EasyMeeting A Group-Decision Support System Release 1

Guy Camilleri, Pascale Zaraté

EasyMeeting
A Group-Decision Support System
Release 1

Report: IRIT/RR--2014--10--FR

Guy Camilleri[1] and Pascale Zaraté[2]

[1] Université Paul Sabatier - IRIT-SMAC - 119 route de Narbonne, F-31062 TOULOUSE CEDEX 9

[2] Université Toulouse 1 Capitole - IRIT-SMAC - Faculté d'Informatique, 2 rue du Doyen Gabriel Marty, 31042 Toulouse Cedex 9, France

{zarate, camiller}@irit.fr

December 14, 2014

CONTENTS

EasyMeeting is a simple, comfortable and free solution for your meetings. With this tool, you will be able to make Efficient, Productive and Collaborative meetings.

CHAPTER

ONE

INTRODUCTION

Collaboration was recognized as a key factor for the success of many organizations [6]. Today, organizations are increasingly often face to complex problems that no individual in the organization is able to solve alone, so it is not surprising that collaborative group work has become a fundamental aspect of organizations. Therefore, in most organizations, the decisions do not appear as a result produced by a single decision maker but as a compromise between several decision makers which often have divergent interests and goals. Indeed, decisions very often call on the expertise of several people, in particular those were concerned by the decision.

Group Decision Support Systems (GDSS) are a widely used collaborative technology that has proven to increase user participation in and the quality of decision-making. They are intended to provide computational support to collaborative decision-making processes (see [5]). In this report, we use the following definition of GDSS proposed by DeSancis et al.: "A GDSS is a computer-based technology designed to help committees, project teams, and other small groups with activities such as problem identification and analysis, decision making, planning, creativity, conflict management, negotiation, and meeting management. GDSSs combine communication, information, and decision support technologies in an integrated environment." p. 552 in [5]. Many studies demonstrated the advantage of using such systems in meetings in order to improve their effectiveness and efficiency [9, 3, 10].

Many types of collaboration tools exist, some of them cover all or a part of the group decision making process (see [11]). According to Desanctis and Gallupe, a GDSS tool provides a set of tools supporting group decision making activities. Unfortunately, the majority of GDSS tools covering most of group decision-making activities are commercial. Few open-source GDSS tools exist, most of them are research prototypes providing a set of tools coupled with a theory or an approach (as GSSOne [7], ActionCenters [1, 2, 8] ...).

In this report, we propose a GDSS tool called EasyMeeting. This system is open source, and is independent to any group theory, following the initial idea of shell presented by Desanctis and Gallupe in [4]. Two main purposes guide the design of EasyMeeting. From practical point of view, the objective is to provide an easy, extensible and integrable system. And for researchers, the goal is to share approaches and tools, to quickly build a system or tools by reusing some components developed by others, and to facilitate the comparison of theories and/or approaches (group processes, tools, etc.).

This report is organized as follow: in the first chapter an overview of EasyMeeting features and

functionalities is presented, then the choice of the framework is very briefly justified in chapter two. After, the use of EasyMeeting system is explained (chapter three). The chapters four and five contain technical documentations on available tools and system core. Finally, we will discuss some possible improvements (chapter seven).

CHAPTER

TWO

EASYMEETING FEATURES AND FUNCTIONALITIES

Easymeeting is an open source web application. This application is used by different types of users:

- designers of collaborative tools (application developers),
- designers of collaborative process (easymeeting users such as collaboration engineers)
- sessions facilitators (users of EasyMeeting)
- average participants (users of EasyMeeting).

Group facilitation is defined as a process in which a person who is acceptable to all members of the group intervenes to help improving the way it identifies and solves problems, and makes decision [4]. Facilitation, on the other hand, is a dynamic process that involves managing relationships between people, tasks, and technology, as well as structuring tasks and contributing to the effective accomplishment of the meeting's outcomes.

In the majority of the cases, in organizations, meeting facilitators also play the role of "average" participant. Therefore, EasyMeeting provides a simple graphical interface, intuitive and friendly. Moreover, it does not disturb too much the role of participant of the facilitator.

Easymeeting can be used in synchronous / asynchronous and distributed / same-place contexts. The application allows the management of multiple sessions in parallel. Each session can take place synchronously (at the same time) or asynchronous (different times). Participants can be geographically distributed (in different places). In addition, the application provides a mode "public session", that is sessions without facilitator accessible to any individual registered in EasyMeeting system.

In order to simplify the use of the tool, we chose to define a group process as a simple sequence of collaborative tools. However, it is possible for tool designers to provide container tools putting in parallel several activities (tools).

Roughly EasyMeeting can perform the basic tasks commonly available in GDSS such as:

- definition / design of a group decision process:
  - definition of tools sequence,
  - selection and configuration of collaborative tools,

  - Recording of defined processes.
- The management (add, modify, delete, ...) of collaborative tools,
- Change during session of group process,
- Management of automatic reporting in form of PDF file,
- Recording the course of the session.

In the current version, it offers the following components:

- users registering and logging component
- meeting creation component (private or public)
- group process creation component
- session management component:
  - session edition, participants invitation, session ending . . .
- group process management component
  - Adding / removing a tool during the session,
  - controlling the sequence of collaborative tools (transition to the next tool, . . .)
  - . . .
- Collaborative tools available:
- brainstorming tool
- clustering tool,
- voting tool
- consensus tool,
- reporting tool.

CHAPTER

THREE

WEB FRAMEWORK : GRAILS

From technical point of view, Easymeeting is implemented in the framework Grails. This framework is based on the Groovy language, a very high level language like Python or Ruby. Groovy can be compiled to Java Virtual Machine bytecode and can interoperate with others java codes or libraries. This framework answers to our main development constraints:

- *a very high-level framework that do the most things as possible and which is supported by a developers community*. The idea is to reuse the most existing components as possible. This limits the application self- code to maintain.
- *a framework based on Java platform, Enterprise Edition (Java EE)*. Java EE is a robust, efficient and widely used platform. Grails has been chosen between classic EJB components, Spring frameworks. EJB components are not a framework. Spring and Grails are frameworks, but Grails is higher-level.

CHAPTER

FOUR

EASYMEETING USE

GDSS Project is a collaborative tool that helps groups to make decisions.

4.1 Before the meeting

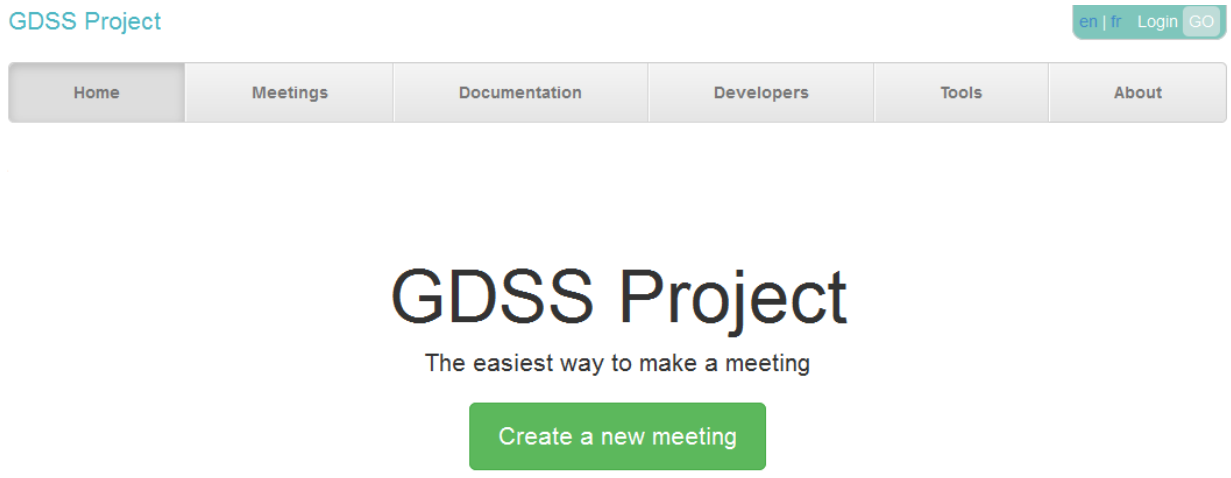
There are several tasks to complete before running a meeting. This includes the following :

- Clearly state the goals you want to accomplish at the meeting
- Create a process using available tools to reach those goals
- Create a meeting based on a process and invite members to the meeting

