Genealogy Web App

PROJECT WORKBOOK – TEAM GNLG2

MAHDI CHAYMAE - COSTA CUNHA IVO - BOUSADIA LAHCENE – MACKPAYEN PRINCE
SAMIA OUSSAMA

Table of contents

Genealogy Web App – Project Workbook – Team GNLG2

1. Overview

1.0. Objective of the project

The crowning of each university or technical formation is always done through an end of studies project or internship. Our training as part of this formation of a professional nature ends with disciplinary project spanning over two months.

In this context out training allowed us to take advantage of many opportunities to conduct group projects to refine our professional objectives.

Disciplinary group projects lead students to develop several transferable skills such as communication, cooperative and teamwork skills like planning, management, leadership and peer support. And so, we hope that this project will allow our team to develop or refine these skills.

The project itself consists of the modernization of a genealogy website called "expoactes" that allows genealogists to work together on old hard to read or partially erased birth, marriage, and death certificates to first digitalize them and secondly deduce family relations through them.

The project also consists of new functions to be added to the old website.

1.1. Organization

The development team members are composed of 1 project manager and 4 developers.

MAHDI Chaymae will assume the role of project manager, will communicate with the client but will also participate in the redaction of documentation, in the conception, and in the development as needed.

The 4 developers will be BOUSADIA Lahcene, MACKPAYEN Prince Divin, SAMIA Oussama, and COSTA CUNHA Ivo. Those developers will have the same tasks which will be the redaction of documentation, the conception and the development which will include of course the mandatory tests.

BOUSADIA Lahcene is nominated to take responsibility for Project Tracking.

The following organogram resume this point.
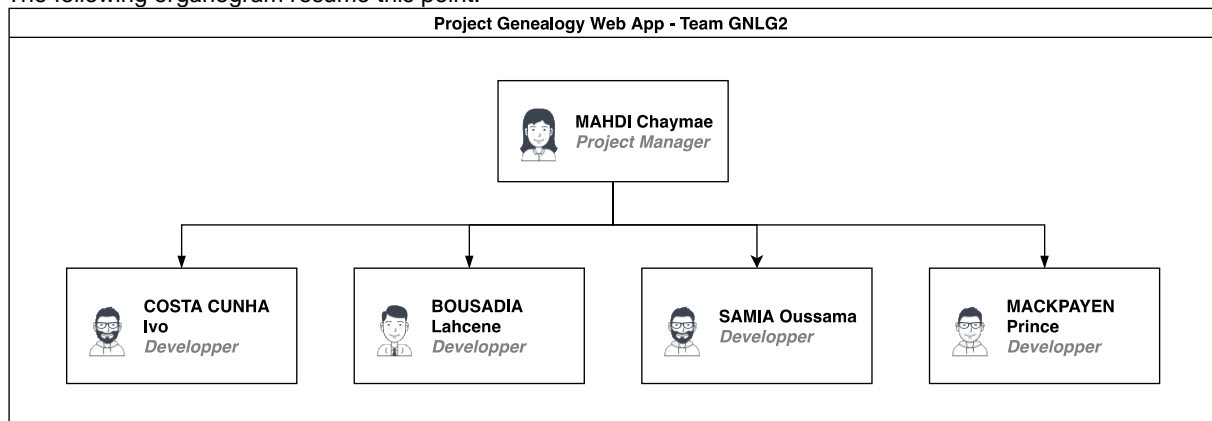
Figure 1 - Team flow chart

Contacts

Name	Mail
Ivo COSTA CUNHA	ivo.costa-cunha@etu.unice.fr
Oussama SAMIA	oussama.samia@etu.unice.fr
Lahcene BOUSADIA	lahcene.bousadia@etu.unice.fr
Chaymae MAHDI	chaymae.mahdi@etu.unice.fr
Prince Divin MACKPAYEN	divinmackpayen@gmail.com

2. Requirements

2.0. Specifications

2.0.1. Specified objective of the project

The client specifications are the following:
- ▪ "expoactes" is a system that allows to add and navigate in civil status acts. It is dated, in old PHP and with a user interface also dated. Upgrade "expoactes" in object-oriented PHP and update the user interface to a modern one. Why also not add visualizations, which is leads to a second part of the project.
- ▪ Using GEDCOM files (an export format that also allows the sharing of genealogy data), the client wants to obtain the following functionalities:
  - ▫ Obtain visualizations and navigations in the genealogy tree: on a map, a "traditional" tree, a "circular" tree (in fact they are graphs, an ancestor potentially being an ancestor for multiples "reasons").
  - ▫ Allowing the search of common ancestor by crossing 2 genealogies.
  - ▫ Have a social aspect by being notified of certain searches concerning the department, city name, patronymic, genealogist name "**(les / sont des et / ou)."**
  - ▫ "standalone" version and a WordPress integration are desired.

The back end of the application must be written in PHP7 using the object-oriented paradigm since it is a client requirement.

Current website: http://www.racinesardechoises.fr/.expoactes/index.php?vue=T&xtyp=N

2.0.2. Expoactes

Expoactes is gradually becoming a tool widely used to disseminate the analysis of old civil status documents and parish registers.
The user needs to click on the red icon to register himself on another page that is independent of the home page.
 We think those steps are confusing and lead to a lot of negative scenarios where the user leaves the website because he is unable to register himself or even know that he must register himself first to access the data. We aim to improve the UI in our modernised application in a way for it to be intuitive.

Figure 1 – Home page of expoactes

a. **Features of Expoact**es

Expoactes allows site visitors to:
- browse the tables by municipality, then in alphabetical order of surnames.
- search for acts by surnames of interested parties or other appearing.
- access (if you authorize it) the details of the acts.
- management of the number of acts that can be seen by your visitors in each period.

For the management of acts, you have access to an administrative part of the site protected by login and password.

The management of acts includes:
- loading of acts in NIMEGUE (Version 2 and 3) or CSV formats.
- the elimination of redundant acts and the reversal of marriage or promises acts.
- the re-export in NIMEGUE format of the documents you have submitted.
- management of access codes for visitors and depositors.
- the control of deposits and modifications of the database.

The main administrator can manage the access rights of visitors and depositors according to a 9-level hierarchy.

b. **Problems on the website**

To access the data a visitor must be register himself or herself and the administrator must validate this inscription, the following alternatives and negative scenarios show some problems:

- The user can get lost in the home page of the site because we do not really see how to be directed on the platform no bouton home page, which is not a good thing for any new user.
- The user cannot find a button to register on the site.
- The user cannot register on the site because he can't find a bouton for that.
- No login and password were given to us.
- We cannot see all the features of the site because we are not allowed only the administrator who can do this kind of features.

2.0.3. GEDCOM file

A GEDCOM file is a genealogical act saved in the format of genealogical data communications (GEDCOM) format. It contains family history acts and genealogical event data, as well as metadata linking the acts. GEDCOM files often contain information about births, deaths, marriages, children, and physical attributes of family members.

a. **Analysis of GEDCOM files**

The GEDCOM format was created to allow to share genealogical data in 1980 by *"L'Eglise de Jesus-Christ des saints des derniers jours"*. The objective was to give a solution to the exchange of data between 2 different genealogy software.

The GEDCOM file contains information about each person, each family, and each event of a tree.

Exporting or importing a GEDCOM file is a mandatory function of genealogy software.

We must be able to process an entire family tree, but also an ascending or descending branch from a selected person.

b. **GEDCOM File format**

To be able to analyse GEDCOM files, our team must first know that all the information in the GEDCOM file is described by keywords (e.g.: GIVN for first names, PLAC for location…) and linked together by references (e.g.: @ I…. @ For individual, @ F… @ for family) which allow genealogy software to reconstruct your tree from the file. So, this information can help us to make a program that is able to parse and read a GEDCOM file.

A GEDCOM file is divided into:

- A header section (HEAD).
- Various recordings of various kinds:

- "person" record (INDI = individual)
- "family" record (FAM = family)
- "note" record (NOTE = note)
- "source" record (SOUR = source)
- "Archive repository" record (REPO = repository)
- multimedia object" record (OBJE = object)
- end of file marker (TRLR = trailer)
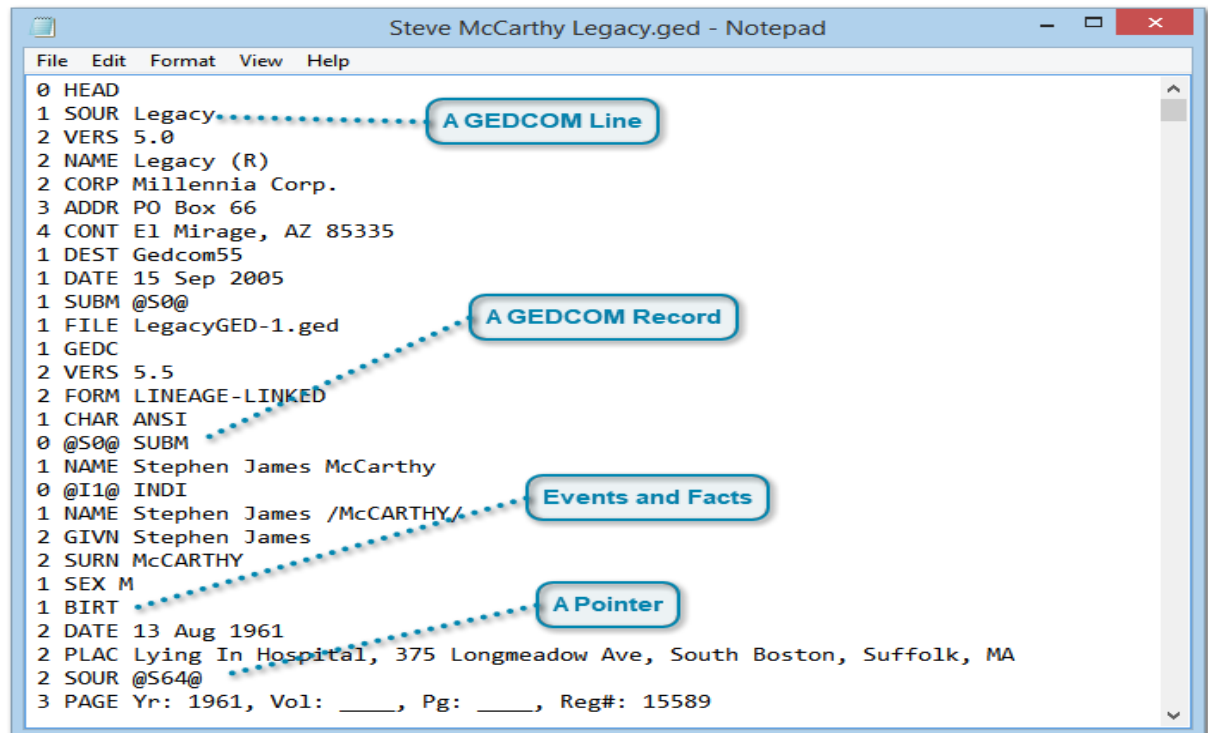
Figure 2 - GEDCOM file format

