Design and implementation of the interface "gAn Web"

Author:
Andrea G.B. DAMIOLI

Abstract

Design and implementation of the interface "gAn Web"

by Andrea G.B. DAMIOLI

The AEgIS Experiment at the CERN aims to verify the weak interaction principle for antimatter. This document talks about "gAn Web", a web application designed to simplify the analysis of physical data under the AEgIS experiment. This analysis can be performed using Root Data Analysis Framework by the Linux Terminal using an application named "gAn", but a graphical interface can ensure a better user experience, ease the user training and improve the productivity. The goal of "gAn Web" is to play the role of graphical interfaces between the human user and "gAn". This document analyses from the Human-Computer Interaction's point of view the production of this graphical interface.

Contents

List of Figures

Chapter 1

Introduction

The goal of this chapter is give some general information about what the application does and how does the AEgIS experiment works (for that is related to this application).

1.1 User friendly Data analysis: gAn Web

GAn is a program that aims to analyse data related to the AEgIS experiment at the CERN, gAn Web is the web interface of gAn, and it is the main topic of this document.

This program receives in input (from another software system) some terabytes of files in ".root" format, and some input parameters inserted by the user; It does an analysis and it gives in output some summarized scientifically interesting information, understandable by humans.

For that regards the file in ".root" format: A file ".root" is a file produced by a variegated group of sensors in a complex machine that accelerates particles and lets them crash together. These sensors produce data continuously (8 hours per day), and a software system validates and saves these data in file with .root extension.

The input parameters inserted by the users are:

1. "Run Number", that identifies in which part of the data the user is interested. The time in this experiment is divided in "runs" (a run lasts about some hundred seconds), so the user, by the run parameter can tell to gAn in which time slice he is interested. For example: run 55614 means that the user is interested in the information related to the 22th of November 2016, taken in the time slice between 15:45 and 15:47. The enumeration of the runs is incremental: so the run 55199 identifies the time

slice immediately after the time slices identified by 55198. The progress of the run numbers is always going on, so new run numbers are continuously added. In this document every time we say "last run number" we don't mean the absolute last run numbers, but the run number that is going on right in this moment. This concept is important because usually in most cases the users use gAn Web on the last existing run number (so on the run number that is going on right at that moment) or on the immediately previous run number. This system used to identify time slices seems to be strange at the beginning but is a standard for all the applications in the AEgIS experiment and it allows a very efficient and precise communication.

How can the user have more information about the runs? On another server exists a RunLog (sometimes also called LogBook). The RunLog is a document organized by date on which at each run a group of people write the run number, some informations (about settings), some observation (at maximum 2 rows, usually one.. so usually brief observations [but if in the run something particular happens, in that rare case the observations can become very long]). Is this document an input or an output for gAn Web? it depends: If the user uses gAn Web on the last existing run (or the run immediately before) probably the output of gAn Web is one of the (many) inputs of the RunLog. If the user is interested in a phenomenon happened some days ago, probably he searches on the RunLog at the page related the date of the phenomenon, he chooses one or more runs related to this phenomenon (he recognize them by the observations), he works with gAn using these runs, and probably he notes parts of the output of gAn Web as observations on the RunLog, that in this situation is both input and output. Some runs are absolutely not related with gAn Web, so in some cases the RunLog is neither an input nor an output.

2. "Type of analysis", that identifies what the program must do with the data and what it must show as output to the user. A "type" is a way in which the program extracts information from the raw data. Each type can extract different information using different parts of data and elaborating them in different ways. For example: a type of analysis named "Tmeas" can extract information about the temperatures of some elemental particles analysing how quickly they move subjected to a force.

The .root files can be analysed using a framework named ROOT

Framework, that consists in a lot of libraries specialized in high-energy physics analysis, and an interpreter able to understand a C++ script. Actually, gAn is the sum of the Root Framework plus a lot of C++ scripts. The goal of gAn is to reduce the big amount of raw data received in input in a little amount of scientifically interesting and easily understandable data in output. To do this it has to filter data, understand which of them are scientifically interesting, choose the parts that are related to the run selected by the user (by the run parameter), elaborate and compare them, and make advanced statistical analysis on them. GAn can be called using a common linux terminal, using a command with parameters.

The output of gAn consists of a single text file with computed, (quite) organized data, and a folder of images in png format. This structure (root files in input, data analysis using Root, images, organized and selected data in output) is very common in the CERN's experiments. The output of gAn is quite understandable by an experienced physicist, but it is disorganized, complex for an untrained user, and the terminal interface can be surely improved using some more user friendly technologies.

GAn Web is a web application, that aims to create a user friendly web interface, based on the most important human-machine interaction principles, between the users and gAn. A web interface can improve the system in two ways:

1. gAn is a stand-alone program based on Root, installable on the user's machine; the user has to install the correct version of Root to avoid compatibility problems (Root is still not perfectly version independent: different versions can lead to different behaviours). Furthermore, this kind of program is continuously changing, the performed analysis is continuously improved (in the first 2 weeks from the debut of the program there are already several different kinds of analysis, because often at the changing of the needs the programmers creates new generations of analysis), so the installed version of gAn is not final and unchangeable, and the user must often update it. Instead, a centralized version installed on a server, with services accessible from a normal browser by the user can avoid (at least reduce) this kind of problems and be more usable.

2. a Linux terminal interface is practical for expert users, but a web based interface can be more attractive for new users, and, if well done, can be easier to use. It is important to notice that

the users need to hardly exploit their memory to use correctly a terminal application, an we want try to make their work easier.

The goal of gAn Web is to allow users to do analysis through a more friendly web interface, without install nothing on their machine. In the following image there is a schema that shows how this program is organized.

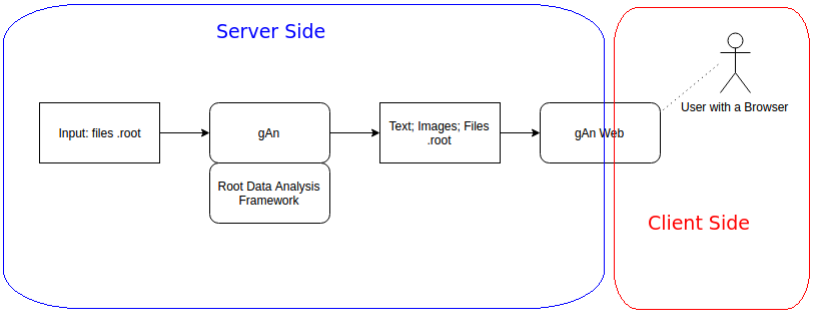

FIGURE 1.1: gAn - gAn Web simple scheme

Chapter 2

General organization of the project

We can firstly explain how this project is organized:

2.1 The roles of the designer and of the users

In this design process the interaction between designer and users is very strict: in fact for some months, divided in blocks of some weeks, the designer has worked together directly with the users in their standard work environment, both during the development process and after, to test the result.

A fundamental point to understand is that the difficulties of this project are not necessarily all related to the technical part: the most difficult part is to understand how an ideal interaction takes place, what are the needs of the users, how can the application be really helpful for them.

It is interesting to note that the domain in which this application works is quite complex and very different from the domains in which the developer is specialized, so, some periods of work directly with the users are necessary to let the developer understand better how does the domain work. This period of time allowed the designer to understand better how exactly the work processes take place, and what are the needs of the users. It is important to precise that this kind of users is quite particular: their education level is very high (almost all of them have a Phd), and their needs are very specialized (related with the physics research world).

In this design process we can identify 3 main actors:

1. The developer, who produced the web interface (gAn Web), and wrote this document.
2. Two super users (university professors in Brescia), that are also co-developer of the application behind this interface (gAn). So they act three roles: the role of the users (in particular they checked the application at every stage, and also before the general tests with the user, so they are pilot-users), the role of the supervisors of the global project, and the role of co-developer (as discussed after in this document the user in this project is participatory, so is an active part of the design team). They are physicist so they can play a "jolly role" creating an absolutely important bridge between the physic domain and the software engineering domain.
3. A group of generic users (around 20) in Geneva. This main group is a resource for the final testing of the application, that is designed on their needs.

In this project the user is not only a customer: the user is participatory. In particular the two super-users are consulted to take every important decision in the project, so are co-developer. Also the generic users have an important part: the tests with the users reveal problems, and some of the user are source of ideas, both during the production of the application and after, during the normal use of gAn Web.

2.2 General organization of the design process

According to the best practices of the Human-Computers Interaction science this project follows the star-shaped life cycle: the development is a group of activities that aren't carried on just one time but repetitively in all the life of the project. The most important activity is the evaluation of the product under the usability and the feasibility points of view. Around this central activity there are activities related to find the real needs of the users, select the functionalities to implement, prototyping, development, test.. All the activities are repeated more times, as explained in the following paragraph.

2.3 The stages of the design project

The design process can be divided in three stages:

1. An early stage, more simple, with basic functionalities, just to investigate what are the best ways to implement the functionalities and to test with a little group of super-users (two) if this software can really be useful and which functionalities are really important; The goal of this stage is to give an initial direction to the development and to improve the developer's knowledge of the domain. At the end of this stage, the confrontation with the two pilot users helps to correct the way.

2. An intermediate stage, more complex, with advanced functionalities obtained listening the request of the super users. The designer needs at this point to execute some tests with the maximum possible number of users to understand if the product is acceptable and useful (and how to improve it).

   In this stage the test with the users are splitted in two blocks: The first block is another test with the pilot users, that are present in all the stages (they are, as well as users, also supervisors), and their hints are applied before let the group of generic users work with the application; The second block is a test with a big group of users that works with the application in the AEgIS Control Room (so, in a real work situation).

   The tests with the users show a lot of useful information and allow the developer to find a big group of problems: the solutions to these problems give the birth at the final version.

3. The third stage is the final stage. At this point the application is modified to accord the observations of the users (we note that the direct impact of the users leads to a lot of modifications). It is important to notice that the third stage (the last one) is a never-finished stage: the needs of the users are permanently changing and evolving, new ideas and proposals still come from them, the application is designed to be adaptable, and to try to satisfy the unknown needs of future users, and unavoidably it needs continuous upgrades and modifications.

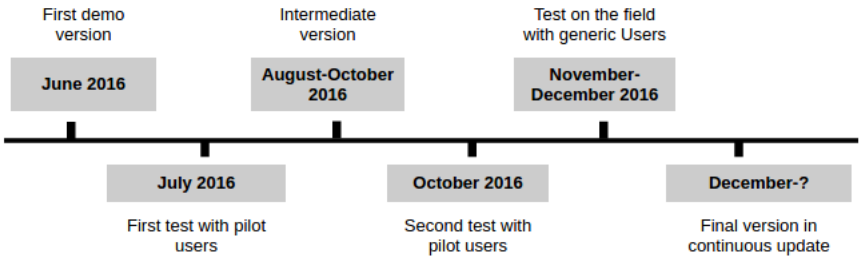

FIGURE 2.1: An overview of the timeline of the project

2.4 The role of aesthetics

The role of the aesthetics is not the most important to ensure an interaction that works well, but surely we cannot neglect it. In this project the role of aesthetics is cured partially in the intermediate version and fully in the final version, immediately before tests this application with the final users: In the first and intermediate versions we can observe some overlapping surfaces, some asymmetrical figures, and some others aesthetic problems, because in the design process we thought first at the functionalities, but from the third version the interface is surely more pleasant and cured, because the aesthetics plays an important role in the satisfaction of the users.

2.5 Analysis with inspection methods and Tests with users

These are methods to try to judge the application (if necessary more than one time) during the developing process. There are two different situation:

1. For that regards the inspection analysis we must note that the evaluator is only one, and he is also the developer. This can be a big limit because only one evaluator can find only a little part of the problems. Also, the assessor is the same person who

is designing this interface; accepted that this kind of analysis will not be the best source of feedbacks about the application a good solution is to check the various parts of the application against the Nielsen principles in the meanwhile these parts are designed and created.

2. For that regards the opportunity to test the application with the users instead the situation is excellent: we have a good amount of users, they are very accessible, their roles and characteristics are known and precisely defined, and we also have two super-users that can help driving the design process into the good direction (actually these users play a role that can be considered also a designer-customer role). At this point we can hope that the confrontation with the users is the best available source for new ideas and a way to find and understand problems.

We note that in this project the evaluation is mostly a "formative evaluation", and only in the last stage it becomes a "summary evaluation". This is because the confrontation with the users is the only intelligent way to understand in which direction work and to enter in their "domain". So the formative evaluation, both from the two super-users, and from the generic users, is probably the most important source of requisites and solutions.

2.6 Prototyping and mock-up

In this project the use of mock-ups is an example of interactive mock-ups. All the mock-ups are produced directly with HTML-scss-javascript-Php so they are real part of the website. In that cases they are mock-ups only because some functionalities related to gAn on the back-end are still not implemented at the moment of the creation of the front-end. In the final version almost all of the website is real and working. Does it means that the application is ready and finished? No, it doesn't: still the server is not working properly (we are pretty sure that we can solve formatting it and re-configuring some of the applications installed on that server) and only some of the analysis prepared for the software are ready (the output of the others is actually a mock-up). The aim of the decision of working with interactive mock-ups directly produced by web programming technologies is related to the fact that the developer is very inexperienced using mock-up software but quite used to work with web development, so working directly with the web development seems to be a quite fast way, and the interactive mock-ups are perfect to be used to show the

progresses of the application to the super-users. Often the problem of interactive mock-ups are that the developer is reluctant to modify them.. and it is true, but in this case the continuous modification of the application through the different development stages is very important to arrive progressively to a good solution able to meet the users needs. In fact we can see that the system in the different stages is very different, actually at each iteration the production of the website re-started almost from zero.

2.7 Ambiguities and doubts

Every time in the design process there is a doubt related to a requirement the adopted solution is to ask directly to the users what solution to implement. This means ask before to the super users (for motivations related to their role of supervisors of this project) end possibly observe the behavior of general users during the tests.

2.8 How the various stages are presented

According to this division in three main stages we can organize the following chapters in this way: each chapter is related to a stage of the development: Early stage (3), intermediate stage (4) and final stage (5). Also there is a chapter able to describe the tests with the users, so it is strictly related to the chapter 5: it is chapter 6 (6). For each of the three stages of the development this document will explain:

1. The requirements of the web interface.
2. The functional analysis of the requirements through some expected scenarios
3. The resulting prototyping (and implementation)

Chapter 3

First version

This chapter analyses which are the requirements of gAn Web at the first version and how they are implemented. This first version is very incomplete and aesthetically very raw, but very useful because it is meant to be a "test" to understand how to implement some functionalities and a base from which start to discuss with the super-users.

3.1 Functional requirements

The definition of functional requirements aims to specifying in detail what the web application can do, and in which way an user can use it. The development of gAn Web is divided in three stages, so the requirements are peculiar and different for each stage. The first version (from now "gAn Web v1") is a very simple application: instead of access the program by a linux terminal like gAn, in gAn Web v1, the user can use the program through a graphical interface. The requirements of this version are the following:

1. The user, in the homepage, can choose the run (only one run, for the moment) in which he is interested, using an input field. This field has a validator, able to understand if the run number is inserted, if it is effectively a number, and if it is in an acceptable range. The user receives an explaining and precise error message directly on the homepage if the input field is empty or if the inserted value is not acceptable. Is it always possible validate the inserted run and ensure that the related Root file exists? No, because in some moments some sensors don't work properly and it is inevitable that some root files related to some runs are incomplete, or even in-existent. This problem will be solved in the next versions.

2. At this moment there is only a generic idea about what kind of analysis can be useful for the user, so in this version there will be only a generic analysis, able to dump in text all the possible information, and a group of example images (this is useless for the goal of the scientific research, but very useful to improve the understanding of the needed features of the web interface). At this stage is not clear if the type of analysis will be chosen by the user of automatically selected by the program , so in gAn web v1 there are no buttons able to allow the user to select the type of analysis (this point will be reconsidered in next versions).

