Software Project Engineering

Final Report

Project leader : Hélène Verhaeghe
Group E
Aurian De Potter(Group leader),
Eddy Ndizera,
Ivan Ahad,
Arnaud Dethise,
Ludovic Fastré,
Anthony Dechamps,
Geoffroy Husson,
Jonathan Legat

INGI2255

1 Introduction & Presentation of The Project

Each year, the ASMAE non-profit organization organizes a tennis tournament that gathers many players during one week-end. Last September, representatives of this organization queried our help to revamp their website. This website is used for the tennis players to register themselves for several kinds of tournaments. However, their current website was old and started to show its limits.

Thus, our goal is to create a new revamped website that allows players to register, court owners to lend their own courts for the sake of the tournament, and having a better experience of the utilization of the website, as it will need to be easy and trivial to use. One important update is to have a fully working staff part to manage the tournaments, the players, the courts, etc.

This report will describe everything you need to know about the software in general and all the important information of how it works and how it has been built.

We will first start by listing all the optional and mandatory requirements that we were asked to implement and which of those have indeed been implemented. We will also add further explanation about our implementation choices for some of the requirements.

Then we will dedicate the third section to scan through all the processes of development of the project. This section will be divided in four parts. The first subsection will be the global organization of the project and the tools that we chose to use to build the software. In the second subsection, we will present you the website in general, where we will show you everything you need to know about it, and all the important functionalities that it required to have. The third one will be a more technical part that is the programming architecture, and finally we will explain how we scheduled all the tasks that were necessary for the development of the project.

2 List of All The Requirements

We will now list all the elements that we were asked to implement in our software. The requirements had different priorities, divided into *Must Have, Should Have, Could Have and Would Like to Have.*

However, rather than dividing them in those four categories, we decided to divide them into mandatory requirements and optional ones (the optional ones are in italic).

We also sorted them depending on if they're functionnal or non-functional. Note that we added some requirements not explicitly said by the clients but that we consider to be relevant for the project (such the privacy parts).

Functional :

- User :
  - The user can enter his data (including special characters)
  - The users can register to a tournament as a pair, selecting his category
  - *The user can reuse old data linked to his email address to automatically fill a registration form*
  - The user can register for activities during a tournament (barbecue, etc) and choose preferences such as taking responsibilities and payment method
  - *The user can register on a tournament alone and be matched with another player*
  - The user can register a personal court as being available for the tournament
  - *Users who registered their own court can see information about players who will play on those.*
  - *The user can use a payment method among several options*
  - The user can leave a comment when registering
- Admin :
  - An admin can start or close a tournament
  - An admin can close the registration for a tournament (and trigger the creation of pools)
  - An admin can create an admin account
  - An admin can create a staff account
  - An admin can delete an account
- Staff :
  - After a pool has been generated, a staff member can manually reorganize it
  - A staff member can manually assign a court for each match, or modify it in case of raining
  - A staff member can use a mail list / newsletter functionality
  - A staff member can enter or edit match result data
  - A staff member can see all the user informations (except credentials)
  - A staff member can visualize and print the evolution of the tournament in a fashionable and displayable form (both graphic and text), step by step using the knockoff, and print the pool matchup sheet
  - A staff member can access the courts list
  - *A staff member can manually match two users as a pair for a tournament*
  - *Common page for staff members*

  - *Files sharing between staff members*
  - *Communication channel between staff members*
- System :
  - *The system can create pools and match ups*
  - *The system must confirm email addresses*
  - *The system must query AFT rankings*
  - *The system can create smaller pools when it rains*

Nonfunctional :

- Interface :
  - The interface must be user friendly
  - *The interface must be mobile friendly*
- Documentation :
  - The software must be well documented

  Privacy :
  - Turn off indexation of private data
- Reasons to use a new software :
  - Interface is easier to use
  - Many tasks are automated
  - Code base is simpler and more modular if future modifications are wanted
  - More flexibility for users such as solo enrollment

2.1 Use cases

Here are a few use cases showing how our website will react upon receiving an action (registration of a player, registration of the owner of a court and creation of a group). Note that those use cases are subject to change when developping the project but it gives a good idea about the general process.

Registration of a player

- The admin enables the registration for a tournament.
- The user must access the registration page.
- The user fill his personal informations on the page.
- Or load the from previously used email address.
- The user must enter the personal information of his tennis partner if he plays in duo.
- The user gets a confirmation email. (Past this point he must contact the staff/admin to modify his information.)
- The admin closes the registration just before the tournament.

