

MASTER PROJECT

GEOME, a web-based platform for an integrated analysis of spatial environmental and molecular data

Environmental Sciences and Engineering Program

Master student : Pauline Emery

Responsible professor : François Golay (LaSIG laboratory, EPFL)

Responsible research and teaching associate : Stéphane Joost (LaSIG laboratory, EPFL)

External expert : Nicolas Ray (enviroSPACE laboratory, University of Geneva)

EPFL, January 2012

LaSIG laboratory: Laboratory of Geographic Information Systems

enviroSPACE laboratory: Laboratory of Spatial Predictions and Analyses in Complex Environments



Abstract

GEOME is a web-based platform for the combined analysis of spatial environmental and molecular data. It implements tools for population genetics and spatial analysis. It has been developed to provide objective criteria for optimal biodiversity conservation measures. Although resource conservation managers increasingly have to use geo-environmental information and spatial analysis approaches, they do not have the tools and are not trained to do so. The web component of GEOME allows centralized service and technical support, encouraging the emergence of a community of users. GEOME therefore supplies a unique service of decision support based on free environmental data and open-source technologies.

Résumé

La plate-forme web GEOME permet l'analyse intégrée de données spatiales environnementales et moléculaires. Elle fournit un outil de traitement de données qui effectue un modèle dit d'association, combinant des concepts d'analyse spatiale et de génétique des populations. Le but de GEOME est de fournir des critères objectifs pour des mesures visant à conserver la biodiversité de manière optimale. En effet, les responsables de la conservation des ressources naturelles rencontrent de plus en plus le besoin d'utiliser et d'analyser des géo-données en relation avec des données génétiques. Cependant ils n'ont pas les outils et ne sont pas formés pour réaliser cette tâche. La composante web de GEOME permet d'avoir un service et un support technique centralisé, favorisant l'émergence d'une communauté d'utilisateurs. GEOME apporte ainsi un service unique d'aide à la décision basé sur des technologies libres et des données environnementales en libre accès.

Contents

Abstract - Résumé	i
Lists of figures and tables	iv
1 Introduction	1
2 Business requirements	2
2.1 Users and their needs	2
2.2 Use case diagram	2
2.3 Paper prototyping	4
3 Technical requirements	7
3.1 GEOME core features	7
3.1.1 Genetic geo-referenced data	8
3.1.2 Environmental data	8
3.1.3 SAM calculation program	9
3.2 Model-View-Controller software architecture	12
3.2.1 Model: UML class diagram	13
3.2.2 View: Website pages	14
3.2.3 Controller: Website skeleton	14
4 Technological choices	15
4.1 Web application frameworks	15
4.2 Python programming language	16
4.3 Django specifications	17
5 Implementation	18
5.1 Django specification applied to GEOME	18
5.2 Specific features	19
5.2.1 Forms validation	19
5.2.2 Access verification	20
5.2.3 ScytheSAM files creation	20
5.3 Web-GIS environment	20
5.3.1 OpenLayers mapping	20
5.3.2 PostGIS spatial database	21
5.4 Tests with different datasets	21
5.5 Hardware configuration	21

6 Possible further developments	22
6.1 Adding new modules to GEOME	22
6.2 Enhancing environmental data features	23
6.3 Improving collaboration between users	23
6.4 Deploying Django to a production server	23
7 Conclusion	24
Acknowledgements	25
References	26
Appendices	29
A Inventory of web-based platforms in population genetics topics	29
B ScytheSAM parameter file structure	31
C User Manual : Tutorial	33
C.1 Homepage	33
C.2 User registration	34
C.3 Studies and datasets	34
C.4 Download environmental data or run calculation	37
D Technical Manual	41
D.1 Installations	41
D.2 Initial set-ups on Django and PostgreSQL	43
D.3 Running the application	47
D.4 Working with Django	47
D.5 Synopsis	50

List of Figures

1	Use case diagram for GEOME prototype	3
2	First paper prototyping: defining the actions of a single user	4
3	Second paper prototyping: reviewing the structure	5
4	Third paper prototyping: solving implementation problems	6
5	GEOME main interconnected modules	7
6	Visualisation of CRU grid represented by it cell centroids	8
7	Schema of the interaction between the user, the web-based interface and the calculation program	10
8	Model-View-Controller design pattern, Source ref. [20]	12
9	UML class diagram of GEOME	13
10	Website skeleton	14
11	Python programming language icone	16
12	Django specification, Source ref. [21]	17
13	Django overview, Source ref. [22]	18
14	Basic example of mapping features with OpenLayers	20
15	Possible new interconnected modules to GEOME, Source ref. [10]	22
16	Non-exhaustive list of web-based platforms dedicated to population genetics topics. Source ref.[10]	29
17	Tutorial: View GEOME homepage	33
18	Tutorial: Login GEOME	34
19	Tutorial: Overview of available studies	34
20	Tutorial: Add a new study	35
21	Tutorial: Example of an empty study	35
22	Tutorial: Add a dataset	36
23	Tutorial: Visualise of a dataset	37
24	Tutorial: Download environmental data	38
25	Tutorial: Fill the form to run the calculation	39
26	Creating <i>geome</i> geodatabase and <i>geome_admin</i> user on PostgreSQL	44
27	Synchronising Django and PostgreSQL geodatabase	45
28	Adding CRU data grid to <i>geome</i> geodatabase	45
29	Adding CRU data parameters to <i>geome</i> database	46
30	Adding PostGIS new function	46
31	Example of how to use PostGIS new function	46
32	Running the server on local base	47
33	Overview of Django's way to work, Source ref.[22]	47
34	Files and folders organisation of <i>geome_platform</i> folder	50
35	Files and folders organisation of <i>input_output_scythesam</i> folder	51

List of Tables

1	Example of sheep genetic data file for a dataset constituted of 4 individuals	8
2	Parameter details of CRU/Oxford/IWMI grids	9
3	Example of environmental values data file for 4 given coordinates	10
4	Example of a parameter file for calculation	10
5	Non-exhaustive list of web application frameworks	15
6	Libraries used by GeoDjango	17
7	Flow graph summary of GEOME	19
8	Datasets parameters and time results	21
9	Required packages for Geodjango	41

1 Introduction

In the present context of global climate change, ecologists and evolutionists show a renewed interest to study adaptation (Holderegger & al.2008 [4]). Local adaptation is an important issue in conservation genetics. Indeed, the level of local adaptation of species to their environment is for instance investigated with the intention to provide unambiguous criteria to characterize conservation areas which are the most worthwhile preserving (Bonin 2007 [9]), or to favour sustainable agriculture and husbandry based on adapted breeds (Joost 2009 [8]). There are two complementary approaches to measure local adaptation: a theoretical approach in population genomics (e.g. Foll and Gaggiotti 2008 [3]) and a spatial analysis approach (Joost & al. 2007 [9]). The theoretical approach identifies candidate loci (specific regions on a chromosome) under natural selection from genetic data, using differences in allele frequencies between populations. The spatial analysis approach, which comes within the scope of a research field named landscape genomics, combines molecular and environmental data (e.g. precipitation or temperature regional values) to identify candidate loci for selection.

Landscape genomics aims to understand how environmental features do structure genetic variation in living organisms (Manel 2003 [12]). It is at the interface of genome sciences, environmental resource analysis, bioinformatics and spatial statistics. Combining these sciences, it is possible to assess a level of association between specific genomic regions and environmental factors to identify loci responsible for adaptation to different habitats (Lowry 2010 [11]). For instance, certain breeds of sheep produce a wool that contains more lanolin (wool wax) in regions with higher rainfall due to the presence of a given marker (Barraud 2011 [1]). Henceforth, knowledge of geo-environmental data and skills in Geographic Information System (GIS) are necessary to develop research or management activities using a landscape genomics approach. However as resource conservation managers are not trained to use GIS together with appropriate geo-environmental information, the development of robust and easy-to-use computer infrastructures is required. This is why since 2010, LaSIG laboratory is developing the idea of a new web-based platform named GEOME (contraction of GEOgraphy and genOME) to carry out this task. Indeed, web-based platforms are ideal in this context as they supply technical support, centralised services and manage interactions with users. Several web-based platforms already exist in the field of genomics or population genetics providing analytical tools and computational resources (see Appendix A). However none currently implements the landscape genomics approach, facilitating access to geo-environmental data, and integrating spatial analysis and population genetics tools.

This Master project contributes to the development of the GEOME platform by edifying its first prototype. The main goal is to implement a webGIS platform able to :

- (a) permit the upload and the visualisation of geo-referenced molecular data;
- (b) facilitate the computation of association models between environmental data provided by the platform and molecular data provided by users;
- (c) facilitate the computation of spatial statistics (spatial autocorrelation);
- (d) create a collaborative platform to permit the sharing of datasets, while respecting user privacy constraints.

This report presents the different steps of the web-based platform prototype development (business and technical requirements, technological choices and implementation) and possible future improvements of the platform.

2 Business requirements

It is necessary to focus on the business requirements to properly outline the features to be implemented in GEOME. Business requirements are established by knowing: (i) who are the users and what are their needs, (ii) what should they be able and allowed to do on the platform, (iii) what are their expectations concerning the user-system interface.

2.1 Users and their needs

GEOME targets different scientific and institutional communities active in the study of living organisms and their relationship to the environment. Those communities are research laboratories in academics, specialized institutions of national states, international organizations (e.g. Food and Agriculture Organisation (FAO) Animal production and Health division), professional associations or private research organisations (e.g. World Wildlife Fund (WWF)).

GEOME users therefore will be landscape ecologists, resource conservation managers, evolutionary biologists or molecular ecologists from around the earth. Indeed GIS is increasingly integrated into evolutionary biology, using environmental variables to understand species spatial distributions (Kozak & al. 2008 [6]). In order to carry out their investigations, GEOME users need to benefit from the results of specific analytical tools (e.g. association models). They also need support to find appropriate geo-referenced environmental datasets to address their research problematic and to integrate their own datasets (e.g. presence or absence of genetic markers in geo-referenced individuals) to a model. Nevertheless, they are not trained to efficiently use GIS together with spatial analysis approaches and they do not have time to acquire those skills.

GEOME fulfils these needs by providing an easy-to-use webGIS-based interface with computational resources. Its web-based component is a benefit to its users as they can log into the platform from any browser, at any time, from anywhere in the world. They do not need to download and install software to view their account and to benefit from GEOME services. A backup of their data will regularly be done and the last version of the software will always be running on the platform. Moreover GEOME will create a community of landscape genomics users, which will exchange through the platform for instance ideas, data or scientific articles.

2.2 Use case diagram

The elaboration of a specific UML¹ use case diagram for GEOME has enumerated the actions of the users on the platform, answering the previous question "what should users be able and allowed to do?". A use case diagram is a description of steps or actions between a user (a person or an external software) and the system (Bittner 2003 [2]). GEOME use case is shown in Figure [1]. It covers user-system interactions and gives a global view of the platform specifications. It illustrates the interaction between four components: the user, the web server, the database and the calculation program.

¹UML: "Unified Modeling Language is a standardized language for describing and visualizing the different parts of software systems. It is used for software designing" Collins dictionary

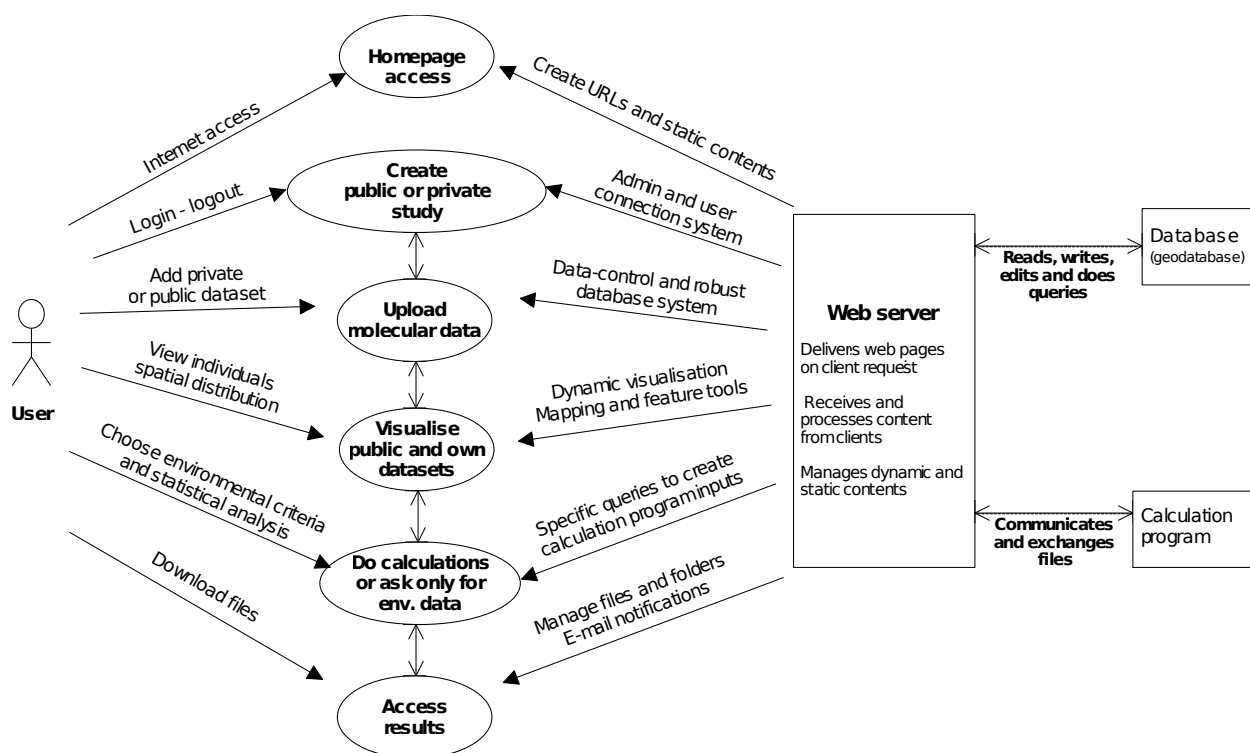


Figure 1: Use case diagram for GEOME prototype

On the platform, the user should be able to upload and access his own molecular data, download or use corresponding environmental data, run different types of calculations and download the calculation results. The user must define his datasets privacy status: private or public. If his dataset is private, he will be the only one to have access and edition rights to it. If his dataset is public, other users can view and use it. Calculation parameters as well as results will be accessible, therefore avoiding the user to repeat the same calculations and to wait for the results. Each dataset is linked to a species. For structure purpose, each dataset is therefore part of a study. Datasets related to the same species will be classified in the same study (see next page). As some users would like to remain totally anonymous on the platform, a study can also be private and shared with a restrictive circle of users. However, the aim of this platform is to be collaborative (giving access to environmental data but also to other molecular data from a given species of interest), so the user will be encouraged to create a private dataset in public study. This way, he could later on change his dataset privacy (e.g. when he will have carried out enough private research on the topic) in order to make it public.

The primary roles of the web server are to deliver webpages on user request, to receive and process content from users and to manage dynamic and static contents from webpages. From the web server point of view, the system must be able to handle administration and user connection system, to carry out a data-control, to communicate with a database system, to perform dynamic visualisation and mapping features, to fulfil communication with other programs and to manage files and folders creation and e-mail notifications.

2.3 Paper prototyping

The best way to understand the expectations concerning the user-system interface is to design several paper prototypes. Indeed, the paper prototyping technique supported GEOME creation phase by building a complete web-site design before writing any programming code. Visualising hand-sketched drawings of the user-system interface led to useful discussions and feedback from LaSIG supervisors and a concrete evolution over the weeks of the global platform design. For GEOME interface, this evolution was done in three major steps illustrated in Figure [2], [3] and [4].

The first step was to focus on a single user point of view enumerating the different actions done on the platform (e.g. connexion to the web-site, upload and visualisation of data, selection of spatial analysis methods and calculations). The results of this first step is shown in Figure [2].

