UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

CGIS Map Production
Architectural Design

Compiled By

Siyabonga Magubane - u15289347
Bernard van Tonder - u15008992
Boikanyo Modiko - u15227678
Cian Steenkamp - u15095682
Robert Trankle - u15092454

2017
#include

Contents

List of Figures

1 Introduction

This chapter of the document is used to identify the Purpose of the document, as well as all references used.

1.1 Purpose

The main purpose of this document is to identify eah sub-system of the CGIS Map production system. It will address the architectural design of each subsystem, by using design patterns and models to conceptualize and visualize the system.

The choice of technologies, will also be provided.

2 Module Design

2.1 Access Module

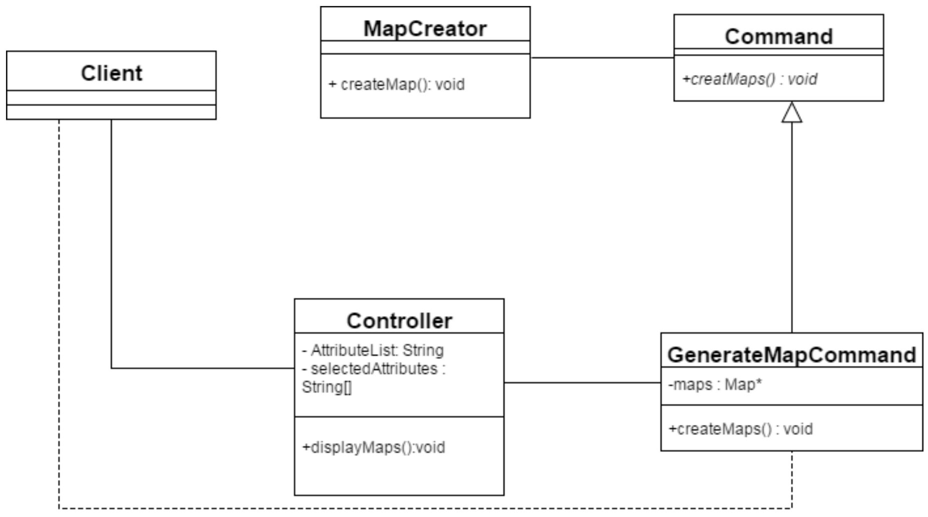

Figure 1: Access subsystem Class Diagram

The above diagram represents the class diagram of the Access subsystem. This subsystem is used to facilitate the communication between the other subsystems on the server side, and the user on the client side.

The design pattern used was the Command pattern. It was chosen because it allows for the encapsulation of a request as an object. Thereby allowing the system parameterize clients with different requests, which would be the different attributes chosen.

The Controller class is used to communicate directly with the client, and display the maps once they have been generated.

The MapCreator class is linked to the generate Maps subsystem, and is part of that system. It is the link between the two subsystems, and is summarized just to show the relation.

The GenerateMapCommand is the concrete class of the Command class(which is a virtual class). It contains an array of Maps which will be generated and the function call to create the Maps.