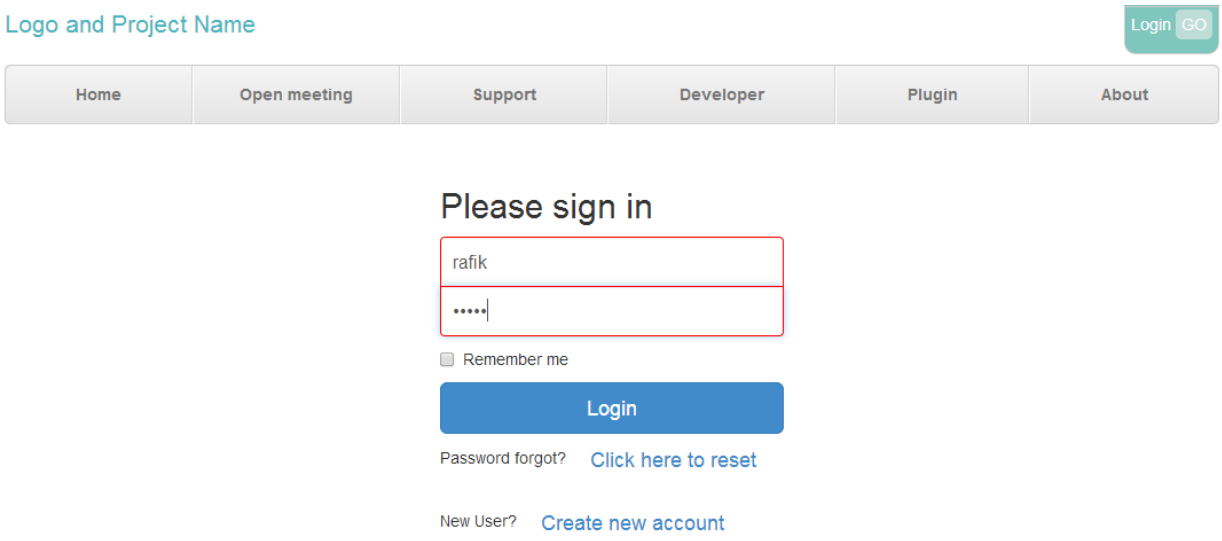
Each of this tasks is detailed in this report.

4.1.1 Starting GDSS

Type the correct url in your browser, the home page is as follows:

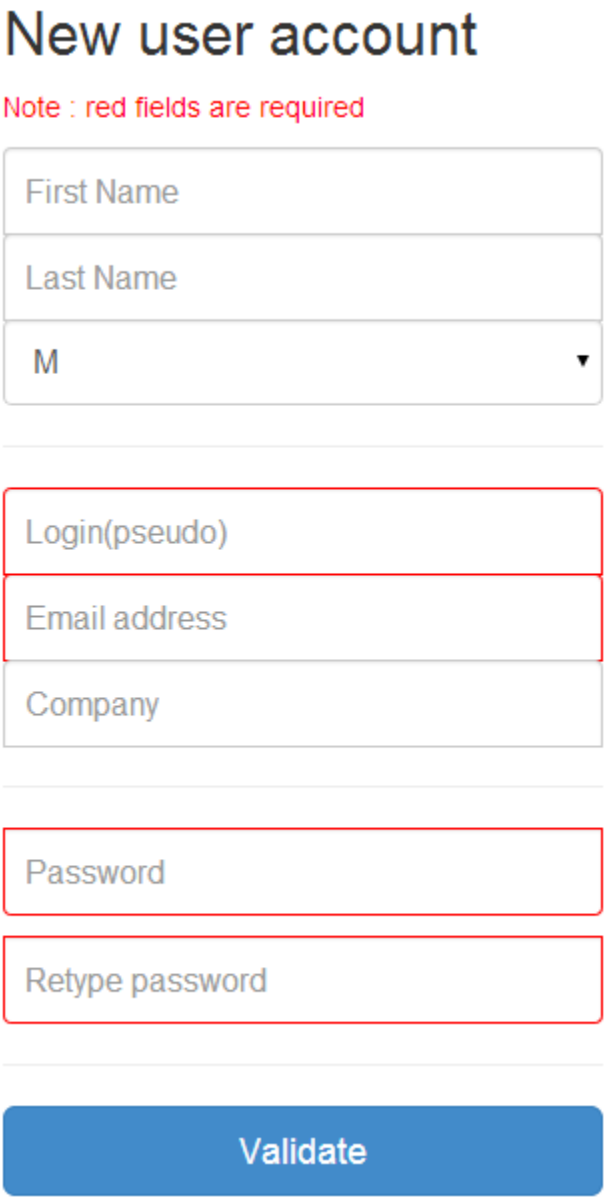


In the above right click on « GO ». The authentification form appears as follows :



- Login name requires that you type your login
- Password allows you to type your GDSS password.

Then press the button « Login ». If you don't have an account you may create a new one using the hypertext link « Create new account », you should fill a form as follows :

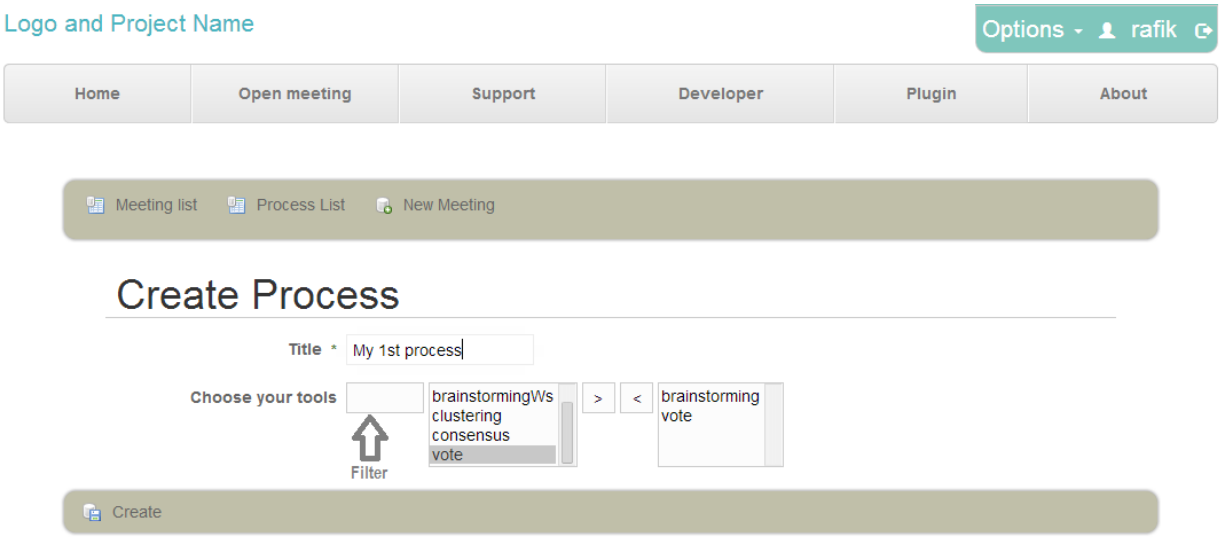


Fields marked with a red color are required. Then press the button « validate ». Congratulations, your account is now created !

4.1.2 Creating a process

From the menu bar of process, select «New process»:

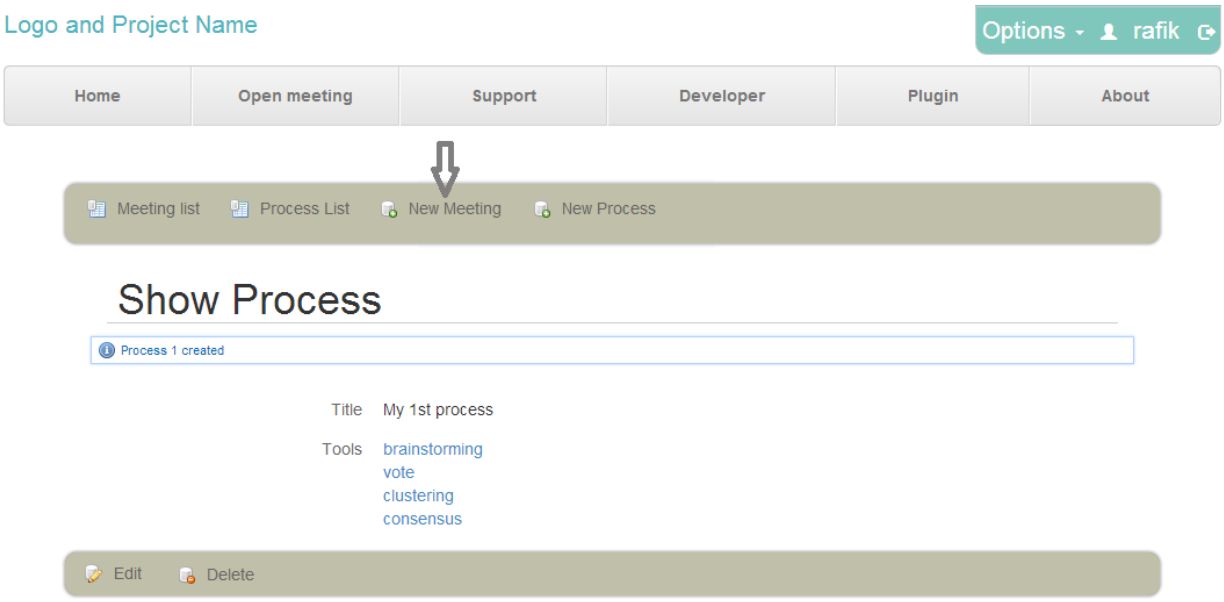
You should see :