Registration of the owner of a court

- The owner must access the registration page.
- The own fill his court informations on the page.
- The owner receive an email confirmation.
- When the admin opens a new tournament, he will be notified of all the previously used courts and asks the owners about their availability.
- The staff will check the court and mark it as suitable.
- When a court is assigned to a tournament, the owner is notified by email.

Creation of a group

- Once the registrations are closed, the system will generate the groups for each category.

- The staff will review the generated list. If there are wishes to fulfill, the list can be manually reorganized. At this step, the staff can also contact the users with payment issues.
- Once the groups are checked, the staff confirms their composition.
- The system sends an email to the users to inform them of the location and time of their matches and of their payment status. The leader is also informed of what is needed of him.

3 Development and Implementation

The requirements below are the ones we planned to develop for the second phase. This time, every requirement has been implemented.

3.1 Organisation and Tools used

We are briefly going to explain the way the group is going to develop the project. After being introduced to the project by our customers, we had to choose between a few development methods to use throughout the making of the software. The group decided pick the **Agile** method.

The first step of this method is writing down user stories. These are used to present all the functionalities of the software which is the website. Afterwards, the group has to discuss about the importance and the difficulty of each user story, and then decide which of them have the top priority.

When this is done, we will use a kanban to organize the process of the project. The kanban contains all the user stories, and is divided into the following categories : "To do, doing this iteration, done this iteration, and done". Every member of the group picks the user stories he thinks he is able to implement, so it allows us to know which member has done each task. For the kanban we will use the **Trello** Website.

The programming language used will be Python, with the **Django** framework. We will use the third version of Python, and the 1.8 version of Django. Lastly, we will use the **Bootstrap** framework for the design of the website (the css part essentially).

To easily share the project's code, we will use a Control Version System, which is in this case **Mercurial**. Mercurial, and its underlying service called BitBucket, is preferred since it allows us to make private repositories.

We also decided to divide the project into 4 iterations, corresponding to the 4 phases presented to us. With the Agile method, there will be a working deliverable for each of these phases. Through each iteration, we will have to add up functionalities to the working project, until each of them is implemented.

To host a live version of the website, we used Heroku.

There are a few reasons why we picked this way of working. First of all, this method was genuinely our first choice since it is the one we are the most comfortable with, as we have been using such a method for a few projects now. Moreover, we liked its versatility since every member of the group has to be involved in every aspect of the project. Furthermore, after each iteration, we will have a meeting with our Assistant. This meeting will allow us to get a feedback of each deliverable. The feedback received will then help us to know if changes need to be done, if there are unnecessary things, or if our project is moving towards the right direction. Finally, with the Agile method, we will first implement all the "must have" functionalities, which allow us to focus on the core of the software, and then add up the functionalities with less priority.

3.2 The Website, Its Functionalities

Here is the live version of our website : http ://sep2015e.herokuapp.com. To get admin access you should add "/admin" in the url.

To get access to the code of our project, you can click on the link to our *Github* repository : https ://github.com/ivanahad/sep2015E

3.3 The Architecture

ORM, UML

3.3.1 The Object-Relational Mapping

Fig.1 shows our ORM diagram. It gives an idea about how the different objects used by the website are linked together. We will give a brief description of what the object represents and its relations to others.

The user represents a player participating in the tournament. It is characterized by a name, an email, an address and the level of the player (meaning how good he is at tennis).

A pair is formed of two users. A single pair can only participate in one pool of the tournament. A pool contains pairs and their match (matchups between pairs). A match is characterized by the pairs confronting each other and the court in which they will play. The court object contains information about the court owner, the address of it and finally some comments about it like the type of surface.