3. The user can start the program with a single click, by a button (usable only if the inserted number is valid).

4. When the program is executed the user can see the text output on the screen. This text is clear for a physicist (it is not clear for a person who doesn't have a specific preparation). This is an important point: there are big differences between the domain of the software engineering and the physics, so is better that this textual output is written by the hand of a physicist, to ensure to avoid misunderstanding due to different meanings assigned to symbols in different domains. Improve the legibility of the output will be a requirements for a more advanced version.

5. When the program is executed the user can see the output images by clicking a button that link to a images-page. The images are ordered and organized by groups (the groups are related about which sensor takes the information necessary to create the image). The user can decide if he prefers to see the image in a little, medium or big format. The user can also decide if the images are distributed in the screen vertically or through a "carousel layout". The user can access the image in full-screen by clicking on it: he is redirected to a page with the image shown in full screen, and can return back to the all-images page by a return button. It is absolutely important to understand exactly which information are interesting for the users, and show in the images only them and all of them, this point will be solved with a confrontation with the pilot-users.

6. The user can modify a configuration file (a .txt file on the server), by a web interface. In this files there are some values the need to be setted (otherwise it uses default values), and the user can do it by radio buttons (in this way he is forced to choose valid values). This configuration file can modify the way in which

gAn works and modify the resulting output (both the text and the images).

3.2 Non-functional requirements

This version (and actually also all the next versions) has some non-functional requisites:

1. The first is quite simple: gAn Web has to ensure that in case of crash of the program the web server mustn't crash too. The point is that on this web server (Apache server, installed on Linux) there are some other important applications, so, if gAn web crashes it is not a big problem, but the crash cannot force Apache, or worst the entire machine, to stop or restart. This requisite is quite easy to meet: a modern web application based on Html, Javascript, PHP and CSS is quite safe, a general crash of the server it is very unlikely to happen. If the C++ application or some Root libraries crashes (for example if the user asks for an inexistent run) the web application gracefully warn the user about the problem, but without uncontrolled behaviours.
2. The application must work without install nothing. Also this requirement is very easy to meet: gAn Web is a web interface, it requires only a browser, nothing else.
3. The application must be compatible with any machine (except mobile phones, not requested), regardless of hardware, operating system, installed software. Also this requirement is achieved because of gAn Web only needs a browser to be used. A good observation is that the application is typically used on the computers of the AEgIS control room, that have a quite big screens, but to be sure is better to create an application able to adapt itself also to (quite) little screens.
4. The application must be easy to be modified and extended in the future by persons who aren't necessarily software engineers. The point is that the student who wrote this program is a "momentary collaborator" in the AEgIS experiment, and all the modifications to the program must be done by other people, in most cases physicists. So the best way is to comment in detail the code and keep the code simple (this is a basic good-programming requirement).

3.3 Scenario based functional analysis

Following there are a list of scenarios in which a user achieves a goal by doing a list of steps. The goal of these scenarios is to show in detail how the interaction between the user and the system takes place. In this first version, the complexity of the scenarios is very low.

1. The user in interested in the run 31111. In particular he is interested in the peak (the highest value) reached by a sensor named Mimito (this kind of task is very common). At this stage we still not work with kinds of analysis. The user, opened a browser in the homepage, insert the run number and push "Send". He waits some seconds (there is a progress bar) and he arrives in a textual output page. At this point he can search in the text the information in which he is interested (it is a numerical value). Probably he is interested also in a image, to understand spatially where this peak is: he clicks on the "Show Images" button, he goes in the images pages, he selects the group "mimito", and the browser shows him a png image showing an histogram in 3 dimension: on the z-axis there is the peak, he can understand in which place (identified by x-axis and y-axis coordinates), the peak takes place, and how is shaped the resulting 3-d figure. From there he can return to the homepage and do another task.
2. The user is no more interested in Mimito, but in another sensor: Faraday Cup. The user wants to know what are the values detected by the sensor in the same run (also this scenario is very common, often when the result of a sensor is unexpected, checking the others is interesting). This other sensor is not automatically enabled, so the user goes through the "Edit Configuration" button in a page able to modify the configuration file. Here the user can see a list of sensors, and he can use radio-buttons to modify their values from no (==disabled), to yes (==enabled). The user enables all the sensors, confirms the changes, and automatically returns in the homepage. Now he can like before insert the number, and get the outputs in textual and in images format.

3.4 Prototyping

This version is very simple and it contains only the basic functionalities. The goal of this version are: to understand if this kind of

system can be useful for the users, and to experiment some technical solutions to create the features needed to achieve the goals.

To create a prototype in this cases the web programming is preferred on programs able to produce mock-ups, because the designer is quite expert in web development and can produce it quite rapidly, but he is not used to produce mock-ups with specialized software so the production of mock-ups would be slower than the complete development (actually, quite complete).

Following are visible some part of the first prototype, with very simple features.

1. The homepage:

FIGURE 3.1: First version: the first homepage of gAn Web

   It is quite clear: There is the AEgIS Logo, an input field where the user can insert a run number (only one in this version) and a "Send" button to start the analysis (using gAn). A button "Edit Configuration File" allows the user to enter in the page dedicated to the configuration of the program.

2. The text output page:

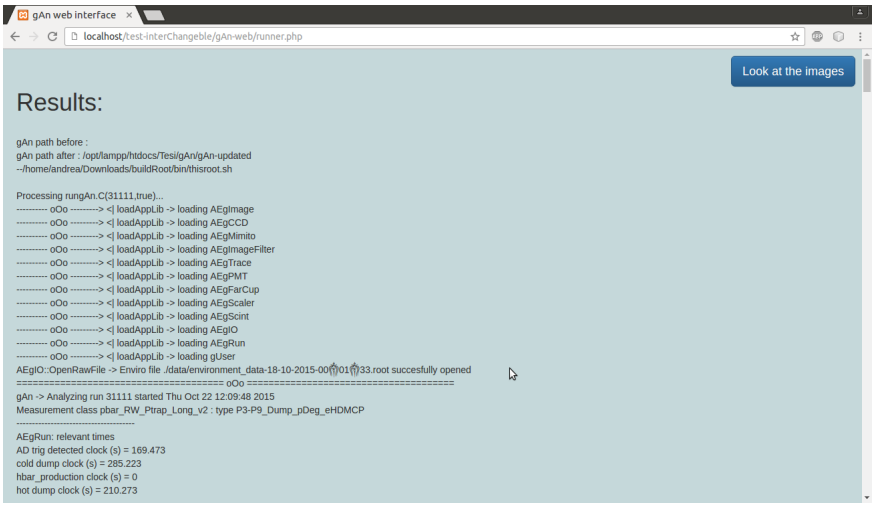

FIGURE 3.2: First version: the page related to the textual output of gAn Web

The textual result of the computation is visible: it seems to be too long and incomprehensible, but for physicists it is quite clear; to understand what is the best way to format this output a further confrontation with the users is needed. The graphics is very minimalist, there is only one button: "Look at the images", that sends the user to the page related to the images.

3. The page related to the images:

FIGURE 3.3: First version: the page able to show the output images in the early prototype

This page shows the images in a dynamic framework, that the user can edit. The user can choose by dropdown menus the dimension, the layout ("vertical", if he prefers the images disposed vertically one above the other, "carousel" if he prefers the images organized horizontally, navigable by a "next" button and a "previous" button), the group to show (each image belongs to a group, each group usually is composed by 2-3 images). Clicking on a image the user can open it in a full page version (but it is still a static image, a png).

4. The page that aims to edit the configuration file:

FIGURE 3.4: First version: the configuration page

This page allows the user to choose by radio buttons (modified using Bootstrap graphic) the value to insert in the configuration file of gAn. Radio buttons force users to insert correct values.

3.5 How the early version can be improved

The early version's goal is just to be a demo. In particular, it is based on the assumption that the user knows in every moment all about gAn (how does it works, what is the meaning of each field of the configuration file, etcetera). It can be improved literally in every point, according with the principles of the Human Computer Interaction.

Chapter 4

Intermediate version

In this chapter the features of the intermediate version are exposed.

4.1 Functional requirements

The intermediate version (from now "gAn Web v2") is more complex than the first one. It was born from the tips and the observation of the pilot users. The modifications are not numerous, but there are a lot of additions of new features. In this version we start to think more deeply to what functionalities are more important and how we can let the accessible by the users. The aesthetic is still not very considered at this stage (it will become important in the next stage). All the new required characteristics are exposed following:

1. The user can insert multiple runs: separated by a semicolon (but in case of errors the system can automatically correct them replacing symbols like "-" or "," or "." with semicolons and giving a more robust service). These runs can be inserted by an input field of by a range select button: this button opens a "modal" that allows the user to choose the first run and the last, and automatically insert the comprised runs (for example, if the user inserts 30000 and 30010 the system inserts automatically all the run numbers between 30000 and 30010). This modal has a validation system, that ensure the correctness of the inserted values. It is not perfectly clear if this solution fits the needs of the users, but the tests with the whole users group will probably solve this doubt.
2. The user can choose which kind of analysis execute. At this point the different analysis are related to the different branches of gAn that at this stage an heterogeneous group of programmers are developing and uploading on Gitlab. It is no clear

which of these branches will be definitive and which no, so the program must be able to use all of them. The type of the analysis depends on the version of gAn downloaded and used for the execution of the program. In gAn Web v2 five complete branches exist, but in the future they can became more. They are externally very similar, the differences are the algorithms in the program, but they give a different output (different output but in the same format: text and images). At this stage is not clear if all this different versions will be used for the final application, to clarify this point the best solution is observe directly the user's behaviour.

3. The configuration file is not only in text format, but also can be in xml format (it depends on the selected version of gAn). The xml ensures a stronger structure, and must be transparent to the user (he mustn't see differences between the configurator that works with a txt file and the one that works with xml). At this stage both text format and xml format are acceptable, to ensure the retro-compatibility of some analysis, but is possible that in further versions the xml-based design will became dominant.

4. The user can choose what version of Root he wants to use for the program. Theoretically different versions of Root are perfectly compatible, but in practice each version of gAn is designed to work with a particular version of Root and to avoid problems it seems to be a good idea to allow the user to choose freely which version of Root use among the installed versions on the server.

5. The user can save images on his hard disk: he can choose from the shown images in the images page an image to download by clicking on a specific download button near the image. Furthermore there is another button "Download All" with whom the user can simple download all the output images.

6. The user can download a reduced version of the root file with information about the images and the results: gAn produce this kind of files as "half-processed" during the computing, and it is not a problem to save this on the hard disk of the server in a specified folder. For an expert user can be scientifically interesting to have this file (this root file contains more information than the application's output, the most of this information is useless [it is an "half-processed" file] , but sometimes an expert

user can find something interesting), so the user must have the opportunity to download this.

7. The super-users prefer the dropdown menu than the radio button, so all the radio buttons in the program are replaced by dropdown menus.

8. The user has to access not only to a png image, but to a root-image. This kind of image is interactive: the user can with a left click of the mouse (a continued click, like the "dragging") select parts of the image and zoom them, and with a right click do dynamically some kind of image processing (set colors, choose what kind of chart to show, modify the chart legend, translate in a 3D space the image etcetera). In this way each user can choose freely which information he wants see in the image (try to overseen all user's needs in this part seems to be too difficult and to overcome this limit give him the freedom of access to the image is a good idea). All of this must be done by the user through a browser window. This requisite seems to be quite complex, but Root provides libraries (these libraries work well but they are poorly documented) to interact with Javascript, and we can in some way resolve the problem.

9. In the homepage the user can see the run number of the last root file produced by the machine, and its creation date and time (so, he can understand what is the maximum of the range of the insertable numbers). Also, the run number is an unit of measurement of the time, so through this number the user can have information about the progress of the experiment. This is an important point because actually in most cases users work with the last existing run or the one immediately before, but the way in which this application can help the user in the selection of the run need to be deeply analyzed with a confrontation directly with the users.

10. There is a login system: the user must insert the password of the office to use the system. The authentication is based on the confrontation between the hash function of the inserted password and the hash function of the AEgIS password. If the password is correct the user receives a cookie, before each action in the site the server request and check this cookie to be sure about the identity of the user.

In the intermediate version there was another functional requisite: ideally the user should have been able to select a gAn version also if not installed in the server machine: in this case the system should

have been capable to automatically search on the AEgIS Gitlab repository the correct version (if existing), download it, unpack it in the server, and use it to execute the program. After some discussion this requirement has been cancelled, because it was considered complex, basically useless, and potentially harmful (on the branches of the repository there are untested and incomplete versions, that can create if executed wrong outputs, so wrong scientific results). At this moment installing manually the stable versions of gAn on the server seems to be a more smart way to work.

4.1.1 Ambiguities (and related solutions)

At least a point seems to be quite ambiguous:

The textual output of gAn needs to be formatted in some way to be more organized and clear? The answer is difficult: for a non-physicists this output seems to be disordered, too long, with too many groups of information, and very difficult to understand, but on this question the pilot users (that are physicists) questioned answered that the output is clear and doesn't need to be modified or improved in any way. The only requests of the users were about the font and the font-size. To check this fact the best solution probably is observe the behavior of the users at work, and eventually ask them information about that. Anyway, in the second version, in case of multiple run selection, there is a "navbar" that allows the user to show only a run-result per time.

4.2 Scenario based functional analysis

Following there are a list of scenarios able to describe samples of interaction. In this situation the interaction is more complex than before.

1. The user wants to analyse the runs between 30000 and 30010, plus the run 31456, to make a confrontation, he is interested both in the text-output and in the images: the user goes to the homepage, he is redirected to the authentication page and he

do the login. If successful he returns automatically in the homepage, and he can insert the runs between 30000 and 30010 manually separating them with a semicolon or better clicking the "add range of runs" button, that opens a modal, in which the user can insert the minimum and the maximum of the range, and confirm (confirmation closes the modal redirects the user to the homepage). After the user can add manually the run 31456 separating it from the others with a semicolon. If the inserted run numbers doesn't make sense the system shows on the page an error message. If the inserted run numbers are valid, the user can click the "send" button (before the button was red and un-clickable, now it is green and clickable) and start gAn. A progress bar is shown, a waiting message appears, and the user waits for some seconds (at this stage of development it is very hard to predict how much time gAn requests to execute). After some seconds the user is redirected in a page that shows the textual results: on the top of the page there is a navbar that shows the computed run numbers: the user uses this navbar to choose what part of the results to show in the screen. This bar is draggable, the user can freely move it. From this page the user can, through the button "show all images", access to another page dedicated to the images. In this page he can configure through dropdown menus the dimension, the layout, the group of the images (each image belongs to a group, the group depends on the sensor that generates the data from that the image is generated) to show. He can also, by clicking on a single image, access to another page, with a single image (the clicked image) that is completely accessible: the user can rotate the image, move it in a 3d space, zoom in, zoom out, select part, do some basic digital image processing and choose the kind of chart to show.

2. The user wants to use the version 5.34 of Root (an old but stable version) to execute gAn: he completes the login, in the home page there is a button name "Choose Root version", clicking on this button the user is sent to a page where, he is informed about the current version of Root, and through a dropdown menu can choose among some version of Root already installed on the server (all the acceptable Root versions are installed on the server) . If the 5.34 version is one of the installed version he can select it and confirm, the goal is achieved. If the 5.34 version is not installed, the user cannot use this version.

3. The user wants to select the "Rug-dev" branch of gAn: the process is very similar to the process that allows the user to choose a Root version. The user completes the login, in the home page there is a button name "Choose gAn version", clicking on this button the user is sent to a page where, he, through a dropdown menu, can choose among some branches of gAn existing on the machine. "Rug-dev" is one of these, the user can confirm and the task is completed.

