GameCourseUI

Mariana Saraiva
Instituto Superior Técnico
Lisboa, Portugal
mariana.saraiva@tecnico.ulisboa.pt

ABSTRACT

Gamification is defined as the use of game design elements in non-game contexts and, recently, has been introduced in many fields. The User Interface (UI) of the system has a huge impact on how it is accepted because, even if the system fills all the requirements, its ease of use and visual appeal can lead to rejection by users. GameCourse is the system used to gamify the Multimedia Content Production MCP course at Instituto Superior Técnico (IST), whose UI has been improved since the previous year. The purpose of this thesis is, besides improving some UI features that are already implemented, add features and mechanisms to allow customization of some views for students with different profiles, as well as ease the configuration and management of the courses. We accomplished our goals by easing the personification of the view as well as improve the User Experience (UX) in the module configuration pages.

Author Keywords
Gamification; Gamification in Education; User Interface; User Experience.

ACM Classification Keywords
HCI

INTRODUCTION

For the past few years, the learning process has benefited from the introduction of several activities which turn it more fun and engaging. Mainly in the education field, different learning processes have been employed by providing students with a more motivating system, in which they feel engaged, without losing their interest in the subject. With monotonous classes, students may lose motivation, and their interest ending up failing. The innovation of the learning process by taking different approaches could be helpful to make it more attractive.

These new learning processes include introducing new elements, mainly game elements, that keep students motivated. This is called gamification, defined as "the use of game design elements in non- game contexts" [1]. The impact of the User Interface (UI) on students can change the way they accept gamification. The visual appeal of the interface and its ease of use, which it influences, are determinants for users to continue using it and, thus, keep them engaged within the application [2]. Therefore, in the education context, gamification can be better received by students if the interface has a good visual appeal and is easy to use.

In the Master's degree in Computer Science and Engineering at Instituto Superior Técnico (IST), there is a successfully gamified course - Multimedia Content Production (MCP), where elements such as a leaderboard, badges, and points are used to create an engaging environment, resorting to the GameCourse system. Currently, this system needs improvements to its UI, and also in the customization of the interface displayed to each student, in order to give the best experience to everyone.

The goal of this thesis is to improve the current GameCourse interface and, particularly, create mechanisms to build different views for students with different profiles and ease the configura- tion and management of the courses.

RELATED WORK

Gamification

Defined as "the use of game design elements in non-game contexts" [1], gamifica- tion has become extremely popular in recent years. Its benefits in increasing intrinsic motivation [3] and the ability to add an entertaining dimension to monotonous tasks has led to its usage in a wide range of settings.

The concept of flow, introduced by Mihaly Csikszentmihalyi, is highly related to competence since flow's properties can lead to a feeling of competence [4]. Flow describes a mental state reached when a person is entirely immersed in an activity [5]. This is experienced when someone's perceived skills are balanced with the perceived challenges. When these two things are balanced, you are in the Flow Zone. Different people have different flow zones for each task/game. If you are overqualified for an activity, you will end up in a boredom state because there are no challenges and the activity is too easy for you. On the other hand, if the challenge is much bigger than your perceived skills, you lose the state of flow and you go to an anxiety state. A well-designed game maintains players in their Flow Zones, and that is why those games are successful - the players enjoy and have fun while playing.

Gamification in Education

One of the main fields in which Gamification is applied is education. The purpose of applying gamification to the

learning process is to make it more engaging for students and, consequently, maximize their success.

To study how gamification affects students' engagement, Garden and Rivera [6] presented a framework based on three theories - Landers' Theory of Gamified Learning [7], Kahu's Framework for Student Engagement [8], and Bedwell's Taxonomy of Game Attributes [9]. The elements from the different theories that are linked to each other are mapped into a single state. The states are connected according to their cause-effect relation. For example, the antecedents of Engagement (e.g. assessment) state precedes the Engagement state (e.g. behaviours/attitudes). In turn, the latter precedes the Consequences of Engagement state (e.g. learning outcome). Kahu proposes that this whole process (Engagement) is circular. Thus, he believes that increased engagement leads to increased learning outcomes (e.g. academic performance), which can directly affect the antecedents of engagement, for instance, the motivation.

