TODO: insert a title here

Author:
Andrea G.B. DAMIOLI

Supervisor:
Dr. Germano BONOMI

Master Thesis

Abstract

Master degree in Computer Engineering

TODO: insert a title here

by Andrea G.B. DAMIOLI

The AEgIS Experiment at the CERN aims to verify the weak interaction principle for antimatter. This document talks about "gAnWeb", 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. This document analyses from the Human Machine Interaction's point of view the production of this graphical interface.

Contents

Chapter 1

Introduction

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.

This program receive (continuously) in input a folder containing several terabytes of files in ".root" format, and some parameters; 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. This sensors produce data continuously (8 hours per day), and a software system validates and saves these data in file with .root extension.

The parameters 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 140 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.
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 "Ingmari's Code" can extract information about the temperatures of some elemental particles analysing how quickly they change direction 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 reduce the huge amount of raw data 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, chose 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 it 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 musts 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 are physicists, not necessarily specialized in computer science, so, create a friendly and easily learnable interface can avoid them problems and time wasting.

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.

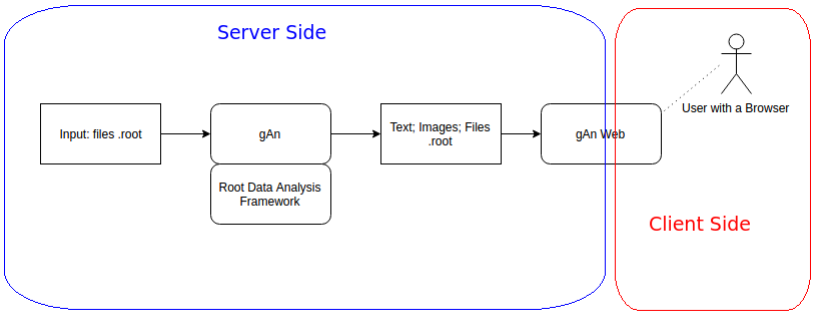

FIGURE 1.1: gAn - gAn Web simple scheme

Chapter 2

Organization of the project

We can firstly explain how this project is organized:

2.1 The role of the designer and the users

In this design process the interaction between designer and users is very strict: in fact for about 2 months, divided in blocks of more of less 2 weeks, the designer has worked together directly with the users.

It is interesting to note that the domain in which this application works is quite complex and different from the domains in which the designer is specialized, so, some periods of work directly with the users are necessary to let the developer understand better how does the domain works. 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 need are very specialized (related with the physics research world).

In this design process we can identify 3 main actors:

1. The designer, who produced the web interface (gAn Web)
2. Two pilot users (university professors in Brescia), that are also co-developer of the application behind this interface (gAn). So they act both the role of the users, and the role of the supervisors.
3. A group of users (around 10-12) in Geneva. This main group is used for the final testing of the application.

The design process can be also divider 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 the little group of pilot-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 improve the designer's knowledge of the domain. At the end of this stage, the test with the two pilot users helps to correct the way.

2. An intermediate stage, more complex, with some advanced functionalities obtained listening the request of the pilot users. This version is installed in the AEgIS control room and the users use it. The designer needs at this point to execute some test 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.

   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. 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, the application is designed to be adaptable, and to try to satisfy the unknown needs of future users.

According to this division we can organize the following chapters in this way: each chapter is related to a stage of the development. 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 the resulting prototyping (and implementation)

Chapter 3

Early version

At this point is important to analyse which are the requirements of gAn web at the first version:

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 chose the run (only one run, for the moment) in which he is interested, using a input field. This field has 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 and if the inserted value is not acceptable. It is 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 inexistent.
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

understand 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).
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 chose 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

All the versions have 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.

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 click on the Show Images button,

he go in the images pages, he select 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 take 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 go 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 return in the homepage. Now he can like before insert the number, and get the outputs in textual and in images format.

3.4 Prototypation

TODO

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. 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 open a "modal" that allows the user to chose 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 fit the needs of the users, but the tests with the whole users group will probably solve this doubt.
2. The user can chose 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, lo 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 behavior.