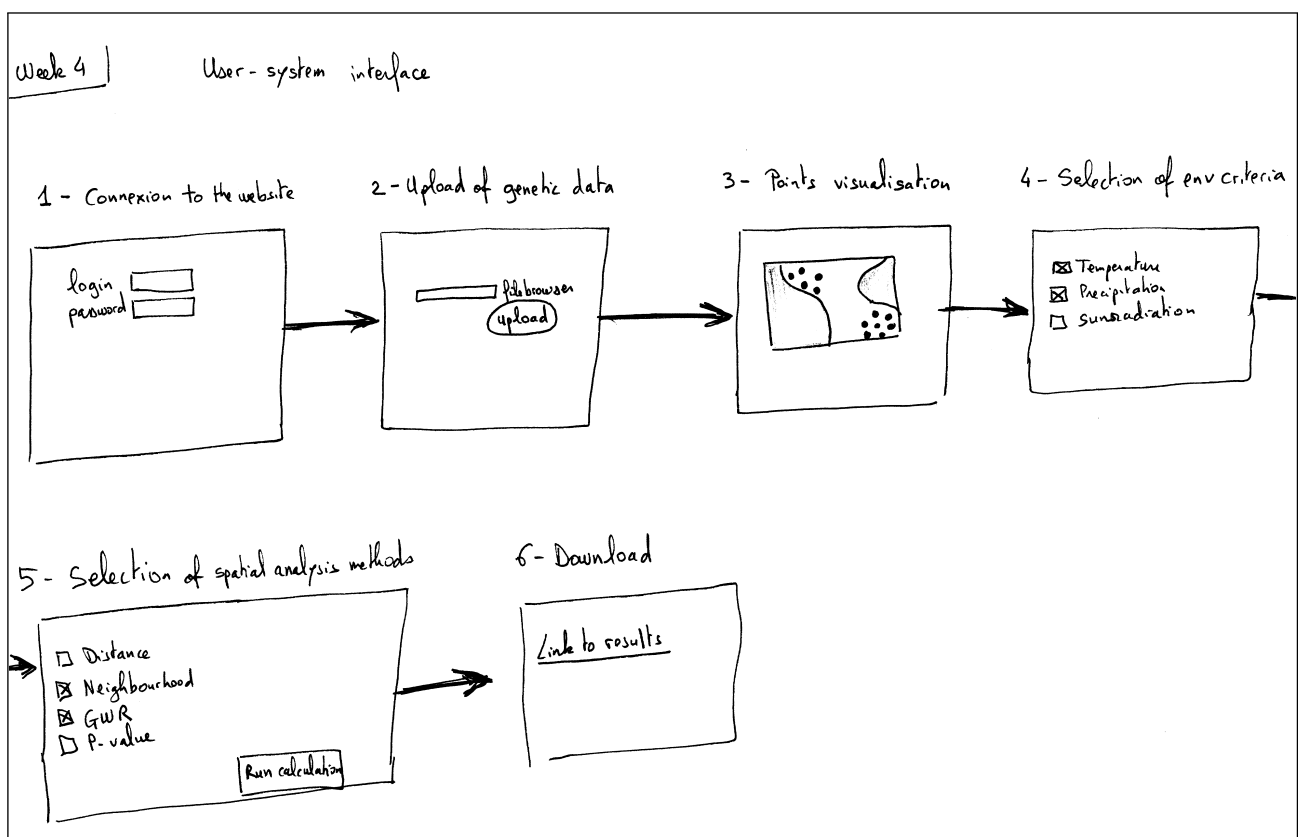


Figure 2: First paper prototyping: defining the actions of a single user

This first paper prototype was linear and outlined a problem of data classification. Indeed, the different datasets needed to be organised into "studies". Each study is defined by a unique species. This way each dataset belongs to a single study. This reflection led to the elaboration of a second hand-sketched drawing shown in Figure [3]. The user-system interface is composed of the previous different actions and of the new structure (e.g. study, dataset, calculation and results). The second step consequently was a structure reviewing.

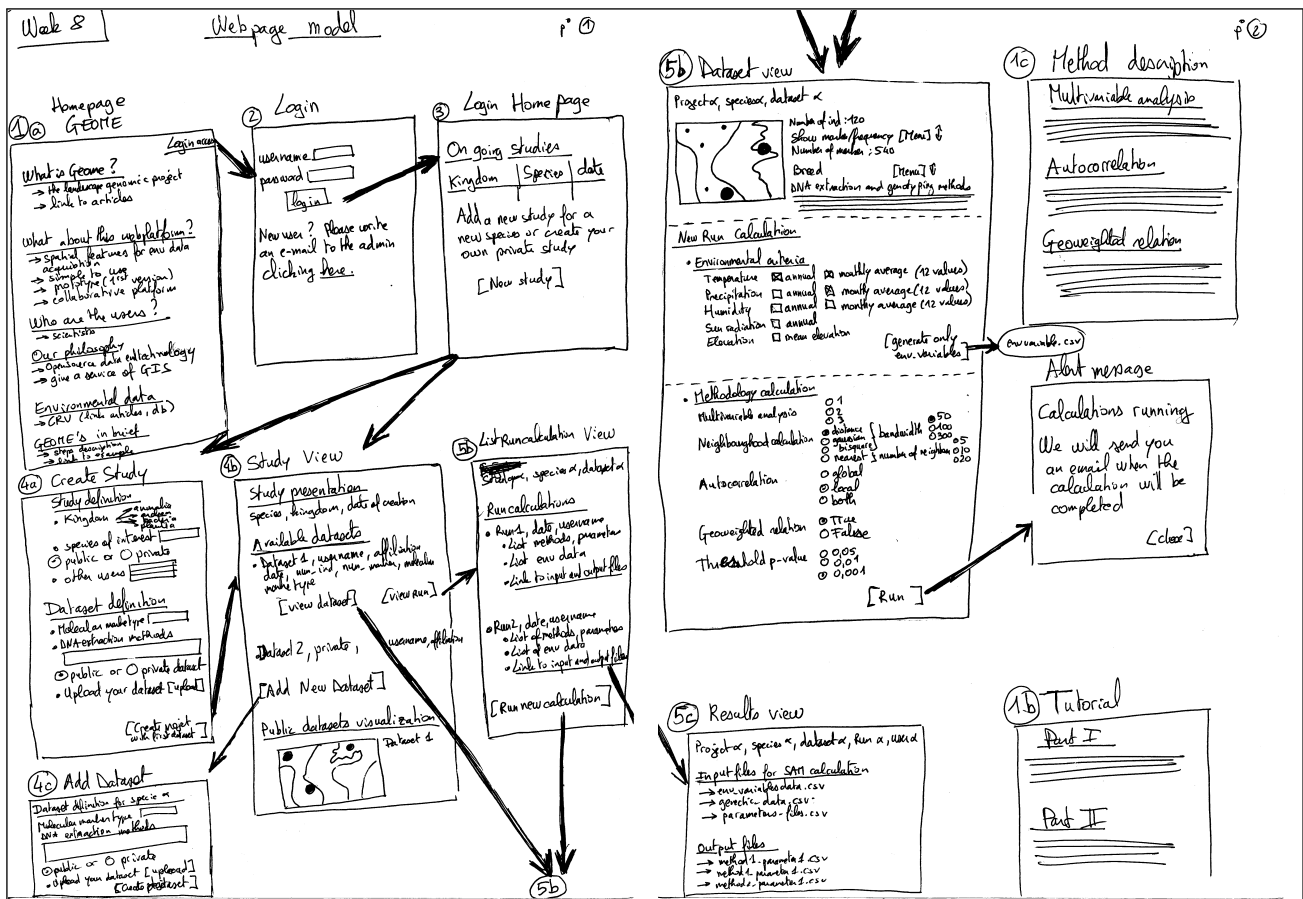


Figure 3: Second paper prototyping: reviewing the structure

This second draft resembles much more a web-site design and corresponds to the user wishes. However it had several implementation problems: how to add a new study and a new dataset simultaneously in the database (see page 4a in Figure [3]) ? How to generate only environmental data in the same form than the calculation form (see page 5a) ? The webpage contents were therefore discussed and reorganisation of the information was done by dividing the forms and the pages, leading to the final paper prototyping presented in Figure [4].

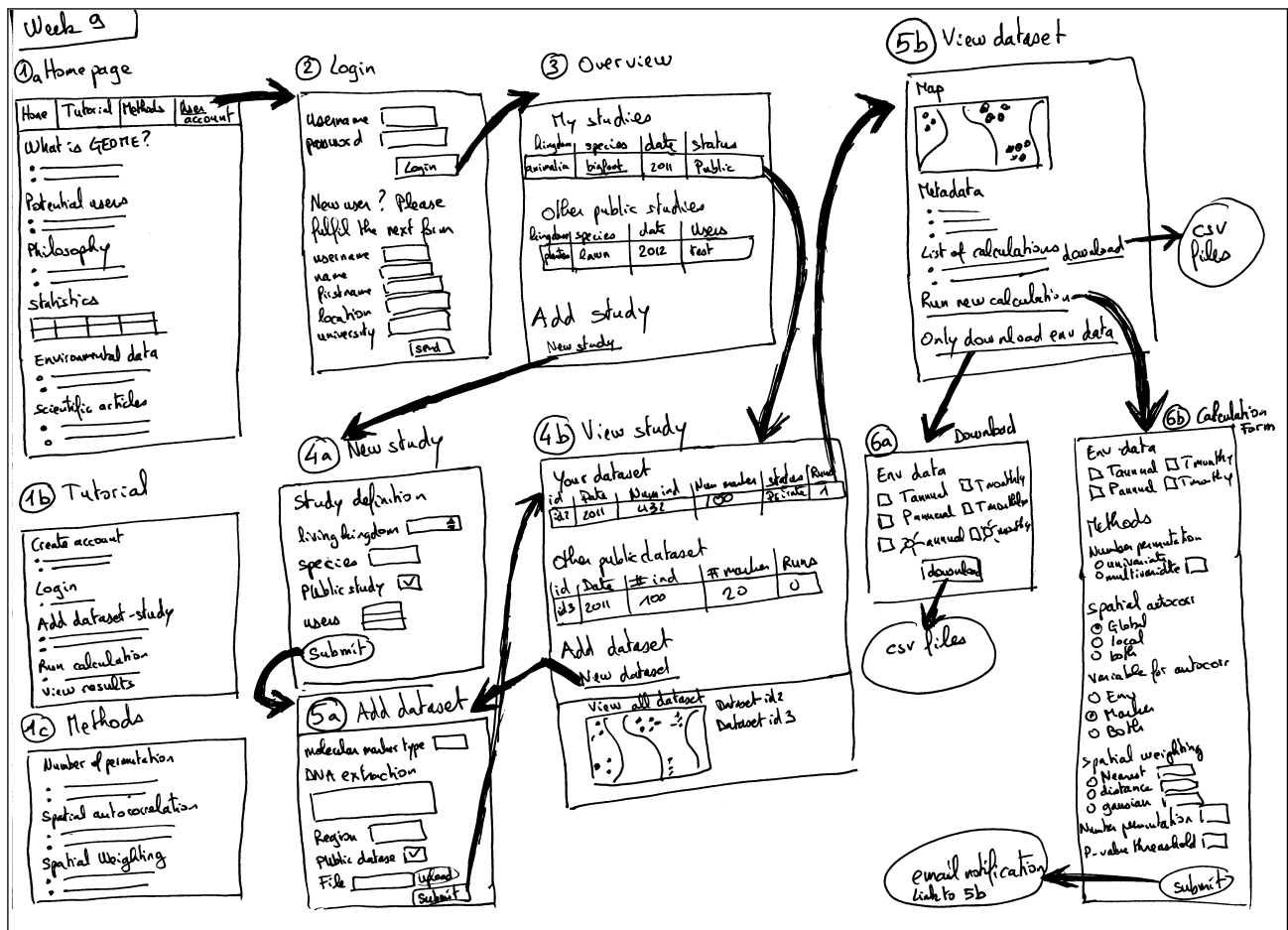


Figure 4: Third paper prototyping: solving implementation problems

Through this last step, each page has its own purpose (add, view or download) and is linked logically to other pages. In terms of webpage contents, a special thought was given to the platform homepage. Moreover, a series of statistics was planned to be displayed on the homepage (e.g. number of users, number of dataset, species of study etc...) in order to give more visibility of the content of the platform for non-registered users, to show its collaborative component and to encourage people to register to the platform.

3 Technical requirements

Technical requirements were considered to define which kind of technology to be used. They expose the core features and the design pattern to take into account to fulfil the business requirements.

3.1 GEOME core features

The implementation of the core features (modules A, B, C, D and E) in Figure [5] was decided for the web-based platform prototype. GEOME modules are divided in different categories of data and tools.

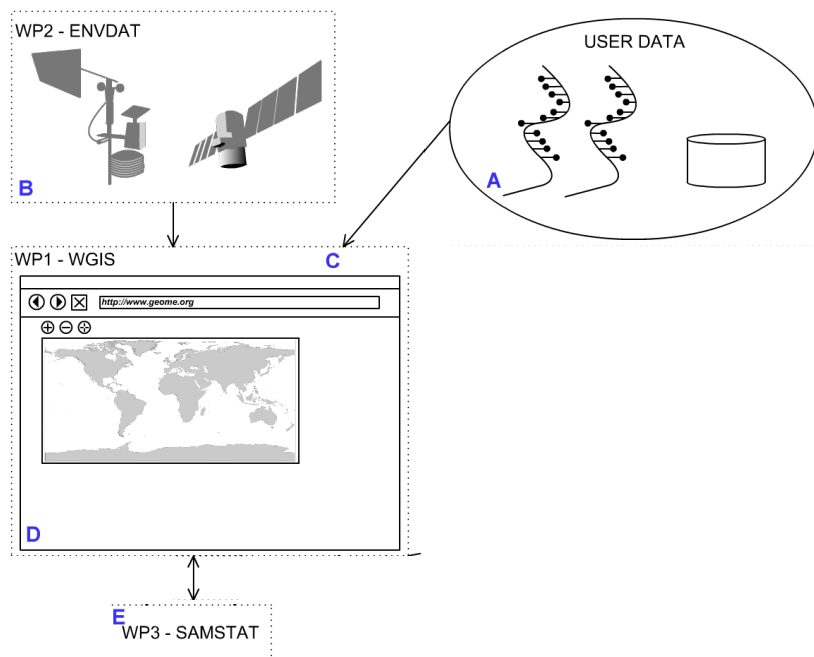


Figure 5: GEOME main interconnected modules

Data

- A. Genetic geo-referenced data with presence or absence of the genetic markers
- B. Environmental data

Tools

- C. Genetic data upload interface
- D. WebGIS environment for the integration of environmental and genetic data and their geo-visualisation
- E. SAM module for the detection of loci associated with environmental parameters (Joost & al. 2007 [9])

Modules A, B and E are going to be detailed in the next subsections as they are existing elements. Module C and D are described through-out this report as they are new features implemented during this Master project. These two modules allow the connexion and interaction between the existing modules.

3.1.1 Genetic geo-referenced data

Genetic data, provided by the users in text files, need to be characterised by geographic coordinates given in latitude/longitude in decimal degrees (Latitude: $[-90.00^\circ, 90.00^\circ]$, Longitude: $[-180.00^\circ, 180.00^\circ]$) with WGS84² geodetic datum in order to retrieve local environmental information. GPS devices or websites, like itouchmap see ref. [7], help to find the latitude/longitude of a given location. Molecular datasets used for SAM are matrixes with the presence (1) or the absence (0) of a given molecular marker type (e.g microsatellite markers or amplified fragment length polymorphism (AFLP)). "NaN" (Not-a-Number) text can be placed for a missing molecular value. Each row of the matrix corresponds to a sampled individual, while the columns are organised according to the sampled individual's geographic coordinates and contain binary information (1 or 0) relating to the status of the genetic marker. An optional field is the breed, usually used in livestockbreed. Table [1] is an example of a sheep genetic data file, located in Europe, for a dataset constituted of 4 individuals.

Individual	Latitude	Longitude	Breed (optional)	Marker_AA128	Marker_AG128	Marker_AG130
Aa1	46.59	7.86	Suffolk	0	1	0
Aa2	46.65	6.65	Suffolk	1	1	1
Ab1	52.31	-3.42	Lincoln Longwood	1	NaN	0
Ab2	51.70	-3.22	Lincoln Longwood	0	0	0

Table 1: Example of sheep genetic data file for a dataset constituted of 4 individuals

3.1.2 Environmental data

Worldwide datasets of 10 average climate parameters for global land areas were selected for the environmental data of the platform. This way GEOME provides environmental data for users around the world. These datasets are available through the Oxford School of Geography, the International Water Management Institute and the Climatic Research Unit (New 2002 [13]) under the Open Database License. All datasets have a grid resolution of 10-minute of angle latitude/longitude ($\approx 20 \times 20 \text{ km}^2$) of the world land areas as shown in Figure [6].

