

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Sharing Botanical Information using Geospatial Databases

João Miguel Rocha da Silva

Report of Dissertation

Master in Informatics and Computing Engineering

Supervisor: Maria Cristina de Carvalho Alves Ribeiro (Professor Auxiliar)

Co-Supervisor: João António Correia Lopes (Professor Auxiliar)

28th July, 2009

Sharing Botanical Information using Geospatial Databases

João Miguel Rocha da Silva

Report of Dissertation
Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Rosaldo José Fernandes Rossetti (Professor Auxiliar)

External Examiner: José Carlos Leite Ramalho (Professor Auxiliar)

17th July, 2009

Abstract

Botany relies heavily on specimen cataloguing. Since Carl Linnaeus established Linnaean taxonomy, the model has been used as the primary method of classification for living organisms. It is a highly structured and universally recognisable classification. This taxonomic classification is the common ground on which botanical databases are currently built. Records are organised and queried using this structure.

The current trend is for botanical information systems to move towards a different kind of information: *georeferenced information*. There are increasing efforts to retrieve spatial information from current botanical records, gathered from historic collection catalogues. With the growing interest in spatial information, there is also the need for *spatial querying*. Using current solutions, users are limited to searching for specimens in specific countries or using free text search on specific locality fields. All these solutions are text-based querying approaches. We propose an alternative: using a kind of database specifically designed for querying spatial data, allowing for more efficiency and flexibility than text-based queries.

We aim at providing a storage and querying service for applications that need it, in exchange for the collection of specimen data. This is done in such a way that users -botanical databases, applications, or botanical parks- can send specimen information easily. Applications can also query the central system in a flexible, yet powerful and technology-independent way. They do not need to know about the implementation details of the underlying geospatial database nor make serious changes to their current systems. Also, information exchange is of the utmost importance and the system must be able to interpret and provide information in standard formats.

By providing these geobotanical services, the proposed repository can become a hub for botanical information exchange, linking several databases. At the same time, it can support various applications, providing clues about species propagation and growth patterns. This is particularly interesting for users who want to combine this data with other types of geospatial information.

To demonstrate the usefulness of this repository in a real scenario, a prototype client for a botanical park was developed. *FEUP Park*, as it was named, supports specimen insertion along with other the common operations of a botanical park, using the repository's geospatial querying capabilities.

Resumo

O domínio da botânica vive da catalogação de espécimes. Desde a invenção da taxonomia de Lineu (Carl Linnaeus) que este modelo tem sido usado como o principal meio de classificação dos seres vivos. Este método permite estruturar as classificações produzidas, tornando-as reconhecíveis em todo o mundo.

Esta classificação taxonómica é a pedra basilar sobre a qual as bases de dados botânicas actuais estão assentes. Os seus registos são organizados com recurso a esta estrutura, e as bases de dados são interrogadas utilizando valores textuais.

A tendência actual dos sistemas de informação ligados à botânica é no sentido da utilização de outro tipo de informação: a informação *geo-referenciada*. Existem cada vez mais esforços no sentido de deduzir informação geo-referenciada a partir de registos botânicos já existentes que são, geralmente, recolhidos a partir de catálogos históricos de espécimes botânicos. Com o crescimento do interesse neste tipo de informação espacial, surge a necessidade de utilizar mecanismos de interrogação também orientados a esse tipo de informação. Este tipo de interrogações chamam-se *interrogações espaciais*.

As soluções actuais permitem aos seus utilizadores a pesquisa de espécimes com recurso a pesquisa livre em texto; esse texto poderá conter pontos de referência que ajudam a localizar os espécimes pretendidos. No entanto, acreditamos que existe uma abordagem diferente. O que se propõe é a concepção de um sistema capaz de realizar consultas muito mais completas, complexas e precisas; este sistema tem que ser flexível, aberto e eficiente, pois é especializado em dados geo-espaciais.

O objectivo principal deste *repositório* de informação botânica é proporcionar serviços de *interrogação espacial* a outros sistemas, em troca da colecção de dados de espécimes botânicos. Para atingir esse objectivo, a inserção de informação por parte dos utilizadores é aberta e simples. O processo de interrogação dos dados no repositório é também muito flexível e ao mesmo tempo poderoso, pautando-se pela ausência de restrições ao nível das tecnologias envolvidas. Isto permitirá aos futuros clientes do sistema abstraírem-se dos detalhes de implementação da base de dados espacial que serve de suporte ao sistema e ao mesmo tempo manter os seus sistemas actuais quase intactos.

Alem disto, a troca de informação é da maior importância neste sistema, que deverá trocar informação, utilizando para isso as representações de informação "standard" do domínio.

A oferta destes serviços geo-botânicos transformará este repositório num centro de troca de informação botânica, ligando todas estas bases de dados. Também suportará outras aplicações, ajudando a descobrir padrões de propagação de espécies ou do seu crescimento, por exemplo. Estas funcionalidades tornam-se particularmente interessantes para utilizadores que desejam combinar esta informação com outra informação geo-referenciada. Para demonstrar a utilidade deste repositório num cenário real, foi desenvolvido um

prótipo de aplicação cliente para um parque botânico. O *FEUP Park*, como foi chamado, suporta as operações mais comuns de um parque botânico, utilizando para isso as capacidades de *interrogação geoespacial* do repositório.

Acknowledgements

I would first like to thank my friends, my team-mates and partners here in this great Faculty of Engineering, for all their support.

I would also like to thank my supervisors for this dissertation, Professor Cristina Ribeiro and Professor João Correia Lopes for their time and support. From Professor Cristina's classes on the second year of the course to being my final dissertation supervisor, we have come a long way... Thank you for making me believe I belonged here.

I also thank Professor Raúl Vidal and Professor Augusto Sousa for making this course a second home for all its students.

And last but not least, I thank my family for being always there for me.

João Miguel Rocha da Silva

*“ What a computer is to me is the most remarkable tool that we have ever come up with.
It’s the equivalent of a bicycle for our minds. ”*

Steve Jobs

Contents

1	Introduction	1
1.1	Background	1
1.1.1	Botanical Taxonomy	2
1.2	Motivation and Objectives	2
1.2.1	Why Does Botany Need GIS?	2
1.2.2	Objectives	2
1.3	Dissertation Structure	3
2	Botanical Information Systems	4
2.1	Introduction	4
2.2	Botanical Databases	5
2.2.1	The PLANTS Database	5
2.2.2	Global Biodiversity Information Facility (GBIF)	6
2.2.3	Harvard University Herbaria	7
2.2.4	The Herbarium Search	8
2.2.5	Calflora Map Viewer	8
2.2.6	BioGeomancer	10
2.3	Approximation Strategies	11
2.4	Localisation and Identification	12
2.5	Localisation of Specimens - Georeferencing	13
2.5.1	GPS	13
2.5.2	Google Earth and Google Maps	14
2.6	Identification of Specimens	14
2.6.1	RFID	14
2.7	Sum-Up and Conclusions	15
3	Introducing a Geo-Botanical Repository	16
3.1	Locality and Lack of Interoperability	16
3.2	Flexibility and Exactness	17
3.3	Standard Formats for Data Exchange	18
3.3.1	Botanical Information Exchange - Darwin Core	19
3.3.2	Geographical Information Exchange - KML	20
3.3.3	Other Formats for Geographical Information Exchange	20
3.3.4	Disadvantages of XML-based Information Exchange	20
3.4	A Different Approach - Introducing an Open Geo-Botanical Repository	21

CONTENTS

4	A botanical data model	22
4.1	Objectives of This Data Model	22
4.2	Data Model Description	22
5	The Repository's Architecture	27
5.1	Design Decisions	27
5.1.1	Repository Server	27
5.1.2	Client Scenario	28
5.2	The PlantGIS Spatial Query Language	29
5.2.1	Language Requirements	29
5.2.2	Examples of PlantGIS Queries	31
5.3	Technological Analysis	33
5.3.1	Open-Source Requirements	34
5.3.2	Object-Relational Mapping Solutions	34
5.3.3	Business Logic Implementation	38
5.3.4	Links to Outside Consumers - API	38
5.4	Overview of Selected Technologies	38
5.5	Implementation Details	40
5.5.1	Server Components	40
5.5.2	Security Concerns	42
5.6	Sum-up and conclusions	43
6	Dataset Acquisition	44
6.1	Taxonomic Classification Standards	44
6.2	Website Crawling	44
6.3	Further Use	45
7	Demonstration Client	46
7.1	Requirements	46
7.1.1	Use Cases	46
7.2	Execution Examples	46
7.2.1	Specimen Insertion	47
7.2.2	Search Occurrences	56
7.2.3	Create Tour	64
7.2.4	View Tours	66
8	Conclusions	68
8.1	Future Work - General	68
8.1.1	A Social Network on Botany	69
8.1.2	Peer Review	69
8.1.3	Paper Repository	69
8.1.4	Handheld Terminal	70
8.1.5	XML Data Compression	70
8.1.6	Data Mining as a Validation Tool	70
8.2	New features for the FEUP Park Web Application	70
8.2.1	Writing a SAX Parser	70
8.2.2	Expert Comments Via Portable Terminals	71
8.2.3	Locating the Visitor Inside the Park	71

CONTENTS

References	76
A Server Relational Schema	77
B Client Application Requirements	79
B.1 Use Case Diagrams	79
B.2 Use Case Specifications	82
C Client/Server Interaction	95
C.1 Record Insertion	96
C.2 Record Searching	97
D PlantGIS Query Language	98
D.1 The Specification	99
D.2 Example Queries	102
E Project Logbook	106

List of Figures

2.1	Complete taxonomic tree of the <i>Haplophyton A. DC.</i> species, available on the PLANTS database [oA09]	5
2.2	Prevalence of the <i>Haplophyton A. DC.</i> species on American States [oA09]	6
2.3	File format options at GBIF	7
2.4	Possible query arguments, from the HUH search page	7
2.5	Locations where the <i>Abronia elliptica</i> (<i>fragrant white sand verbena</i>) was spotted by collectors. Listed in the Harboreum [GA08b].	8
2.6	Points where the species <i>Artemisia californica</i> has been spotted (in California) [Cal]	9
2.7	Overlay capabilities in the <i>Calflora Map Viewer</i> [Cal]	9
2.8	Main query window with display, on the BioGeomancer Website [Gro09a]	10
2.9	Detailed view of the first specimen on figure 2.8 [Gro09a]. This representation suggests point-radius geographic location methods [JW04]	11
2.10	GBIF - Occurrence density zones	12
2.11	Group of occurrences in the GBIF database (KML viewed in Google Earth)	13
3.1	The Iberian Peninsula, some botanical occurrences and a polygon	18
4.1	UML class diagram for botanical specimens and their sites	23
5.1	A high-level diagram of the proposed architecture for the PlantGIS system	28
5.2	Physical diagram for a client scenario	29
5.3	The spatial operators offered by the Geometry entity (from the OpenGIS Consortium Specification [OGC95])	30
5.4	An example of a simple query, specified using the <i>PlantGIS</i> query language	31
5.5	Three polygons (A, B and C) used as arguments in PlantGIS query samples (Google Earth View)	32
5.6	PlantGIS Simple Query	32
5.7	PlantGIS Intermediate Query	33
5.8	PlantGIS Advanced Query	34
5.9	A function taken from the Avoir API [Avo09]	36
5.10	Technologies used in the server application	38
5.11	Component diagram for the repository application	41
7.1	The main insertion page	48
7.2	KML File Upload	49
7.3	Viewing the uploaded KML and the list of spatial entities	50
7.4	Selecting a polygon from the uploaded KML file	51

LIST OF FIGURES

7.5	Selecting a point from the uploaded KML file	51
7.6	Uploading an image	52
7.7	Auto-completion of taxonomic information, powered by AJAX	53
7.8	Add Species Button, available when the user selects a polygon as the location of the occurrence	54
7.9	The new taxonomic information areas are also AJAX-enabled, with support for all taxonomic levels	54
7.10	Areas to insert occurrence-related identification information	54
7.11	Submit record button	55
7.12	The main occurrence search page	57
7.13	Creating a KML file in Google Maps	58
7.14	The uploaded KML file is shown by the system. Its polygons are shown in the combo box at the bottom	58
7.15	The user can add several spatial restrictions to the query, using the polygons from the KML file	59
7.16	Common text-based criteria is also available	59
7.17	Selecting the output format for results output	59
7.18	Viewing Query Results in Google Maps	60
7.19	The user can download the query results and view them in Google Earth	61
7.20	Viewing Query Results in Google Earth. The image on the bottom shows a more close view of the records	62
7.21	Polygons where there are several species will show all the different taxonomic trees	63
7.22	Create Tours - Main Page	64
7.23	Ordering occurrences in the tour and Inserting tour information	65
7.24	Tour viewing page. The tour paths and individual specimen information is shown inside each placemark balloon.	66
7.25	Details for a specimen present in the tour (Step 4 of the tour path).	66
A.1	Relational schema of the repository database	78
B.1	Use Case diagram for Users and Authenticated Users	79
B.2	Use Case diagram for Botanists	80
B.3	Use Case diagram for Visitors	80
B.4	Use Case diagram for Gardeners	81
B.5	Use Case diagram for Journalists	81
C.1	Activity diagram showing the record insertion process	96
C.2	Activity diagram showing the search process	97
D.1	The PlantGIS Query Language Specification - Part 1	99
D.2	The PlantGIS Query Language Specification - Part 2	100
D.3	The PlantGIS Query Language Specification - Part 3	101
D.4	PlantGIS Basic Query	102
D.5	PlantGIS Intermediate Query	103
D.6	PlantGIS Advanced Query - Part 1	104
D.7	PlantGIS Advanced Query - Part 2	105

LIST OF FIGURES

E.1	Project Logbook	107
-----	---------------------------	-----

List of Tables

4.1	Varieties of two species of plants (in [Wik09f])	24
5.1	Comparison between PostgreSQL with PostGIS and MySQL with Spatial Extensions	34
5.2	Comparison between several available frameworks	35
B.1	Search Occurrences Use Case specification	82
B.2	Save Occurrences Use Case specification	83
B.3	View Tours Use Case specification	84
B.4	Save Tours Use Case specification	85
B.5	View News Use Case specification	86
B.6	Post Remark Use Case specification	87
B.7	Post Comment Use Case specification	88
B.8	Add Occurrence Use Case specification	89
B.9	Create Tour Use Case specification	90
B.10	Add Intervention Use Case specification	91
B.11	View Interventions Use Case specification	92
B.12	View Intervention Details Use Case specification	93
B.13	Post News Use Case specification	94

Abbreviations

WWW	<i>World Wide Web</i>
MVC	Model, View, Controller
DMBS	DataBase Management System
OGC	OpenGIS Consortium
SQL	Structured Query Language
LOC	Line(s) Of Code
HQL	Hibernate Query Language
CRUD	Create, Read, Update and Delete, the basic functions of persistent storage [Wik09a]
TDWG	Taxonomic Database Working Group
XML	Extensible Markup Language
KML	Keyhole Markup Language
API	Application Programming Interface
GPS	Global Positioning System
RFID	Radio-frequency Identification
RUP	IBM Rational Unified Process
DoS	Denial of Service (attack)
Wi-Fi	(Synonym) WLAN (Wireless LAN)
GIS	Geographic Information Systems
AJAX	Asynchronous Javascript and XML
UML	Unified Modelling Language
GDAL	Geospatial Data Abstraction Library
DNA	Deoxyribonucleic acid
OTS	Off-the-Shelf

Chapter 1

Introduction

Botany is built on observation, analysis and classification. In the past, these classifications were stored in large volumes, and these volumes were stored in even larger libraries. Nowadays, physical records are being replaced with database records. Written information is being replaced by bytes and impulses.

In the beginning, information systems were designed to resemble their physical counterparts, increasing the speed at which information was processed and the available storage space. This allowed the processing of larger information volumes.

Now, a different approach is being adopted in the botanical information systems' domain. No longer is the taxonomic tree the only means to classify, identify and catalogue botanical specimens. The geographic information about each specimen occurrence is becoming increasingly important. This need is driving a large demand for georeferencing and an efficient way to query this specific type of information. Geographic Information Systems (GIS) come as an answer for this need. This work covers the use of GIS in the context of Botany, their advantages and disadvantages, and proof that they can, in fact, change the way botanical information is viewed.

1.1 Background

Since the beginning of Human history, the study of plants and their characteristics has been an important area of interest. From the first treaties on the subject in 300B.C. to the gene sequencing of plants' DNA [oEN09], technology has come a long way regarding the process of plant classification. The first step in this evolution was the invention of the Linnaean Taxonomy.

1.1.1 Botanical Taxonomy

The first step in creating a systematic classification method is to provide a model for such classification.

Such a model was established in the 18th century by Carl Linnaeus for all living beings. It is a series of successive subdivisions, called *Linnaean Taxonomy*. Taxonomy relies on this series of organism groupings, called *taxons* [Wik09d].

1.2 Motivation and Objectives

1.2.1 Why Does Botany Need GIS?

Plant taxonomy and categorisation are complex and meticulously documented. Botanical Taxonomy, in particular, has a characteristic that makes it very interesting for information systems development. A hierarchical taxonomy is a *tree* structure of classifications for a given set of objects [Wik09e].

Having these characteristics, taxonomic information is very prone to structured and automated representation, as well as systematic processing.

Using GIS, one can hope to improve these cataloguing methods: Classifying plants not only by their taxonomy on a large scale, but on a *specimen-level*, and having precise geographical positioning of those specimens.

This is especially useful for botanists, botanical parks, and botanical science hobbyists. These systems could allow people to visit an 1000 acre botanical park and have a guided electronic assistant to guide them along the path of their choosing. Also, owner of parks could schedule different visiting paths for their visitors using GPS (Global Positioning System) information on the specimens present in the park.

For all these reasons, this area of development is of great interest; Not only for profit and non-profit companies (*e.g.* the botanical parks or botanical databases in need of geospatial support) but also end-users interested in botany.

1.2.2 Objectives

The current ways of structuring and querying information regarding plants' botanical properties in existing databases are limited to text-based search terms, such as their taxonomic terms: *family*, *genre*, or *scientific name*, for example.

To present a different approach, based on *spatial querying*, we have established the following goals for this study:

- Analysing the largest and most widely used web-based botanical databases, identifying their strong points and areas where they could benefit from the addition of *spatial querying* capabilities.

- Perform a study of current technologies used in GIS and specimen identification and classification.
- Devising an open¹, flexible and expandable repository for providing *spatial querying* capabilities to a broad audience of client applications.
- Providing a prototype client to demonstrate the usefulness of the spatially-enabled repository.
- Obtaining a data set from a website, via automated methods, to populate the database with sample taxonomic data.

1.3 Dissertation Structure

Chapter 2 is an analysis of several current databases on the botanical domain, their strengths and weaknesses, and how they could benefit from GIS. Chapter 3 will present the proposed repository that will address the issue of *spatial querying* and offer these features to other *non-spatial* databases. In Chapter 4, a data model for the repository is proposed. Following the data model, an architecture to implement it is shown in chapter 5. Highlights include the *PlantGIS* Query Language, especially designed for client/server communication. Chapter 6 documents the population of the taxonomy, using the Wikispecies [Wik09f] website as a source. In Chapter 7, a prototype client website is shown, demonstrating the usefulness of the repository server. It uses the repository's spatial querying as support for the implementation of its use cases. Finally, conclusions are shown in Chapter 8, along with some perspectives on future works.

¹fully *open-source* based and capable of exchanging data in standard formats

Chapter 2

Botanical Information Systems

2.1 Introduction

There are several areas of development regarding botanical information aggregation and retrieval. As such, the main emphasis of a botanical database can vary drastically, from the culinary uses of a given plant [Gre09] to its complete taxonomy [oA09].

There are several public botanical databases that use global positioning to better document their plant species [GA08a]. However, these results are approximated from historical records, being therefore inaccurate.

A good solution to make these botanical records richer is to associate information from GPS receivers to them. People without specific Information Technologies education are gradually becoming accustomed to using spatial concepts in their everyday life. To bring botanical databases closer to this user base, they could provide an open, easy to use interface for spatial information input. This would allow more organisations to share their specimen occurrence records, including their geographical information. As a consequence, the approximated records currently used would be gradually replaced by more geographically precise records.

Identification is, on par with Classification, another part of botanical specimen cataloguing. One of the technologies that can be used is Radio-Frequency Identification (RFID). RFID is a radio-wave technology used for item tagging. An application of this technology to botanical specimen identification will also be analysed.

This project takes inspiration from existing databases, incorporating their strong points. Then, it introduces a different approach at botanical information querying. This is called *spatial querying*. The use of spatial querying in the botanical domain is the main focus of this dissertation.

Classification:

Haplophyton A. DC.

Click on a scientific name below to expand it in the PLANTS Classification Report.

Kingdom	Plantae – Plants
Subkingdom	Tracheobionta – Vascular plants
Superdivision	Spermatophyta – Seed plants
Division	Magnoliophyta – Flowering plants
Class	Magnoliopsida – Dicotyledons
Subclass	Asteridae
Order	Gentianales
Family	Apocynaceae – Dogbane family
Genus	Haplophyton A. DC. – haplophyton

Figure 2.1: Complete taxonomic tree of the *Haplophyton A. DC.* species, available on the PLANTS database [[oA09](#)]

2.2 Botanical Databases

We will now present a brief analysis of current botanical databases available on the Web, showing their strong points and potential improvement possibilities for these solutions.

2.2.1 The PLANTS Database

This database offers a very complete set of information regarding taxonomic classification of a given plant. One can find the full taxonomic classification tree for each specimen on this public database. An example can be seen in Figure [2.1](#).

It is also one of the largest and more complex databases found on the Internet, being also perhaps the most complete regarding flora present in the United States:

It includes names, plant symbols, checklists, distributional data, species abstracts, characteristics, images, crop information, automated tools, onward Web links, and references. [oA09]

This database takes a first approach at geographical data storage regarding plant species. For a given species, the user can see its distribution among the several North-American states (Figure 2.2).

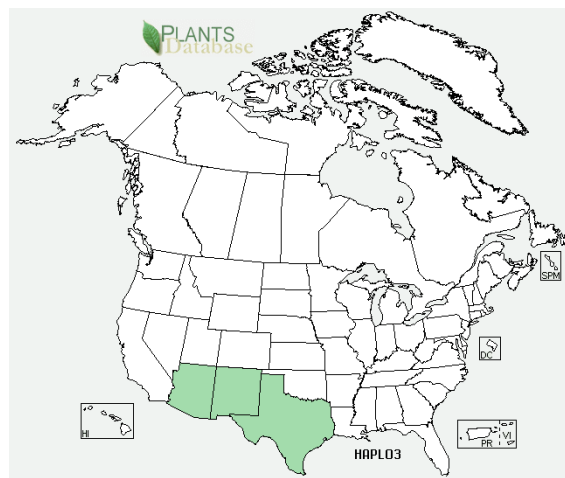


Figure 2.2: Prevalence of the *Haplophyton A. DC.* species on American States [oA09]

This approach is valuable for users seeking an "at-a-glance" preview of the distribution. It should however be improved if one wants to have more *in-depth* information, like for example the location of several specimens belonging to a particular species of plant.

2.2.2 Global Biodiversity Information Facility (GBIF)

GBIF is a portal that connects to several data sources. It gathers their data and provides them in standard formats, making the access to all these data sources easier and more transparent.

This attempt to standardise information formats appeared due to the large number of different databases currently in existence. Each of these databases has its own data that must be converted to a community data format [GN05]. To cope with these issues, great emphasis is being placed on data set format translation. The final objective is to put together a single service capable of accessing all the data and retrieving it in a standard format.

GBIF uses two data formats for its datasets. These are *Darwin Core* and KML (Keyhole Markup Language) as can be seen in Figure 2.3. In this figure, we can see a series



Figure 2.3: File format options at GBIF

of links on the right of the "Download:" header. The user can select any of these to download the results in the chosen format. This database also focuses strongly on geo-referenced information. However, this geographical information is approximated from historical records.

2.2.3 Harvard University Herbaria

The Harvard University Herbaria (HUH) are one of the 10 largest Herbaria in the world in the number of specimens [PoHC09]. Of the over 5 million specimens in these herbaria, approximately 180,000 have been recorded in a database. This database offers several criteria for querying, as seen in Figure 2.4

It is interesting to note the *Location* search field. In the instructions of use shown in the search page, we can see the following:

"For location, enter a single placename of a country or a smaller geopolitical entity."

By inputting the name of a country, such as "India" the search yielded several results, all of them having "India" in a *Country* field. As we can see, this database also uses text-based arguments to build its queries, despite saving a basic locality element.

Genus	%		Family/higher taxon	
Specific epithet	%		Infraspecific epithet	
Author	%		Year published	
		☐ Types only	Type status	any
Collector	%		Collector no.	
Location			Barcode	
		☐ with images only		

Figure 2.4: Possible query arguments, from the HUH search page

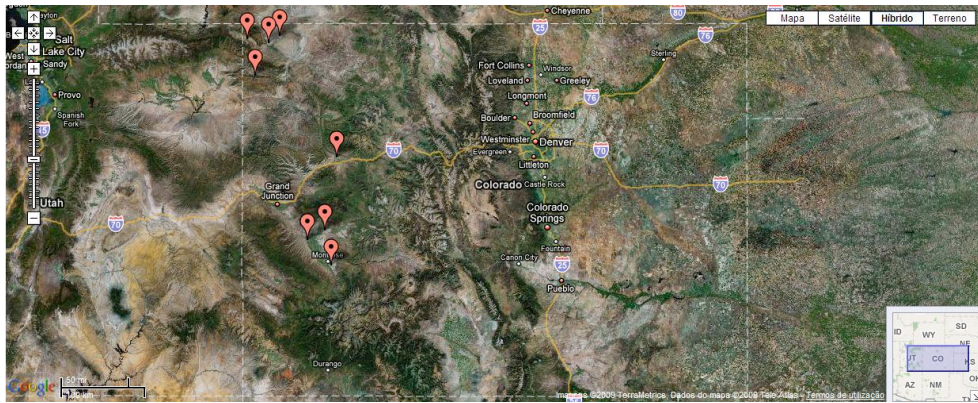


Figure 2.5: Locations where the *Abronia elliptica* (fragrant white sand verbena) was spotted by collectors. Listed in the Harboreum [GA08b].