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 chose 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 allows 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 chose 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 informations 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 have this file (this root file contain more information than the 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 first little group of user prefers the dropdown menu to 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, chose 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 user's needs in this part seems to be too difficult and not useful). 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 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.

   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.

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.

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 informations, and very difficult to understand, but on this question the pilot users (that are physicists) questioned answered that the output is perfectly 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 informations 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 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 make doesn't make sense the system show 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 chose 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 chose 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 complete the login, in the home page there is a button name "Chose 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 chose 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 chose a Root version. The user complete the login, in the home page there is a button name "Chose gAn version", clicking on this button the user is sent to a page where, he, through a dropdown menu, can chose 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.
6. 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), 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.
7. 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 Prototypation

TODO

Chapter 5

Final version

Here will be explained the features of the final version, and how we are arrived at it.

5.1 Functional requirements

The last version (from now "gAn Web v3") is a modified version of the intermediate version. The second version's modification come from two sources: some adding requirements are proposed from the pilot users (that are also supervisors); These adding requirements are the following:

1. The main kinds of analysis are now more clear: they are 4-5, and they are quite stable. Their role and when they are useful is now steadily fixed. Each user, according on his task, knows (should know) in every moment which analysis fits better the situation, so, the best solution is allows 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). 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 the user chose directly if he wants work with a single run or more than one.
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.

Furthermore, the version 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 and information. The impact of the debut with the users highlights some problems to be overcome, and from the analysis and the solution of these problems the requirement of the last version was defined. The 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 divided 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. A similar reasoning can be done for the dropdown that allows the user to switch between the vertical and carousel menu (everybody use).
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 need 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 all the users modifies it quite often (manually). So is better to let them to do it by the interface

5.2 Scenario based functional analysis

TODO

5.3 Prototypation

TODO

Chapter 6

Prototypation

6.1 Early version

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 experiment some technical solutions to create the features needed to achieve the goals. At this stage the application is evidently incomplete, there are numerous deficiencies and inconsistencies and the goals of the Human Machine Interaction are ignored. All this point are corrected in the second version, and this first version can be used to show a "before-after comparison". GAn Web is a web application (as the name suggests) so the interface is created using HTML and CSS. This languages are very advanced and allows the programmer to create and modify an interface very easily. The framework Bootstrap, used in this project, even improves the performances of these languages. So all the prototypes are directly created by code, without paper-based mock ups.

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

1. The homepage:

FIGURE 6.1: The early 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:

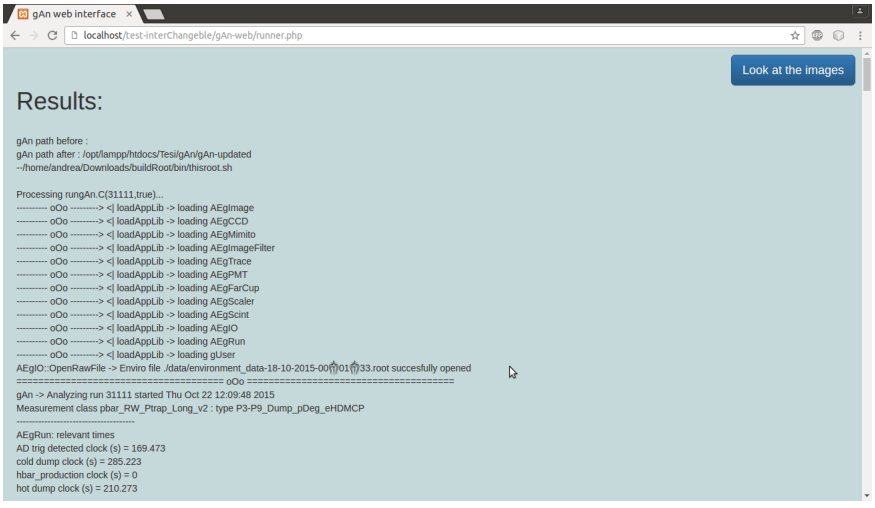

FIGURE 6.2: The early page related to the textual output of gAn Web

The textual result of the computation is visible: it seems to be too long and incomprehensible, but for physicists it is quite clear. 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 6.3: 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 6.4: 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.