c. GEDCOM visualization

There is several software allowing to edit GEDCOM files, as an example, we can quote "geneatic" which works on version 7 of GEDCOM.
==> *https://garde-du-voeu.com/wp/exporter-un-gedcom-de-geneatique/*
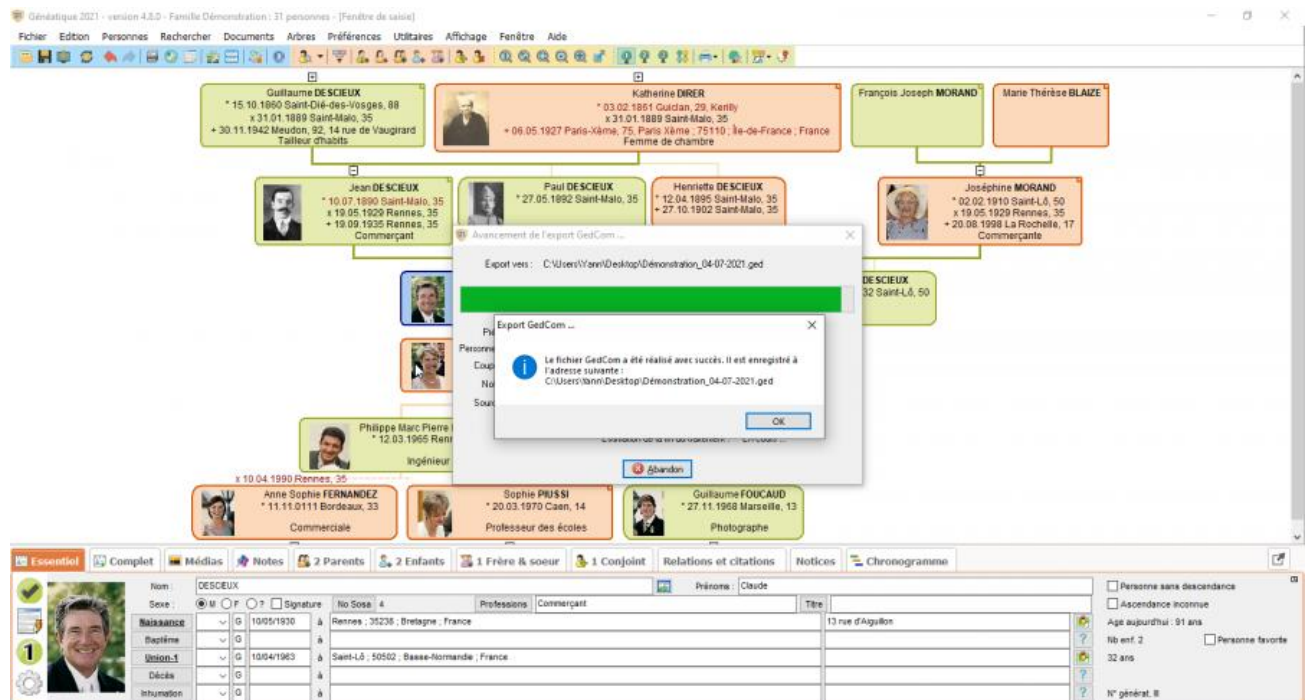
Figure 3 - GEDCOM tree format

2.1.4 Functional requirements

Functional requirements are those that must meet the requirements of the future system by terms of functionality. The identified needs are as following:

- The system must allow the user to visualize the different genealogical trees and navigate inside them either using a map visualization, a traditional tree or a circular tree.
- The user must be allowed to search a common ancestor by crossing 2 genealogies.
- The user must be allowed to find a potential cousinhood by crossing cities and names.
- The system must notify users about searches concerning department, city name, patronymic, or a genealogist name. "**(les / sont des et / ou)**"

2.1.5 Non-functional requirements

Non-functional requirements describe all the constraints to which the system is subjected for its realization and its proper functioning. These are the following:

- The system must be reliable.
- The UI must be ergonomic, user friendly and flexible.
- the system must be able to minimize the processing time in maximum 5s.
- The code must be visible, commented, and understandable to allow its evolution and extendibility.

2.1. Environment

To develop this web application our team will use the following software and technologies:

2.1.1. Software

- **GitHub:** It is the obvious choice to host our code in the cloud. It allows also allows versioning and the management of conflicts during a merge.
https://github.com/Master1-MIAGE-UCA/GNLG2

- **Balsamic:** It will allow us to create UI models for documentation and as a guideline to our developers.
https://balsamiq.com/wireframes/

- **Visual Studio Code:** This simple code editor shines in web and node development. Its functionalities extended by modules make it very adaptable to any project.
https://code.visualstudio.com

- **PhpStorm:** It's one of the best IDEs for PHP and that alone makes it an interesting choice for our project.
https://www.jetbrains.com/phpstorm/

- **Draw.io:** Easy to use, accessible in any navigator, web app that allows the creation of UML diagrams.
https://app.diagrams.net

- **Trello:** It will allow us to organize the project tasks and user stories, associate task with developers and follow what's done and what isn't at deadlines.
https://trello.com/

- **ArgoUML:** Another UML diagram editor.
https://sourceforge.net/projects/argouml/

2.1.2. Technologies

- **HTML/CSS/JavaScript:** The markup web language and its associated language CSS used to alter easily markups. It will affect the front-end of our application. JavaScript is the web scripting language today used extensively for a multitude of applications.

- **PHP7+:** Historical server-language and sometimes also used for the front-end that still updated. It will be used for back end in our application.
https://www.php.net

- **VueJS:** It's a framework of JavaScript that has simple to use UI design libraries that will be used for the front-end our application and that will allow us to develop a modern UI with the use of the Material UI library. Il also can be used to compile apps that can be uses as standalone apps via Electron as desired by the client.
https://vuejs.org

- **Bootstrap5:** A UI JavaScript library that can be used as standalone with vanilla JavaScript.
https://getbootstrap.com

- **Laravel:** Laravel is a PHP framework with expressive and elegant syntax. It allows you to quickly develop the different functionalities of a project and improves PHP syntax for readability.
https://laravel.com

2.3. Use cases

2.3.1. Actors

Administrator: The site administrator has administrative right so he can allow visitors to make accounts that allow to have extensive access to the web site / app functionalities.

Registered visitor: A person with an account that can upload GEDCOM files.

Visitor: A person that visit the site without account.

2.3.2. Scenarios

After identifying and describing the different actors of the system, we can now describe the different possible scenarios.

a. Log In

The various system actors must log in or have an account before accessing the system's functionalities.

b. Give access to visitors

The administrator must be able to give access to simple visitors to consult their information even if they have an account to connect, they must wait for the right of access according to the administrator of the application.

c. Send circular mails

Admin must be able to send circular emails, on another term he can send email to all group members. and he must have feedback if an email address no longer exists.

d. Change the status of a user

The administrator must be able to change the status of registered users so that he can for example change the access rights of a user from level 3 to a level still high of 4.

e. Block IP

When a user IP is banned by the admin it will be added to a list of banned IPs.

f. Remove from IP list

The administrator must be able to delete IPs from a list

g. Keep IPs

The administrator must be able to keep an IP in the list but set its status to unbanned.

h. Consult family trees

The administrator and the registered user must be able to consult the geological trees.

i. Research at tree level

The administrator and registered user must be able to search at tree level.
The administrator and the registered user can also do research by crossing two genealogical trees.

j. Import GEDCOM files

The administrator and registered user can upload new different GEDCOM files. Either of births, deaths, or marriage. For the user to upload new GEDCOM files the administrator must give to the registered user special permission, since in a normal case only the administrator can upload those. The registered user also become responsible for this GEDCOM file that he or she uploaded.

k. Get visualisations and navigations in family trees

The administrator and the registered user must be able to visualize the family trees either in card form, on traditional tree or on circular shaft.

2.3.3 Use case diagrams

The in-depth study of the specifications allowed us to identify several use cases. It allowed us to structure the needs of the users and the corresponding objectives of the system,
We represent, in the figure below, all the basic use cases to have a global view of how our application works, as well as any relationships that can take place.
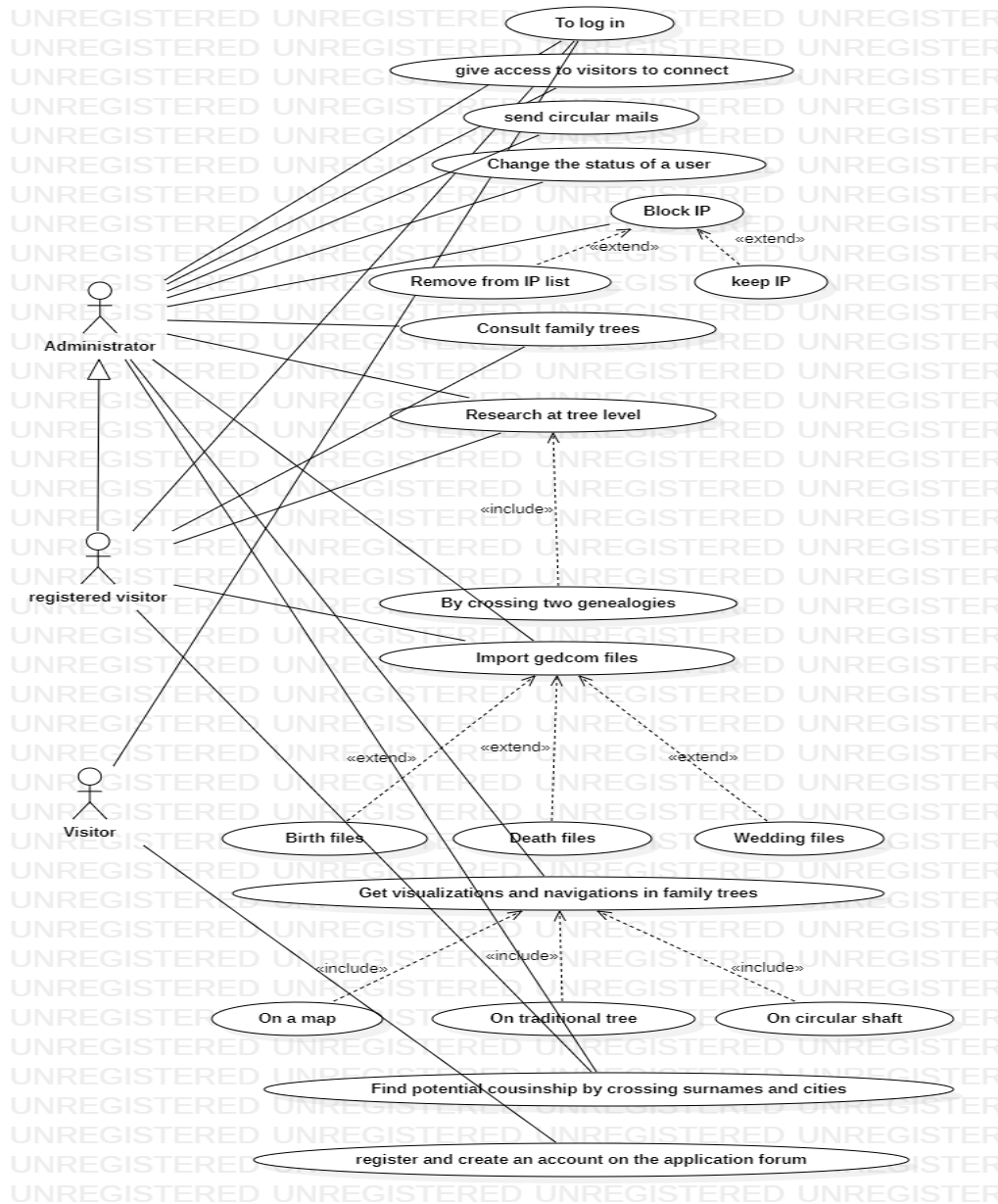
Figure 4 - Current system (Use Case)

In the use case diagram, we presented a relation between the administrator and the registered user by a generalization relation because we thought that in the case where the administrator is absent the registered user can replace him and make use of some of his exclusive functionalities while waiting for him or her to get back.

