Implementing a New Graphical User Interface For GameCourse - GameUI

Patrícia Isabel Dias da Silva
patricia.i.d.silva@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2021

Abstract

Education is taking a step forward into the new technologies. A Learning Management System (LMS) is a system that handles all the process related to learning, through which a teacher can create a course and manage the activities related to it. Like any other system, its User Interface (UI) impacts its acceptance, which means that although it may have well implemented features, it can be rejected if the user is not able to understand what he/she can do with the system and how. GameCourse is a web-based LMS used at the Multimedia Content Production course at Instituto Superior Técnico, that uses a basic and minimalist UI where users get confused and lost while performing simple tasks since they feel like there is nothing there to help them. The purpose of GameUI is to create a user-friendly UI for this system, making it clear, consistent, easy to use for either experts or beginners, so the user can take full advantage of the features of the system.

Keywords: Learning Management System; Graphical User Interface; Usability; User Experience.

1. Introduction

Throughout the years teachers and researchers have been trying to find an efficient way to improve student's performance and engagement in classes. New technologies have been incorporated in and out of the classroom on the past years including the use of Learning Management Systems (LMS).

A LMS [1] is a framework that handles all the process related to learning, through which a teacher can create a course, manage its activities, assign different learning paths to his/her students and gather results and statistics. Moreover, a LMS can even integrate game elements (gamification) and make the learning process more fun and exciting for the students [5, 4]. Applying a LMS in a course can bring a lot of benefits [1]: each student can go at his/her own pace; and the teacher knows exactly where each student needs help to give precise feedback and support.

However, the efficiency of LMS is not only based on the its features, it also relies on the user interface (UI), as it is through it that the user can interact with the system. We can have a perfect system with the perfect features and still have it fail outside the lab if the user does not understand how to use it, like the case of the Opensource LMS Illias [6].

For the past ten years, GameCourse has been used on the Multimedia Content Production course from the Msc. of Information Systems and Computer Engineering at IST. It is a web-based LMS that allows anyone who wants to teach to create and manage their courses. Besides creating a course and adding the corresponding Students and Teachers, we can define different roles and associate them with a user. We can also create personalized pages, restrict their access depending on the viewer's role, and have dynamic information also depending on the viewer. Furthermore, it is possible to enable gamification related modules, making the course more fun and interesting.

Although a lot of changes have been done through the years to improve GameCourse [3, 2], its UI has only had two version and is seen as "old". It gets users confused while performing basic tasks, which may lead them to loose interest on the system. The objective of this thesis is to develop a new and more user-friendly Graphical User Interface (GUI) for the GameCourse system, that respect the principles of UI, Usability, User Experience (UX) and make it easier for teachers to create and manage their courses.

2. Related Work

There are more and more LMSs every day and it is hard for a teacher to choose which system is the best. That's why studies, that bring up special features and compare their usability are so important.

Back in 2007 Machado and Toa [7] made a

study where they compared the UX of Moodle and Blackboard through questionnaires using the Likert scale. From the four teachers questioned, 3/4 said that they would rather use Moodle. From a group of students that tried both systems, 71% reported that Moodle was easier to use and 75% would prefer to use Moodle instead of BlackBoard. The neutral level on the Likert scale is still very populated, which means that both Moodle and Blackboard still have some usability flaws.

Later in the year 2008 a group of six researchers did an Heuristic evaluation about Moodle, Sakain and dotLRN [8] with 200 checkpoints. Their results showed that the LMS with highest compliance for the total of checkpoints was dotLRN with 78%, followed by Sakai with 77%, and Moodle being far behind with 68%. The heuristic where all the systems had worst results was H8 - ”Flexibility and efficiency of use”, with values below 45%. The heuristics where they had better scores were H5 - ”Help users recognize, diagnose and recover from errors” and H9 - ”Aesthetic and minimalist design”, with all the systems getting above 75%.

In 2016 [9], Sahid and Santosa analyzed the UX of Moodle and Schoology, using the User Experience Questionnaire. On a scale of -3 to 3, Moodle's highest score was 1.5, regarding whether the system was understandable or not, and Schoology's was 2.0 on whether the system was good or bad. Regarding each category: Moodle's highest points were Perspicuity and Efficiency with values just above 1.0; Schoology highest points were Attractiveness and Efficiency with more than 1.5 points. It is noteworthy that on Schoology, only one category scored below the 1.0, Novelty, which puts Schoology in front of Moodle in terms of UX.

As no information was found regarding the visual elements of these LMSs, we conducted our analysis to identify GUI patterns among eight systems. Regarding the structure of the systems: 6/8 used breadcrumbs to describe the menu path, the other two used no elements as they not have enough depth of pages to require a path; 7/8 used a navbar to show the possible tasks/pages of the system, whereas the other used a dropdown menu and the main page to display them; for the main page 2/8 used a news feed, 6/8 showed the course listing, from these six, three of them have a special main page for the admin user where 2/3 show statistics and the other allows a custom page.