6.2 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 Machine Interaction.

6.3 Late version

The late version is based on the early version, some pages (and functionalities) are added, some existing pages are improved. Following all modifications are explained.

6.3.1 Modified pages

The homepage:

FIGURE 6.5: The homepage of gAn Web without input

FIGURE 6.6: 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 10 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 6.7: 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 take 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. In the following image the progress bar:

FIGURE 6.8: The progress aimed 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).

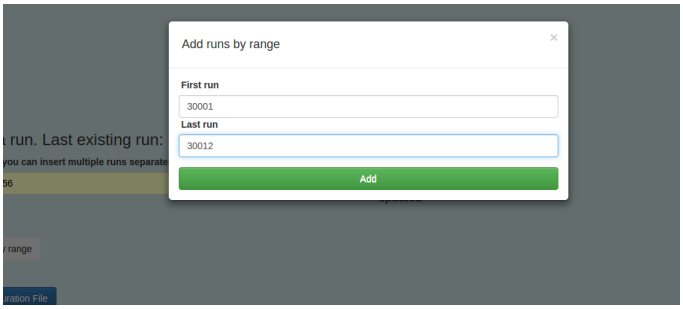

FIGURE 6.9: The modal opened by clicking the button "Add runs by range"

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 he doesn't change the configuration (the default branch is the last used, because usually when a group of users starts to use a branch, it continues to use it for some days)
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:

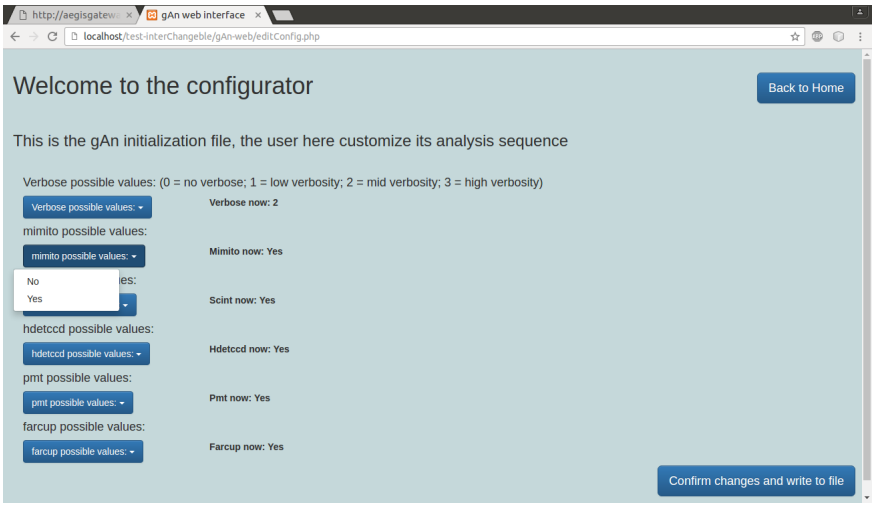

FIGURE 6.10: The edit-configurator page of gAn Web

This page doesn't use anymore radio buttons, because some users ask the developers to use dropdown menus (for aesthetic reasons). The user can read near the button the currently selected value for each field. There aren't tool-tips able to describe the signification of each field here, because the users to which gAn Web is intended have a perfect understanding of the names and the features of each sensor (mimito, scint, farcup, etcetera are sensors).

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

FIGURE 6.11: 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:

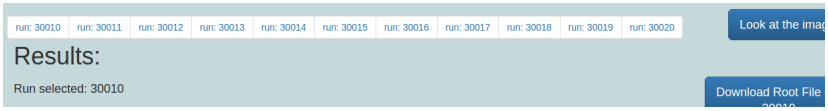

FIGURE 6.12: By this "navbar" the user can choose the run results to show

   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: also 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:

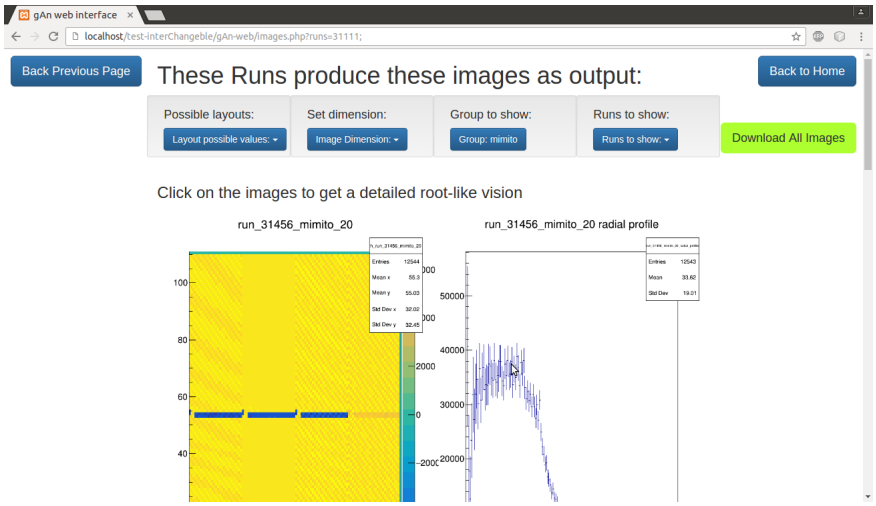

FIGURE 6.13: This page shows all the images that gAn produces in output

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 late design introduces this requirement because commonly the users want to 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:

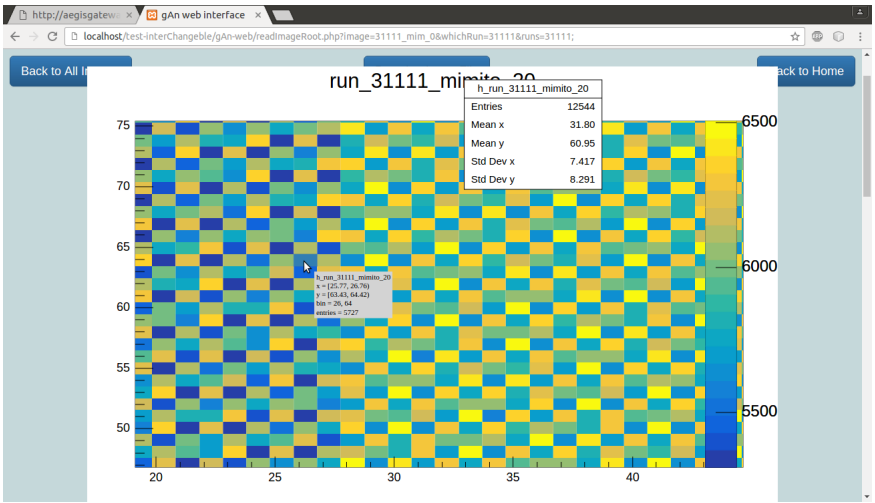

FIGURE 6.14: Moving the cursor the system shows the value of this histogram in the selected point