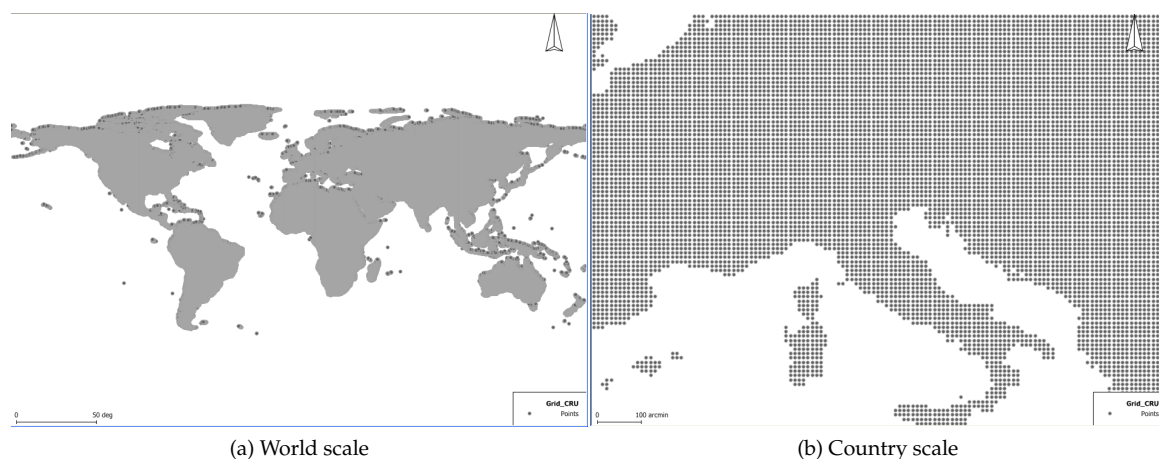


Figure 6: Visualisation of CRU grid represented by its cell centroids

²WGS84: World Geodetic System is a standard for use in cartography, geodesy, and navigation. WGS 84 is the reference coordinate system used by the Global Positioning System (GPS).

Their common grid is composed of 566'261 points representing the centroid of each cell. This way only points are stored instead of polygons. This concentrates all the information of a cell in a single element instead of four. It is also faster and easier to associate a point in space to a point of the grid, by searching the nearest point of the grid in a radius of 20 km.

The environmental datasets consist of 10 climate parameters, see Table [2]. Their values (given to each center of the grid) were interpolated from a dataset of station average values for the period centred on 1961 to 1990 (New 2002 [13]).

Parameter	Nomenclature	Units
Precipitation	pre	Millimetres per month (mm/month)
Cumul value of precipitation	cvpre	Percentage (%)
Wet-days frequency	rd0	Number of days with >0.1mm rain per month
Mean temperature	tmp	Degree Celcius (°C)
Mean diurnal temperature range	dtr	Degree Celcius (°C)
Relative humidity	reh	Percentage (%)
Sunshine duration	sunp	Percentage of maximum daylength (%)
Ground-frost frequency	frs	Number of days with ground-frost per month
10m windspeed	wnd	Meter per second (m/s)
Elevation	elv	Kilometers (km)

Table 2: Parameter details of CRU/Oxford/IWMI grids

All grids, except elevation, have 15 columns: latitude, longitude, 12 monthly values (from January to December) and an annual average value. Latitude and longitude columns correspond to the centre of each cell (10-minute of angle latitude/longitude) of the grid and are given in degrees decimal. The elevation grid has 3 columns: latitude, longitude and mean elevation.

Due to the variety of parameters and the worldwide resolution of this grid, it is possible to do association models between the genetic geo-referenced data and the environmental data of the sampling regions. This association model is compute by SAM calculation program.

3.1.3 SAM calculation program

Spatial coincidence allows to relate genetic characteristics (e.g. markers) of species with environmental parameters through joint coordinates. A simple calculation of correlation is not possible since it is a situation of presence or absence of markers. To work through this obstacle, statistics have developed the concept of logistic regression. The relationship of "all or nothing" is replaced by a sigmoidal shape function, so that it is possible to associate a probability of presence-absence of genetic markers according to environmental context.

SAM calculation program carries out multiple or univariate logistic regressions to test for association between allelic frequencies at marker loci and environmental variables. A confidence level of models consisting of all possible pairs (genetic marker x environmental parameter) is calculated to identify the most significant models that possibly play a role in the adaptation process.

GEOME must therefore ensure the communication with ScytheSAM, a SAM calculation program developed with Scythe Statistical Library in C++ open source programming language, in order to compute association models. GEOME creates ScytheSAM input files, lounge the calculations and gives back the results to the user, once the calculations are finished as shown in Figure [7].

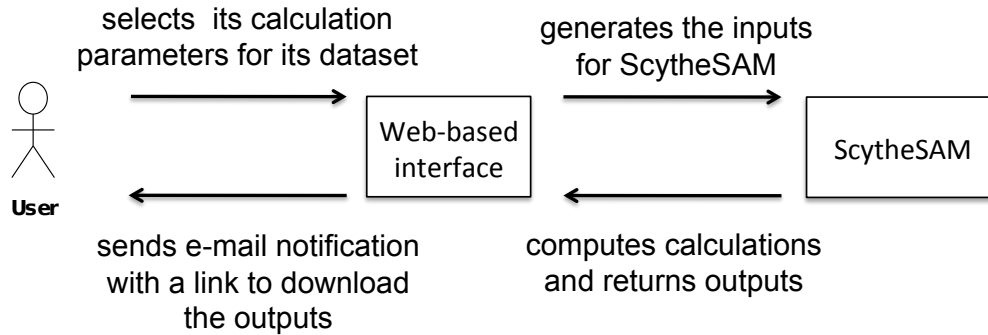


Figure 7: Schema of the interaction between the user, the web-based interface and the calculation program

ScytheSAM works with three input files: a genetic geo-referenced data file (e.g. Table [1]), its corresponding environmental data file (e.g. Table [3]) and a parameter file (e.g. Table [4]). The environmental data file is built thanks to a spatial query. This spatial query finds which local environmental parameters (e.g. from CRU grid) must be linked to the individual sampling coordinates, as those coordinates do not correspond to the coordinates of the grid centroid. The execution of this spatial query is the one of GEOME services.

Latitude	Longitude	Temperature (°C)	Precipitation (mm/month)	Elevation (km)
46.59	7.86	16	216	1.560
46.61	7.03	16	216	0.827
52.31	-3.42	13	540	0.236
51.70	-3.22	13	540	0.265

Table 3: Example of environmental values data file for 4 given coordinates

The file parameter structure composition is detailed in Appendix B. It is based on the possible execution of several logistic regressions methods, as spatial autocorrelation, spatial weighting, predictions, threshold or permutations.

HEADERS YES NUMVARENV 3 NUMMARK 3 NUMINDIV 5 DIMMAX 1 SPATIAL Longitude Latitude SPHERICAL NEAREST 10 AUTOCORR GLOBAL ENV SAVETYPE END BEST 0.01

Table 4: Example of a parameter file for calculation

The outputs of ScytheSAM depend on the calculation methods defined in the parameter file. The table containing the results can be composed of different groups of statistical data. Each group is constituted of n rows and m columns, where n and m represent respectively the number of environmental variables and the number of genetic markers. These groups contain the following information :

- 1 Log Likelihood2
- 2 Log Likelihood1
- 3 Degrees of freedom
- 4 G value
- 5 P value for G
- 6 Null hypothesis rejected for G (default confidence level = 99%)
- 7 Wald for Beta 0
- 8 Wald for Beta 1
- 9 P value for Wald Beta 0
- 10 P value for Wald Beta 1
- 11 Null hypothesis rejected for Wald Beta 0 (default confidence level = 99%)
- 12 Null hypothesis rejected for Wald Beta 1 (default confidence level = 99%)
- 13 Dynamic null hypothesis analysis for G and Wald Beta 1 : Null hypothesis for G
- 14 Dynamic null hypothesis analysis for G and Wald Beta 1 : Null hypothesis for Wald Beta 1
- 15 Dynamic null hypothesis analysis for G and Wald Beta 1 : Cumulated test

A complete procedure on how to deal with the ScytheSAM outputs and to identify the more significant associations is given on <http://www.econogene.eu/software/sam/> website.

The basis elements of the GEOME platform (inputs, tools, outputs) need to be integrate into the MVC software architecture pattern in order to create the web-based platform.

3.2 Model-View-Controller software architecture

In a web-based platform a Model-View-Controller (MVC) design pattern allows to clearly map the input, processing and output tasks of a web-based platform. It is an important phase of the GEOME platform design as it explains its technical behaviour and how each of its core features should interact.

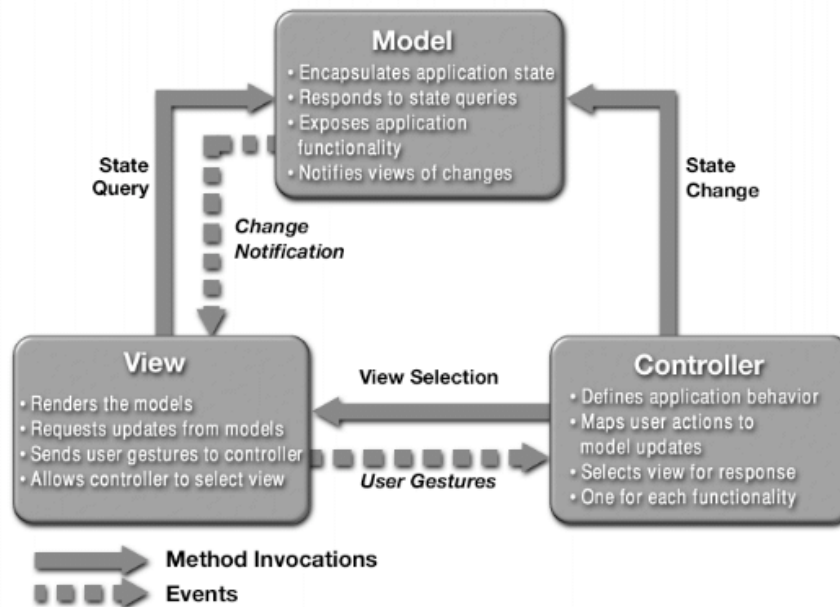


Figure 8: Model-View-Controller design pattern, Source ref. [20]

- M.** The model represents the application database and the rules that govern its access and update.
- V.** The view renders the contents of a model. It accesses data through the model and specifies how that data should be presented.
- C.** The controller translates interactions with the view into actions to be performed by the model. In a web application, user interactions appear as GET and POST HTTP requests. The actions performed by the model include activating processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.

Here following is the explanation of the MVC design pattern applied to GEOME platform prototype, with its UML class diagram (Model), its website design (View) and its website skeleton (Controller). The previous core features (module A, B, C, D and E) are used all around the MVC design pattern. In other word, the core features are the pieces of the platform and the MVC design pattern is the way to assemble them.

3.2.1 Model: UML class diagram

The best way to illustrate the Model of MVC design pattern is using a Unified Modeling Language (UML) class diagram. This schema is a static description of the classes and their attributes of GEOME database. It is a key element of the prototype design, because it must include as much as possible a general behaviour of the platform. Indeed, it is not recommended to alter the database structure (by adding connections or attributes) once it has been created, due to the need of data persistence. Figure [9] illustrates connections between different classes with their cardinality and their level of association (aggregation \diamond , composition \blacklozenge , specialisation \uparrow).

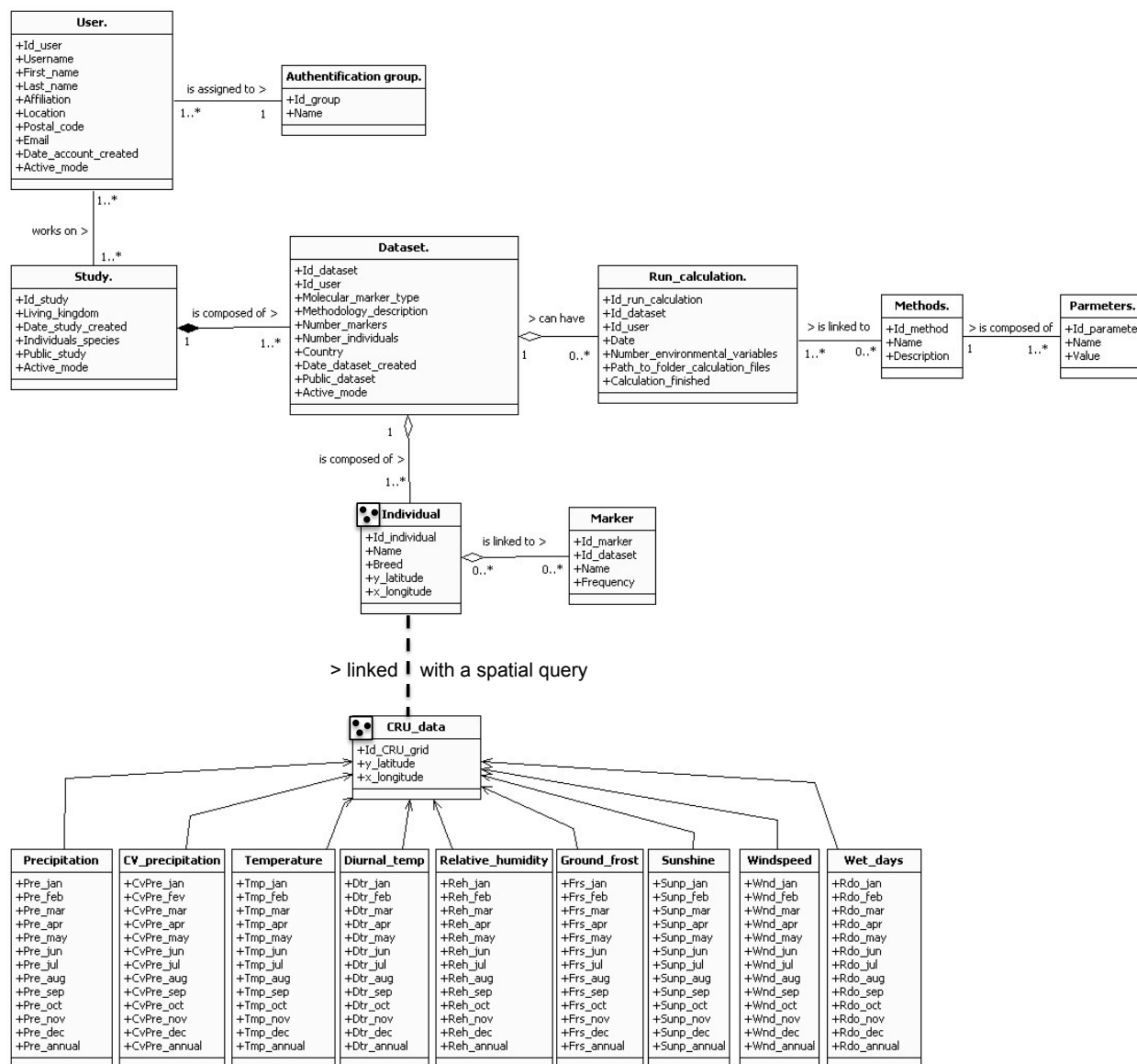


Figure 9: UML class diagram of GEOME

This UML class diagram is composed of 19 classes that can be divided in four categories: the user configuration (e.g. user and authentication group classes), the molecular data (e.g. study, dataset, individual and marker classes), the calculations (e.g. run calculation, methods and parameters classes) and the environmental data (e.g. CRU_data and the 9 climate parameters classes). The dataset class is the central node of the diagram.