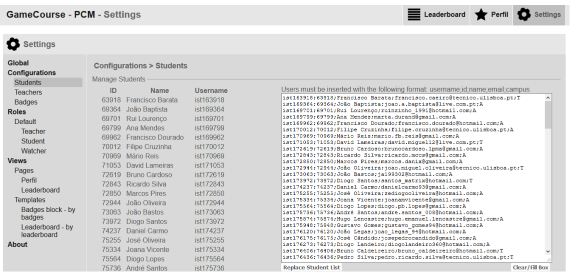
When adding new information to the system, two options were presented a modal and a new page. For 3 to 5 inputs, most of the systems opted for a modal, while with more than ten inputs, they opened a new page. To show information regarding the users, courses, and extra features, the most used elements were tables and cards. Nonetheless, simple lists were also found. In some cases, there was a need to encapsulate information, and the accordion was the chosen option. 2/8 systems showed an interesting additional element to show information, the slider, which on one of them ended up creating visual noise.

New pages, on the majority of the systems, could only be done with text, html and css, making it impossible to create dynamic pages like in GameCourse. Only 1/8 systems allowed extra customization by giving the option to select the layout and the correspondent content of the admin dashboard, but within limited options.

3. The GameCourse system

On the GameCourse system we had several concepts that defined the system's structure. The *Course*, to save all the base information of a course and to serve as reference to add additional information and functionality. The *User*, which kept the base information of a user and defined if it had admin privileges. Then we had the *Course User*, the connection between the last two concepts, saving all the user's information exclusive to a course. On these we could attribute *Roles*, to organize the users within groups. Finally, the *Modules* that could be activated within a course and extend its functionalities. One of the existing modules was called *Views*, which allowed the creation of HTML structure that could be used on *Templates* or *Pages*. On these views we could use the *Expression Language*, which fetched information from the system to be displayed or iterated, to create multiple elements with the same structure. The information correspondent to these concepts was saved and managed on a MySQL database populated through API requests and external scripts that would fetch information from external data sources, like user's information and module's specifications.

The initial UI was simple and minimalistic, used a monochromatic color scheme, but made no distinction between sections, and the system and course domains.



ID	Name	Username
63918	Francisco Barata	ist163918
69364	João Baptista	ist169364
69701	Rui Lourenço	ist169701
69799	Ana Mendes	ist169799
69962	Francisco Dourado	ist169962
70012	Filipe Cruzinha	ist170012
70969	Mário Reis	ist170969
71053	David Lameiras	ist171053
72619	Bruno Cardoso	ist172619
72843	Ricardo Silva	ist172843
72850	Marcos Pires	ist172850
72944	João Oliveira	ist172944
73063	João Bastos	ist173063
73972	Diogo Santos	ist173972
74237	Daniel Carmo	ist174237
75255	José Oliveira	ist175255
75334	Joana Vicente	ist175334
75564	Diogo Lopes	ist175564
75736	André Santos	ist175736

Users must be inserted with the following format: username;id;name;email;campus

```
ist163918;63918;Francisco Barata;francisco.caeiro@tecnico.ulisboa.pt;T
ist169364;69364;João Baptista;joao.a.baptista@live.com.pt;A
ist169701;69701;Rui Lourenço;ruinzinho_1991@hotmail.com;A
ist169799;69799;Ana Mendes;marta.durand@gmail.com;A
ist169962;69962;Francisco Dourado;francisco.dourado@hotmail.com;A
ist170012;70012;Filipe Cruzinha;filipe.cruzinha@tecnico.ulisboa.pt;A
ist170969;70969;Mário Reis;mario.fb.reis@gmail.com;A
ist171053;71053;David Lameiras;david.miguel12@live.com.pt;T
ist172619;72619;Bruno Cardoso;brunocardoso.lpma@gmail.com;A
ist172843;72843;Ricardo Silva;ricardo.mccs@gmail.com;A
ist172850;72850;Marcos Pires;marcos.danix@gmail.com;A
ist172944;72944;João Oliveira;joao.miguel.oliveira@tecnico.ulisboa.pt;T
ist173063;73063;João Bastos;ja199302@hotmail.com;A
ist173972;73972;Diogo Santos;santos_matrix@hotmail.com;T
ist174237;74237;Daniel Carmo;danielcarmo93@gmail.com;A
ist175255;75255;José Oliveira;zediogooliveira@hotmail.com;A
ist175334;75334;Joana Vicente;joanamvicente@gmail.com;A
ist175564;75564;Diogo Lopes;diogo.pb.lopes@gmail.com;A
ist175736;75736;André Santos;andre.santos_008@hotmail.com;A
ist175874;75874;Hugo Lencastre;hugo.emanuel.lencastre@gmail.com;A
ist175948;75948;Gustavo Gomes;gustavo_gomes94@hotmail.com;A
ist176120;76120;João Legas;joao_legas_94@hotmail.com;A
ist176175;76175;José Cândido;josepedrocandido@gmail.com;A
ist176273;76273;Diogo Landeiro;diogolandeiro360@hotmail.com;A
ist176406;76406;Bruno Caldeireiro;bruno_caldeireiro@hotmail.com;T
ist176436;76436;Pedro Silva;pedro.ricardo.silva@tecnico.ulisboa.pt;T
```

Figure 1: Users page old system.

3.1. User Tests

To collect more information about the UI problems of the current system, we conducted User tests with a group of ten users, where they were asked to think-a-loud while performing a set of tasks.

On the tests 30% of the users reported problems with the labeling and look of the navbar as ”it was not obvious to me that the same button, settings, would lead to different pages”. When trying to create a new course all of them went first to the course listing page and just then to the settings, where it was possible to finish the task.

Although 100% of the users were able to add a new user to a course: 30% had trouble finding the correct page; 100% got confused with the button ”Replace all”, as they were looking for an ”Add” button instead; and 50% showed frustration towards the way of inserting data (Figure 1).