2.1.3. Current application architecture
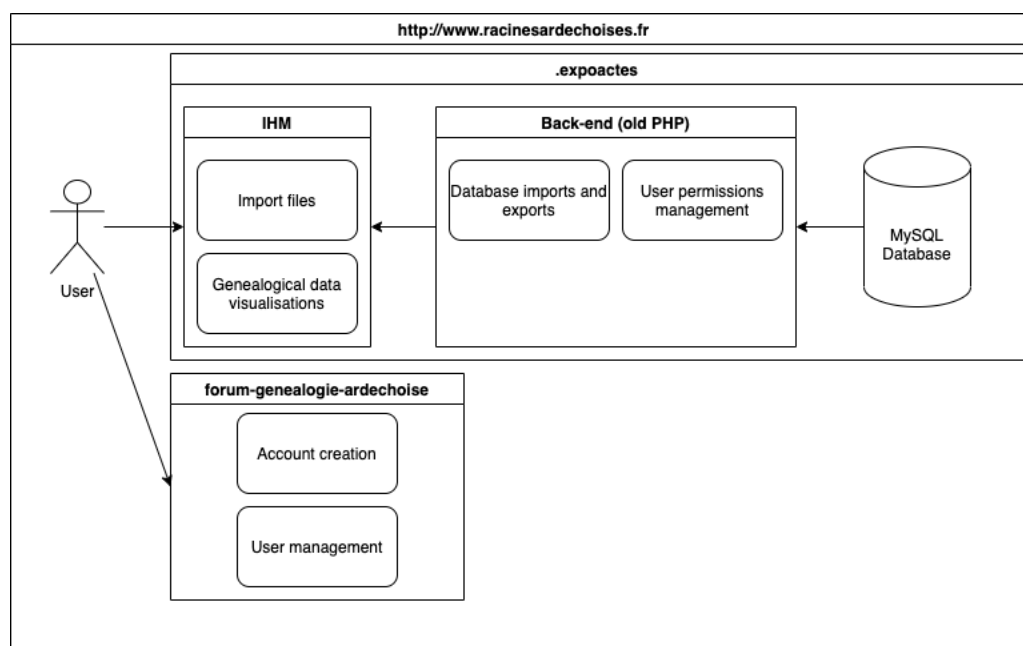
Figure 5 - Current application architecture

The current application architecture is an architecture made up of an application server and a database server. Between its two entities that make up the project, there is the PHP language which allows users to add metadata and query the database.

2.2. Requirement analysis

2.2.1. Objectives of our team

We will update the current application as requested by the client but starting from scratch and then rewriting a new code that will serve us to make the same concept of the expoactes system but in a new ergonomic and easy to use application, which will incorporate the functionalities of the old one, we will make sure that the new application will provide all the same functionalities, using the Laravel PHP framework to respect the PPO PHP code approach asked by the client.
We will however keep the old MySQL database intact to facilitate the migration of the old database data. If we need more tables, we will add them however we won't alter old tables.
Once that's done, we will add the asked features listed in 2.3.2.

2.2.2. Functions to be added

a. Cousinhood search

The cousinhood search should follow this scenario.

Find potential cousinship by crossing surnames and cities

Scenario as-is
1. The user seeks to find potentials cousinship by crossing surnames and cities
2. The use selects "cousinhood search"
3. The user selects a city
4. The user selects a name
5. The system search using the city and name as filter
6. The system find a match
7. The system show in the UI the results of the research

Extension 1
3.a The user enters an incorrect city name
3.b The systems show an error to the user when the user finish typing
Follow up on 3

Extension 2
4.a The user enters an incorrect name
4.b The systems show an error to the user when the user finish typing
Follow up on 4

Variant 1
6.a The system doesn't find a match
6.b The system displays in the UI that there is no match.
Follow up on 1

2.2.3. Access levels

The main administrator can manage the access rights of visitors and depositors according to a 9-level hierarchy. If necessary, a points mechanism makes it possible to limit the number of consultations of acts during a given period.

The 9-level hierarchy:

- Access to the list of communes and parishes
- Access to the list of patronymics (by commune/parish and act type)
- Access to the tables (names and dates)
- Access to details of acts
- Authorized to load data with NIMEGUE only
- Authorized to load data of CSV type
- Authorized to manage all data Administrator

The general policy is that any registered visitor has access up to level 4.

Creation and edition of user profiles:

A user profile is0 composed of:

- name
- surname
- mail address
- password (automatically created if wanted)
- login code
- access level
- status
  - N -> Normal
  - B -> Banned
  - + 2 others used during auto register

2.2.4. Import and export of acts by registered users

"expoactes" allows upload and export of acts by user if they are given the "holder" status of an act, otherwise only the administrators in the only one who is authorised to do it.

2.2.5. Other functions

"expoactes" also have function, for the administrator, to send circular emails, and if during the inscription the visitor choose to being send by mail his credentials to login later, he can choose.
The use case which is in the annex represents the different interactions possible between the system and the actors as well as between the actors.
It will help us to understand the functions and the roles of each actor of this system

3. Design

3.0. Design rationale

3.0.1. Team decisions concerning the problems met

a. New application

It was decided that our team won't use the old code except from inspiration. The old app code base is abandoned. A new Laravel project will allow us to create a better and up to current standards code. It will also enable us to create a code base that will permit other developer teams to be able to extend it easily.

b. Deadline problems

The subject of the project has several aspects to deal with as it is already mentioned in the previous paragraphs of requirement analysis, after having analysed the working time on the project, the time will not be enough for us to deal with all these aspects, we will do the big thing point of the project is to create a new expoactes system with the management of GEDCOM files, for the other requests which are desirable to do we will do our best to do it otherwise if we are able to do it that's a question problem of time, the desirable points and which we are not sure that we will do them are:
- Standalone App
- WordPress integration

c. Standalone App

We determined that that task would take alone at least 1,5 months to our team and so it's not possible to do it in the time given by the client to the project.

d. WordPress integration

While the team determined that this task would take scientifically less time than the standalone app it still prolongs the project past deadline.

e. Database restricted access

The team figured out that the new app require access in writing to the actual used MySQL database, however we only have the reading access. We need the client to give us writing permissions to complete the new application.

3.0.2. Features that we will be put in place in the new app

a. Website host

The current website uses a MySQL database and is written completely in PHP.
Since the project has as requirements the modernisation of the PHP code and to not change the database a host that allows both must be used.
OVH was proposed and seems to do both. https://www.ovhcloud.com/ in the cloud by OVHcloud.
In fact, OVH is dedicated to web hosting and Cloud Computing, therefore the dedicated OVHcloud server turns to the priority of security which allows us to have the opportunity to obtain protections against various attacks. It makes us to virtualize several projects and turn to specific ranges. The OVHcloud dedicated server can also guarantee the availability of our data without having to wait for a long load.

b. Creation and integration of an HTML template in a Laravel project

For the creation of our project, we decided to create a Laravel project to properly structure our code files and to integrate an HTML template into this project.
We decided to work on a Laravel project because it is an open-source web framework written in PHP respecting the model-view-controller principle which allows us to properly organize the structure of our project, in addition to that it's fully developed in object-oriented programming.
Laravel contains several object-oriented libraries which will facilitate the management of GEDCOM files, and which contains documentation that is clear and easy to understand.

c. Creating diagrams using ArgoUML

We modelled the diagrams of our project using the ArgoUML tool because it is free, multilingual software that supports code generation and reverse engineering, and especially that the generation of code from class diagrams is supported in PHP languages.

d. Creation of application mock-ups using balsamic software

Balsamic software is a software editor that has helped us to make UI design, that allows us to collaborate, create mock-ups, version control and run user tests.

The creation of the models of the application will serve us to see the real structure of the different pages of our applications and its functionalities and we decided to make these models to concretize the ideas of the project. which can then serve other secondary objectives: to explore an idea, to test it, or to convince stakeholders or its hierarchy of the concept.

e. familytree365/Laravel-GedCom

It is a Laravel compatible project that will permit us to analyse GEDCOM databases and import them as Laravel models. It allows, among other things, to use them and convert them back into other file formats.

3.1. Static view

3.1.1. New website architecture

The new web Laravel based application is composed of a front-end that includes the pages we previously listed build using Laravel tools, a back end using also Laravel that will include the back-end functionalities that will permit for the user to accomplish the tasks available in the IHM. While **we won't use the current web page code**, we will keep a MySQL database linked to our Laravel back-end since we must be able to use the old data without problems. Doing so makes things easier for use like the database migration.
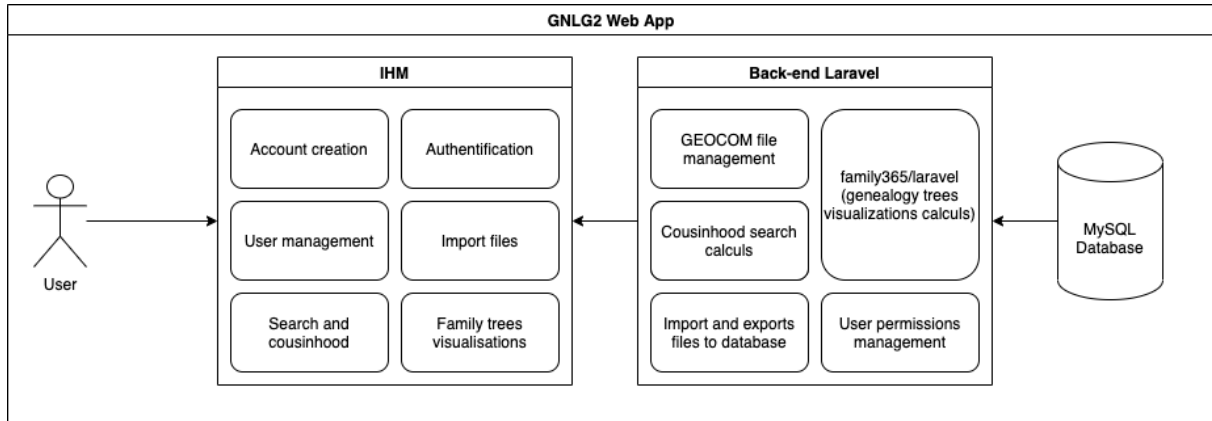

Figure 6 - GNLG2 Web App architecture

– **Control System HMI:** This part allows to build the Runtime environment, including the asset model structure and content, mimic definitions, and their associations to assets and It's composed of:

- **Account creation:** The user has the possibility of creating a new account on *expoact* by entering personal information (name, first name, address, date of birth, address, email, password) in order to see the desired genealogies.
- **Authentication:** Authentication requires two additional parameters, namely username and password. The username is checked to see if it is present in the database.
- **User management:** This part concerns everything related to user management such as adding a new user, modifying information, deleting an existing account User management enables admins to control user access and on-board and off-board users to and from IT resources.
- **Import files:** The user has the possibility of importing *gedcom* files on the platform in order to visualize the genealogy therefore this part is responsible for the users' imports.
- **Search and cousinhood:** The user have the possibility of importing *gedcom* files on the platform in order to visualize the genealogy therefore this part is responsible for the users' imports.
- **Family trees visualisations:** Obtain visualizations and navigations in the family tree: on a *map*, in a *traditional* tree, in a *circular* tree (in fact, these are graphs, an ancestor can be an ancestor for several "reasons").
- **Back-end laravel:** In this part we will work with laravel family 365 tree that allow easily create the own family tree by importing existing data or manual data entry. Storage of all data is securely on your own server and does not leave your environment

3.1.2. UI models

We plan to develop the following app pages for the new website:
- Authentication / Registering
- Main
- Add files to database
- Default viewing
- Viewing page (traditional form)
- Viewing page (circular form)
- Research by crossing 2 genealogies
- Research potential cousinage
- Notification page

a. Authentication page

authentication page

If you don't have an account click here :   Registration

Log In

you must identify yourself

User name :

Password :

Log In   cancel

consult the conditions of access on the private site

Figure 7 - Authentification page UI model

b. Registering page

Registration page

If you have an account click here :   Log In

Register

First name :

Last name :

User name :

Email :

Password :