4. The user wants to make the computation using only the data taken from the sensor named "Mimito": The user, after the login, in the homepage can use the button "Edit Config" the reach a page in which, through some dropdown menus, he can change the configuration file of gAn. Each of the dropdown menus is related to a sensor (often they are 5-6, it depends on the branch), and the option of the dropdown are "yes" or "no": if "yes" is selected the sensor's data are used in the computation, if "no" they aren't. One of the dropdown menus is named "Mimito", the user select "yes" for this sensor, "no" for all the others.

5. The user want to download all the images related to the runs 40001 and 40002: He can, after the login, insert the runs in the homepage (it is possible both by input field and by range selector. Two ways to do the same thing: probably in the next version this point will be rethought), run gAn, wait the end of the execution, click "Show all images", and from here click "download all images". All the images will be downloaded in png format.

6. The user wants to download the semi-processed root file related to the runs 31111 and 31112: The steps are the same as the steps used to download the images, but instead of the button "Show all images", the user has to use the button "download root files".

4.3 Prototyping

The intermediate version is based on the first version, but it is quite different and re-build from zero the application (re-using only some parts) is maybe the smartest solution. Some pages (and functionalities) are added, some existing pages are improved. Also in this case the mock-ups are interactive and really working, the developer

chooses to program directly with a common web-development process Html-javascript-php based, to optimize the time.

Following all modifications are explained.

4.3.1 Modified pages

The homepage:

FIGURE 4.1: Intermediate version: the homepage of gAn Web without input

FIGURE 4.2: Intermediate version: the homepage of gAn Web ready to start

There are some modifications:

1. The user is informed about what is the last existing run: he can read "last existing run: nnnnn, from dd/mm/yy". This point is important because in most cases the user searches results regarding the lasts 2 runs.
2. The button named "send" was unclear, the word "START" is more clear, the user can immediately understand that the goal of this button is to start gAn. The button is red and unclickable if there are problems (like in the following image) with the inserted runs (or if the input field is empty), green and clickable if there are no problems.

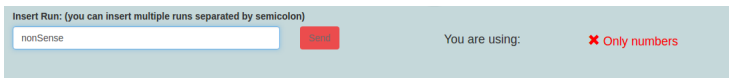

FIGURE 4.3: Intermediate version: the "Start" button is red if the values in the input field are invalid

When the user clicks start a progress bar appears. Unfortunately it is very hard to understand exactly how long the computation will last, because different runs can take different time (it depends on the amount of data that the sensors take about the run, and on the workload of the server machine, that is in common with other applications). On average is observed that the computation takes five seconds multiplied by the number

of selected runs, but if another user asked for that computation before the system already has the results in memory and the computation is faster. A wait of several seconds can be not comfortable for the user, the progress bar is imprecise but ensure to the user that the system is working correctly to ensure the correct answer. The problem is not solved: the wait time is still too long, but a solution in this case comes from the back-end, a re-factor of the code will soon make the wait time shorter. In the following image the progress bar:

FIGURE 4.4: Intermediate version: the progress bar aims to make more comfortable the user's waiting

3. The input field has a place-holder, that shows to the user how to correctly insert the runs separated by semicolon (there is an automatic system that corrects the inserted values if the separator is not a semicolon)

4. It is possible to insert a group of runs selecting them by range (inserting the first and the last): the button "Add runs by range" opens a modal (shown in the image). The user can choose the minimum and the maximum of the range, the system validates the inserted values (maximum must be more that minimum, they must be numbers etcetera).

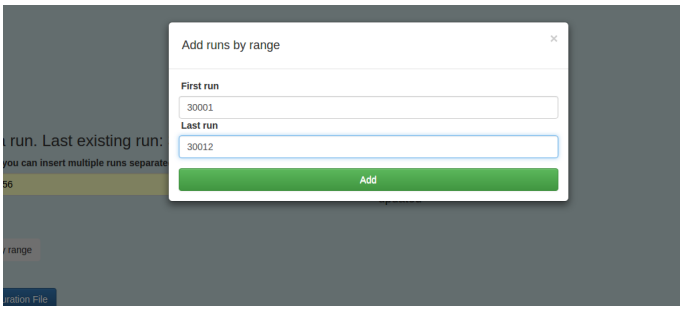

FIGURE 4.5: Intermediate version: the modal opened by clicking the button "Add runs by range"

5. There is a little message "You are using: nameOfGAnBranch" that informs the user about which is the branch of gAn used. By default if the user doesn't change the configuration the used branch is the last used, in the last successful execution.

6. There are two new buttons: "Choose Root version", "Choose gAn version". These buttons redirect the user to pages able to modify the version of Root and gAn used in the computation.
7. Each button and each field have a tooltip: a little explaining text that appears when the user moves the mouse over the object. In this way an inexperienced user can easily understand what each component does.

The page related to the modifications of the configuration file of gAn:

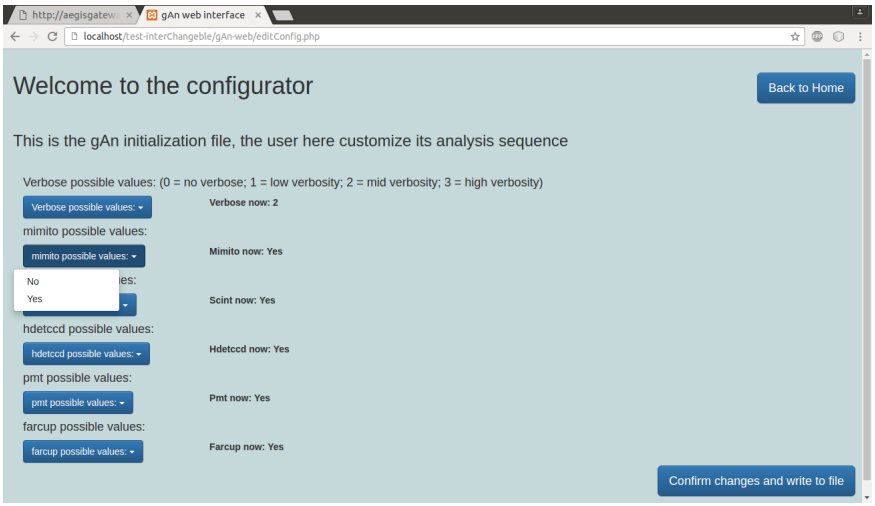

FIGURE 4.6: Intermediate version: the edit-configurator page of gAn Web

This page doesn't use any more radio buttons, dropdown menus are preferred for aesthetic reasons. The user can read near the button the currently selected value for each field.

The page that shows the textual output of gAn is exposed in the following images, has some modifications:

FIGURE 4.7: Intermediate version: the page who shows the textual output of gAn

1. The user can choose what results he wants to show on the screen by clicking the corresponding run number from the "navbar", like in the image:

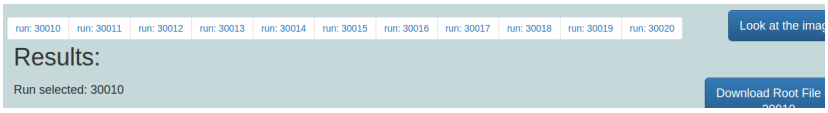

FIGURE 4.8: Intermediate version: by this "navbar" the user can choose the run results to show

   This navbar is in fixed position related to the screen, and can be dragged by the user to allow him to decide where put it.

2. "Download Root File of: nnnn" is a button that allows to user to download the semi-processed file .root with some output information regarding the computation.
3. "Back to Home" gives the user the opportunity to directly return to the homepage.
4. "Back to Home" and "Look at the images" are in a fixed position on the screen: even if the user scrolls down or up the screen he is always able to see these buttons.

The page that shows in an organized way the images that gAn produces in output. The following image shows the new appearance of the page:

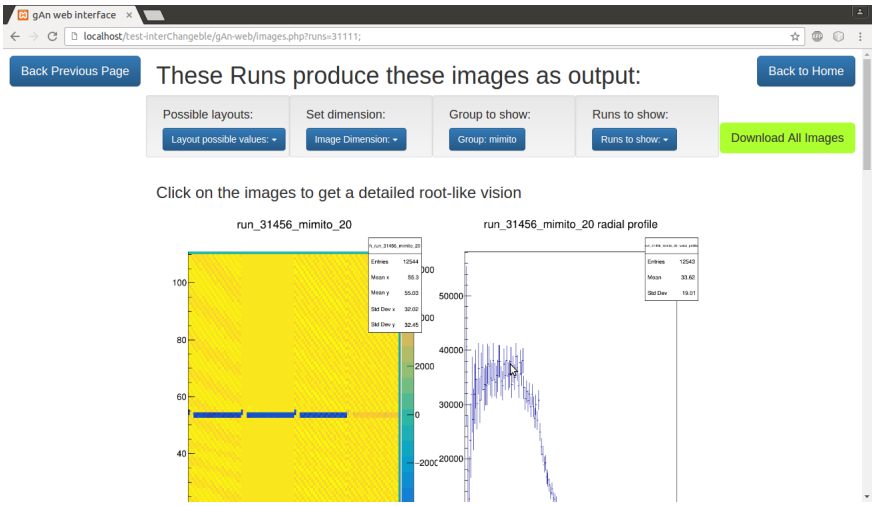

FIGURE 4.9: Intermediate version: this page shows all the images that gAn produces in output

Modifications:

1. The dropdown menu "Runs to show" allows the user to select only the images produced by a single run (by default the system shows the images related to all the runs). The users widely use this option, because in this way they can compare images extrapolated in the same way but related with runs with different configurations.
2. "Download All Images" allows the user to download by a single click all the images related to the execution. The intermediate version introduces this requirement because commonly the users download the images using the right click of the mouse and the browser's button "Save Image As". In this way this process is faster and easier.
3. The buttons "Back to Previous page" and "Back to Home" are links towards the textual output page and the homepage. They are in fixed position.
4. By clicking on the image the user reaches a page that shows the image in full screen, but not in a static format: the image is dynamically accessible like shown in the following images:

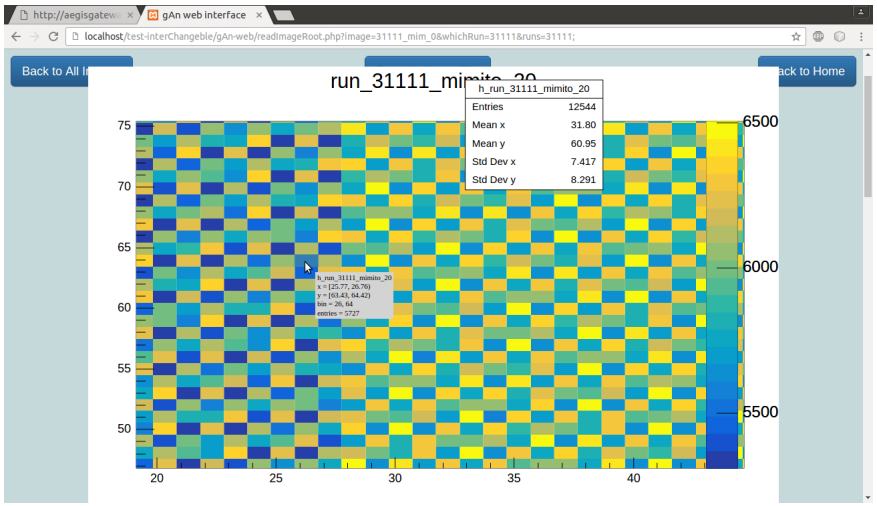

FIGURE 4.10: Intermediate version: moving the cursor the system shows the value of this histogram in the selected point

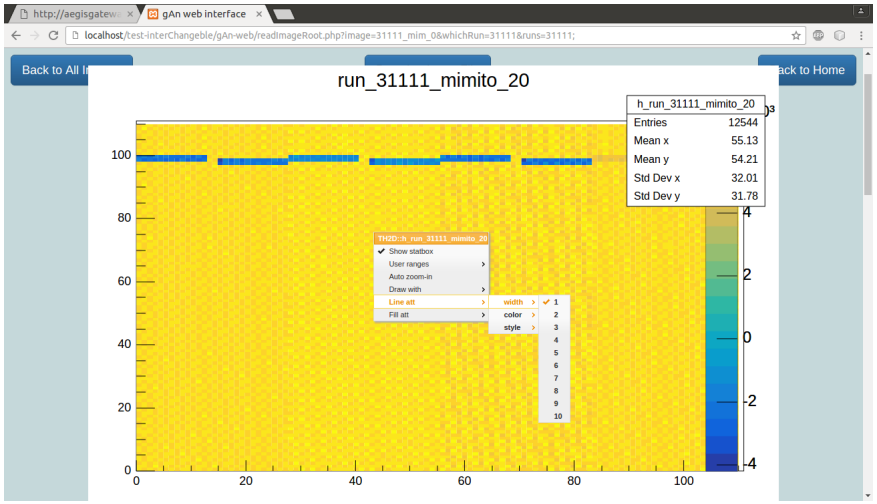

FIGURE 4.11: Intermediate version: the user can modify numerous settings in the generated image

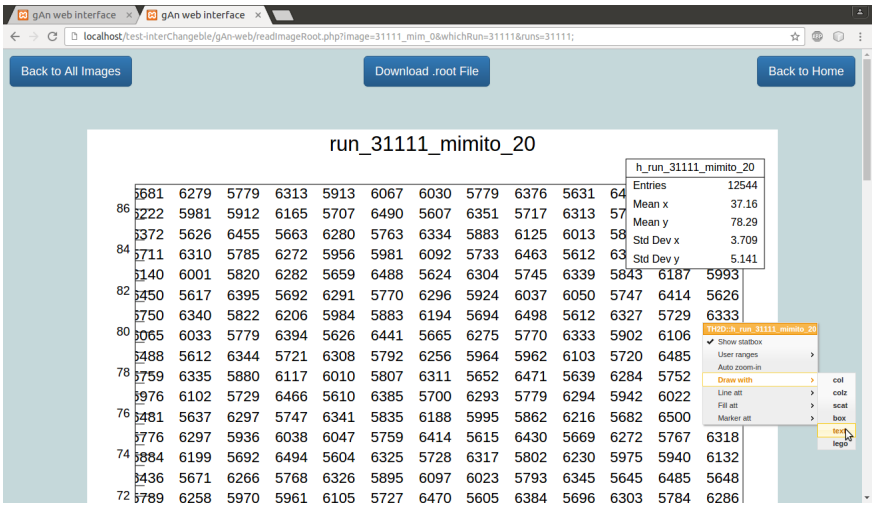

FIGURE 4.12: Intermediate version: the user can show the histogram not only in traditional format, but also in a numerical format where the numbers are the value of the function in their position

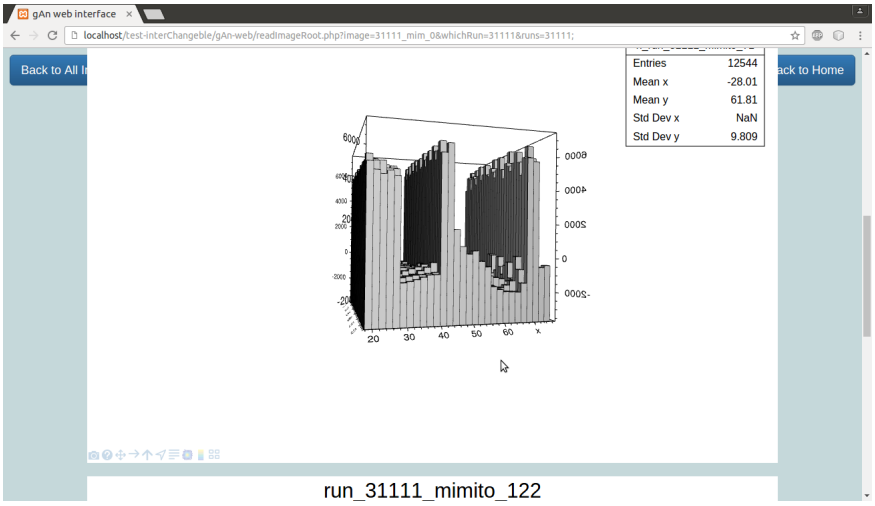

FIGURE 4.13: Intermediate version: another solution is to generate a 3d image in lego-style of the histogram

4.3.2 Added pages

Some pages in the intermediate version are completely new, because they implement new functionalities.