3.2. Requirements

With all the information collected during the user tests, we were able to create the list of requirements for this project:

- **Full restyle**, so it stops looking ”old” and ”outdated” as users reported;
- **Page reorganization**, to put together all the related information, divide non related information and make sure the flow of the system is correct. New subsections will also be created to complement and complete this requirement;
- **Settings retouch**, so all the information related to the system and course settings is easy to see and to access, using more clear and familiar menus with self-explanatory labels;
- **Documentation rebuild**, to make it easier for both beginners and experts users to find specific information;
- **Creation of modules configuration pages**, to remove the external script that gathers information about the modules and make them easier to configure;
- **View editor fix**, to make it less painful to use and provide extra help for the most confusing sections.

4. GUI Design

The next step on this project was designing the new GUI of the system. We started by creating the new page organization (Table 1), where all the information and actions related to a concept are together on the same section of the system. To create a new course the user no longer needs to access the settings, because all the actions related to courses are on the courses page.

To make the drafts of the GUI we used an online design workspace called Whimsical[1], which allows us to build wireframes, where we created the pages' structure while having a real notion of sizes and proportions of elements and white space.

[1]Last version of the wireframes on whimsical: https://whimsical.com/tese-Tv7J6qNoKvSuyuPwY5ZME9

Inspired by the analyzed LMSs, we decided to use modal dialogue boxes to require user input. There we placed simple outlined input boxes with only a placeholder to define them, a caption aligned at the left with the inputs, a close icon on the top corner, and a save button aligned at the bottom right with the inputs while leaving a significant margin at the sides of the modal. To finish the general look of modals, we added a semi-transparent background layer behind them. We also use modals to confirm destructive actions on the system. However, all the information here was centered, and two buttons were put available to either cancel or proceed with the action.

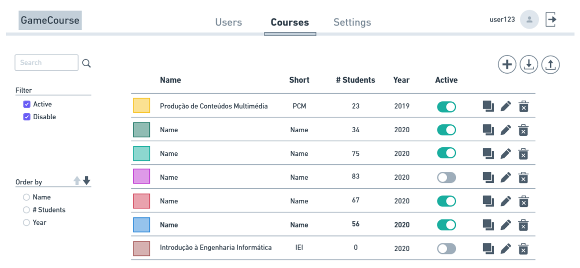
Scope	Pages	Actions
System	Main Page	
System	Courses	add, duplicate, edit, delete, access, import, export
System	Users	add, edit, delete, give/remove admin permissions, import, export
System Settings	Installed Modules	import, export
Course	Users	add (new and existing), edit, remove, add/remove roles, import, export
Course Settings	Roles	create, move, delete, define landing page
Course Settings	Modules	enable/disable, configure
Course Settings	Views	edit, create page/template, create/change specialization, globalize template, use global template

Table 1: Page organization on the new GUI.

We decided to keep the navbar on the top of the page but removed the icons of each page. All the titles were then centered, the system's logo was placed at the left side, and the username and logout icon at the right.

On the listing pages: Users and Courses, we decided to use a table to display all the information needed and allow item manipulations. We used a border separating each item, leaving a clear line that informs which information belongs to which object. To visually separate information display from actions, we used standard icons for each action, including a pencil icon to edit and a bucket to delete. For Boolean type information, we chose on/off buttons as they allow instance interaction while giving the value status. To make it easier to find a particular item on these pages we added a search, filter and order section fixed at the left side (Figure 2).

For the Modules and Views pages, we used cards to display each item, as they have visual elements that represent each one. To the items where status information was needed, we added a status bar with the colors green and red at the card's bottom. To keep the consistency among these two pages and the ones before, all the action buttons



	Name	Short	# Students	Year	Active
🟨	Produção de Conteúdos Multimédia	PCM	23	2019	●
🟩	Name	Name	34	2020	●
🟩	Name	Name	75	2020	●
🟪	Name	Name	83	2020	○
🟥	Name	Name	67	2020	●
🟦	Name	Name	56	2020	●
🟫	Introdução à Engenharia Informática	IEI	0	2020	○

Figure 2: Courses page - wireframe.

for add, export, and import were placed at the top right corner of the page.

On the documentation pages, we added a sidebar to guide the user between available sections on the "How to" page and available libraries on the "Functions" page. We went for a tab style for the sidebar due to the documentation page's resemblance to a physical folder of documents. Inside each section, the text content was kept the same. However, the functions list was replaced by an accordion with all the function's names that opens to show the correspondent descriptions. To match the tables' look, each item is only divided by a top border, and at the end of each line, we placed arrow buttons to open and close the accordion.

Over the view editor we restyled each view element to match the system current style, added a slider at the side as a shortcut for the documentation pages and for a tutorial. These tutorial would be available to explain each section and step of the view editor through highlighted boxes with "next" and "previous" options to navigate it. Also, to let the user know he/she is on tutorial mode we added a bar at the top of the page to remind him/her. To add a new element to the view we created a modal mainly visual where we select the options through icons instead of a dropdown.