2.2.4 The Herbarium Search

This database adds a very interesting emphasis on geographical data, storing coordinates of specimen occurrences. It has a good connection with Google Maps™, showing specimen placemarks in a real-world location (Figure 2.5). In this figure, we can see several place marks (pin-like banners) over a map. These placemarks represent the places where the plant species *Abronia elliptica* (fragrant white sand verbena) was spotted by collectors.

Given the times of the spottings reported on this website, one can say that these are based on historical recollections. They range from the 1930's to the end of the 20th century. Thus, most of the records in this database were prior to the proliferation of GPS technology and terminals.

One can conclude that there is room for improvement here. If spatial data could be obtained from GPS terminals, the records would be much more precise. This is an example of migration from regular records to georeferenced ones through coordinate approximation.

2.2.5 Calflora Map Viewer

This website [Cal] is similar to the previous one, but dedicated to the species present in the state of California.

It uses a map interface similar to Google Maps (Figure 2.6), showing several sightings of plants. This website differentiates the positions of several sightings of the same species of plant.

Figure 2.6 is taken from the *Map Viewer* section of the Calflora [Cal] website. It is the example shown in the tutorial that the website displays to introduce users to using the

Botanical Information Systems

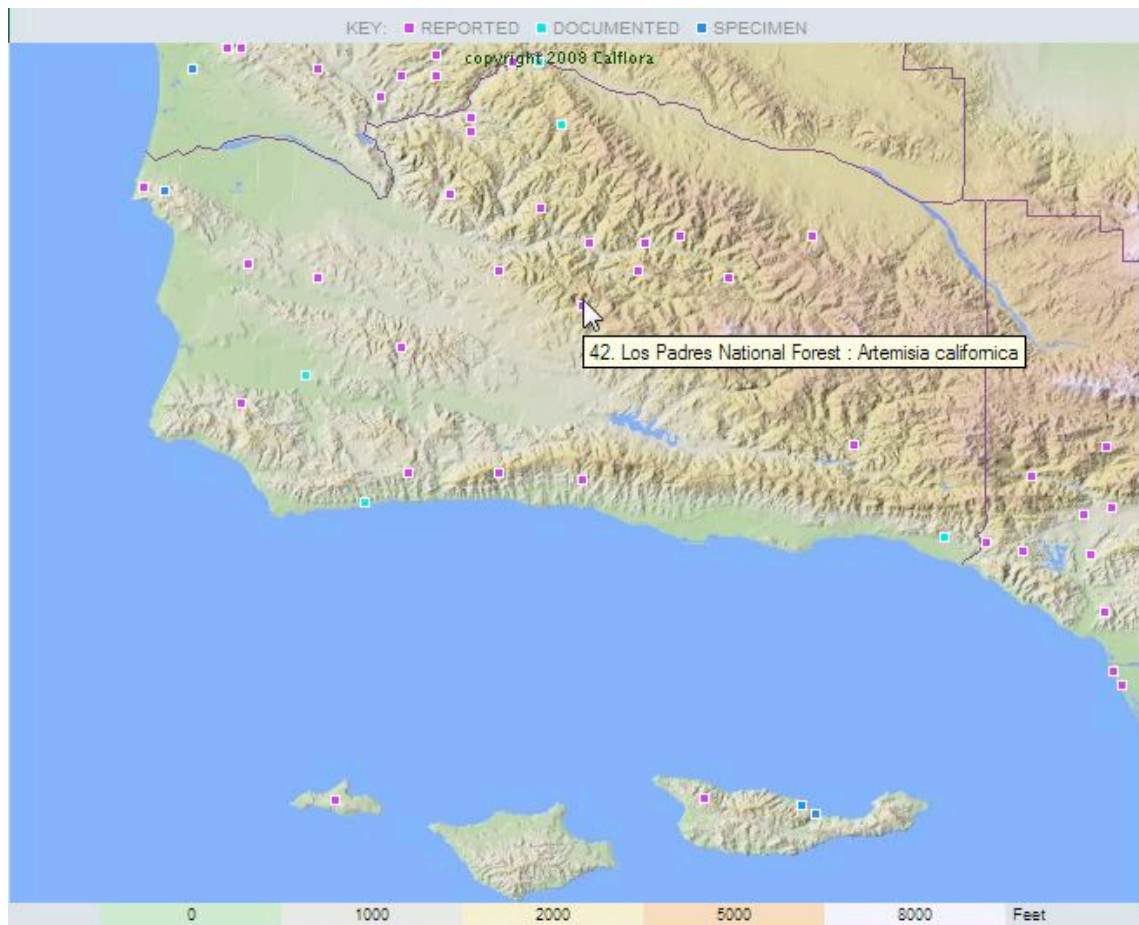


Figure 2.6: Points where the species *Artemisia californica* has been spotted (in California) [Cal]

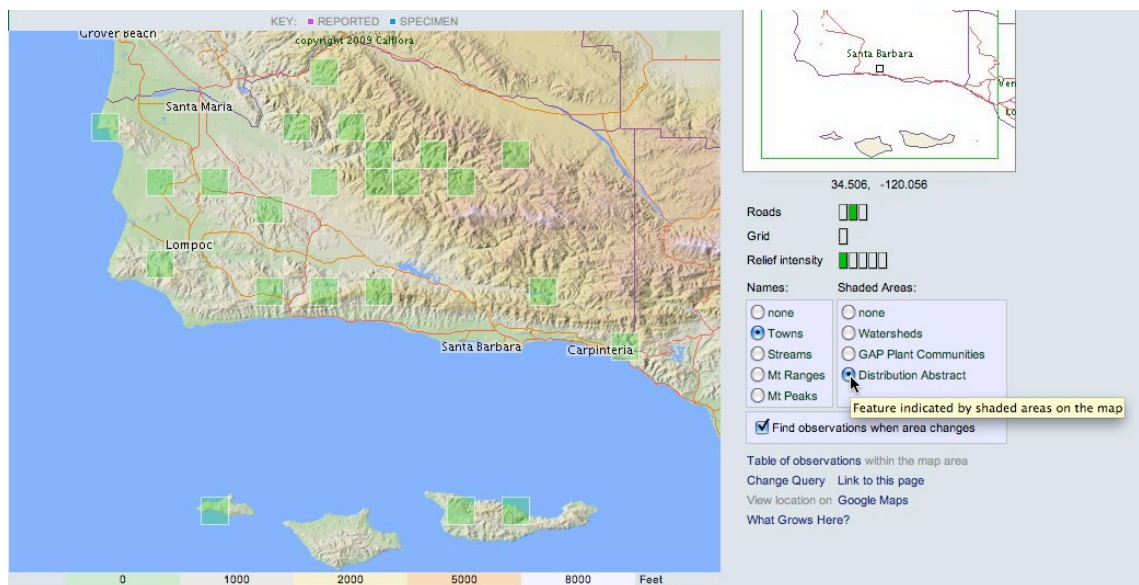


Figure 2.7: Overlay capabilities in the *Calflora Map Viewer* [Cal]

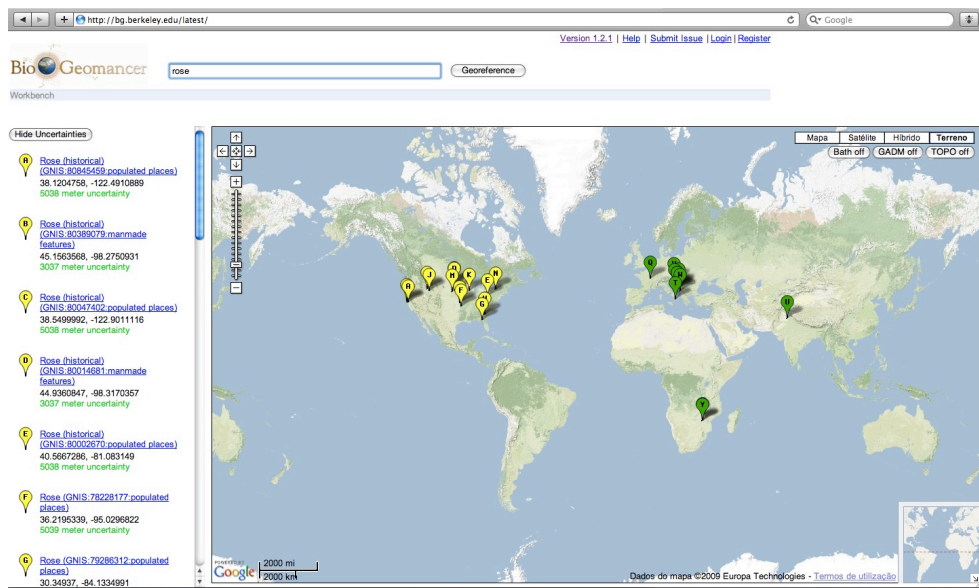


Figure 2.8: Main query window with display, on the BioGeomancer Website [Gro09a]

map viewer. It shows a series of small dots over a map of an area of California. Each of these points symbolises an observation recorded on the website.

Figure 2.7 is obtained by selecting the "Distribution abstract" option on the map visualisation's menu. It shows how this website also has the ability to summarise the occurrences on the map. It can overlay series of *squares* on the map, depicting areas inside of which the given species was sighted. This feature is very interesting on an information retrieval basis, since it can help to group and organise the sightings into higher-level entities, improving the information structuring.

2.2.6 BioGeomancer

This website showcases a standard querying system, with a large Google Maps visualizer as the main page element. When the user enters the desired keywords, the system displays a world map with a specimen point overlay as seen in Figure 2.8. Each of the pin-like banners marks the place of a botanical specimen sighting.

The interface allows for users to zoom in on the specimen's location. This in turn shows the location as a small circle delimiting the occurrence (Figure 2.9).

This website has some strong points, like the simple way users can search for what they need. The visualizer is also very simple and intuitive (built using Google Maps). Its weakest aspect is the limited amount of options that could be useful to the more advanced users like the PLANTS database, for example.

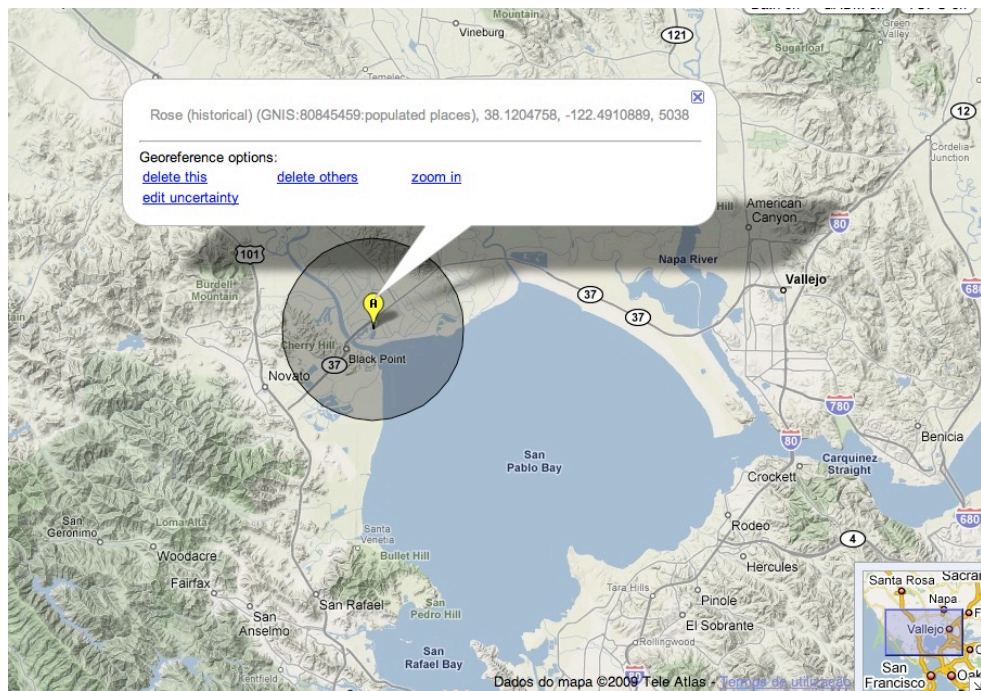


Figure 2.9: Detailed view of the first specimen on figure 2.8 [Gro09a]. This representation suggests point-radius geographic location methods [JW04]

2.3 Approximation Strategies

Since specimen occurrences recorded in current botanical information systems are almost entirely based on historical records, they need systems that approximate these textual records to geographical coordinates. These systems implement a series of manual and semi-automated techniques. These are classified as *retrospective georeferencing methods* [GN05]. An example of such automated methods is natural language processing [PCM04]. These methods are able to take written locality references such as "On Van Stadens Berg, nearest to Galgebosch"¹ and estimate geographic coordinates of the occurrence. They can also provide estimations for the of uncertainty for the approximated records. Most of current biological information databases use these approximations as their primary source of georeferenced information [SLG06].

CalJep [JHQ06] is a database reconciliation effort that aims at cross-referencing the Calflora and Jepson databases. From the analysis, it was concluded that Calflora breaks up the the map of California into several areas, named *subcounties*. Specimen occurrences are then associated to each of these small areas. They are not therefore treated as individual place marks with their own geographical coordinates but placed inside groups. GBIF also resorts to an approximation approach for adding geographical references to its records (see Figure 2.10). By opening the KML file containing the occurrences and

¹ from the HUH [PoHC09] database, for the specimen with Collection Number 4697

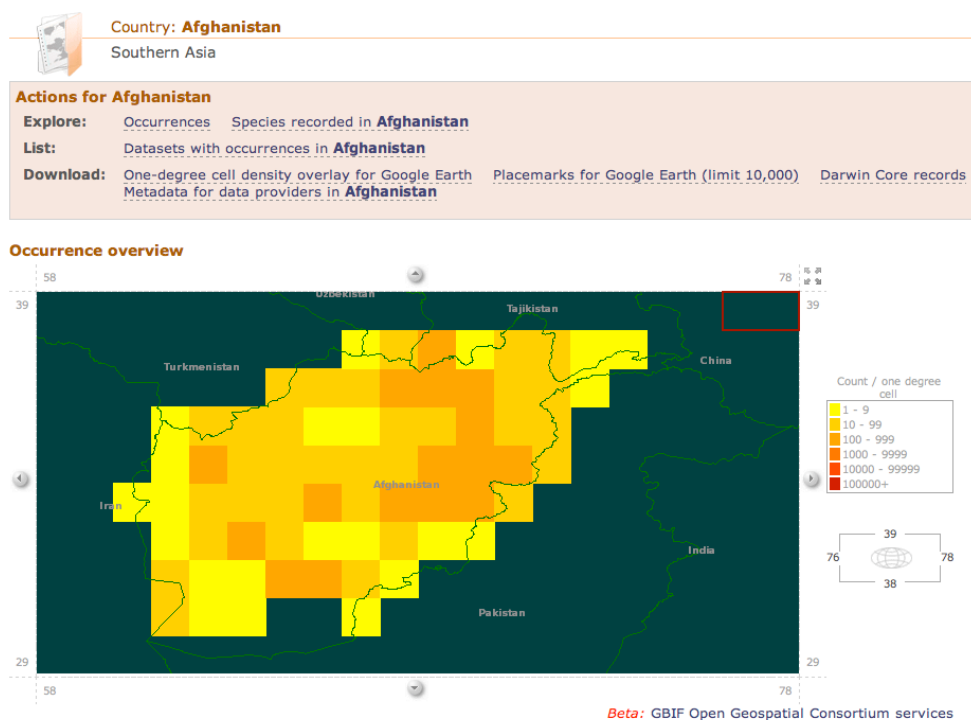


Figure 2.10: GBIF - Occurrence density zones

”zooming in” on one of the placemarks inside, one can see that occurrence coordinates are approximated (Figure 2.11). In this figure, we can see several coincident placemarks. They all have the exact same geographical coordinates, which is highly improbable in a real-world situation. This is most likely caused by the minimum granularity of the distribution cells used by GBIF in its approximations.

These approximation methods are remarkable in the sense that they can deduce geographical information from written records. Since most biological records were recorded this way, this approach is a great way to retrieve large sets of georeferenced information.

A similar specimen grouping approach has also been used in a system developed by Unicentro to help process georeferenced information in Paraná, Brazil [PCdOF05]. To group tree occurrences, the developers decided to insert a mapping class called *Parcela*, that represented fixed areas of 0.25 hectare. By creating 100 of these *Parcelas*, they were able to group all the trees in the analysed area.

2.4 Localisation and Identification

Botanical specimens need to be uniquely identified. Following identification, proper georeferenced information can make the records richer by helping in the location of specimens. These are two distinct notions: Identification and Localisation. Only after the



Figure 2.11: Group of occurrences in the GBIF database (KML viewed in Google Earth)

specimen is uniquely identified (Identification) can we append information that will allow us to locate it (Localisation).

2.5 Localisation of Specimens - Georeferencing

2.5.1 GPS

GPS technology was firmly established in 1993[[Wik09b](#)]. It allows users to locate people or goods anywhere on Earth, via satellite constellations orbiting the planet.

Even though this technology had been originally created for military purposes, GPS has been used for a few years for civilian applications. These include the common GPS receivers currently found in mobile phones and portable terminals, which are targeted at the home user. For companies, there are some solutions such as real-time fleet management, or route optimisation.

For this proposed solution, GPS can be used as a source of geographical information about plant specimens. Users of the system will be able to tag these using GPS enabled-terminals so that this information can be available on the Web.

2.5.2 Google Earth and Google Maps

Google Earth is a desktop application by Google that allows users to view the surface of the Earth. It also allows users to mark certain places in the globe, adding placemarks (points) or areas (polygons). Rich media can be added to these placemarks, such as HTML, images or other multimedia components. Google Maps is a web-based application that implements a subset of Google Earth's features and can be embedded in any web page to show map visualisations. It supports overlays of polygons and points. Small "balloons" can be placed on each of these spatial entities, showing HTML content like in Google Earth, but with more limited media support.

Both these solutions use KML as their default document format for saving and retrieving placemarks. They are becoming the most prevalent solution for both Web and Desktop-based viewing of geographic information [Rei08].

2.6 Identification of Specimens

2.6.1 RFID

RFID (Radio-Frequency Identification) technology has been used to identify plants on a specimen-level (Sean Hoyt, Doug St. John *et al.* [SHB03]). After analysing their findings, we will now state the most common procedures and uses for RFID technology and its potential uses in the botanical domain.

An RFID tag can be *active* or *passive*. Active tags have their own power source, usually a battery. Passive tags are simpler devices, which require no power source of their own, but are instead remotely activated when the receiver sends its waves throughout the transport medium (usually the air). The detailed phenomenon is out of the scope of this dissertation and therefore will not be described in further detail.

Passive tags are particularly interesting for the tagging of particular specimens of plants. Some of the reasons are:

- **Reduced cost versus their active counterparts**

Makes mass deployment and replacement of malfunctioning or broken tags cheaper.

- **Simpler circuitry**

It is enough to associate a simple identification code to a specific specimen: The relevant data about that specimen would be centralised in a remote database. This, in turn, requires an active connection to that remote database to exist during the reading process.

- **Compactness**

They are usually more compact, which makes their placement on plants possible.

- **Robustness**

Using less, simpler components in any apparatus usually yields better robustness, so passive tags should last longer; this is especially true in the harsh conditions where some plants live.

RFID technology is steadily becoming more prevalent in many modern applications where tagging and inventory management has specific demands. Perhaps the most interesting ability of RFID technology is that it can be used to recognise several items at once very quickly. Using traditional methods such as written tags or even optical reading bar codes, the user is required to *scan* the item. This is done either through direct visual contact or via an optical terminal for input reading. On the other hand, RFID's wave technology allows for remote recognition of the intended object without need for direct line of sight between the tag and the operator. The only requirement for the receiver/identifier to detect the nearby presence of a tagged object is being within range of the sender/tag.

However, RFID technology still has a main limitation: its costs. Not only the tags go from \$0.50 to \$50 (current prices from Barcoding Inc. [[Inc09a](#)]) but they require specific receivers which can range from \$151 to \$11,818. These numbers were drawn from the Web Site of an online RFID-related items retailer [[Nex09](#)] at the time of writing of this thesis.

2.7 Sum-Up and Conclusions

Botanical database users are becoming increasingly more concerned with geospatial information. Then, the databases devised approximation methods to retrieve it from their current information, usually based on historical records.

The methods used by these databases to query botanical information are also approximated when it comes to location-based querying. They lack geospatial querying support, and use mainly text-based searches. Spatial queries, however, are the key to retrieving more precise results and allow for more complex spatial queries.

Related technologies that can be used to tag and locate specimens include RFID and GPS. Applications like Google Maps and Google Earth are an interesting way to track specimens and present spatial information to end-users.

Chapter 3

Introducing a Geo-Botanical Repository

There are several excellent public botanical databases available. However, they all share a common characteristic: They use text-based restrictions as their means to query the data they contain. A different approach will be presented: *spatial querying*.

Another issue surrounding botanical information systems is that there are many of them. This would be good if they were able to co-operate by exchanging information. In most cases, however, this does not happen and their information is therefore "locked away" in their repositories. This problem of *locality* is the main reason why there should be a botanical repository, interoperable and open to all institutions.

3.1 Locality and Lack of Interoperability

It is very difficult to adopt a single system for storing all botanical specimen information. One of the main reasons behind this is that users or institutions tend to focus on their own specimens, that are recorded in specific areas [GN05]. For example, not all of these institutions provide easy access to those specimens, which makes the design of centralised solutions very difficult, requiring massive data format translation efforts like that of GBIF and other botanical museums [SLG06]. GBIF, for example, has gathered several datasets from many countries around the world and has centralised them. Now, a user visiting GBIF can access those datasets and export them in a community standard format.

To solve this locality problem, any proposed solution should:

- Offer an easy way for its users to access only their specimens within the whole data set.
- Provide a means for them to inquiry only about specimens in a specified area.

The approach proposed in this dissertation has taken these two needs into consideration, in an attempt to draw the interest of other botanical databases.

3.2 Flexibility and Exactness

Conventional databases are taking steps at completing their records with locality and geographical references. The most common example of these attempts is to include the country/nation where the specimen was recorded to the record itself. However, when querying a database for all the specimens recorded in a specific country, the possibilities are rather limited. The need for a more flexible and precise query method is very important, as will be now shown.

Spatial querying allows for a much greater degree of flexibility than text-based queries. The key reason behind using a spatial querying can be stated as the following:

What if the user wants to specify an arbitrary *polygon* and query about the specimens recorded *within/outside/...*¹ the polygon?

This kind of questions can only be answered by spatial querying. In order for these queries to be run in reasonable time, the underlying database must be spatially-enabled. This kind of database uses special techniques, designed to handle spatial data in a much more efficient way than conventional databases. Examples of these techniques are the usage of bounding boxes to progressively filter out polygons, as well as the usage of indexes especially conceived for geometric data.

A simple example is now presented to demonstrate the potential and usefulness of this approach.

In Figure 3.1 we can see a map of the Iberian Peninsula. In this map, the banners are placemarks for some botanical occurrences. It seems obvious that botanical occurrences are not constrained by country borders, so it makes sense to search for occurrences in user-specified areas. It would be interesting for the user (person or machine) to use a polygon such as the one in Figure 3.1 as the query restriction, or "country border". In this case, the system would retrieve the two banners inside the polygon (a dark one and a light-coloured one).

This sort of *fine-grained* geospatial restrictions yields several interesting uses. Just to name a few:

- Finding patterns between botanical and weather-related phenomena in a given region;
- Tracing species propagation and identifying potential pests or invasive species;
- Providing information to tourists interested in visiting places where certain species of plants exist;
- Helping botanical parks to compose their daily tours using spatial queries;

¹...or any OpenGIS Consortium [OGC95] Specification spatial restriction operator

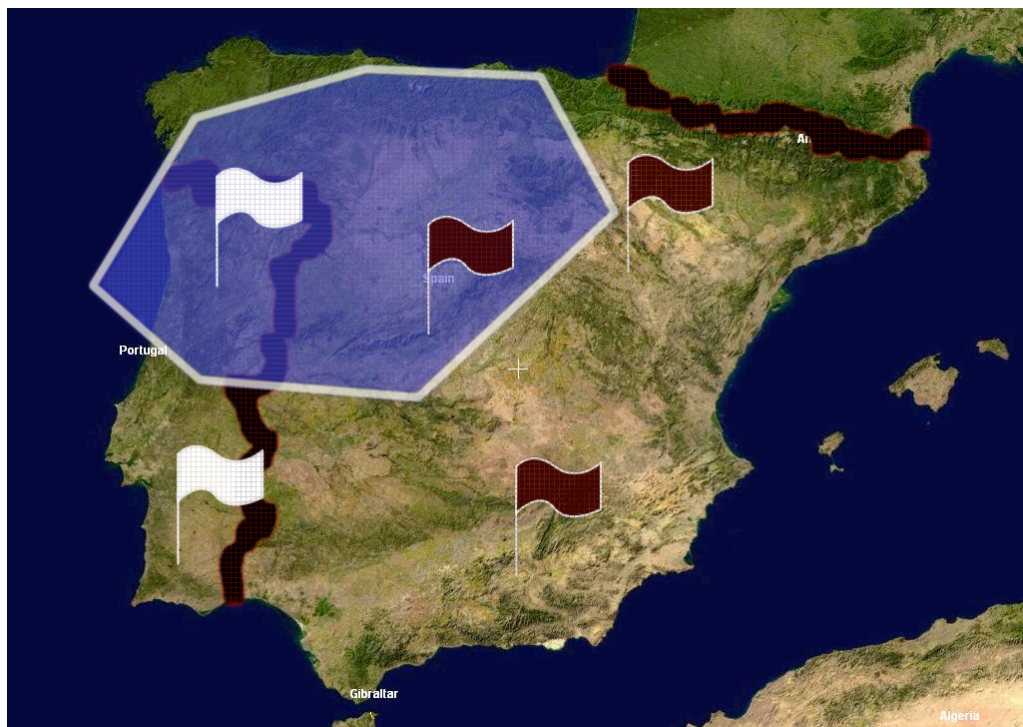


Figure 3.1: The Iberian Peninsula, some botanical occurrences and a polygon

- Helping pharmaceutical companies keep trace of places where botanical species are discovered;
- Helping environmentalists keep trace of species evolution in a given area;
- Helping architects in charge of city planning/landscaping to keep track the specimens planted in any city garden or park;

Each of these can translate into a client for a spatially enabled botanical information repository, using its spatial querying capabilities to support the client's business logic.