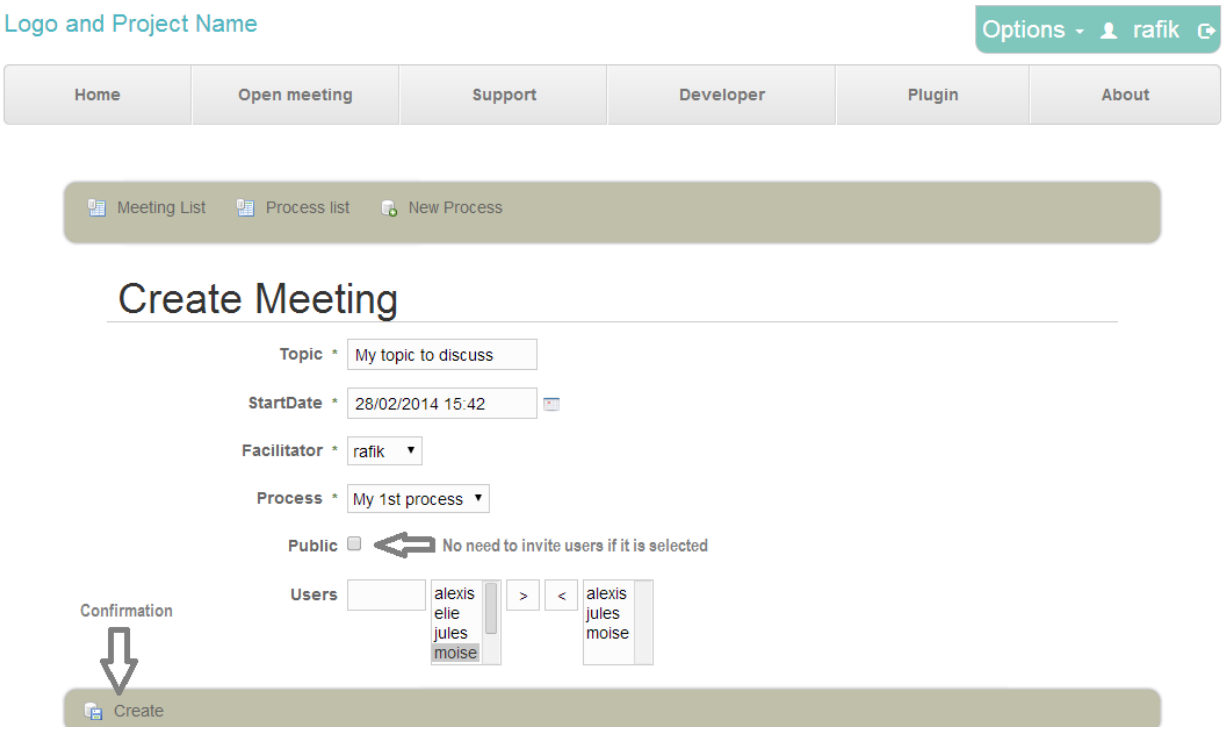
- Choose a title for your process
- Add existing tools to build your process, you may use the filter box.
- Press «Create»

4.1.3 Creating a meeting

Once your process is created, a summary appears as follows:



Then press « New meeting » to create a new meeting :



- Choose a topic for your meeting
- Select a starting date using the calendar
- Choose a facilitator from users list
- Select a process for your meeting
- Select the checkbox « Public » to render your meeting as public

- Invite users from the selection list to your meeting, note that if you have selected the proprety public there is no need to invite users as everyone can access the meeting.

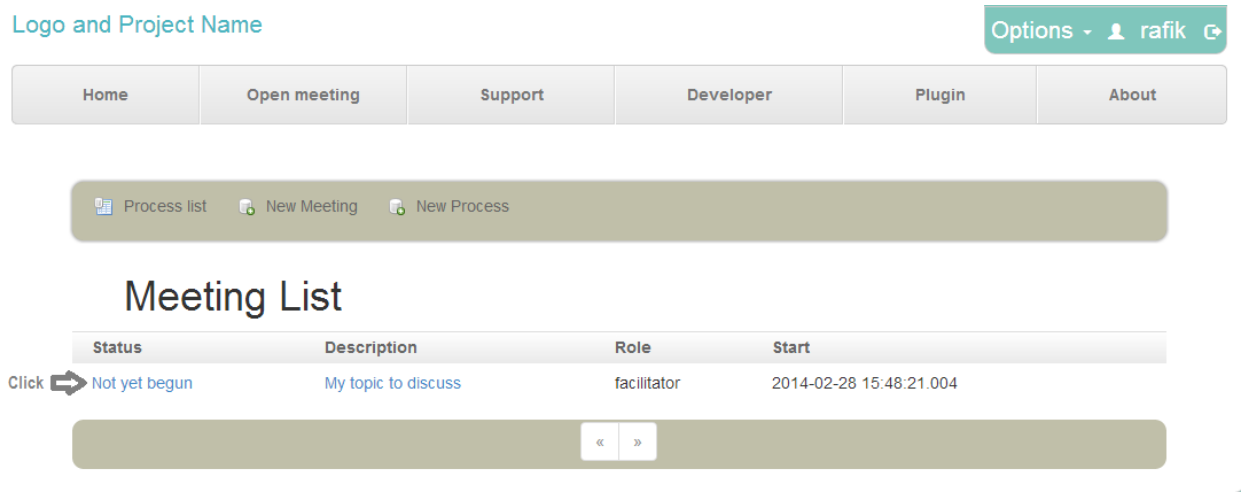
Once all fields marked with "*" are filled, press the button "Create".

4.2 Running the meeting

Now that you have prepared the meeting, you are ready to run your activities with participants.

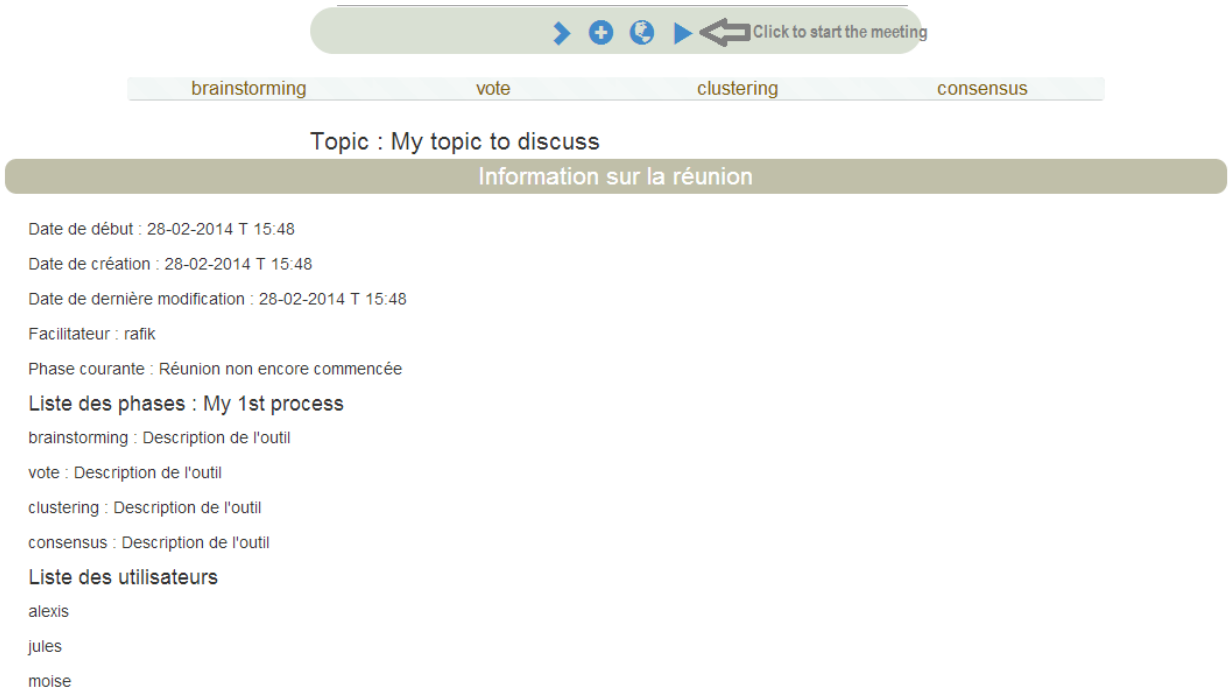
4.2.1 Start a meeting

Once the meeting is created, you should see a list of all created meetings and your role in each one :