4.1. Low Fidelity Prototype (LFP)

With all the page's wire-frames and a set of extra intermediate screens, we created an LFP over the platform called marvel[2] with a set of tasks to simulate the system behavior. This was later used to perform a Heuristic evaluation with seven UI & UX experts, where 57 problems were reported. From those, 70% were declared as level 2 or 1 on the severity scale, and only four were declared as level 4, the hights scale. The most affected heuristic was H4-"Consistency and Standards" with 15 problems mentioned, followed by H2-"Match between system and real-world" with 14 problems reported. These included things like missing attributes of the objects and wrong labels. This evaluation allowed us to correct most of the problems before the developing stage.

[2]LFP: https://marvelapp.com/prototype/6ifjeg5

5. Development

To complete the development phase, we implemented one page after another with concern for the correspondent concept. We started by preparing the PHP class and database in case of missing attributes. Then prepared the API by altering the existing functions and adding new ones to collect all the needed data and submit all the user's actions. After defining the pages' structure, we were able to implement all the functionalities of the GUI and make the system work. In this section, we will highlight some of the work done during this phase.

5.1. Navbar resize

When creating a new page on a course, a new option is also added to that course's navbar. This ability can lead to a scalability problem. To fix it, we created an algorithm that will check the size of the navbar, and if it is bigger than expected, it will move some of the options into an "Other Pages" dropdown menu. This algorithm is run every time the page is resized, reloaded, and when we enter or exit a course. On the resize case, to consider the possibility of having more space after the action, we first reset the navbar, placing the options that were already on the "Other Pages" dropdown back to the main menu and just then run the algorithm. This change on the navbar is made considering that we added a dropdown at the end of the menu for the settings and that this has to be the last option on the navbar.

5.2. Courses page for admin and non-admin users

The Courses page is the only listing page that can also be accessed by non-admin users. Because of that, we created two versions of it. For admin users, we displayed all the system courses and made available all the actions. We created the new and edit modals with inputs for all class attributes and included a color-picket to make it easier to select the course color. We also added the delete modal to confirm the destructive action and added quick edit requests, for the active and admin attributes, by making an immediate submission of the change on the on/off buttons. To improve the usability of the table, we highlight each line when hovering it.

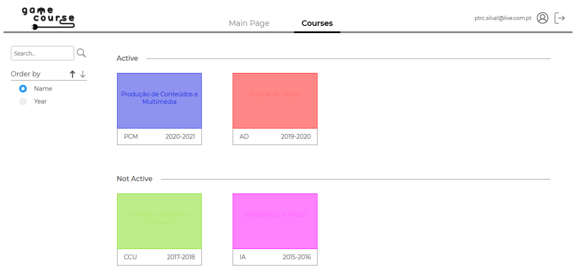


Figure 3: Courses page for non-admin users.

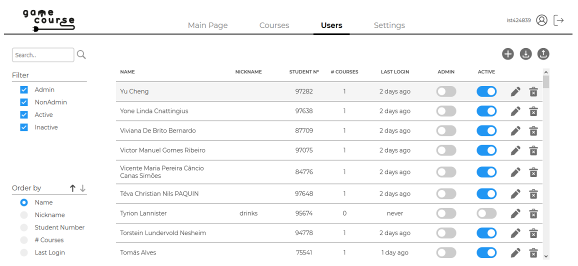
For the non-admin users, we only display the visible courses where the user is registered. However, instead of using a table, we used cards and separated the active and not active courses for quick access for currently lectured courses. The short name and the year of the course are placed on the card label, and the name of the course appears on hover on the colorful top part of the card (Figure 3).

5.3. Search, filter and order algorithm

Finding a particular item among a big list of items can be difficult and time-consuming. To make it easier, we created a sidebar to search, filter, and order by a particular attribute on the Courses and Users pages (Figure 4). For this mechanism to work, we started by duplicating the array that keeps the items, the original is used to display, and it is where we apply the filter and search. The second is used to restore the original array before any new filter or search. To keep the consistency of data displayed, every time one of them is triggered, both will be run, first search then filter.

The search will be triggered each time the user types on the search box. After validating the input, it will search for items where at least one of the attributes contains the input. Whereas on the filter, the trigger occurs once any of the checkboxes are clicked. After checking if the combination of options is possible, it checks if any item matches the selected conditions. The results of each step are placed on a temporary array and then copied into the original one. If no match is found, an alert message is presented on the screen where the items would be.

On order by, as only one option can be selected at a time, the system keeps track of the last selected attribute and direction of order. Once something changes, it starts by checking what caused the change. If it was the direction, it will simply reverse both our arrays. If it was the attribute, it starts by applying a sort on both arrays. This sort is done on ascending order, arrow down, so then it checks the selected arrow, and if it is up, descending order, it reverses the arrays.



NAME	NICKNAME	STUDENT Nº	# COURSES	LAST LOGIN	ADMIN	ACTIVE
Yu Cheng		97282	1	2 days ago	○	●
Yone Linda Cnattingius		97638	1	2 days ago	○	●
Viviana De Brito Bernardo		87709	1	2 days ago	○	●
Victor Manuel Gomes Ribeiro		97075	1	2 days ago	○	●
Vicente Maria Pereira Câncio Canas Simões		84776	1	2 days ago	○	●
Téva Christian Nils PAQUIN		97648	1	2 days ago	○	●
Tyrion Lannister	drinks	95674	0	never	○	○
Torstein Lundervold Nesheim		94778	1	2 days ago	○	●
Tomás Alves		75541	1	1 day ago	○	●

Figure 4: Users page on the system domain.