3.3 Standard Formats for Data Exchange

Since a lot of information has already been catalogued in this domain, any new-born solution must integrate with currently existing databases, so that it can grow quickly. The key to achieving this goal is the data exchange format used. These requirements for data formats interoperability can be considered part of the *Information Requirements* [IEE98] for the repository.

3.3.1 Botanical Information Exchange - Darwin Core

Darwin Core [Aut09b] is an XML (eXtensible Markup Language) [Inc09b] specification for organising specimen-level records, including taxonomical data, with a special emphasis on specimen-level records. It is not a standard yet, but at the time of writing, version 1.4 of the specification is due for review by TDWG [Aut09a]. This specification allows for some very interesting possibilities for the current project. This is mainly because of its two main characteristics:

- **XML**

XML is a very flexible meta-language with broad tool support. Therefore, programmers can process data in this format in a standard way, without having to build analysis tools from scratch. This kind of markup language is also appropriate for carrying information throughout the Web, because it offers a flexible, *lightweight* format, and much less cumbersome than typical relational database records, since it consists of plain text files. Also, *valid* XML must not only be *well-formed*, but also conform to a specification [W3C04]. This reduces errors and development time because the *validness* of the produced XML can be immediately assessed during development. This ensures that the shared XML will be processed in the same way on any parser prepared to use the shared specification. As a result, programmers can write their clients to exchange this kind of data with their servers with an assurance of similar *syntax* on both sides.

- **Good Support Become a Standard**

The Darwin Core specification has been through a constant improvement process since 2001, when it was first created at the University of Kansas. Since that time, version 1.21 of the specification has been used in several databases (GBIF [Sec09], MaNIS [otUoC09], HerpNet [Fou09a], OrNIS [Sys09], and FishNet2 [209]) [Aut09b]. Also, the recent submission for review at TDWG is proof that the project is strong enough to attain standard status.

Darwin Core also follows an interesting principle, similar to that present in the XML markup language guidelines:

”The number of optional features in XML is to be kept to the absolute minimum, ideally zero.” [W3C09]

This approach is adopted because it allows for a more robust specification. With increases in the number of optional features, some vendors may decide to implement some methods and others may not. This in turn can cause incompatibilities between these different systems.

In Darwin Core's case, the specification tries to set clear divisions between modules. It aims therefore to remove all optional elements from its Core. So, the Darwin Core itself only provides support for the most common information:

"The Darwin Core is intended to accommodate data that are relatively common – concepts that can be used across more than one discipline, concepts that are commonly found in relevant databases, and populated frequently enough to be broadly useful (...). The Darwin Core is also expected to be insufficiently complete for the needs of specialized disciplines of biodiversity. Data concepts deemed useful or required by a community can be specified in an extension of the Darwin Core(...) [[Blu09](#)]

Due to this succinctness, this project will have to make use of Darwin Core and its *Geospatial Extension* [[Col09](#)]. The Geospatial Extension provides an additional XML specification to Darwin Core documents, so that they can carry the geo-referencing information of every particular specimen record.

3.3.2 Geographical Information Exchange - KML

KML is also an XML specification for building geography-related XML files. This format is used by Google for information exchange between their two main geographical applications (Google Maps and Google Earth). These are becoming the most popular applications for geo-referencing nowadays [[Rei08](#)].

3.3.3 Other Formats for Geographical Information Exchange

Some websites require geographical information output in other formats than KML. For these cases, there is a command-line utility dubbed 'ogr2ogr' [[GDA09b](#)]. This utility relies on a library called GDAL (Geospatial Data Abstraction Library), and can be used for such format conversion purposes. Note that this only handles the geographical information; taxonomic information and other specific fields will always be formatted in Darwin Core. The list of available formats is too long to list here; it can be found in the GDAL Vector Formats Web Page [[GDA09a](#)]. For systems using these formats supported by GDAL, the conversion can be done in a transparent way to an output format specified as an argument. This includes practically all existing formats used for carrying geographical data [[GDA09a](#)].

3.3.4 Disadvantages of XML-based Information Exchange

The usage of XML has its advantages, but one has also to consider its main disadvantage: the cost of sending of very large XML files throughout the Web. In [[DP07](#)], this problem

was identified and some solutions have been proposed. These include the compression of the XML documents and then their conversion to more compact binary formats. However, the parsing of documents continues to be very costly in terms of computational resources. A mitigation procedure is to resort to SAX parsers when handling these larger files. In the case of the proposed system, queries are relatively small XML files, but to be able to handle extremely complex queries the system might benefit from a migration to a SAX parser.

3.4 A Different Approach - Introducing an Open Geo-Botanical Repository

Botanical databases are currently facing some issues:

- Botanical information is spread across multiple databases, without serious concern for interoperability;
- Lack of precise geo-referenced information for each of the databases' specimens;
- Use of approximation methods for determining specimen distribution, instead of adopting a dedicated *spatial querying* solution;

We propose a centralised repository for geo-botanical information. With a spatially-enabled database as its back end, it allows users to perform much more geographically precise queries. Botanical databases are an example of client for this system. This repository addresses the interoperability problem; it communicates with its clients via standard data exchange formats (*Darwin Core* and KML) all based on XML. Since XML is a technology-agnostic metalanguage, it can be used by almost any type of client. This repository will be used by clients to support their own activities whenever spatial querying is required.

Chapter 4

A botanical data model

We must develop a data model in order to represent the several real-world entities relevant for the botanical domain in this proposed system. This model allows us to relate each entity to its associated information and clarify the interactions between these different entities.

The model for the proposed repository was represented in UML and is shown in Figure 4.1.

4.1 Objectives of This Data Model

After analysing the most currently used botanical databases, we came to the conclusion that they could benefit from the adoption of a spatial querying system. They are all taking steps towards georeferencing, using approximation strategies (as seen in 2.3) among other methods.

This data model aims at representing the most important entities that must be present in a centralised system such as this. It is important to note that the model comprises entities that are not represented in the repository itself, but on an example client (the *Botanical Park* scenario). Those mapped by the park are signaled in Figure 4.1. They are inside contained inside the "Park" package.

4.2 Data Model Description

The model is divided in four packages. The `Taxonomy` package handles the information related to the taxonomic classification of botanical specimens. The `Identification` package handles the identification of botanical occurrences, on a specimen or area basis. The `Geography` package handles all the geographical data processing. Finally, the `Park` package organises the information about all the relevant entities represented on a *Botanical Park* scenario.

A botanical data model

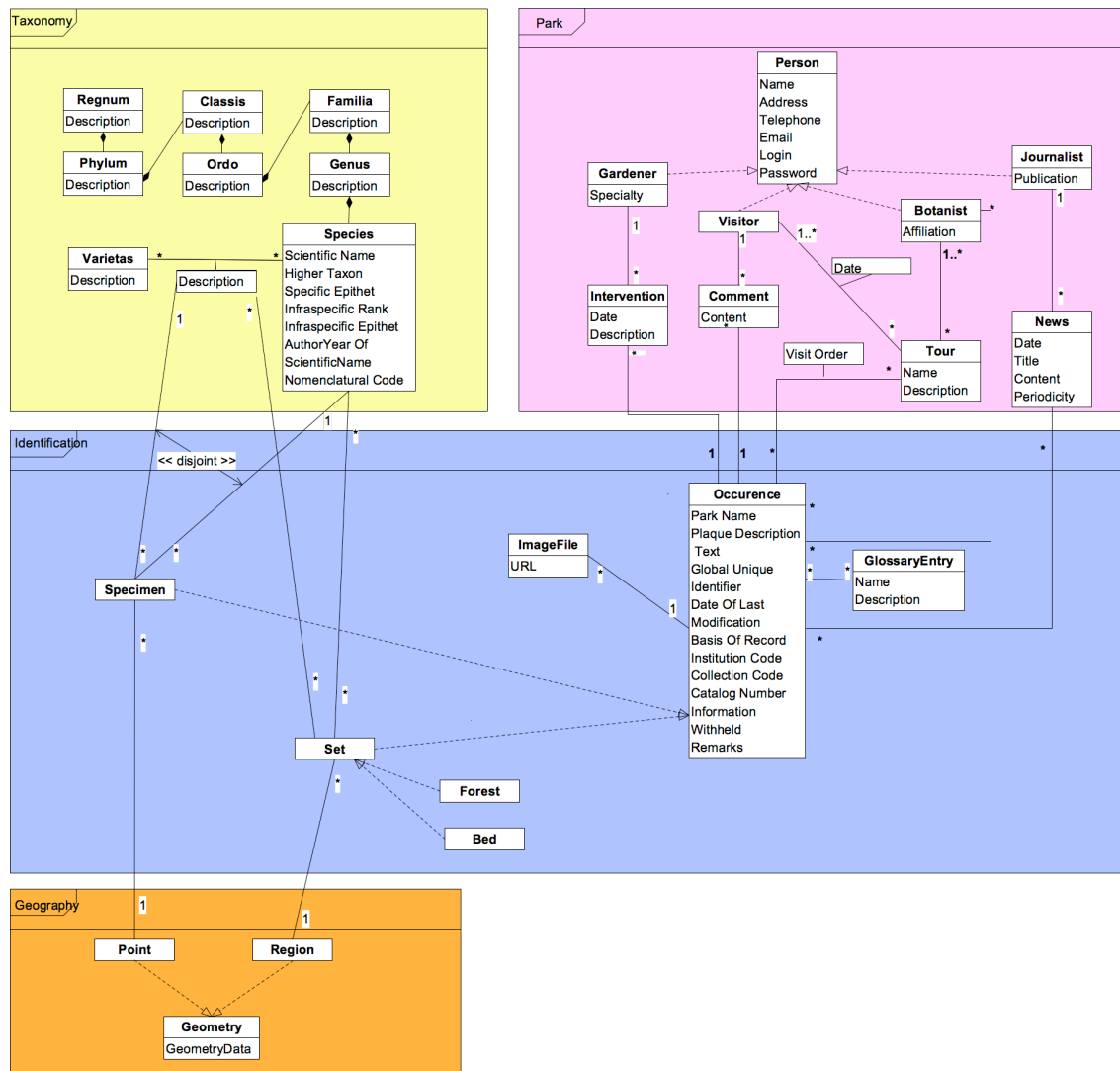


Figure 4.1: UML class diagram for botanical specimens and their sites

	Species 1	Species 2
	Phlox cuspidata	Vicia venosa
Varieties	P. c. var. <u>cuspidata</u> P. c. var. <u>grandiflora</u> P. c. var. <u>humilis</u>	V. venosa var. <u>cuspidata</u> V. venosa var. <u>venosa</u>

Table 4.1: Varieties of two species of plants (in [Wik09f])

Occurrence

Occurrences represent plant sightings, and are the basis of all cataloguing.

Taxonomic Classification

The Regnum, Phylum, Classis, Ordo, Familia, Genus and Species keep the taxonomic classification of all occurrences. This makes the classification structured. Darwin Core does not enforce a structured classification and its subdivisions; this system must therefore be more structured than its information exchange norm.

Species and Varietas

Varietas means "Variety". A species of plants can have many varieties, and a variety can be attributed to many species. As an example, we present two species of plants in Table 4.1. The underlined word in the table is a variety that is present in both species.

Specimens and Sets

In this system, Occurrences can be of two kinds: Specimen and Set. A Specimen represents an individual plant. This means that it has enough value, by itself, to be individually catalogued. A good example is the bi-centennary *Ginkgo biloba* at Jardim das Virtudes, Oporto [PVA09]; it makes sense to uniquely identify this tree, so it will be represented by a Specimen. In places where several plants of the same Species or Variety are present, it may be unreasonable to identify them individually. This is the case of areas populated by groups of small shrubs or individual flowers, for example. For these situations, the Set class was created. An object of this class can have several Species objects associated to it, unlike the Specimen and it can only be associated to a Region spatial entity, because it denotes an area and not an individual point. Specimens, on the other hand, can only be associated to Points, because they denote single plants.

Variety: An Optional Taxonomic Level

An Occurrence must have all the fields in the taxonomic tree for it to be considered valid. The Variety taxonomic level is, on the other hand, optional. If the Occurrence has an associated Variety, it must have an associated Species, because the Variety is ambiguous. To solve this, a *disjoint* constraint was added to the model (Figure 4.1). Linking the Specimen class to the association class between Species and Varieties makes it possible to know which variety of which species the specimen is.

Intervention

This class represents maintenance records for botanical occurrences of the botanical park, such as plant trimming, plantation or removal. They are created by the park's gardeners.

Comment

Visitors may like to leave their remarks on an occurrence. This class represents these comments.

Tour

Tours are paths specified by the park's Botanists for visitors to follow. They represent ordered sets of Occurrences. Since Tours can be undertaken by Visitors on different occasions, there is an association class between these two, containing the date of each visit. The Tour has a name (such as "The Roses of FEUP Park") and a brief description, such as "Take this tour to see all the species of roses within the FEUP Campus".

News

News are articles posted by a Journalist about some of the park's Occurrences. Some news are periodic, becoming active during certain parts of the year (for example, the blooming of some plant species). To provide these news in an automated fashion, the News class has a Periodicity attribute; This is the number of days after which this news article will be re-posted automatically. This period is specified by the Journalist during the creation of a new News article.

Gardener, Visitor, Botanist and Journalist

The Gardener takes care of the botanical specimens of the park, performing Interventions. The visitor comes to the park, goes on several Tours and can post comments for any of

A botanical data model

the Tour's Occurrences. The Botanist has an affiliate institution (FEUP, for example). He specialises in cataloguing botanical Occurrences and creating interesting Tours for the park.

This data model was mapped to a relational database. A graphical representation of the server database is located in the Annex (Figure [A.1](#))

Chapter 5

The Repository's Architecture

We now propose an architecture for a botanical repository and its clients. As the title of this dissertation implies, the main focus of this system will be its *openness*. Therefore, this architecture will aim at being as platform-independent and technology-independent as possible. There will be an important emphasis on expandability, following the conclusions of the article by I. Besembel et al. [JGI05]. It is a multi-tier architecture, providing a high degree of flexibility for future expansion, since expansion can be performed at any of these particular tiers.

This project is to be part of a larger one, which will focus on several other aspects of botanical research. Therefore, it must be designed to be as contained as possible; in an API, reducing the number of connect points means to display less methods to the user to minimise clutter, while maintaining the system's flexibility.

5.1 Design Decisions

This system focuses on openness and future expansion capabilities, so there are two main levels in the proposed architecture. The *server* and the *clients*. The repository is intended to provide common services to different types of clients. In the system developed by I. Besembel et al. to help catalogue several botanical collections [JGI05], some guidelines for botanical databases modelling are proposed, increasing flexibility and expandability; These approaches were taken into account for this system.

5.1.1 Repository Server

It was concluded makes sense that the server should support only the common needs of all clients. All other particular features are left for clients to implement.

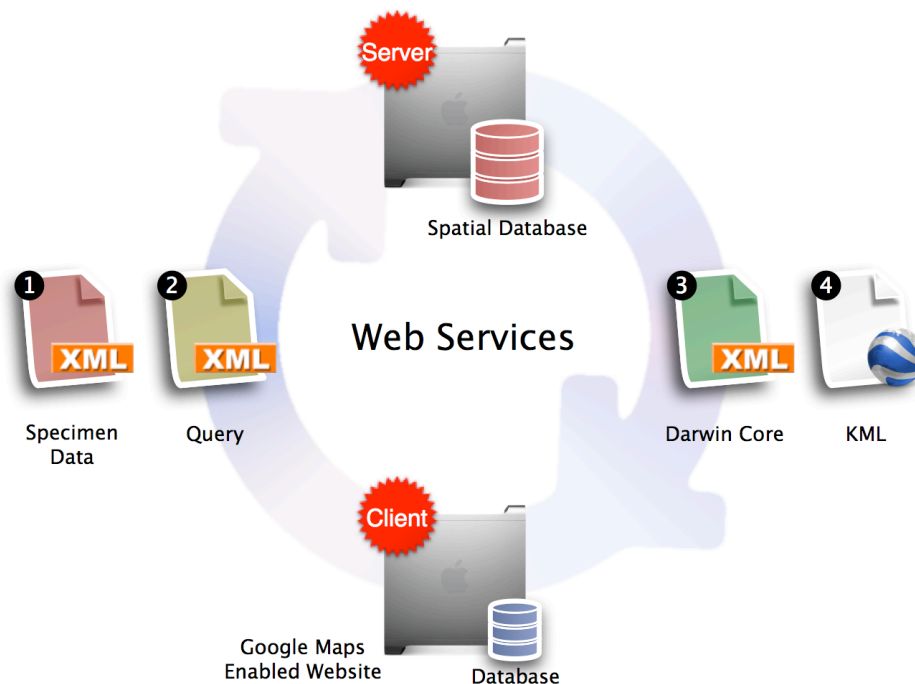


Figure 5.1: A high-level diagram of the proposed architecture for the PlantGIS system

Figure 5.1 shows how the server uses a spatially-enabled database to support its geospatial querying functionality. This kind of database is especially built to deal with spatial data. To insert specimens' information in the repository, the client (in this case a Web site) can send a Darwin Core document containing that information. To perform a query, the client must build a XML document containing the desired restrictions and send it to the server. In reply, the server may return the results in Darwin Core (for automated processing) or KML for visualisation in Google Maps or Google Earth.

To ensure communication between the repository server and its clients, an efficient protocol is required. This protocol must be platform independent and easily understandable by clients, while being flexible and powerful. The solution was to develop an XML language specification as part of this communication protocol.

5.1.2 Client Scenario

An hypothetical client scenario is shown in Figure 5.2. In this case, the client is a website, with its own database and business logic. The user interacts with the client, which in turn uses the server's spatial querying capabilities to provide users the information they need.

The proof of concept for this system (in Chapter 7) implements a similar scenario.

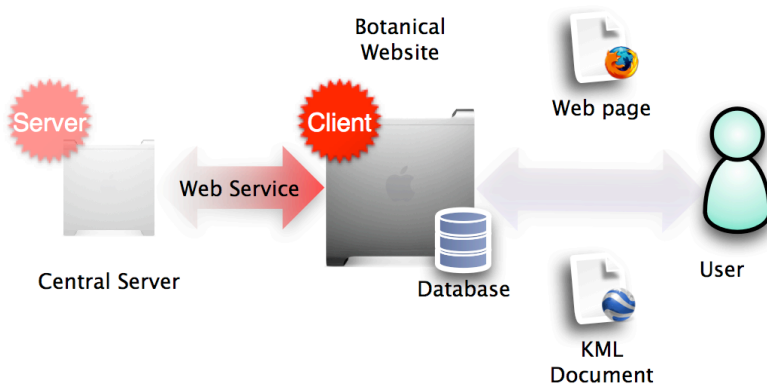


Figure 5.2: Physical diagram for a client scenario

5.2 The PlantGIS Spatial Query Language

Usually, API's are designed as abstraction layers, displaying a set of methods that clients can invoke. In this system, this approach was turned down because it would make the system static, both in the number of features offered and in its expansion possibilities.

The proposed solution was the development of an XML language. For the clients, it does involve the added effort of developing additional business logic to generate and process XML documents, but allows for much greater flexibility.

5.2.1 Language Requirements

The query language, intended to help clients access the information in the repository, was designed with the following features in mind:

1. **Simplicity**

Hiding intricate low-level implementation details of spatial database access, as these are mostly irrelevant to end-users. As an example, to retrieve all specimens inside an area, they only need to know what a polygon *intersection* is.

2. **Flexibility**

Allowing for elaborate or simple queries, depending on each end user's needs.

3. **Expressiveness**

Allowing users to access the whole set of features offered by the underlying database.

4. **Technology Independence**

Allowing client programmers to access the central database, regardless of the programming language that they use or other technological constraints.

The Repository's Architecture

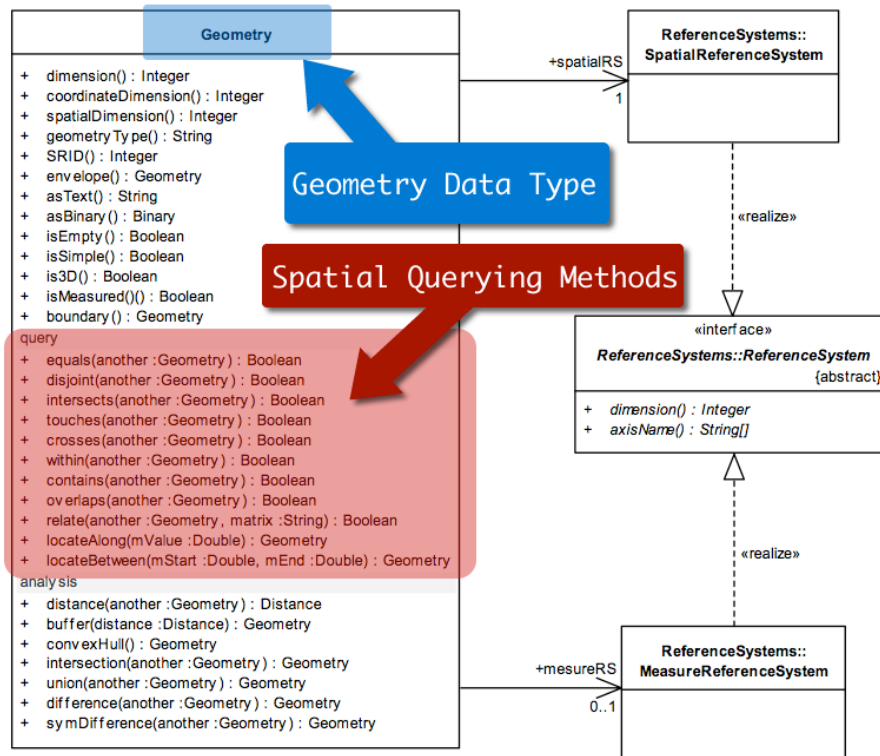


Figure 5.3: The spatial operators offered by the Geometry entity (from the OpenGIS Consortium Specification [OGC95])

To fulfil all these requirements, the *PlantGIS* query language is developed. *PlantGIS* takes its inspiration from the standard SQL notation, some *Darwin Core* attribute names and, of course, the OGC (OpenGIS Consortium) spatial restriction operators [OGC95]. These are part of the spatial restrictions from the OGC specification. An excerpt from that specification is shown in Figure 5.3. It shows a UML class diagram including the *Geometry* data type that is used to represent a geometric object, and the spatial querying methods that can be applied to it. These methods were transposed to the *PlantGIS* language in order to give clients the same flexibility of OGC-standards compliant *spatial* SQL queries, while concealing the implementation details of the spatial database used to retrieve the results.

As can be seen in the example in Figure 5.4, the language itself is very simple, resorting to the meaning of each of the spatial operators defined by OGC [OGC95]. To compose this query, the user has to know that the *within* operator will include all occurrences *spatially within* [OGC95] the argument polygon. To use a more informal description, this operator will return polygons "fully inside" the argument polygon.

To provide a high degree of flexibility, the language supports the **AND**, **OR** and **NOT** boolean operators. The precedence is given by the level of each query in the XML tree. Common *text-based* querying capabilities widely used in other databases are also avail-

```

<gisquery>
  <or>
    <query>
      <within polygon_field="location">
        <polygon>
          <label>Olive Trees Area</label>
          <point>
            <longitude>-8.595123274518716</longitude>
            <latitude>41.17710722023982</latitude>
            <altitude>0.0</altitude>
          </point>
        </polygon>
      </within>
    </query>
    <query>
      <equals string_field="ScientificName">
        <argument>Olea europaea</argument>
      </equals>
    </query>
  </or>
</gisquery>

```

Figure 5.4: An example of a simple query, specified using the *PlantGIS* query language

able here. Both types of queries can be combined freely, as demonstrated in Figure 5.4, where the user searched for all the specimens within a specific polygon OR having *exactly* "Olea Europaea" (the widely known Olive Tree) as their Scientific Name.

Using XML allows interaction with a broad range of clients. From handheld terminals to full-fledged Web servers, there is almost always an XML parsing library available. When transported via Web Services, this query format dramatically reduces technology dependence.

5.2.2 Examples of PlantGIS Queries

We will now show some examples of PlantGIS valid queries. A KML file was produced in Google Earth (Figure 5.5), and will be used as the basis of these examples.

Basic

One of the most basic possible PlantGIS queries is shown in Figure 5.6. It selects all Occurrences whose Geometry¹ is within² polygon B, shown in Figure 5.5.

Intermediate

Figure 5.7 shows another use of the Query Language. In this example, the client combines two restrictions (within B and NOT within C) to select all specimens within the new composite polygon, shown inside the balloon (at the root of the XML tree representation).

¹ see the class diagram of the system (Figure 4.1) for more details

² spatial operator

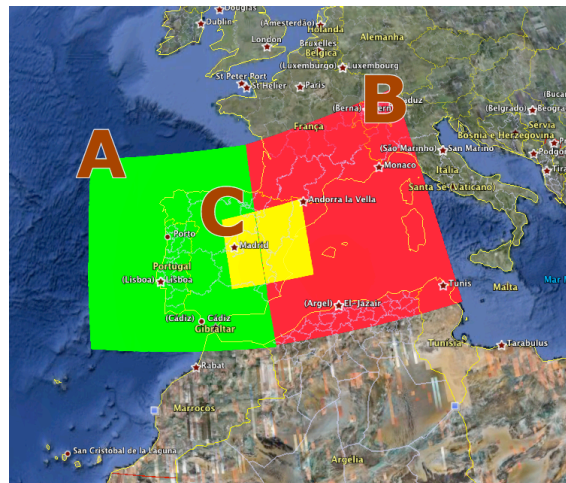


Figure 5.5: Three polygons (A, B and C) used as arguments in PlantGIS query samples (Google Earth View)

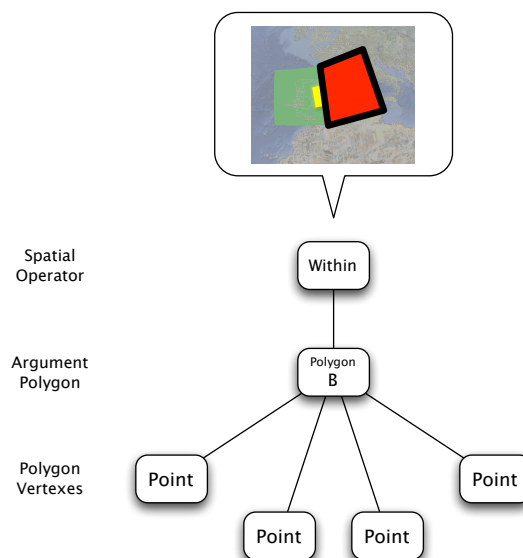


Figure 5.6: PlantGIS Simple Query

The Repository's Architecture

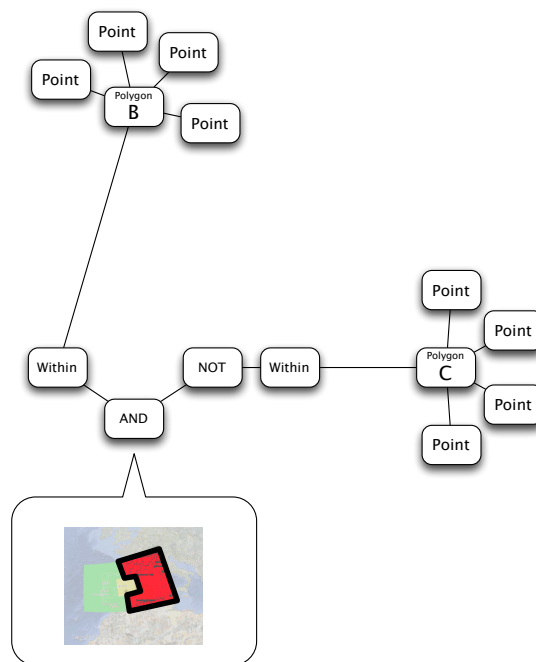


Figure 5.7: PlantGIS Intermediate Query

This shows the ability to build queries excluding certain areas. This can be used to exclude areas where users do not want to search.