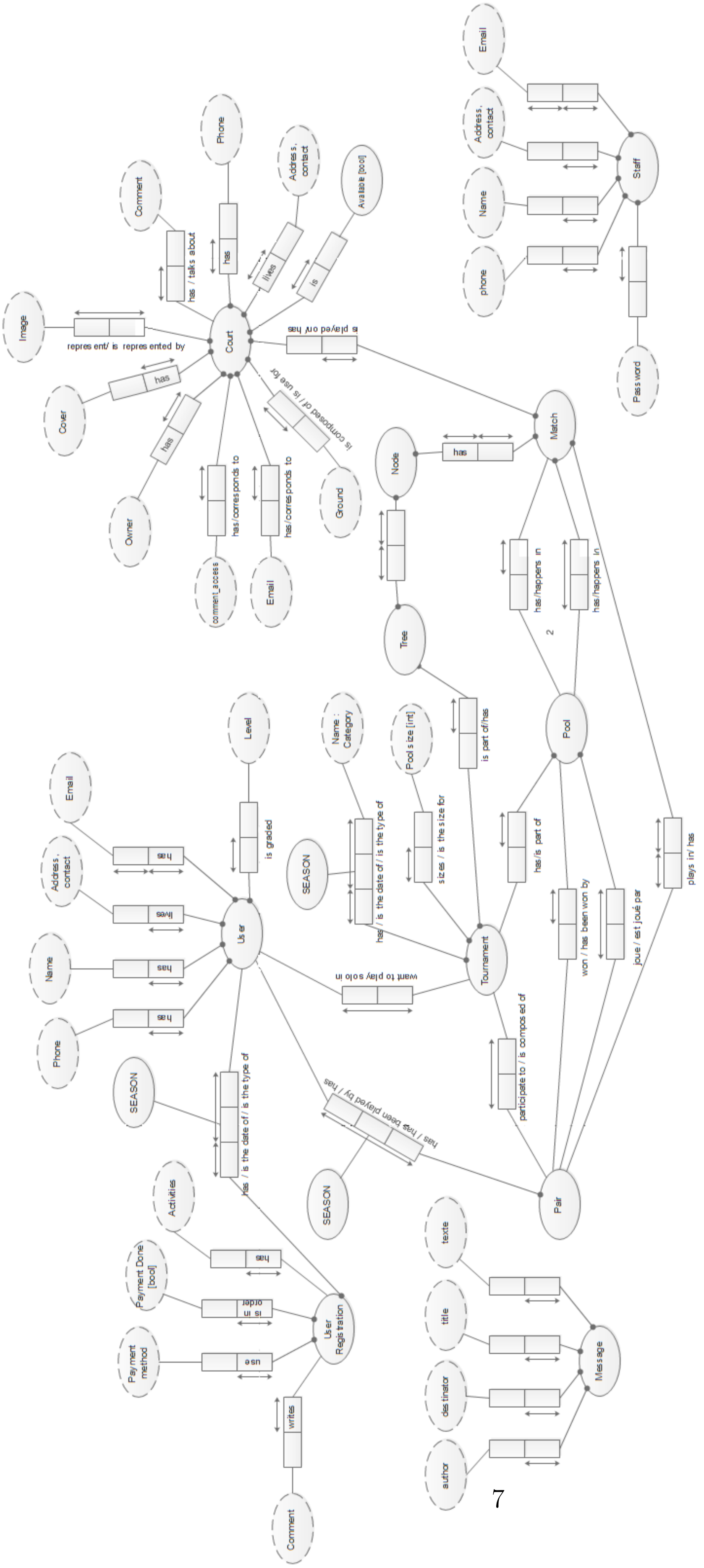
As we said, a tournament is composed of pools but also of a knock-off tournament that we called tree in our diagram. Moreover, a tournament is defined by its category. Finally, the tree is composed of nodes that is basically a pair of two users.

A staff, similarly to a user is characterized by it's name, address, contact (phone and email) but also by it's password. Also a staff is part of group representing the actions he can do.

A message is defined by it's author which is a staff member, by it's title and by it's content (text). A file is defined by it's title, it's owner and the file.

This concludes the description about the ORM diagram.

Figure 1 – Revamped ORM Diagram



3.3.2 The Uniform Modeling Language

UML

3.4 Planning and Scheduling, Evolution of the Project

3.5 Implementation & choices

Here are a few other features that we implemented during this phase.

Upon encoding all the scores of the matches in a pool, the winner of this group is displayed in green and the knock-off tournament is created in the database. Plus, it is now possible to see the bracket with a pretty graphical version. The players that are the winners of a group will then get to the bracket. The scores are, for now, only editable by an admin account. Another feature that was added is that a user can now delete his own email address from the database, so that he can delete his own information on his own.

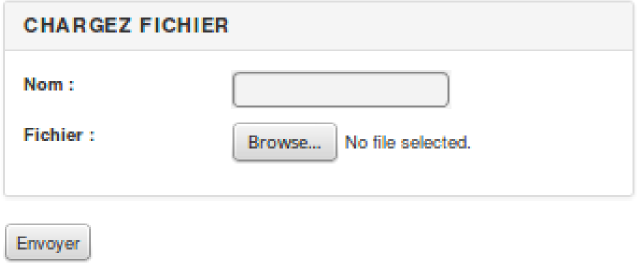
For the implementation of *"The system must query AFT rankings" and "The system can create smaller pools when it rains"*, as these two functionalities were optional, we decided not to implement them. It thought it would take too much time to do so, and that implementing wouldn't be a great benefit after all.