The class diagram logic can be given as following:

- a user has personal information and belongs to an authentication group;
- a user can work on several studies;
- a study is defined by a species and is composed of datasets;
- a dataset is composed of individuals;
- a individual is related to markers and can be linked to the environmental data (CRU data) by a spatial query
- all climate parameter classes are specialisation of CRU_data class;

The attributes of the different classes give basic information about the class and should not be repeated from one class to the other if tables are correctly linked. The individual and the CRU_data classes have spatial data in it (latitude and longitude attributes). In order to do a spatial query, those tables need to be specified as point feature classes (with the three dots symbol). The climate parameter classes are not point feature classes but they are directly linked to the CRU_data by a specific identification code.

3.2.2 View: Website pages

The design and content of the website pages (the View) matches the last step of the paper prototyping technique of the business requirements, see Figure [4] on page 6. In fact it fulfils the technical requirements thanks to the second step of the paper prototyping technique in which attention was taken on how to integrate the information given by the user into the UML class diagram.

3.2.3 Controller: Website skeleton

A schematic way to conceive the Controller is to design the website skeleton of the platform as shown in Figure [10].

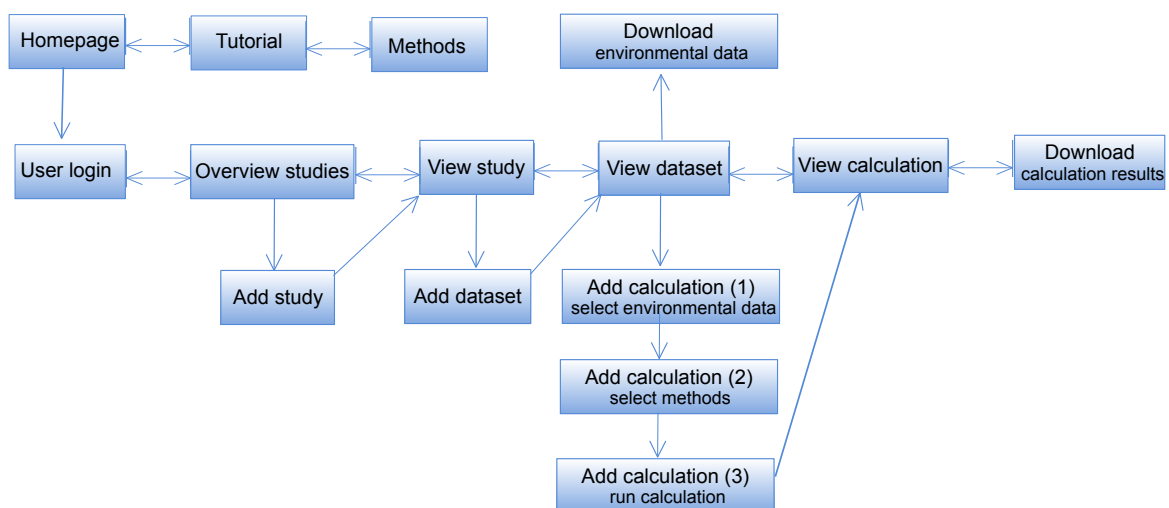


Figure 10: Website skeleton

The interactions between the Controller and the Model are implied with the "add", "view" and "download" actions. The relation between the Controller and the View are symbolised by the arrows between the different pages.

4 Technological choices

In order to develop a web-based platform from scratch in four months, a web application framework was needed to avoid having to re-implement basic features needed for each web application. Opting for the appropriate framework needed several steps. The first step was to find the standard frameworks that would satisfy the prototype technical requirements and LaSIG laboratory further use. The technological choices have been focussed on open-source technologies, as they are also able to fulfill GEOME requirements. The second step was to choose an appropriate programming language. The final step was to compare the existing web frameworks in that language with mapping possibilities. As GEOME is an evolving platform, the technical choices also underline briefly the future possible developments.

4.1 Web application frameworks

According to DocForge community project ref.[14], a web application framework is a set of codes, which provides functionality common to a whole class of applications, specifically designed to help developers to build web applications. A web application is a program, which interacts with users through a web browser over a network. Web frameworks typically provide core features common to web applications, such as user session management, data persistence and templating systems. Unlike a set of static pages, most web application pages are dynamically generated from persistent data. Frameworks can provide generic user accounts, so people can register, login and reset passwords. Frameworks can automatically check and require authentication before generating a page. They can also provide user management for administrators (create user accounts, manage permissions, change page content, generate reports or alter data). Frameworks also provide a consistent application programming interface (API) to access multiple data storage systems, to simplify storage and retrieval of data objects, such as with object-relational mapping (ORM). The most popular overall design pattern of web application frameworks is Model-View-Controller (MVC). The initial code of the framework evaluates the URL and passes responsibility to the appropriate application controller. The controller then performs any necessary actions with the application's model and then let the view build the actual response content. Webpage templates help keeping business logic separate from display logic. The templating system may be keyword replacement or generated content such as form fields from parameters. Most often the framework will make available developer-defined fields or data structures available within templates (i.e. push). Conversely a templating system may request data back through the framework (i.e. pull). There are many web application frameworks, as shown in Table [5]:

Language	Framework names
Java	Apache Struts / JavaServer Faces / Jt Design Pattern Framework / Apache Wicket
PHP	CakePHP / CodeIgniter / Symfony / Zend Framework
Python	Django / Grok / Pylons / Turbogears / web2py / Pylatte / repoze.bfg
Ruby	Ruby on Rails / Ramaze

Table 5: Non-exhaustive list of web application frameworks

4.2 Python programming language

Python programming language has been chosen, due to previous LaSIG programming backgrounds and because it fulfils the open source criteria for the platform. Herebelow, Python main characteristics are given according to Python Official website [19].



Figure 11: Python programming language icone

Python is a powerful dynamic programming language implemented under an open source license that is used in a wide variety of application domains. Python's core concept is keeping things understandable. It is available for all major operating systems: Windows, Linux/Unix, OS/2, Mac, Amiga, among others. The same source code will run unchanged across all implementations. Python also comes with complete documentation. The developer and user community maintains a wiki, hosts international conferences and contributes to online code repositories. Python is often compared to Perl, Ruby or Java. Its key distinguishing features include:

- clear and readable syntax (which is sensitive to indentation, so very few brackets needed);
- strong introspection capabilities;
- intuitive object orientation;
- full modularity, supporting hierarchical packages;
- exception-based error handling;
- high level dynamic data types;
- extensive standard libraries;
- extensions and modules easily written in C, C++ (or Java for Jython, or .NET languages for IronPython).

Among Python web-frameworks cited in Table [5], two are largely used in the field of web-mapping applications: Pylons with MapFish and Django with Geodjango. Pylons framework is in an evolving period (Pylons Project FAQ [15]), as Pylons 1.0 code-base has hit a point of diminishing returns in flexibility and extendibility. The Pylons web framework 1.x line will not be enhanced. The future of Pylon-style web application development is Pyramid. Its initial release was in December 2010. This recent outcome brings about possible documentation and features problems. Pyramids therefore is not as mature as Django (initial release was in July 2005). Django is renowned to be a rapid-to-learn web framework, which emphasises the principle of DRY (Don't Repeat Yourself). The Django documentation is understandable for both the experienced and inexperienced Python/Django user. The Django Book (Holovaty 2009 [5]) is available online for free. Django was therefore chosen for these global reasons to be used as the web-framework of GEOME prototype. However Django has also many specifications which have confirmed further on this choice.

4.3 Django specifications

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It provides high-level abstractions of common web development patterns, shortcuts for frequent programming tasks, and clear conventions for how to solve problems. It can be associated with others programs to establish web-mapping applications, as shown in Figure [12]. Django is improved by GeoDjango skills when it is needed as a world-class geographic web framework. Geodjango makes possible to build webGIS applications and harness the power of spatially enabled data.



Figure 12: Django specification, Source ref. [21]

As shown in Figure [12], GeoDjango uses different datastore (e.g. PostgreSQL and PostGIS) and libraries (described in Table [6]). The mapping features are given in the user interface with OpenLayers and jQuery. OpenLayers is a pure JavaScript library for displaying map data in web browsers, with no server-side dependencies. jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HyperText Markup Language (HTML). GeoDjango manages many serialised data (e.g. Keyhole Markup Language (KML), Geography Markup Language (GML)). Geodjango composition give therefore to GEOME many possibilities for further developments (e.g. adding layers and mapping interactions).

Library	Name	Use
Open Source Geospatial Consortium	OSGeo	Is a simple feature implementation for SQL specification, which handles geometry objects (e.g. PointField, MultiPointField etc.).
Geospatial Data Abstraction Library	GDAL	Reads and writes to a variety of vector file formats, (e.g. shape files).
Cartographic Projections Library	PROJ.4	Manages spatial reference systems and projections (e.g. the mercator projection commonly used by Google Maps, MapQuest) and ensures that geometry objects stored in PostGIS use the appropriate projection.
Mapnik	Mapnik	Builds customised maps and handles rasterizing vector objects when large datasets are encountered.

Table 6: Libraries used by GeoDjango

After choosing the technological tools, the next step was to implement the platform prototype.

5 Implementation

GEOME implementation is the realisation of the technical requirements through programming and deployment, with the tools selected during the technological choices. The purpose of this section is to understand the general behaviour of Django. The user manual (Appendix C) gives a general view of the platform by guiding the user through each webpage of GEOME. The technical manual (Appendix D) and Django's online documentation ref.[5] give more technical and programming details about GEOME implementation. Specific features implemented and the webGIS environment are also presented in order to underline reflections that came across during the implementation. GEOME hardware configuration and dataset tests are lastly exposed to comprehend its performance.

5.1 Django specification applied to GEOME

Django web framework is designed to encourage loose coupling and strict separation between the pieces of an application (e.g. data access logic, business logic and presentation logic) . This way it is possible to make changes to one particular piece without affecting the others. The overall design of a database-driven Django web application is given in Figure [13]: it combines MCV design pattern, configuration settings (settings.py, manage.py and django-admin.py) and interactions with the browser, the web server and the database.

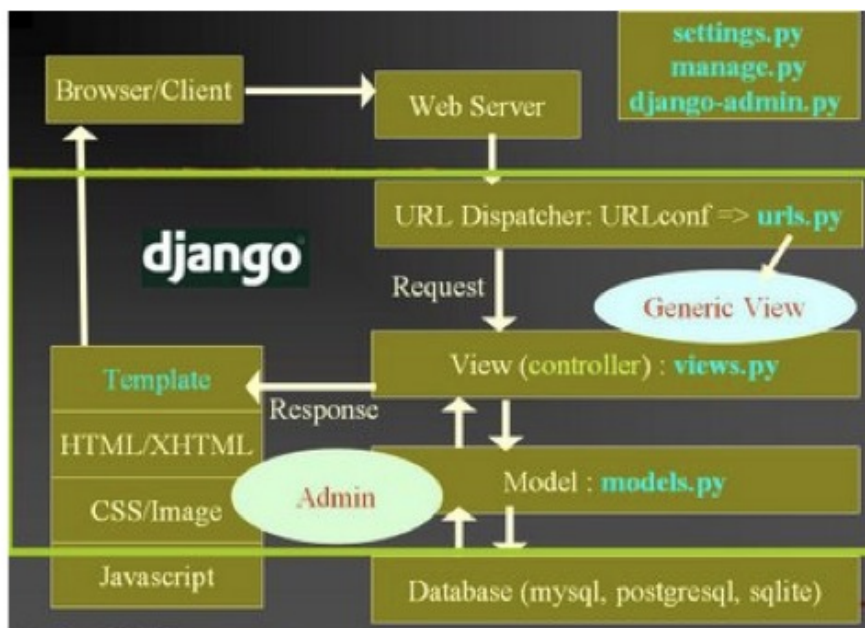


Figure 13: Django overview, Source ref. [22]

Django applies the MVC software architecture in the following way:

M is the data-access portion, is handled by Django's database layer;

V is the portion that selects which data to display and how to display it, is handled by views and templates;

C is the portion that delegates to a view depending on user input, is handled by the framework itself by following your URLconf and calling the appropriate Python function for the given URL.

Because the Controller is handled by the framework itself, Django is known to have an MTV (Model-Template-View) design pattern:

M stands for "Model," the data access layer. This layer contains the information about the data: how to access it, how to validate it, which behaviors it has and the relationships between the data.

T stands for "Template," the presentation layer. This layer contains presentation-related decisions: how something should be displayed on a webpage or other type of document.

V stands for "View," the business logic layer. This layer contains the logic that accesses the model and defers to the appropriate template(s). It is the bridge between models and templates.

To illustrate this MTV design pattern, the Table [7] gives the relation between the webpage from the paper prototyping (see Figure [4] on page 6), its corresponding URL (in `urls.py`), the view function executed (in `views.py`), the objects and conditions of the model (in `models.py`) and the html file returned in response by the view function.

Webpage (paper prototyping)	URL (<code>http://root.../</code>)	View (function)	Model (object condition)	Template (HTML file)
1a Homepage	<code>/homepage</code>	<code>staticview()</code>	Dataset all	<code>1a_homepage.html</code>
1b Tutorial	<code>/tutorial</code>	<code>staticview()</code>	-	<code>1b_tutorial.html</code>
1c Methods	<code>/methods</code>	<code>staticview()</code>	-	<code>1c_methods.html</code>
2a Login	<code>/accounts/login</code>	<code>login_required()</code>	-	<code>default_login.html</code>
2b Logout	<code>/account/logout</code>	<code>logout_view()</code>	-	<code>1a_homepage.html</code>
3 Overview	<code>/overview</code>	<code>on_going_studies()</code>	Study active, user	<code>3_overview.html</code>
4a New study	<code>/study</code>	<code>create_study()</code>	Study new object	<code>4a_new_study.html</code>
4b View study	<code>/study/id1</code>	<code>view_study()</code>	Study id1, public (or private but user is owner)	<code>4b_view_study.html</code>
4c New dataset	<code>/id1/dataset</code>	<code>add_dataset()</code>	Dataset new object	<code>4c_add_data.html</code>
5a View dataset	<code>/id1/dataset/id2</code>	<code>view_dataset()</code>	Dataset id2, public (or private but user is owner)	<code>5b_view_dataset.html</code>
5b Download data	<code>/id2/envdata</code>	<code>download_env_data()</code>	Dataset id2	<code>5b_env_data.html</code>
5c Run calculation	<code>/id2/calc/id3</code>	<code>add_calculation()</code>	Calculation id3	<code>5c_calculations.html</code>
6 Download results	<code>/id3/results</code>	<code>view_results()</code>	Calculation id3	<code>6_results.html</code>

Table 7: Flow graph summary of GEOME

5.2 Specific features

Several features implemented in GEOME needed to be carefully examined to avoid conflicts with the business and technical requirements. Three specific features are explained here bellow.

5.2.1 Forms validation

GEOME needed handle user data input. The best way to integrate user inputs into a database is to create a form from the database. Django automatically verifies if the field types are respected. This feature could be used to create new studies or datasets. However the user input file (molecular data file) had to be handled by a specific function, called by the view function, which verifies the file structure and information before adding them into the database.

5.2.2 Access verification

As GEOME must assure privacy settings, a control is done before loading each webpage to be sure that the user is authorised to have access to it. Indeed, the urls are accessed by the study id and the dataset id. If for instance the dataset 1 of study 2 is private, we only want its owner to access the urls `http://lasigpc49.epfl.ch:8080/study/2/dataset/1/`. This test is done in the view functions and return an "Access denied" page if the user is not the owner when the dataset is private.

5.2.3 ScytheSAM files creation

In order to do model computation, ScytheSAM needs three different inputs (see 3.1.3 SAM calculation program on page 9). The aim was to duplicate at least at possible the information of the database into files. Indeed, as datasets start growing, it is important to preserve storage memory. This is why, Django creates the molecular inputs only once for all calculations as this file is not changed from a calculation to the other.

5.3 Web-GIS environment

As GEOME aims to be a web-GIS platform for molecular data visualisation and spatial query realisation, its implementation required the use of mapping features (i.e. OpenLayers) and spatial databases (i.e. PostGIS).

5.3.1 OpenLayers mapping

OpenLayers is an Open Source JavaScript library released under the FreeBSD License. OpenLayers implements a JavaScript API for building rich web-based geographic applications, similar to the Google Maps with one important difference - OpenLayers is Free Software, developed for and by the Open Source software community. OpenLayers allows dynamic maps in webpages. It can display map tiles and markers. A basic example of markers display in OpenLayers map is given in Figure [14].