Advanced

In Figure 5.8 we can see a more advanced usage of the PlantGIS query language. It combines several features such as spatial queries, boolean operators, and backwards-compatible text-based queries. The initial filter (higher level AND constraint) can be used by a botanical park or a botanical database to automatically select only their specimens. Other constraints are then applied to show several possibilities offered by the language.

Full XML Documents

The PlantGIS XML examples shown here as diagrams are included in the Appendix (see D.2).

5.3 Technological Analysis

Some details about the architecture of the server will be explained in the following section. There has been an attempt to implement an MVC (Model-View-Controller) tier logic at this level, so that the purpose of every component is well defined and separated from the

The Repository's Architecture

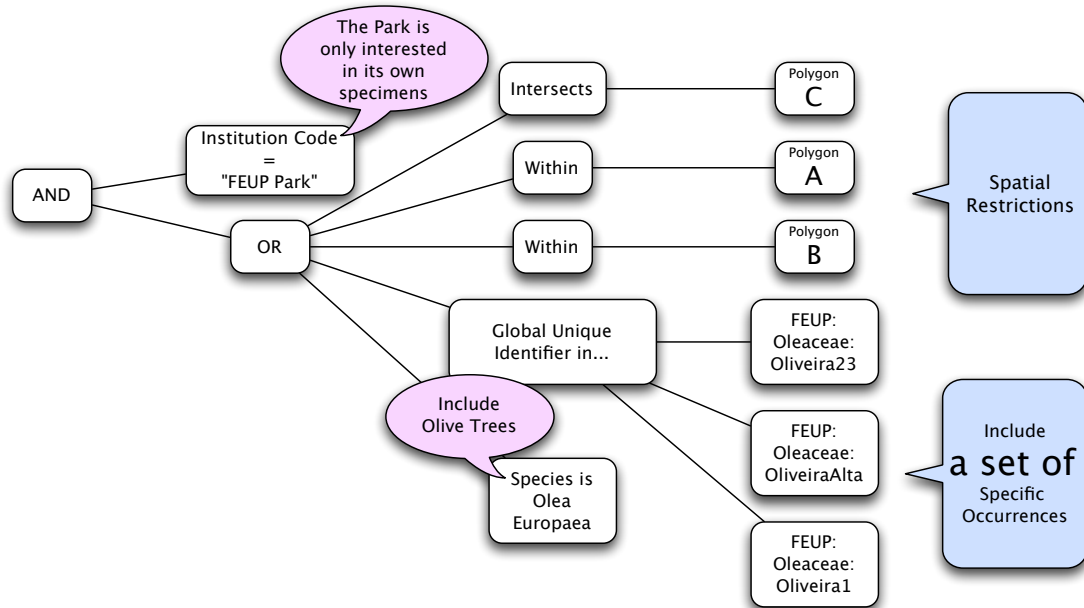


Figure 5.8: PlantGIS Advanced Query

rest. The description will start with the most low-level layer and end with highest-level one.

5.3.1 Open-Source Requirements

Since the start of the project, the only restriction imposed on technologies was that every piece of software used had to be Open-Source. Thus, some popular technologies had to be ruled out from the start. This excludes the .NET Framework, for example.

5.3.2 Object-Relational Mapping Solutions

The database access layer is the equivalent to the Model section of the MVC paradigm.

One of the requirements of the system was that it has to resort only to *open-source* technological solutions. The two most relevant open-source DBMS (Database Management System) alternatives, at the present time, are PostgreSQL and MySQL (Table 5.1).

Parameters	PostgreSQL+PostGIS	MySQL
OpenGIS Consortium Spec compatibility	Full [Con09]	Partial
Chosen Alternative	Yes	No

Table 5.1: Comparison between PostgreSQL with PostGIS and MySQL with Spatial Extensions

Parameters	PHP + PostGIS Wrapper	RoR + GeoRuby	Java EE + Hibernate
Direct SQL queries	Yes	Yes	Yes
OGC Spec Support	Partial	Partial	Full
Object-Oriented	Yes	Yes (Enforced)	Yes (Enforced)
Querying Language	PHP	Ruby	HQL
DBMS Compatibility	PostGIS	MySQL, PostGIS	MySQL, PostGIS, Oracle
Last Update (DD/M-M/YY)	05/10/05	02/03/09	15/08/08

Table 5.2: Comparison between several available frameworks

The main reason why PostgreSQL was chosen instead of MySQL is that MySQL does not currently implement the whole Open GIS Consortium specification for simple geospatial feature collections [MA09, For09]. The next step is selecting the appropriate abstraction layer for linking to the database; it had to be *open-source* based, so the Microsoft®.NET Framework™ was not thoroughly analysed.

Within the open-source community there are three main options:

- **PHP + Avoir Project - PostGIS PHP Wrapper**
- **Ruby on Rails ActiveRecord+ GeoRuby**
- **Java EE + Hibernate-Spatial**

Table 5.2 is a short side-by-side comparison between these three technologies.

All systems support direct SQL queries to the database. However, this functionality is not very useful when one is attempting to build an easily maintainable and scalable application. In these cases, writing SQL directly inside source files is not a good practice and should only be used as a last resort. For example, in order to support the specific functions offered by PostGIS that are not supported by the OGC, one must write stored procedures and invoke them from the database. However, the core of the application should be written in a higher-level language to ensure portability between different DBMS.

OGC Spec Support is vital for the usefulness and compatibility of the application; The PostgreSQL + PostGIS DBMS solution features all of the required functions. To take advantage of this, one must however find an object-relational mapping solution that is able to transpose all those capabilities into usable methods for the business logic to use. It happens to exist such a solution, called Hibernate Spatial; it will soon be presented.

```
array overlaps (string $dbname, string $tablename, string $geom1, string  
                $geom2);
```

Figure 5.9: A function taken from the Avoir API [[Avo09](#)]

PHP and Avoir

Object-Orientation and enforcement of good programming practices is a vital item for software quality nowadays. PHP is more flexible than Ruby on Rails and Java, because the user can decide to structure his application using classes or not. However, this flexibility does not enforce good programming practices.

Regarding the query language, this PHP library acts as a *wrapper* for PostGIS functions. This means that queries must be done in a step-by-step manner. The wrapper only allows for *individual* PostGIS functions to be called. Figure 5.9 was taken from the Avoir API [[Avo09](#)] and shows an example of a method that maps the `overlaps` spatial method (it can be seen in the OGC spec, Figure 5.3).

In Figure 5.9 we can see that in order to use this function one must already have queried the database for the two argument geometries (`geom1` and `geom2`) and then do a third query to see if they overlap. This limits the usefulness of the class, because its functions are purely geographical. One cannot mix simple queries, like selecting all specimens whose species is "Petroselinum crispum" (*common Parsley*) that are also inside a specific radius of one point in space. One would have to select and retrieve all their geometries. Then, check each and every one of them for compliance with the second condition. This would overload the database with unnecessary requests. The key here is to come up with a solution that can perform these *hybrid* queries in native SQL, eliminating all this overhead. Also, database optimisations (like indexes and bounding boxes) must be taken advantage of.

Ruby, GeoRuby, and GeoKit

Ruby on Rails offers a very transparent way to work with the database. Its mapping technology (`ActiveRecord`) allows for complete abstraction and 2-way mapping and reverse engineering (of the database tables to classes and vice-versa). However, the GeoRuby website states the following:

The data model roughly follows the OGC Simple Features for SQL specification, although without any kind of advanced functionalities (such as geometric operators or reprojections). [[Geo09](#)]

This makes the plugin of limited use. GeoRuby inputs and outputs EWKB, Hex-EWKB and EWKT [[Geo09](#)], but leaves to the Ruby program the task of calculating

distances, reprojections and other sort of functions already implemented natively in the PostGIS extensions.

The Ruby on Rails plug-in has another problem: It's lack of native PostGIS functions mapping. It requires an external library to do distance and projection calculations, and at the time only one exists: GeoKit [LE09a]. Analysing the Geokit API [LE09b], one can clearly see that it was designed for basic distance calculations, IP-based location and Geocoding. It was not designed with OGC standards compliance in mind, but particular needs:

”Geokit provides key functionality for location-oriented Rails applications:

- Distance calculations, for both flat and spherical environments. For example, given the location of two points on the earth, you can calculate the miles/KM between them.
- ActiveRecord distance-based finders. For example, you can find all the points in your database within a 50-mile radius.
- IP-based location look-up utilising hostip.info. Provide an IP address, and get city name and latitude/longitude in return.
- A before_filter helper to geocoder the users location based on IP address, and retain the location in a cookie.
- Geocoding from multiple providers. It currently supports Google, Yahoo, Geocoder.us, and Geocoder.ca geocoders, and it provides a uniform response structure from all of them. It also provides a fail-over mechanism, in case your input fails to geocode in one service. Geocoding is provided by the Geokit gem, which you must have installed.

”

in GeoKit API [LE09b]

Although useful for other sorts of applications, these extensions do not offer the expandability and efficiency one should expect when building a GIS-related API. This project aims at leaving a solid base for further development but using such limited solutions would severely hinder its possibilities in the future.

Java EE and Hibernate

Java EE with Hibernate also offers a reasonable degree of abstraction and bidirectional mapping. It does not, however, have the same simple feel of Ruby on Rails and is more ”heavy-weight”. Java EE applications are usually much larger (in terms of lines of code) than their Ruby on Rails counterparts. Both offer mapping of CRUD (Create, Read, Update and Delete) operations and integrate well in a MVC-based approach to application

development. But the most important feature of Hibernate-Spatial is its ability to map HQL (Hibernate Query Language) code to *native* SQL code for PostGIS, MySQL or Oracle databases, depending on the adapter used.

5.3.3 Business Logic Implementation

This Business Logic layer plays the role of the Controller section of the *MVC* paradigm.

It was decided that the technology to be used would be the Java EE suite, because the selected object-relational solution (Hibernate) is designed for usage with this programming language.

5.3.4 Links to Outside Consumers - API

The links to outside consumers can be seen as the View section of the *MVC* paradigm. This is the visible part of the system, linking it to all the clients that wish to use its functionality.

These connections are ensured via the exchange of Darwin Core, KML documents and the PlantGIS query language, as seen in Figure 5.1.

In this section of the system, the chosen technologies are all part of the Java Enterprise Edition suite.

5.4 Overview of Selected Technologies

The server application was designed to be an API. As such, it must be flexible, open, and easily expandable. Also, all selected technologies had to be based on *open-source* software.

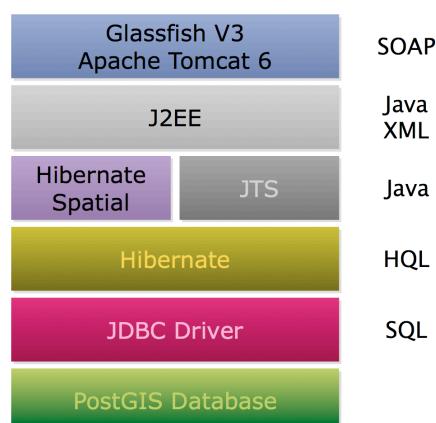


Figure 5.10: Technologies used in the server application

Figure 5.10 shows the choice of technologies for this system. To the right of each layer is the predominant language/information exchange format it handles.

PostGIS Database

PostGIS is an geospatial extension for the PostgreSQL DBMS. This extension adds dedicated spatial data types, such as `Geometry`, and spatial operators from the OpenGIS Consortium. The most prominent *open-source* alternative to PostGIS is MySQL with Geospatial Extensions. However, one feature sets of this extension sets *PostGIS* apart from *MySQL* with Geospatial Extensions; its full OpenGIS standards compliance [Con09, MA09].

Hibernate

Hibernate is a widely used object-relational mapping solution for Java. It makes database interaction transparent to the programmer, mapping each table in the database to Java classes. Its own query language, HQL, is translated by the framework to SQL in runtime and injects it into the database. This introduces additional overhead in the system but allows for complete Object Oriented querying, since HQL can include references to Java object fields and not only table/column names. An equally important feature is database vendor independence. Since no *native* SQL is actually written, the database adaptor (JDBC Driver) can be changed if one decides to replace the underlying database. The Hibernate framework will just map its HQL to a different SQL dialect to encompass the change. Also, using Netbeans, database changes are quickly dealt with because the IDE allows for automatic regeneration of the corresponding Java objects, and since there is no hard-coded SQL, the IDE itself helps with the process of *refactoring*.

Hibernate Spatial

Hibernate natively supports the most common database datatypes (`integer`, `varchar` and so on). Geospatial data types, such as `Geometry` are handled by Hibernate Spatial, allowing Hibernate to map these data types and making their usage transparent to the programmer. This plugin takes HQL with geospatial data types and generates the native SQL (depending on the underlying JDBC adapter) to take advantage of the indexes and optimization of spatially enabled databases.

Java Topology Suite (JTS)

Java Topology Suite is an efficient library for handling geometry-related operations. It provides methods for calculating unions, intersections and other operations between `Points` and `Polygons`. Hibernate Spatial works in conjunction with this library

to perform all the necessary calculations when querying the database using spatial restrictions³ and return the records that match the criteria.

5.5 Implementation Details

5.5.1 Server Components

Every system is composed of several subsystems. In this particular case, it is imperative that the boundaries of each module be clearly defined, as documented in the system's Component Diagram [5.11](#).

Query Service and Insertion Service

These services are the two entry points of the system. The Query Service receives and processes *PlantGIS* queries. The Insertion Service receives and processes new botanical occurrences.

Output Package

In the output package are two parsers that receive query results (Hibernate objects) produced by the Persistence Logic. Depending on the format requested by the client, these results are translated to KML or Darwin Core.

PlantGIS Parser

This XML parser translates *PlantGIS*-formatted documents to HQL queries that are sent to the Persistence Logic component.

New Records Parser

This parser translates XML documents containing information on new occurrences. It separates the taxonomic classification of the occurrences from the included media. The taxonomic information is then sent to the Persistence Logic. Much like the system described in [\[JGI05\]](#) media is stored in the repository. In this case, images can be sent inside these XML documents, in Base64 [\[Gro09b\]](#).

Base64 Decoder

The Base64 decoder is part of the Apache Commons Codec library [\[Fou09b\]](#). It is a library that implements several encoding and decoding algorithms. Base64 is referenced

³OpenGIS Consortium specification [\[OGC95\]](#) restrictions such as `within` or `intersects`

The Repository's Architecture

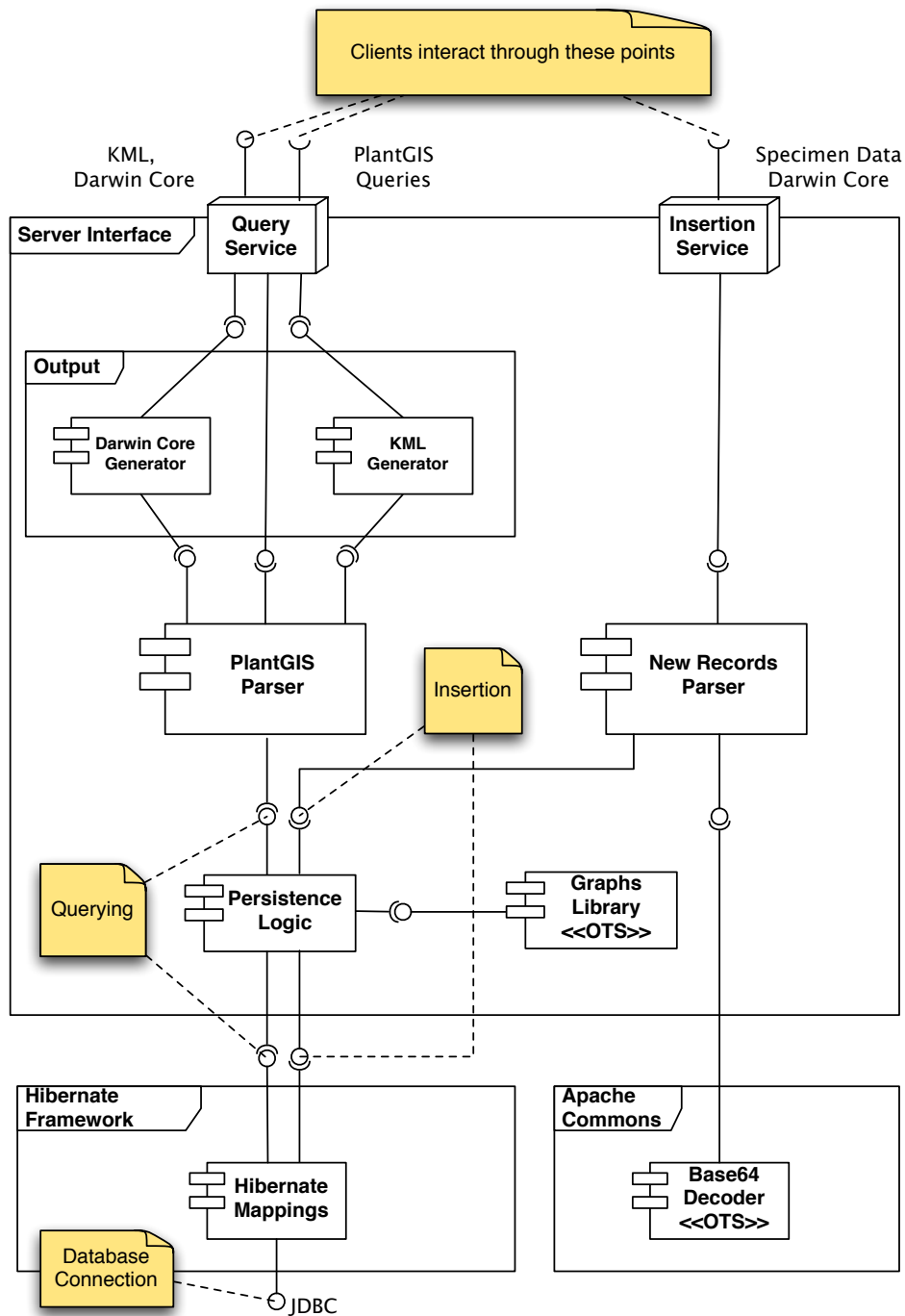


Figure 5.11: Component diagram for the repository application

by RFC 2045 [[Gro09b](#)] a format for exchanging non-human readable information. It is also a standard encoding method for e-mail attachments.

Graphs Library

An implementation of Dijkstra's Shortest Path algorithm is used by the persistence logic to match the foreign keys in different tables, helping to build the JOIN clauses for the HQL queries.

Persistence Logic

The persistence logic component handles all interfaces with the database, through the creation, usage and disposal of Hibernate-mapped Java objects.

This separation allows the system to be more easily expanded and facilitates the replacement of any components.

5.5.2 Security Concerns

To prevent the overloading of the system through queries involving a very large amount of data there is a limit to the number of rows returned. For example, a query using the `NOT within` query with a very small polygon as its argument can return all the specimens in the database. Such queries must be restricted, and limiting the number of rows is the simplest way to do it.

Since this is a Web-based system, there must be internal barriers against maliciously crafted queries. We will now show the mechanisms currently implemented in the system to help prevent these security issues.

XML specification of the querying language

The specification of the querying language in XML allows for a first barrier against malicious users. Only query XML validating against the PlantGIS specification is accepted, and all other documents are rejected.

Internal matching of queryable fields

The XML specification has specific names for fields that can be used as restrictions in a client's query. To make sure that these field names are not tampered with, the arguments of the XML queries are validated inside the parser. Any field not corresponding to specifications will trigger an immediate rejection of the query document. This allows for a second validation of inputs, denying access to database fields that should not be public. On the other hand, it provides a simple means to do the mapping between XML attributes

and their corresponding fields in the database. Programmers wanting to expand this API only need to add the new attributes to the specification, the fields in the database, and the matching in the parser (adding an entry to an `Hash Table`).

Hibernate support

By not using hand-coded SQL and relying on the Hibernate framework, every line of code that enters the database is *automatically generated*. This means that typical attack methods, such as *SQL injection* can be effectively countered.

5.6 Sum-up and conclusions

It has been shown how the presented architecture intends to serve as a geo-botanical repository. A query language for interacting with this repository was developed, granting openness and ease of use to the system. Some examples of the features offered by this language were shown. They illustrate its flexibility, simplicity and expressiveness.

A technological analysis of current solutions has been made in an attempt to select the best technologies currently available for the implementation of this system. The choice was Java as the programming language, PostGIS as the DBMS and Hibernate as the object-relational mapping solution.

Also, we have explained the formats of data interchange between the different *tiers* of the application. The usage of standard technologies such as XML specifications add to the cross-compatibility of the application. This solves the problem of *syntactic* compatibility and validation. On the other hand, one needs to ensure *semantic* comprehension between interconnected systems. To address this technology we have exposed two standards built on top of these technologies (KML and Darwin Core). KML addresses the visualisation and exchange of purely geographic data *per se*. Darwin Core aims at facilitating the exchange of taxonomic and specimen-related information between this system and its *clients* (websites or other botanical databases). It also supports the embedding of geographical information using its optional Geospatial Extension.

Chapter 6

Dataset Acquisition

One of the most important steps for this sort of applications and systems is making sure that the data supplied by the system is well structured and organised. On the other hand, one must be concerned with specific domain constraints. In this case, there are several *dialects* for the taxonomic classification of plant species. This in turn makes the retrieval and organisation of this kind of data slightly more complex, because of the required pre-processing.

6.1 Taxonomic Classification Standards

One of the first problems discovered during the course of this investigation was that everyone has their own "*standard*" for taxonomical classification. To acquire information from a website, one must prepare the intended program to process different kinds of taxonomic trees. Some taxonomic levels are synonyms, such as "Family" and "Familia". Most websites do not follow the same nomenclature, as was already shown (see [3.1](#)). Sometimes there are discrepancies even inside a single website. Wikispecies, the website that was processed for this data acquisition effort, is such an example.

6.2 Website Crawling

For the correct dimensioning and testing of any information system, one must supply it with relevant and useful information. In this case, the required data set would have to start with a reasonably-sized set of taxonomical information. This would, in turn, support the listing of specimens, linking them to their taxonomical classification.

The next step is to look for alternative means obtain such data. The most obvious way would be to input the structures by hand. With around 350.000 [\[Wik09c\]](#) documented

species of plants in the world, this task is not feasible in the time available (or pleasant in any way!).

On the other hand, this information is available on many websites. Some of these websites are created, maintained and expanded by public organisations (e.g. the PLANTS Database). However, the whole information stored in their databases is not available for everyone to download. They provide listings for each of their species of plants, and some even have specimen information, as seen in the Botanical Databases section [2.2](#).

So, we turn to automated methods. The easiest way to retrieve a sizeable collection of plant species information and their taxonomy is to write a simple program. This kind of programs, usually named 'Webcrawlers', are used to navigate throughout a website and retrieve their structure, or other data deemed interesting.

One of this kind of programs was written for this project, and some details on it are shown as follows.

Wikispecies

After the adoption of the Hibernate framework as the main object-relational mapping solution for the server, a program was developed to retrieve information from the Wikispecies [[Wik09f](#)] website, already using the Hibernate mechanisms for persistence handling. This included roughly 30.000 different species of plants, each with their whole taxonomic classification tree. Some of these also have their varieties, which were also inserted into the database.

The program not only dumped the information, but also structured it, creating the appropriate tables (as needed) and allowing a backtracking process. This makes the retrieval of the whole taxonomical tree for a given species easy, while at the same time keeping the relational schema normalised.

More than 30.000 different species of plants were retrieved from this crawling effort, including their full taxonomic classifications.

This data is useful for validating user input when inserting new occurrences. It can also be used to assist the user in inputting the values for taxonomic levels. A demonstration using AJAX is shown in the client application described in section [7.7](#)).

6.3 Further Use

One of the requirements for this *dataset* is its ability to be used for later works. This means that one must be able to *export* the database structure and data to a standard format. In this case, the adopted format will be the Darwin Core, much like GBIF [[Sec09](#)].

Chapter 7

Demonstration Client

To demonstrate the features offered by the server application, a prototype client was developed. This client is designed for a small botanical park, and was named *FEUP Park*.

7.1 Requirements

FEUP Park is intended to be a simple Web-Based application, supporting some of the most important use cases of a botanical park. Its objective is to showcase the abilities of the server application in a possible usage context.

7.1.1 Use Cases

A full specification of the system's Use Cases is shown in the Appendix. Section [B.1](#) contains a series of Use Case diagrams, and Section [B.2](#) contains their respective specifications, according to the IEEE-830-1998 standard [[IEE98](#)]. The main objective of this prototype is to show the support that the repository can lend to a client. Being so, more common use cases such as the posting of news and permissions management have been deliberately left out of this client's implementation. The main effort was then directed towards georeferenced specimen insertion and spatial querying.

7.2 Execution Examples

Some usage examples will now be presented, showing the implemented usage scenarios for the client.

7.2.1 Specimen Insertion

Main Page

Figure 7.1 shows how the occurrence insertion page is designed. There are four areas in this page.