The first new page that the user sees is the login page. It is very simple, it doesn't authenticate a single user, but asks only the password of the office. This basic system of authentication aims simply to ensure that only the people that works in AEgIS experiment can use this software.

Following the login page image:

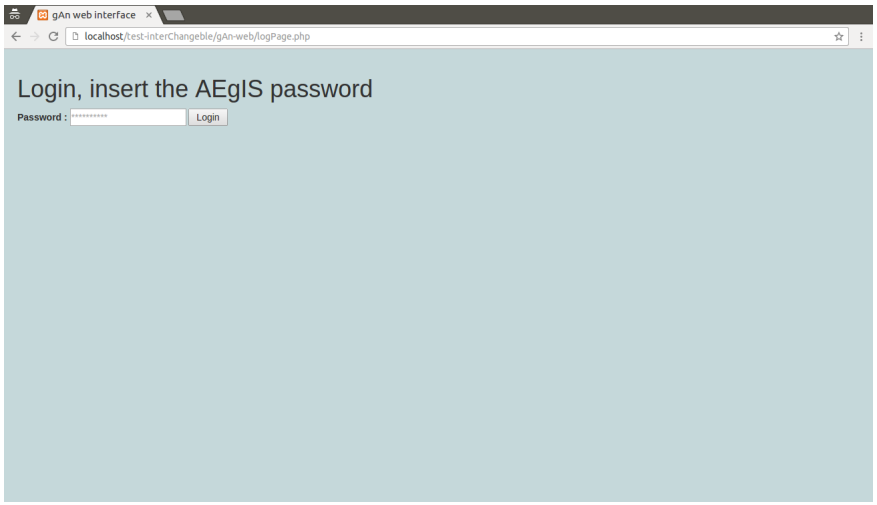

FIGURE 4.14: Intermediate version: simple login page

In this version there is also the opportunity for the user to choose which branch of gAn and which version of Root Framework to use to execute the program. The pages that the user can user are quite similar, and quite simple. The user must use dropdown menus to make these choices, and there is always a default safe choice (the system remembers the last working version of Root and the last working branch of gAn), so it is impossible make errors or inconsistent choices.

Following there are some screen-shots of these pages

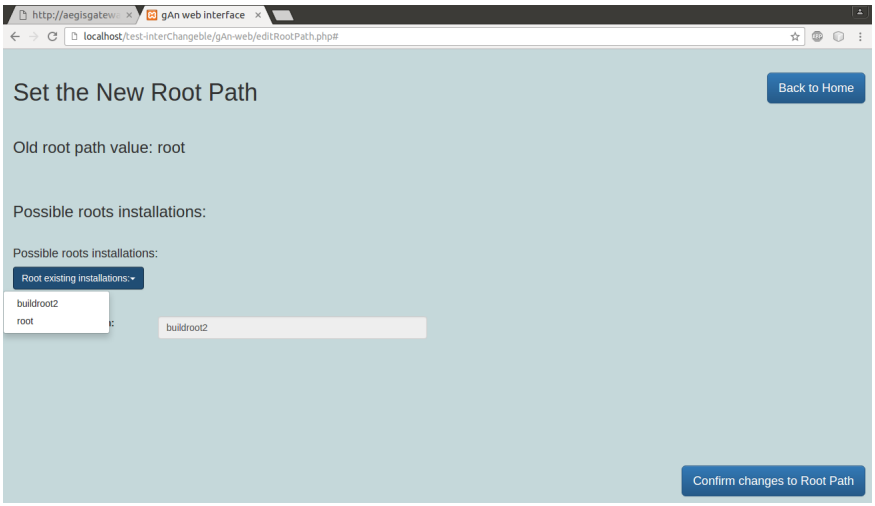

FIGURE 4.15: Intermediate version: page where the user can choose the Root version to use

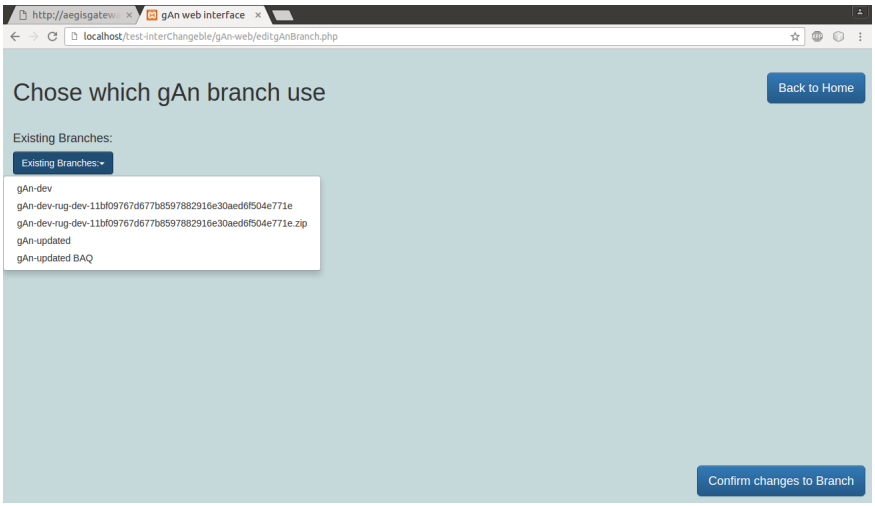

FIGURE 4.16: Intermediate version: page where the user can choose the Branch of gAn to use

Chapter 5

Final version

Here will be explained the features of the final version, and how we are arrived at it. This version is very near to what we want to create as definitive application.

5.1 Functional requirements

The last version (from now "gAn Web v3") is a modified version of the intermediate version. The second version's modifications come from two sources: some adding requirements and modifications are proposed from the super users and some else come from the tests made with the generic users; The first group of adding requirements (the ones proposed by the supervisors) are the following:

1. The main kinds of analysis are now more clear: they are around 10, and they are quite stable (their goals, so their input and expected output are quite stable, but still we'll probably modify the algorithmic ways in which these goals are achieved; from the point of view of the web interface the situation is quite stable). Their roles and when they are useful is now steadily fixed (more precisely: we are pretty sure about the role of the existing analysis, but in the future we can add more analysis..). Each user, according on his task, knows (should know) in every moment which analysis fits better the situation, so, the best solution is to allow the user to choose the type of the analysis through a dropdown menu directly in the main page (exactly like he chooses the run number). To make the user's life easier the application can use a tooltip for each analysis, to remember him in which situation it is useful. At this point the possibility of choose the gAn version is useless, because the definitive version of gAn include all the existing types of analysis.

2. The kind of use of this software is quite different if you decide to work with a single run or a group of runs, so is better if at the beginning (at the application starting) the user chooses directly if he wants work with a single run or more than one.

   So the application becomes modal: it can be a problem and can create confusion, but we hope that the simplicity of the interaction and the presence of a box that constantly informs the user about which "mode" he is using can reduce confusion instead of let it grow.

   If the user works with a single run the focus is mainly on the "exploration" of the information in the data, with the multiple runs the focus is more on the possibility to compare and see what is different.

   In the second case (more then one) is better to tell if the runs are a range (form N to M, all the runs between) or a group of non-consecutive runs (it can be useful, because of the feasibility to compare non-consecutive runs can be interesting). If the user is interested in a group of runs that are not a range (we observe that this situation is not common, but we must manage it), the previous solution of multiple runs separated by a comma is not a good idea: often in these cases the researched runs are little consecutive groups, so a list of ranges. It is better if the user can have an input area (like a sheet) in which he can put the intervals, divided in different lines (or other separators).

3. There is an effort in the developing of the whole project to make it independent of the Root version, so the interface won't ask to the user to choose a Root version anymore.

4. The aesthetic part in this application was neglected in the first and intermediate versions, but now it needs to be cured and improved to ensure the best possible satisfaction to the users.

Furthermore, the version with this last requirements was tested with the users: the developer studied their behavior observing them at work with the existing version, listening to their comments, and asking them opinion. The impact of the debut with the users highlights some problems to be overcome, and also provide some interesting ideas: from the analysis and the solution of these problems and from the user's proposals the requirements of the last version was defined.

In this document there is a chapter (6) aimed to explain how the tests with the users took place and what are the results, to know more about the argument.

The main problems observed (and for each the proposed solution) are the following:

1. It is important to help the user in some way to choose the run number: the user needs a view on the logbook of the run, that is a text in which there are information about each run number grouped by date.
2. Actually nobody use the button to modify the dimension of the images: they all use the biggest version.. is better to remove (or move in a less central place) this dropdown menu. A similar reasoning can be done for the dropdown that allows the user to switch between the vertical and carousel menu (everybody use vertical, clearly at this point the carousel version isn't useful). Also nobody use the run selector to choose which image see: they prefer base themselves on the image's title.. at this point a surely better recognize that the existing navbar is not adapt to satisfy the needs of the user, a better solution is to create a selector with the image's titles, precisely a group of checkboxes, each with the title of an image as label, using that the user can select one or more (or all if he wants) images.
3. The read of the textual output takes too much time: it can be improved highlighting the most important parts (or better, the most important parts for the selected type of analysis)
4. The user needs to choose by the configuration page the degree of precision (the minimal error) of the x-axis (the time related axis) of each time-related values images. This parameter seemed to be not very important and in the second version actually there wasn't, but almost all the users modify it quite often (manually), and verbally ask for the possibility to set it through the interface. So is better to let them to do it by the interface
5. The general organization of the output is simply not the best possible: the users need to switch easily from textual to images output and vice-versa, and the buttons are too many spread on the screen, so too difficult to find.

5.2 Non-functional requirements

There is another special requirement emerged in this version, not strictly related to the common behavior of the interface but quite

important: the machine used as server is also used for the testing of single analysis when they are written; sometimes these analysis crashes (quite normal in a test phase), and they remain in a "hung up" state, without doing nothing but occupying Ram memory, and if the blocked application are numerous that can be a problem for the performances of the machine. To ensure a correct behavior gAn Web must check this situation at starting and eventually resolve it.

5.3 Scenario based functional analysis

Following we can show some examples able to explain how the typical interaction must work in the definitive version:

1. Let's suppose a very common situation in a shift: In the shift is running the run NNNN. The shifter wants to calculate the temperature of some particles with a (complex) process based on the shape that they create on a sensor (the basic idea is: if a particle is hotter, its speed is bigger, it can escape more easily from a magnetic trap, and the sensor can observe more particles outside the trap. At this point the software tries to fit, with an exponential function, the shape of the signal that the sensor shows as output, and through the integral of the slope of this exponential function can estimate the temperature of the particle). The user chooses the single analysis, inserts the run NNNN, selects the kind of analysis called "TemperatureMeasurement", clicks the "start" button; The system shows to the user a textual output and some images: The textual output proposes a lot of information, but in the ninety five per cent of the cases the user is interested in only three parameters, that are clearly highlighted by a special font, a color and a visible border.

   The user must also check the image: there is the histogram of a signal, an a red line where the function is fitted. The user checks if the red line has an exponential shape (if it is a straight line the situation is too aleatory and the data are not affordable); the user also checks if the fit line is too long (it must be very little to accept the results of the analysis, because of physical reasons).

   Let's suppose that the fit function is a straight line: the user can retry the same analysis with a different setting: he returns to the homepage (through the button "New Analysis" on the Navbar),

he clicks "Advanced Settings" to modify the configuration, he inserts a new value through a slider, to have a more precise graph, in the hope to fit it better. At this point he can retry the previous algorithm (a more precise function will lead a slower computation but probably will give an affordable output)

2. The user is interested in a two distinguished groups of runs, from 57001 to 57005 and form 57016-57020. This situation is very common because most of the groups of measurements are repeated in two different slices of times (to be sure about the results). In this case the user, after having selected "Multiple runs", can click on "By InputArea" and a text area appears. In the area there are the value inserted the last time that the user used this function (we observe that is quite common repeat analyses with the same groups). The input is quite free: user can insert a group inserting the first and the last separated by a dash, a space, or a comma, and other group by changing the line (so each not empty line is a group). The user at this point selects the analysis and run the program.

3. The user is interested in a range of runs, from 58001 to 58010, to perform the same analysis and check if and how some values are varying. He can select multiple runs, insert the first and the last, select the analysis (the multiple runs analysis are taken from a different set from the single runs analysis).

4. The user inserts a run, and selects an analysis names PMTs. This kind of analysis takes information from a group of sensors and produces in output (usually) seven images. The user often is not interested in all the seven images but in one or two. At this point, using the check-boxes selector he can select/deselect images to show (by default they are all selected).

5.4 Prototyping - Implementation

Here is explained with some screen-shots how the final version looks like, and how does it work.

Login page:

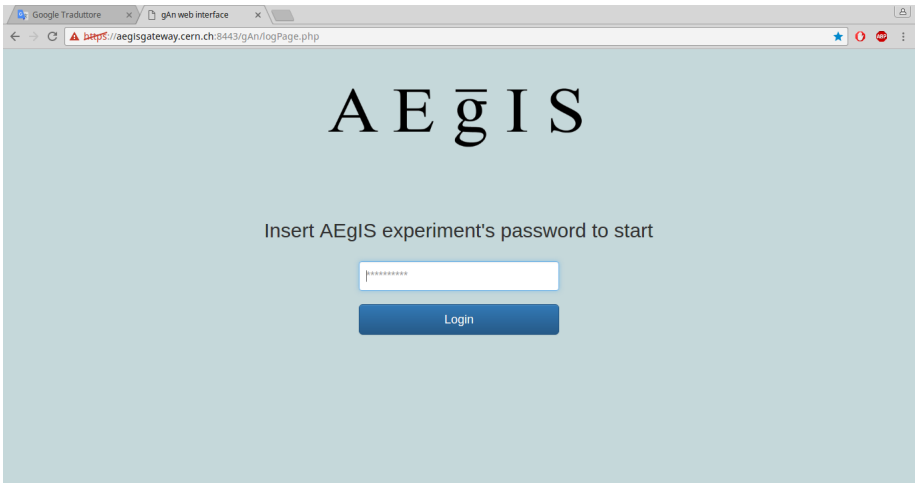

FIGURE 5.1: Final version: the Login Page of gAn Web

First of all the login page: it is very simple, only one user exists (all the people in the AEgIS experiment are equal when they use this software, and no others experiments use this application) so only the password needs to be inserted, in order to ensure that only people who work in the experiment can use the software. We can see the typical objects taken from the Bootstrap's default graphic objects.

Home page, initial choice:

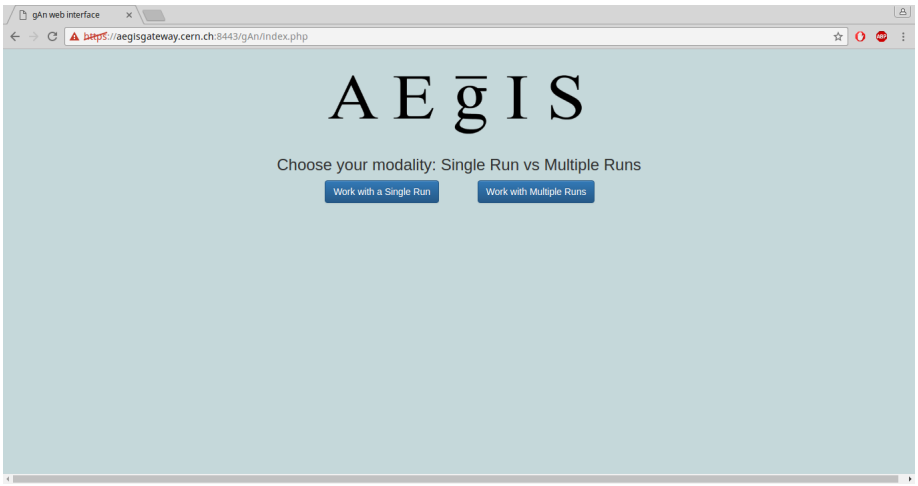

FIGURE 5.2: Final version: the first thing that a logged user sees

The neo-logged user can here choose if he is interested in a single run analysis of in a multiple runs one. These two situations are quite different, so the application works with them in two different ways.