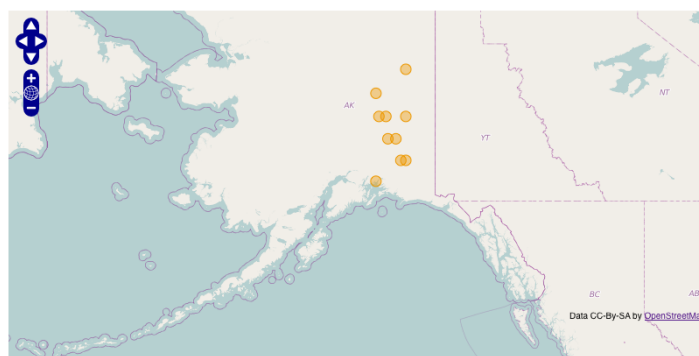


Figure 14: Basic example of mapping features with OpenLayers

OpenLayers API is or will be used in GEOME for instance to add layers, to create clusters of points depending of the scale and to show for instance only the individuals related to a given marker.

5.3.2 PostGIS spatial database

A specific aspect of GEOME is that it must be able handle spatial objects (e.g. point objects) and to relate them to other spatial object thanks to spatial queries (e.g to relate CRU database to the individuals). By installing PostGIS many functions are available to compute spatial queries.

5.4 Tests with different datasets

After finalising the prototype implementation, tests with three real datasets were done as shown in Table [8].

Species	Location	Individuals	Markers	Time to upload	Time to download
Frog	France	139	249	1 min	4 sec
Brown bear	Sweden	730	224	4 min	15 sec
Goat	Europe	1220	102	3 min	29 sec

Table 8: Datasets parameters and time results

The time to upload user inputs is important as the platform needs to validate each information given and to incorporate them into the database (average time: 41'000 entries per minute). The time to download environmental files are quicker as the matrixes generates are smaller (average time: 40 individuals x 117 environmental values / second). As ScythSAM calculation times depend of the parameters chosen and of the dataset configuration, they can not be compared directly in this table.

5.5 Hardware configuration

GEOME implementation was realised on a single machine. The development server, the geo-environmental data (CRU data) and the program executing the calculation are handled on the same machine. This 8-cores-computer will initially take care of the calculation parallelization in the different nodes locally available. The hardware configuration in a single machine helped to the established the communication between the different actors of the platform (e.g. Django, PostgreSQL, ScytheSAM), however it would possible to do remote communication in the future.

For the moment, GEOME works with the core features presented in the technical requirements. Possible further developments will be presented here below.

6 Possible further developments

GEOME prototype was designed expecting further improvements. New modules and features could be added to GEOME in order to give new services to its users or to improve the existing services.

6.1 Adding new modules to GEOME

Four principal modules could be added to GEOME prototype. The relationships between them are shown in Figure [15]. GEOME future web-based services could indeed be provide two complementary approaches to measure local adaptation: a spatial analysis approach (Joost & al. 2007 [9], existing module E) and a theoretical approach in population genomics (e.g. Foll and Gaggiotti 2008 [3], future module F). GEOME will also have a result analysis module (module H) that will help users to understand the difference between the two models. A new service of mapping of potential species habitats could also be implemented (modules G). An HPC environment (module I) would accelerate and parallelize module E, F and G calculations.

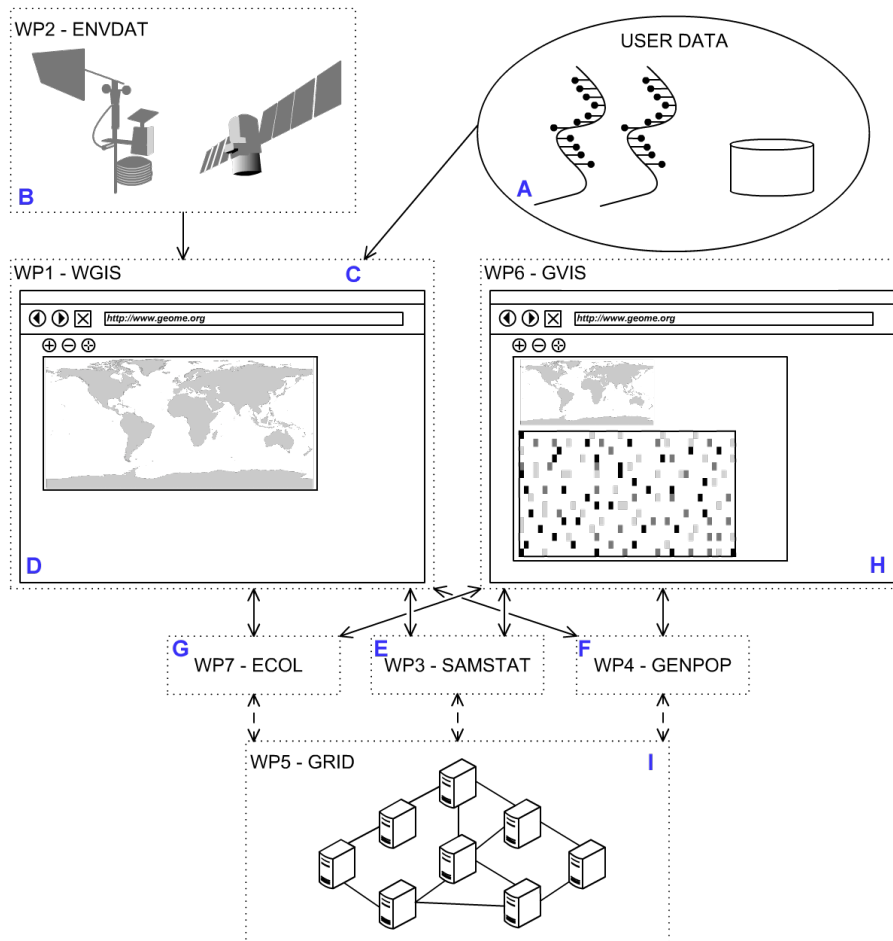


Figure 15: Possible new interconnected modules to GEOME, Source ref. [10]

Modules A, B, C and D are described in Figure[5] on page 6.

- F. Theoretical population genetics module (BayeScan) to differentiate neutral loci from loci under selection (Foll and Gaggiotti 2008 [3]);
- G. Predictive habitat modelling module to generate maps of potential habitat. It will include a statistical tool able to run Generalized Linear Models (GLM), in connection with module E. Indeed, the SAM approach is not restricted to molecular data and can also be used to detect environmental variables significantly associated with species presence data.
- H. Result analysis module with visual facilities for the comparison of results provided by modules E and F;
- I. High Performance Computing (HPC) module connected up to modules E, F and G to guarantee a powerful processing capacity.

6.2 Enhancing environmental data features

Django can handle multiple-database interactions ref. [16] and remote database connexions (by defining its host connexion). As the relation between a position and an environmental parameter is done by a spatial query, it is possible to access other regional or worldwide geo-environmental databases. For regional regions, it will be necessary to create a dynamic environmental data form, in order to give to possibility to the user to only choose available environmental data (a spatial test will be done on the dataset location before generating the environmental data form).

Thanks to Mapnik (see Figure [6] on page 17) tiles of existing environmental databases can be created at different scales (e.g. geographic or time scales) and be added on the interactive map of the dataset. This feature will give to users more information about the environmental data available on GEOME.

6.3 Improving collaboration between users

To improve the collaboration between users, an idea would be to give them more tools to interact. It would be possible to create a forum, see ref. [17], where users could add links to scientific articles and reviews from their datasets, as well as general comments on GEOME. Those tools will favour the emergence of a landscape genomics interactive community.

6.4 Deploying Django to a production server

For now GEOME is accessible with EPFL virtual private network (VPN) connection under the url: `http://lasigpc49.epfl.ch:8080/`. A domain name like for instance `http://www.geome.org` need to be bought before deploying Django to a production server. It is necessary to have a restricted circle of user testing and given feedback of the platform. It is also important to find out which is the limited number of simultaneous client connections in order to avoid that the server becomes unresponsive. This way GEOME will handle final details/problems with the implementation. Django supports Python's standard, called WSGI (Web Server Gateway Interface). This means that Django can run on any server that supports the rather simple WSGI standard. However Apache server with `mod_python`, see ref. [18], historically has been the suggested setup for using Django on a production server (Holovaty 2009 [5]).

7 Conclusion

GEOME presently provides tools necessary to compute association models in a complementary way with the ScytheSAM program. It gives access the CRU worldwide environmental data, to computational resources, to shared molecular datasets and spatial analysis result calculations. It is for now a collaborative platform in the sense that datasets can be shared, while preserving user privacy constraints by allowing the latter to keep studies or datasets private. It needed important conceptual and technical steps before being able to program and meet the business requirements. Those steps were decisive for choosing an adapted set of programming tools (i.e. Python and Django) which fulfilled the technical requirements.

GEOME prototype is now able to provide the whole circle of people active in the study of evolutionary processes and of biodiversity conservation in general with relevant environmental data sets, permitting the implementation of the landscape genomics approach together with relevant information on spatial autocorrelation. However, it first needs to be first tested by a small group of users to receive their feedback.

The next main important step will be the integration of a population genomics approach to be able to directly compare results provided by both approaches (BayeScan developer, Dr Matthieu Foll, has already been contacted for this purpose). Then GEOME will go on evolving, until it reaches a certain level of maturity to be deployed on a production server. Due to web technological improvements, it is expected web-based platforms to become new daily tools for scientists to carry out their investigations, to discuss and to share their researches within a community. GEOME final platform should be able to become one of those web-based research tools.

Acknowledgements

Many people deserve my deepest appreciation and thanks. I am grateful to LaSIG direct supervisor Stéphane Joost and his collaborators, Sylvie Stucki and Nicolas Lachance-Bernard, who guided and advised my work during all this semester. I would like to thank Thomas Heinis for his UML class diagram recommendation. Elisabeth Salazar deserves a special thought for the chatting and relaxing moments at LaSIG. I need also to express my gratitude to my dear friends Julien Eberle and Matthias Brändli for their technical support concerning Django and to Professor Elizabeth Wentz and my family for carefully proof-reading this report. Finally I would like to thank my responsible professor at EPFL, François Golay, who supported this project to the EPFL's administration.

References

- [1] E. Barraud, 2011 *Evolution of species is recorded in the genes*, Mediacom EPFL
<http://actu.epfl.ch/news/evolution-of-species-is-recorded-in-the-genes/>
Last visited: December 2011
- [2] K. Bittner, I. Spence, 2003. *Use case modeling*. Addison-Wesley. ISBN-13: 978-02017-09-131.
- [3] M. Foll, O. Gaggiotti, 2008. *A Genome-Scan Method to Identify Selected Loci Appropriate for Both Dominant and Codominant Markers: A Bayesian Perspective*. *Genetics* 180, 977-993
- [4] R. Holderegger, D. Herrmann, D. Poncet & al., 2008. *Land ahead: using genome scans to identify molecular markers of adaptive relevance*. *Plant Ecology & Diversity*, 1, 273-283.
- [5] A. Holovaty, J. Kaplan-Moss, 2009. *The Definitive Guide to Django: Web Development Done Right*, Apress publisher, Web ISBN: 1-4302-1936-X
- [6] J. Kozak, C. Estreguil, K. Ostapowicz, 2008. *European forest cover mapping with high resolution satellite data: The Carpathians case study*, *International Journal of Applied Earth Observation and Geoinformation*, 10, 1, 44-55
- [7] I-touch website <http://itouchmap.com/latlong.html>
Last visited: December 2011
- [8] S. Joost, 2009. *Early stirrings of landscape genomics: Awaiting next-next generation sequencing before take-off*, *FAO/IAEA International Symposium on Sustainable Improvement of Animal Production and Health*, 8-1, Vienna, Austria
- [9] S. Joost, A. Bonin, M.W. Bruford, L. Després, C. Conord, G. Erhardt, P. Taberlet, 2007. *A Spatial Analysis Method (SAM) to detect candidate loci for selection: towards a landscape genomics approach to adaptation.*, *Molecular Ecology*, 18: 3955–3969.
- [10] S. Joost, A. Pointet, N. Ray, S. Vuilleumier, 2010 *Application for CTI funding*
- [11] D.B. Lowry DB, 2010. *Landscape evolutionary genomics*. *Biology Letters*, vol 6. no.4 502-504
- [12] S. Manel, M. Schwartz, G. Luikart, P. Taberlet, 2003. *Landscape genetics: combining landscape ecology and population genetics*. *Trends in Ecology & Evolution* 18, 189-197.
- [13] M. New, D. Lister, M. Hulme, I. Makin, 2002 *A high-resolution data set of surface climate over global land areas*, *Climate Research*, *Clim Res* 21 p°1–25
- [14] DocForge community project with an open wiki for software developers, 2011 *Web application framework*, http://docforge.com/wiki/Web_application_framework
Last visited: December 2011
- [15] Pylons Project FAQ <http://docs.pylonsproject.org/en/latest/faq/pylonsproject.html>
Last visited: January 2011

- [16] Django: multiple-database interactions <https://docs.djangoproject.com/en/dev/topics/db/multi-db/>
Last visited: January 2011
- [17] Django: create a forum <http://code.google.com/p/django-forum/>
Last visited: January 2011
- [18] Django: mod_python http://www.djangoproject.com/r/mod_python/
Last visited: January 2011
- [19] Python Programming Language – Official website <http://python.org>
Last visited: December 2011
- [20] Java BluePrints, Model-View-Controller <http://java.sun.com/blueprints/patterns/MVC-detailed.html>
Last visited: January 2012
- [21] Odeon consulting group <http://od-eon.com/presentations>
Last visited: January 2012
- [22] Python and Django blog timchen119.blogspot.com
Last visited: January 2012

Appendices

A Inventory of web-based platforms in population genetics topics

Name of the Web platform	Function	Access mode	URL (last accessed -March 2010)
BIOPORTAL	Phylogenomic analysis, population genetics and high-throughput sequence analysis	Free	http://www.bioportal.uio.no
GeneGrid	Reliable and scalable access to large volumes of public and private biological data. Services to allow the generation and analysis of cancer and infectious disease genetic information from various distributed sources	Commercial	http://www2.besc.ac.uk/ http://www.omii.ac.uk/repository/projectj.html
GOTree Machine (GOTM)	A Gene Ontology (GO) enrichment analysis tool. It compares a user-uploaded gene list with all GO categories to identify those with enriched number of user-uploaded genes. Its primary purpose is to help users sort for interesting patterns in gene sets (Zhang et al. 2004).	Free License: GNU GPL	http://bioinfo.vanderbilt.edu/gotm/
GENOME Canada Bioinformatics Platform (BioMOBY)	Webservices for interoperability between biological data hosts and analytical services. Genome Alberta and Genome Canada fund the project through a Bioinformatics Platform. Genome Alberta and Genome Canada are not-for-profit organizations leading Canada's national strategy on genomics with \$ 600 million in funding from the Canadian federal government.	Freely available through open source licenses to ensure that Platform resources are accessible to all Genome Canada researchers and easily utilized	http://www.gcbioinformatics.ca/
GENOUEST (Genocluster platform)	Bioinformatics platform offering complete set of tools in bioinformatics, public databanks updated on a daily basis, and a range of links to seminars, training courses, platform news.	This is a public platform providing free services. On a paying basis, GENOUEST can carry out specific bioinformatic projects and training.	http://www.genouest.org
GEYSIR (NIAID)	A web application designed to correlate case-control association data with genetic and physical maps of the human genome, including key human genes associated with host immune response. This web platform was developed by the National Center for Genome Resources (http://www.ncgr.org/) as part of a population genetics project funded by the National Institute of Allergy and Infectious Diseases	GEYSIR is publicly available for the visualization and analysis of human genomic neighborhoods and SNP variants.	http://geysir.ncgr.org
SWISS-MODEL	A fully automated protein structure homology-modeling server, accessible via the ExPASy web server, or from the program DeepView (Swiss Pdb-Viewer).	Protein modelling is accessible to worldwide biochemists and molecular biologists (Arnold et al. 2006).	http://swissmodel.expasy.org
BABELOMICS	A complete suite of web tools for functional analysis of genome-scale experiments. Babelomics 3.2 includes the use of web services and Web 2.0 technology features, a user interface with persistent sessions and an extended database of gene identifiers. (Al-Sharour et al. 2006).	Free, users may register and subscribe to a mailing list, every user has a 1.5Gb of disk quota	http://www.fatigo.org/
PASE	Visualization of all the dimensions of "OMIC" data from genomic, transcriptomic, and proteomic domains (Hishiki 2006)	Free	http://www.geneexpo.jp/GeneOMIMViewer/jsp/viewer.html
KOBAS	Annotation of DNA sequences (Wu et al. 2006), etc.	Free registration required, dedicated user space	http://kobas.cbi.pku.edu.cn/
GVP (Green Visions Plan)	A web-based platform to support interactive environmental planning. Partnership between Southern California's state land conservancies and the University of Southern California, was forged to create a visionary plan and practical planning tools to promote habitat conservation (Maps of Target Species Habitat), watershed health and recreational open space (Ghaemi et al. 2009)	Free, login required	http://gv-server.usc.edu/GVWebTools_public_v2/viewer4.16/signin.asp http://www.greenvisionsplan.net/html/welcome.html
BIOS (Biogeographic Information & observation System)	BIOS is a system designed to enable the management, visualization, and analysis of biogeographic data collected by the Department of Fish and Game and its Partner Organizations. BIOS facilitates the sharing of those data within the BIOS community.	Open to the public (only non-sensitive data are included) and restricted access for additional secured layers included	http://bios.dfg.ca.gov/
CARES	A Decision Support System for Planning and Management of Sustainable Livestock Production in the Midwest	Free	http://cares.missouri.edu/projects/completed/old/LP.aspx

Figure 16: Non-exhaustive list of web-based platforms dedicated to population genetics topics. Source ref.[10]

An interesting example is the web-based service platform, BIOPORTAL, developed at University of Oslo for phylogenomic analysis, population genetics and high-throughput sequence analysis. BIOPORTAL is one of the largest public available computer resource with a cluster of 300 dedicated computational cores.