User Interface

Another important factor to drive motivation and engagement in a learning app is its visual appeal and ease of use [2]. In general, perceived visual appeal significantly influences the perceived ease of use. And perceived visual appeal can affect trust stronger than ease of use. Website design features have a different effect on website trust evaluation for different genders. Men consider both visual appeal and ease of use to evaluate a website trust while women give more attention to the website visual design and less to its ease of use.

As we have seen so far, the use of gamification is increasing fast in this later decade. However, some features should be improved, for instance, the customization of the gamification. Monterrat et al. presented a model to adapt the gamification features taking into account the player profile of the learners, providing them with game features suitable to their personality [10]. The main goal was to present different interfaces, with different game elements, depending on their profile.

Multimedia Content Production

MCP is an MSc course in Information Systems and Computer Engineering at IST. MCP has been a gamified course for almost 10 years, being a target for studying student engagement and behaviour in a gamified system [11] [12].

Currently, MCP assessment is divided into in-class assessment, such as weekly questionnaires and lab assignments, and online, through the Moodle platform, with online assignments, of which not all are mandatory, that is, the students have the freedom to choose what they want to do to achieve their goals. Online assignments are graded along with helpful and useful feedback. All the assessment objects are graded with Experience Points (XP) and throughout the semester students are gathering XP that, in the end, will be translated to their final grade (corresponding to their level in the leaderboard).

Gamification has been employed by adding game elements such as a **leaderboard**, where students can see their performance relative to the others, their total **points** and corresponding **level** as well as other students' points and level; **badges**, for which students need to complete accomplish- ments, for instance, attending to theoretical classes, participate in Moodle by providing extra material about each class and show creativity and quality in their online assignments; and a **skill tree**, which is a set of assignments, distributed in four levels, and possible to complete at any time of the semester. To achieve level 2, you have to complete the required assignments from level 1 and so on. This allows students to have numerous possible paths to gather the maximum XP of the skill tree. Both badges and skill tree are elements than can be checked in the profile page of each student as well as their performance and progress through the semester.

THE GAMECOURSE UI

The main technology used to develop the UI of the system is AngularJS1. It takes care of the interaction between pages through states, controllers, and scopes. On states, the routing of the GameCourse system is defined, that is, we define which pages are available to navigate and through which URLs. On the controllers, the pages' HTML structure is generated and the needed requests to the API to obtain the desired data are made. This data is stored in the scope, managed using scope functions, and associated with GUI elements, which enabled automatic re-renders when the associated information changes.

The front-end part of the project includes, essentially:

- Controllers, which are JavaScript files that build the pages with the information gathered from the Application Program Interface (API). The API functions are defined in PHP files.
- Other JavaScript files that create dynamic pages.
- Images, Cascading Style Sheets (CSS) files, and other user handlers.

On the Courses and (system) Users pages we can add new ones, import more data such as new courses or new users, export a file with that information, filter and change the order in which it is presented (Figure 1).

Once inside a course, the navigation bar presents the pages of the course (in this case, the AwardList), the Users tab, in which we can see all the users (students and professors) of that course, and the tabof the Settings of the course, depicted in Figure 2. This Users page has the same features as thesystem users, described above. However, in this page, we can see the role each user has in that course, for example, whether it is a teacher or a student, for example.