Register   cancel

consult the conditions of access on the private site

Figure 8 - Registration page UI model

c. Main page
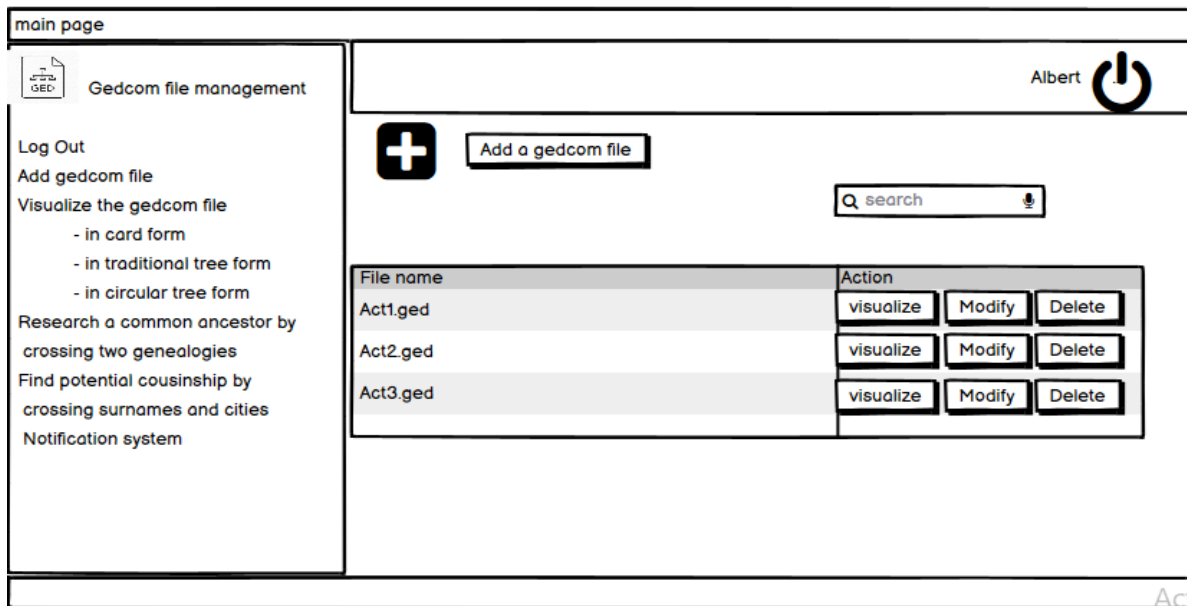
Figure 9 - Main page UI model

d. Add files to database page
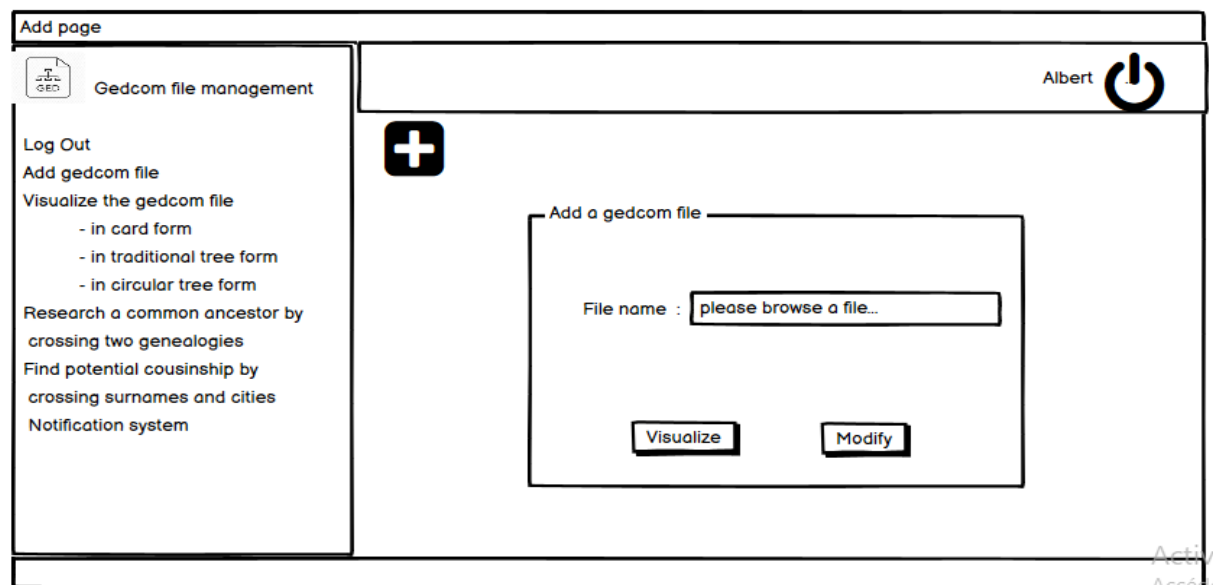
Figure 10 - Add page UI model

e. Default viewing page
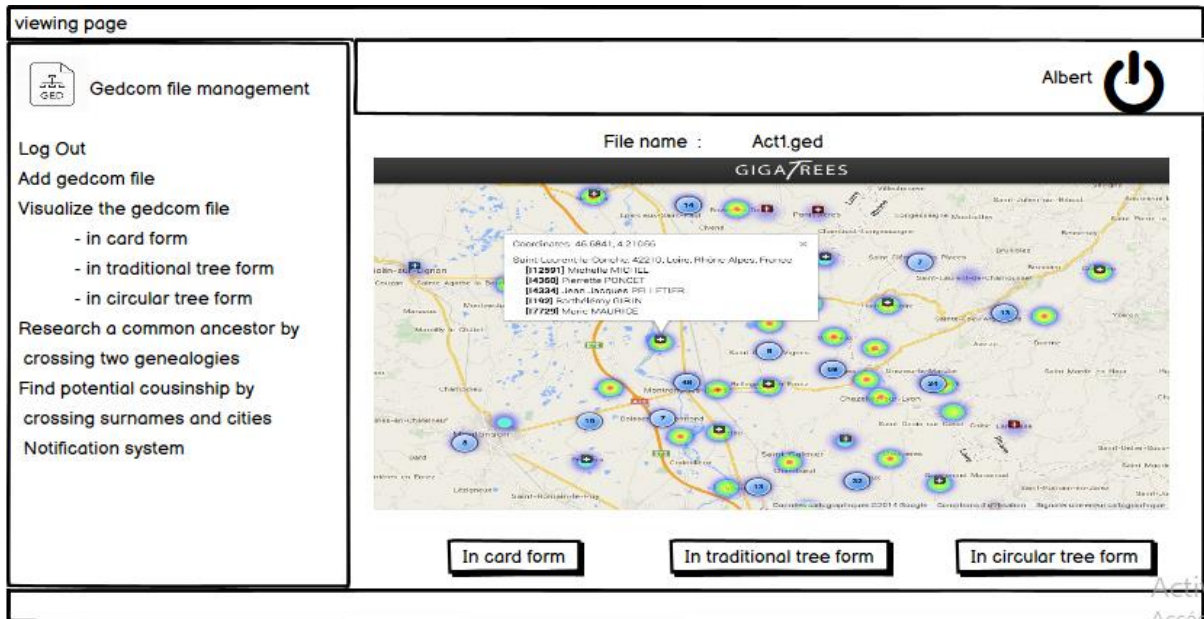
Figure 11 - Default viewing page UI model

f. Viewing page (traditional form)
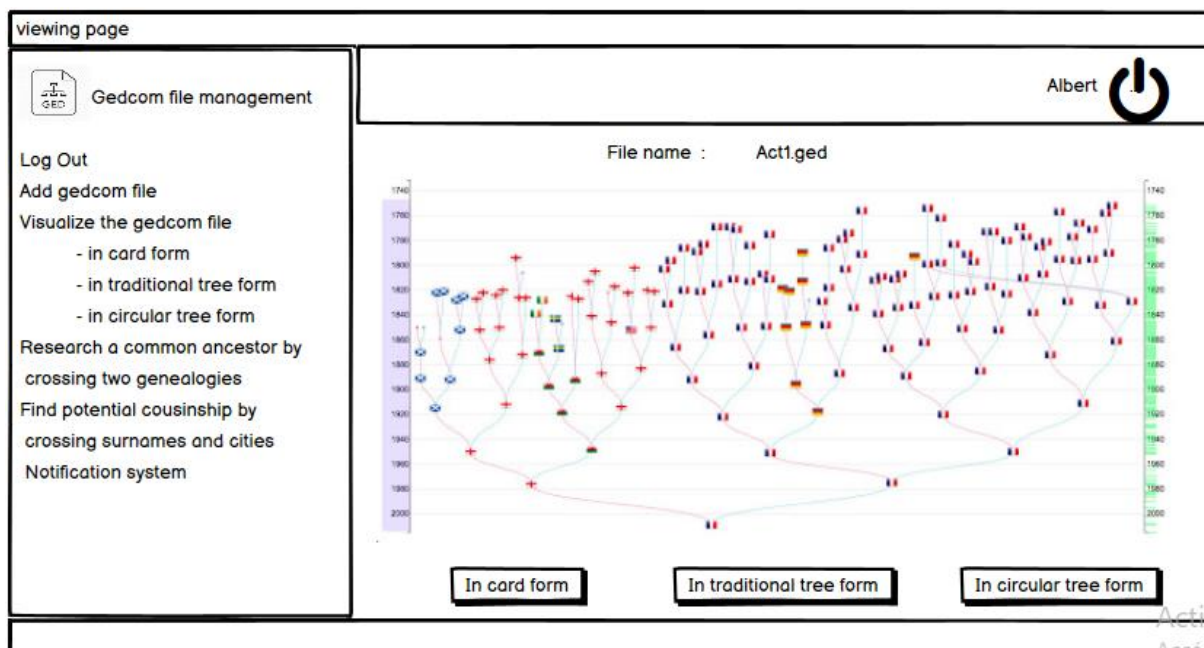
Figure 12 - Viewing page (traditional form) UI model

g. Viewing page (circular form)
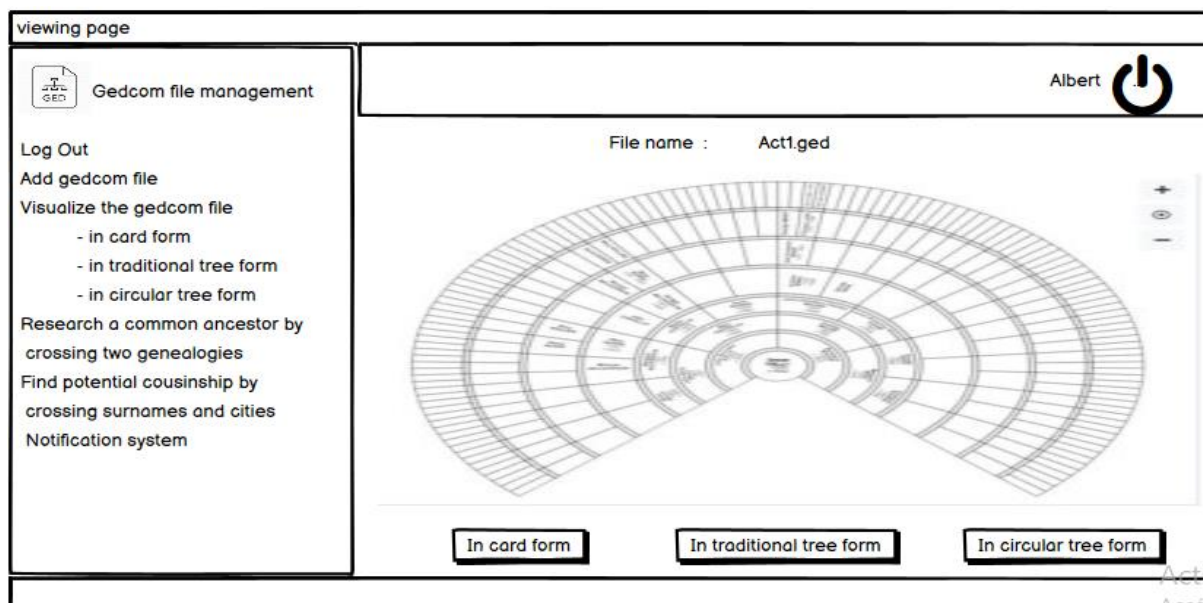
Figure 13 - Viewing page (circular form) UI model