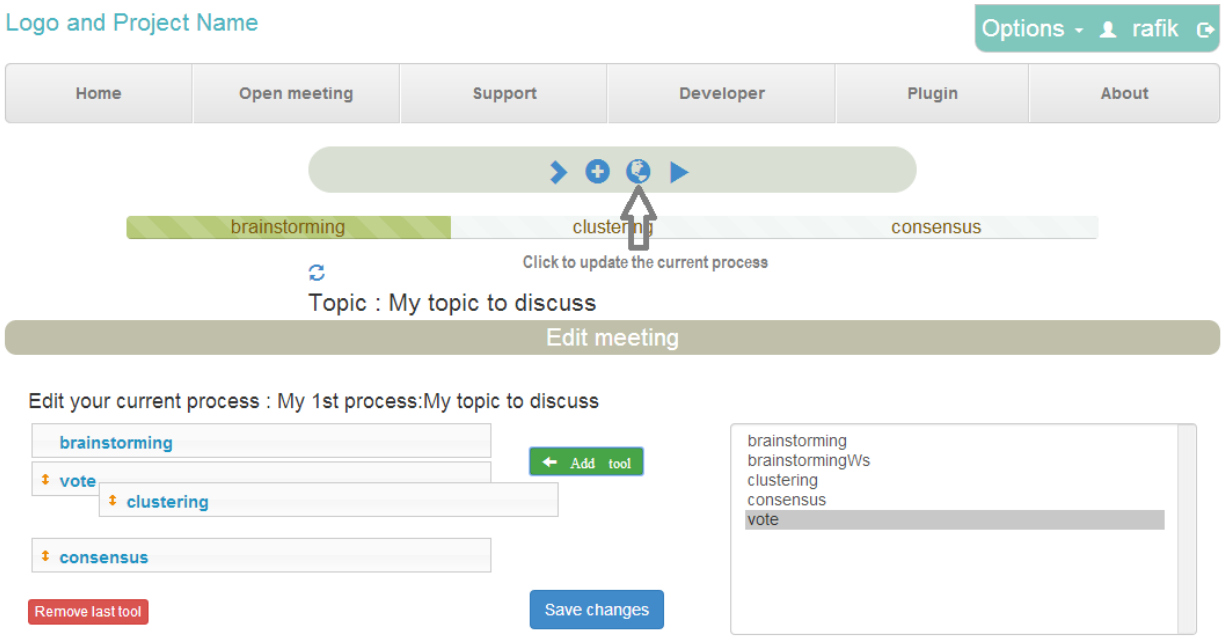
For meetings that you have the privilege « facilitator », you can manage them by a simple click as it is marked in the above screen. You will have then an admin bar, starting from left to right there are four options :

- Next icon to move to the next step of the process wich is summarised in a progress bar
- Add icon to invite new users during the meeting
- Edit icon that allows the facilitator to edit the current process, for example he may delete or add new tools for new steps.
- Start icon which lunchs the meeting



4.2.2 Edit the workflow

The interface gives the possibility to facilitators to update the process of a meeting in real time. By pressing the edit icon you should see a screen as follows :



The available functionalities are :

- Add a tool : Select a tool from the right list and click on the button « Add tool » to add it to

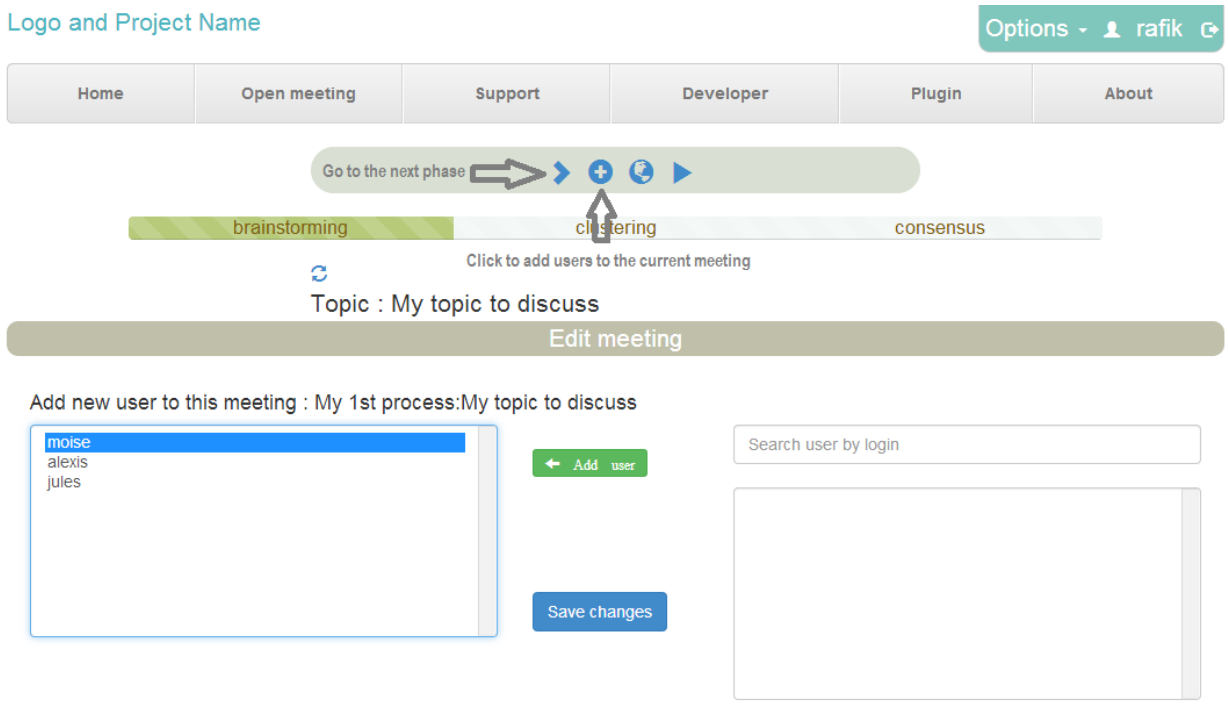
your list of tools.

- Remove a tool : you can only delete the last tool in your list, so if you want to delete a tool drag and drop it to the last position, then press the button « Remove last tool ».
- Change order of tools : It easy to change the order of your tools in your workflow, just drag and drop tools and arrange them in the way you want.

Once you have brought the modifications that meet your needs, press the save button to save changes and update the view.