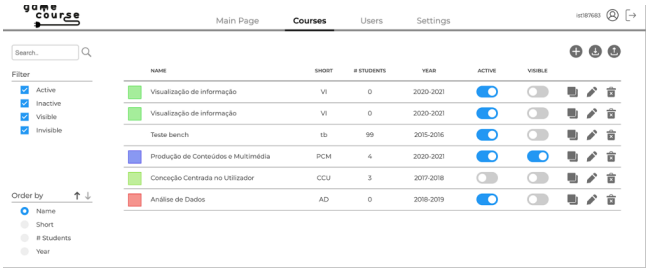


Figure 1. Courses page

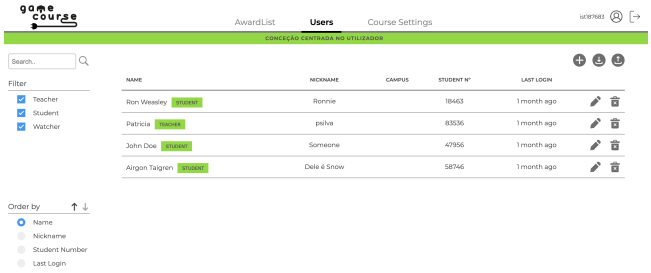


Figure 2. Course Users page

To better understand how the GameCourse system works, regarding the UI, we first need to be aware of the concepts of **views** and **pages**. Views are built with the help of a view editor (Figure 3), which has different layout elements available – text, image, table, block, chart and template, where the charts are available only if the modules charts is enabled – , and the system's Expression Language (EL), which allows us to present the viewswith dynamic content. A **view** can be a single element, such as an image or text. By compounding these small views, we can create more complex views. For example, we can create a list of images by repeating one single view previously built. When we have a view, we can choose to show it as a **page** with a certain name. Then, this page can be available on the navigation bar whose content is that view.

Figure 3. View editor

The system automatically selects the more suitable aspect for each user when he/she enters. There are two types of aspects for views:

- **Role - Single**: a view that can show a different aspect according to the viewer's role.
- **Role - Interaction**: a view that can show a different aspect according to the roles of the viewer and the user associated with the page. For example, a profile page will have as user the person whose information is displayed and the viewer is the person who is seeing the page.

Inside the view editor, when we select a part of the layout, some buttons appear at the bottom of the page. By clicking on these buttons, we can edit fields, change the layout, delete items, switch a part of it for another possible, duplicate a part and save the layout as a template.

DEVELOPMENT

First, we needed to have a stable version of GameCourse from which we could start our implementation of the new features or the necessary improvements. Thus, we started by fixing the existing small bugs. Besides, as this was done while I was writing my thesis project and my requirements were not clearly defined, I also did some improvements (required/suggested by my previous colleagues), for example, the improvement of the import feature.

File Explorer

To allow the user to access the files in the server, or to upload new ones, we created a modal from scratch in which the user can either upload a file from their computer or browse files saved on the server side (Figures 4 and 5).

On the upload tab, when the user uploads a file, it will be saved on server side, and it will appear below. By clicking it, it will be selected and then the user can delete it (clicks on the delete button on bottom left) or select it to include it on the page (clicks the select button on bottom right). On the other hand, if the user wants to choose another one, he/she must upload another file of their choice.

On the browser tab, the user can only browse files from the current course. In this folder, the user can navigate through every folder in the respective course.

This file explorer can be used in every page of the GameCourse. However, in some cases, not every file extension can be chosen. Thus, the developer is accountable for allowing which type of files are acceptable in each case by showing only the permitted ones, by provide a list of allowed extensions to the function that will instantiate the file explorer.

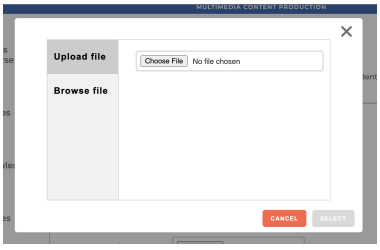


Figure 4. Upload file tab.

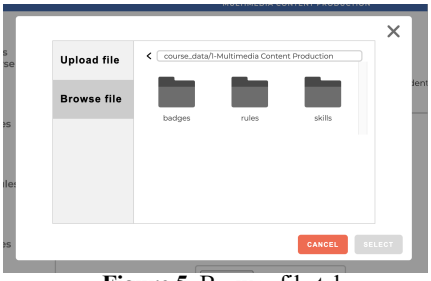


Figure 5. Browse file tab.

Modules Configuration

The modules' configuration had a poor UI, and some configurations were simple text boxes where the user must copy and paste a text file to them, typically in csv format. This part of GameCourse truly needed improvements in its UI, and also UX. Besides, each module had its own controller, and some- times there was duplicated code. Thus, we decided to use one single Controller (Configuration Controller) for most of the configurable modules which had a similar configuration. For example, XP and Levels, Badges and Skills all have a list of general items - levels, badges, and skills, respectively.

XP and Levels

This module configuration page was composed of a text box on the right where we pasted the content of a text file with a button below to confirm the data and another one to clear the box. On the left, we could check the values introduced, as we can see in Figure 6.