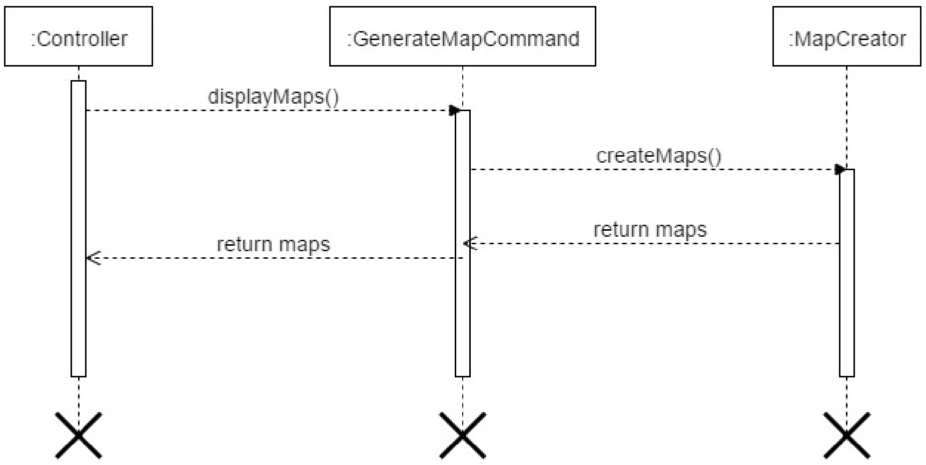

Figure 2: Access subsystem Sequence Diagram

The figure above depicts the sequence diagram for the access subsystem. The sequence diagram illustrates the sequence of interaction amongst objects (or classes). The Controller class/object will call displayMaps() which will invoke the generateMapCommand object which in turn calls the createMaps() function which represents the factory method used to generate/prepare the thematic maps. The generated map(s) will then be returned.