As you can see, there weren't many requirements as scheduled for this phase. As stated in the previous report, this phase was much more dedicated to improving some underlying functionalities of the website to make it easier to use.

Implementations according to meeting with client

First of all, as requested, we implemented a system that allows to exchange messages. We also added a system that allows to share files that they can upload/download. The staff members already have other means to communicate, so these two features are important for very important announcements. They are available on a dedicated page. You can see in figure 2 below how a staff member can put an announcement, and how one can upload files in figure 3

Figure 2 – Announcement creation



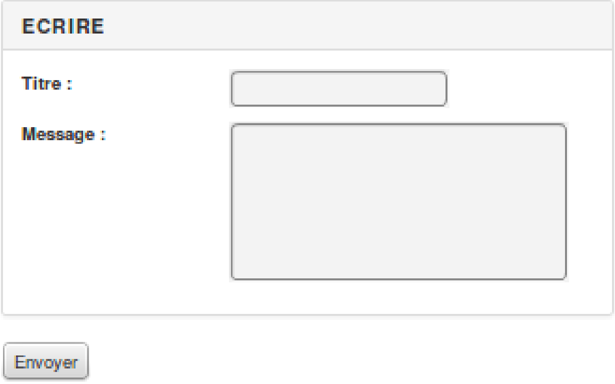
CHARGEZ FICHIER

Nom :

Fichier : No file selected.

It is also now possible to modify the pools by taking into account the commentaries made.

Figure 3 – File sharing



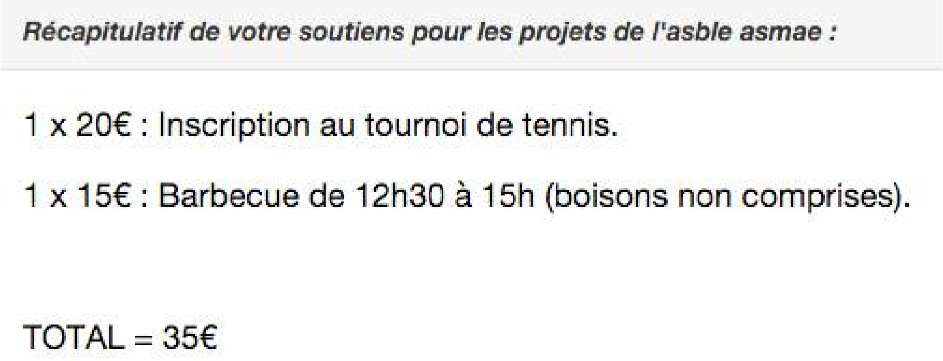
We also added a functionality that allows to look for players and courts, but also pairs. We then added a list of all the players and all the courts.

Changes during registration

Secondly, we focused on modifying the registration of a player, as it was asked by the client. We have implemented the smart registration, where a player is automatically assigned to a tournament according to his information.

Moreover, when a form is filled, there is a summary of all the player information, allowing him to check what he is going to pay afterwards. You can see in figure 4 how the information is presented after validation.

Figure 4 – Summary of player information



Récapitulatif de votre soutiens pour les projets de l'asble asmae :

1 x 20€ : Inscription au tournoi de tennis.

1 x 15€ : Barbecue de 12h30 à 15h (boisons non comprises).

TOTAL = 35€

Another modification is that now the payment choice is chosen after a user has entered and validated all his information.

4 The Final Product : The software as a Website

5 For next phase

5.1 Tests to improve the software

To improve our software, we are going to ask random people to test the website to see if it is user-friendly and easy to use, and then we will proceed to modify it accordingly.

We will also navigate randomly to search for bugs and strange behaviors that may arise after modifications of the database.

We will also test our website on different monitors with different screen sizes to see if it fits the window correctly.

5.2 Documentation

The documentation will be one of our main focuses. It will be divided into two parts : the first will be dedicated for the staff and admin and the second for the maintainer of the website. For the admin and staffs, the documentation will be a guide on how to navigate on the website and on how to basic stuffs like as modifying a pool, editing a player, etc. This documentation will be found on a wiki but links will be on the website to directly access the content specific to a part of the website. For the maintainer of the code, the documentation will be more technical. We will explain our implementaion, what we use and how we do it.