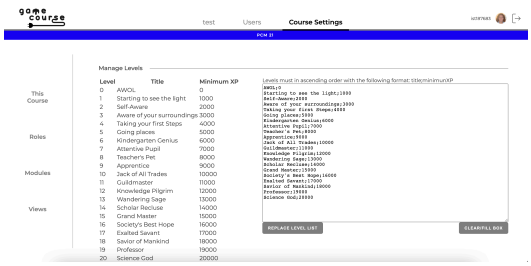


Figure 6. Previous Levels configuration page.

In order to keep the system consistent, and give a better look, as well as UX and usability, we changed this page (Figure 7). We used the generic Configuration Controller, which has three possible sections: General Inputs, Listing Items and Personalized section. For this module, we only needed one of the them: listing items. In the Levels module, we defined in the function *get_listing_items* the name of the list, in this case, Levels, the name of the columns (Level, Title, and Minimum XP), the items to show, and the corresponding attributes of each item to each column. The items were given through a database query. Then, we defined five new features, two of them are general features that affect all the items – Import Levels and Export Levels - , and the other three are individual features – Add Level, Edit Level and Delete Level.

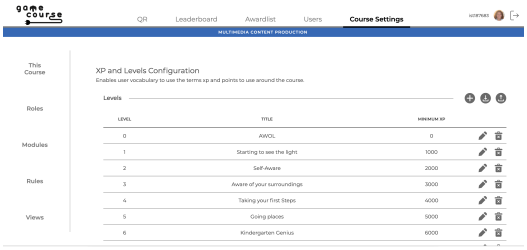


Figure 7. New Levels configuration page.

Skills

This module configuration page was similar to the levels page. It was composed of three text boxes on the right - one for the skills, one for the tiers, and one for setting up the maximum skill tree reward. In the first two, we could paste the content of a text file, and for each, we had a button below to confirm the data and another one to clear the box. In the bottom box, we wrote the number and then saved it by clicking the save button. On the left, after saving all the configurations, we saw the graphic visualization of the configured skill tree (Figure 8).

Once again, in order to keep the system consistent, and give a better look, as well as User Experience and usability, we changed this page (Figure 9).

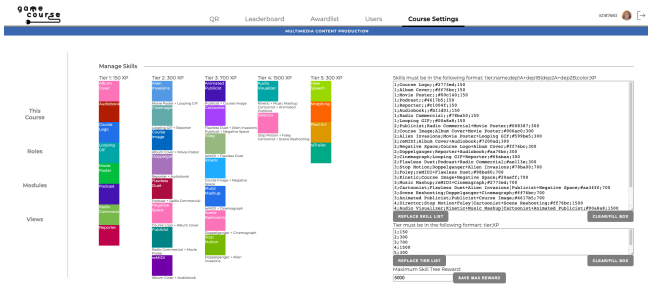
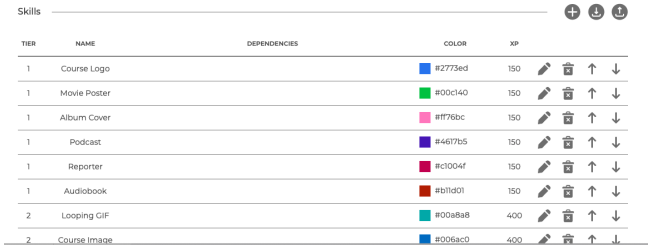


Figure 8. Previous Skills configuration page.

By using the generic Configuration Controller, we used the three possible sections: generic items - for the maximum reward -, listing items - for the skills, and the personalized section - for the tiers. In the Skills module, for the general item, we defined the function *get_general_items*, which, in this case, will return the information only regardingthe maximum XP. For the skills, we defined in the function *get_listing_items* the name of the list, in thiscase, Skills, the name of the columns (Tier, Name, Dependencies, and Color), the items to show, andthe corresponding attributes of each item to each column. The items were given through a database query. For the tiers, we defined a personalized function - *get_tiers_items* - that returns the information forthe Tiers table, similar to the skills' but the name of the columns are Tier and XP. For these two lists weadded a

new feature – reordering. By adding two columns to the lists, the user can manage and reorder the skills and the tiers as he/she wants (moving up and moving down). These changes are propagated to both tables, in other words, if the



user reorders the tiers table, the skills table will reorder automatically to reflect those changes.

Figure 9. New Skills configuration page.