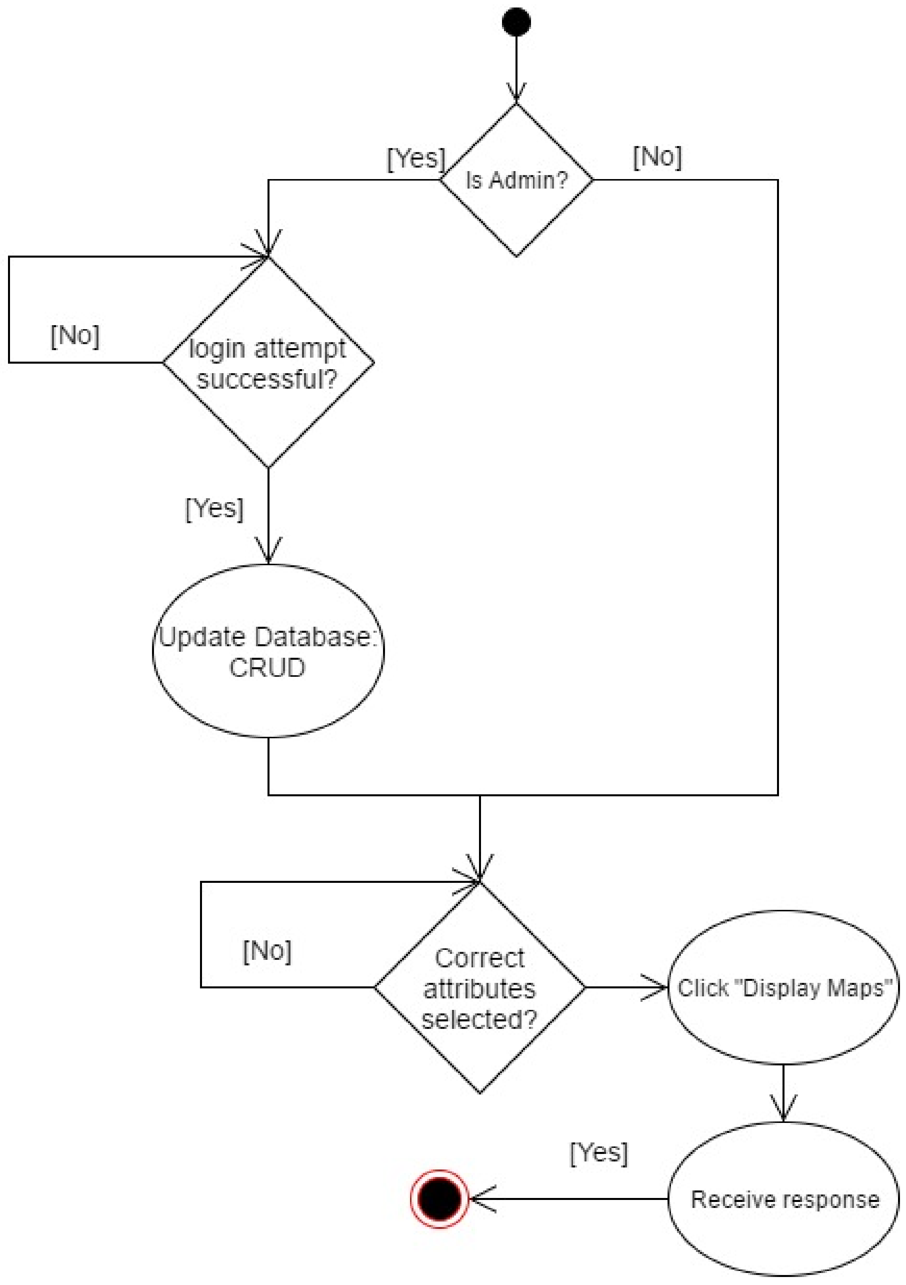

Figure 3: Access subsystem Activity Diagram

The activity diagram above models the flow control from one activity to another. The activity diagram models the activity flow of the system. If the user is an administrator then he/she will need to login in order to be able to update the database (CRUD). Any user (administrator or normal user) can then select the attributes to be used to generate the thematic maps. Once the user selected the attributes he/she can click the "Display Maps" button to generate the 3 thematic maps.

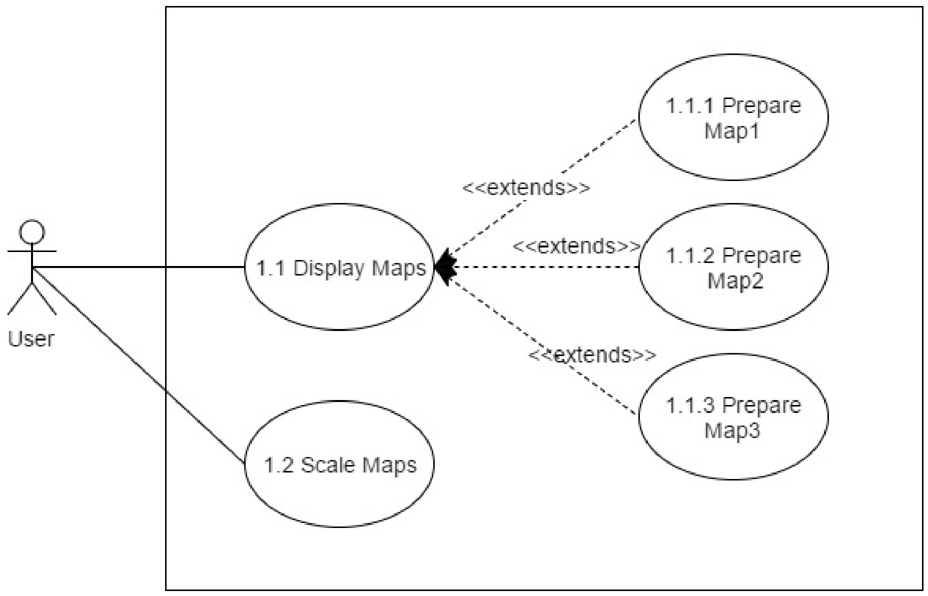

Figure 4: Access subsystem Use Case Diagram

2.2 Database Manager Module

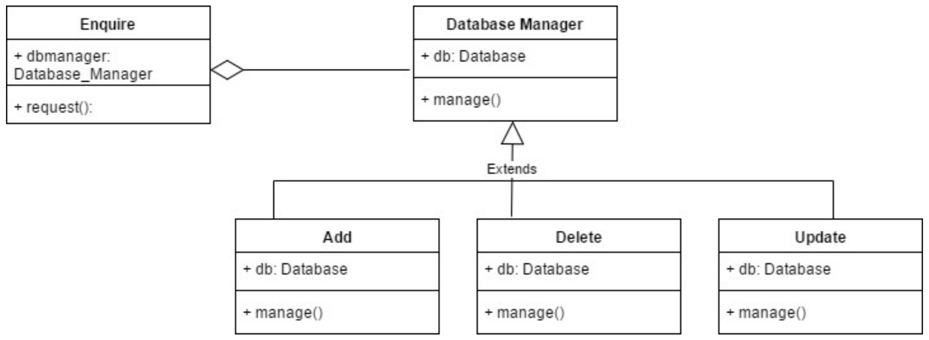

Figure 5: Database Manager subsystem Class Diagram

The above diagram represents the class diagram of the database Module subsystem. This subsystem is used to facilitate communication between the other database module and the other subsystems.