5.4. Add existing user to a course

On the Users page inside a course, when implementing the add modal with all the necessary inputs, we realised there was a very important action missing: add existing users of the system. So we quickly created two more versions of this modal:

- **Select state**, to chose between creating new or add existing user;
- **Add existing state**, to display all the users that are not in the course, allowing the user to select as many as he/she wants and give a specific role to all of them.

The "select" state is composed of two buttons to chose the add method and the "add existing" of a search box, a listing box, and a dropdown (Figure 5). The search box has two purposes: filtering the list below and showing the already selected users, allowing a quick verification and deselection if needed. The tags are clickable and remove the correspondent item from the selected list. The dropdown shows the roles that can be attributed to the selected users, and only by selecting one of them will the save button be available.

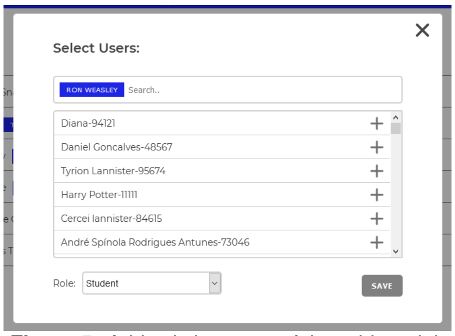


Figure 5: Add existing state of the add modal.

5.5. Undo/Redo functionality on the Roles page

On the roles page, we incorporated the previous separated actions to define each role's landing pages. With an extra task and much responsibility in one page, as roles might affect user's authorizations and landing pages the usage of the course, we felt the need to create an undo/redo functionality. We started by creating the State Manager (SM), which keeps three arrays: one for past states, one for current state, and the third for future states, and created the functions needed for the undo/redo functionality:

- **New State:** the SM will move the current state to the end of the past states, replace the current state for the submitted one and clean the future states.
- **Undo:** the SM will move the current state to the end of the future states and the last state of the past states into the current state.

- **Redo:** the SM will move the current state to the end of the past states and the last state of the future states into the current state.

Every time the roles hierarchy was altered, a new role was added or deleted, and a landing page redefined, we created a new state with all the page information. Only after the first action does the SM allows to undo, and only after a previous undo does it allow redo. After either actions, a state is received from the SM, and the page is re-rendered to match the now current information.

5.6. New Configuration System for Modules

Before this project, the only way to configure a module was through the use of external scripts that would collect external data and populate the GameCourse database. To reduce the amount of effort and time spent on this task, we created a configuration system available to all current and future modules of the system. This system generates a page based on blocks and not only saves time to the Teacher or Admin that is going to use it to configure their course, but also to the developers of modules who don't need to create a whole new page for their module configuration. The generated configuration page has three possible sections:

- **General inputs**: to display all the variables that define and change the behavior of the module.
- **Listing items**: for modules like Badges, that have inner objects that can be created, edited and deleted. This section works and looks very much like the other listing pages, with a table and options to manipulate the items.
- **Personalized section**: sometimes neither of the previous sections are enough or a variable is too complex to a single input field, so we added a personalized section where the developer can create what he/she wants.

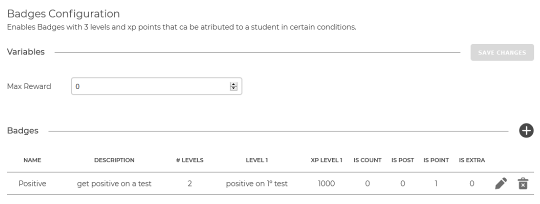
To build each section of the configuration page we have a "has", "get" and "save" function that will tell if the correspondent section is going to be needed, what information is needed and to save information coming from the user inputs. The "has" functions work as verification, only returning true or false values, and the other ones have specific rules in what they must return ("get" function) and in what they receive. If the developer intends to use any of the sections he must redefine the correspondent functions on the module's class.

On the General inputs section, the get function must return a list with all the inputs needed, including: their name to be displayed as label; the id to later be able to collect the information; the type of the input; the options in case the type is "select"; and the current_val to set an initial value on the input. The system will then create each input together with a save button, that will call the save function where we can access the user's input by using the ids defined earlier. The developer can next use this value as needed.

On the Listing items section, the get function must return an array containing: listName, to give a name to the section; itemName, to be used as the header of the modals; header, to select the labels of the columns of the table; displayAtributes with the item's attributes to display on the table on the same order as the header; the list of items; and allAtributes, to define the inputs on the add/edit modal (follows the same syntax as the inputs of the previous section). The save function will be called once an add, edit or delete is requested on the page, where the developer has access to the type of action and the content associated, so he/she can save the information.

On the Personalized section, the developer is free to create his/her own code and generate a page as he/she desires using all the AngularJS 1 advantages. He/she needs to create a Javascript function for that, save it on the module's folder, and return the function's name on the personalized section get function. Here the is no save function as the developer is in charge of creating that part.

To prove and demonstrate this system's value, we defined the Badges configuration page (Figure 6), using the first two sections to define the max reward value and the list of badges available on the course.


Badges Configuration

Enables Badges with 3 levels and xp points that ca be atributed to a student in certain conditions.

Variables

Max Reward: 0

Badges

NAME	DESCRIPTION	# LEVELS	LEVEL 1	XP LEVEL 1	IS COUNT	IS POST	IS POINT	IS EXTRA
Positive	get positive on a test	2	positive on 1º test	1000	0	0	1	0

Figure 6: Badges configuration page.