- **KML File**
Allows the user to upload a file containing geographic information. This file is then parsed and the entities inside can be used to add georeferencing to the botanical occurrence.
- **Upload Images**
Allows the user to select a series of images that will be stored in the repository along with the specimen occurrence.
- **Taxonomy**
Allows the user to insert the taxonomic classification of the specimen in this occurrence.
- **Specimen-related Information**
Allows the user to add identification to this occurrence. Includes the *RFID Tag Code* field, designed to store the reference for the identification code also present in the RFID tag (this information is saved locally in the park).

Execution Flow

To insert a new specimen occurrence, an user should take a series of steps in the system. This execution sequence shows an example of how to perform the insertion.

If an user does not want to add localised information to the record, the following sequence may be skipped until the part where the user specifies the images to associate to the record. In Figure 7.2 we can see a file upload box. Users should use this to select a KML file containing the spatial entities that the user wants to use as geographical references for the new botanical occurrence. After uploading a valid KML file, a Google Maps visualisation will be shown, as well as a list of points and polygons contained in that file (Figure 7.3).

The user can select a polygon or a point to append to the botanical record and then make it georeferenced (Figures 7.4 and 7.5).

To upload images of the occurrence, saving them in the central repository, the user can use an upload box. Small thumbnails are shown in the web page to help the user see which images he has already sent in (see Figure 7.6).

If the user selected a point as the spatial entity to associate to the occurrence, he may only add one taxonomic classification tree to that occurrence. Selecting a polygon will

Demonstration Client

add occurrence

kml file

Please select geospatial entities to link to the record

Valid polygons and points will be listed here for selection after uploading a KML file.

no file selected

upload images

Associate images to the occurrence

no file selected

taxonomy

Please fill in the taxonomic information for this observation

species 0

Kingdom:

Phylum:

Class:

Order:

Family:

Genus:

ScientificName:

Varieties:

specimen-related information

Additional Information

Collection Code:

Catalog Number:

RFID Tag Code:

finish process

Confirm submission

Figure 7.1: The main insertion page

Demonstration Client

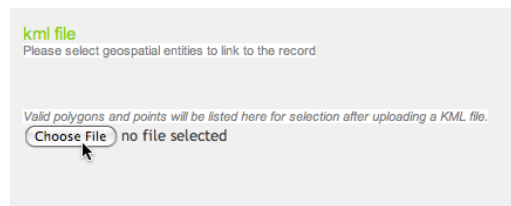


Figure 7.2: KML File Upload

enable the user to add multiple taxonomic classifications to the occurrence. An area can contain several species of plants that cannot be separated and uniquely classified, as seen in the data model (Chapter 4). To add multiple species, the user must click the "Add Species" button (seen in Figure 7.8).

The new taxonomic information input area is also AJAX (Asynchronous Javascript and XML) enabled (as can be seen in Figure 7.9), allowing the insertion of all taxonomic levels, including varieties (also seen at the bottom of Figure 7.9).

The user must then insert the identification information for this occurrence. The collection code and the catalog number help the park's database to request specific unique identifiers for its specimen. The RFID tag helps to identify the specimen in the future, since its tag value code will be recorded in the database. If, when adding a new occurrence, the park staff decides that this specimen should be tagged, they may add the tag's value in this field for future usage.

After inserting all the taxonomic and identification information, and optionally images and the polygon/point to associate to the occurrence, the user must select the "Submit Record" button (see Figure 7.11) to process the request.

Demonstration Client

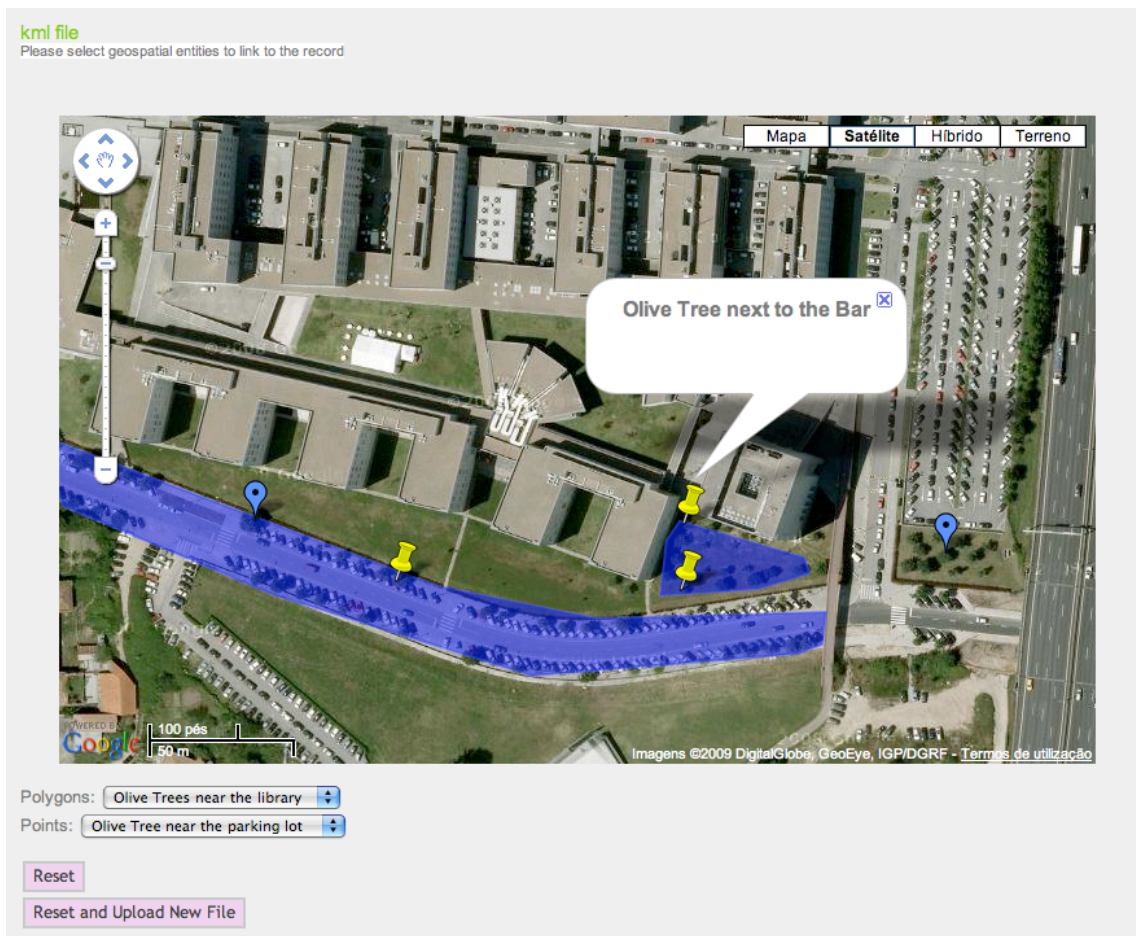


Figure 7.3: Viewing the uploaded KML and the list of spatial entities

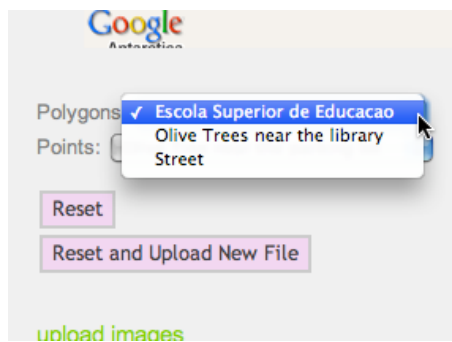


Figure 7.4: Selecting a polygon from the uploaded KML file

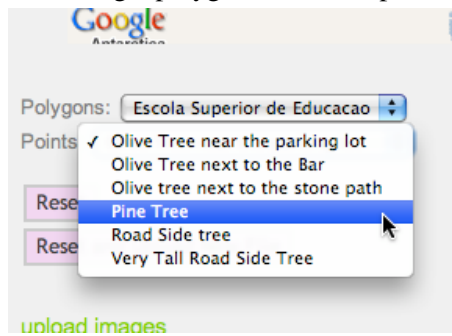


Figure 7.5: Selecting a point from the uploaded KML file

Demonstration Client

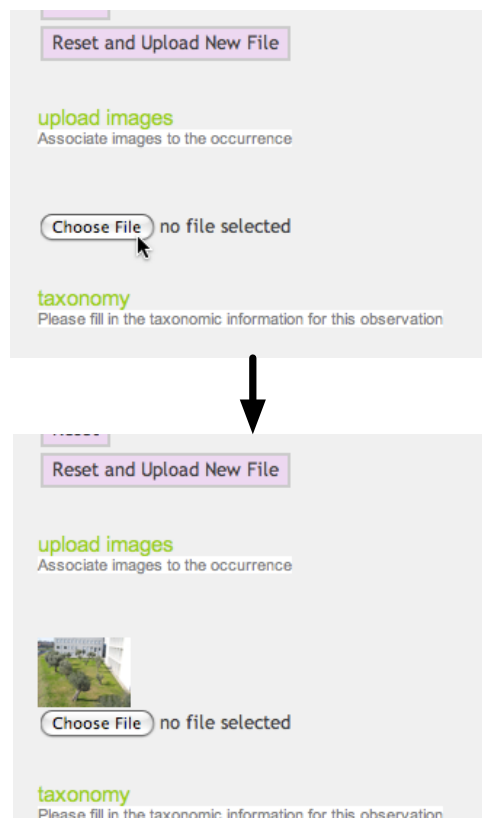


Figure 7.6: Uploading an image

Demonstration Client

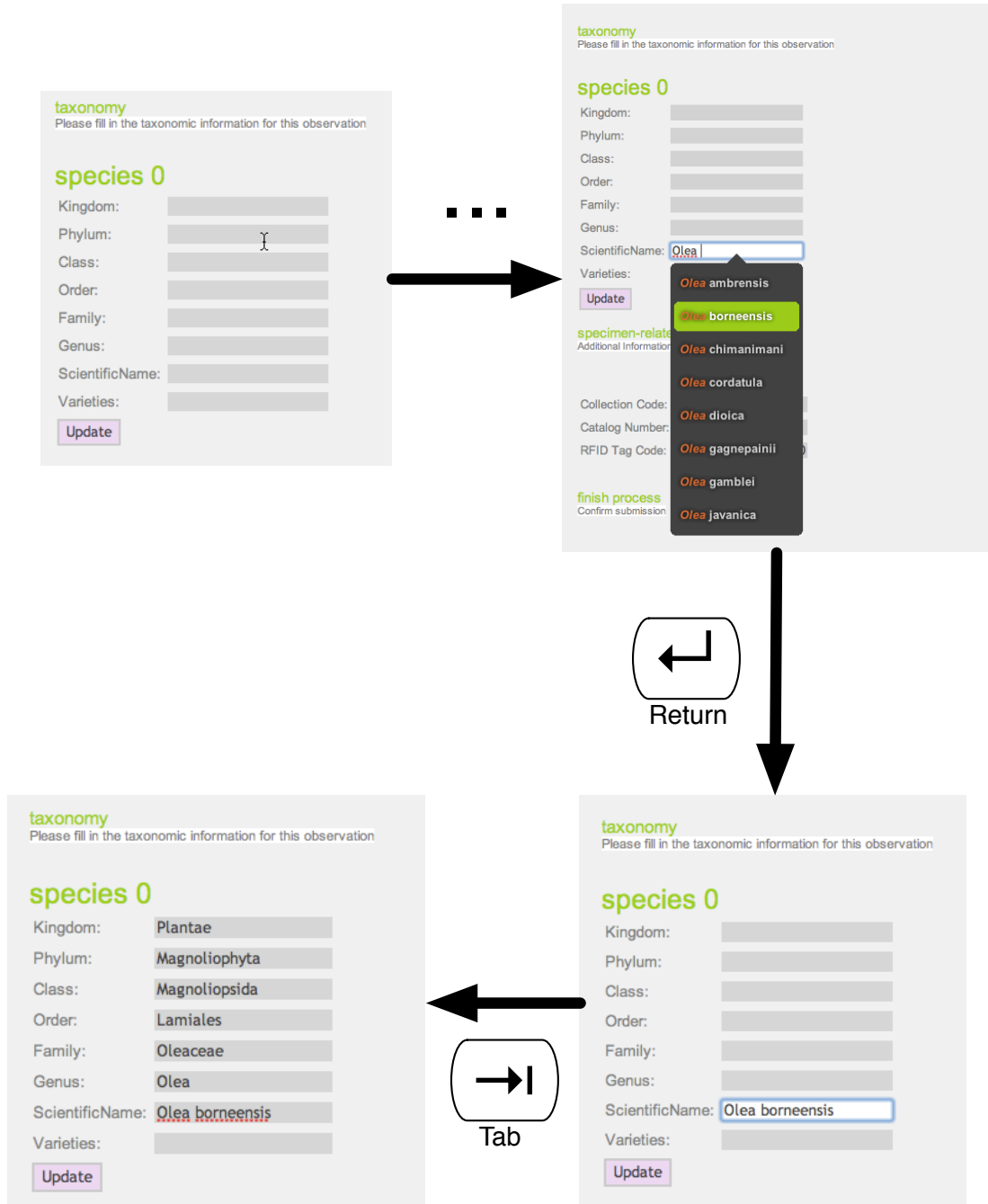


Figure 7.7: Auto-completion of taxonomic information, powered by AJAX

Demonstration Client

taxonomy
Please fill in the taxonomic information for this observation

species 0

Kingdom:

Phylum:

Class:

Order:

Family:

Genus:

ScientificName:

Varieties:

Figure 7.8: Add Species Button, available when the user selects a polygon as the location of the occurrence

taxonomy
Please fill in the taxonomic information for this observation

species 0

Kingdom:

Phylum:

Class:

Order:

Family:

Genus:

ScientificName:

Varieties:

species 1

Kingdom:

Phylum:

Class:

Order:

Family:

Genus:

ScientificName:

Varieties:

species 2

Kingdom:

Phylum:

Class:

Order:

Family:

Genus:

ScientificName:

Varieties:

specimen-related information
Additional Information

Collection Code:

Catalog Number:

RFID Tag Code:

Tooltip:
r.var.arborescens
S.r.var.racemosa

Figure 7.9: The new taxonomic information areas are also AJAX-enabled, with support for all taxonomic levels

specimen-related information
Additional Information

Collection Code:

Catalog Number:

RFID Tag Code:

Figure 7.10: Areas to insert occurrence-related identification information

Demonstration Client

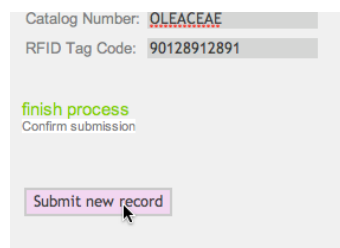


Figure 7.11: Submit record button

7.2.2 Search Occurrences

The occurrence search page can be seen in Figure 7.12. It includes three sections: an area for specifying spatial parameters, another for text-based parameters and series of options that can be used to select the output format for the search results. The first area displays a file upload box for KML files. When the user selects a valid KML file, this area will change into a map visualisation showing the contents of the file; it will also display some controls that will allow the user to add spatial restrictions to the search.

An example of a specimen search will now be shown. The user can search for specimens using just text-based restrictions, but since the main objective of the client is to demonstrate the spatial querying capabilities of the repository, we start with specifying some of these restrictions.

First, the user must create a KML file in Google Earth containing a series of polygons that can be used for building the query. An example polygon is shown in Figure 7.13. After saving the spatial entities as a KML file, the user can upload it to the search page using the file upload box at the top of it (Figure 7.12). The page will then show a map visualisation of the uploaded file 7.14. Also, the user may then specify a series of restrictions to include in the query. These are pairs composed by of a spatial operator and a polygon obtained from the KML file (see Figure 7.15). Text-based restrictions are also possible and can be combined with spatial restrictions. To specify a text-based restriction, the user must first select one of the available text parameters, then choose if an exact match is required or if it is sufficient to have only partial matches. Then, the user inserts the value to match to the parameter in the text box on the right. This procedure is shown in Figure 7.16.

After selecting the format that should be used for displaying the results of the search, the user completes the search by selecting the "Get Results" button (Figure 7.17). If the user selects "KML (Download)" or "Darwin Core" as the output format, the system will return the results in as a KML or Darwin Core XML formatted file.

If the selected output format was KML (View) the results will be shown in a Google Maps visualisation, as seen in Figure 7.18. By clicking on each of the placemarks (the round, pin-like banners), the user can bring up the details on that specific occurrence.

If the user wishes to save a copy of the search results, this can be done by selecting the "Download results" button at the bottom of the results page (Figure 7.19). Viewing the file in Google Earth, we can see that the placemarks contain the same information as the ones seen in the Google Maps visualisation. Also, the results are separated from the arguments used in the query. This enables the user to keep track of what were the areas of interest that originated the query and the results obtained. It also makes it easy to activate and deactivate the overlays for easy viewing, by ticking the boxes on the left.

Results can be polygons or points, as we have already seen. Polygons will represent

search database

geospatial parameters
Location-Based Restrictions

Valid polygons will be listed here for selection after uploading a KML file. no file selected

text parameters
Simple Text-Based Restrictions

output format
Select the desired output format for the results

☒ KML (View) ☐ KML (Download) ☐ Darwin Core

Figure 7.12: The main occurrence search page

areas where it is known that specific species of plant grow, but users have no interest in cataloguing each individual specimen (forests or beds). More than one species can be associated to the area, and that is also shown in the map visualisations. As Figure 7.21 shows, an area where several species of recorded will show the taxonomic tree classifications of all those species.

Demonstration Client

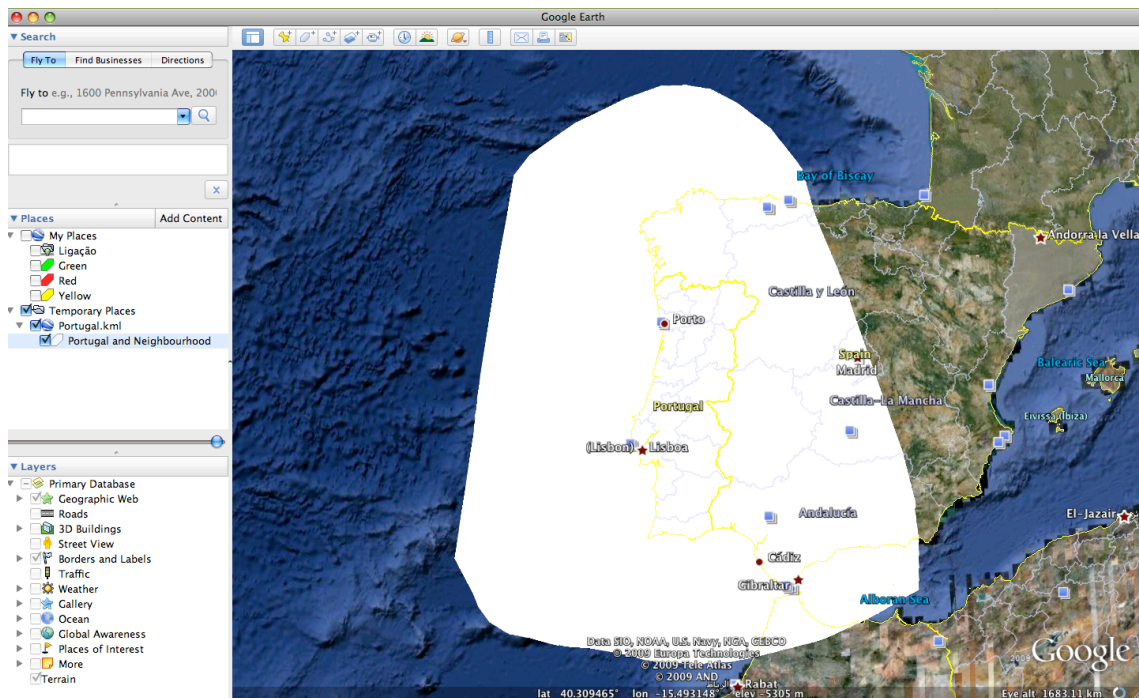


Figure 7.13: Creating a KML file in Google Maps

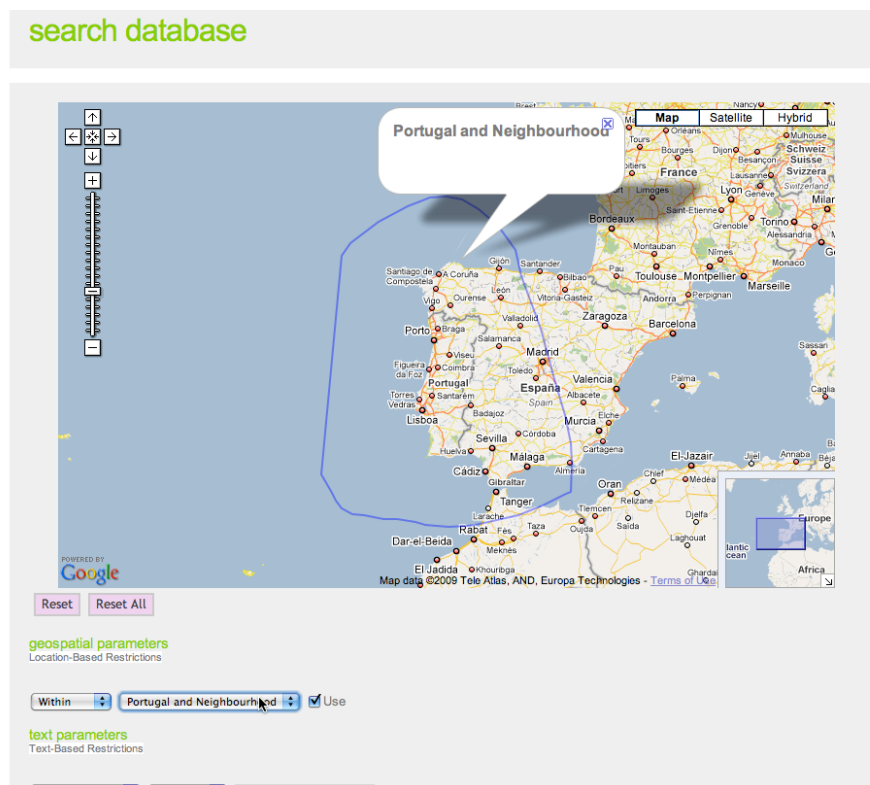


Figure 7.14: The uploaded KML file is shown by the system. Its polygons are shown in the combo box at the bottom

Demonstration Client

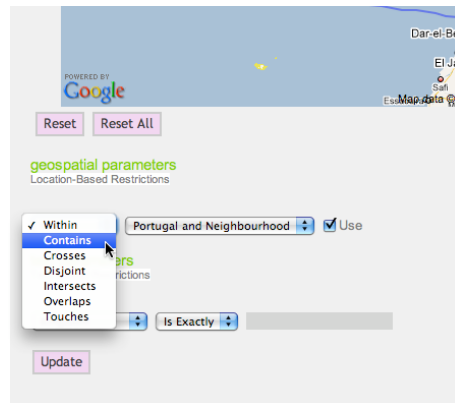


Figure 7.15: The user can add several spatial restrictions to the query, using the polygons from the KML file