Regarding preventing unwanted moves, our mechanism prevents moving upthe first tier or moving up the first skills of a tier, as well for moving down the last tier or the last skill of atier. The maximum reward box is now on the general items section (on top of the page). Then, we defined new features to deal with the management of these items in this new configuration: add, edit and delete options for both skills and tiers, as well as import and export skills.

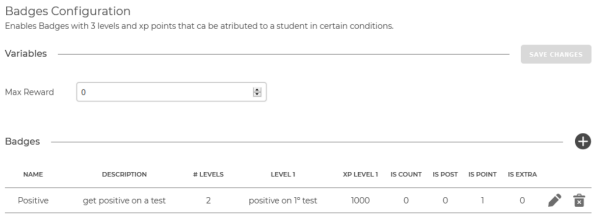
Moreover, each skill has its description which is an HTML page, and to write the description, we used a framework – Quilljs - to help us building the text editor. We have look for alternatives, such as the RichTextEditor, but we ended up choosing Quilljs because it is cleaner, with a customizable toolbar, where we can choose which settings we want to include in it. The alternative had a default toolbar with too many elements that would end up not being useful to our system and even confuse the users. With Quilljs, the user can build a page with text and images, links, with many features such as using different fonts, bullet points and format the text. Then, this page will be saved as an HTML file in the server side, in the course folder. For the image picking, we used the picking file modal mentioned before.

Badges

This module already had its configuration page using the Configuration Controller. However, this page only allowed the user to add one badge at the time and set the maximum reward value (Figure 10).

Consequently, the first thing we added was the import and export features, to allow users to import a file with all the badges and their information, such as name, XP, description, among others. We have kept an add button if the user wants to add only one badge (Figure 11).

By importing a text file, it is not possible to define the image(s) for each badge. Previously, these images were in the system, and they were global. To give more flexibility,

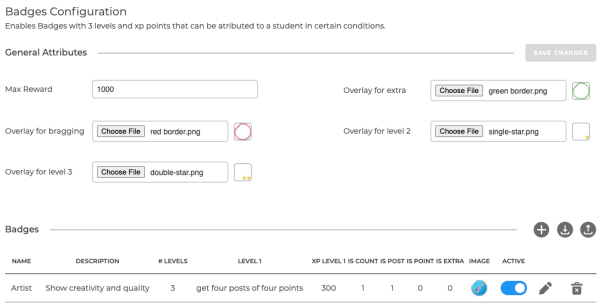


we moved the badges folder to the courses folders, since courses may have different badges.

Badges can have up to 3 levels, worth extra points and be bragging and worth nothing. All these features can be

Figure 10. Previous Badges configuration page.

represented with a layer in the badge image. These layers were the same for all badges. Once again, to give GameCourse more flexibility, these layers can now be chosen in the badge configuration page. Alongside max reward input, we added 4 new fields to pick up an image (using our file explorer), one for each – extra overlay, bragging overlay, level 2 overlay, and level 3 overlay. These overlays will then be used for every badge of the course. To permit a user to pick up an image for each badge, we added a field in the new and edit badge modals to choose an image, which use our file explorer to upload or browse an image. After the user picks an image, we generate the other images of the badges according to its specifications (number of levels, is extra, is bragging). To do so, we used a library that merge images. For each badge,



if it is extra, then we merge the extra overlay for every level, and the same if it is bragging. Besides, for levels 2 and 3, if exist, we also merge the respective overlay.

Figure 11. New Badges configuration page.

View Templates and Pages

In the first place, there was a huge misunderstanding regarding the difference between a page anda template and the code reflected it. So, we start focusing on understanding it, since our work would essentially lean over this part of the project, and, therefore, we must have a clear understanding of these concepts. We established the difference between a view template and a page: **View Templates** are acomposition of views, which are defined in

the chapter “The Gamecourse UI”. As we explained there, when we have a view, we can choose to show it as a **page** with a certain name. Due to these adjustments, other ones werenecessary, especially, on the modules. Most of them have a default template that is created when therespective module is enabled. In addition, previously, it was also created a page with the same content and included directly in the navigation bar. The pages should be connected to the modules, therefore, we no longer create any page as a module is activated. If a user wants a page with the content of the module’s view template, he/she should create a page and choose that template to show.

