bottomY>-25000000</bottomY>
            <rightX>25000000</rightX>
            <topY>25000000</topY>
        </maxExtent>
        <layer>world_cities</layer>
        <layer>ind-storage-p</layer>
    </overlayer>
    <overlayer transparent="true">
        <name>sc500k</name>
        <type>WMS</type>
        <url>http://mapas.igeo.pt/wms/sc500k?</url>
        <maxExtent>
            <leftX>-10.1943</leftX>
            <bottomY>36.7636</bottomY>
            <rightX>-5.71067</rightX>
            <topY>42.2787</topY>
        </maxExtent>
```

```
            <layer>sc500k</layer>
        </overlayer>
    </layers>
    <data>
        <connection password="qwerty" host="alfarrabio.di.uminho
            .pt" user="mike" dbname="mygisdb">
            <table name="world_cities">
                <column name="name">
                    <widgets>
                        <widget enabled="true">search</widget>
                        <widget enabled="false">editable</widget
                            >
                        <widget enabled="false">tooltip</widget>
                    </widgets>
                    <type>string</type>
                </column>
            </table>
            <table name="hydro-aqueduct-canal-l">
                <column name="name">
                    <widgets>
                        <widget enabled="false">search</widget>
                        <widget enabled="true">editable</widget>
                        <widget enabled="false">tooltip</widget>
                    </widgets>
                    <type>string</type>
                </column>
            </table>
            <table name="TM_WORLD_BORDERS_SIMPL-0.3">
                <column name="ele_popu">
                    <widgets>
                        <widget enabled="false">search</widget>
                        <widget enabled="false">editable</widget
                            >
                        <widget enabled="true">tooltip</widget>
                    </widgets>
                    <type>numeric</type>
```

```
                </column>
                <column name="inflation">
                    <widgets>
                        <widget enabled="false">search</widget>
                        <widget enabled="false">editable</widget
                            >
                        <widget enabled="true">tooltip</widget>
                    </widgets>
                    <type>numeric</type>
                </column>
                <column name="pays">
                    <widgets>
                        <widget enabled="false">search</widget>
                        <widget enabled="false">editable</widget
                            >
                        <widget enabled="true">tooltip</widget>
                    </widgets>
                    <type>string</type>
                </column>
                <column name="population">
                    <widgets>
                        <widget enabled="false">search</widget>
                        <widget enabled="false">editable</widget
                            >
                        <widget enabled="true">tooltip</widget>
                    </widgets>
                    <type>numeric</type>
                </column>
            </table>
        </connection>
    </data>
    <header>
        <content></content>
        <height>74</height>
    </header>
</mapfish>
```

Bibliography

[AHB02] Aa. Alesheikh, H. Helali, and Ha. Behroz. Web gis: Technologies and its applications, 2002.

[Bau09] Peter Baumann. The ogc web coverage processing service (wcps) standard. *GeoInformatica*, 2009.

[BM02] Maria Antonia Brovelli and Diego Magni. An archaeological web GIS application based on Mapserver and PostGIS. In M. Ciolli and P. Zatelli, editors, *Proceedings of the Open Source Free Software GIS - GRASS users conference 2002*, September 2002. *http : //www.ing.unitn.it/ ∼ grass*.

[CGR+08] Scott Crowther, Abe Guerra, George Raber, Angel E. Tomala-Reyes, and Murali Vridhachalam. Building a geospatial information system, Part 1: Understanding the basics. Technical report, IBM, October 2008.

[DG10] Geomajas Developers and Geosparc. *Getting started with Geomajas.* Geosparc nv, 2010.

[Dra04] Suzana Dragicevic. The potential of Web-based GIS". *Journal of Geographical Systems*, 6:79–81, June 2004.

[ESR10] ESRI. Gis dictionary, 2010. `http://resources.arcgis.com/glossary/term/1042`.

[Kro05] Bill Kropla. *Beginning MapServer: Open Source GIS Development (Expert's Voice in Open Source).* Apress, August 2005.

[LGMR05] Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science.* John Wiley & Sons, April 2005.

[MHS05] Marjan Mernik, Jan Heering, and Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.

[Mit05] Tyler Mitchell. *Web Mapping Illustrated.* O'Reilly Media, Inc., June 2005.

[MLM00] Makoto Murata, Dongwon Lee, and Murali Mani. Taxonomy of xml schema languages using formal language theory, 2000.

[NM08] Markus Neteler and Helena Mitasova. *Open Source GIS: A GRASS GIS Approach.* Springer, New York, third edition, November 2008.