4.2.3 Add new users

During a meeting, the facilitator may invite new users to the meeting. In the admin bar, press the add icon to display the interface of users management. You should see a screen as follows :



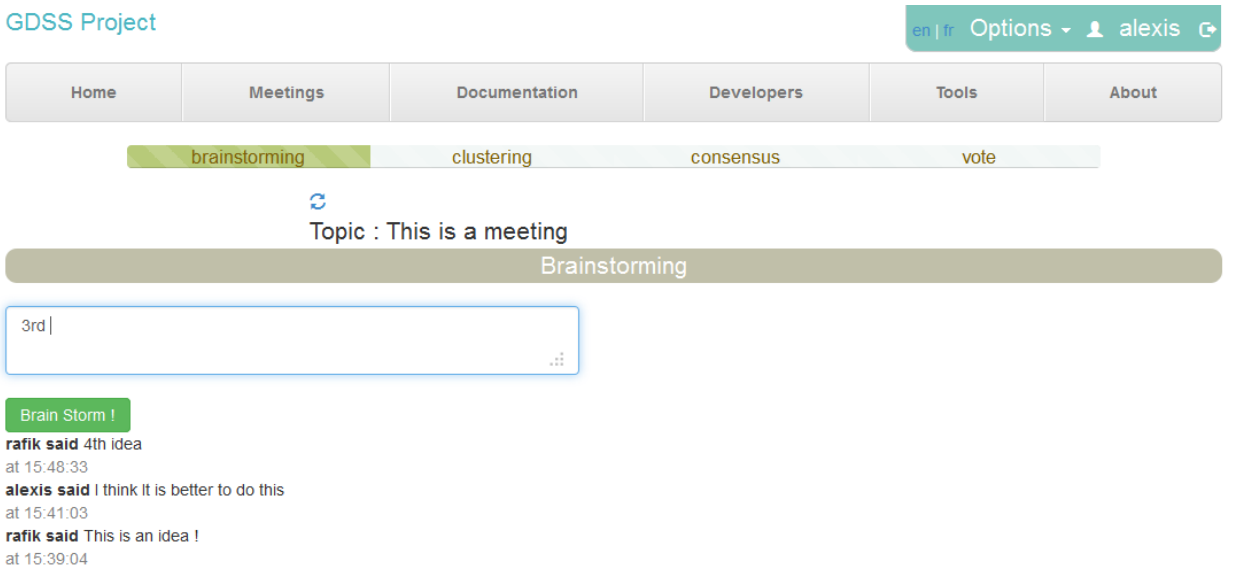
At the left hand there is the list of users who participate to the current meeting, at the right you can see the list of registred users that you may select and invite them by clicking the button « Add user ». Finally press « Save changes ».

4.3 Tools description

We will present here some collaborative tools available by default in the application.

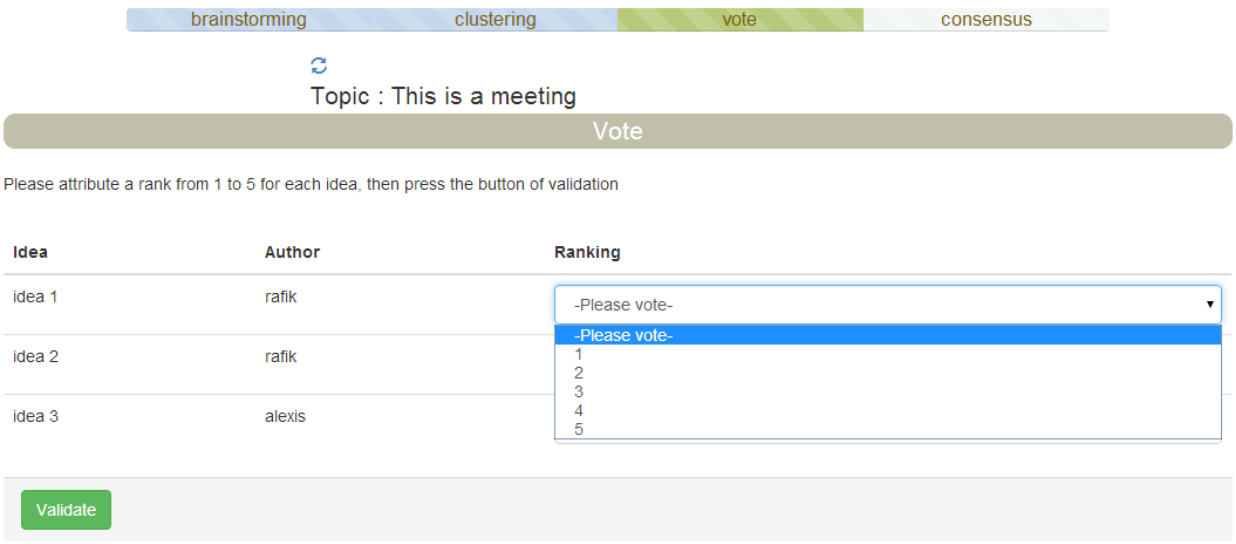
4.3.1 Brainstorming

This is the first tool which appears in a meeting workflow. Users add ideas that the other participants can see.



4.3.2 Vote

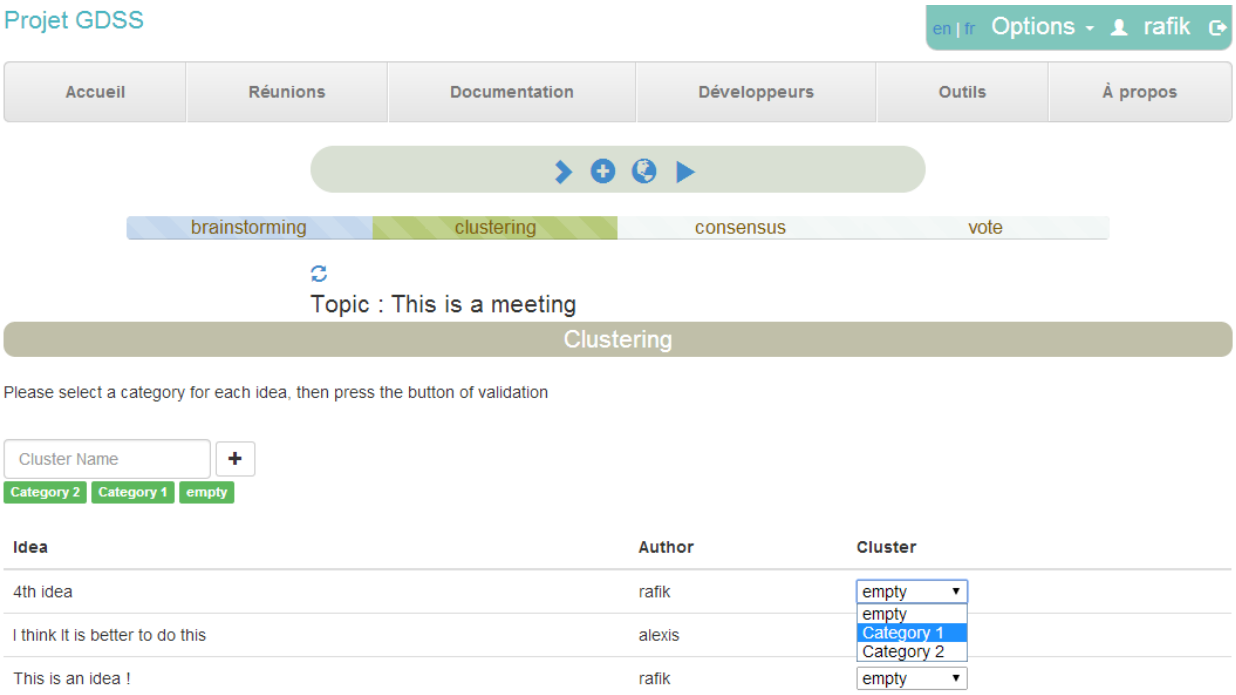
This tool helps to prioritize the brainstormed list of ideas. Normally the facilitator defines type of criteria scale :



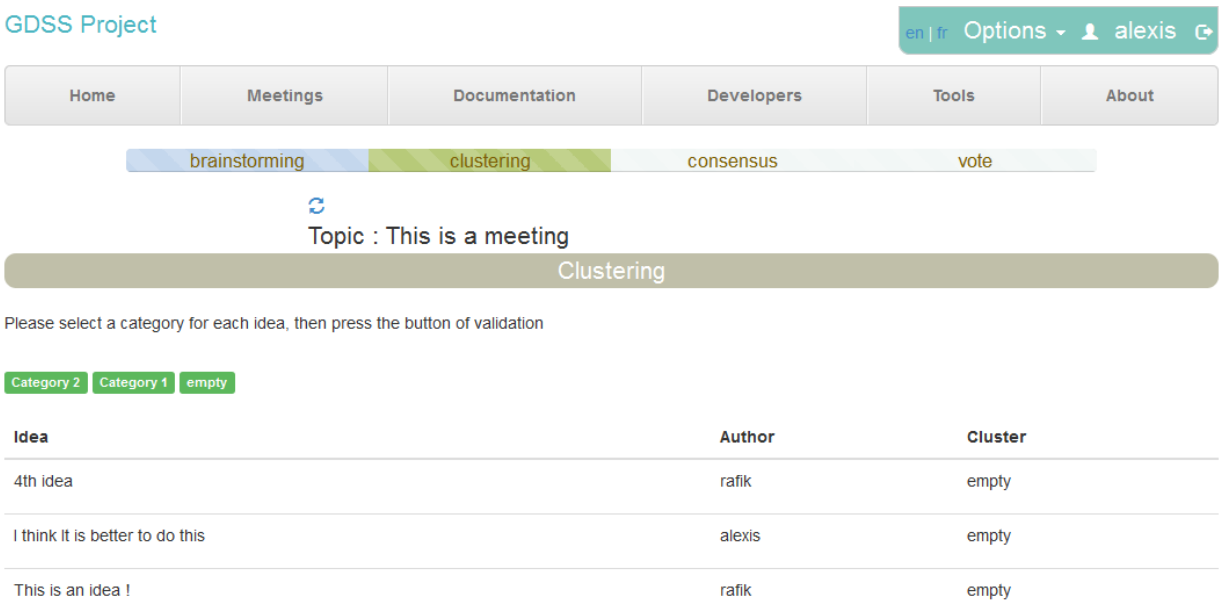
4.3.3 Clustering

Two differents views are available :

- For a facilitator: he defines and add categories, then he can put the brainstormed ideas into clusters.