After the choice, the user can always change idea, by the appropriate button:

FIGURE 5.3: Final version: how the user can change his choice

Home page, differences between single and multiple runs:

Here we can see how the two modalities of work (single and multiple) appear.

Single run:

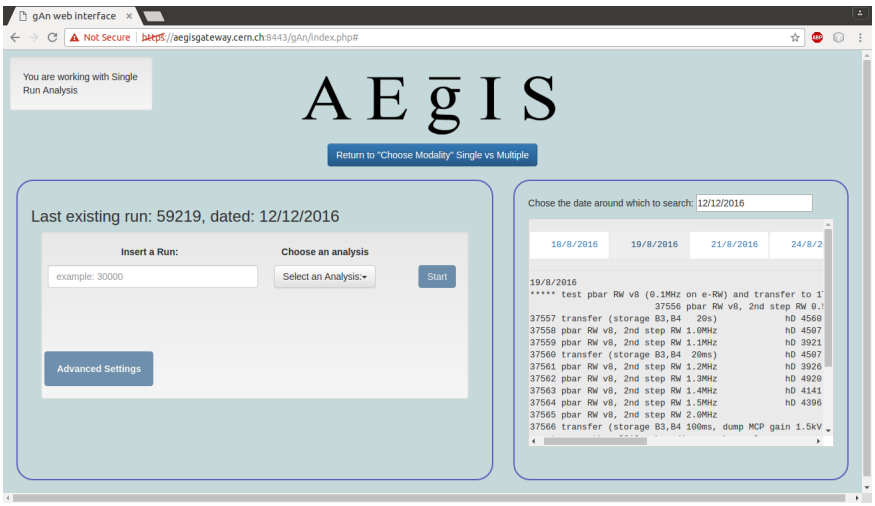

FIGURE 5.4: Final version: the Homepage if the user chooses Single run analysis

the single one is more simple, on the left of the screen the user inserts a run number and a kind of analysis. We can see the button related to access the configuration page and modify the settings (Advanced Settings). On the right part of the screen there is a view on the RunLog of the experiment. The user uses the right part when he wants more information about what run use to work (actually this situation is quite seldom). In the case the user can insert a date using a datepicker, the system opens some pages related to the days around that date and in each page shows the content of the logs related to this date. The goal of the datepicker, provided by Bootstrap, is to reduce the charge on the memory of the users (it shows as default the current date, and shows as result the pages of the RunLog related not only to the selected date but also to the days before and after, to give the user some references and allow him to make errors of 2-3 days still finding the result) and to allow him to insert the date in a correct format. The logs are always divided in runs, and for each run there are some information and comments. At the moment this functionality is a mock-up (it is one of the few mock-ups exposed in this document).

On the top-left of the screen there is always a little note with some information about the status of the program: this is a little cognitive aid:

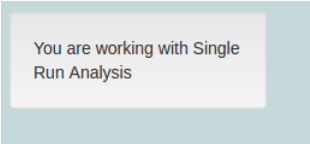

FIGURE 5.5: Final version: a label with some information about the current status is always visible in the homepage

On the screen there is always wrote the last known run, because in most cases the users use it or the immediately previous one.

The interface informs the user if he makes some errors inserting the input. On the first implementation the error messages was programmed to became visible every time the inserted value was incorrect, so also at the moment the user was entering in the page and still hasn't inserted the values. This behavior is quite aggressive and unkind to the user.. it is surely not our intention so is a better solution if the error messages appear only when the user try to click send without having inserted correct values. The messages are hopefully aimed to explain the user how to solve the problem:

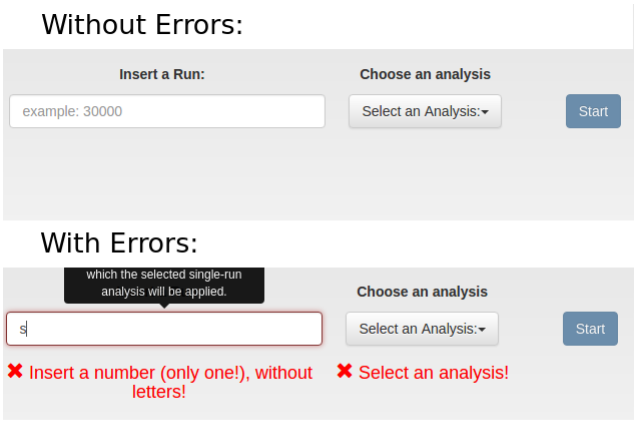

FIGURE 5.6: Final version: how error messages work

The dropdown menu usable to choose the kind of analysis is quite long (up to 10 options). A good idea is to order it and organize it in homogeneous groups:

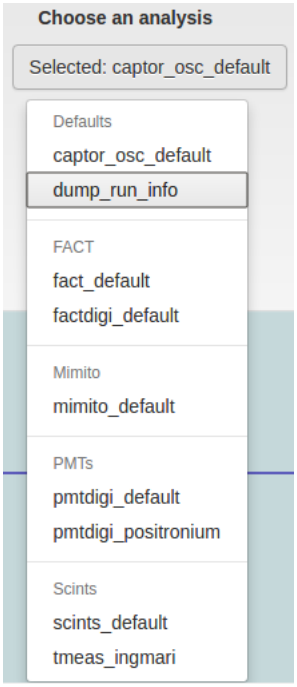

FIGURE 5.7: Final version: grouped dropdown menu

Multiple runs: There is two different ideas about how to implement this page, because the new requisite about the fact that now the user can choose if insert one or more range using two input fields (for the first and for the last run) or a more flexible (but less efficient) "text area" is different from the previous version . So two different solution are evaluated, for convenience in this paragraph they are called "first" and "second", where the second is the correction of the first. So the first one was used to understand if the functionality is useful, the second one is a correction to ensure a more clear and better organized interface. At the end the second is the only one adopted in the application.

Multiple runs first version:

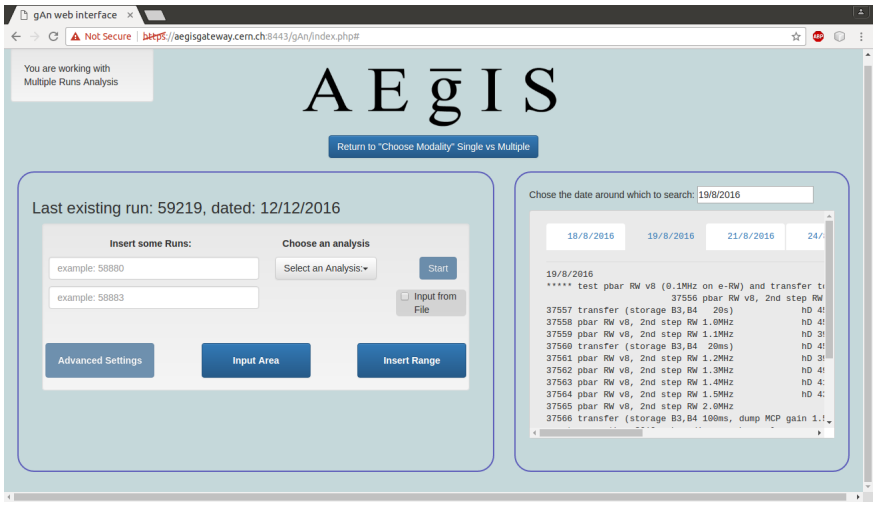

FIGURE 5.8: Final version: the Homepage if the user chooses multiple runs analysis (first implementation)

The multiple runs modality is similar to the single run but more complex: in a standard situation the user inserts two runs, the first and the last. The alternative (less used) situation is the one in which the user, through the "Input Area" button inserts a list of groups of runs (this runs are written in a file, that the system is able to accept as input). This modality is sometimes useful, and can be activated with the check-box "Input from file". The activation of this modalities denies the access to the two input fields (you must only one way to insert runs..). Following a little example of how the user can insert data through an Input Area in the first version:

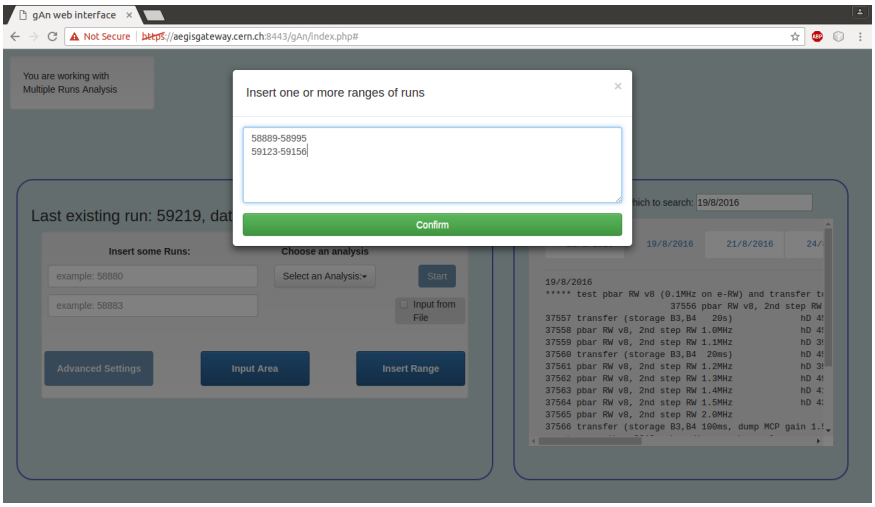

FIGURE 5.9: Final version: modal using which the users insert different ranges of runs (first implementation)

Multiple runs second version: The differences are only related to "how" the users can access the input field and the text area, the back-end is the same. Now the user can through a switching selector select which way he wants use to insert the runs. Obviously the selection of one way hides momentary the other one:

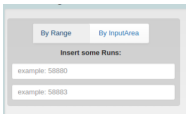

FIGURE 5.10: Final version: here is if the user selects "By Range" (second implementation)

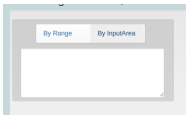

FIGURE 5.11: Final version: here is if the user selects "By InputArea" (second implementation)

Now the little check-box near the start button is no-more useful (actually it created confusion), like the "Input Area" button. The interface is now more clear, without modals, with less commands but able to do the same things. A well surrounds the "runs related" part of the input system that now is quite composed and needs to be perceived

like a "group". Following how the multiple-runs interface appears now:

FIGURE 5.12: Final version: how the homepage appears (second implementation)

Advanced Settings page:

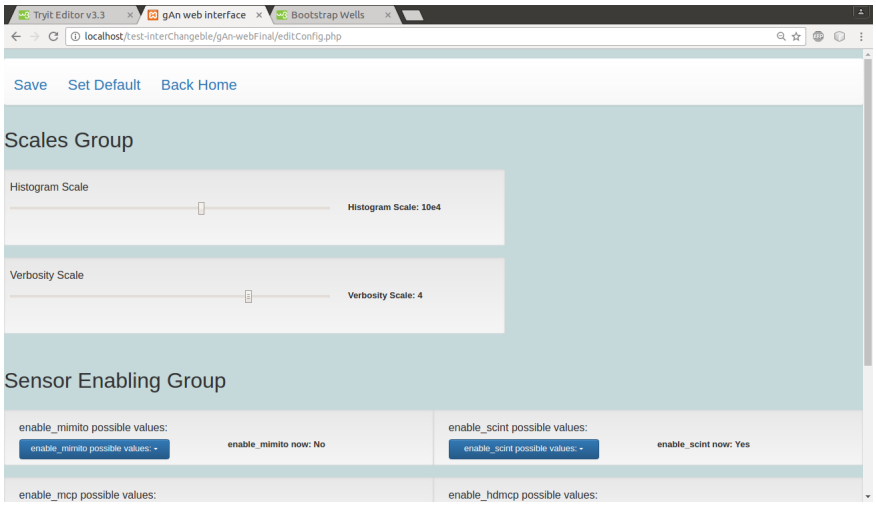

FIGURE 5.13: Final version: the page related to edit configurations

This page aims to allow the user to modify some groups of settings. The general rule is to use dropdown menus if the choice is true/false or Yes/No (quite common) or between a reduced number of textual options, an to use instead seekbars (an input bar moving which the user can insert a bigger or smaller numerical value) when the user must set a numerical value that goes from a minimum to a maximum value. On the top of the screen the user chooses to save the current setting, to return to the default setting, to go back home (without saving).

In some cases the values to insert by the seekbars are in a very big range and is not easy to the user to have a too long seekbar. A good solution is to choose an exponential values seekbar: so the user chooses the exponentiation, not the number. This choice is quite reckless for normal users.. but for physicists the understanding of how an exponential value works is not a problem:

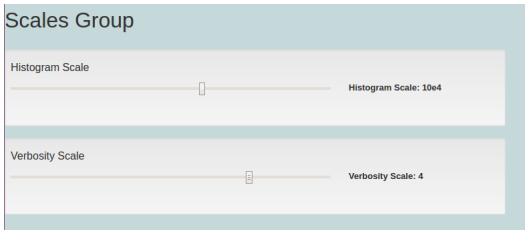

FIGURE 5.14: Final version: an example of exponential scale (the first) followed by a non-exponential scale (the second)

Other setting are more easily accessible by some yes/no dropdown menus:

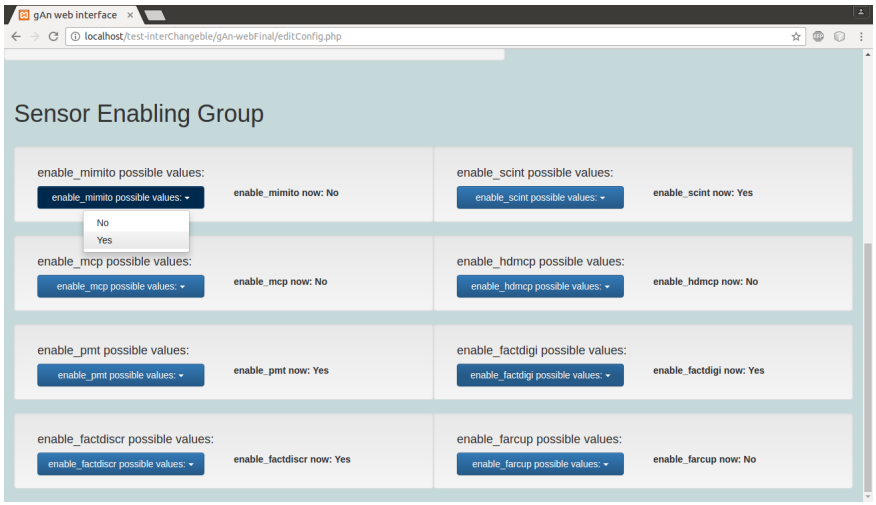

FIGURE 5.15: Final version: a group of dropdown menus

Output:

There are some innovations. The first one is the navbar: now the user can find here all the most important commands. Before there was more buttons, distributed in the screen, and during the tests the users had problems to found them and to understand their functionalities. Now the navbar is more clear and organized:

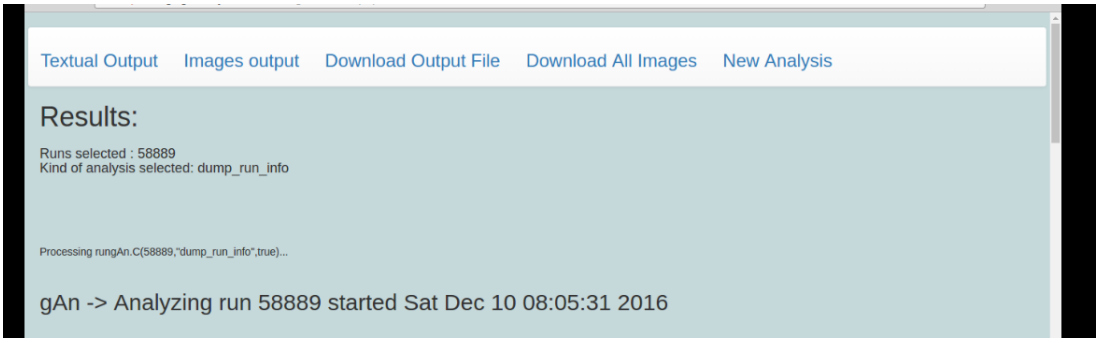

FIGURE 5.16: Final version: the Navbar

Textual Output:

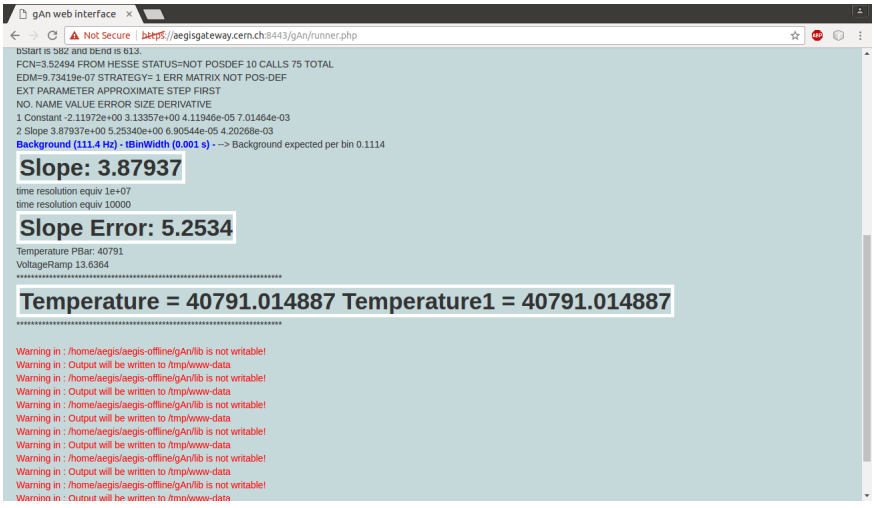

FIGURE 5.17: Final version: the output in textual version

Here we can see out a textual output appears: there are more information, but in the 95 per cent of cases only a few are really useful, so they are highlighted with a special font and a big border (the hope is to use perceptive suggestions to advise the user about where to find what he searches). In this example we can see also some warnings:

where something works wrong (or simply not perfectly) the system informs the user about the problem, with a red color, to ensure that he sees it.

Let's make a confrontation between the same textual output before and after being formatted:

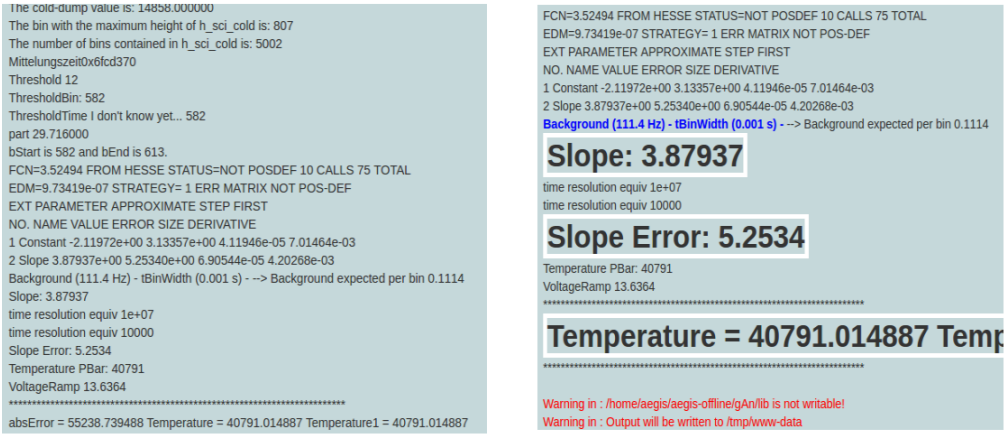

FIGURE 5.18: Final version: formatted vs unformatted output

Images output:

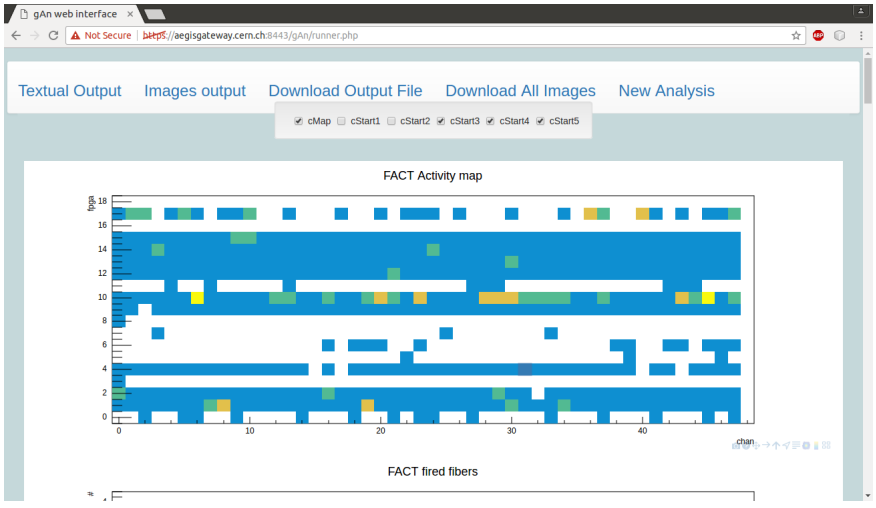

FIGURE 5.19: Final version: the output images

Here we can see that there is no more the opportunity to select the dimension of the images (but still there is a zoom function in the image), or the layout, because none of the users never used these functionalities in the tests .. nor the possibility to choose a particular run of a group of images: this is because like discussed previous in this chapter the existing of the "kinds" of analysis makes this choice useless. Actually the navbar with these options is completely removed: the goal of this choice is to avoid the "information overload", so not to give the users too many commands and options, even more if these options are considered quite useless by the users. Instead there is a list of check-boxes that if selected or deselected let the images appear and disappears. This feature is not determinant with 2 or 3 images, but sometimes the images are 7-8-9 so it is necessary.

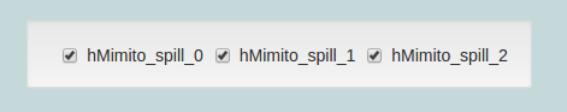

FIGURE 5.20: Final version: how the user can choose which images to show

An observation: sometimes the output is a list of images, vertically organized in a framework, some other times is a single image with more graph on a single sheets (a graph for each run inserted by the user as input, now all with the same color, but soon each run with a personal color); why? The output is a single image on a single sheet when the goal is to compare different charts to see what is different, the output is an array of separate images when the goal is more related to the exploration, when the different images don't contain the same type of information. Sometimes is interesting make a confrontation between two images: in that case they are usually positioned horizontally. Who can decide what is the goal in each case? The solution is simple: who creates the analysis, so the back-end of this system codifies directly in the program how to show the charts in one or more sheets. This is because nobody better than who creates a particular analysis can understand what are the goals of this analysis.

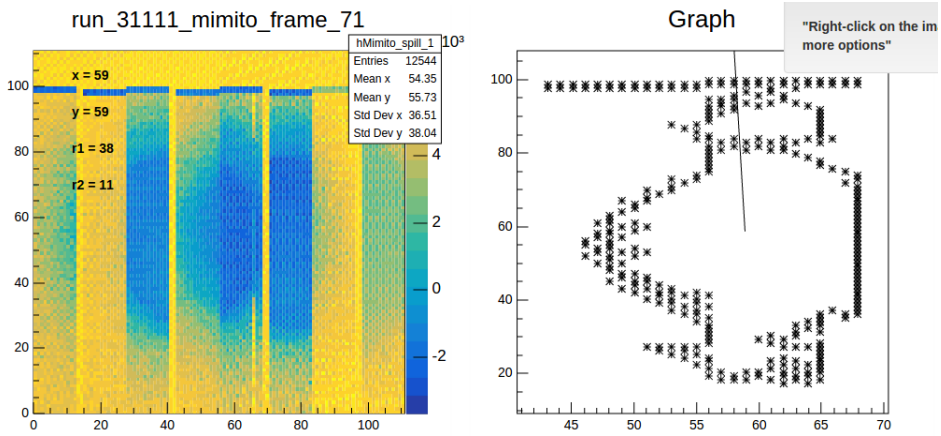

FIGURE 5.21: Final version: Images that are meant to be compared

Also in the "output image" case we observed that the users in most cases search in the images (almost) always the same information, so in some images the most important data (often they are dimension of peaks or integrals of spaces, or distances between peaks, or, in the case of charts with a time on the x-axis, auto-generated timelines about asynchronous events and triggers happened etc..) are directly written on the picture, like the following examples:

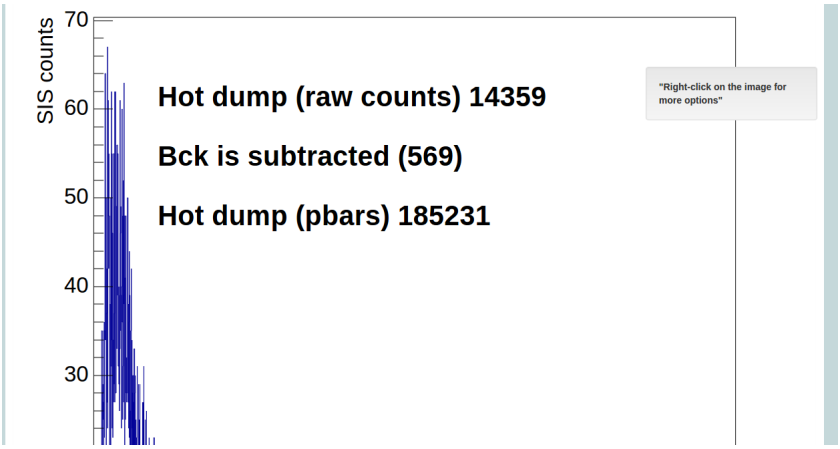

FIGURE 5.22: Final version: images with info

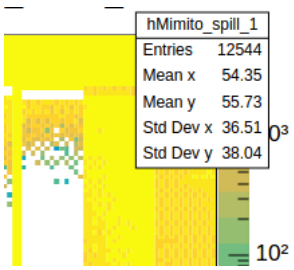

FIGURE 5.23: Final version: other image with special info in a clear position

There we can observe that the important information can be shown directly on the plot of the image (if they are graphical information, for example peaks, position of triggers etc) or in a special square on the top-right of the image (if they are not directly related to the image, but come form calculation or machine's settings).

There is again the opportunity to access more information right clicking on the graph, of modifying it in various ways for example showing the graph in three dimensions:

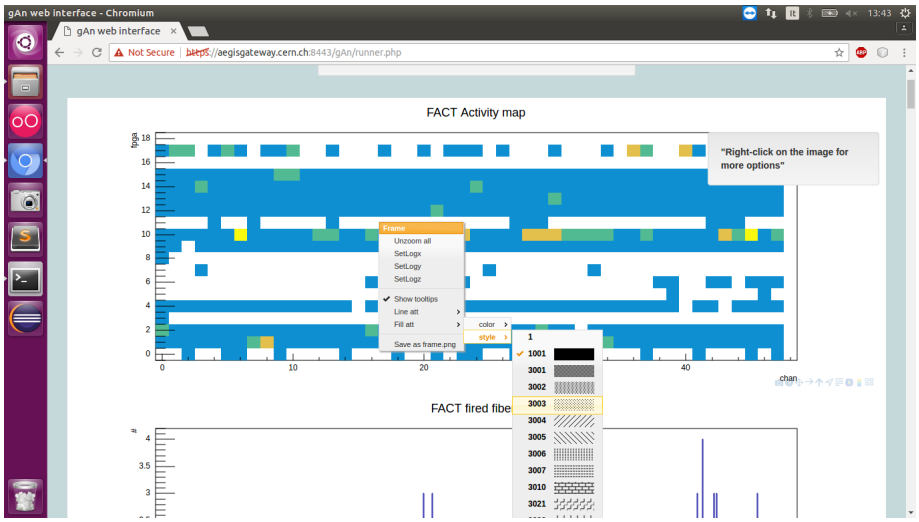

FIGURE 5.24: Final version: some options

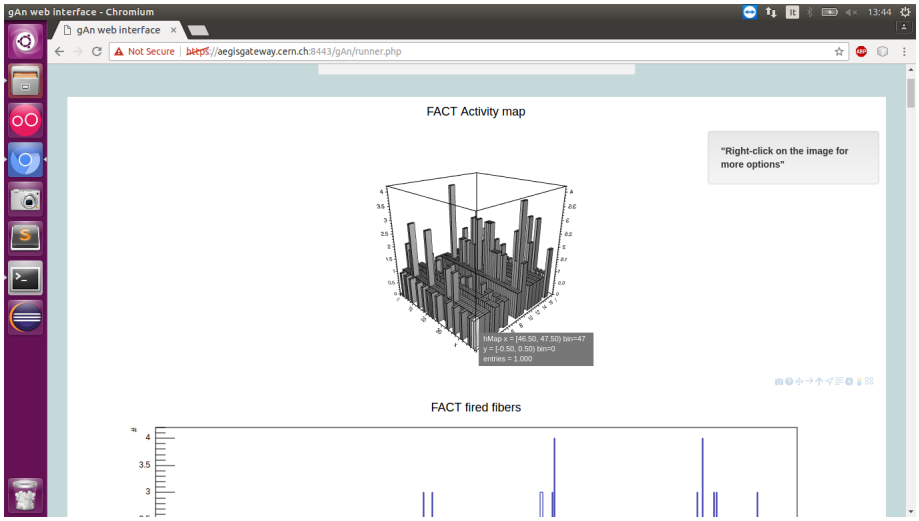

FIGURE 5.25: Final version: other option

Often the users in the tests had problems understanding that with a right click they could access these functionalities, so now there is a specific label on the right of the screen to inform them (this label is in fixed position, but became visible only if the user arrive with the scrollbar on the images):

FIGURE 5.26: Final version: label to inform the user about a feature

There are some additional commands, that actually nobody used in the tests. Having on the screen commands that nobody uses is a potential source of confusion, but the developer is not absolutely sure about the idea of delete them. A good solution is to have them in the bottom of the frame of the image, not very highlighted, because they are not very important. Probably in a future confrontation with the super-users (is important to remember that this stage is never finished and always open to modification) these commands will be deleted. They are in the following image:

FIGURE 5.27: Final version: not frequently used controls

Chapter 6

Users and Tests with them

In this chapter first of all is exposed how the users are defined and what are their characteristics. Also, here are described the tests conducted with them and in which ways they are useful to improve the gAn Web's features.

6.1 User's Profile: why?

Generically a correct understanding of the user is very important to adapt the features of an application to him; In this case is also more important, because the kind of users to that this application is intended to is quite particular.