[NOJGN99] Merritt R. Nelson, Thomas V. Orum, Ramon Jaime-Garcia, and Athar Nadeem. Applications of geographic information systems and geostatistics in plant disease epidemiology and management. *Plant Disease*, 83(4):308–319, 1999.

[OPKJ09] Leslie M. Orchard, Ara Pehlivanian, Scott Koon, and Harley Jones. *Professional JavaScript Frameworks: Prototype,YUI, ExtJS, Dojo and MooTools.* Wrox Press Ltd., Birmingham, UK, UK, 2009.

[PSM01] Leitung Prof, Dr. D. Saupe, and Mario Melle. Atlas2000, 2001.

[PTM07] Scott Pezanowski, Brian Tomaszewski, and Alan M. Maceachren. An open geospatial standards-enabled google earth application to support crisis management, 2007.

[Ram07] Paul Ramsey. The State of the Open Source GIS. Open Source GIS, September 2007.

[Ram09] Paul Ramsey. *PostGIS Manual.* Refractions, 2009.

[Ray03] Erik T. Ray. *Learning XML.* O'Reilly Media, Inc., 2003.

[Tit03] Ed Tittel. *XML.* Bookman, 2003.

[UCF+05] Helton Nogueira Uchoa, Renata Juliana Cristal Coutinho, Paulo Roberto Ferreira, Luiz Carlos Teixeira Coelho Filho, and Jorge Luï¿½s Nunes e Silva Brito. Anï¿½lise do mï¿½dulo postgis (opengisï¿½) para armazenamento e tratamento de dados geogrï¿½ficos com alta performance e baixo custo. In *Anï¿½lise do mï¿½dulo PostGIS (OpenGISï¿½) para armazenamento e tratamento de dados geogrï¿½ficos com alta performance e baixo custo*, 2005.

[vDKV00] Arie van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, 2000.

[Vli02] Eric van der Vlist. *XML Schema.* O'Reilly Media, Inc., 2002.

[WH09] Eric B. Wolf and Kevin Howe. Web-client based distributed generalization and geoprocessing. In *Advanced Geographic Information Systems & Web Services, 2009. GEOWS '09. International Conference on*, pages 123–128, 2009.

[YS07] Seokchan Channy Yun and Daum Communications Corp Seoul. The 11 th international seminar on gis oct. 24, 2007 the user-participated geospatial web as open platform 1, 2007.

[Zam09] Frank Zammetti. *Practical Ext JS Projects with Gears.* Apress, Berkely, CA, USA, 2009.

[ZZL+09] Zhonghai Zhou, Bin Zhou, Wenwen Li, Brian Griglak, Carmen Caiseda, and Qunying Huang. Evaluating query performance on object-relational spatial databases. *Computer Science and Information Technology, International Conference on*, 0:489–492, 2009.

Glossary

A

`API` Application Programming Interface.

`Application Server` A computer program that receives user requests through a client application and returns results to the client.

`Architecture` The internal design of an application or software package; the way software or hardware components are organized into a functioning unit.

`Atlas` A collection of maps usually related to a particular area or theme and presented together. Examples of atlases include world atlases, historical atlases, and biodiversity atlases.

B

`Base Layer` A data layer in a GIS to which all other layers are geometrically referenced.

C

`Cartography` The art and science of expressing graphically, usually through maps, the natural and social features of the earth.

D

`Data` Any collection of related facts arranged in a particular format; often, the basic elements of information that are produced, stored, or processed by a computer.

`DTD` Document Type Definition (DTD) is a set of markup declarations that define a document type for SGML-family markup languages (SGML, XML, HTML).

F

`feature` Is an entity with a geographic location, typically describable by (for example) points, arcs, or polygons, which comprises vector source data. It is exchangeable in Geography Markup Language (GML) format.

`framework` A framework is a basic conceptual structure used to solve or address complex issues.

G

`Geographic Data` Information describing the location and attributes of things, including their shapes and representation. Geographic data is the composite of spatial data and attribute data.

`geospatial` Is a term widely used to describe the combination of spatial software and analytical methods with terrestrial or geographic datasets.

`GIS` Information system that integrates, stores, edits, analyzes, shares, and displays geographic information.

`GML` Acronym for Geography Markup Language. An OpenGIS Implementation Specification designed to store and transport geographic information. GML is a profile (encoding) of XML.

`GUI` Acronym for graphical user interface. A software display of program options that allows a user to choose commands by pointing to icons, dialog boxes, and lists of menu items on the screen, typically using a mouse. This contrasts with a command line interface in which control is accomplished via the exchange of strings of text.