The design pattern used in this module is the state design pattern. The Database Manager class is the State class, Enquire is the context class and the add, delete and update are the ConcreteState classes. Enquire sends a request to the data base manager and the data base manager updates the dataset.

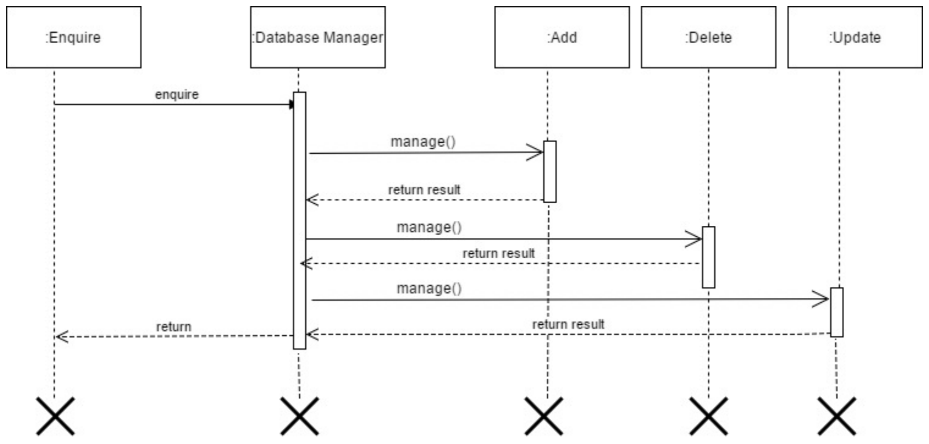

Figure 6: Database Manager subsystem Sequence Diagram

The figure above depicts the sequence diagram for the Database Manager subsystem. The sequence diagram illustrates the sequence of interaction amongst classes.

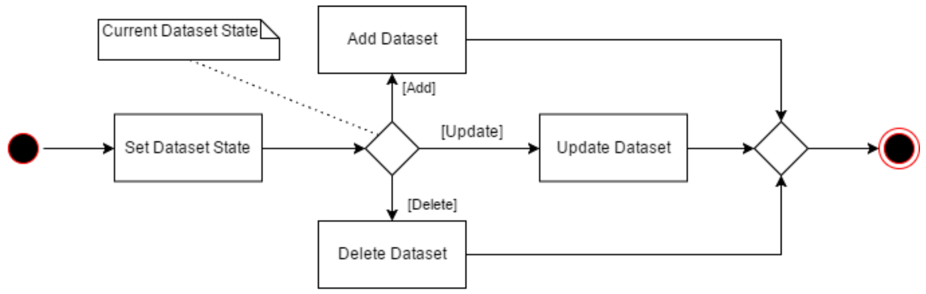

Figure 7: Database Manager subsystem Activity Diagram

The behavior of the Database Manager depends on its state. The state is set, and the relevant action is executed.