h. Research by crossing 2 genealogies
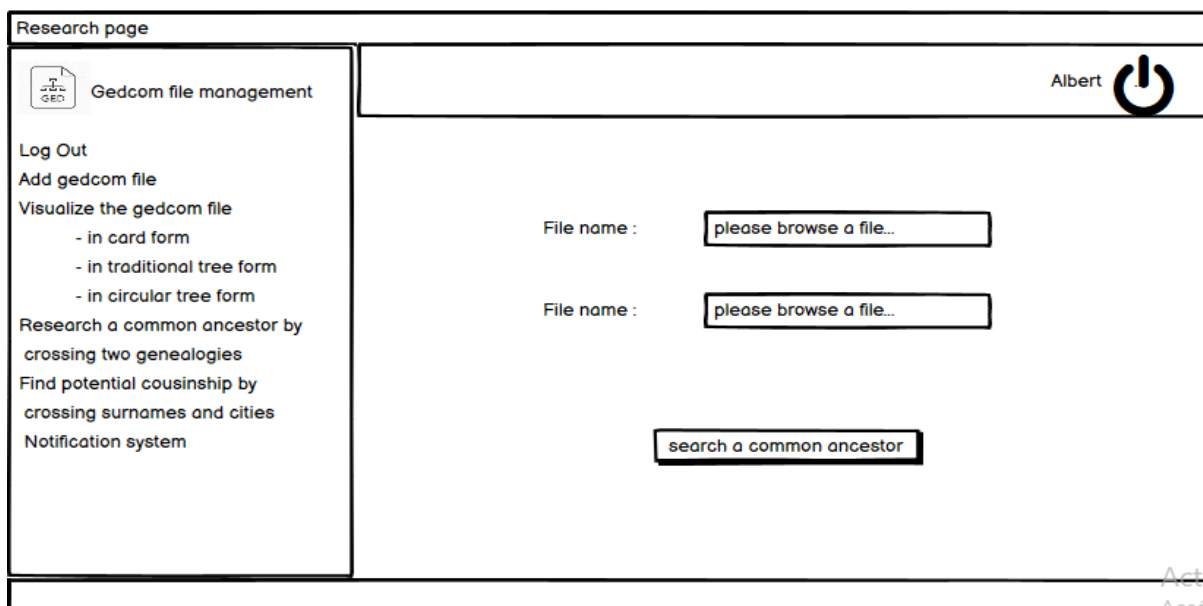
Figure 14 - Research by crossing 2 genealogies UI model

i. Research potential cousinhood

Figure 15 - Research potential cousinhood UI model

j. Notification page

Figure 16 - Notification page UI model

3.1.3. Processing GEDCOM files

For that, we decided to create an application that allows the management and processing of GEDCOM files, which we must be able to:

- Obtain visualizations and navigations in the family tree: on a map, in a "traditional" tree and in a "circular" tree.
- Allow the search for a common ancestor by crossing 2 genealogies.
- Find potential cousinship by crossing surnames and cities.
- The web app must have a social aspect be informed of certain searches concerning department / city name / by patronymic / name of the researcher (genealogist).

In this part we will define the different tasks to be carried out for the second aspect of the project which are as follows:

- Research the handling of GEDCOM files

- Make app mock-ups
- Make a class diagram for this application.
- Creation of a Laravel project and integration of an HTML template in there.
- Creation of a BD where we will store the GEDCOM files selected by the user
- Learn how to read GEDCOM files and understand how to retrieve information from annotations used like @Fam ...
- To learn how to navigate in the different family trees.
- Make and implement algorithms for the visualization of family trees in the form of a map
- Make and implement algorithms for the visualization of family trees in the form of a circular tree
- Make and implement algorithms for the visualization of family trees in the form of a traditional tree
- Make known and implement how to cross two genealogies
- Research a common ancestor by crossing two genealogies
- Find potential cousinship by crossing surnames and cities
- Be notified of certain searches concerning department / city name / by patronymic / researcher's name.

3.1.4. The use of laravel framework

Laravel is a cross-platform PHP framework for building web applications. Laravel allows a developer to take advantage of a large library of pre-programmed features (such as authentication, routing, and HTML template creation). Access to this library simplifies the rapid creation of robust web applications while minimizing the amount of code required.

Laravel offers a very functional development environment, as well as intuitive and expressive command line interfaces. Additionally, Laravel uses Object Relational Mapping (ORM) to simplify data access and manipulation.

In addition to that, Laravel is primarily a Backend development framework, although it does offer some Frontend functionality. However, many features of Laravel are Frontend-agnostic. It follows the Model-View-Controller (MVC) architecture. which will allow us to organize our work and help us to clarify our code.

3.1.5. Database tables

We will import the old database structures and data as it is, and **we won't alter it,** which facilitates the migration of the database from the old application to the new application which represents an advantage for us. If we need at any moment to add tables, we will add them without altering the old ones.

act_metadb
- ZID INT
- dtable VARCHAR(1)
- zone VARCHAR(15)
- groupe VARCHAR(3)
- bloc TINYINT(1)
- typ VARCHAR(3)
- taille INT
- OV2 INT
- OV3 INT
- oblig CHAR(1)
- affich CHAR(1)
- Indexes

act_sums
- COMMUNE VARCHAR(40)
- DEPART VARCHAR(40)
- TYPACT CHAR(1)
- LIBELLE VARCHAR(50)
- DEPOSANT INT
- DTDEPOT DATE
- AN_MIN INT
- AN_MAX INT
- NB_TOT INT
- NB_N_NUL INT
- NB_FIL INT
- DER_MAJ DATETIME
- Indexes

act_div3
- BIDON VARCHAR(10)
- CODCOM VARCHAR(12)
- COMMUNE VARCHAR(40)
- CODDEP VARCHAR(10)
- DEPART VARCHAR(40)
- TYPACT VARCHAR(1)
- DATETXT VARCHAR(10)
- DREPUB VARCHAR(25)
- COTE VARCHAR(40)
- LIBRE VARCHAR(50)
- SIGLE VARCHAR(5)
- LIBELLE VARCHAR(50)
- NOM VARCHAR(30)
- PRE VARCHAR(35)
- SEXE CHAR(1)
- ORI VARCHAR(35)
- DNAIS VARCHAR(10)
- AGE VARCHAR(15)
- COM VARCHAR(70)
- PRO VARCHAR(35)
- EXCON VARCHAR(50)
- EXC_PRE VARCHAR(35)
- EXC_COM VARCHAR(70)
- P_NOM VARCHAR(30)
- P_PRE VARCHAR(35)
- P_COM VARCHAR(70)
- P_PRO VARCHAR(35)
- M_NOM VARCHAR(30)
- M_PRE VARCHAR(35)
- M_COM VARCHAR(70)
- 43 more...
- Indexes

act_dec3
- BIDON VARCHAR(10)
- CODCOM VARCHAR(12)
- COMMUNE VARCHAR(40)
- CODDEP VARCHAR(10)
- DEPART VARCHAR(40)
- TYPACT VARCHAR(1)
- DATETXT VARCHAR(10)
- DREPUB VARCHAR(25)
- COTE VARCHAR(40)
- LIBRE VARCHAR(50)
- NOM VARCHAR(30)
- PRE VARCHAR(35)
- ORI VARCHAR(35)
- DNAIS VARCHAR(10)
- SEXE CHAR(1)
- AGE VARCHAR(15)
- COM VARCHAR(70)
- PRO VARCHAR(35)
- C_NOM VARCHAR(30)
- C_PRE VARCHAR(35)
- C_COM VARCHAR(70)
- C_PRO VARCHAR(35)
- P_NOM VARCHAR(30)
- P_PRE VARCHAR(35)
- P_COM VARCHAR(70)
- P_PRO VARCHAR(35)
- M_NOM VARCHAR(30)
- M_PRE VARCHAR(35)
- M_COM VARCHAR(70)
- M_PRO VARCHAR(35)
- 17 more...
- Indexes

act_traceip
- id INT
- ua VARCHAR(255)
- ip VARCHAR(50)
- login VARCHAR(15)
- datetime INT
- cpt INT
- locked SMALLINT
- Indexes

act_log
- user INT
- date DATETIME
- action VARCHAR(40)
- commune VARCHAR(30)
- nb_actes INT
- Indexes

act_mar3
- BIDON VARCHAR(10)
- CODCOM VARCHAR(12)
- COMMUNE VARCHAR(40)
- CODDEP VARCHAR(10)
- DEPART VARCHAR(40)
- TYPACT VARCHAR(1)
- DATETXT VARCHAR(10)
- DREPUB VARCHAR(25)
- COTE VARCHAR(40)
- LIBRE VARCHAR(50)
- NOM VARCHAR(30)
- PRE VARCHAR(35)
- ORI VARCHAR(35)
- DNAIS VARCHAR(10)
- AGE VARCHAR(15)
- COM VARCHAR(70)
- PRO VARCHAR(35)
- EXCON VARCHAR(50)
- EXC_PRE VARCHAR(35)
- EXC_COM VARCHAR(70)
- P_NOM VARCHAR(30)
- P_PRE VARCHAR(35)
- P_COM VARCHAR(70)
- P_PRO VARCHAR(35)
- M_NOM VARCHAR(30)
- M_PRE VARCHAR(35)
- M_COM VARCHAR(70)
- M_PRO VARCHAR(35)
- C_NOM VARCHAR(30)
- C_PRE VARCHAR(35)
- 39 more...
- Indexes

act_nai3
- BIDON VARCHAR(10)
- CODCOM VARCHAR(12)
- COMMUNE VARCHAR(40)
- CODDEP VARCHAR(10)
- DEPART VARCHAR(40)
- TYPACT VARCHAR(1)
- DATETXT VARCHAR(10)
- DREPUB VARCHAR(25)
- COTE VARCHAR(40)
- LIBRE VARCHAR(50)
- NOM VARCHAR(30)
- PRE VARCHAR(35)
- SEXE CHAR(1)
- COM VARCHAR(70)
- P_NOM VARCHAR(30)
- P_PRE VARCHAR(35)
- P_COM VARCHAR(70)
- P_PRO VARCHAR(35)
- M_NOM VARCHAR(30)
- M_PRE VARCHAR(35)
- M_COM VARCHAR(70)
- M_PRO VARCHAR(35)
- T1_NOM VARCHAR(30)
- T1_PRE VARCHAR(35)
- T1_COM VARCHAR(70)
- T2_NOM VARCHAR(30)
- T2_PRE VARCHAR(35)
- T2_COM VARCHAR(70)
- COMGEN TEXT
- IDNIM INT
- 9 more...
- Indexes

act_metalg
- ZID INT
- lg VARCHAR(3)
- etiq VARCHAR(50)
- aide VARCHAR(500)
- Indexes

act_params
- param VARCHAR(25)
- groupe VARCHAR(30)
- ordre INT
- type CHAR(1)
- valeur TEXT
- listval VARCHAR(255)
- libelle VARCHAR(250)
- aide TEXT
- Indexes

act_user3
- login VARCHAR(15)
- hashpass VARCHAR(40)
- nom VARCHAR(30)
- prenom VARCHAR(30)
- email VARCHAR(50)
- level INT
- regime INT
- solde INT
- maj_solde DATE
- statut VARCHAR(1)
- dtcreation DATE
- dtexpiration DATE
- pt_conso INT
- REM VARCHAR(50)
- libre VARCHAR(100)
- ID INT
- Indexes

act_mgrplg
- grp CHAR(2)
- dtable CHAR(1)
- lg CHAR(2)
- sigle VARCHAR(5)
- getiq VARCHAR(30)

act_prenom
- prenom CHAR(30)
- Indexes

Figure 17 - Old system database structure

The database of the expoact system is a large database which contains 13 tables, each table has several fields to process for example the table of act_div which represents the table of divorce certificates contains 73 fields to process.