`GWT` Google Web Toolkit is an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java.

I

`Interface` Point of interaction between two systems. Also used to define the communication mode (graphical or textual) between the computer and the user.

J

`JSON` Acronym for JavaScript Object Notation. A lightweight, human-readable data interchange format. An alternative to XML, JSON is language independent but relies on common programming language structures such as objects and arrays.

L

`Layer` A spatial dataset containing a common feature type. Layers are also referred to as coverages or themes.

`Library` In object-oriented programming, a logical grouping of classes, usually with a header section that lists the classes in the library.

M

`Map` Any graphical representation of geographic or spatial information.

`Map Cache` A setting that allows temporary storage of geodatabase from a given map extent in the desktop computer's RAM, which may result in performance improvements for editing, feature rendering, and labeling.

`Map Extent` The limit of the geographic area shown on a map, usually defined by a rectangle. In a dynamic map display, the map extent can be changed by zooming and panning.

`Map Unit` The ground unit of measurement - for example, degrees, feet, miles, meters, or kilometers - in which coordinates of spatial data are stored.

`Max Extent` The maximum bounding rectangle (in x,y coordinates) of an on-screen map. Users cannot zoom out beyond the max extent.

O

`Overlay` A spatial operation in which two or more maps or layers registered to a common coordinate system are superimposed, either digitally or on a transparent material, for the purpose of showing the relationships between features that occupy the same geographic space.

`Overview Map` A generalized, smaller-scale map that shows the limits of another map's extent along with its surrounding area.

P

`PHP` Hypertext Preprocessor is a widely used, general-purpose scripting language that was originally designed for web development to produce dynamic web pages.

`projection` A map projection is one of many methods used to represent the 3-dimensional surface of the earth or other round body on a 2-dimensional plane in cartography (mapmaking). This process is typically, but not necessarily, a mathematical procedure (some methods are graphically based).

`Publish` To produce data and information in a distributable format, such as digital, Internet, or print media.

`Pylons` Pylons is an open source web application framework written in Python.

`Python` Python is an interpreted, general-purpose high-level programming language whose design philosophy emphasizes code readability. Python aims to combine remarkable power with very clear syntax, and its standard library is large and comprehensive.

R

`Raster Data` A spatial data model that defines space as an array of equally sized cells arranged in rows and columns, and composed of single or multiple bands. Each cell contains an attribute value and location coordinates. Unlike a vector structure, which stores coordinates explicitly, raster coordinates are contained in the ordering of the matrix. Groups of cells that share the same value represent the same type of geographic feature.

S

`Scale` The ratio or relationship between a distance or area on a map and the corresponding distance or area on the ground, commonly expressed as a fraction or ratio. A map scale of 1/100,000 or 1:100,000 means that one unit of measure on the map equals 100,000 of the same unit on the earth.

`Shapefile` A vector data storage format for storing the location, shape, and attributes of geographic features. A shapefile is stored in a set of related files and contains one feature class.

`SLD` Styled Layer Descriptor is an XML schema specified by the Open Geospatial Consortium (OGC) for describing the appearance of map layers.

`Spatial Data` Information about the locations and shapes of geographic features and the relationships between them, usually stored as coordinates and topology.

`Style` An organized collection of predefined colors, symbols, properties of symbols, and map elements. Styles promote standardization and consistency in mapping products.

V

`Vector Data` A coordinate-based data model that represents geographic features as points, lines, and polygons. Each point feature is represented as a single coordinate pair, while line and polygon features are represented as ordered lists of vertices. Attributes are associated with each vector feature, as opposed to a raster data model, which associates attributes with grid cells.

W

`Web Application` A software program that communicates via the World Wide Web and delivers Web-based information to the user in HTML format. Web applications are typically used to add customization and interactivity to Web pages. Web applications may also be called Web-based applications.

`Widget` Reusable element of a graphical user interface (GUI) that displays an information arrangement and provides standardized data manipulation.

`Wiki` Website that allows the easy creation and editing of any number of interlinked web pages via a web browser using a simplified markup language or a WYSIWYG text editor.

`WYSIWYG` Acronym for What You See Is What You Get. This term refers to the editing tools providing visual feedback of the desired end result of an editing operation while the process is still underway.

X

`XML` Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

`XML Schema (XSD)` XML Schema (XSD), published as a W3C recommendation in May 2001, is one of several XML schema languages. XSD can be used to express a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema.

`XSL` eXtensible Stylesheet Language (XSL) is a family of recommendations for defining XML document transformation and presentation.