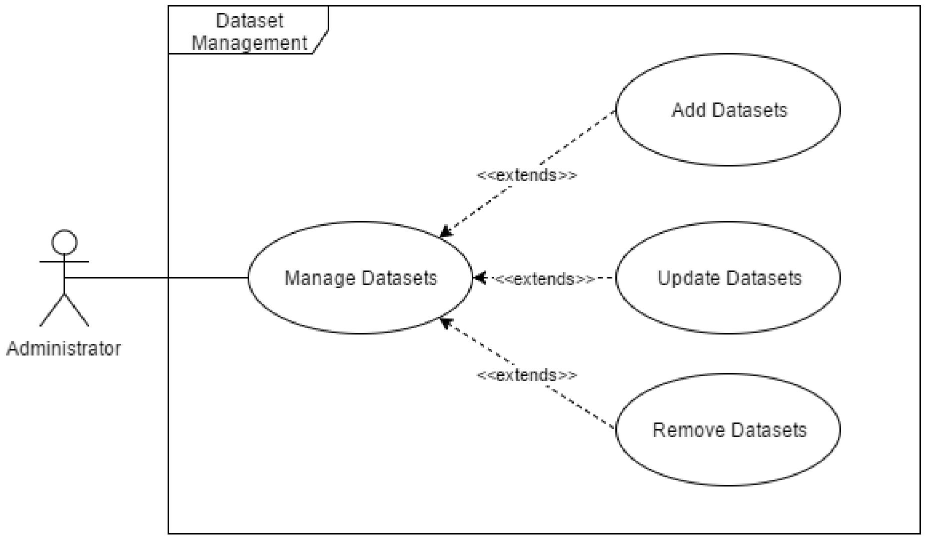

Figure 8: Database Manager subsystem Use Case Diagram

2.3 Map design Module

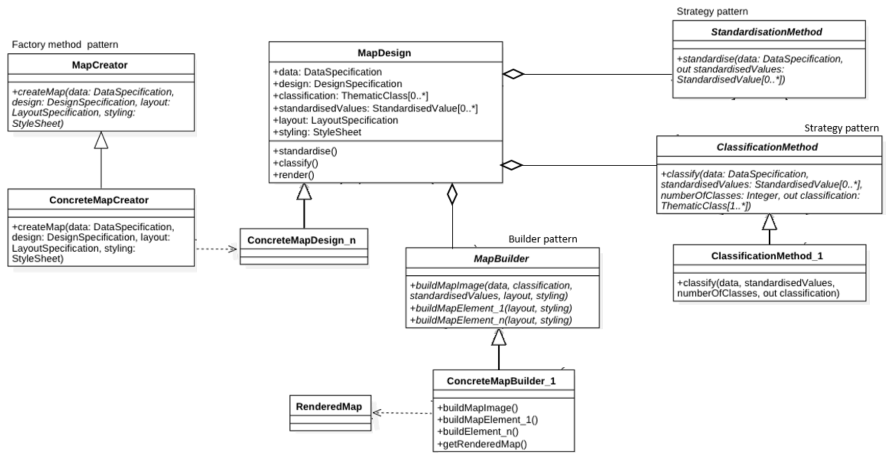
Factory method pattern
MapCreator
+createMap(data: DataSpecification, design: DesignSpecification, layout: LayoutSpecification, styling: StyleSheet)
ConcreteMapCreator
+createMap(data: DataSpecification, design: DesignSpecification, layout: LayoutSpecification, styling: StyleSheet)
MapDesign
+data: DataSpecification
+design: DesignSpecification
+classification: ThematicClass[0..*]
+standardisedValues: StandardisedValue[0..*]
+layout: LayoutSpecification
+styling: StyleSheet
+standardise()
+classify()
+render()
ConcreteMapDesign_n
Strategy pattern
StandardisationMethod
+standardise(data: DataSpecification, out standardisedValues: StandardisedValue[0..*])
Strategy pattern
ClassificationMethod
+classify(data: DataSpecification, standardisedValues: StandardisedValue[0..*], numberOfClasses: Integer, out classification: ThematicClass[1..*])
ClassificationMethod_1
+classify(data, standardisedValues, numberOfClasses, out classification)
Builder pattern
MapBuilder
+buildMapImage(data, classification, standardisedValues, layout, styling)
+buildMapElement_1(layout, styling)
+buildMapElement_n(layout, styling)
ConcreteMapBuilder_1
+buildMapImage()
+buildMapElement_1()
+buildElement_n()
+getRenderedMap()
RenderedMap

Figure 9: Map design Class Diagram

The above diagram represents the class diagram of the map design subsystem. This subsystem is used generate the thematic maps.

The design patterns used were the Factory Method, Builder pattern, and the Strategy pattern.

The Factory method defines an interface for creating maps.