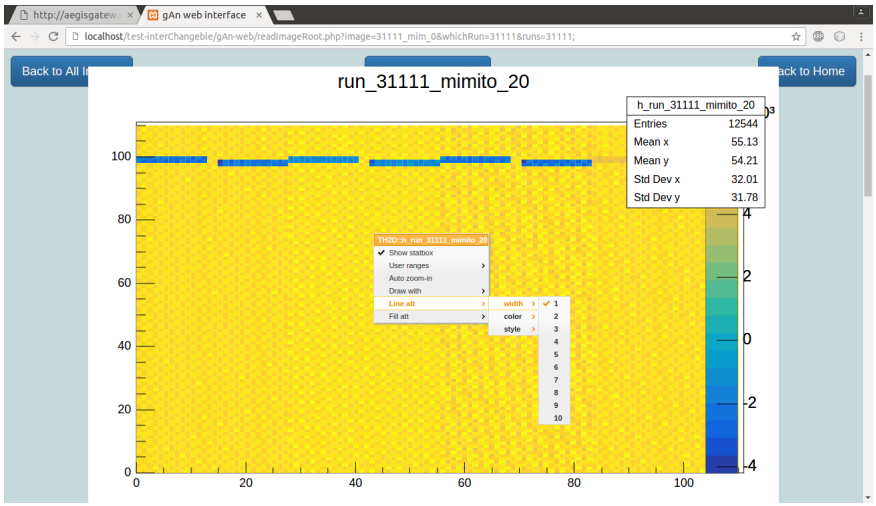

FIGURE 6.15: The user can modify numerous settings in the generated image

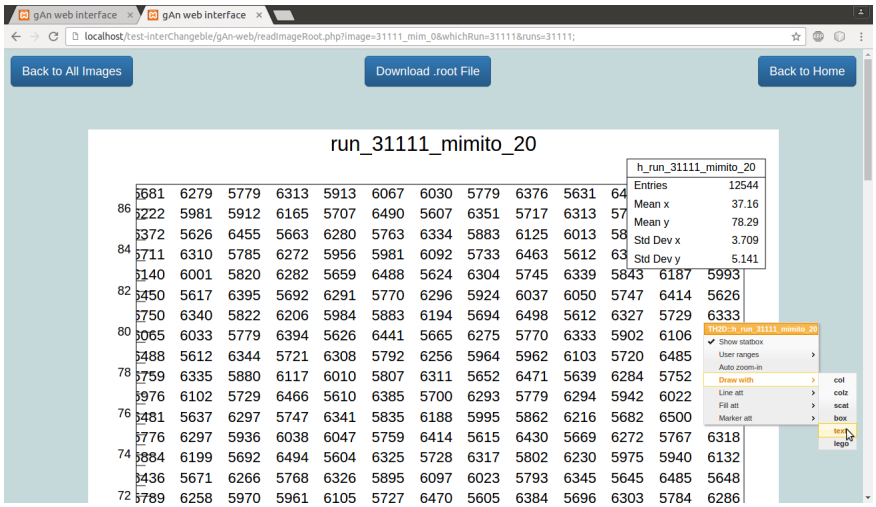

FIGURE 6.16: The user can show the histogram not only in traditional format, but also in a numerical format where the numbers are the value of the function in their position

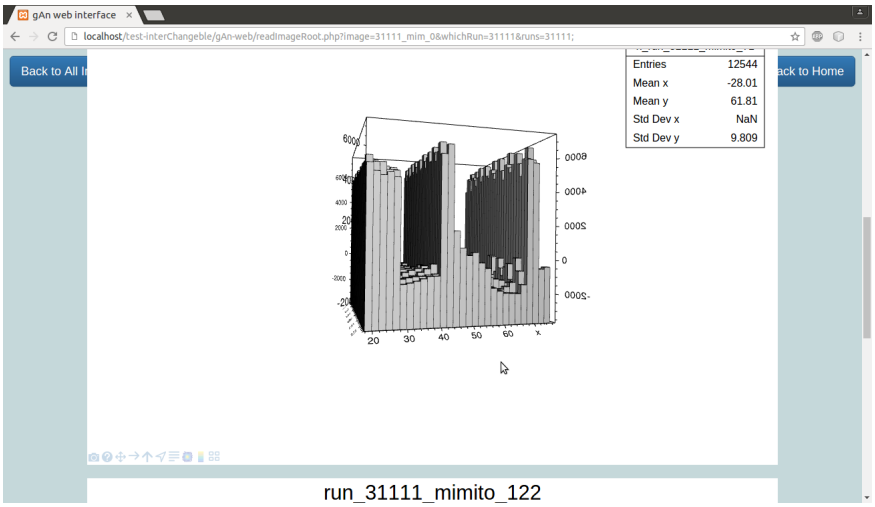

FIGURE 6.17: Another solution is to generate a 3d image in lego-style of the histogram

6.3.2 Added pages

Some pages in the late 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:

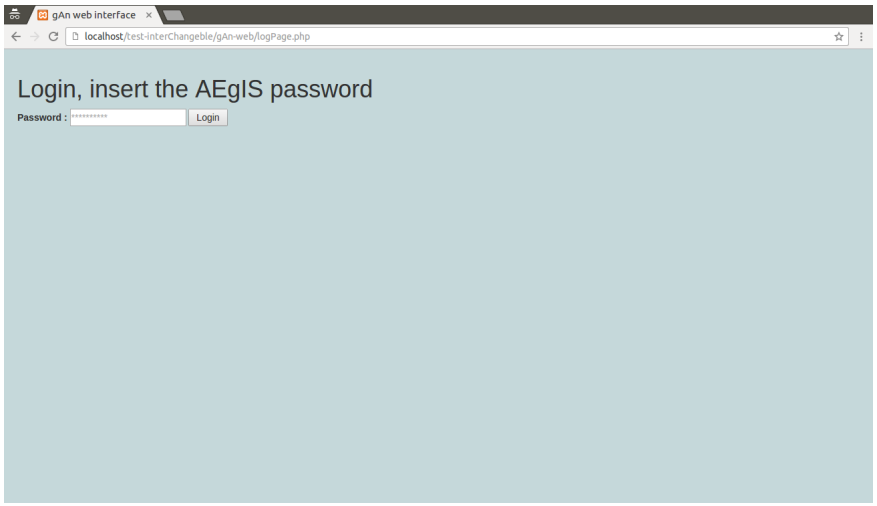

FIGURE 6.18: Simple login page

In the late 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

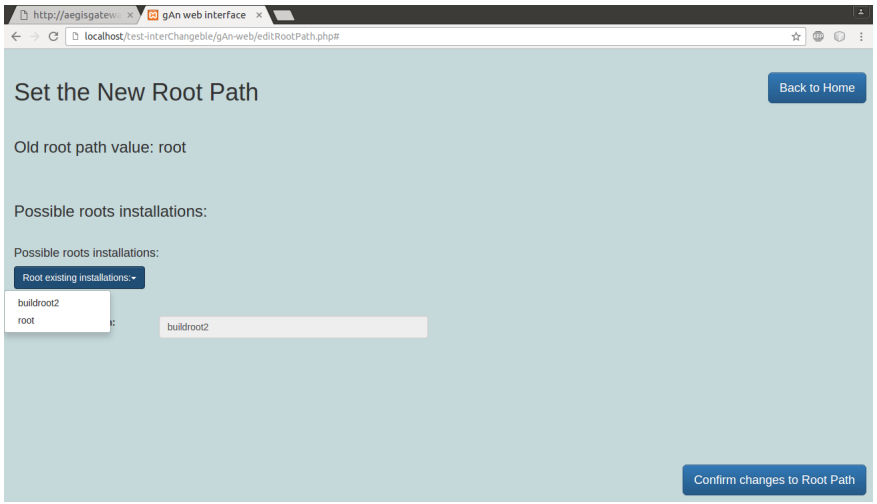

FIGURE 6.19: Page where the user can choose the Root version to use

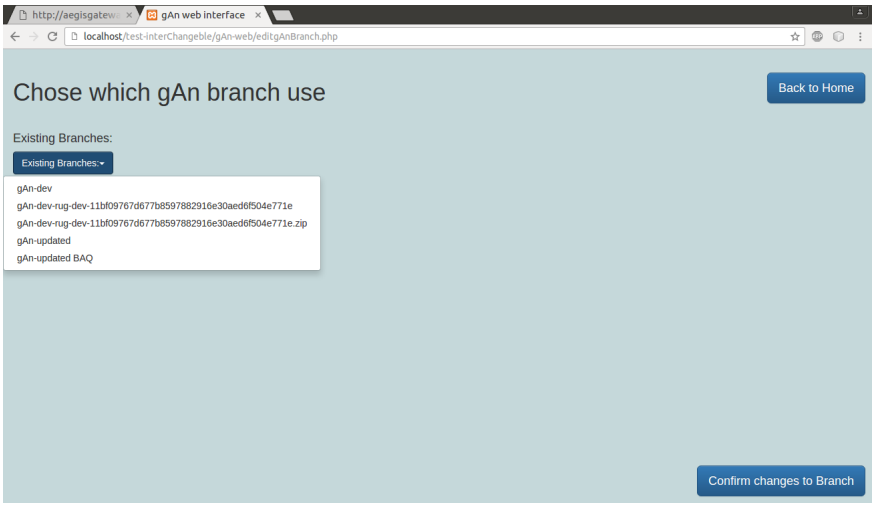

FIGURE 6.20: Page where the user can choose the Root version to use

Chapter 7

Implementation

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.