5.7. View editor changes on the toolbar

Inside the View Editor, we started by removing the sidebar and recreating the breadcrumb so the user can click on it to go back to the listing page. Then we changed the behavior of the toolbar of each element. Instead of using the hover, with *mousein* and *mouseout* events to show the correspondent toolbar, we changed it to be a click event with selected and not selected sates. This allowed us to move the toolbar to the bottom of the page (Figure 7) since we can access it outside the visual space of the selected element and make its options bigger, which was not possible before because it could cover other elements.

Previously, there were two options on the toolbar for saving purposes, one to do it keeping the reference to the original piece of view and other copying by value. We decided to incorporate both actions on one single option and distinguished them on the modal by using an on/off button, that would then determine the functionality chosen.

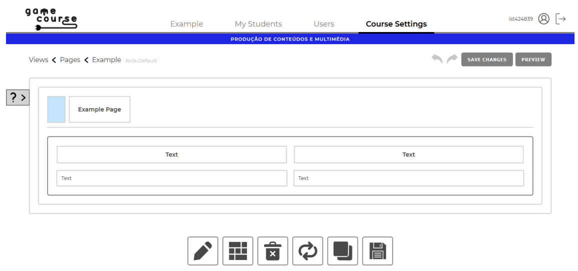


Figure 7: View Editor while selecting a block.

6. Results & discussion

To validate this project's success and the impact of the new GameCouse, we did User Tests, which included the performance of tasks on the old version and the new version of GameCourse.

6.1. User Tests

We performed summative user tests, where potential users were asked to perform the same 12 tasks on both systems (plus five more on the new version). To evaluate these tasks regarding effectiveness and efficiency, we collected the time a user needed to perform a task and if he/she succeeded. After performing the tasks on each version, the users answered a questionnaire with questions from the NASA Task Load Index (NASA TLX)[3], the System Usability Scale (SUS)[4], and the User Experience Questionnaire (UEQ)[5].

Theses tests were performed through the software Zoom[6], that allowed us to create a video call for the experiment, share our screen, and receive interaction from the other side of the call. The new version of GameCourse was installed on a Virtual machine[7], and the old version was installed on the computer from where the interviewer was making the video call.

We conducted these experiments with 20 users, two women and 18 men. 11 of them already had used the Gamecourse system on the MCP course but had no contact with the back-office part, only visible for admin users. All users reported previous experience with other LMSs, except for one, and were connected to the Computer Science field.

Each experiment started with the project's presentation, followed by a questionnaire to obtain information about the user. The users were then asked to continue the experiment either to the new or the old version of the system (the users with odd id used first the old system then the new, and even ids the opposite). In each system, the users were given 5 minutes to navigate it and ask questions. After this time or the user approval to begin the test, we gave the user a list of tasks to perform, in random order, so there is no direct influence of the learning curve of the system:

1. Create a new course[8].
2. Delete a course.
3. Make a user admin of the system.
4. On a course show the list of users that have the role Teacher.
5. On a course change the role of a user.
6. On a course add a user[8].
7. On a course create a new role.
8. On a course activate a module.
9. On a course create a new Page.
10. On a course edit a page on the default Specializations and add new parts.
11. On a course edit a page on the default Specializations and change the content.
12. Access the documentation and search for a specific function.
13. Edit a user[9].
14. Check the information about the user logged in[9].
15. Verify if a module is installed on the system[9].
16. Logout of the system and try to login with a Likedin account[9].
17. On a course access the configuration page of a module and add a new item[9].

6.2. Results

Regarding the tasks performance, we started by calculating the Success rate of each task. From the results, we learned that the success rate on the new version of the system is 100% in almost all tasks and higher than the old version, except for task 10, where the old version got 10% more success.

Then we calculated the mean and standard deviation values of each task on both systems (Table 2 and 3). By looking at the mean value, we can see that the time spent on each task is lower on the new version for tasks 2, 5, 6, 8, and 12, but on the remaining tasks, the values are close. To have an exact result that tells us if the change of the GUI had an impact on the time spent to complete the tasks, we tested the null hypothesis (H0) of "both versions have the same impact on the performance of each task". First, we did a Shapiro-Wilk Normality Test to verify if the data of each task follows a normal distribution on both versions. In Table 2, we have the

[3]https://humansystems.arc.nasa.gov/groups/TLX/
[4]https://www.usabilitest.com/system-usability-scale
[5]https://www.ueq-online.org/
[6]https://zoom.us/
[7]https://pcm.rnl.tecnico.ulisboa.pt/gamecourse/

[8]This task is slightly different on the old system
[9]This task was only performed on the new version

results for the tasks which had normally distributed data, and in Table 3, the ones which did not. Then for the tasks on the first table, we did a Student's T-test and on the second table a Wilcoxon Signed-Rank Test.