6.2 User's Profile: classification

6.2.1 Generic Characteristics

In this part of the document we can try to classify the user: The estimated number of users of the application is around at most a few tens. All the users of gAn Web work as "Shifters" in the AEgIS experiment: so they work with data acquisition and (simple) data analysis. A good advantage is that the users are already strictly defined, we have detailed information about them, and they are easily accessible for tests and co-developing. Usually the typical user is quite young: around 30-35 years. This is because the most of the time personnel with a major seniority doesn't work with tasks related to data acquisition. Half of them are male, and half female: in this particular case we think that it is not relevant in their interaction with gAn Web. A

considerable amount of the users use eyeglasses, and probably some of them work too much time in front of a screen, but the kind of interaction designed with gAn is usually fast and isn't a big commitment for human eyes. We tried to avoid too many white backgrounds and the use of dazzling colours, but for the images' background they are considered acceptable to improve contrast and let the image to be more clear.

6.2.2 Education Background

The cultural and educational level of the users is very high: almost all of them have a Phd, and they do a complex scientific job. They often have in mind a really structural model of the processes that go on in all the AEgIS experiment applications.

All of them have an extraordinary deep understanding of the particles physics: in particular they understand the technical terminology, the jargon in which the logbooks are written (because all the personnel, in shifts, write the logbooks) and the names of the sensors (this is quite important, each sensor has a proper name, they are very numerous, and the fact that the users can identify them easily is a big advantage).

Other important information: almost every time a user uses gAn he is quite sure about what he is looking for. Situation in which the user searches a run number and reads generically the output of an analysis (or thinks about which is the correct analysis to use) is not common: in the most of the use cases the user is pretty sure about what is the information to search and what is the correct analysis.

A different approach exists for the run number, in this case we must identify two different situation:

1. The most of the times (around 2/3) the user works with the last run, or the one immediately before. In this case is quite simple identify the correct run (the last is always shown on the application homepage).
2. Sometimes the user searches information regarding a past run, that he remembers to be interesting for some reason. In this case often the user remembers the date (the precise day, or the

week) of the interesting run, but not the run number exactly. To manage this situation in the homepage there is a "view" of the RunLog, on that there are information about all the runs, organized by date and run numbers (the implementation of this view is a consequence of the tests with the users).

Most of the users are used to use the Linux's terminal, so they are used to quite complex but efficient interaction. A good advantage is that the users are very used to approach new applications, even complex ones, and they learn very rapidly: the advantage is that they are used to give developers hints about what they expect from a new application, because they do it quite often. GAn Web offers them a way to interact efficiently like in the terminal but easier (hopefully). They also are used to have a feedback for every action they do (using the Linux's terminal you can always know if the program is running, if it is crashed, and what are its outputs).

One of the tested users is (also) a web developer, so his hints are particularly interesting because is quite expert in the creation of interfaces.

All the users have an excellent proficiency in English (they are used to speak English in their work environment).

6.2.3 Models in the user's mind

An interesting additional observation about the users is that more than half of them (mainly the most expert) have in mind a structural model about how do the processes work. In fact in their mansion there are tasks related the the design, the installations, the linking system of the sensors on which gAn is based. So more than half of the users use gAn Web understanding directly the processes that take place behind them. Adding to this, a part (in this case less than half, maybe one third) of the users have a good understanding of software programming so can also really understand "how" the application is working.

6.2.4 User's psychological characteristics

The style of reasoning is quite deductive: the users seem to act in a very scientific way: they observe the situation, they make hypothesis and they experiment them to check if they are true. From the test highlights that the users think mostly in an analytic way, using only seldom their intuition. In gAn Web we can observe that when they are for the first time in front of the application they observe the screen, reading all the tooltips, they understand (of better, they make hypothesis about) what each function does, and they test the functionalities making comments regarding if the application's behavior is the expected one. According the users propensity the designer of gAn Web tries to let the application act like the user expects.

Generically the users work in a quite enthusiast and positive way: the analysis done previous with the Linux terminal was quite time-consuming, so gAn Web is considered as an improvement (or at least, a step in the right direction) and the users seems to be quite happy to use it. All the people showed in the tests the maximum possible commitment and professionalism (in this way the opportunity to test the application directly with real users was very helpful).

It is possible to analyse the arousal of the users from the point of view of the Yerkes–Dodson's Law: the situation seems to be favorable: nobody of the user has problems with low arousal or too much performance anxiety (probably no one seemed himself under the judgement), so the personal activation of the users seems to be optimal.

6.2.5 User's level of expertise

At this stage all the users are beginners, but generically the application is simple and, except that the first moment for understanding the basic functionalities they can reach a good level of expertise. A big advantage is that almost all the users understand what the application does, which are the mathematical principles on which gAn is based, and how this application is integrated in their tasks. All the users are absolutely expert pc-users. The two pilot-users play the role of super-users, able to be a bridge between the design of the

application's interface and the needs of the users. They are also the developer of the pre-existent application, on which the interface is based. So they can be an extremely useful resource.

6.2.6 Environment

The environment is a normal office. Often the users work in night shifts, and the only possible negative effect of the environment is the tiredness of the users, able to damage their arousal.

6.3 Relation developer-user and progressive sematization

The developer of the application is a student in Software Engineering, and his domain is quite different from the users one. But in the time in which users and developer worked together the mutual adjustment between the two parts and the common approach to solve problems, united with the opportunity of use the pilot users (also supervisors) as bridges leads to a progressive semantization, so we can expect (hopefully) that under this point of view the developer has a deep understanding (or at least the deepest possible) of the real needs of the users.

6.4 How the tests take place

This test is divided in three parts:

1. The first part is a direct observation of the user when he uses the application. It take place during the shifts: both the users and the designer partecipated in this task divided in groups of 3-4-5 people, so this is a perfect opportunity, also because actually it is not a simulated test, is a real test on the field in a real work situation. This test is an observation of the behavior of the user with the application: the developer tells to the user only a few information: the fact that this application can

transform a run number and a kind of analysis in an output. The first idea followed a pattern in which the developer asks to the user to use this application during the shift, in the situations that correspond to the expected scenario of gAn Web: in this situation the user talks about what he is doing, and makes comments, and the developer observes and takes notes about what happens (and this pattern was effectively applied with 4 users), but to test the biggest possible numbers of users and recover more information, in other cases the developer asks to the user to try to use the application in a simulated situation (not a really work situation) and to reason about what he is doing. The reason of this choice is that wait the moment in which the application can be used during the shifts to test isn't efficient (but still is a good resource) and "create" artificially the situation is a good idea to save time.

2. The second consists in the proposition of a standard survey about the application to the user, to be compiled. This survey is the standard SUS survey. This choice is related to the fact that using a standard survey we can have an high quality already tested survey with precise and organized closed questions, able to give us an idea about the overall result of the interface.

3. The third part consists in a brief discussion with the user (some chosen users, selected by their willingness to dedicate a moment to discuss about the application): the designer asks him in which way this application can be more useful and if there are in his opinion some good ideas to implement able to improve the helpfulness of the application. The discussion is quite free, but a fix topic is about what are the sources of the problems that the users had during the use of the application in the first part of the test. In fact the big limit of the observation of the users is that it is easy to find if there are problems but more difficult to understand why the problem there are and how they can be solved. So a direct confrontation of the user can overcome this limit. The developer tried to ask to the users some of the 9 questions proposed by Nielsen, but not in an automatic way, searching a more flexible approach and trying to follow the user and let him talk freely. Another fix topic of the discussion is a generic request about tips and ideas that the users can have to improve the application (one of the interviewed user is also a web developer, who creates some of the application and related interfaces used in the experiment, so in that case this second question occupied the biggest part of the interview, to

exploit at maximum his experience in this field).

6.4.1 Preliminary observations

It is interesting to observe that it is very important to encourage the users to criticise the application: if fact the developer and the users work together for some weeks, and it is possible that the users are too gentle in the judging of the application, because they don't want to be offensive. Instead, is necessary say them that all their critics are absolutely useful to improve the result, because the developer cannot understand alone what can be modified.

6.4.2 What the test is evaluating?

Often some tests measure the time that the user needs to execute tasks to understand how difficult the interaction can be; but in this case the developer takes the decision of not to evaluate the time: all the processes in the AEgIS experiment are very time consuming, and this is not a problem, because the goal is not the efficiency but the effectiveness. It is not important if the user makes an analysis in a few seconds, it is important that he can use all the functionalities that he needs correctly to achieve his goal. So the test are evaluated in these dimensions: how many functionalities the users use? If they search a particular functionality the search from the beginning in the expected place? Do they find the correct command or they ask to the developer for a hint? Does the functionality that the user search exists? These are the interesting answer that we need from the tests.

6.4.3 Execution and Results

The first part of the test, the one related to the observation of the users working with the application and talking about what they do, gave a big amount of useful information and lets the developer observe a good amount of problems and errors. The biggest part of the problems are founded by observing the behaviors of the user and asking directly to him if there is a problem: for example, if the user searches for a while how he can enable a sensor and has some difficult to understand how to do it, the developer can listen his comments, understanding his reasons about where he thinks this option can be, shows him where actually the sensor is, and understand that probably the location of the sensor option is not clear. Here is a list of problems found in this phase:

1. Almost all the users tried very often to change a particular parameter by the configuration page, named "scint rebin", related to the thickness of the points that are fitted to create a function. Moving this parameter the user can modify a particular type of analysis named "TMeas", according to different exigences. Before the test was not so evident that this parameter was so important, and actually in the intermediate version it is not in the list of parameters that the web interface can modify. So in the tests the most of the user manually changed it by a text editor in the configuration file on the server: surely it is unacceptable and this problem must be solved to allow the application work in this particular situation.

2. The main options in the configuration page are related to the activation or de-activation of sensors, but the users actually don't use them very often, because the default values are acceptably correct. So a good idea is to highlight other setting options, and move the rarely used ones in other part of the screen.

3. In the page that shows the images the user can by a selector choose if he wants to see the images in a big, a middle, or a little format. During the test no one used this opportunity, and when the developer asked to the users if this control can be useful they simply say that the correct dimension of the image the "big" solution, with the image that cover (almost) all the screen in width. In this case the user can easily read all the information in the image. At this point is possible to observe that the "image dimension control" is useless, and it is a good idea remove it (or at least move it in another part of the screen). Also

the other buttons in that group resulted not very useful, and also not very clear: this part of the interface must be carefully re-thought.

4. The interaction of the users with the personal output is particular: during the test the users read it rapidly, but actually they take from there only few information contained in 2-3 rows. Actually in each type of analysis in the most cases there are a few important information, and only in the other cases the user must read all the rest of the output (but in that cases he must). So a good idea is to highlight the more important parts (that varies based on the type of the analysis), to organize in groups the "sometimes important" parts, and to hide all the others (there are some rows that are reasonably important only in the debug phase, but before hide them it is better evaluate this point again with the super-users).

5. One of the worst problems is the maybe half of the users don't understand that if they right-click on the image they can access a lot of information such as other representation, drawing options, opportunity to zoom in or out, labels. In that cases almost all the users asked directly to the developer how to act this commands, because they are necessary to the correct use of the application. These controls are probably one of the biggest improvement between gAn and gAn Web, because give to the users new opportunities of understanding what is happening in the experiment through information related to the images (the previous version, before gAn Web, had only png-format images, without this controls), so the fact that the users often don't find it is a big deal. A solution can be create a label-advice to give the user a clearer vision.

6. Some users, when is in the textual-output page, have some difficulties in finding the command to move to the images pages (and vice-versa): probably the button is not sufficiently evident in the screen. This problem can be an impediment with new users so it is important to solve it, re-thinking the distribution of the buttons in this page, that are actually not very clear. At the end the adopted solution is a complete re-distribution of the output pages, that now can be switched by a unique Navbar, that contain all the important commands.

7. Users seems to be confused when they press start and must wait some seconds for the results, and some of them ask is the program is working correctly, even if there is an infinite

progress bar and a "wait message". The wait time is longer that we expect, so it becomes a usability problem. This is unacceptable, but we think that the waiting time is related to the server that is too busy: in fact this is not the only application that works on it. In the spring 2017 we are going to format the server and re-distribute some applications, so we hope that some seconds of wait will become less than one second (so, acceptable). According to this fact we hope to solve this problem working on the machine, not on the interface, obviously if on the new-configure server the problem will persist we'll have to create an exit-strategy for the user to eventually stop the wait, and give him more precise temporal information about the remaining waiting time, maybe with a percentage progress bar instead of an infinite one.

8. It would be a good idea think a system able to help the users to link the run numbers to something they search without using human memory: a view on the RunLog is necessary.

The second part, the one related with the surveys, has as goal to give an overall idea about the utility of gAn Web. This part of the test was unfortunately a bit less useful, because the results seemed to be too high if compared with the amount of problems founded in the first test (probably the users simply tried to be gentle with the developer). Generically the strategy based on the survey was used no more after this attempts (the direct tests with the users seems to be more promising).

The third part, the one that consists in the discussion with the user, was very helpful: almost all the selected users proposed new ideas (all of them are useful, the most of them are effectively implementable), and new functionalities. Some of the functionalities was asked by almost all the users, so they probably are absolutely important to improve the interaction quality. The biggest part of the users observations and ideas is coherent with the observation made observing them

Here is a list of observations obtained in the third phase, through discussions with the users:

1. Probably inserting a tooltip on each button the overall clearness of the product can grow, but it is important to be careful with the risk of covering with the tooltips other buttons. For some commands the tooltips must be more long and precise, because they are not so easy to understand (the pilot users can understand them better because they are very expert in the domain,

but it is not always clear for everybody; a more clear choice of what to write in the tooltips can make the application more usable).

2. Actually the names of some buttons are not clear. At this stage the developer is not a deep expert of the domain, but probably is more expert than at the beginning, so a general re-discussion of all the names of the buttons (and related tooltips) seems to be a good idea.
3. The application is though to be used in the standard AEgIS' experiment screens, that are quite large. But sometimes the users use it from home, and the application must be adaptive to smaller screens (there are some little problems, such as buttons overlapping).

6.4.4 Is the testing finished here?

No it is not. The application is entering right in this moment in the standard working processes of the AEgIS experiment. The web application can be prepared to meet the exigences that exist right now, but according to the dynamism of the experimental environment is highly probable that the continuous changing of the back-end (gAn) will lead to a need of constant upgrade of the requirements. In this situation the users will continue to utilise the application, to find new problems, and this fact will route the developer to a continuous process of adaptation.

Chapter 7

Design and Patterns

Developing gAn Web it was decided to design together the physical part and the conceptual part. This is the holistic approach to design. In this project the users play a special role: they are absolutely participatory, they are an important part of the developing team. The point is that in this project the needs of the users are very particular, and very hard to understand for a person who doesn't work in the physics world. So the users must take part in each decision and give continuously opinions and feedbacks, to avoid misunderstandings and to obtain a good result.

7.1 Used patterns

The use of some patterns can help the designers to create a better application reusing good solutions already known. The main source of patterns of this project is the website `http://designinginterfaces.com/patterns/`, based on the book "Designing interfaces", written by Jennifer Tidwell, in the edition of 2010.

7.1.1 Clear entry point

In each page there are only one or two entry points, clearly visible for the user. The first entry point is always the arrive from the logically previous page, and the second is in case the result of a "go back" button somewhere.

This result is quite easy to achieve because in the application there aren't an incredible amount of pages, and in each page, according the logic, is quite easy to understand where the user desires to go.

To enforce this result there is a block system: if the user accesses inserting directly the URL in the browser for example in the configuration page without passing through the normal path of the program the system understands that some cookies doesn't exist and re-forces automatically the URL to the first page: the Login one. These is a designer's decision according to that the limitation of the freedom of the user is acceptable to ensure to show him always consistent pages with consistent contents

In conclusion in each kind of task that the common user executes with gAn there is a "standard" path, hopefully very easy to find.

7.1.2 Wizard

The fact that each task in gAn has a "standard" path leads that each task executed with the web interface can be carried out using a wizard. The user loses some "freedom of movement" in the application, but in this way he is forced to follow the right way and can avoid easily errors or omissions. Is this a limitation? We must observe that if, for hypothesis an user is interested only in a textual output the program forces him to see also the image-based output, but, both output are in the same page, and the navbar allows the user to go rapidly where he wants.