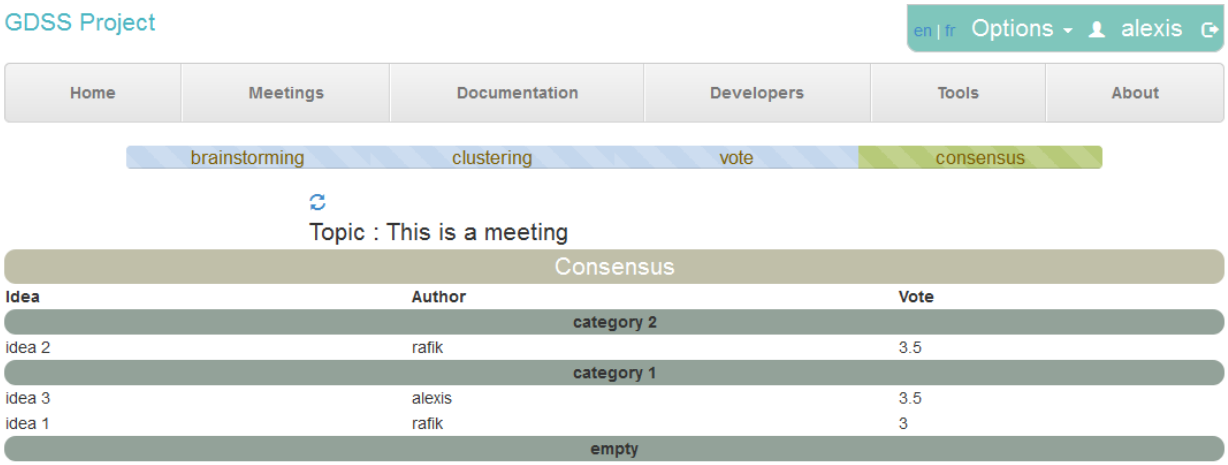


- For a lambda user : He doesn't participate into clustering, he can see clusters of ideas updated in real time while the facilitator is doing his job.



4.3.4 Consensus

This tool is supposed to summarize all the data of the meeting, especially votes. That is why it is recommended to update the consensus tool when a new tool is made. The facilitator can decide to set a new activy of vote or any other tool to bring participants to a consensus



CHAPTER

FIVE

COLABORATIVE TOOLS DOCUMENTATION

Documentation of colaborative tools and how make your own tools.

Contents:

5.1 How to add a new collaborative tool?

Supposing we want to add a new tool called "foo". There is the three necessary step to add it:

- Create the directory grails-app/domain/gdss/tools/foo/ and place your domain classes inside (facultative, only if needed).
- Create a new controller named FooController in grails-app/gdss/tools/
- Create the directory grails-app/views/foo/ and place yours views inside.

Read above for more details.

5.1.1 Create a controller

A controller has to be implemented, here is the main actions :

- start() : called only one time by the facilitator when the tool starts
- index() : called each time a user access to the phase running this tool
- finish() : called only one time by the facilitator when the tool ends

The tool data is stored as a String version of a JSONObject in a object meetingData owned by a object meeting.

5.1.2 Create a view

At least one view has to be implemented. This view will be called after the index() action. It can either use the entire JSONObject inside the meeting variable given by the controller through the

index action or take dynamicly the JSONObject from an specific action in the controller (polling data) using <g:remoteFunction> and JQuery.

Since the JSON structure is better used in javascript, the views are easier to develop using javascript and the tool data.

5.1.3 Tool configuration in JSON

The controller will need two different functionalities :

- Add, modicate or remove elements from the data
- Give the relevant data to the view

Example A

We suppose that the previous tools made data about ideas, comments and votes

Integration of a vote “1/5” created by the user “lambdaUser” on the idea “This is an idea”

```
{
"ideaList":[
    {
        "message":"this is an idea",
        "author":"cool_user",
        "commentList":[
        {
            "messageComment":"comm!",
            "author":"user1"
        },
        {
            "messageComment":"this is a comment",
            "author":"Jeje"
        }]
        "voteList":[
            {
            "rankAttributed":"1/5",
            "author":"lambdaUser"
            }]
    },
    {
        "message":"idea2",
        "author":"user00",
        "commentList":[
        {
            "messageComment":"comm",
            "author":"C"
```

```
        }]
        voteList:[
        {
            "rankAttributed":"3/5",
            "author":"aaa"
        },
        {
            "rankAttributed":"4/5",
            "author":"bbb"
        }]
    }]
}
```

A specific action of the controller should be typically called to add an element like this vote in the JSONObject.

The parsing of a JSONObject is used by the function get(key) returning the value, keys() returning the set of keys accessible and we can add an element by the function put(key,value). For more details, see the Javadoc of the class JSONObject.

Example B

Basic view generation of a tool that summarize the ideas and their comments without exploiting any other data :

1. [Idea example 1 by user4] 1 : comment1 by Anonymous007 2 : commb by Jeje
2. [Idea example 2 by user00] 1 : hello by userA

This is done by going all over the JSONObject and looking for each “message” and “author” in a ideaList and each “messageComment” and “userComment” in a commentList.

5.2 JSON data structure

This document details the actually json key used. You can reuse these keys in your own plugin with respect of their formats.:

```
{
    "ideaList": [                          list of ideas
        {
            "message":                     content of the idea
            "dateCreated":                 date when the idea has been written
            "author":                      author of the idea
            "cluster":                     index number of the cluster that the id
            "voteList": [                  list of votes
                {
```

```
                    "rankAttributed":       content of the vote
                    "author":               author of the vote
                }, ...
            ]
        }, ...
    ]
    "clusterList": [                        list of clusters
        {
            "label":                        name of the cluster
            "clusterCreator":               author of the cluster
        }, ...
    ]
}
```

5.3 Existing plugins

5.3.1 Brainstorming tool

Introduction

This tool is used to generate ideas related to the topic of the meeting. For each idea, the tool captures : * the message which is the content of the idea * the author who has written the idea * the date when the idea has been written

Data

The JSONObjects created or modified by the tool will fulfil this form ::

```
"Topic":
"ideaList":[
    {
        "message":
        "author":
        "dateCreated":
    },
    ...
    {
        "message":
        "author":
        "dateCreated":
    }
]
```

Controller actions

- start(): Create a key named "Topic" and put the topic of the meeting in. Create a key named "ideaList" and put an empty JSONArray in. Redirect the client to index().
- finish(): Redirect the client to the action start() of the next tool controller
- index(): Redirect the client to the index page
- updateIdea(): Call the function addIdea() to add an idea in the JSONArray at the key "ideaList". Returns the data : message, dateCreated and author to reorganize the view
- lastMessages(): Returns all the data at the key "ideaList" from the JSONObject of the meeting
- userList(): Returns all the users during the meeting

Functions

- addIdea(String message, JSONObject json): add the message, the current date and the session user in the JSONObject

5.3.2 Clustering tool

Introduction

This tool is used to create clusters and associate them with ideas that are already been generated. The tool captures : * the label of the cluster * the creator of the cluster * the ideas linked to an existing cluster

In the current interface, the creator of a cluster can only be the facilitator of the meeting.