T	V	Mean	Std.	T-test	H0	FV
1	old new	00:23 00:44	00:06 00:11	0,000	R	Old
2	old new	01:15 00:09	01:00 00:03	0,000	R	New
3	old new	00:17 00:12	00:10 00:06	0,055	NR	-
4	old new	00:17 00:19	00:10 00:20	0,602	NR	-
5	old new	01:03 00:26	00:47 00:07	0,005	R	New
7	old new	00:44 00:43	00:25 00:31	0,910	NR	-
9	old new	00:24 00:24	00:12 00:08	0,847	NR	-
10	old new	02:48 02:51	01:10 01:04	0,876	NR	-

Table 2: Results for the tasks 1 to 5, 7, 9 and 10. (R - Rejected; NR - Not Rejected; T - Task; V - Version; Std - Standard deviation; FV - Fastest version)

T	V	Mean	Std.	W SR	H0	FV
6	old new	01:46 00:31	00:32 00:11	0,000	R	New
8	old new	00:25 00:20	00:09 00:05	0,001	R	New
10	old new	02:34 01:16	01:07 00:49	0,004	R	New

Table 3: Results for the tasks 6, 8 and 12. (R - Rejected; V - Version; T - Task; Std - Standard deviation; FV - Fastest version; W SR - Wilcoxon Signed-Rank)

In order to reject H0, both tests needed to return a significance $p < 0.05$. For tasks 2, 5, 6, 8, and 12, we were able to reject it and confidently say that the new version of the system allows a quicker performance on these tasks. On task 1, we were also able to reject the null hypothesis, however the mean value indicates the quicker system is the old one. This happens because the new version requires more input fields than the old one.

Even though the View editor's look was changed, on task 11 the user needed to use the Expression Language, which was not altered. Thus, the results of the second version tested suffered a direct impact, especially if the user was able to complete the task on the first encounter. In table 4, we can see this exact influence as the user takes a lot less time performing the task the second time he/she encounters it, and the success rate increases significantly. To test the null hypotheses of "both versions have the same impact on the performance of task 11" we considered use the first encounter of each version. However, this analysis was impossible to make, due to the low success rate of this task. We then performed a Shapiro-Wilk Normality Test to verify if the data follows a normal distribution on both versions. As it does we applied a T-test, from which we got a $p = 0.318 > 0.05$. not rejecting our null hypothesis.

1st V	V	SR	Mean	Std.
old	old new	30% 80%	03:40 02:07	00:59 00:22
new	new old	60% 80%	03:35 01:17	01:17 00:43
-	old	55%	02:12	01:19
-	new	70%	03:21	01:19

Table 4: Results of task 11 separated by the first tested version. (1st V - First version tested; V - Version; SR - Success rate)

For the last tasks, 13 to 17, we started by calculating each task's success rate. All of them reported 100% success, except for 14 with 95%, where one of the users did not understand what he/she was asked to do. Then we calculated the mean, standard deviation, and the 95% confidence interval values. Table 5 demonstrates that for task 13, the users took less than 40 seconds to complete it, for tasks 14 to 16 less than 20 second, and for task 17 less than 2 minutes, as it requires filling in data.

T	Mean	Std.	95% Confidence Int.
13	00:29	00:10	[00:24-00:35]
14	00:08	00:09	[00:03-00:12]
15	00:09	00:03	[00:08-00:11]
16	00:13	00:03	[00:12-00:15]
17	01:38	00:33	[01:29-01:56]

Table 5: Results of tasks 13 to 17 on the new version (T - Task).

6.2.1 NasaTLX

For the NasaTLX we used a variant called Raw TLX which includes only the sum of the results of the questions. We started by performing a Shapiro-Wilk Normality Test to verify if the data follows a normal distribution on both versions. The results show that there was indeed a normal distribution, which allow us to move to the Student's t-test. From this test we got a $p = 0.01 < 0.05$ rejecting the null hypothesis of "both systems having the same impact on the workload of the student" (Table 6). Therefore, we can conclude that the new version of the GameCourse has less impact on the workload of the user.

Version	Mean	Std.	Student's t-test
old new	49,800 29,800	26,824 19,352	0,010

Table 6: Results of the NasaTLX questionnaire.

6.2.2 System Usability Scale

From the data collected on the System Usability Scale (SUS) questionnaire we started by calculating the SUS score:

- For each of the odd numbered questions, subtracted 1 from the score.
- For each of the even numbered questions, subtracted their value from 5.
- Added the previous results for the total score. Then multiplied it by 2.5.

We can immediately see by the final results (Table 7) for both versions that the new one has a much higher score with more the double the points. To prove the impact we first did a Shapiro-Wilk Normality Test, which indicated that the data did not follow a normal distribution on both versions. Then we performed a Wilcoxon Signed-Rank Test from which we got a $p = 0.00 < 0.05$. This rejected the null hypothesis of "both systems having the same level of usability" (Table 7) and proved that the new system has a greater level of Usability.

Version	Mean	Std.	Wilcoxon S-R
old	35,8	15,046	0,000
new	77,1	18,051	

Table 7: Results of the SUS questionnaire.

6.2.3 User Experience Questionnaire

To analyse the results from the User Experience Questionnaire we started by transforming the data so the values were between -3 and 3 instead of 1 and 7. Then calculated the mean ans standard deviation for each scale (Attractiveness, Perspicuity, Efficiency, Dependability, Stimulation and Novelty). From these values we can see that there is a clear distinction on the results of both versions (Table 8). We have the old version's mean on the negative side and the new version one on the positive side. To reveal the true impact of the change regarding UX, we did a Shapiro-Wilk Normality Test that showed that not all the scales follow a normal distribution. Therefore we performed a Wilcoxon Signed-Rank Test that rejected the null hypothesis of "both systems provide the same level of User Experience" (Table 8) with all scales having a $p = 0.00 < 0.05$. This analysis proves that the new version has a better UX.

6.3. Discussion

The user tests showed us that the tasks can be done quickly and without much effort. In addiction: finding some information on the documentation takes half of the time it used to take; deleting a course takes one minute less to be completed; and that changing user's roles also takes half of the time it used to take.