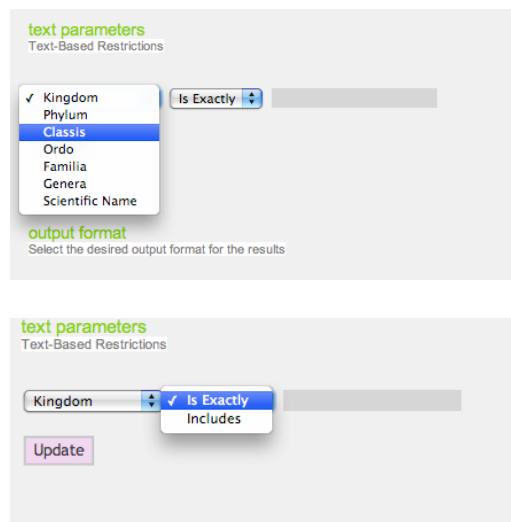


Figure 7.16: Common text-based criteria is also available

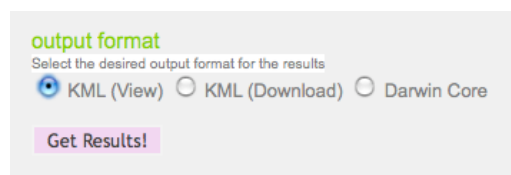


Figure 7.17: Selecting the output format for results output

Demonstration Client

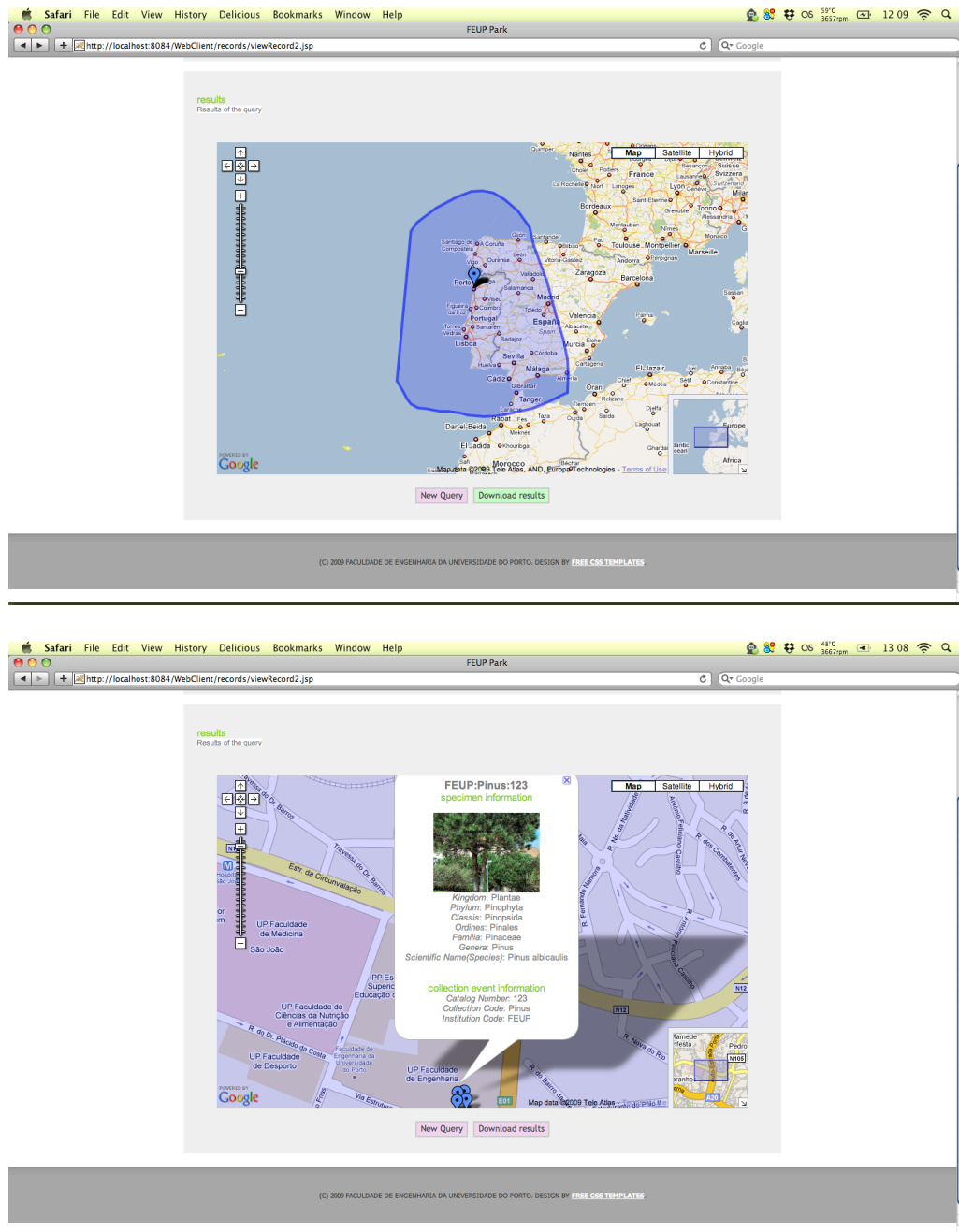


Figure 7.18: Viewing Query Results in Google Maps

Demonstration Client

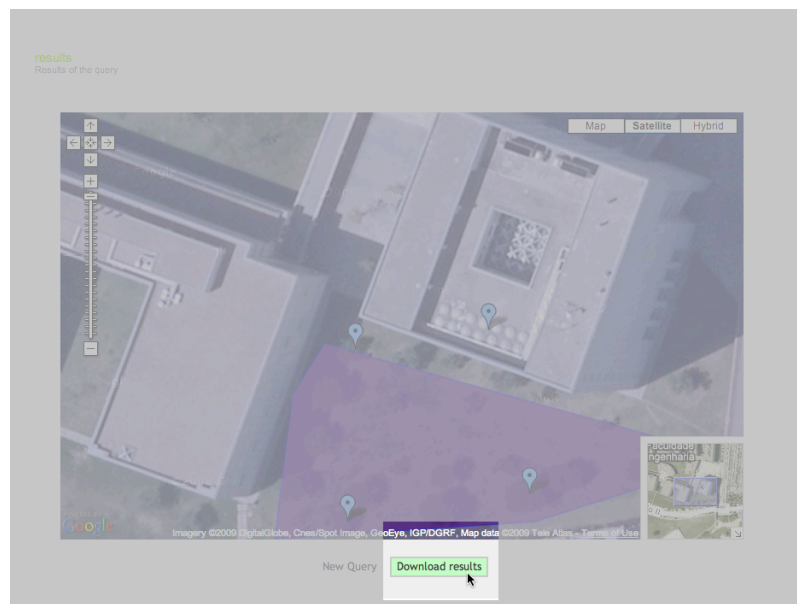


Figure 7.19: The user can download the query results and view them in Google Earth

Demonstration Client

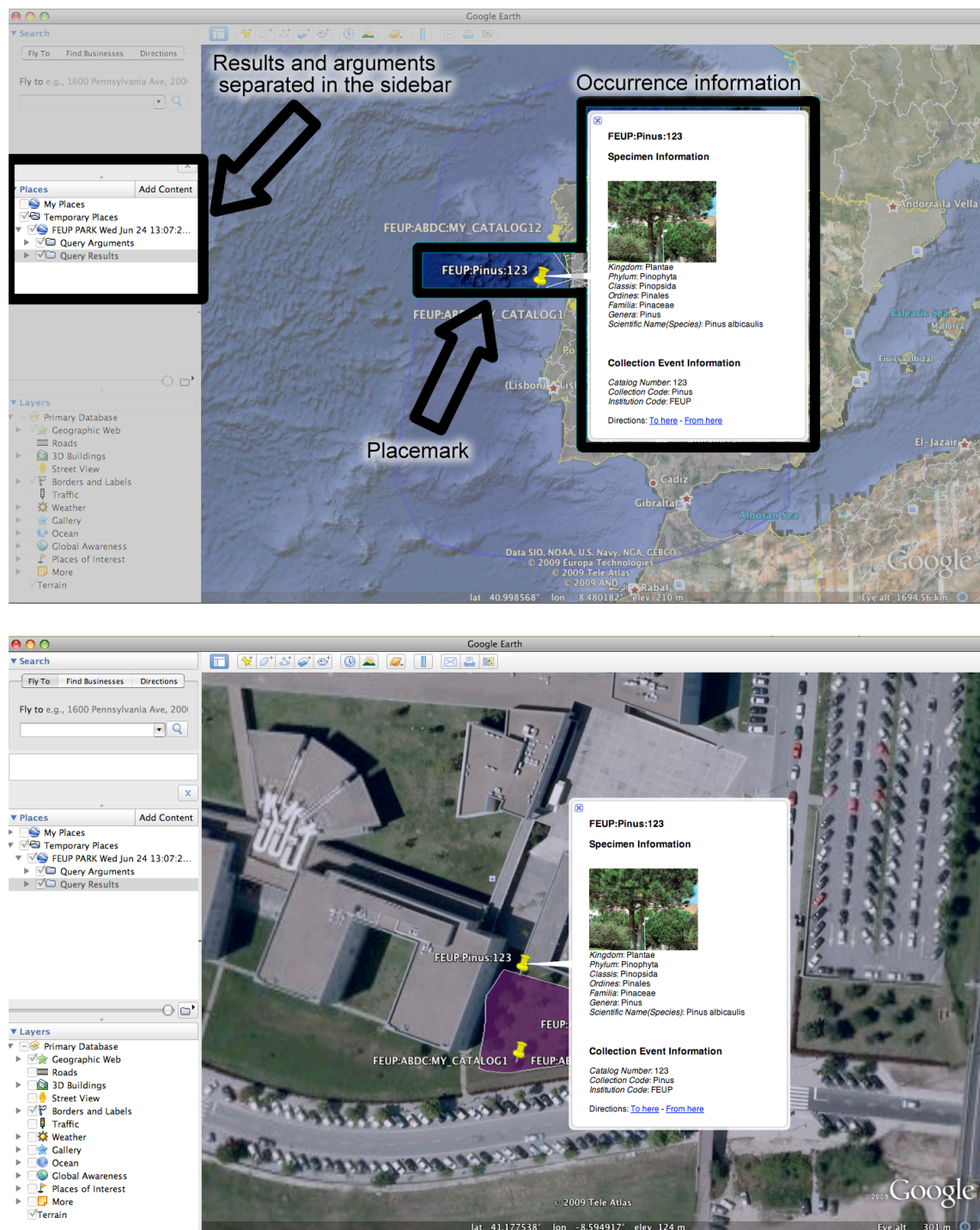


Figure 7.20: Viewing Query Results in Google Earth. The image on the bottom shows a more close view of the records

Demonstration Client

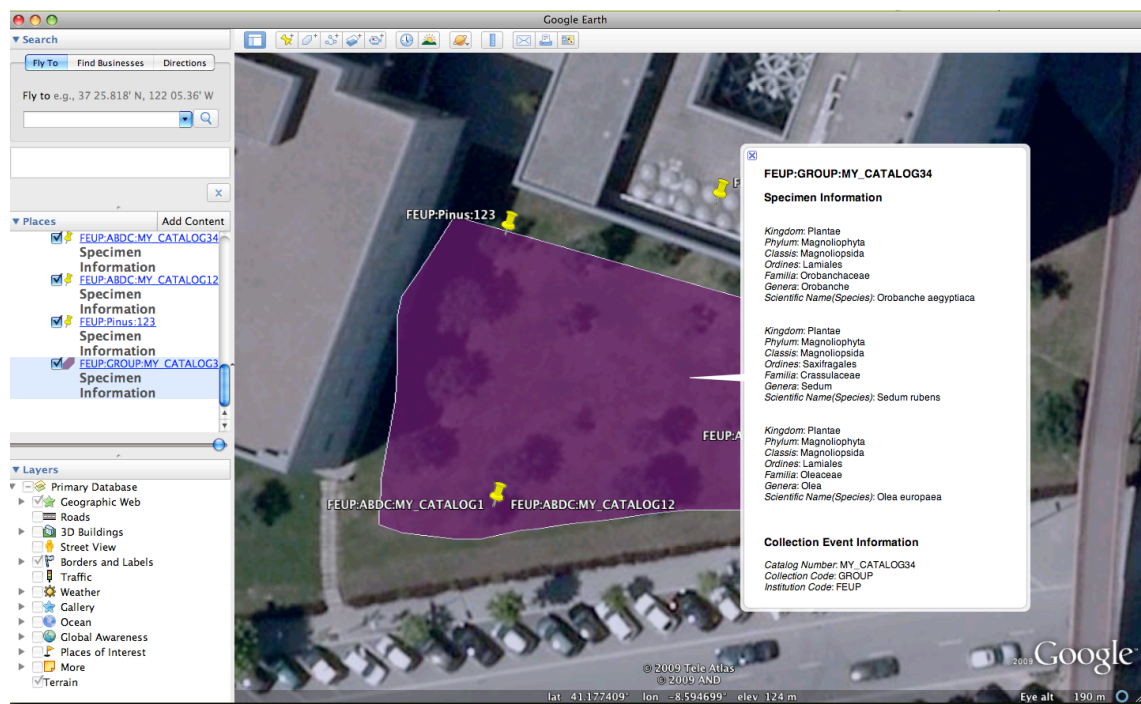


Figure 7.21: Polygons where there are several species will show all the different taxonomic trees

7.2.3 Create Tour

t:8084/WebClient/tours/addTours.jsp

Google

feup park
A SOLUTION FOR BOTANICAL DATA RETRIEVAL USING GEOSPATIAL DATABASES

home add search add tour tours about

create new tour - step 1

geospatial parameters
Location-Based Restrictions

Valid polygons will be listed here for selection after uploading a KML file. no file selected

text parameters
Simple Text-Based Restrictions

Kingdom Is Exactly

get results
Update occurrence selection boxes for tour building

Figure 7.22: Create Tours - Main Page

The user must perform a query, like was shown in the previous section (7.2.2). Results will be shown in a list, so that the user can order them and insert the name of the tour and its description (Figure 7.23). The user may then save the tour by clicking the "Submit Tour" button.

Demonstration Client

tour order
Select and order points of interest.

Available Observations

- FEUP:ABDC:MY_CATALOG1
- FEUP:ABDC:MY_CATALOG12
- FEUP:ABDC:MY_CATALOG34

Selected Observations

- FEUP:GROUP:MY_CATALOG34
- FEUP:Pinus:123

Buttons: Add >> Move up Move down Remove

create new tour - step 3

name and description
Name this tour and describe it

Navigation: << Back Submit Tour >>

Tour name: Olive Trees Tour

Tour Description: "Insert tour description"

create new tour - step 3

name and description
Name this tour and describe it

Navigation: << Back Submit Tour >>

Tour name: Olive Trees Tour

Tour Description: Here are some interesting trees, a staple of our gardens!

Figure 7.23: Ordering occurrences in the tour and Inserting tour information

7.2.4 View Tours

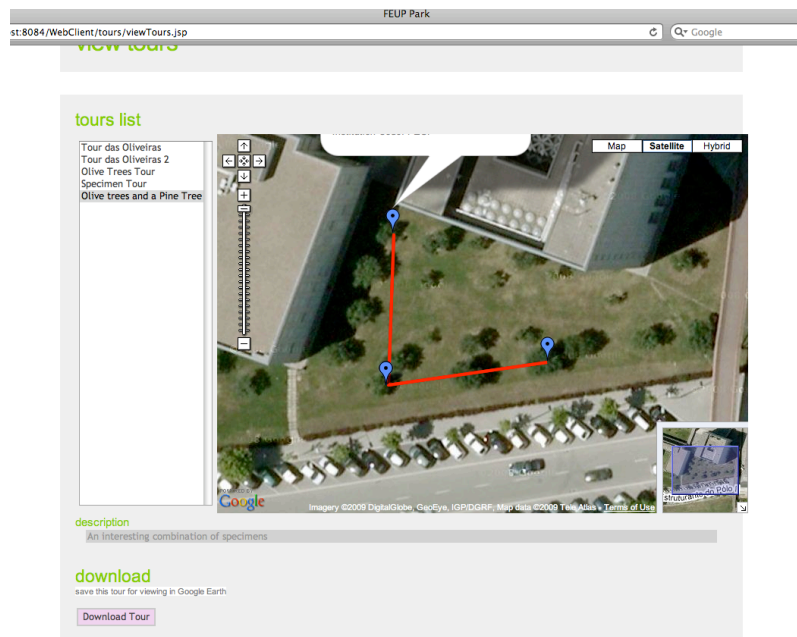


Figure 7.24: Tour viewing page. The tour paths and individual specimen information is shown inside each placemark balloon.

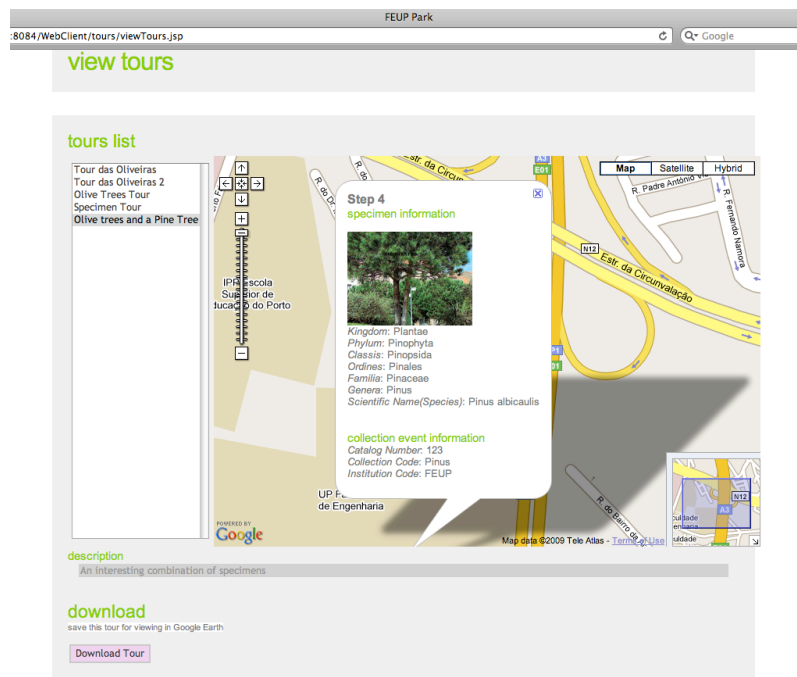


Figure 7.25: Details for a specimen present in the tour (Step 4 of the tour path).

To view the tours available in the botanical park, the user has to access the "View Tours" page and select the desired tour from the list on the left (Figure 7.24). The results

will then be shown in a Google Maps visualisation on the right of the list, including the path that visitors should follow. By clicking the specimen placemark banner, the user can see additional information on that specimen. Note that the tour order is shown at the top of the balloon itself. These features can be seen in Figure [7.25](#) (note the title of the "balloon": *Step 4*).

Chapter 8

Conclusions

Botanical information systems are undergoing a major change in the way they catalogue their specimens. This is because of two emerging issues: georeferencing and easy data exchange between different systems.

Georeferencing is assuming an important role in these systems because the indexing of written records are being complemented with more geographically-oriented approaches. The most prominent botanical information systems [[Sec09](#), [Cal](#)] are building sophisticated modules to approximate geographic locations for each specimen identification.

This common drive towards more precise spatial information is an opportunity for the development of a repository that can help gather, classify and query botanical specimens, using their geographic locations. These are commonly called GIS solutions. Using GIS allows for higher flexibility for locating botanical specimens.

To tackle the data exchange issue, large botanical information systems are also developing solutions designed to aggregate several different data sources, translating their information into community standard formats [[DP07](#)]. In this repository we have also covered this issue. Platform-independent services are made available for inserting and querying specimen information. This information is also exchanged in domain-standard formats to encourage the adoption of such a system by potential clients. These clients can cover a wide range of scenarios; to demonstrate the spatial querying capabilities of the repository, a prototype client for a botanical park ("FEUP Park") was developed.

This system shows how botanical information systems and Botany in general can benefit from open spatially-enabled botanical information repositories.

8.1 Future Work - General

Some future development perspectives will now be presented. This project is to be a part of a larger one, so it makes sense to leave some thoughts on future features that can be

built using the developed framework. Others are support features that aim at enhancing the quality of its information.

8.1.1 A Social Network on Botany

Nowadays social networks are some of the fastest growing web sites. Whenever there is a group of people interested in similar subjects, there is a website to fulfil the need for these people to communicate and share their experiences.

The proposed repository relies on its openness to make occurrence insertion less complicated; it also hopes to gather a larger amount of information than conventional, more "closed" databases. An interesting client for the system would be a social network where people could share their botanical findings. Then, their peers could validate them and post comments on the botanist's work, for example. They could also join groups of people interested in specific species of plants. Another interesting feature would be the sharing of tours. Personalised paths could be shared among users, escaping the rigidity of common botanical databases. At the same time, the repository would grow more quickly.

8.1.2 Peer Review

This idea is somewhat similar to the social network proposal, and may be one of its features.

When a new occurrence is added to the system, the species selected by the botanist at insertion might not be correct. Even though the taxonomic tree (the chain of *taxons*) is correct, the species of the plant might have been confused with another. Validating the submitted records one by one would require a great deal of manpower, so an alternative is to create a *peer review* process where submitted occurrences could be flagged as incorrect by other members. After public scrutiny, the occurrences could be removed by an Administrator of the system, or kept in store. This *peer review* process could also be used to reduce invalid or redundant data, such as duplicate occurrences.

8.1.3 Paper Repository

Google Scholar [[Goo09](#)] is a website that gathers information about papers and articles, among other academic publications. An interesting measure of importance for a publication is the number of papers that *cited* it. Since the repository holds a large volume of specimen information, it might make sense to develop a repository for papers that reference certain specimen occurrences present in this developed botanical repository.

8.1.4 Handheld Terminal

Any database has to grow and provide useful information for its users. Since this is a collaborative effort between botanical databases and botanical parks, it would be of great interest to offer a simple means for these contributors to publish their information via mobile devices such as mobile phones and PDA's. Since the server only requires interaction via XML, client applications written using J2ME or Windows Mobile can be connected to the server for specimen insertion or viewing. All that is required is Internet connectivity and the corresponding XML parsers (for J2ME, see [Knu02], for Windows Mobile, see [Cor09]).

8.1.5 XML Data Compression

Some problems with handling large volumes of information using XML were mentioned in Section 3.3.4. A compression algorithm could be used to reduce the size of the XML documents sent between the server and its clients.

8.1.6 Data Mining as a Validation Tool

To further improve the quality of the information stored in the system, a data mining system could be implemented. This kind of validation was used in [DP07] to identify *outlier* values in the occurrence data set. In this case, this would pinpoint incorrect data by identifying groups of occurrences; if a specific occurrence was recorded in an area far away from all the others, the system would signal that occurrence for deletion. This alternative, when combined with the *peer-review* process already proposed, would grant the repository's data set a high level of correctness.

8.2 New features for the FEUP Park Web Application

8.2.1 Writing a SAX Parser

The current implementation of the "FEUP Park" web application uses a DOM parser to produce its queries and process the results received from the repository. DOM does not handle large XML documents very well. SAX parsers, on the other hand, are more adequate for handling this sort of files, so it may be interesting to change the implementation if the need should arise; for example, if there is large group of clients that need to perform much more complex queries are than those currently expected.

8.2.2 Expert Comments Via Portable Terminals

Usually, the tours are specified by the park owners or other people of high responsibility, and can be either *static* or not. When tours are *static*, visitors are lead through the same path within the park on every visit, or a given number of tours. It can be difficult to set up different tours for different types of users, because this would require many more park employees in order to accompany all the different visitor parties.

A mobile terminal can gather information about the location of the visitor at any time. This information can be acquired via GPS or RFID sensors.

The notifications shown on the mobile terminals can include several types of media:

- Description text
- Images
- Correct pronunciation of the species' names
- Audio comments by experts

When an occurrence search is executed, results are returned as a KML file linking to media files hosted in the park's web server. Another alternative could be the the download of a KMZ file containing the resources themselves. This is an extension to the feature already present in the FEUP Park client; currently it only supports the inclusion of images inside the small "balloons" over each place mark present in the map visualisation (Google Maps or Google Earth). These images are also stored in the repository and not in the park's local database.

8.2.3 Locating the Visitor Inside the Park

GPS Technology For Navigation

If there is unrestricted visibility between the sky and the mobile terminal, it can send periodic GPS *heartbeats* to a central server. These *heartbeats* report the position of the terminal to the server. When the user enters a specific area where a specific species exists, a notification can appear on the mobile terminal.

This sort of location-based service would provide point-based notifications (when the user gets within close distance of a specific specimen). It would also help provide adequate notifications when a user enters an *area* where specific types of plants grow.

RFID Applications

Another alternative for tracking the paths followed by park visitors would be to place RFID tags on the park's specimens. After reading the RFID tag (by coming close to it)

Conclusions

the terminal would send the identification of the specimen to the server. The user would then receive the same sort of notifications shown on the previous scenario. In this case, however, they would be shown only when the user went inside the area coverage for that specific RFID tag. Wide areas, such as forests or beds would not be identifiable using this method. However, the heavy *polling* process used in [8.2.3](#) would be eliminated.

Both of these scenarios would require Wi-Fi coverage throughout the botanical park for the portable terminal to report to the server and receive the replies.

References

- [209] FishNet 2. Fishnet 2 home, March 2009. Consulted on <http://www.fishnet2.net/index.html>.
- [Aut09a] Several Authors. Darwin core, March 2009. Consulted on <http://rs.tdwg.org/dwc/index.htm>.
- [Aut09b] Several Authors. Darwincore, March 2009. Consulted on <http://wiki.tdwg.org/twiki/bin/view/DarwinCore/WebHome>.
- [Avo09] Avoir. Postgis class, March 2009. Consulted on <http://gforge2.uwc.ac.za/download.php/48/postgis-docs.zip>.
- [Blu09] Stanley Blum. Purpose and design goals of the darwin core, March 2009. Consulted on <http://wiki.tdwg.org/twiki/bin/view/DarwinCore/DesignAndPurpose>.
- [Cal] Calflora. Calflora map viewer, March. Consulted on <http://www.calflora.org/app/zmapview>.
- [Col09] BioGeomancer Collaboration. Geospatial extension concept list, March 2009. Consulted on <http://wiki.tdwg.org/twiki/bin/view/DarwinCore/GeospatialExtension>.
- [Con09] OpenGIS Geospatial Consortium. Compliance certificate, March 2009. Consulted on <http://www.postgis.org/files/opengis-certification.png>.
- [Cor09] Microsoft Corporation. Xml core services and dom for windows mobile-based devices, May 2009. <http://msdn.microsoft.com/en-us/library/ms851601.aspx>.
- [DP07] Xiaofang Zhou David Pullar, Jack Fan Zhang. Integrative environmental queries using geospatial web services. 2007.
- [For09] MySQL Forge. W12377: Gis: Add all missing gis features to mysql, March 2009. Consulted on <http://forge.mysql.com/worklog/task.php?id=2377>.
- [Fou09a] National Science Foundation. Herpnet, March 2009. Consulted on <http://www.herpnet.org/>.
- [Fou09b] The Apache Software Foundation. Commons codec - home, May 2009. <http://commons.apache.org/codec/>.

REFERENCES

- [GA08a] Jim Graham and Jennifer Ackerfield. Herbarium search, Unknown 2008. Consulted on <http://wsprod.colostate.edu/cwis440/herbarium/index.asp>.
- [GA08b] Jim Graham and Jennifer Ackerfield. Information for *abronia elliptica* (fragrant white sand verben), Unknown 2008. Consulted on <http://wsprod.colostate.edu/cwis440/herbarium/plantinfo.asp?PlantID=2082>.
- [GDA09a] GDAL. Ogr vector formats, March 2009. Consulted on http://www.gdal.org/ogr/ogr_formats.html.
- [GDA09b] GDAL. ogr2ogr, March 2009. Consulted on <http://www.gdal.org/ogr2ogr.html>.
- [Geo09] Georuby. Georuby + spatial adapter, March 2009. Consulted on <http://georuby.rubyforge.org/>.
- [GN05] Robert Guralnick and David Neufeld. Challenges building online gis services to support global biodiversity mapping and analysis: Lessons from the mountain and plains database. *Biodiversity Informatics*, pages 56–69, February 2005.
- [Goo09] Google. Google scholar, Consulted on May 2009. <http://scholar.google.pt/>.
- [Gre09] Mrs. M. Greive. A modern herbal, February 2009. Consulted on <http://www.botanical.com/botanical/mgmh/comindx.html>.
- [Gro09a] BioGeomancer Working Group. Geomancer workbench, March 2009. Consulted on <http://bg.berkeley.edu/latest/>.
- [Gro09b] Network Working Group. Multipurpose internet mail extensions (mime) part one: Format of internet message bodies, May 2009. <http://www.ietf.org/rfc/rfc2045.txt>.
- [IEE98] IEEE. Ieee standard 830-1998. June 1998.
- [Inc09a] Barcoding Inc. How much does an rfid tag cost?, March 2009. Consulted on <http://www.barcoding.com/faq/rfid-tag-cost.shtml>.
- [Inc09b] O'Reilly Media Inc. What is xml?, March 2009. Consulted on <http://www.xml.com/pub/a/98/10/guide0.html>.
- [JGI05] J.Estrada J. Gaviria and D.Hernandez I.Besembel. Botanical database modelling. February 2005.
- [JHQ06] James H. Thorne Joshua H.Viers and James F. Quinn. Caljep: A spatial distribution database of calflora and jepson plant species. *Science*, pages 1–18, 2 2006.
- [JW04] RJ Hijmans J. Wieczorek, Q. Guo. The point-radius method for georeferencing locality descriptions and calculating associated uncertainty. *Biodivers Converg*, pages 745–767, 2004.