B ScytheSAM parameter file structure

Each line must begin with the name of the parameter. If a parameter is optional (F for "Facultatif"), the entire line can be omitted. The required parameters are marked with an (O for "Obligatoire"). The choices are bracketed.

- FILENAME (F) (These details can be provided as an argument to the program) Name(s) of file(s) containing the data (If the files are separated, write first the environmental data and second marker data)
 - HEADERS (F) presence or absence of names for variables [Y, N]. These names will be read on the first line. If no headers, the environmental variables are numbered P1, P2, P3 ... and markers M1, M2, M3.
 - NUMVARENV (O) Number of columns in environmental data file
 - NUMMMARK (O) Number of column in genetic data file
 - NUMINDIV (O) Number of individuals per dataset
 - COLSUPENV (F) Inactive columns for environmental data file. They are identified by their name if HEADERS is True, or by their number [0, ..., N-1] if headers is False.
 - COLSUPMARK (F) Inactive columns for genetic data file. They are identified by their name if HEADERS is True, or by their number [0, ..., N-1] if headers is False.
 - DIMMAX (F) Maximum size of the calculation for multivariate analysis [1 = univariate, 2= bivariate, 3 = trivariate ...]. DIMMAX is 1 if omitted.
 - SPATIAL (F) Names (or numbers) columns for longitude and latitude coordinate type [SPHERICAL, CARTESIAN], neighborhood type of calculation [DISTANCE, GAUSSIAN, BISQUARE, NEAREST] scale parameter (bandwidth or number of neighbors as appropriate)
 - AUTOCORR (F) (requires SPATIAL field) autocorrelation type [GLOBAL, LOCAL, BOTH], variables [ENV, MARK, BOTH] and number of permutation (by default 9999)
 - GWR (F) (requires SPATIAL field)
 - IDINDIV name_column_id_env name_column_id_mark; gives the name of the ID columns for each file
 - SUBSETVARENV (F) Selection of geographic data, list data to consider
 - SAVETYPE (O) Backup type: in real time or at the end [REAL END], backup (or not) of all data [ALL, BEST], if BEST, p-value limit for model selection
- / / Settings supervision
- 1: Input file name, environment variable file names
 - 2: Name parameter file
 - 3: Number of environment variables
 - 4: Number of genetic markers
 - 5: Number of lines
 - 6: Block Size
 - 7: Number of calculations for Windows and Unix

C User Manual : Tutorial

This tutorial gives guidelines on how to use GEOME. It is divided in subsections corresponding to the main steps to follow. If you are a GEOME new comers, please read this Tutorial before adding your own data.

C.1 Homepage

GEOME home page gives you general information about GEOME. This page explains the philosophy of the platform. It shows a series of statistics with basic information about each dataset. It gives references to open source environmental data and to scientific articles.

[Homepage](#) [Tutorial](#) [Methods](#) [Users](#) [Admin](#)

What is GEOME ?

GEOME is a web-based landscape genomics geocomputation platform for the integrated analysis of environmental and molecular data through the implementation of combined geocomputation, databases, spatial analysis and population genetics tools.

- This is an evolving platform (first prototype January 2012, more data and services to come);
- it gives spatial features for environmental data acquisition;
- it prototype aim is to be efficient but simple to use;
- this is a collaborative platform.

Potential users

GEOME targets different scientific and institutional communities active in the study of living organisms and their relationship to the environment: research laboratories in academics, specialized institutions of national states, international organizations (e.g. FAO), professional associations or private research organizations (e.g. WWF).

The potential users are:

- scientists (e.g. landscape or molecular ecologists, evolutionary biologists);
- professional users (e.g. resource conservation managers).

Philosophy

- This is a collaborative platform with user privacy;
- we work with OpenSource data and technologies;
- our mission is to give access to environmental data and to provide efficient calculation tools.

Statistics

Kingdom	Species	Number of individuals	Number of markers	Molecular marker type	Country	Status
Animalia	Bigfoot	12	4	microsat	Canada	Public
Animalia	Unicorne	12	4	AFLP	USA	Private

OpenSource environmental data

- [World wild climate data](#): Climate Research Unit (CRU) in Norwich
- [Shuttle Radar Topography Mission \(SRTM\)](#): U.S. Geological Survey (USGS), NASA global digital elevation models (DEMs) with a horizontal grid spacing of 1, 3 and 30 arc seconds (approximately 30m, 90m ([download](#)) and 1 kilometer).
- [Global Land Cover Characterization](#): U.S. Geological Survey (USGS), National Center for Earth Resources Observation and Science (EROS), University of Nebraska-Lincoln (UNL) and Joint Research Centre of the European Commission 1km resolution global land cover characteristics data base for use in a wide range of environmental research and modeling applications.
- [CORINE Land Cover](#): land cover database for the 15 EC Member States and other European and North African countries, at an original scale of 1: 100 000, using 44 classes.
- [International Soil Reference and Information Centre \(ISRIC\)](#): Wageningen
- [International Livestock Research Institute \(ILRI\)](#): layers directly related to livestock, such as distribution, health and production.
- [GeoGratis](#): portal provided by the Earth Sciences Sector (ESS) of Natural Resources Canada (NRCan)

Scientific articles

- Integrating geo-referenced multiscale and multidisciplinary data for the management of biodiversity in livestock genetic resources S. Joost, L. Colli & al., published online (2010, [Link](#))
- A Spatial Analysis Method (SAM) to detect candidate loci for selection: towards a landscape genomics approach to adaptation, S. Joost, A. Bonin, & al. (2007), *Molecular Ecology*, Vol.16, No 18, pp. 3955–3969.
- Spatial Analysis Method (SAM): a software tool combining molecular and environmental data to identify candidate loci for selection, S. Joost, M. Kalbermatten, A. Bonin (2008), *Molecular Ecology Resources*, 8:957–960.
- Landscape genomics and biased FST approaches reveal Single Nucleotide Polymorphisms under selection in goat breeds of North-East Mediterranean, L. Pariset, S. Joost, P. Ajmone Marsan, A. Valentini (2009), *BMC Genetics*, 10:7.

Figure 17: Tutorial: View GEOME homepage

At any time, you can click on a part of the menu to come back to one of the principal sections (e.g. Homepage, Tutorial, Methods, User).

C.2 User registration

In this development period, accounts are given directly by the administrator. If you are interested in joining GEOME, please write an e-mail to stephane.joost@epfl.ch, please specify your name, first name, affiliation, location and e-mail. Once you receive your user account, you can access the "User" section of the general menu. You need to provide your user name and password to verify your account privacy and authorisations. This way GEOME makes sure that no other user can view for instance your private dataset.

Figure 18: Tutorial: Login GEOME

C.3 Studies and datasets

Once you are logged in, you will see two tables: one with your own datasets and the datasets in which you are running calculations, and another one with all the remaining public datasets. In this section you can add a new study and view a specific study.

Kingdom	Species	Creation date	Status	Authorised users
Animalia	Bigfoot	2011 / 12 / 28	Public	lasig_admin
Animalia	Unicorne	2011 / 12 / 28	Private	lasig_admin

Kingdom	Species	Creation date	Authorised users
Animalia	T-Rex	2012 / 01 / 7	test_user

Figure 19: Tutorial: Overview of available studies

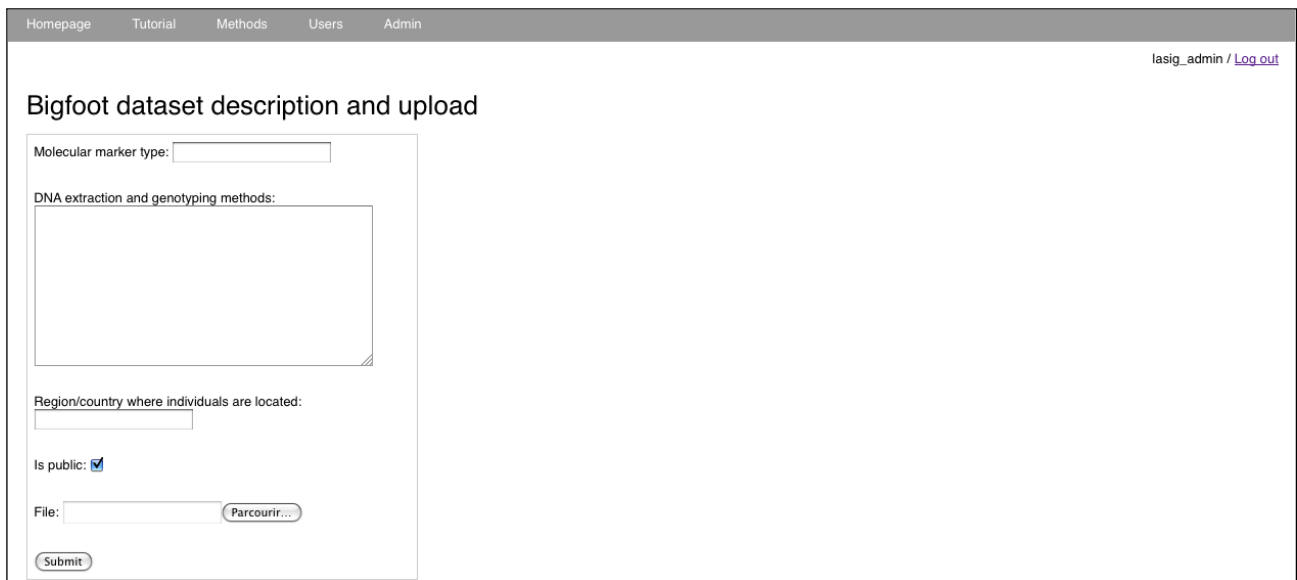
To add a public or private study click on the link "New study", then fill in the form shown in Figure [20]. If you choose to create a private study, you should carefully select the users that can work on this study. We recommend to create public studies and private dataset. This way it will be easier to share it later on by changing its private status to public.

Figure 20: Tutorial: Add a new study

To validate your form, click on the “Submit” button. If the form has been correctly fill in, you will be redirected to the page of the study you have just created. This study will currently have no dataset, see Figure [21].

Figure 21: Tutorial: Example of an empty study

It is possible to add one dataset by clicking on “Add (public or private) dataset” link. A new form will appear as shown in Figure [22]. You need to upload the molecular data file containing your dataset. This file must have the following columns: ind |xlon |ylat |breed (optional) |marker1 |... |markerN. Each marker can have it own name. Each line of the matrix corresponds to an individual. The longitude and latitude columns must be in decimal degrees in WS84 coordinates system (e.g. GPS). The presence or absence of markers in an individual must be notified by 1 respectively 0 number. If a data is missing, please fill in its cell with a “NaN” (Not-a-Number). Any invalid cell will be filled in by a “NaN” value in the system.



The screenshot shows a web interface for adding a dataset. At the top, there is a navigation bar with links for 'Homepage', 'Tutorial', 'Methods', 'Users', and 'Admin'. In the top right corner, the user is identified as 'lasig_admin' with a 'Log out' link. The main heading is 'Bigfoot dataset description and upload'. The form contains several fields: 'Molecular marker type:' with a text input; 'DNA extraction and genotyping methods:' with a large text area; 'Region/country where individuals are located:' with a text input; 'Is public:' with a checked checkbox; 'File:' with a text input and a 'Parcourir...' button; and a 'Submit' button at the bottom left.

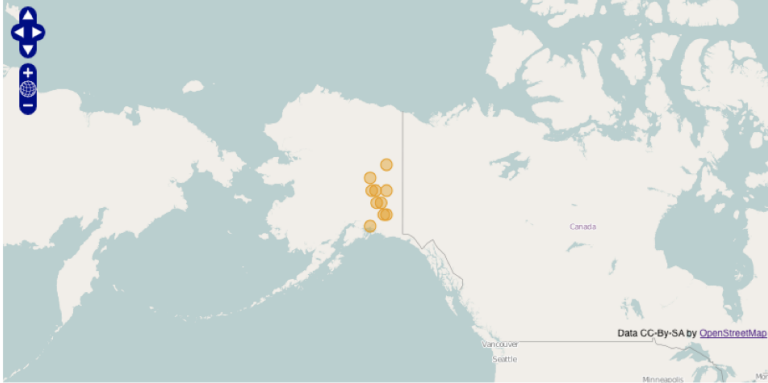
Figure 22: Tutorial: Add a dataset

After fill in the form, click on "Submit". This operation takes approximately 1 minute for a 40'000 cells matrix. Please be patient as the system verifies and records your data. You then will be directly redirected to view your dataset, see Figure [23]. By viewing a dataset, you can have a look at its spatial distribution in the interactive map, at its meta-information and at its marker frequency. From this point, it is possible to view previous calculations done with ScytheSAM calculation program, to download environmental data corresponding to the dataset or to run new calculations with different parameters.

Homepage
Tutorial
Methods
Users
Admin

lasig_admin / [Log out](#)

Map: dataset 21 (Animalia, Bigfoot)



Metadata of Dataset 4

Owner: lasig_admin
 Date uploaded: Jan. 16, 2012, 9:31 a.m.
 Status: Public
 Number of individuals: 10
 Number of markers: 4
 Molecular marker type: AFLP
 DNA extraction and genotyping methods: Description

Display list of individuals

Ind5: (-144.0, 64.0)
 Ind4: (-147.0, 61.0)
 Ind3: (-146.7, 64.0)
 Ind2: (-145.8, 63.0)
 Ind1: (-144.5, 62.0)
 Ind5: (-144.0, 66.0)
 Ind4: (-147.0, 65.0)
 Ind3: (-146.0, 64.0)
 Ind2: (-145.0, 63.0)
 Ind1: (-144.0, 62.0)

Marker statistics

Marker	a403	a50	a99	a149
Frequency	100.0 %	0.0 %	60.0 %	60.0 %

List of calculations

[Run a new calculation](#)
[Only download environmental data](#)
[Go back to the other datasets for Bigfoot](#)
[Study overview](#)

Figure 23: Tutorial: Visualise of a dataset

It is now possible to download only environmental data or to run new calculation.

C.4 Download environmental data or run calculation

To download environmental data, click on "Download environmental data" link. A new form opens as shown in Figure [24]. Once you choose the environmental parameters to download and click on the "Submit" button. The browser asks you where to save the environmental_data.csv file (pop-up window in the right of Figure [24]).