The Builder pattern allows multiple classes to help in the creation of mapBuilder objects. The strategy allows the definition of a family of classificationMethod algorithms, encapsulates each one, and makes them interchangeable.

The Strategy pattern allows the standidisationMethod to vary independently from the clients that use it.

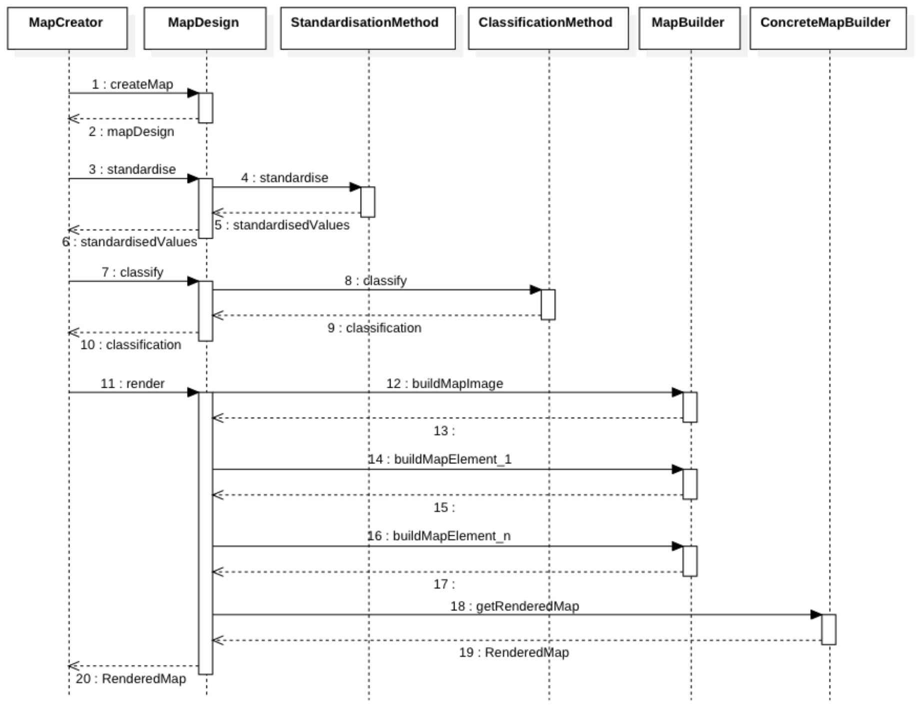

Figure 10: Map design Sequence Diagram

The figure above depicts the sequence diagram for the map design module. The sequence diagram illustrates the sequence of interaction amongst objects (or classes).

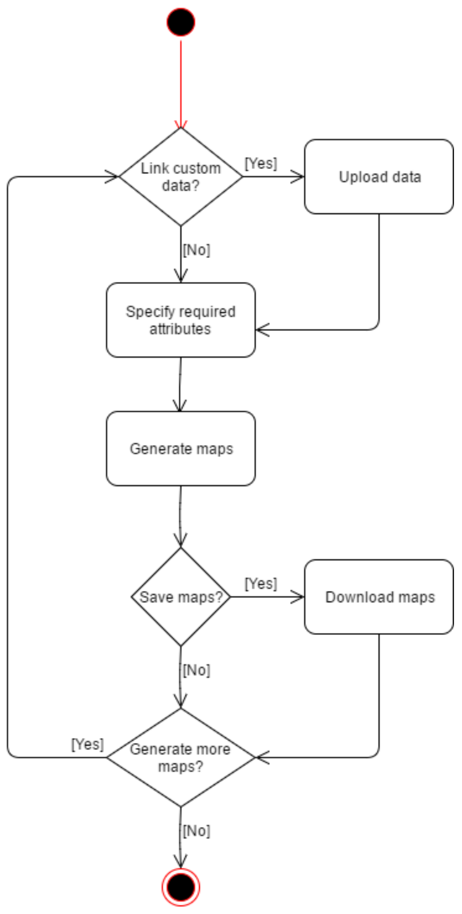

Figure 11: Map design Activity Diagram

The figure above depicts the activity diagram for the map design module. The user has an option to link their own database to the system or use the data which has already been provided.