REFERENCES

- [Knu02] Jonathan Knusden. Parsing xml in j2me, March 2002. <http://developers.sun.com/mobility/midp/articles/parsingxml/>.
- [LE09a] Andre Lewis and Bill Eisenhauer. Geokit - a ruby gem rails plugin for easier map-based applications, March 2009. Consulted on <http://geokit.rubyforge.org/readme.html>.
- [LE09b] Andre Lewis and Bill Eisenhauer. Geokit plugin: Geocoder, distance finders, and more, March 2009. Consulted on <http://geokit.rubyforge.org/api/geokit-rails/index.html>.
- [MA09] Sun Microsystems MySQL AB. Mysql conformance and compatibility, March 2009. Consulted on <http://dev.mysql.com/doc/refman/5.0/en/mysql-gis-conformance-and-compatibility.html>.
- [Nex09] Inc. International NexTag. Rfid reader - electronics accessories - compare prices, reviews and buy at nexttag - price - review, March 2009. Consulted on <http://www.nextag.com/serv/main/buyer/OutPDir.jsp?search=rfid+reader&psort=12&node=300151>.
- [oA09] United States Department of Agriculture. The plants database, March 2009. Consulted on <http://plants.usda.gov/>.
- [oEN09] Department of Environment and Climate Change NSW. Plant dna sequencing - botanic gardens trust - sydney, australia, Consulted on May 2009. http://www.rbgsyd.nsw.gov.au/about_us/businesses_and_opportunities/plant_dna_sequencing.
- [OGC95] Inc. Open GIS Consortium. Opendgis simple features specification for sql revision 1.1, May 1995. Consulted on http://portal.opengeospatial.org/files/?artifact_id=829.
- [otUoC09] The Regents of the University of California. Mammal networked information system, March 2009. Consulted on <http://manisnet.org/>.
- [PCdOF05] Atilio Antonio Disperati et al. Paulo Costa de Oliveira Filho. Um sistema de informacoes geograficas como suporte a um experimento florestal na flona de irati-pr. April 2005.
- [PCM04] et al. Paul C. Murphey, Robert P. Guralnick. Georeferencing of museum collections: A review of problems and automated tools, and the methodology developed by the mountain and plains spatio-temporal database- informatics initiative (mapstedi). *PhyloInformatics* 3, September 2004.
- [PoHC09] President and Fellows of Harvard College. Harvard university herbarium, May 2009. Consulted on http://asaweb.huh.harvard.edu:8080/databases/specimen_index.html.
- [PVA09] Manuela D. L. Ramos Paulo V. Araújo, Maria P. Carvalho. Dias com Árvores, Consulted on May 2009. <http://dias-com-arvores.blogspot.com/2004/09/jardim-das-virtudes-visto-da-rvore.html>.

REFERENCES

- [Rei08] Don Reisinger. Analyst: Google maps reigns supreme, September 2008. Consulted on <http://www.techcrunch.com/2008/09/29/analyst-google-maps-reigns-supreme/>.
- [Sec09] GBIF Secretariat. Gbif portal - home, March 2009. Consulted on <http://www.gbif.org>.
- [SHB03] Denise Wilson Sean Hoyt, Doug St. John and Linda Bushnell. A tree tour with radio frequency identification (rfid) and a personal digital assistant (pda). pages 2035–2040, 2003.
- [SLG06] Toshimi Minoura et al. Stephanie L. Greene. Webgrms: Prototype software for web-based mapping of biological collections. *Biodivers Conserv*, August 2006.
- [Sys09] ORNithological Information System. Ornis, March 2009. Consulted on <http://olla.berkeley.edu/ornisnet/>.
- [W3C04] W3C. Xml schema part 0: Primer second edition, October 2004. <http://www.w3.org/TR/xmlschema-0/>.
- [W3C09] W3C. Extensible markup language (xml) 1.0 (fifth edition), March 2009. Consulted on <http://www.w3.org/TR/REC-xml/#sec-origin-goals>.
- [Wik09a] Wikimedia. Create, read, update and delete, March 2009. Consulted on http://en.wikipedia.org/wiki/Create,_read,_update_and_delete.
- [Wik09b] Wikimedia. Global positioning system, March 2009. Consulted on <http://en.wikipedia.org/wiki/GPS>.
- [Wik09c] Wikimedia. Plant, March 2009. Consulted on <http://en.wikipedia.org/wiki/Plant>.
- [Wik09d] Wikimedia. Taxon, March 2009. Consulted on <http://en.wikipedia.org/wiki/Taxon>.
- [Wik09e] Wikimedia. Taxonomy, March 2009. Consulted on <http://en.wikipedia.org/wiki/Taxonomy>.
- [Wik09f] Wikimedia. Wikispecies, free species directory, March 2009. http://species.wikimedia.org/wiki/Main_Page.

Appendix A

Server Relational Schema

Server Relational Schema

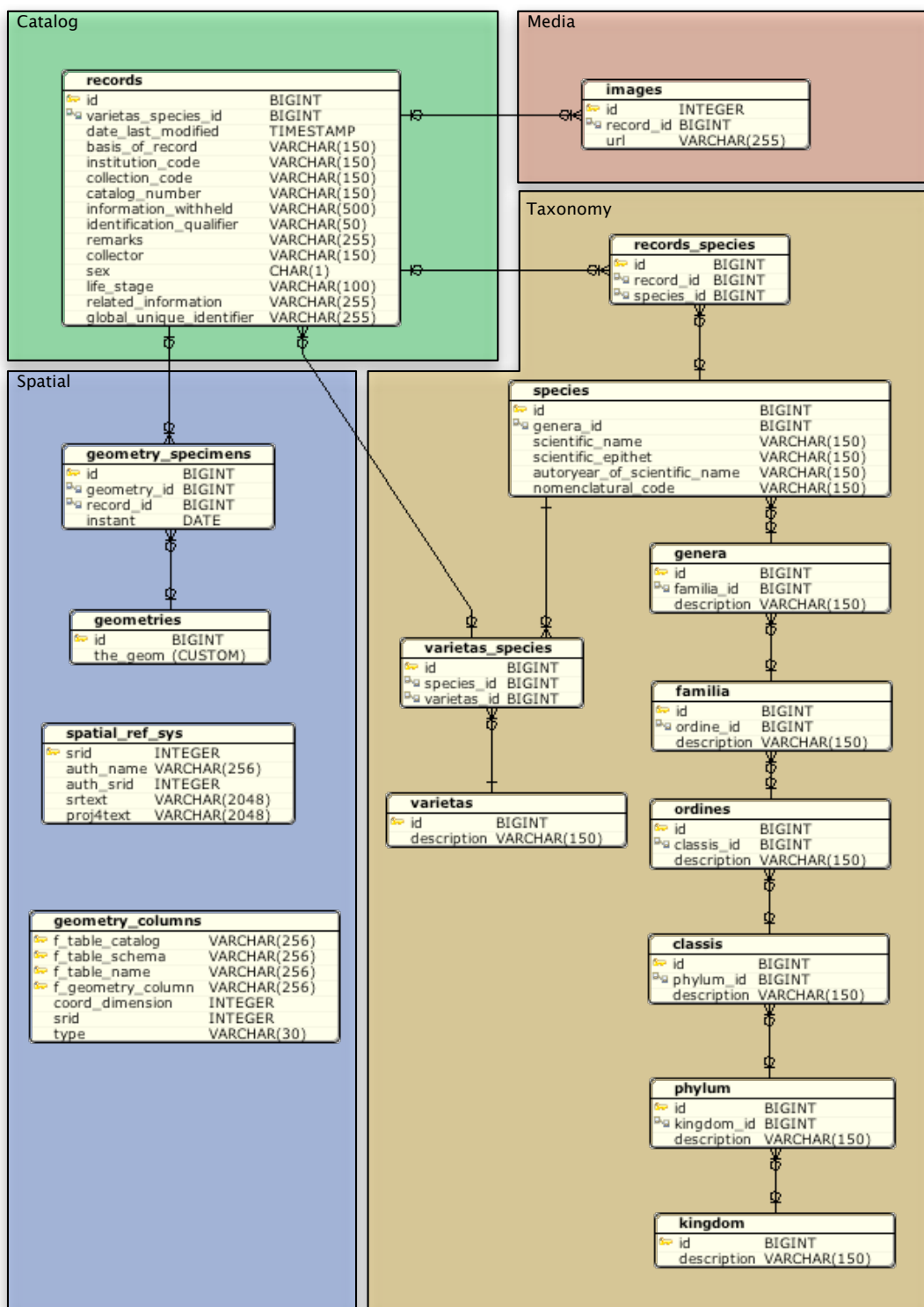


Figure A.1: Relational schema of the repository database

Appendix B

Client Application Requirements

B.1 Use Case Diagrams

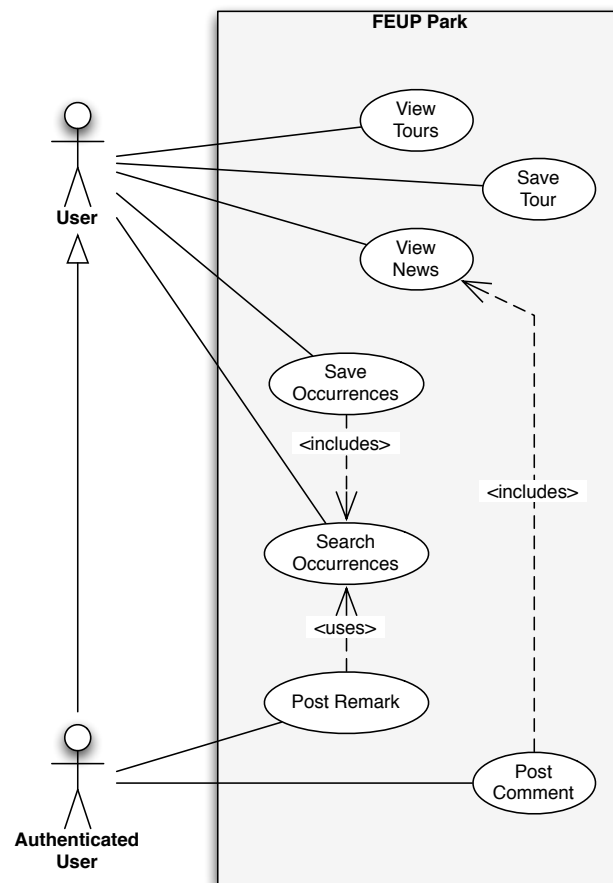


Figure B.1: Use Case diagram for Users and Authenticated Users

Client Application Requirements

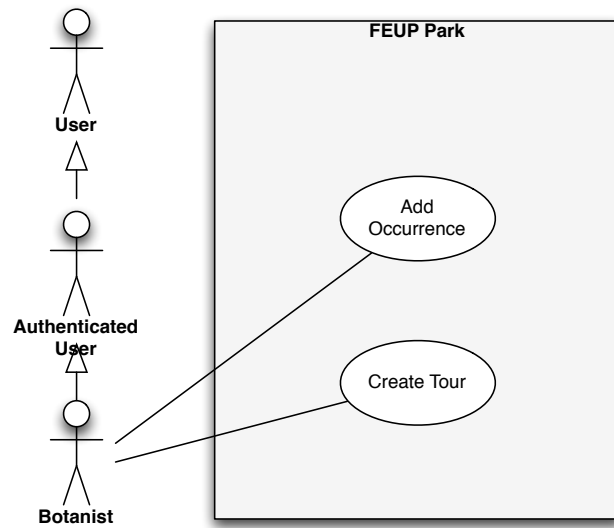


Figure B.2: Use Case diagram for Botanists

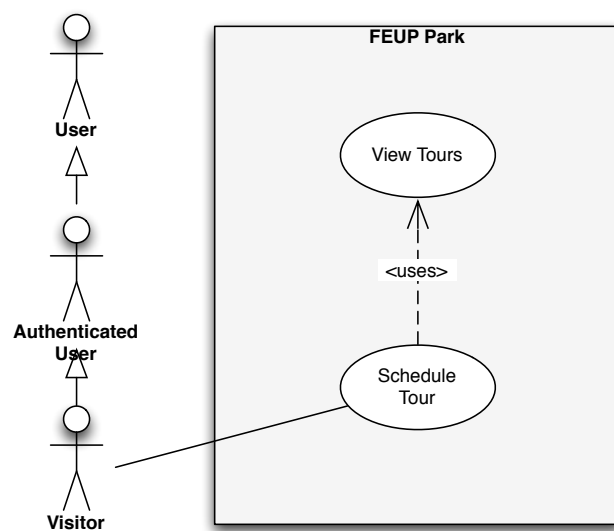


Figure B.3: Use Case diagram for Visitors

Client Application Requirements

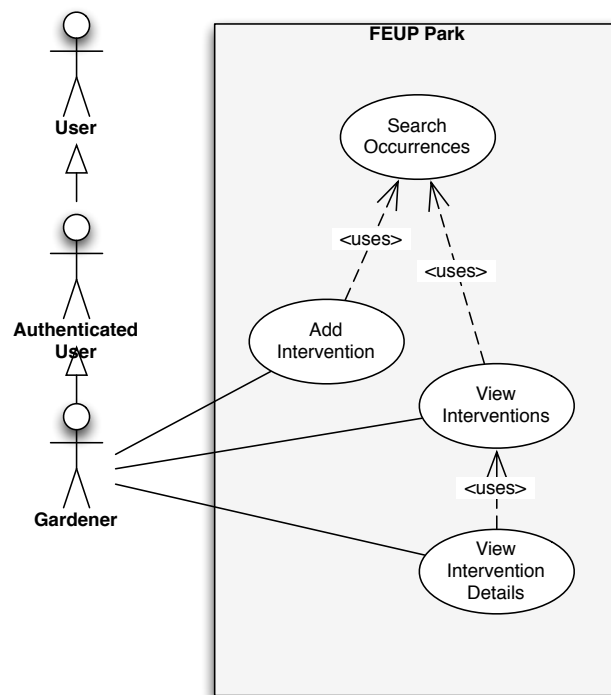


Figure B.4: Use Case diagram for Gardeners

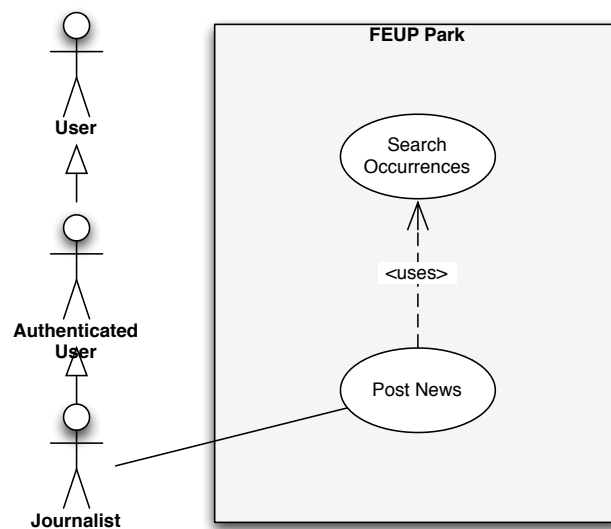


Figure B.5: Use Case diagram for Journalists

B.2 Use Case Specifications

Identifier	UC-1
Use Case name	Search Occurrences
Actors	User
Description	This Use Case allows users to search for botanical occurrences, specifying both spatial and conventional criteria.
Preconditions	There must be an established network connection between the client and the remote repository
Post-Conditions	The system successfully records the query; If output visualisation is selected during the specification of the search parameters, this output is successfully shown to the user. If the user selected KML or Darwin Core as the output format, the system must offer the user an option to download the results in the form of a file, for off-line viewing.
Normal execution flow	1. The user selects a KML file for upload 2. The system lists the polygons contained in the uploaded file and a visual representation of these polygons 3. Addition of spatial restrictions (may repeat several times) (a) The user selects a spatial operator from a list of available spatial operators (b) The user selects a polygon from the list of available polygons 4. Addition of conventional restrictions (may repeat several times) (a) The user selects a taxonomic level from a list (for example "Genus") (b) The user chooses between an exact or partial match between the records to be listed and the value inserted in the parameter area (next step) (c) The user inputs the desired value (for example "Olea" as the <i>Genus</i> of the desired occurrences) 5. The user selects the desired output format for the results. Available choices are: Visualisation, KML and Darwin Core. 6. The user confirms the submission of the query. 7. Alternative • If the user selected the Visualisation option in the format selection area, the browser will show a map representation of the argument polygons used in the query, as well as a graphical representation of the results. This representation must include the full taxonomic classification of each occurrence, as well as any available images for the occurrence.
Alternative flows	The user does not select geospatial restrictions. All steps comprehended inside "Addition of spatial restrictions" are skipped.
Exceptions	An Invalid KML file is uploaded → The system shows an error message.
Priority	High
Usage frequency	It is expected that every user of the system will use this functionality at least once for the duration of an individual visit. A visit is intended to be the period of time since the user first entering the portal until the moment when the user navigates to a webpage outside the system's domain.
Notes	None

Table B.1: Search Occurrences Use Case specification

Client Application Requirements

Identifier	UC-2
Use Case name	Save Occurrences
Actors	User
Description	The user saves the results of an occurrence query
Preconditions	A successful query has been performed (UC-1)
Post-Conditions	The output file containing the last query's results is successfully downloaded to the client's machine.
Normal execution flow	1. After showing the visualisation described in UC-1, the user selects an option to download the results 2. The user is prompted to save or open the results file or cancel 3. The user selects Open or Save 4. The file is downloaded to the user's machine
Alternative flows	• The user cancels the download procedure, by selecting Cancel in the file download prompt • No file is downloaded
Exceptions	None
Priority	High
Usage frequency	Approximately the same as UC-1
Notes	In the case of results represented in a KML file, the query arguments must be correctly separated from the query results, so that when the file is opened in Google Earth, there is a clear distinction between the two.

Table B.2: Save Occurrences Use Case specification

Client Application Requirements

Identifier	View Tours
Use Case name	UC-3
Actors	User
Description	This feature allows users to browse the currently available tours in the Park.
Preconditions	There must be at least one Tour present in the system for display
Post-Conditions	The tour is shown to the user in a map visualisation
Normal execution flow	1. The user enters the area designed for tour viewing 2. The user selects the desired tour 3. A visualisation is shown to the user, depicting the several occurrences included in the tour. Occurrence information includes the taxonomic information and any images of the occurrence. This visualisation must also show the order in which occurrences are to be visited and a path joining all the geographical elements that represent the occurrences on the visualisation.
Alternative flows	None
Exceptions	There are no Tours available in the park → The system must show an error message to the user, offering to redirect to the previous page.
Priority	Medium
Usage frequency	It is expected that this functionality will be used as frequently as UC-1
Notes	None

Table B.3: View Tours Use Case specification

Client Application Requirements

Identifier	UC-4
Use Case name	Save Tours
Actors	User
Description	This enables users to download files containing tour information, for off-line viewing.
Preconditions	The user must have already selected a desired tour and its visualisation must have been shown (UC-3)
Post-Conditions	A file containing the information about the selected tour is correctly downloaded to the user's machine
Normal execution flow	1. The user follows the steps described in UC-4 2. The option to download the selected tour is selected 3. File Download Step: The system prompts the user for action (cancel, download or open the file) 4. Option: (a) The user selects "Open" and the file is opened with the default viewer (for example, for KML, Google Earth will be opened upon file download) (b) The user selects "Save" and selects the desired path to store the file 5. The file is downloaded to the client's machine
Alternative flows	1. The user cancels the download procedure at the File Download Step. 2. No file is downloaded nor opened
Exceptions	None
Priority	Medium
Usage frequency	Since not all users download all tours present in the system during a visit, it is expected that this Use Case will be less prevalent than UC-3
Notes	r10c2

Table B.4: Save Tours Use Case specification

Client Application Requirements

Identifier	UC-5
Use Case name	View News
Actors	User
Description	News concerning the botanical park and its specimens are displayed to website visitors.
Preconditions	There must be at least one news article present in the system.
Post-Conditions	News are displayed in the website's home page, ordered by date, in decreasing order
Normal execution flow	1. The user accesses the botanical park's start page 2. If there are news registered in the system, the last 4 are displayed in the home page.
Alternative flows	None
Exceptions	There are no news available → the system show an information message
Priority	Low
Usage frequency	News are expected to appear every time a visitor enters the website. It is therefore expected that this Use Case will be executed intensively by all visitors.
Notes	None

Table B.5: View News Use Case specification

Client Application Requirements

Identifier	UC-6
Use Case name	Post Remark
Actors	Authenticated User
Description	This feature allows users to post small messages regarding the recorded Occurrence. It is designed as support means for the process of <i>peer-review</i> (submitting the occurrence's taxonomic information to public scrutiny, for example)
Preconditions	The user must be registered and authenticated in the system
Post-Conditions	The remark is successfully associated to the selected Occurrence
Normal execution flow	1. The user accesses the occurrence search page 2. An occurrence search is conducted, similar to what is specified in UC-1 3. One of the resulting occurrences is selected 4. The user inputs a subject for the remark and the its content text 5. The remark is submitted 6. The system displays the occurrence search page for the user to view the new remark associated to the occurrence.
Alternative flows	After completing the execution flow of UC-1, no occurrences are returned → the system show a warning message and returns to the initial state
Exceptions	None
Priority	Low
Usage frequency	This feature will not be used frequently
Notes	Remarks can include URLs and emails. This data must be handled properly by the system. As an example, URLs may be very long.

Table B.6: Post Remark Use Case specification

Client Application Requirements

Identifier	UC-7
Use Case name	Post Comment
Actors	Authenticated User
Description	Users registered in the system may post their comments on recent news of the park
Preconditions	The user must be authenticated in the system
Post-Conditions	The comment is added to the news article and shown in the comments list for that article
Normal execution flow	1. The user executes the steps in the Normal execution flow of UC-5 2. Selects a news article from the list 3. Selects the option to add a comment 4. The system shows two input areas: one for the title and one for the text of the comment. The user inserts the desired values 5. The user confirms the submission 6. The system shows the list of comments for the commented news article
Alternative flows	The user
Exceptions	The user inputs an empty comment, either on the title area or in the text → The system shows a warning message if the user attempts to submit such a comment, aborting the submission. No information is persisted in the system.
Priority	Low
Usage frequency	Low to medium. Most users are not expected to comment news articles
Notes	None

Table B.7: Post Comment Use Case specification

Client Application Requirements

Identifier	UC-8
Use Case name	Add Occurrence
Actors	Botanist
Description	This feature allows Botanists to insert new botanical occurrences in the system
Preconditions	The user must be authenticated in the system as a Botanist
Post-Conditions	The new Occurrence is saved in the system and in the remote repository. Uploaded images are also saved correctly in the remote repository
Normal execution flow	1. Accessing the occurrence insertion area 2. Selecting a KML file for upload 3. The system will show the spatial entities contained in the KML file (polygons and points) 4. The user selects one of these entities to assign to the new occurrence 5. If the user selected a point 6. Inputting a taxonomic classification tree for the occurrence 7. Selecting and uploading images to associate to the occurrence 8. Confirm occurrence submission 9. The occurrence is saved successfully in the occurrence repository 10. Local information on the occurrence is saved locally, including the global identifier returned by the remote repository.
Alternative flows	1. If, during the normal execution flow, the user selects a polygon instead of a point, the system must allow the association of several taxonomic classifications instead of only one.
Exceptions	The user inserted an invalid value at any of the taxonomic levels → the system shows a warning message and the occurrence is not created
Priority	High
Usage frequency	Medium to High. This feature is expected to be used by all Botanists
Notes	None

Table B.8: Add Occurrence Use Case specification

Client Application Requirements

Identifier	UC-9
Use Case name	Create Tour
Actors	Botanist
Description	The park's Botanists may create tours of the occurrences present in the park. These tours reference specimens identified by the park, in an ordered fashion.
Preconditions	The park must have recorded at least two occurrences
Post-Conditions	The tour is successfully created and registered in the park's system
Normal execution flow	1. Running a specimen search (normal execution flow of UC-1) 2. The system shows a list of the resulting occurrences 3. The user can order that list to specify the visiting order of the occurrences for the tour 4. A map visualisation is shown, depicting the occurrences and the path along them (representing the tour path) 5. The user selects an option to submit the tour 6. The system shows a confirmation message and the user confirms the submission
Alternative flows	The user chooses to cancel the insertion procedure in the last step → The process is cancelled; No changes are done in the system; The system returns to the occurrence ordering area
Exceptions	No specimens are returned by the search operation → The system shows a warning message and returns to the step of occurrence search
Priority	High
Usage frequency	This Use Case is expected to be rarely used
Notes	None

Table B.9: Create Tour Use Case specification

Client Application Requirements

Identifier	UC-10
Use Case name	Add Intervention
Actors	Gardener
Description	This feature allows Gardeners to register maintenance operations done on park specimens
Preconditions	The user must be authenticated in the system as a Gardener. The system must have at least one recorded occurrence.
Post-Conditions	The occurrence is successfully recorded in the system
Normal execution flow	1. The gardener searches for occurrences, following the Normal execution flow of UC-1 2. A list of resulting occurrences is shown 3. The gardener selects one of these occurrences 4. The system shows a list of the available types of interventions. These are: "Spray", "Fertilize", "Irrigate", "Harvest", "Prune", "Trim", "Paint" and "Other". 5. Selecting one of the options mentioned in the previous step. If the selected option is "Other", the system must allow the gardener to insert a name for the operation 6. A subject -small description of the occurrence- must be inserted, as well as a brief description of the operation 7. The submit option is selected 8. The system shows a confirmation message. The user confirms the submission and the intervention is recorded.
Alternative flows	1. The user inserts an empty value for the operation after selecting "Other" as the operation type → the system shows a warning message and returns to the operation type selection status. 2. The user does not insert a value for the occurrence subject or the occurrence description → the system aborts the submission procedure, showing a warning message instead. 3. The user cancels the occurrence submission at the last step
Exceptions	No specimens are returned by the search operation → The system shows a warning message and returns to the step of occurrence search
Priority	Medium
Usage frequency	It is expected that this Use Case be used periodically. Some peaks of usage are expected at specific times of the year, for example during the times of pruning.
Notes	None