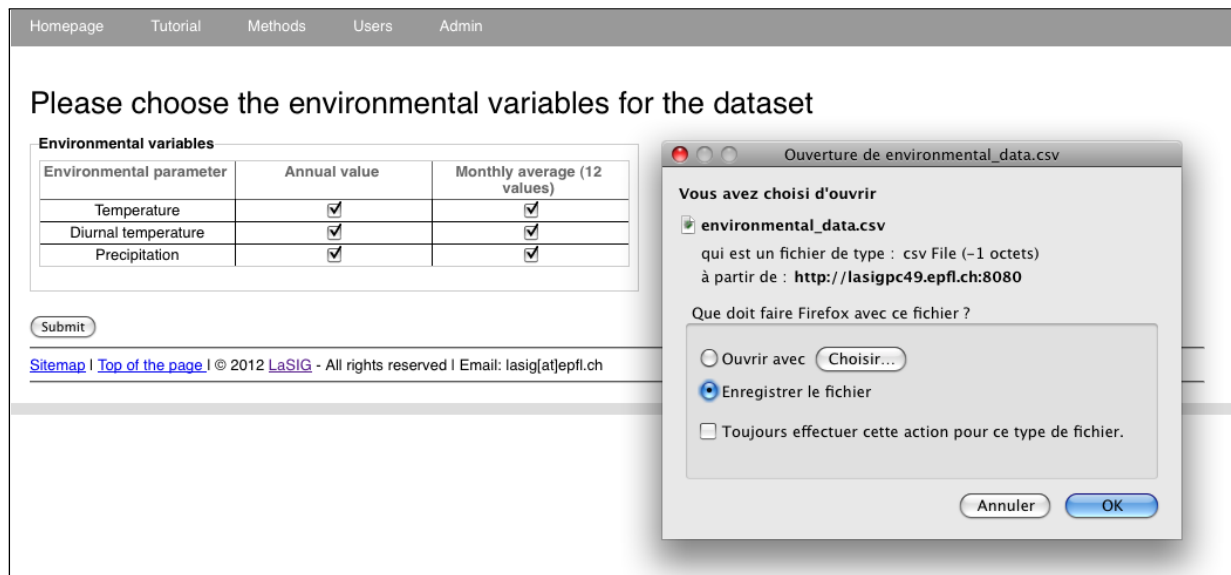


Figure 24: Tutorial: Download environmental data

Going back to the dataset in Figure [23], you can also run a calculation by clicking on "Run a new calculation" link. The calculation methods handle by ScytheSAM are: uni or multivariate predictions, spatial autocorrelation measure, spatial weighting, number of permutation and P-value threshold. Only select the calculation methods in which you are interested for. Don't forget that those calculations are time expensive, so you should carefully choose the methods you are interesting in. If you didn't select a required calculation, default values will be taken (e.g P-value of 0.05, 9999 permutation, univariate mode etc...)

Homepage
Tutorial
Methods
Users
Admin

Please choose the environmental variables and the methods for ScySAM calculations

Environmental variables

Environmental parameter	Annual value	Monthly average (12 values)
Temperature	<input type="checkbox"/>	<input type="checkbox"/>
Diurnal temperature	<input type="checkbox"/>	<input type="checkbox"/>
Precipitation	<input type="checkbox"/>	<input type="checkbox"/>

Number of prediction

Univariate
 Multivariate (maximal dimension)

Spatial autocorrelation measure

Global
 Local
 Both

Variable for autocorrelation

Environmental
 Marker
 Both

Spatial weighting

Nearest neighbours (number)
 Distance (kilometers)
 Gaussian (bandwidth) GWR
 Bisquare (bandwidth) GWR

Number of permutation

value from 0 to 50'000

P-value threshold

value from 0.001 to 0.05

[Sitemap](#) | [Top of the page](#) | © 2012 LaSIG - All rights reserved | Email: lasig[at]epfl.ch

Figure 25: Tutorial: Fill the form to run the calculation

After clicking on the "Submit" button, you will lounge the calculations. These calculation can take several minutes depending on the size of the dataset and on the calculation that you want to do. You will therefore receive and e-mail notifying that the calculation are over and that you can download the results and the log file from ScytheSAM.

D Technical Manual

This technical manual has been written to explain the set-ups and the technical implementation of GEOME prototype. The terminal commands are given for Linux operating system. Its aim is to expose where folders and files are located on lasigpc49 machine and what they are used for, in order to give the key elements of the web-based platform. Many information (e.g. "docs string" and comments) are also inside the different python scripting files and please have a look on README.txt files.

D.1 Installations

This section explains the installations required to run GeoDjango web framework and spatial analysis method program ScythSAM. In general GeoDjango installation requires: Python 2.4+, Django, Spatial Database (PostgreSQL and PostGIS) and Geospatial Libraries.

Django and Python installation

Download the last released version (here below 1.3.1) on <https://www.djangoproject.com/download/>

```
1 $ tar xzvf Django-1.3.1.tar.gz
2 $ cd Django-1.3.1
3 $ sudo python setup.py install
```

Geodjango libraries installation

The required binary prerequisites for Geodjango packages are given in Table [9]:

Library	Purpose
binutils	for ctypes to find libraries
postgresql-8.3	for database management
postgresql-server-dev-8.3:	for pg_config
postgresql-8.3-postgis	for PostGIS 1.3.3
libgeos-3.0.0, and libgeos-c1	for GEOS 3.0.0
libgdal1-1.5.0	for GDAL 1.5.0 library
proj	for PROJ 4.6.0 – but no datum shifting files
python-psycopg2	for translation of Python to PostgreSQL
python-setuptools	for easy_install
libgeoip1	for GeoIP support
gdal-bin	for GDAL command line programs like ogr2ogr
python-gdal	for GDAL's own Python bindings (raster manipulation)

Table 9: Required packages for Geodjango

All those libraries are installed with the following commands.

```
1 $ sudo apt-get install binutils gdal-bin postgresql-8.4-postgis postgresql-server-dev-8.4 python-
   psycopg2 python-setuptools
2 $ sudo easy_install Django
```

PostgreSQL 9.1 problems

On lasigpc49, PostgreSQL 9.1 version was installed.

```
1 $ sudo apt-get install postgresql postgis
```

PostgreSQL needed unusual set-ups due to an incompatibility between Django 1.3 and PostgreSQL 9.1: DatabaseError (invalid byte sequence for encoding "UTF8": 0x00). This problem should be resolved with Django 1.4. Using Django 1.3 and PostgreSQL 8.4, there is no incompatibility. Two steps were needed to solve the incompatibility problem. The first step was to change PostgreSQL cluster initialisation with Collation³ and Ctype⁴ set at C. We changed the configuration of the database cluster to alter both previous variables.

```
1 $ sudo -s # to be in root
2 root@lasigpc49:~# /etc/init.d/postgresql stop # to stopper PostgreSQL
3
4 # Deleting the existing cluster (with specific Ubuntu scripts)
5 root@lasigpc49:~# pg_lsclusters
6 Version Cluster  Port Status Owner    Data directory          Log file
7 9.1      main      5432 down  postgres /var/lib/postgresql/9.1/main  /var/log/postgresql/
      postgresql-9.1-main.log
8
9 root@lasigpc49:~# pg_dropcluster
10 Usage: /usr/bin/pg_dropcluster [--stop] <version> <cluster>
11
12 root@lasigpc49:~# pg_dropcluster 9.1 main
13 root@lasigpc49:~# pg_lsclusters
14 Version Cluster  Port Status Owner    Data directory          Log file
15
16 # Cluster deleted
17
18 # Creation and start of the cluster with the C local configuration
19 root@lasigpc49:~# pg_createcluster 9.1 main --locale C -e UTF8 --start
20
21 root@lasigpc49:~# /etc/init.d/postgresql start #to restart PostgreSQL with the new Cluster
22 root@lasigpc49:~# exit # to quit root mode
```

The second step was to modify postgresql.conf file by deactivating standard_conforming_strings (which by default is "on"). Source <https://github.com/coderholic/django-cities/issues/25>

```
1 $sudo -s # to be in root
2 root@lasigpc49:~# /etc/init.d/postgresql stop #to stopper PostgreSQL
3
4 $ vim /etc/postgresql/9.1/main/postgresql.conf
5 #Change "on" by "off" and uncomment standard_conforming_strings = off
6
7 root@lasigpc49:~# /etc/init.d/postgresql start # to restart PostgreSQL
8 root@lasigpc49:~# exit # to quit root mode
```

³A collation is a set of rules about how a set of objects should be ordered

⁴CType assures supported character sets

D.2 Initial set-ups on Django and PostgreSQL

The following list of initial set-ups need to be done only once when installing GEOME prototype on a new server environment.

Create *geome* geodatabase and *geome_admin* user

Several steps are needed to create *geome* geodatabase: (i) add PostGIS template to PostgreSQL, (ii) create *geome_admin* user on PostgreSQL, (iii) create *geome* geodatabase with PostGIS template and *geome_admin* owner .

To create a geodatabase in PostgreSQ, we need to have the `template_postgis`. We have to execute once the bash script `create_template_postgis-1.5.sh` (here below is the script) is given by PostGIS website. NOTE: on Ubuntu, `pg_config` does not work.

```

1 #!/usr/bin/env bash
2 POSTGIS_SQL_PATH=usr/share/postgresql/9.1/contrib/postgis-1.5
3 PSQL_PARAMS=""
4 createdb ${PSQL_PARAMS} -E UTF8 template_postgis # Create the template spatial database.
5 createlang ${PSQL_PARAMS} -d template_postgis plpgsql # Adding PLPGSQL language support.
6 psql ${PSQL_PARAMS} -d postgres -c "UPDATE pg_database SET datistemplate='true' WHERE datname='
   template_postgis';"
7 psql ${PSQL_PARAMS} -d template_postgis -f $POSTGIS_SQL_PATH/postgis.sql # Loading the PostGIS SQL
   routines
8 psql ${PSQL_PARAMS} -d template_postgis -f $POSTGIS_SQL_PATH/spatial_ref_sys.sql
9 psql ${PSQL_PARAMS} -d template_postgis -c "GRANT ALL ON geometry_columns TO PUBLIC;" # Enabling
   users to alter spatial tables.
10 psql ${PSQL_PARAMS} -d template_postgis -c "GRANT ALL ON geography_columns TO PUBLIC;"
11 psql ${PSQL_PARAMS} -d template_postgis -c "GRANT ALL ON spatial_ref_sys TO PUBLIC;"

```

To execute this script run the command on a terminal:

```

1 $ sudo -u postgres bash create_template_postgis-1.5.sh
2 # If it worked, terminal should display:
3 .....
4 INSERT 0 1
5 COMMIT
6 ANALYZE
7 GRANT

```

Then access PostgreSQL 9.1 and run the commands as shown in Figure [26]. This commands create the new user *geome_admin* and the new geodatabase *geome* with *geome_admin* as owner. PostgreSQL commands `"\l"` and `"\q"` are used to list the databases and to quit the program. PostgreSQL commands can be found on PostgreSQL documentation webstie <http://www.postgresql.org/docs/>. We can see in in Figure [26] that *geome* database was created.

```

lasigadmin@lasigpc49:~$ sudo -u postgres psql
psql (9.1.1)
Type "help" for help.

postgres=# \l
              List of databases
  Name          | Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
 postgres      | postgres | UTF8     | C       | C     | 
 template0     | postgres | UTF8     | C       | C     | =c/postgres      +
               |         |         |         |         | postgres=CTc/postgres
 template1     | postgres | UTF8     | C       | C     | =c/postgres      +
               |         |         |         |         | postgres=CTc/postgres
 template_postgis | postgres | UTF8     | C       | C     | 
 (4 rows)

postgres=# CREATE USER geome_admin WITH ENCRYPTED PASSWORD 'geome1306';
CREATE ROLE
postgres=# CREATE DATABASE geome WITH TEMPLATE template_postgis OWNER geome_admin;
CREATE DATABASE
postgres=# \l
              List of databases
  Name          | Owner   | Encoding | Collate | Ctype | Access privileges
-----+-----+-----+-----+-----+-----
 geome          | geome_admin | UTF8     | C       | C     | 
 postgres      | postgres | UTF8     | C       | C     | 
 template0     | postgres | UTF8     | C       | C     | =c/postgres      +
               |         |         |         |         | postgres=CTc/postgres
 template1     | postgres | UTF8     | C       | C     | =c/postgres      +
               |         |         |         |         | postgres=CTc/postgres
 template_postgis | postgres | UTF8     | C       | C     | 
 (5 rows)

postgres=#

```

Figure 26: Creating *geome* geodatabase and *geome_admin* user on PostgreSQL

Note: If there is any problem of PostgreSQL access, have a look in `pg_hba.conf` file.

Django initialisation

Synchronise Django and PostgreSQL geodatabase

When launching GEOME application, we must ask Django to create the Tables corresponding to its Model (`geome/model.py`) in PostgreSQL *geome* database. In the application folder *geome*, where `manage.py` and `setting.py` are located, run the command:

```
1 $ python manage.py syncdb
```

The `syncdb` command is a simple “sync” of the models to the database. It looks at all of the models in each app in the `INSTALLED_APPS` setting of `setting.py`, checks the database to see whether the appropriate tables exist yet, and creates the tables if they don’t yet exist.

Note: `syncdb` does not detect changes in models or deletions of models. The update of the database, if a model is changed or deleted, must be done “by hand” in PostgreSQL.

When running the previous command, a list of table should appear as shown in Figure [27].

```

Creating table auth_group
Creating table auth_user_user_permissions
Creating table auth_user_groups
Creating table auth_user
Creating table auth_message
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table geome_userextrinfos
Creating table geome_study_user
Creating table geome_study
Creating table geome_dataset
Creating table geome_method
Creating table geome_parameter
Creating table geome_runcalculation_method
Creating table geome_runcalculation
Creating table geome_individual
Creating table geome_marker
Creating table geome_relationship
Creating table geome_crudata
Creating table geome_crutemperature
Creating table geome_cruprecipitation
Creating table geome_crucvprecipitation
Creating table geome_crudiurnaltemperature
Creating table geome_crurelativehumidity
Creating table geome_crugroundfrost
Creating table geome_cruwetdays
Creating table geome_crusunshinepercent
Creating table geome_cruwindspeed

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (Leave blank to use 'lasigadmin'): lasig_admin
E-mail address: lasig@epfl.ch
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
No fixtures found.
lasigadmin@lasigpc49:~/Documents/django/geome_platform$

```

Figure 27: Synchronising Django and PostgreSQL geodatabase

CRU environmental data

CRU environmental data is added (only once) in *geome* database in two steps. The first step, add the CRUdata index table, which is a spatial class by calling the script `load_env_data.py` and call the function `load_CRUdata_shp()` as shown in Figure [28]. This step takes approximately 5 minutes.

```

lasigadmin@lasigpc49:~/Documents/django/geome_platform$ python manage.py shell
Python 2.7.2+ (default, Oct 4 2011, 20:03:08)
Type "copyright", "credits" or "license" for more information.

IPython 0.10.2 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object'. ?object also works, ?? prints more.

In [1]: from geome import load_env_data

In [2]: load_env_data.load_CRUdata_shp()

```

Figure 28: Adding CRU data grid to *geome* geodatabase

The second step, add the CRU parameters, with the function `add_env_tables()`, as shown in Figure [29]. As there are 8 parameters tables to link to CRUdata index table, it takes approximately one hour to insert all these tables into geome database.

```
lasigadmin@lasigpc49:~/Documents/django/geome_platform$ python manage.py shell
Python 2.7.2+ (default, Oct 4 2011, 20:03:08)
Type "copyright", "credits" or "license" for more information.

IPython 0.10.2 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object'. ?object also works, ?? prints more.

In [1]: from geome import load

In [2]: ls
geome/  __init__.py  __init__.pyc  input_output_scythsam/  manage.py  old_data/  settings.py  settings.pyc  templates/

In [3]: from geome import load_env_data

In [4]: load_env_data.add_env_table()
frs: Done
```

Figure 29: Adding CRU data parameters to *geome* database

Add PostGIS new function

The operation of adding a PostGIS function, see Figure [30] must be done only once.

```
lasigadmin@lasigpc49:~/Documents/django$ sudo -u postgres psql geome
[sudo] password for lasigadmin:
psql (9.1.1)
Type "help" for help.

geome=# CREATE OR REPLACE
geome-# FUNCTION nearestCRUPoint(point1 geometry, distance_radius double precision)
geome-# RETURNS integer AS $BODY$
geome$#     SELECT idcrugrid
geome$#     FROM geome_crudata
geome$#     WHERE expand($1,$2) && geome_crudata.geom
geome$#     ORDER BY distance(geome_crudata.geom, $1) LIMIT 1
geome$#     $BODY$
geome-#     LANGUAGE 'sql'
geome-#     VOLATILE;
CREATE FUNCTION
geome=#
```

Figure 30: Adding PostGIS new function

It uses a combination of "expand" and "distance" pre-existing functions. Expand function finds if a point of the CRUdata (`geome_crudata.geom`) is inside a circle of a radius `distance_radius` and centre `point1`. Distance function gives the distance between to point, as we use "ORDER BY" and "LIMIT 1" SQL options, we obtain the closest CRU point in the perimeter. Afterwards, it is possible to run the `nearestCRUPoint` function as shown in Figure [31]. This function is used in the `manage_inputs_outputs.py` in order to associate to each coordinates of an individual, a CRU point grid and its climate parameters.

```
geome=# SELECT nearestCRUPoint(ST_PointFromText('POINT(46.59 7.86)', 4326), 0.1);
nearestcrupoint
-----
133845
(1 row)
```

Figure 31: Example of how to use PostGIS new function

D.3 Running the application

To make GEOME web-based platform accessible from `http://lasigpc49.epfl.ch:8080`, we need to run the command in Figure [32].

```

lasigadmin@lasigpc49: ~/Documents/django/geome_platform$ python manage.py runserver 0.0.0.0:8080
Validating models...

0 errors found
Django version 1.3.1, using settings 'geome_platform.settings'
Development server is running at http://0.0.0.0:8080/
Quit the server with CONTROL-C.