After specifying the data that will be used, the user can then select which attributes they would like their maps to contain. Based on the attributes selected, the system will generate three different thematic maps for the user to view.

The user can then save the specific maps by downloading them in a format of their choice.

This process can repeated for as many times the user deems necessary.

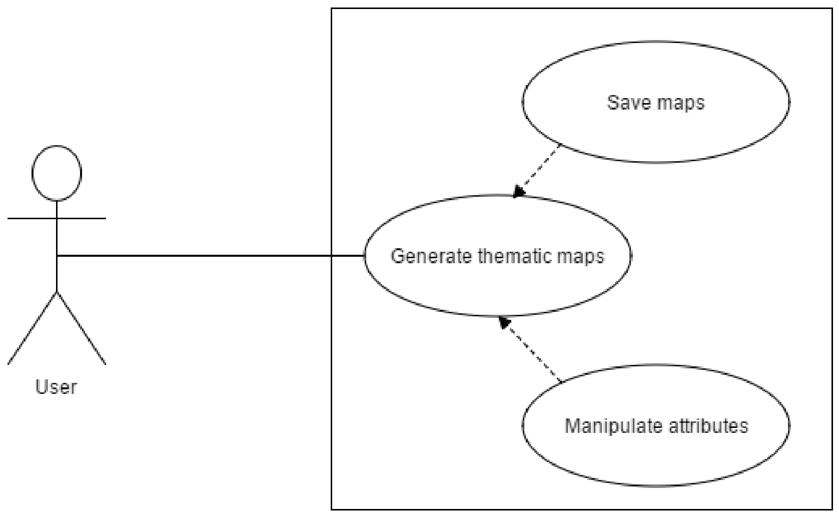

Figure 12: Map design Use Case

3 System Assessment

3.1 Deployment Diagram

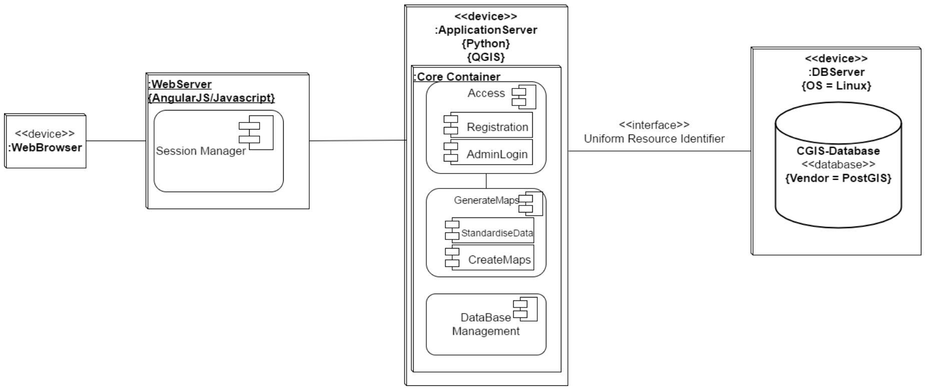

Figure 13: CGIS Deployment Diagram

3.2 Technologies Used

The technologies used will include a range of languages and Open Source Software.

The following key technologies have been identified, in which the team deems applicable to the purpose of the system :

- OpenLayers : Is an open source JavaScript library for displaying geospatial data in web browsers.
- GeoServer : Is an open source server for sharing geospatial data. Designed for interoperability, it publishes data from any major spatial data source using open standards.
- Eclipse Jetty : is a Java HTTP (Web) server and Java Servlet container.
- PostGIS : It will be used to manage and edit the Geospatial data which the admin will be able to manage, through the Database Management subsystem.
- HTML/CSS/Bootstrap : All of the following languages pertain to the creation of the front end of the system. It will be used to create the user interface, where users will be able to interact with the system.
- jQuery : jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

4 References

Source Making. 2017. Design Patterns. [ONLINE] Available at: https://www.sourcemaking.com/. [Accessed 19 June 2017].