Data

The JSONObjects created or modified by the tool will fulfil this form ::

```
"clusterList": [
    {
        "label":
        "clusterCreator":
    }, ...
]
"ideaList":[
    {
        ...
        "cluster":
```

```
    },
    ...
    {
        ...
        "cluster":
    }
]
```

Controller actions

- start():Create a key named “clusterList” and put an empty JSONArray in. Add a value the previous JSONArray which is a default cluster with “default” as label and “none” as clusterCreator. Create a key named “cluster” in all the ideas stored in the JSONArray at the “ideaList” key and put “0” as value. Redirect the client to index()
- finish(): Redirect the client to the action start() of the next tool controller.
- index(): Redirect the client to the index page.
- updateCluster(String clusterName): Add a new value in the clusterList which has clusterName as label and the session user as clusterCreator. Returns the data : clusterName to reorganize the view
- lastIdeas(): Returns all the data at the key “ideaList” from the JSONObject of the meeting.
- lastClusters(): Returns all the data during the meeting.
- userList(): Returns all the users during the meeting.
- associateIdea(): Update the value at the key “cluster” of the related idea.

5.3.3 Vote tool

Introduction

This tool is used to rank ideas depending on their popularity amoung participants. The tool captures for each idea and each participant :

- the rank attributed
- the author who has voted

Domain

This tool use a domain “Vote” used to know what vote values are correct.

Properties:

- values: the values available
- labels: the labels associates with each value.

The domain propose by default a rank-scall on five, but can be instencied with, for exemple ::

```
new Vote(values: [1, 2, 3], labels: ["low", "middle", "high"])
```

The domain is used in the view to know labels to show and values to put in option value field.

Controller actions

- start(): Set a new json key « vote » to 1. That allow others tools to know that vote data are available.
- finish(): This method process each vote and compute the average. The average is save in a json key « vote-average » in each idea.
- index(): This function process the idea list and, eventually, the cluster list, and send them to the view in order to render the form.
- validate(): This function add the user vote to the json data structure. For that, a key "votes" is created for each voted idea. This key contain a arry with the voter login as key and rank as value.

5.3.4 Consensus tool

Introduction

This tool is used to show the data that the participants have created since the beginning of the meeting. The facilitator can thus take the ideas and decide to stop or continue the meeting by adding a new activity. This tool does not capture anything, it is just set to illustrate the data

Controller actions

- start(): Redirect the client to index().
- finish(): Redirect the client to the action start() of the next tool controller.
- index(): Redirect the client to the index page.

5.3.5 Report tool

Introduction

This tool is used to show the data that the participants have created since the beginning of the meeting. This tool does not capture anything, it is just set to illustrate the data

Controller actions

- start(): Redirect the client to index().
- finish(): Redirect the client to the action start() of the next tool controller.
- index(): Redirect the client to the index page.

CHAPTER

SIX

CORE DOCUMENTATION

This is the documentation of the core application.

If you just want to add a collaborative tool, you can read the *Colaborative tools documentation*.

Contents:

6.1 Technical choices

6.1.1 Grails

We chose the framework Grails. It is based on the Groovy language, a very high level language like Python or Ruby. Groovy can be compiled to Java Virtual Machine bytecode and can inter-operates with others java codes or libraries. This framework meets well to the constraints :

- It's a high-level framework doing the most things as possible. The idea is to reuse the most existing components as possible. This limits the application self code to maintain.
- It's a JEE application. We chose Grails rather than classic EJB components and Spring framework. We rapidly exclude EJB components as they are not a framework. Spring is also a framework like Grails, but Grails is higher-level.

We used version 2.3.6 of Grails with openjdk 6 or 7 JDK.

Official website: <*http://grails.org/*>

Documentation: <*http://grails.org/doc/latest/*>

Integration with Eclipse: Groovy-Eclipse, Groovy/Grails Tool Suite, Spring Tool Suite (availables in market place)

6.1.2 JSON

In order to save collaborative tools' data, we chose to use JSON. A database has a fixed structure and we can't upgrade a plugin without upgrading the structure of the database to the corresponding

structure used by the new version of the plugin. For one plugin, this is not too restrictive but with several plugins, this creates inter-dependency issues : we need to upgrade all plugin at the same time by supposing they modify the database structure in the same way. JSON doesn't have a fixed structure, plugins can share data just using the same conventions. And if a plugin evolves, we can add easily new key without breaking existing keys used by other plugins. However a common documentation explaining existing keys and its uses is necessary. So, we propose a wiki on the official website for this purpose.

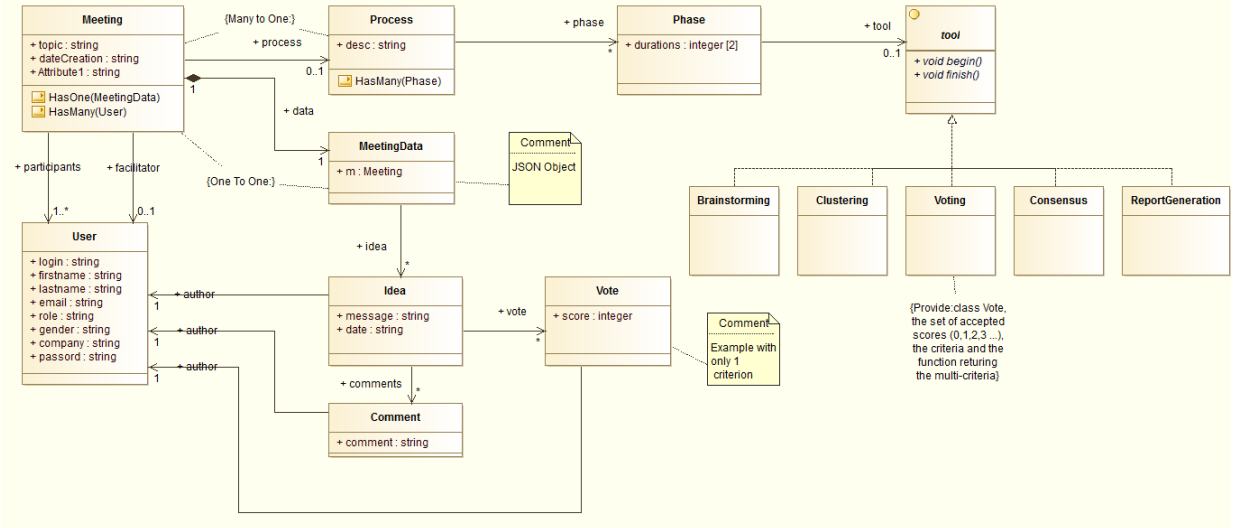
6.2 Architecture

Contents:

6.2.1 UML diagram

In the application core, there are five differents class :

- Meeting class is associated to a meeting with its participants, its facilitator, a meetingdata and a process.
- MeetingData class fits the data which transits between the collaborative tools during the meeting.
- Process class is the registered process with a title and list of tools which will execute during the meeting.
- Tool class represents a collaborative tool which can be chosen by the process creator.
- User class represents a user instance with all the attributes.



6.2.2 Domains

This section lists and documents the domain classes of the core of the application.

User

A user has an email, a last name, a first name, a login and a password. There are two different roles for a user:

- admin: he can manage all the users and has access to specific actions.
- user: it's the ordinary user.

Meeting

This domain represents a meeting. It has a topic and a description, some date information (created, started), a eventually duration (in order to show remaining time), a facilitator and other users as participants. A meeting can be public or private. If the meeting is public, there are no participants. The meeting appears on the meeting list for everyone, even if people are not connected. However, you can join to the meeting only if you are connected.

Only the meeting facilitator or an admin can delete a meeting.

MeetingData

This domain class is used to save tools data.

Attributs:

- *json*: This string is used to save JSON data. Read more about *JSON*.
- *parameters*: This string is used to save tools parameters.

Process

A process is a ordered list of tools. It is essential for a meeting in order to define its steps. A process can be shared between two meeting. This allow you to define one process and reused it for several meetings.

Attributs:

- *title*: Title of the process as it is show in new meeting form.
- *visible*: Default is true. During the meeting, if you modify the process, it will copy in a new process with visible argument to false. The new process is not visible : you can't choose it when you create a new meeting.

Only an admin can delete a process. However, it will just put the visible argument to false. Moreover, only an admin can see a non visible process in the process list.

Tool

When you create a Process, you need to choose your tools that are represented by this domain class. Tools objects are automatically created by the process controller by detecting tools controllers.

Attributs:

- *title*: The tool title. Automatically detected from controller name.
- *description*: The tool description. Not yet used.

6.2.3 Controllers list

This section lists and documents the controllers of the core of the application. For all controllers, the related views respect the grails convention : the views are in a subdirectory with the same name as the controller.

HomeController

This controller handles static web page like home page, tools description, about page, etc...

Actions:

- *index*: The home page, reachable from the main menu.
- *tools*: Tools description, reachable from the main menu.
- *about*: About page, reachable from the main menu.