Even though the visual change did not affect the time spent to complete some of the tasks, it improved the way a user sees and interacts with the system. All three questionnaires had a significantly better score on the new version of GameCourse, with almost half of the previous workload, twice the usability, and permanent positive results on UX.

Scale	V	Mean	Std.	Wilcoxon
Attractiveness	old	-1,38	0,96	0,000
	new	1,83	0,86	
Perspicuity	old	-1,24	1,08	0,000
	new	1,54	1,05	
Efficiency	old	-0,21	1,13	0,000
	new	2,05	0,59	
Dependability	old	-0,09	1,11	0,000
	new	1,93	0,65	
Stimulation	old	-0,70	1,20	0,000
	new	1,61	0,78	
Novelty	old	-1,25	1,17	0,000
	new	1,46	1,04	

Table 8: Analysis per system of the results of the UEQ questionnaire.

7. Conclusion

For this thesis, we designed and implemented a new GUI for the GameCourse system that is clear, simple, minimalistic, and respects UI & UX principles. Besides restructuring the system so all the concepts are encapsulated on their correspondent pages, this thesis added new functionalities including: modules configuration pages; user configuration on the system level; search, filter and order capabilities on the listing pages; and undo/redo behavior on the roles management. These changes guaranty that all the use cases established for the system are available whilst improving the system's usability, making each task more accessible to the user.

The GameCourse now provides two versions of the Courses page: one where an admin user can find the courses registered on the system, manage their information, and access a course; other where the non-admin user can see and access all the courses in which he/she is enrolled. It also provides a new Users page on the system domain where all the user's information can be edited. This page on the course domain allows adding existing users and manage all course user's roles. There is also a search, filter, and order mechanism that allows quick access to a particular object, on these pages.

We have also implemented a new state manager that allows the user to undo and redo actions made while managing the list of roles, their hierarchy, and correspondent landing pages. The state manager was implemented independently, meaning that it can also be used on future pages.

On the View editor, the toolbar of each item will now appear fixed at the bottom of the page with bigger icons, after the correspondent part is selected by click. The pages triggered by each option on the toolbar were replaced by modals, that are used throughout the system to receive user's input. To take advantage of the visual aspect of this GameCourse's section and the user's ability to associate icons with tasks, some of the options on the modals were created with a visual selection instead of dropdown menus. To help understand this section and the expression language, the GameCourse now provides an always visible slider with shortcuts to the documentation pages. These pages were also changed to allow easy access to information.

We created a modular system that generates a configuration page for every module that requires it. This allowed us to remove the external scripts that fetch data for the module's configuration. This system allows the creation of configuration pages for both existing and future modules, which means that developers working on future modules only have to define the requirements in the module class instead of learning how to create a page on the system. This feature was essential since it guarantees consistency throughout the system whilst decreasing the developing effort.

The work of this thesis improved the system's usability, which now scores a 77 out of 100 compared with a previous 36. It also reduced the system's workload by almost half, leaving the users more satisfied and less stressed with the experience. Finally it brought all the user experience scale points to the positive side with mean values around 1 in a scale of -3 to 3.

This system and its GUI have not yet reached its final state as some improvements can still be made. As proved by the final User tests, the View editor is still very painful to use and time-consuming. The whole system needs additional code for a mobile version. Even though we had into consideration the resize of the window, making it flexible and dynamic, it is not enough to cope with very small screens.

With the current work, it is finally possible to implement the courses' themes to add a particular visual and take the gamification experience to a new level.

References

[1] An Argument for Clarity: What are Learning Management Systems, What are They Not, and What Should They Become? *TechTrends*, 51(2):28–34, mar 2007.

[2] A. M. Baltazar. Smartboards. Master's thesis, Instituto Superior Técnico, Lisbon, 2016.

[3] A. H. Dourado. Gamecoursenext. Master's thesis, Instituto Superior Técnico, Lisbon, 2019.

[4] K. Erenli. The impact of gamification-recommending education scenarios. *International Journal of Emerging Technologies in Learning (iJET)*, 8(2013):15–21, 2013.

[5] F. Grivokostopoulou, K. Kovas, and I. Perikos. Examining the impact of a gamified entrepreneurship education framework in higher education. *Sustainability*, 11(20):5623, 2019.

[6] S. Y. Hock, R. Omar, and M. Mahmud. Comparing the usability and users acceptance of open sources learning management system (lms). *International Journal of Scientific and Research Publications*, 5(4):1–5, 2015.

[7] M. Machado and E. Tao. Blackboard vs. moodle: Comparing user experience of learning management systems. In *2007 37th annual frontiers in education conference-global engineering: Knowledge without borders, opportunities without passports*, pages S4J–7. IEEE, 2007.

[8] L. Martin, D. R. Martínez, O. Revilla, M. J. Aguilar, O. C. Santos, and J. G. Boticario. Usability in e-learning platforms: heuristics comparison between moodle, sakai and dotlrn. In *Sixth International Conference on Community based environments. Guatemala*, pages 12–16. Citeseer, 2008.

[9] D. S. S. Sahid, P. I. Santosa, R. Ferdiana, and E. N. Lukito. Evaluation and measurement of learning management system based on user experience. In *2016 6th International Annual Engineering Seminar (InAES)*, pages 72–77. IEEE, 2016.