The only framework used in the interface (Root Framework is used in the back-end) is Bootstrap. 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 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. Bootstrap graphics is an enormous database of object, for this project only a few are used, in particular:

1. The buttons

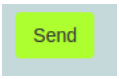

FIGURE 7.1: A button created with Bootstrap's graphics

2. The navbar: This object allow to select a button by clicking on it: when the button is selected the graphical effect changes automatically the color, that became more dark, and give the graphical impression of the button physically pushed, making the user aware of what is the selected button.

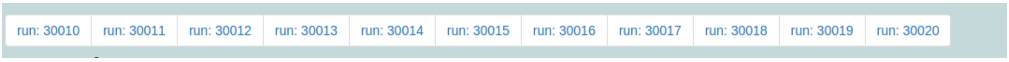

FIGURE 7.2: A navbar containing some buttons created with Bootstrap's graphics

3. The progress bar

FIGURE 7.3: 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.

4. The input fields

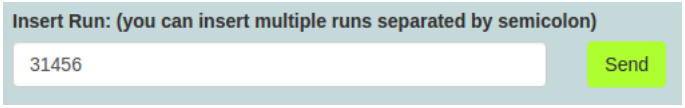

FIGURE 7.4: An example of input field taken from Bootstrap's graphics

5. The dropdown menus

FIGURE 7.5: A dropdown menu created with Bootstrap's graphics

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

6. The tooltips

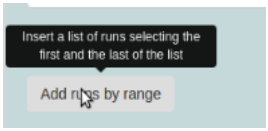

FIGURE 7.6: An explaining tooltip obtained form Bootstrap's graphics

The user can see this kind of tip just moving the mouse over the interested object. The tip explain briefly what the object can do.