6 Architecture discussion, choices & UML Diagram

You can see our UML diagram in Figure 4. Since the last phase, we changed.

In the architecture, in this phase, there isn't much that has changed. However, one big change is that we added many methods that now allow to modify information about users, courts, tournaments, etc. Making things changeable manually was one of our main coucerns for this phase, as it was requested by the client.

Sequences diagram

You can see in figure 6 the sequence diagram for how a player registers to a tournament with an email confirmation. It has been updated since the last phase's report.

The second sequence diagram describes the life cycle of a tournament, how players are added into groups, how the knock-off tournament is created, how the tournament is created, etc.

7 Teamwork

In this section, we will explain how we worked as a group.

The main focus of the organization was the **Agile** method. As explained earlier in section 3.1, we valued this technique of work, as it allowed us to work independantly from each other. Once all the tasks were defined, each member assigned himself to implement the ones we thought he was capable of doing.

However, we quickly came to the conclusion that this technique was hard to use in practice. With this method, the load of work was not identical for every member, thus some members achieved more and some members couldn't keep up with the pace and the progress of the project. This was also due to the fact that the communication between the members was very sloppy. We must have been more strict about the deadlines by fixing some intermediate once. The lack of communication once lead us to forget submitting one of our intermediate reports.

There was also a lack of intermediate meetings between the members that would have helped us see the progress of the project. Those meetings could have helped the members that had difficulties acheving some of their tasks. Moreover, they would have helped us keep our focus on the important things so that we wouldn't start going towards a wrong direction.

At the end of the day, we still managed to make a fully workable software, but we think we could have been more focused on those details to help us optimize our work. However, the team got along very well, there was no friction between the members, and the atmosphere was always very good.