On the other hand, the expoact system database tables are entirely separate, in other words, there is no relation or sharing of foreign keys between them.

3.2. Dynamic view

After having decided to implement the new functionalities in an application independent of the expoact system, we must show the different interactions between the components of the system and for that we have produced a sequence diagram that will allow us to know how the different components interact between them.

Here is an example of a sequence diagram that shows some system functionalities such as: registration authentication, adding a GEDCOM file and viewing it on the map:
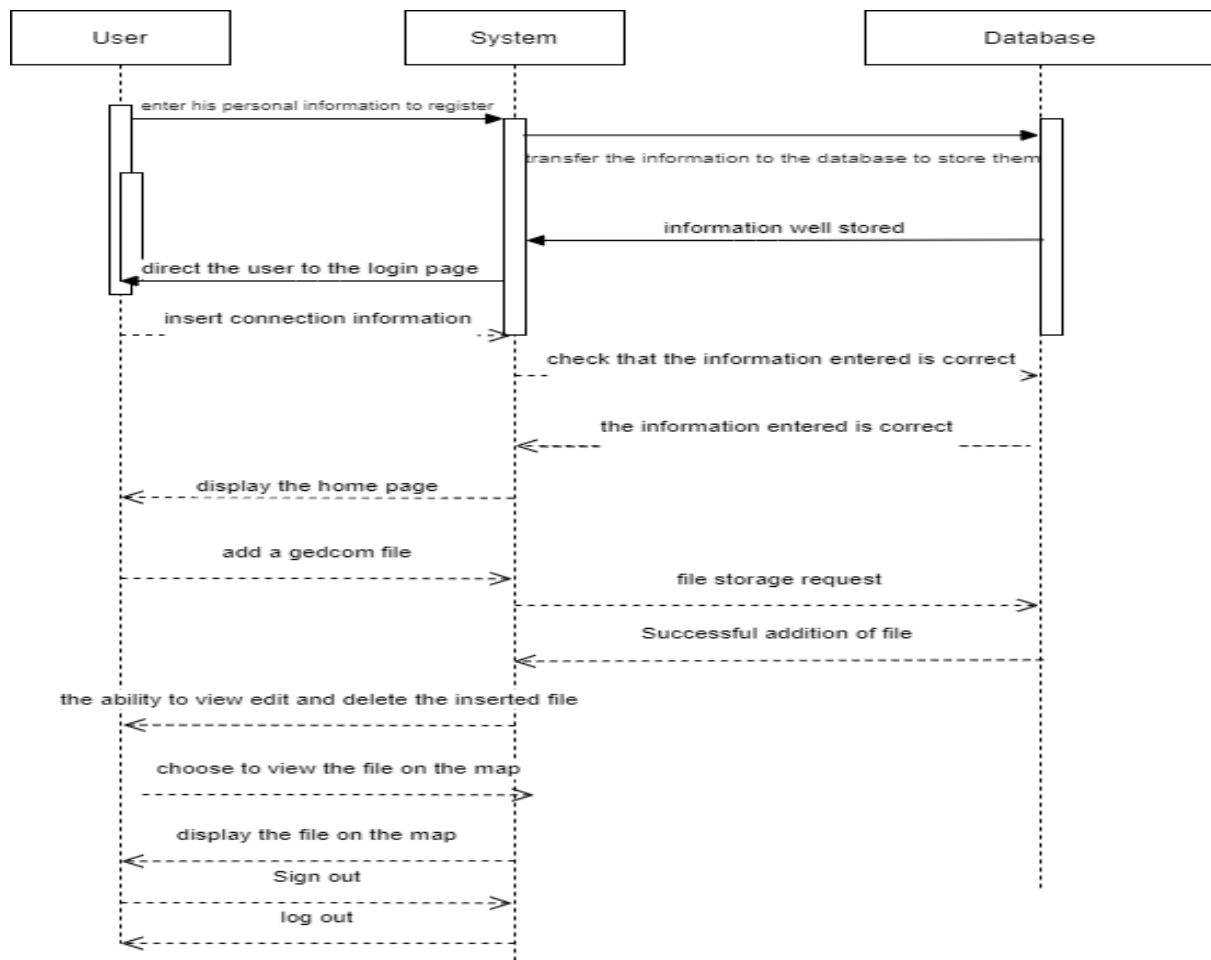

Figure 18 - System usage sequence diagram

3.3. Development planning

3.3.1. Task IDs

For each task type we assign an ID. They will also correspond to project tracking ones.
- GM: General management
- GM.MET: Team meetings
- WR: Workbook Redaction
- PT: Project tracking
- PC: Project conception
- PC.APP: App conception
- DEV: Development
- DEV.DB: Database tasks
- L: Learning
- L.GF: Learning GEDCOM files
- L.DB: Learning DB structure
- DEV.VIS: Make and implement algorithms for the navigations and visualizations of family trees.
- DEV.RES1: Research a common ancestor by crossing two genealogies.
- DEV.RES2: Find potential cousinship by crossing surnames and cities.
- DEV.NOTIF: Be notified of certain searches.

3.3.2. Team IDs

Each member of the team has an ID made from her or his name initials.
- ICC: Ivo COSTA CUNHA
- CM: Chaymae MAHDI
- PDM: Prince Divin MACKPAYEN
- LB: Lahcene BOUSADIA
- OS: Oussama SAMI

3.3.3. Task assignment and estimated times

Id	Task description	Time Estimate (in m.d)	Task performed (in m.d)	Remains to be done (in m.d)	Actors
WR	Writing the workbook	5	2	2	CM + ICC
PT	Make the follow-up file (Project tracking)	1	1	1	CM + ICC
L.GF	Research the handling of GEDCOM Files and understand how to retrieve information from annotations	5	2	2	PDM
CD	Make a class diagram for this web application	1	0.5	1	ICC
CP	Creation of a Laravel project and integration of an HTML template in there	5	4	4	OS
DEV.DB	Creation of a DB where we will store the GEDCOM files selected by the user	1	2	1	OS
PC.APP	Make app mock-ups	2	1	0	CM
DEV	Development of application functionalities	5	2	7	CM ICC OS PDM LB
GM.MET	Team meetings	3	1	2	CM ICC OS PDM LB

4. Engineering

4.0. Software organization

4.0.1. Git repository organisation

Our git repository is organised in 3 parts:
- "GNLG2" contains:
- Build scripts
- Other projects directories:
  - "db" directory: MySQL creation table scripts and MySQL insertion scrips
  - "laravel" directory: the source code of the application
  - "project_management" directory: project managements documents

4.0.2. Classes to be added

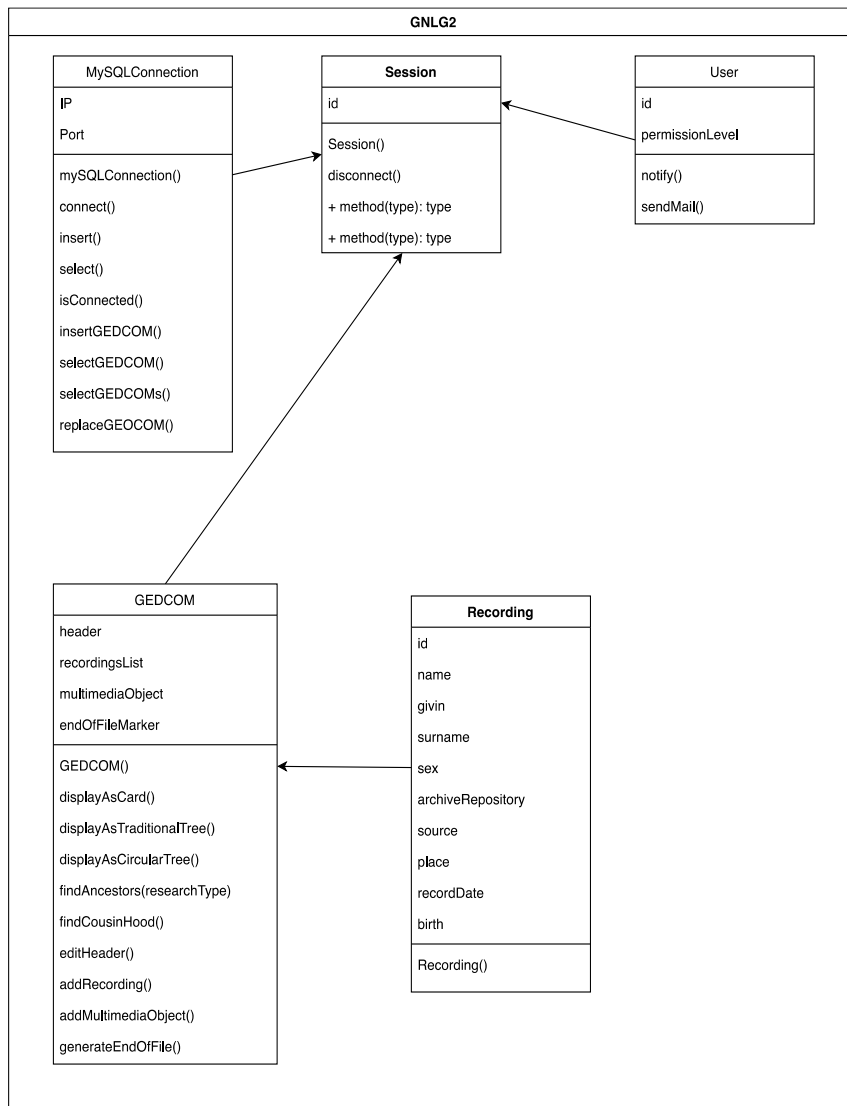Since we aim to use object-oriented programming paradigm a class diagram must be done.

Figure 19 - class diagram to be implemented

4.1. Build process

4.1.1. Prerequisites

- **Prerequisite's list**

The build process has some mandatory software requirements:
- PHP@8+ (Laravel requirement)
- Composer
- MySQL@8.0.27
- NodeJS

- **PHP 8+**

Not only all is written in PHP in this project but also, we use PHP based packet managers.

- **Windows**

To install PHP 8+ on windows just follow the instructions on https://www.apachefriends.org/download.html and install XAMPP.

- **macOS**

If you are on Mac first install Brew with the following command on any terminal.
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Then install PHP via Brew.
$ brew install PHP

- **MySQL set-up**

The application requires a permanent connexion to a MySQL database, the database can either be local on hosted in a distant server. Distant MySQL database doesn't require a local installation, however if you want to use a local database follow the following instructions.

If you are on windows XAMPP installs MySQL with PHP however you can also download the latest .msi file on https://dev.mysql.com/downloads/installer/ and launch it once downloaded.

If you are on macOS just use the following terminal command.
$ brew install mysql

- **NodeJS**

Laravel requires some NPM packets so NodeJS in necessary.
Follow instructions from https://nodejs.org/en/download/ to install NodeJS.

- **Composer**

Composer will be installed via the scripts in the Build process section.

4.1.2. Build process

To have a good functioning of the site, there are two scripts in our git repository to choose for the execution and it all depends on the version you have on your machine.
- Firstly, make sure MySQL local server is running if you have a local one.
- Launch the script *« https://install_mac.sh/ »* or windows_*«install_windows.bat»* depending on your OS always as administrator. This will start a web server. Then open a navigator and go to *http://localhost:8000/* and the app will be available to you to use.

the way of installing and executing the code of the application depends on the operating system of your machine, we explain the latter for the two most used operating systems which are windows and macOs :

- **Windows:**

For this type of operating system, it would be necessary to launch the script *«install_windows.bat»* which allows to import the database which includes all the existing acts as well as to install some as composer, which is a free dependency manager written in PHP.

It allows its users to declare and install the libraries in a project so not to have problems with the installation of the libraries **--> « php composer-setup.php ».**

Once the installation of composer is finished the script: "echo "generating and importing sql files to laravel MySQL database …
--> « mysql -u %db.user% --password=%db_pass% -e »
--> "CREATE DATABASE %db_name%." ...