ProcessController

This controller is almost to scallfold controller for Process domain class.

MeetingController

This controller is almost to scallfold controller for Meeting domain class.

DefaultController

The main controller for managing a meeting. It implements most of the actions facilitator for the meeting such as updating the process, invite or remove participants during the meeting. It also allows the facilitator to move from one tool to another or terminate the meeting.

Actions:

- *openMeeting*: Provided a list of public meetings and private meetings in which the user is participant or facilitator
- *meeting*: Displays details of the current meeting
- *startMeeting*: Lets start a meeting
- *accessMeeting*: Allows to access a meeting
- *nextTool*: Moves to the next tool
- *updateProcess* : Rendering the page that allows to update the process of meeting
- ***validateProcess*: Validate a new process for the current meeting** @param params.process : new list of tools
- *updateUsers* : Rendering the page that allows to update the list of participants
- ***validateUsers*: Validate a new list of participants** @parm params.users : new list of participants
- ***searchUsers*** [Search users whose Login contains params.word] @param params.word : word to find
- ***poll*** [Long polling][Allows a participant to wait for moving to the next tool.] @params params.login : particpant Login
- *allNotification*: show a notification

UserController

This controller is very close to the scallfold controller for User domain class. Some functions was added to permit users to modify their profile.

6.2.4 Filters

Filters all access to meetings of the application. For example, it checks that the meeting to which we try to access exist and the access rights (participant, facilitator).

SecurityFilters

- startAndFinishFilter: filter all access to Start and Finish actions of the tools. Access to these shares is restricted only to the meeting facilitator.
- toolsFilter: filter all the access to all tools actions except Start and Finish actions. It ensures that users who access a tool are invited to the meeting and that it is the current tool of the meeting.
- agendaFilter: Filter all the access to specific actions to the meeting facilitator. The actions such as nextTool closeMeeting, updateProcess ... can only be used by the facilitator of the meeting.

6.2.5 Bootstrap

There is a special file executed at the beginning of the application. This file is *grails-app/conf/BootStrap.groovy*.

There are two action done in this file :

- Creation of the administrator account if it doesn't exist.
- Creation of the tools automatically by detecting existing controllers.

6.3 Configuration

The configuration values specific to this application can be found at the end of the file *grails-app/conf/Config.groovy*.

First, you have documentation configurations.

- **gdss.doc.uri**: The base uri for the documentation. This uri is used in main menu, for user documentation (adding "/user/") and developer documentation (adding "/core/").

Next, you have admin configurations. The login and the password are used to log in as an administrator in the application.

- **gdss.admin.email**: The email of the admin user
- **gdss.admin.firstname**: The first name of the admin user
- **gdss.admin.lastname**: The last name of the admin user
- **gdss.admin.login**: The login of the admin user
- **gdss.admin.passwd**: The password of the admin user

6.4 Known issues

This section is a non-exhaustive list of known issues that would be fixed in forthcoming version.

- When you add participants in a meeting, you can add several times the same user.
- Sometimes, the automatic refresh in a meeting doesn't work.
- An error appears if a meeting starts with an empty process.
- In the meetings list pagination, if you go to the next meetings in the private meetings, it will do the same for the public meetings.
- Some complex notifications are not translated in french.

6.5 Possible improvements

This section is a non-exhaustive list of possible future improvements.

- Tools parameters are actually saved in MeetingData. It seems a good idea to save its in the Process in order to give the same settings when you create a new meeting from a existing Process.
- Allow saving a process modified during the meeting.
- Refuse to add the same idea twice.
- Permit to deal with several brainstorming and associate clustering, vote, etc. For that, we imagine split the JSON tree in a dataSet array, with a new entry for each set of data. A tool (like vote, etc) can works on any set, but, *a priori*, works on the last one.
- Show the number of person who voted for the facilitator.
- Permit to modify his votes :
  - Add a button « modify » with the validation message
  - Select automatically the note selected precedently
- Order vote results in consensus and reporting.

6.6 Deployment

In order to deploy the application on a "cloud server", you can refer to the official documentation at http://grails.org/Deployment/.

The documentation is not a part of the grails application. If you want to propose the documentation under the */doc/* path, you need to configure your webserver for that.

Here is an example for the apache web server. We suppose that the html version of the documentation generated by Sphinx is in the */srv/www/easymeeting/documentation/* directory, and that the grails application listen on port 8080.:

```
<VirtualHost *:80>
    ServerAdmin admin@easymeeting.org
    ServerName easymeeting.org
    ErrorLog "/srv/www/easymeeting/log/error.log"
    CustomLog "/srv/www/easymeeting/log/access.log" combined
    RewriteEngine On
    RewriteRule ^/doc/(.*) /srv/www/easymeeting/documentation/$1 [L]
    ProxyPreserveHost On
    ProxyRequests Off
    ProxyPass / http://localhost:8080/
    ProxyPassReverse / http://localhost:8080/
</VirtualHost>
```

CHAPTER

SEVEN

CONCLUSION

The presented version of EasyMeeting system partially met our main objectives. This application provides full range of basic functionalities required for the realization of group decision-making sessions. In addition, for a participant or a facilitator, it is quite simple to use. It provides a graphical interface facilitating a quick start, and it is independent of any research theory and approach.

Nevertheless, the integration of collaborative tools should be improved. Indeed, in this version, it is not possible to plug "in-live" new collaboration tool. In addition, it is interesting to make the core of the application more modular. For this, we propose to use the concept of web service. Indeed, web services technology would easily make a generic module plugins. It would also be interesting to re-design the core of application in the form of web services. In this way, it would become more modular and it would be possible to connect EasyMeeting (or part thereof) in an existing information system, and to deploy it in cloud servers.

BIBLIOGRAPHY

[1] Robert O. Briggs, Gwendolyn L. Kolfschoten, Gert-Jan de Vreede, Conan C. Albrecht, and Stephan G. Lukosch. Facilitator in a box: Computer assisted collaboration engineering and process support systems for rapid development of collaborative applications for high-value tasks. In *HICSS*, pages 1–10. IEEE Computer Society, 2010.

[2] Tanja Buttler, Jordan Janeiro, Stephan Lukosch, and RobertO. Briggs. Beyond gss: Fitting collaboration technology to a given work practice. In AdrianaS. Vivacqua, Carl Gutwin, and MarcosR.S. Borges, editors, *Collaboration and Technology*, volume 6969 of *Lecture Notes in Computer Science*, pages 126–141. Springer Berlin Heidelberg, 2011.

[3] Gert-Jan de Vreede, Douglas R. Vogel, Gwendolyn L. Kolfschoten, and Jeroen Wien. Fifteen years of gss in the field: A comparison across time and national boundaries. In *HICSS*, page 9, 2003.

[4] Gerardine DeSanctis and R. Brent Gallupe. A Foundation for the Study of Group Decision Support Systems. *MANAGEMENT SCIENCE*, 33(5):589–609, 1987.

[5] Gerardine DeSanctis, Marshall Scott Poole, and Ilze Zigurs. The minnesota gdss research project: Group support systems, group processes, and outcomes. *J. AIS*, 9(10), 2008.

[6] Frost and Sullivan. Meetings around the world: The impact of collaboration on business performance. 2007.

[7] S.W. Knoll, M. Horning, and G. Horton. Applying a thinklet- and thinxel-based group process modeling language: A prototype of a universal group support system. In *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pages 1–10, Jan 2009.

[8] A. Mametjanov, D. Kjeldgaard, T. Pettepier, C. Albrecht, S. Lukosch, and Robert Briggs. Arcade: Action-centered rapid collaborative application development and execution. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–10, Jan 2011.

[9] J. F. Nunamaker, Alan R. Dennis, Joseph S. Valacich, Douglas Vogel, and Joey F. George. Electronic meeting systems. *Commun. ACM*, 34(7):40–61, July 1991.

[10] Amy Soller, Alejandra Martínez-Monés, Patrick Jermann, and Martin Muehlenbrock. From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. *I. J. Artificial Intelligence in Education*, 15(4):261–290, 2005.

[11] Pascale Zaraté, Jacqueline Konate, and Guy Camilleri. Collaborative Decision Making Tools : A Comparative Study Based on Functionalities (regular paper). In Bilyana Martinovski, editor, *Group Decision and Negotiation (GDN), Stockholm, Sweden, 17/06/2013-21/06/2013*, pages 111–122. Department of Computer and Systems Sciences, Stockholm University, juin 2013.