7.1.3 Spatial memory

In each page the opportunities to move to another page (back to home, new analysis etc) or to execute tasks (confirm, download something, show images) are concentrated in the navbar on the top of the

screen. The hope is that in this way the user can automatically associate the position of the navbar with the concept of "commands". All the navbars in the program have the same buttons? Not exactly, but all the important commands are in a navbar. The goal is to create a "link" in the mental model of the user: this link is between "I search an important command" with "navbar position".

7.1.4 Grid of equals

This pattern recommends to position the elements in the screen in a way that creates a "grid". All the elements of the grids must have the same visual weight, to seem equally important to the user's eyes (unless hide some of them is the goal of the developer, there is a discussion about this regarding some commands in the image output page). This pattern is used in the page that allows the user to edit the configuration file of gAn, this disposition of elements is useful to give the user an ordered exposition of the editable options in the file.

7.1.5 Responsive enabling

There are some buttons in the web interface that executes tasks which need some pre-conditions. For example: to start the gAn program it is mandatory to insert one or more runs as input for the program. If this condition is not achieved the button is inactive, and only when the user inserts correctly the requested values it became clickable. A similar behaviour happens in the configuration page. In this way is possible to avoid useless errors and wastes of time, without create frustrating experiences for experienced users. In the intermediate version in order to say to the user if the button is clickable or not we used the colors red-green, but in the final version the use of the standard Bootstrap's graphic that for this point prefers to use the variation of opacity, seems to be a better idea (for aesthetic reasons).

7.1.6 Progress indicator

The starting of gAn is a complex task that takes some seconds. This waiting can be disturbing for the user, who can think that the program is crashed. Another problem is that is quite difficult at this moment estimate the waiting time with precision. A solution can be create an infinite progress bar, that cannot show to the user exactly how much time he must still wait, but can inform him that the system is effectively working and has receipt his request. In the finished product, with a completely working server, we expect that the waiting time is something around 2 seconds (in the worst case) , no more.

7.1.7 Go Back to a Safe Place

In each screen the user can easily return to the homepage , that is the common start point of all the possible tasks. When the user returns at this situation the program re-starts asking him if he prefers a multi-runs analysis or a single-run one, so the program is ready to start a new task.

7.1.8 Liquid layout

This useful pattern recommends to create a dynamic structure able to adapt itself automatically to the screen's dimensions. The concept of "adapt itself" doesn't include only the ability to modify the dimensions of each object, but also to modify the layout of the screen. This pattern is very useful in applications that must work both on laptops and on mobile devices. In the particular case of gAn Web the users never use mobile devices (a correct analysis request a big screen), so the application has to adapt itself only to screens with a dimension of minimum around 10 inches. For this reason the automatics system of

object-dimensioning provided by Bootstrap (based on the partitioning of the screen in columns and sub-columns) is sufficient to ensure that all the system is usable by all the "target" screens. In fact all the objects that compose the system's graphic can adapt themselves easily to all the expected screen dimensions.

7.1.9 Hover tools

This kind of helper allows the user to get information about the tasks that each object can carry on just moving the mouse over it. The interface of gAn Web is quite simple, but for a new user is useful to have a some tips about the components potentialities. Bootstrap provides some easy commands to create the tooltips accessible by a "on hover" event, giving to the developer a fast way to implement this pattern. What to write in these tooltips is not trivial, the developer needed the advice of the super users (sometime also of common users) to be sure about what to write.

7.1.10 Harmless default

This pattern is used in 2 ways:

1. In the homepage, where the user has to insert the run numbers (before he can press the start button) some place-holders are visible. These place-holders aim to teach the user how to insert correctly the numbers.
2. Another use of this pattern is related to the page that aims to modify the configuration file of gAn: here the user can see setted the last known working values, avoiding to re-insert values that he doesn't want to change. He also can, through the related button, re-set all the configurations to default values.

7.1.11 Same-page error messages

This pattern allows the user to understand immediately if and where there is an error in the inserted values, showing him a visible message in the page able to explain him how to correct the error. In most cases in gAn Web the values are inserted using dropdown menus, so it is impossible make errors (or better, it is impossible choose inconsistent values..), but in each part where there is a "free" input field this pattern is used: if the inserted value doesn't meet the requirements of a validator a message appears on the screen explaining to the user what to do to solve the problem.

7.1.12 Alternative views

This is an example of pattern that we tried to use, mainly in the intermediate version, but at the end we decided to leave. This pattern proposes to let the user choose among alternative views that are substantially different from the default view. GAn Web tried to apply this pattern in the intermediate version where the system showed images to the user: the user was enabled to choose if see a static image in png format, that is simple and contains all and only the needed information (or better, the information that the designer thought the user needs), or a dynamic, modifiable and navigable image in Root format, in which the user can decide exactly what to see. Also the user was able to choose the dimension of the image (little, medium, big). But to the test with users we found that actually almost nobody used this opportunity: the users always chose big images, root like version. So this choice opportunity was removed. But a smaller example of this pattern is still used: when the user see and image that contains more than 2 dimensions he can see it both through a projection on the plate and through a 3-D rotatable and accessible image.

Chapter 8

Implementation

8.1 Introduction

The designer has evaluated different options to create this software: at the end the option of creating a web application seemed to be the best solution. Some non-functional requirements, such as the goal to create a solution easily compatible with any machine, without installing nothing, and without creating problems related to the version of Root Framework or the version of gAn, are quite hard to achieve in a stand-alone application. In a web application instead it is easy avoid all the installation and compatibility problems, because it is sufficient solve them only one time on the server, and the user will never has to think about these points. Furthermore, a machine with a windows environment can in this way access to a service executed by a linux system, without problems.

8.2 The use of Bootstrap

The only framework used in the interface is Bootstrap (Root Framework is used in the back-end). Bootstrap is quite useful in the production of the front-end because allows the programmer to manage the dimension of each object on the screen in a smart way and to automatically adapt this dimension to the varying dimension of the screen. It was evaluated the possibility to use a framework for the part related to PHP, but it was preferred not to do so. The point is that the PHP part of the program is quite little and quite atypical: in fact the main task of the PHP part is to talk with the C++ scripts of Root, and for this kind of atypical situation the use of a framework can lead more problems that advantages.

Everywhere in the program is easily visible the sign of Bootstrap, in particular almost all the graphical parts are taken from Bootstrap's components (all visible and freely accessible here `http://getbootstrap.com/components/`). The goal of this choice is to exploit the wide diffusion of Bootstrap's graphics on the internet: a common user used to navigate on the net surely is used to see the graphical object of Bootstrap, and is unconsciously used to associate these symbols to their signification. So the using on the Bootstrap's graphics can improve the affordance of the interface.

So all the graphical objects in the application are taken from standard Bootstrap's graphics? No: in fact we checked during the development all the options case per case. Definitely we can observe that for practical and aesthetic reasons graphic objects taken from Bootstrap are often preferred to the alternatives.. so they are used in most cases, but their adoption is not automatic.

Bootstrap graphics is an enormous database of object, for this project only a few are used, in particular:

1. The buttons

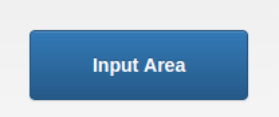

FIGURE 8.1: A "primary" button created with Bootstrap's graphics

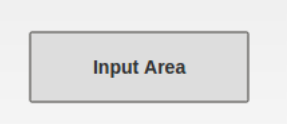

FIGURE 8.2: A "secondary" button created with Bootstrap's graphics

FIGURE 8.3: A "disabled" button created with Bootstrap's graphics

The main used button is the "primary", but for some secondary function is also used the "secondary". Bootstrap allows also to

use the "disabled" version, useful to let the user understand if a command is unacceptable. In the intermediate version there was some coloured buttons (green and red) to communicate the concept of enabled/disabled, but in the final version we preferred to use the default Bootstrap's behavior to enable/disable each button.

2. The navbar:

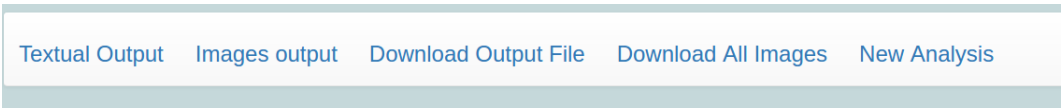

FIGURE 8.4: A navbar containing some buttons created with Bootstrap's graphics

This object allows to select a button by clicking on it: when the button is selected the graphical effect changes automatically the color, that becomes more dark, and gives the graphical impression of the button physically pushed, making the user aware of what is the selected button. It is the ideal solution to show options on the top of the screen. Putting some object together in a common structure clearly separated from the rest of the screen gives them a meaning of group (Gestalt's laws): in this case the goal is create a group of "generical commands" able to contain all the command that the user can use at this point of the execution.

3. The progress bar

FIGURE 8.5: An infinite ProgressBar created using Bootstrap's graphics

This solution makes the waiting time less frustrating for the user. The bar is infinite (without a percentage, it go on like an endless screw until the program is ready to show the results), because it is impossible to estimate precisely the amount of time that the user must wait (if the server works properly we talk about around 2 seconds.. no more).

4. The input fields

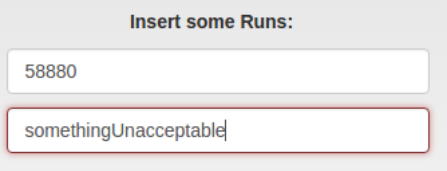

FIGURE 8.6: An example of input field taken from Bootstrap's graphics

with different colors if the input is acceptable or not. Also there are error messages to inform the user about "why" the input is acceptable or not and about "how" solve the situation.

5. The dropdown menus

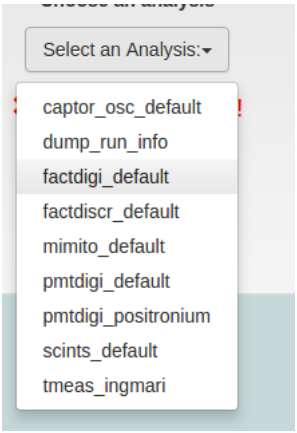

FIGURE 8.7: A dropdown menu created with Bootstrap's graphics

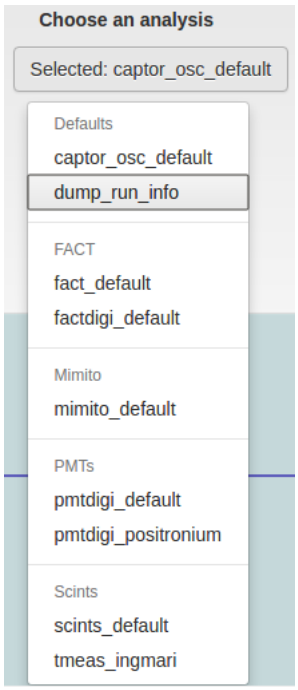

FIGURE 8.8: A grouped dropdown menu created with Bootstrap's graphics: it needs more space, but it is more clear

It is the common solution to force the users to choice a value among some options. According to the super users it is aesthetically better than the radio buttons.

6. The tooltips

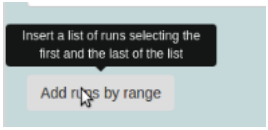

FIGURE 8.9: An explaining tooltip obtained from Bootstrap's graphics

The user can see this kind of tip just moving the mouse over the interested object. The tip explains briefly what the object can do. A tooltip on the name of the analysis and on the name of the sensors is very useful, but it needs to be written with the help of domain experts (at this stage some of the tooltips are mock-ups, so the written information is realistic but still not checked by domain experts).

7. The modal form

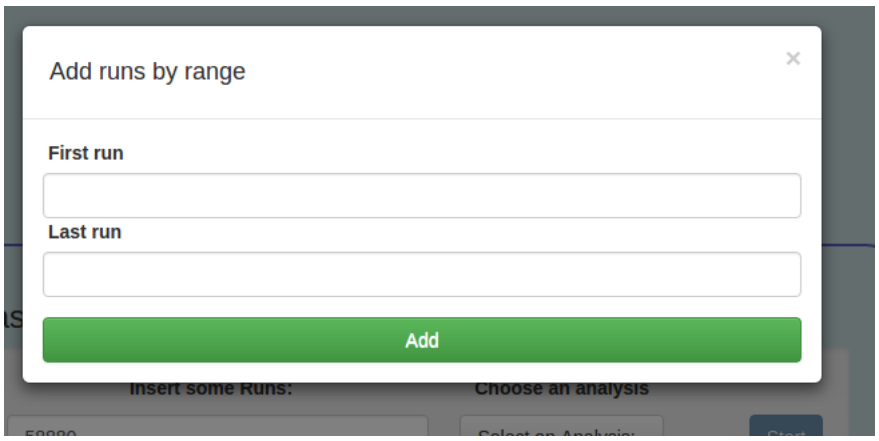

FIGURE 8.10: A modal form obtained from Bootstrap's graphics

This structure, named "modal form", is a window that appears on the screen as response to some event, takes the focus, and allows the user to execute some activities.

8. The "well"

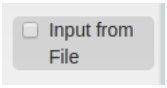

FIGURE 8.11: A well

This feature, named "Well" allows to highlight gracefully an element (or a group of elements) surrounding it with an edgeless square with different background color.

8.3 The use of non-Bootstrap components

The only one interesting component that isn't taken from a Bootstrap collection is the seekbar:

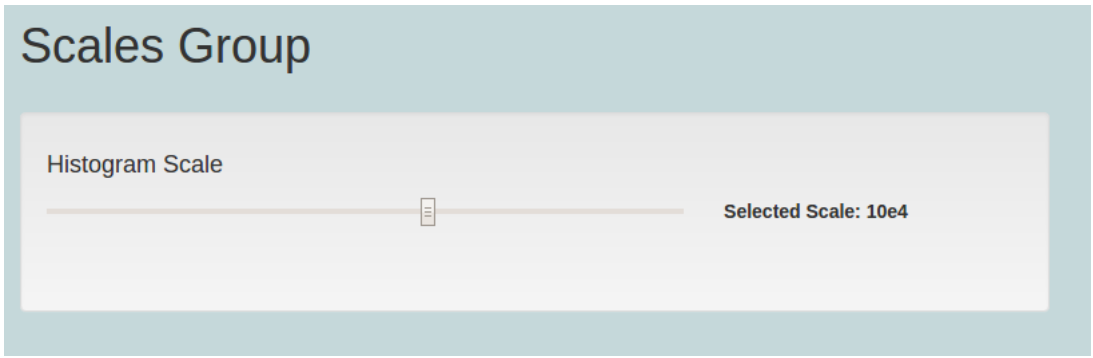

FIGURE 8.12: A seekbar

moving this bar forward and backward the user can set bigger of smaller values in the setting page. In this case the seekbar is exponential, so is better if the user has a continuous feedback on the right (selected scale M raised to the power of N) to know what value is going to insert.

Chapter 9

Conclusions

In this last chapter we try to summarize the obtained results of this interface and the observation about how the design process took place.

9.1 Conclusions

In this cyclical process of development, after much trying and retrying, we can surely say that we have tested a lot of different ways to give the user the best possible product. The big modifications between different stages, both the ones derived from the pilot users, and the ones derived from the general users probably identified the most of the problems on this application, that now, with the final version are hopefully solved.

The main advantages that we have enjoyed during the development is surely related to the possibility to get hints from the pilot users, that are both users and supervisors, and surely gave a big help to correctly interpret the needs of the typical users. Also the availability of the general users to test the application and to help with ideas and opinions was surely important.

I conclusion we must observe that the needs of the users are not stable, are continuously changing, so, even if gAn Web is designed to be easy to modify, the biggest challenge for the future is try to maintain the application up to date in a unstable and dynamic environment.