7. The modal form

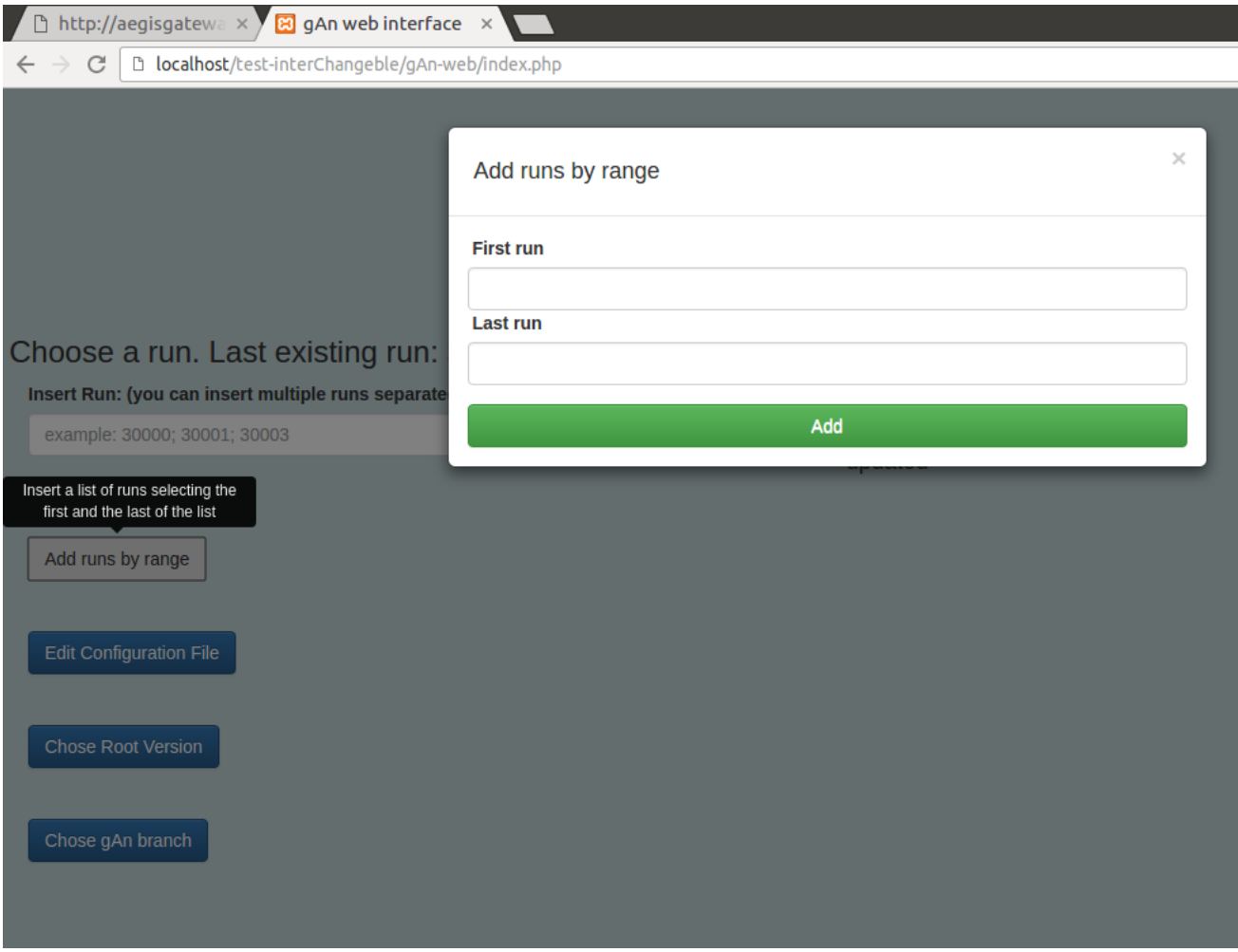

FIGURE 7.7: A modal form obtained form Bootstrap's graphics

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

7.1 TODO

Chapter 8

Design

Developing gAn Web it was decided to design together the physical parts 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 arrive to a good result.

8.1 Used patterns

The use of some patterns can help the designers to create a better application reusing good solution 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.

8.1.1 Clear entry point

In each page there are only a few entry points, very visible for the user. 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 understand where the user desires to go. In fact in each kind of task that the common user executes with gAn

there is a "standard" path, and using this advantage it is possible to reduce the entry points in each page to 4-5.

8.1.2 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.

8.1.3 Spatial memory

In each page the opportunities to move to another page(back to home, show images etcetera) or to execute tasks (confirm, download something, etcetera) are concentrated in the top of the screen or, if they are more then 2, in the top and in the right. They are all in fixed position, so even if the user scrolls down the page can be sure to see them in the same positions. The hope is that in this way the user can automatically associate the position of the buttons with the task that these buttons can execute.

8.1.4 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.

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.

8.1.5 Responsive enabling

There are some buttons in the web interface that executes task which need some pre-condition. 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 red and inactive, and only when the user inserts correctly the requested values it became green and clickable. A similar behaviour happens in the configurators. In this way is possible to avoid useless errors and waste of time, without create frustrating experiences for experienced users.

8.1.6 Progress indicator

The starting of gAn is a complex task that take 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.

8.1.7 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. In the page related to

the analysis of the images the user can also return to the page where all the images are shown, because this page is a good point to start the image analysis.

8.1.8 Modal panel

This pattern is quite useful in some situation, because it forces the user to do some stuffs before he can return to the normal execution of the program. This pattern is used in the homepage where the user can choose if he prefers insert some runs by selecting the first and the last of a list. In this situation a modal is opened and he inserts the requested values (that must be valid). The user can exit from the modal only by canceling the operation (through the "X" button) of by inserting valid values. Also this solution helps to avoid useless errors.

8.1.9 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, so the application has to adapt itself only to screens with a dimension of minimum 13-14 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.

8.1.10 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.

8.1.11 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 (as is shown in the figure

   These place-holders aim to teach the user to insert correctly the numbers: separating them with a semicolon (if the user uses a dot, a slash, or a dash, the application can repair the error and run correctly the same).

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 as default the last known working values, avoiding to re-insert values that he doesn't want to change.

8.1.12 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, 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.

8.1.13 Alternative views

This pattern proposes to let the user choose among alternative views that are substantially different from the default view. GAn Web applies this pattern where shows images to the user: he can show if he prefers to see an 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.