8 Conclusion

Figure 5 – UML diagram

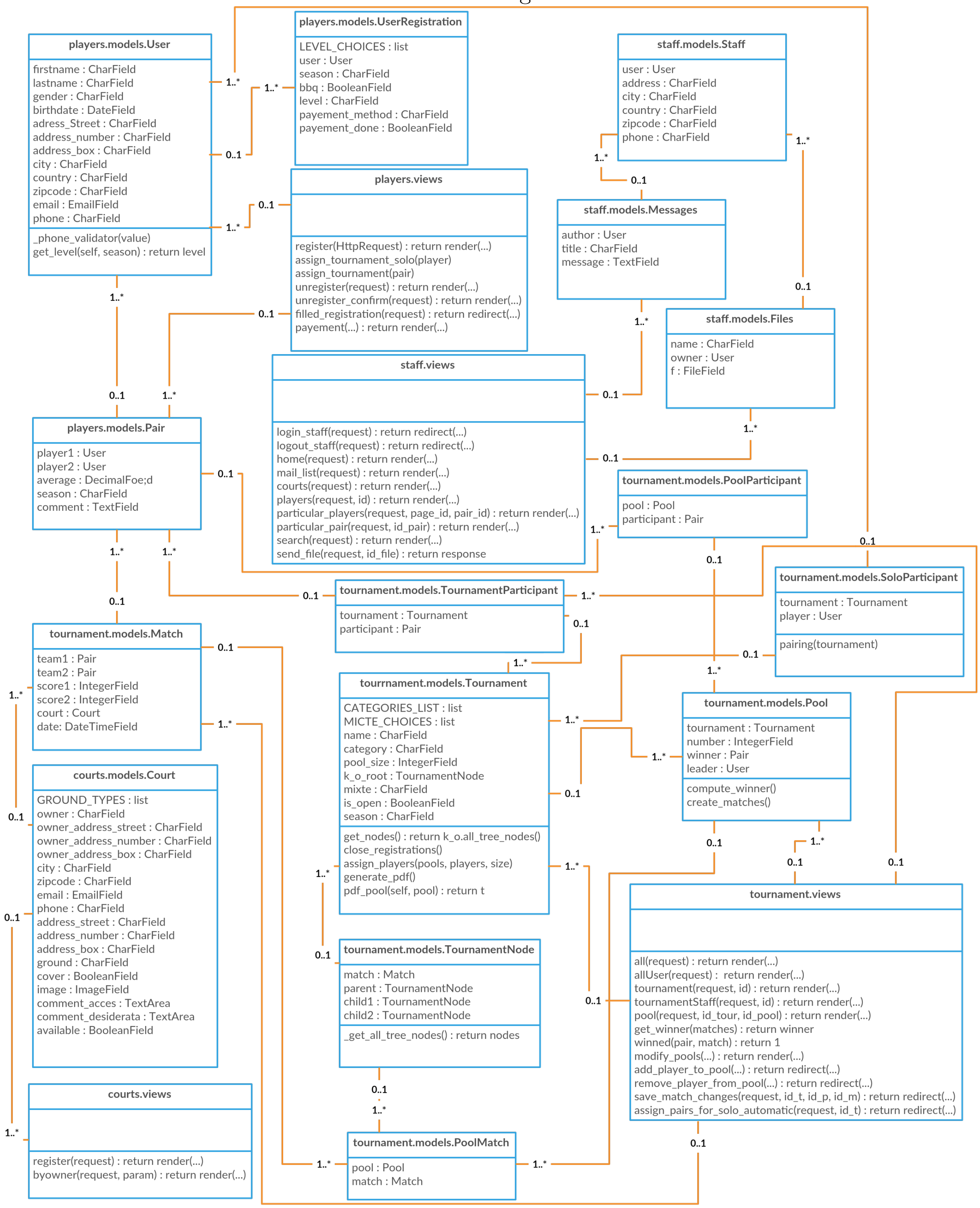

players.models.User
firstname : CharField
lastname : CharField
gender : CharField
birthdate : DateField
adress_Street : CharField
address_number : CharField
address_box : CharField
city : CharField
country : CharField
zipcode : CharField
email : EmailField
phone : CharField
_phone_validator(value)
get_level(self, season) : return level
players.models.UserRegistration
LEVEL_CHOICES : list
user : User
season : CharField
bbq : BooleanField
level : CharField
payement_method : CharField
payement_done : BooleanField
staff.models.Staff
user : User
address : CharField
city : CharField
country : CharField
zipcode : CharField
phone : CharField
players.views
register(HttpRequest) : return render(...)
assign_tournament_solo(player)
assign_tournament(pair)
unregister(request) : return render(...)
unregister_confirm(request) : return render(...)
filled_registration(request) : return redirect(...)
payement(...) : return render(...)
staff.models.Messages
author : User
title : CharField
message : TextField
staff.models.Files
name : CharField
owner : User
f : FileField
staff.views
login_staff(request) : return redirect(...)
logout_staff(request) : return redirect(...)
home(request) : return render(...)
mail_list(request) : return render(...)
courts(request) : return render(...)
players(request, id) : return render(...)
particular_players(request, page_id, pair_id) : return render(...)
particular_pair(request, id_pair) : return render(...)
search(request) : return render(...)
send_file(request, id_file) : return response
players.models.Pair
player1 : User
player2 : User
average : DecimalFoe;d
season : CharField
comment : TextField
tournament.models.PoolParticipant
pool : Pool
participant : Pair
tournament.models.TournamentParticipant
tournament : Tournament
participant : Pair
tournament.models.SoloParticipant
tournament : Tournament
player : User
pairing(tournament)
tournament.models.Match
team1 : Pair
team2 : Pair
score1 : IntegerField
score2 : IntegerField
court : Court
date: DateTimeField
tourrnament.models.Tournament
CATEGORIES_LIST : list
MICTE_CHOICES : list
name : CharField
category : CharField
pool_size : IntegerField
k_o_root : TournamentNode
mixte : CharField
is_open : BooleanField
season : CharField
get_nodes() : return k_o.all_tree_nodes()
close_registrations()
assign_players(pools, players, size)
generate_pdf()
pdf_pool(self, pool) : return t
tournament.models.Pool
tournament : Tournament
number : IntegerField
winner : Pair
leader : User
compute_winner()
create_matches()
courts.models.Court
GROUND_TYPES : list
owner : CharField
owner_address_street : CharField
owner_address_number : CharField
owner_address_box : CharField
city : CharField
zipcode : CharField
email : EmailField
phone : CharField
address_street : CharField
address_number : CharField
address_box : CharField
ground : CharField
cover : BooleanField
image : ImageField
comment_acces : TextArea
comment_desiderata : TextArea
available : BooleanField
tournament.models.TournamentNode
match : Match
parent : TournamentNode
child1 : TournamentNode
child2 : TournamentNode
_get_all_tree_nodes() : return nodes
tournament.views
all(request) : return render(...)
allUser(request) : return render(...)
tournament(request, id) : return render(...)
tournamentStaff(request, id) : return render(...)
pool(request, id_tour, id_pool) : return render(...)
get_winner(matches) : return winner
winned(pair, match) : return 1
modify_pools(...) : return render(...)
add_player_to_pool(...) : return redirect(...)
remove_player_from_pool(...) : return redirect(...)
save_match_changes(request, id_t, id_p, id_m) : return redirect(...)
assign_pairs_for_solo_automatic(request, id_t) : return redirect(...)
courts.views
register(request) : return render(...)
byowner(request, param) : return render(...)
tournament.models.PoolMatch
pool : Pool
match : Match
1..*
0..1