After composer and the database, the script starts to set the file .env and install NPM as well as generate the key for the .env file.
cp .env.example .env
composer install
npm install
php artisan key:generate

Finally, the script launches the command **« php artisan serve »** which allows to launch the server on localhost and browse the site.

- **MacOS:**
This operating system requires the execution of the file **« https://install_mac.sh/ »** after the script will do the same as type windows so it will install what we mentioned before and finally it will directly launch the server on localhost.

4.2. Tests

We will use the build in Laravel unit tests for the method tests.
Laravel uses **PHPUnit** which represents a framework which is installed as a dependency of Laravel in development mode to perform unit tests. In addition to the PHPUnit methods, we have **helpers** to integrate the tests into our application made with Laravel.

This framework is installed as a dependency of Laravel in development mode:

```
"require-dev": {
    ...
    "phpunit/phpunit": "^9.3"
},
```

To check that it works, we run the following command:

```
php phpunit(numéro de version).phar -h
```

To run the tests, we do:

```
php artisan test
```

As we said before, the tests are carried out in a particular environment, which comes in handy. the phpunit.xml file contains the configuration which we find environment variables like APP_ENV, BCRYPT_ROUNDS...
We can obviously add the variables we need. For example, if during the tests I want to use MySql:

```
DB_CONNECTION=mysql
```

To build a test, we generally proceed in three steps:
- we initialize the data.
- we act on these data.
- we check that the result is in line with our expectations.

When we run the test, we get here:

```
OK (1 test, 5 assertions)
```

The test files are located in the tests folder located at the root of the Laravel project. But before we get there we will have to set up our environment for testing. This setting is carried out either in the phpunit.xml file which is in the public folder or by creating an environment file for testing directly at the root of the project by naming the .env.testing file which overrides the existing .env file as we did :

```
APP_ENV=testing
APP_DEBUG=true
APP_URL=http://127.0.0.1:8000/

BCRYPT_ROUNDS=4
CACHE_DRIVER=array
SESSION_DRIVER=array
QUEUE_DRIVER=sync
MAIL_DRIVER=array
```

5. Deployment

5.0. Deliverables

The project deliverables present the different products and services given to the client, in our case it is the completion or outcome of the entire project.
our deliverables are:

- **WorkBook**: document which summarizes your needs within the framework of an IT project. It describes, in detail, all the functionalities requested, and the corresponding technical specifications.

- **Project planning**: in the form of an excel file which allows to show the time management of the project and how we have distributed the phases of the project so that we can respect the time of the project.

- **Application code**: the application development code that is in our git repository, The directory GNLG2/laravel directory is the deliverable, we are going to give a zip format file to the client which will bring together all the application files. Since we work with PHP, we don't compile any code and the sources are the deliverables.

- **Installation scripts**: contains instructions that guide the client to launch and run the application.

- **Application database:** we will give the client the database of the application which contains the same tables of the database of the old expoact system as this we respect the client condition which is the prevention database change.

5.1. Installation

5.1.1. Local method

Make sure MySQL local server is running if you have a local one.

Launch the script mac_start.sh or linux_start.sh or windows_start.bat depending on your OS always as administrator. This will start a web server.
Then open a navigator and go to http://localhost:8000 and the app will be available to you to use.

5.1.2. Installed libraries

During this project we installed several libraries to make the site work properly and without errors:

- Authentication for a web application is a process that allows the application to ensure the legitimacy of the access request made by an entity (a user, a client or a system, ...) to authorize the access of this entity to resources of the application.

  To develop our web application faster, the Laravel PHP framework offers kits like Breeze to set up the routes, controllers, views, and other resources that are needed to register and authenticate the users of the application.

  laravel/breeze is a package that implements the authentication features of Laravel. It sets up roads, controllers, and views to:

- Register a new user with a name, email address, and password
- Connect a user with an email address + password
- Recover a user's password
- Verify a user's email address
- Disconnect a user

- For the import of files of birth or marriage certificates in Excel format, Laravel offers us the **maatwebsite/excel** library is a package that facilitates the task of importing and exporting the files and among these features we find:

  - Easily export collections to Excel.
  - Export queries with automatic chunking for better performance.
  - Queue exports for better performance.
  - Easily export Blade views to Excel. Easily import to collections.
  - Read the Excel file in chunks.
  - Handle the import inserts in batches.

5.2. Operations

5.2.1. Authentication:

To connect to the website, you must first type in your username and password and if the password is wrong the user will receive an error message to repeat it correctly.

5.2.2. Management of acts:

Once connected to the site, there is a tab for managing acts or this function allows you to load acts in CVS or Excel form, the ExpoActes program automatically recognizes the version of the export file that is present to it. In this part, the user can add a new birth certificate by inserting the information of the person (Name, first name, date of birth ...) as well as cells of his parents, witnesses, reference, etc ...,
He can also delete an existing birth certificate from the database by clicking on the delete birth certificate button. All the acts that exist is displayed in the form of a table or on sees well the cities and the periods can select a city to browse the people of this city with the possibility of modifying or deleting the acts

5.2.3. User management:

This tab is responsible for displaying and listing all users who have access to the site, seeing their status names (access if it is allowed or not) with the access level, email, with an icon for actions if the administrator wishes to delete or modify the information of each user.

You can also add a new user just by clicking on the add user button, user can load acts in CVS or Excel form, the ExpoActes program automatically recognizes the version of the export file presented to him.

In general, count tables have in the first row the names of the columns specifying the type of information that can be stored in them. For example, Last Name, First Name, Gender, Father's Name, Mother's Name, ... Then in the following lines are the stripped-down information, one act per line. You should start by saving the sheet in CSV format, taking care to choose the semicolon or tab to separate the fields. As much as possible, it is better to have the fields framed by quotation marks. This is not required unless the encoded texts themselves have semicolons.

Once the import is made a window is displayed to inform the user that the addition has passed, this large battery of functions of (pre-)processing of the information should facilitate the conversion of many data without the need for any manipulation of the spreadsheet

Each act is associated with the code of the person who uploaded that act to the database (the applicant). Navigation in the documents in administration mode makes it possible to display the name of the depositor of the document. In addition, when these are documents that you have filed, the system adds icons to modify or delete the act:

  - The Edit icon is used to correct the data of an act.
  - The Delete icon allows you to permanently delete an act (the system asks for confirmation but then, there is no way to find the act).

5.2.4. Software administration:

Data administration makes it possible to carry out some fewer common tasks but is useful for the proper management of the Expoacts database. One of these actions consists in making backup copies of the data or "backup" and sometimes in the restoration of backed-up data.

This task is accessible only to users who have strong access to the site, to facilitate the configuration, the settings have been grouped into several themes: Display, System, Users ... All these settings can also be saved or restored as needed (especially in case of migration of the database from one host to another).

5.2.5. Help tab:

The help tab that exists on the site allows you to guide the user to the different features of the site as he can guide you to manage his birth certificates or to modify and even delete them... You will have information on the operation of the site and each tab such as software administration so the user can go to the old ExpoActs site to see the FAQ if he has any questions or any problem about the progress and features.

6. Annexe

In this part we will show some screens of the future expoact system:
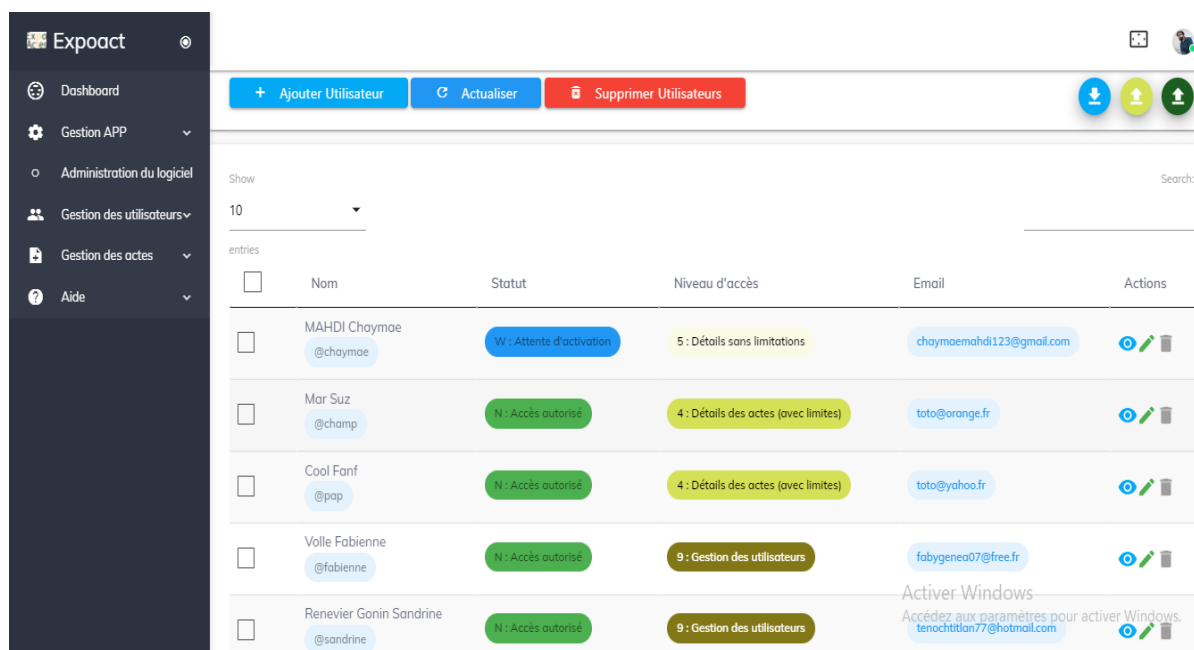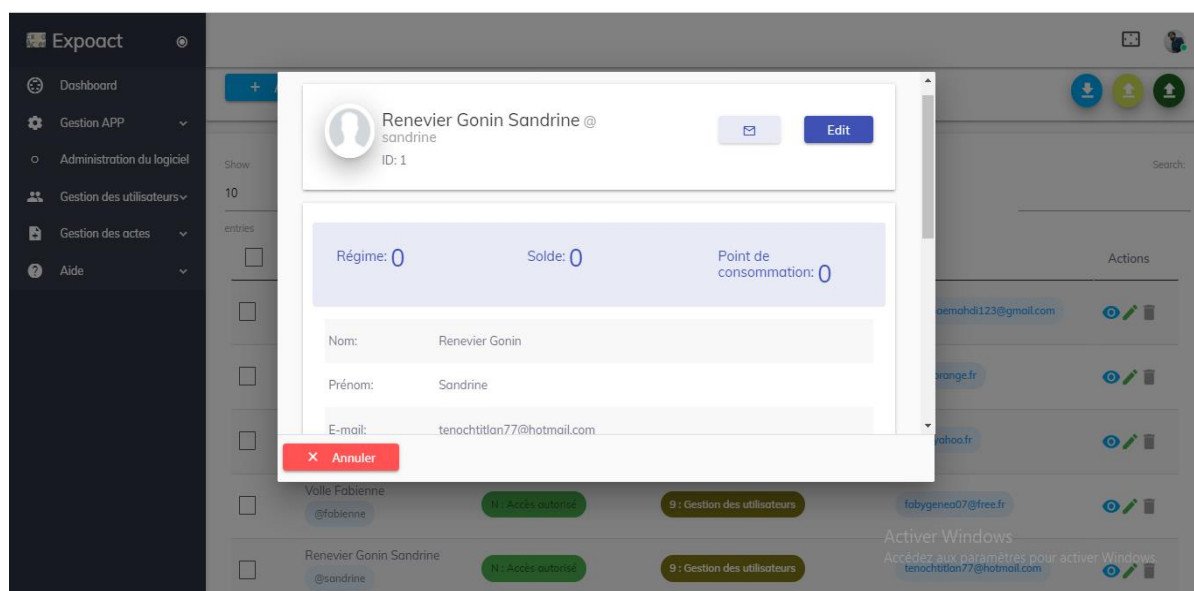