```

Figure 32: Running the server on local base

This console can be useful when debugging. It shows each “print” command done in the Python scripts and each Http requests from the web-browser (e.g. Firefox).

D.4 Working with Django

Django web framework communicates with the browser, the web server and the database. The explanations are in part extracted from the Django documentation, ref. [5]. A schema of Django’s organisation is given in Figure [33]: it combines MVC design pattern and configuration settings.

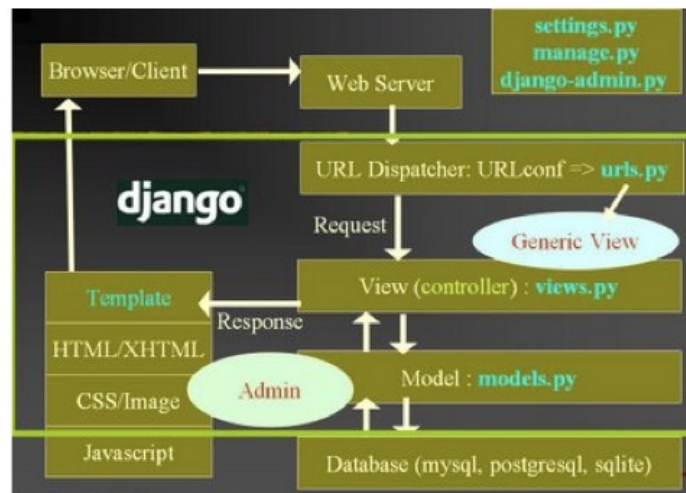


Figure 33: Overview of Django’s way to work, Source ref.[22]

In summary for Figure[33]:

- A request comes in to a particular URL.
- Django determines the root URLconf by looking at the `ROOT_URLCONF` setting.
- Django looks at all of the URLpatterns in the URLconf for the first one that matches the particular URL.
- If it finds a match, it calls the associated view function.
- The view function interacts with the `models.py` and the database and returns an `HttpResponse`.
- Django converts the `HttpResponse` to the proper HTTP response with this template, which results in a webpage.

Django works with project (configuration) and apps (portable and reusable codes across multiple projects). The contents of the page are produced by a view function, and the URL is specified in a URLconf. Django is a database-driven framework: the content of the website is stored in a relational database. This allows a clean separation of data and logic (in the same way views and templates allow the separation of logic and display). A database-driven Web site connects to a database server, retrieves some data out of it, and displays that data on a webpage. The site might also provide ways for site visitors to populate the database on their own.

In a Django project three files are required to fulfil the configuration: `__init__.py`, `manage.py` and `settings.py`. They are created while initialising the project with the "run django-admin.py startproject geome_platform" command, `geome_platform` being the name of GEOME django project. `__init__.py` is an empty file required for Python to treat `geome` directory as a package (i.e., a group of Python modules).

manage.py

This file is a command-line utility that Django project.

```
1 $ python manage.py syncdb # to synchronise PostgreSQL database and Django models
2 $ python manage.py shell # to run a interactive console (as Ipython) which can access Django modules
3 $ python manage.py runserver # to run on localhost on port 8000 the website (\url{http://localhost
   :8000/})
4 $ python manage.py runserver 0.0.0.0:8080 # to run on the computer IP on port 8080 the webstie (e.g
   .\ \url{http://lasigpc49.epfl.ch:8080/}
```

When running `python manage.py runserver`, the script looks for a file called `settings.py` in the same directory as `manage.py`. Then it checks each of the URLpatterns in that URLconf, in order, comparing the requested URL with the patterns one at a time, until it finds one that matches. When it finds one that matches, it calls the view function associated with that pattern, passing it an `HttpRequest` object as the first parameter.

settings.py

This file defines the configuration for the projec (`ROOT_URLCONF`, `TEMPLATE_DIRS`, `DATABASE_NAME`). `ROOT_URLCONF` tells Django which Python module should be used as the URLconf for this website.

The "python manage.py startapp geome" command creates a a Django app (e.g. a `geome` directory within the `geome_platform` directory) which contains four files: `__init__.py`, `models.py`, `views.py`, `urls.py`.

views.py

To create a webpage, a regular expression needs to be written in the `urls.py` file which is mapped to a view function. The view function only requires a request object as an argument (HTTP request) but other arguments can be given, and once it is done processing it either generates a response (e.g. render an HTML page with content from the database), or redirects to a file. A view function is therefore just a Python function that takes an `HttpRequest` as its first parameter and returns an instance of `HttpResponse`.

models.py

Table names are automatically generated by combining the name of the app (`geome`) and the lowercase name of the model (e.g. `user`, `study`, `dataset`). Django adds a primary key for each table automatically — the `id` fields if no primary key field is defined. By convention, Django appends “`_id`” to the foreign key field name. The foreign key relationship is made explicit by a `REFERENCES` statement.

urls.py

This file is like a “table of contents” of the Django-powered site giving the possible URLs. The URLs are defined thanks to regular expressions <http://docs.python.org/library/re.html>. If we had used the pattern `^hello` (without a dollar sign at the end), then any URL starting with `/hello/` would match, such as `/hello/foo` and `/hello/bar`, not just `/hello/`. Similarly, if we had left off the initial caret character (i.e. `hello/$`), Django would match any URL that ends with `hello/`, such as `/foo/bar/hello/`. If we had simply used `hello/`, without a caret or dollar sign, then any URL containing `hello/` would match, such as `/foo/hello/bar`. Patterns include a caret (^) and a dollar sign (\$). These are regular expression characters that have a special meaning: the caret means “require that the pattern matches the start of the string,” and the dollar sign means “require that the pattern matches the end of the strings.”

Templates

Django templates are basic HTML with some variables, template tags and filters thrown in. Any text surrounded by a pair of braces (e.g. `{{person.name}}`) is a variable. This means “insert the value of the variable with the given name”. Any text that’s surrounded by curly braces and percent signs (e.g. `{% if con %}... {% else %}... {% endif %}` or `{% for item in item_list %}... {% endfor %}`) is a template tag. The definition of a tag just tells the template system to execute an (e.g. a for loop or logical “if” statement). Any text that’s surrounded by double curly braces is a filter, which is the most convenient way to alter the formatting of a variable. In this example, `{{datefield|date:"F j, Y"}}`, we’re passing the `datefield` variable to the `date` filter, giving the date filter the argument “`F j, Y`”. The date filter formats dates in a given format, as specified by that argument. Filters are attached using a pipe character (`|`), as a reference to Unix pipes.

To avoid duplication of HTML code, the solution with Django is to use a strategy called template inheritance. In essence, template inheritance lets build a base “skeleton” template that contains all the common parts of your site and defines “blocks” that child templates can override. Generally, the more `{% block %}` tags in the base templates, the better. The child templates don’t have to define all parent blocks, so you can fill in reasonable defaults in a number of blocks, and then define only the ones you need in the child templates.

Admin interface is an application built with the framework that comes bundled with it. Django provides an optional administrative CRUD (create, read, update and delete) interface that is generated dynamically through introspection and configured via admin models.

D.5 Synopsis

The synopsis gives a basic description of all the folder and files needed and their role within the framework. There are two categories of files and folders, one corresponding to `geome_platform`, see Figure [34] and the other corresponding to ScytheSAM inputs and outputs, see Figure [35].

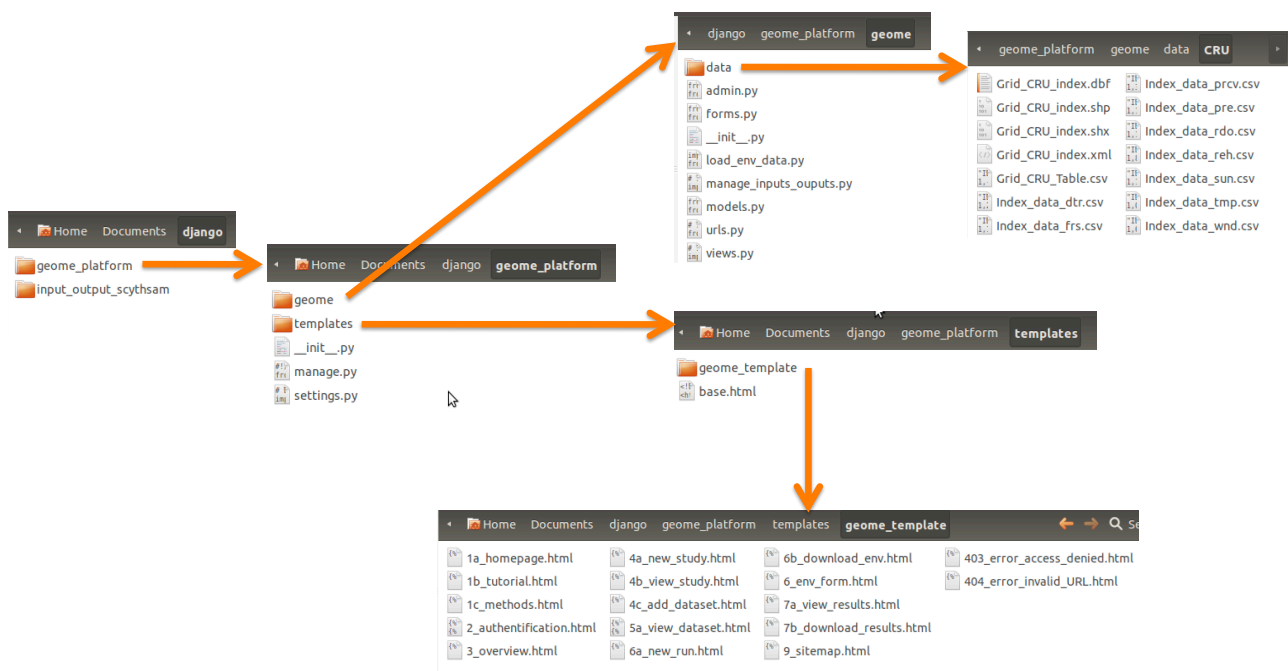


Figure 34: Files and folders organisation of `geome_platform` folder

In the folder `geome_platform`, there is the configuration files (`__init__.py`, `settings.py` and `manage.py`) and two folders (e.g. `geome` folder contains the logic of the platform and `templates` folder that contains the display of the platform). In `geome` folder:

- `admin.py` controls the administration part of the platform;
- `models.py` defines the database classes and their relations;
- `forms.py` gives the form field types;
- `urls.py` structures the website access;
- `views.py` does the bridge between `urls.py` and `models.py`
- `manage_inputs_outputs.py` handles the inputs from the user and the file creation for ScytheSAM;
- `load_env_data.py` automatizes the integration of CRU data into the PostgreSQL database.

The `data` folder contains the CRU data folders (e.g. a shape file of the grid of centroids and the parameters text file with the code of the grid and the monthly and annual climate parameters values). In `templates` folder, there

is a `base.html` file and a folder `geome.templates`. The `base.html` file is the basic template of the platform and it incorporates the CSS (e.g. design) features. It defines the menu bar, the footer and all the blocks that the inherited templates could fill with content. In `geome.templates` folder we can find all the templates corresponding to the webpages. Their numbering is done according to the paper prototyping. The inheritance model is again used in those files. Indeed, all webpages inside the user interface must have a mention of "user and log-out" so those elements are only defined once in `2_authentication.html`.

In the folder `input_output_scythesam`, as shown in Figure [35], the inputs and outputs from ScytheSAM are kept in an organised way. Each dataset has its folder containing its `molecular_data.csv` file and folders with the different runs. This is done to keep only once the molecular information as it is required for each calculation. In each `run_id` folder, there are the environmental data, the parameters, the ScytheSAM log and outputs files. Those files are different for each calculation.

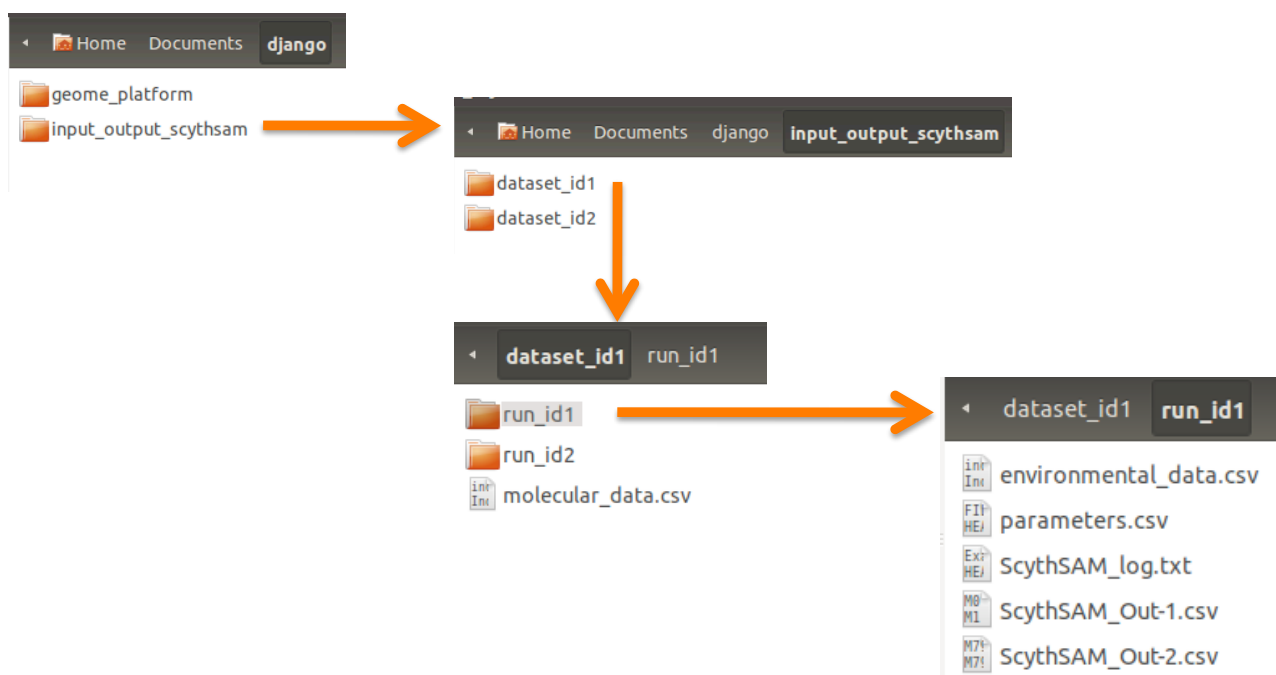


Figure 35: Files and folders organisation of `input_output_scythesam` folder

The user will have access to those inputs and outputs through Django's interface.