Table B.10: Add Intervention Use Case specification

Client Application Requirements

Identifier	UC-11
Use Case name	View Interventions
Actors	Gardener
Description	Gardeners may consult interventions done on any of the park's occurrences.
Preconditions	The user must be authenticated in the system as a Gardener. The system must have at least one recorded occurrence.
Post-conditions	The interventions for the selected specimen are listed, in de-crescent order, by the creation date
Normal execution flow	1. The gardener searches for occurrences, following the Normal execution flow of UC-1 2. The system shows a list of the results 3. The gardener selects one of the results 4. The system shows a list of the occurrences associated with that occurrence
Alternative flows	None
Exceptions	No specimens are returned by the search operation → The system shows a warning message and returns to the step of occurrence search
Priority	Medium
Usage frequency	The usage level of this Use Case is expected to be approximately the same as that of UC-11
Notes	None

Table B.11: View Interventions Use Case specification

Client Application Requirements

Identifier	UC-12
Use Case name	View Intervention Details
Actors	Gardeners
Description	After listing interventions for a botanical occurrence, gardeners can view a specific intervention in more detail.
Preconditions	The same as those of UC-11
Post-conditions	A detailed view of the selected occurrence is shown
Normal execution flow	1. Following the Normal execution flow of UC-11 2. Selecting one of the resulting interventions 3. The system will then show a report containing the date of creation of the occurrence, its type, subject and description (see UC-11)
Alternative flows	None
Exceptions	No interventions are recorded for the selected occurrence → the system shows an information message instead of the list, and returns to the occurrence selection area
Priority	Medium
Usage frequency	The expected usage frequency for this Use Case is expected to be higher than that of UC-11; for each time an user views the interventions associated to an occurrence, it is expected that more than one will be viewed in detail.
Notes	None

Table B.12: View Intervention Details Use Case specification

Client Application Requirements

Identifier	UC-13
Use Case name	Post News
Actors	Journalist
Description	The park's journalists write articles. These can be associated to the park's occurrences. For example, if a news article is about a flower bed that bloomed, the journalist will be able to reference that flower bed for readers to consult.
Preconditions	The user must be authenticated in the system as a Journalist
Post-conditions	The news article is successfully saved and is shown in the home page (see UC-5)
Normal execution flow	1. The journalist accesses the news insertion page 2. Inserts the title of the news article and the content 3. If desired, the journalist may also select one or more images (local files) to associate to the article 4. The journalist may select a time period in which the news article will be repeated. The available choices are: "None", "Weekly", "Monthly", "Yearly" or "Custom". If the journalist selects "Weekly", the news article will be re-posted automatically in 7 days. If "Monthly" was selected, it will be repeated in 30 days. If "Yearly" was selected, it will repeat in 365 days. If "Custom" was selected, the user must input a value for the number of days in which the news article will be repeated. The "None" option will turn off automatic re-posting for this article. 5. The user submits the article 6. A confirmation message is shown. If the journalist confirms the submission, the news article is saved, including the selected images (if there were any).
Alternative flows	Adding occurrence references to a news article: The journalist may want to add references to one or more occurrences recorded in the park's system. To do so, before the submission step of the Normal Execution Flow of this Use Case, the following steps must be taken: 1. The journalist searches for occurrences, following the Normal execution flow of UC-1 2. The results are listed 3. The journalist may select one or more to assign them to the news article
Exceptions	The same as UC-1
Priority	Low
Usage frequency	This Use Case is expected to be lightly used. A maximum of three news articles per day is to expected.
Notes	None

Table B.13: Post News Use Case specification

Appendix C

Client/Server Interaction

C.1 Record Insertion

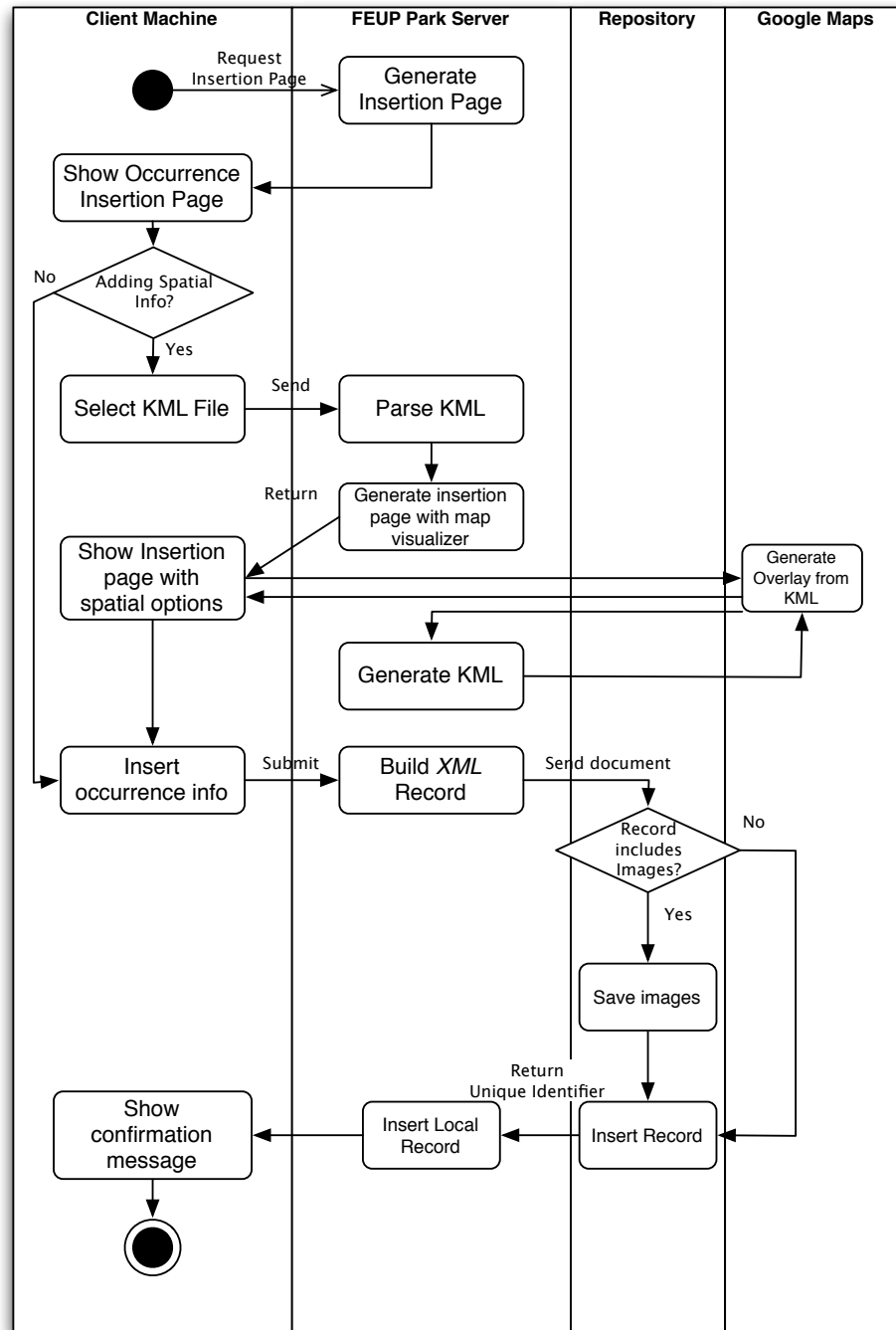


Figure C.1: Activity diagram showing the record insertion process

C.2 Record Searching

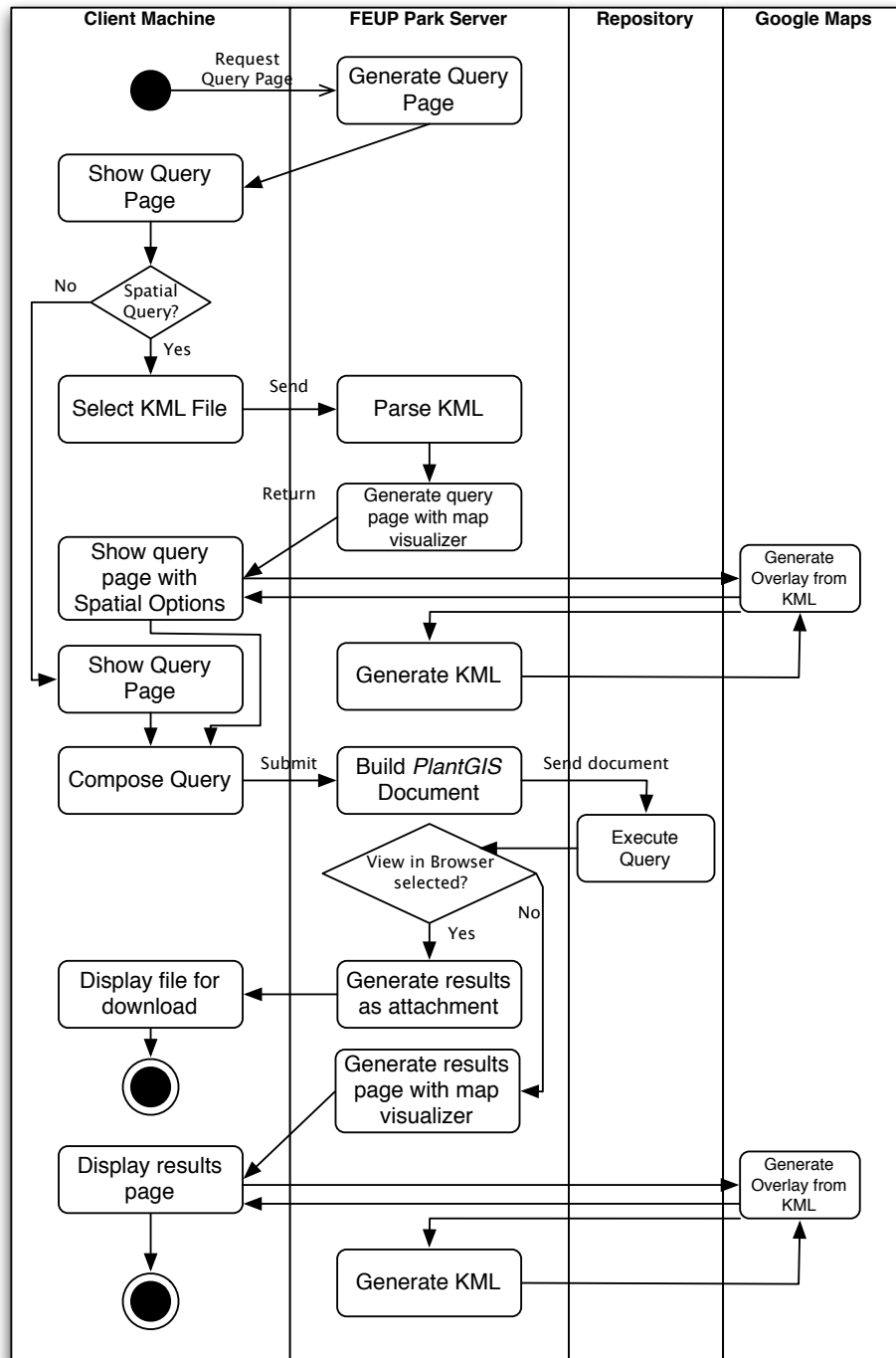


Figure C.2: Activity diagram showing the search process

Appendix D

PlantGIS Query Language

D.1 The Specification

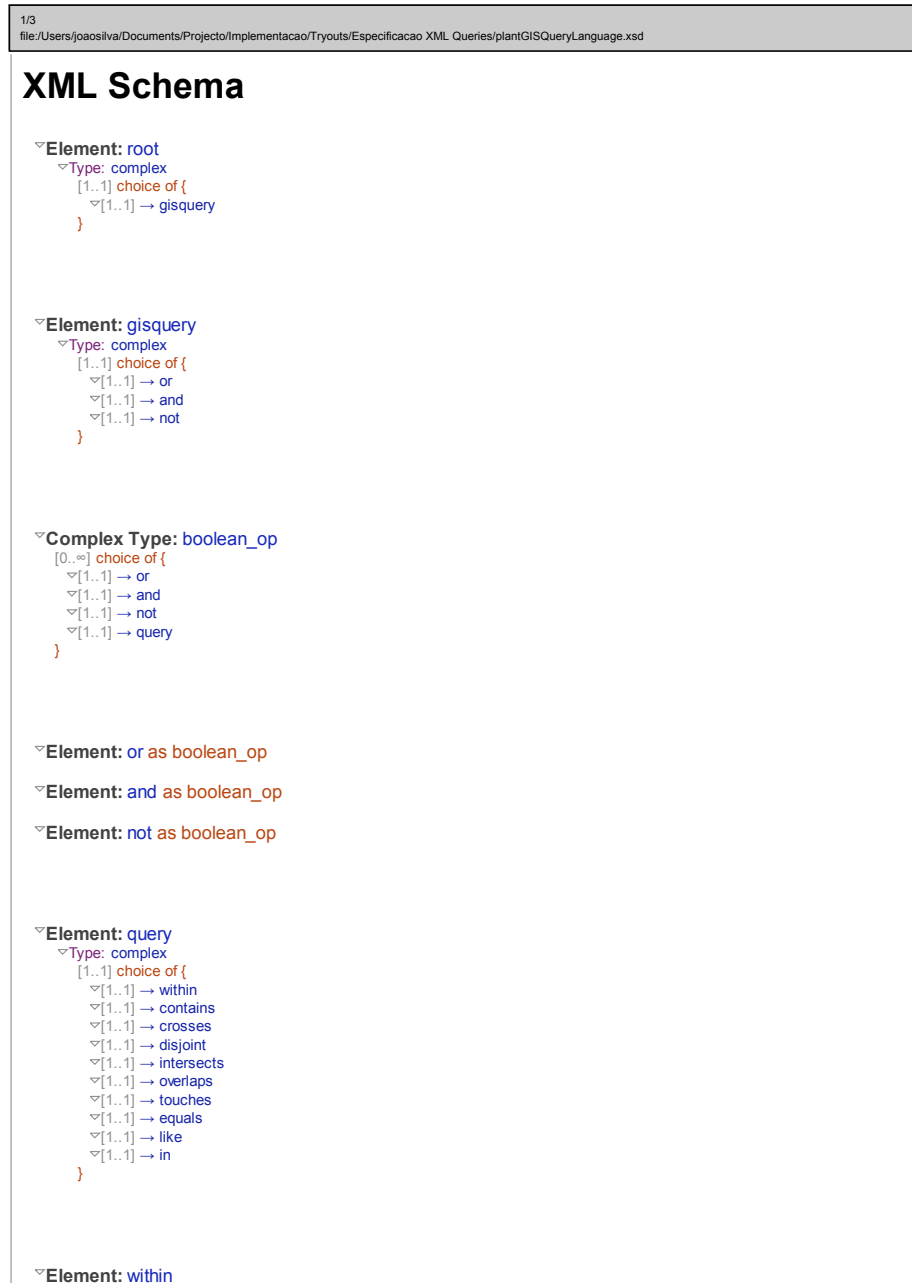


Figure D.1: The PlantGIS Query Language Specification - Part 1

PlantGIS Query Language

2/3 file:/Users/JoaoSilva/Documents/Projecto/Implementacao/Tryouts/Especificacao XML Queries/plantGISQueryLanguage.xsd	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: contains	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: crosses	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: disjoint	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: intersects	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: overlaps	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Element: touches	
▽Type: complex	[1..1] all of { ▽[1..1] → polygon }
▽Attribute: → polygon_field	
▽Complex Type: stringField	[1..1] all of { ▽[1..1] argument as xs:string }
▽Attribute: string_field as string_field	
▽Complex Type: valueField	[1..1] sequence of { ▽[1..∞] set_element as xs:string }
▽Attribute: string_field as string_field	
▽Element: equals as stringField	

Figure D.2: The PlantGIS Query Language Specification - Part 2

PlantGIS Query Language

3/3 file:/Users/joaoasilva/Documents/Projecto/Implementacao/Tryouts/Especificacao XML Queries/plantGISQueryLanguage.xsd	
▽Element: like as stringField	
▽Element: in as valueField	
▽Element: point	
▽Type: complex	
[1..1] sequence of {	
▽[1..1] latitude as xs:double	
▽[1..1] longitude as xs:double	
▽[1..1] altitude as xs:double	
}	
▽Element: polygon	
▽Type: complex	
[3..∞] sequence of {	
▽[1..1] → point	
}	
▽Attribute: label as xs:string	
▽Attribute: color as xs:hexBinary	
▽Attribute: transparency as xs:double	
▽Simple type: string_field	
Restricts xs:string	
enumeration dwc:Kingdom	
enumeration dwc:Phylum	
enumeration dwc:Classis	
enumeration dwc:ordo	
enumeration dwc:Familia	
enumeration dwc:Genus	
enumeration dwc:Species	
enumeration dwc:Variety	
enumeration dwc:GlobalUniqueIdentifier	
▽Attribute: polygon_field	
▽Type: simple	
Restricts xs:string	
enumeration location	

Figure D.3: The PlantGIS Query Language Specification - Part 3

D.2 Example Queries

```

1/1
file:/Users/joaosilva/Desktop/PlantGIS%20Samples/Sample1.xml
<?xml version="1.0" encoding="UTF-8">
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://paginas.fe.up.pt/~ei04032/plantGISQueryLanguage.xsd">
  <gisquery>
    <and>
      <not>
        <query>
          <within>
            <polygon label="Yellow">
              <point>
                <latitude>41.94312930232344</latitude>
                <longitude>-4.506891176354285</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.1859297908352</latitude>
                <longitude>-4.035639347114066</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.57650228526596</latitude>
                <longitude>1.641924906051866</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.57650228526596</latitude>
                <longitude>1.641924906051866</longitude>
                <altitude>0</altitude>
              </point>
            </polygon>
          </within>
        </query>
      </not>
    </and>
  </gisquery>
</root>

```

Figure D.4: PlantGIS Basic Query

PlantGIS Query Language

```
1/1
file:/Users/joaosilva/Desktop/PlantGIS%20Samples/Sample2.xml

<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://paginas.fe.up.pt/~ei04032/plantGISQueryLanguage.xsd">
  <gisquery>
    <and>
      <query>
        <within>
          <polygon label="B">
            <point>
              <latitude>45.96114032884673</latitude>
              <longitude>-2.443541536486871</longitude>
              <altitude>0</altitude>
            </point>
            <point>
              <latitude>35.02745884131014</latitude>
              <longitude>-1.373860843005441</longitude>
              <altitude>0</altitude>
            </point>
            <point>
              <latitude>35.72982031959973</latitude>
              <longitude>11.24363562716208</longitude>
              <altitude>0</altitude>
            </point>
            <point>
              <latitude>47.95291810171177</latitude>
              <longitude>9.954098389473451</longitude>
              <altitude>0</altitude>
            </point>
            <point>
              <latitude>45.96114032884673</latitude>
              <longitude>-2.443541536486871</longitude>
              <altitude>0</altitude>
            </point>
          </polygon>
        </within>
      </query>
      <not>
        <query>
          <within>
            <polygon label="C">
              <point>
                <latitude>41.94312930232344</latitude>
                <longitude>-4.506891176354285</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.1859297908352</latitude>
                <longitude>-4.035639347114066</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.57650228526596</latitude>
                <longitude>1.641924906051866</longitude>
                <altitude>0</altitude>
              </point>
              <point>
                <latitude>38.57650228526596</latitude>
                <longitude>1.641924906051866</longitude>
                <altitude>0</altitude>
              </point>
            </polygon>
          </within>
        </query>
      </not>
    </and>
  </gisquery>
</root>
```

Figure D.5: PlantGIS Intermediate Query

PlantGIS Query Language

```
1/2
file:/Users/joaosilva/Desktop/PlantGIS%20Samples/Sample3.xml

<?xml version="1.0" encoding="UTF-8"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://paginas.fe.up.pt/~ei04032/plantGISQueryLanguage.xsd">
  <gisquery>
    <and>
      <query>
        <equals string_field="dwc:InstitutionCode">
          <argument>FEUP Park</argument>
        </equals>
      </query>
      <or>
        <query>
          <intersects>
            <polygon>
              <point>
                <latitude>-8.5621298811</latitude>
                <longitude>41.177812756</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>-8.5628771276</latitude>
                <longitude>41.178987821</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>-8.5612334898</latitude>
                <longitude>41.177412901</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>-8.5612334898</latitude>
                <longitude>41.177412901</longitude>
                <altitude>0.0</altitude>
              </point>
            </polygon>
          </intersects>
        </query>
        <query>
          <within>
            <polygon>
              <point>
                <latitude>9.121219890</latitude>
                <longitude>20.98127611</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>9.121219890</latitude>
                <longitude>20.98127617</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>9.121219890</latitude>
                <longitude>20.98127613</longitude>
                <altitude>0.0</altitude>
              </point>
              <point>
                <latitude>9.121219890</latitude>
                <longitude>20.98127611</longitude>
                <altitude>0.0</altitude>
              </point>
            </polygon>
          </within>
        </query>
        <query>
          <within>
            <polygon>
              <point>
                <latitude>9.121219890</latitude>
                <longitude>20.98127611</longitude>
                <altitude>0.0</altitude>
              </point>
            </polygon>
          </within>
        </query>
      </or>
    </and>
  </gisquery>
</root>
```

Figure D.6: PlantGIS Advanced Query - Part 1

PlantGIS Query Language

```
2/2
file:/Users/joaosilva/Desktop/PlantGIS%20Samples/Sample3.xml

<latitude>20.91287676</latitude>
<longitude>10.09278381</longitude>
<altitude>0.0</altitude>
</point>
<point>
  <latitude>20.90812786</latitude>
  <longitude>10.18763246</longitude>
  <altitude>0.0</altitude>
</point>
<point>
  <latitude>20.98186277</latitude>
  <longitude>10.43585786</longitude>
  <altitude>0.0</altitude>
</point>
<point>
  <latitude>20.91287676</latitude>
  <longitude>10.09278381</longitude>
  <altitude>0.0</altitude>
</point>
</polygon>
</within>
</query>
<query>
  <in string_field="dwc:GlobalUniqueIdentifier">
    <set_element>FEUP:Oleaceae:Oliveira1</set_element>
    <set_element>FEUP:Oleaceae:OliveiraAlta</set_element>
    <set_element>FEUP:Oleaceae:Oliveira78</set_element>
  </in>
</query>
<query>
  <equals string_field="dwc:Species">
    <argument>Olea Europaea</argument>
  </equals>
</query>
</or>
</and>
</gisquery>
</root>
```

Figure D.7: PlantGIS Advanced Query - Part 2

Appendix E

Project Logbook

Project Logbook

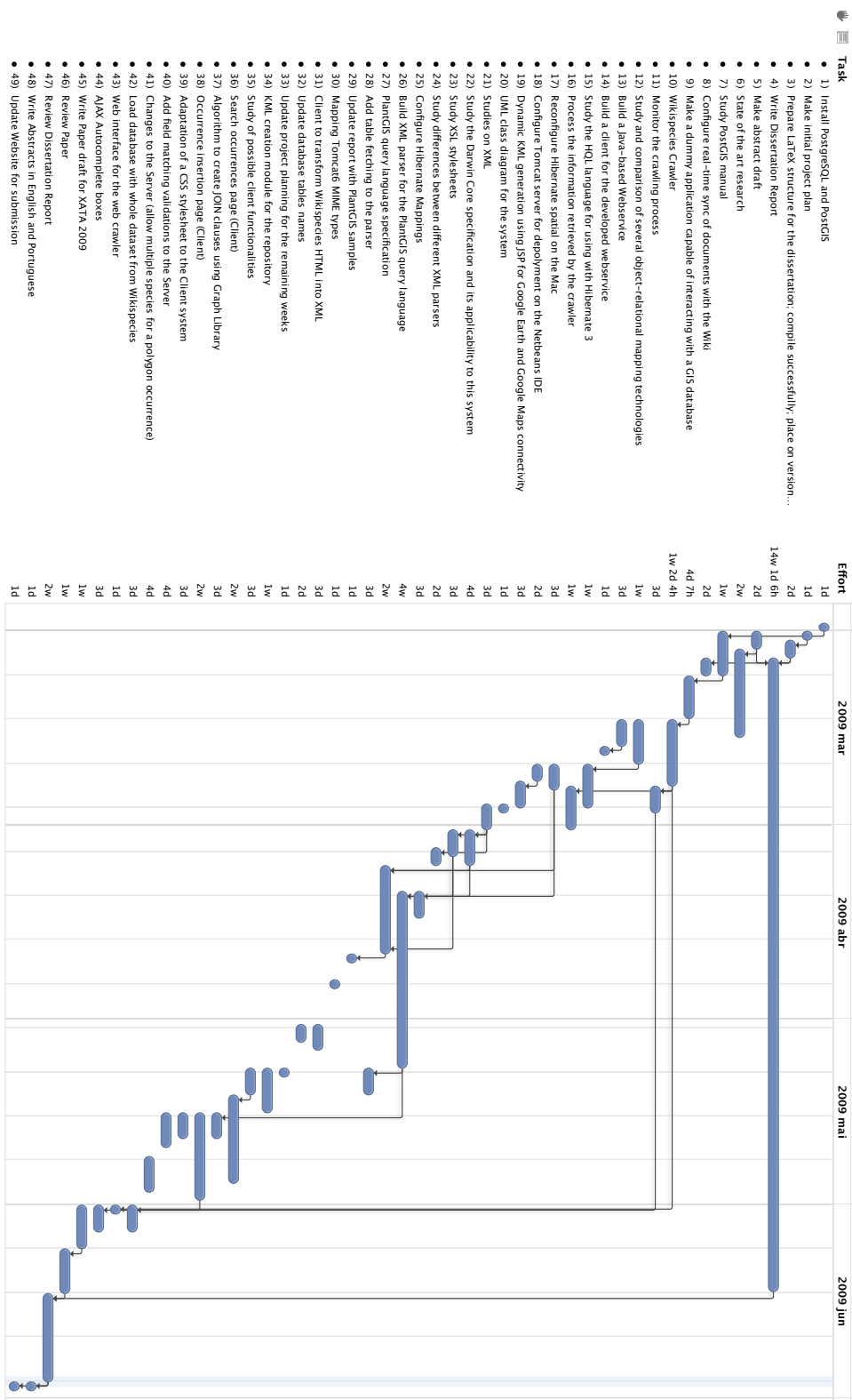


Figure E.1: Project Logbook