Figure 19 : User management page

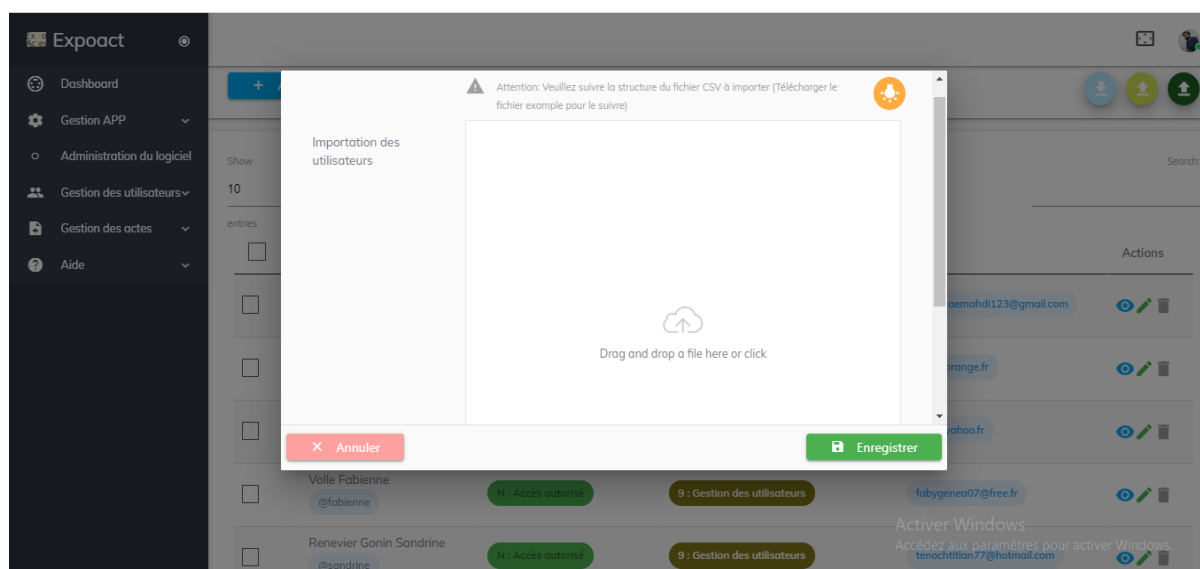
Figure 20: display of user information
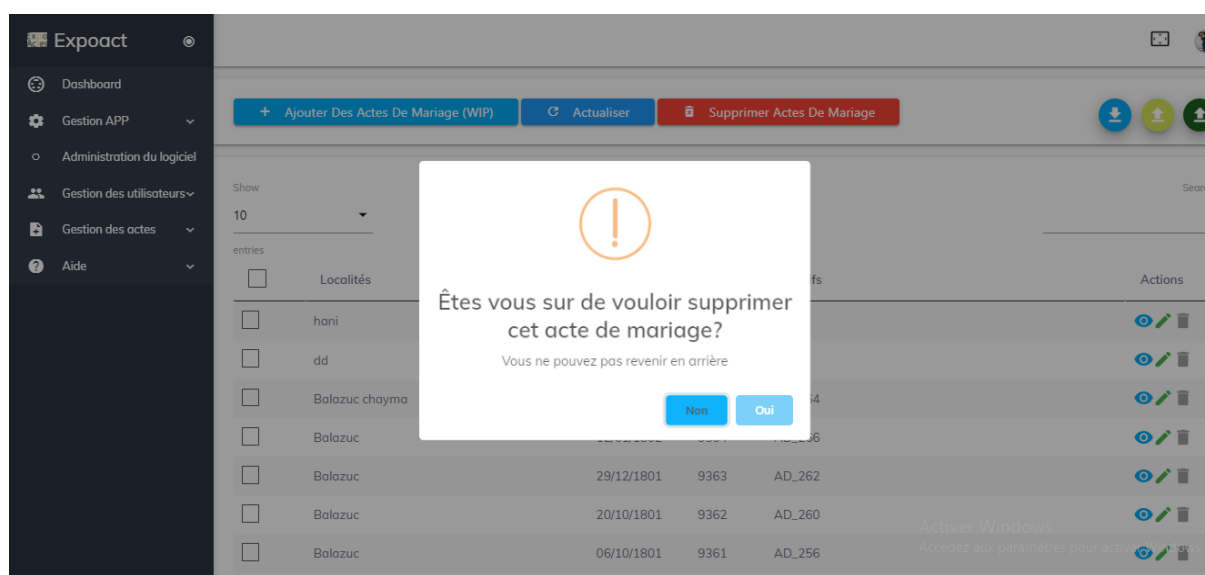
Figure 21: importing excel files

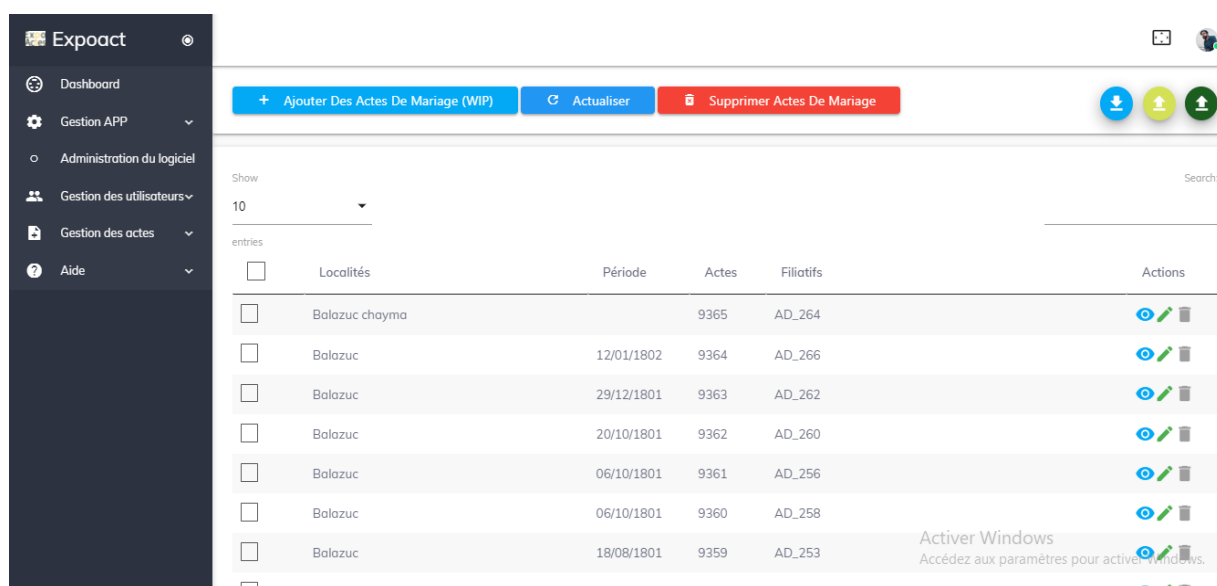*Figure 22: Deletion of an act*

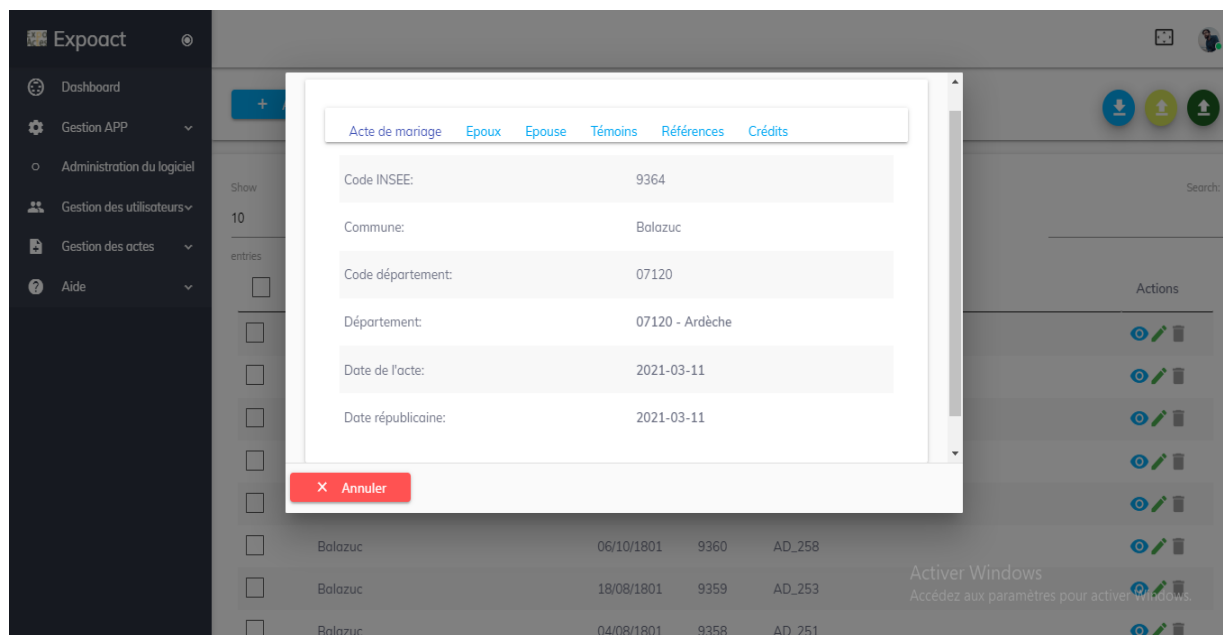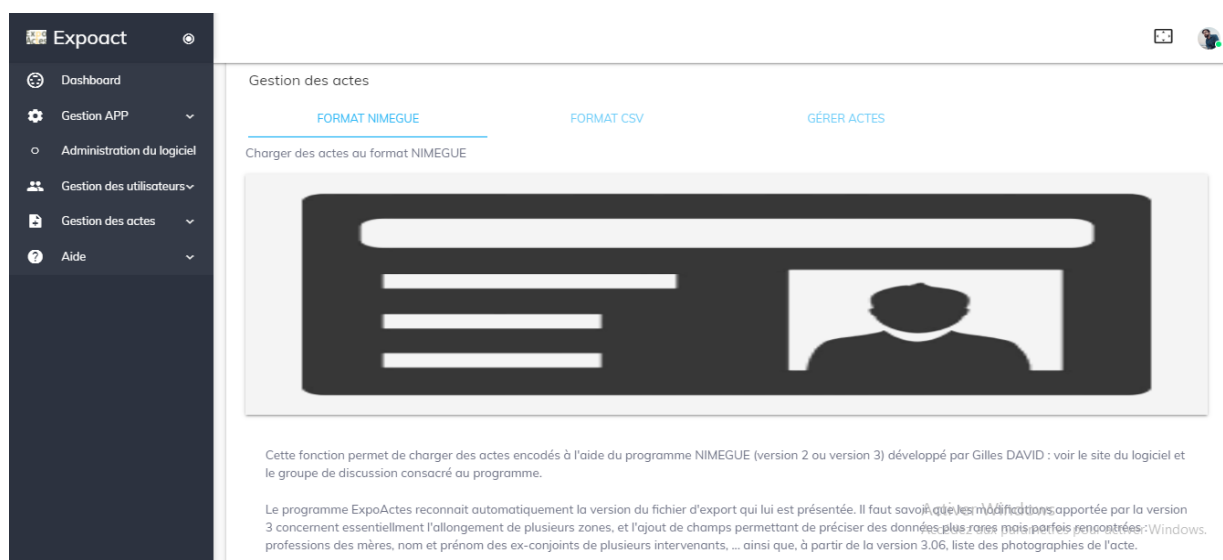*Figure 23: display of marriage certificates*

Figure 26: information displays of an act

Figure 25: Indication's page