Figure 6 – Sequence diagram

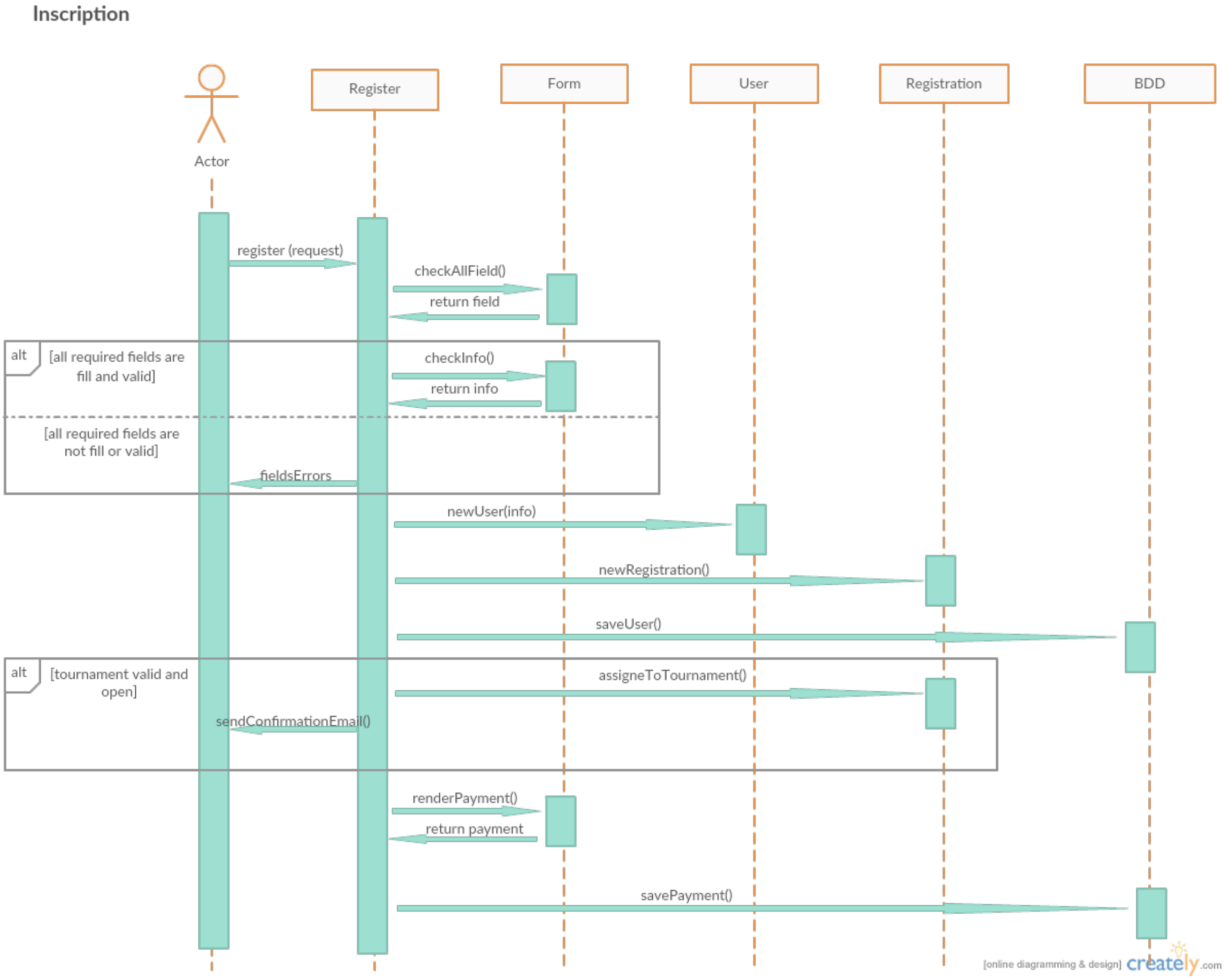
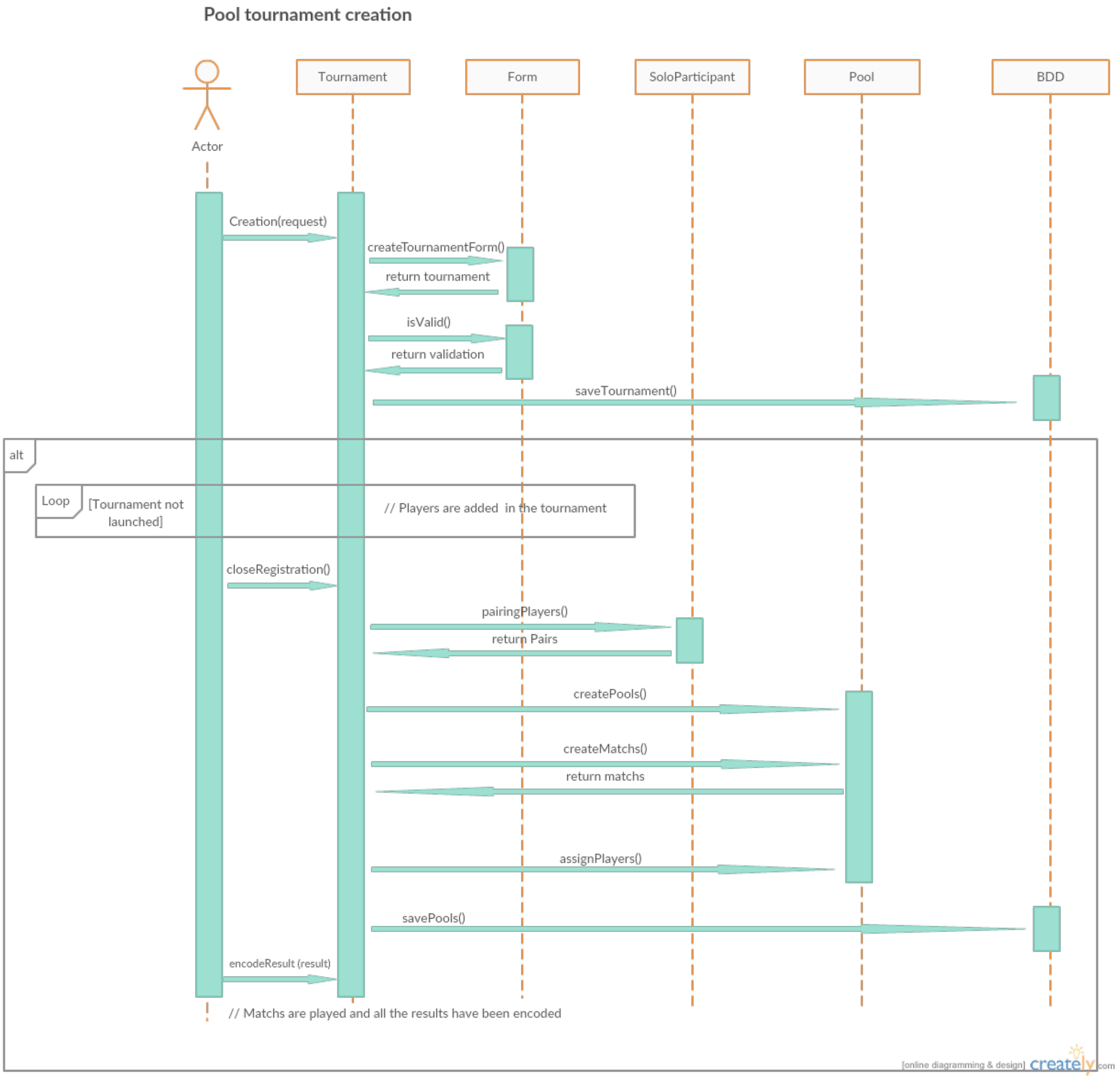


Figure 7 – Sequence diagram for tournament


Pool tournament creation
Actor
Tournament
Form
SoloParticipant
Pool
BDD
Creation(request)
createTournamentForm()
return tournament
isValid()
return validation
saveTournament()
alt
Loop
[Tournament not launched]
// Players are added in the tournament
closeRegistration()
pairingPlayers()
return Pairs
createPools()
createMatchs()
return matchs
assignPlayers()
savePools()
encodeResult (result)
// Matchs are played and all the results have been encoded
[online diagramming & design] creately.com