Views Page

On the views page, many modifications were needed, mainly due to the modifications in the view templates / pages logic. Taking that into consideration, it did not make sense to access the view editor by clicking on a page card to edit it. So, we removed that, and we now only allow the user to access the view editor when he/she wants to edit a template. Another adjustment we made was to have the role type related only to the templates. Previously, both pages and templates had their role type, but it only makes sense to keep it for those we create views and, perhaps, aspects.

Furthermore, when we created a page, it will always appear on the navigation bar, and to remove it from there, we had to delete it. This was not agood approach, because we might want to deactivate it for a period. Hence, we added a new propertyto the pages - *isEnabled*, whose value is stored in the database. This new property allows us to managethe pages that should appear (or not) on the navigation bar. Besides, it was not possible to change any property of a page or a template, such as their name. Consequently, we add the configuration icon to the cards.

Navigation Bar

Even though we improved the functionality of the navigation bar by adding the *isEnabled* property to the pages, there was one feature that was not there. It did not consider the role of the user. For example, if we create a page whose template only has content for role = Teacher, it should not appear on the navigation bar when the user has role = Student. The navigation bar also needed one fix respected to the highlights. In other words, when we were on the users’ page, for instance, and then we reloaded the page, the Users tab on the navigation bar did not appear highlighted as it should, since this was the page we were at. This also occurred for the custom pages. These two features are now fully implemented, preventing those situations.

View Editor

The view editor was the focus and the main goal of this project. This part was not fully implemented and the logic behind the code was not appropriate for the required functioning of the system. Therefore, many changes were needed, including a redefinition of the schema of the database.

Logical improvements

Before diving deep into the view editor’s functionality, we fixed a bug that came from beforehand and brainstormed about the restructuring of the database and how to interact with the editor. The bug was regarding the toolbar of the tables – if we clicked on the table and then on an outside view, both respective toolbars would appear and therefore, they would end up overlapping.

Furthermore, our brainstorming led us to change the logic behind the aspects and the way we deal with them. First, aspects are variations of a view for different roles. For example, a view can be showed differently according to the person who is seeing it. Previously, the views only had an incremental id and an aspectClass (Foreign Key for the table *aspect_class*). The value of the aspectClass field was common to all the views of every aspect of the view template, or, if the view template only had one aspect, it was set to null. Then, we have the table *view_template* that saves the template id and the correspondent view id. The aspects of a view template were found by searching the views whose *aspectClass* value was equal to the *aspectClass* of the view whose id was on the *view_template* table. Also, the first view of each aspect was found with the condition **part == block && parent == null**, that is, they were obligatorily blocks. This condition was changed and now we can have view templates that are simply a text element. Thus, we had to manage this logic, to find all the aspects of a view regardless their type. As we have seen, each view can have several aspects. Inside the view editor, and while building the view templates, we use only views and their aspects. So, we had to find a way to save these correspondences between the views and the respective aspects, which was important to further build the view to present to each role. Therefore, the main changes regarding the logic to deal with the aspects were:

- Removing the *aspect_class* table since it was unnecessary in the new model as well as the *aspectClass* column of the view table.
- Adding the column *viewId* to the view table in the database, to identify all the aspects of a view.
- Saving each view / aspect according to its role. As mentioned above, before we had one entire view per aspect, it was not possible to create an aspect for a view inside a template. Now, every view can have aspects.
- Adjusting the flexibility for the aspects: now each view template can have any part as main view (block, text, image, etc.).

We could have maintained the *aspect_class* table and not adding the *viewId* column, but, in this case, we could not find the aspects of a specific view, which is now possible with the *viewId*. Regarding the *aspect_class* table, we found it useless in our new logic, and its usability was not clear even previously. As stated before, we can add a template to another view, and this can be done as a copy or by

reference, and, in this case, it means that the views will have different parents in different contexts. This enforced us to change the logic of the parent/child relation. Beforehand, a view only had one parent and it was set on the view table. We removed that column and created a new table – *view_parent* -, which has as columns: **parentId**: id of the parent view; **childId**: *viewId* of the child view; **viewIndex**: indicates the order in that relation. The *parentId* is the id of the view since a view only has one parent. However, the *childId* is the *viewId* since a view can have many aspects of a view as children.

UI improvements

Since we removed the page where the user would choose which aspect he/she wanted to create or modify, we needed to add this feature to the view editor. We decided to use dropdowns to choose the different aspects. The options are the aspects for which there is any content. By changing these dropdowns, the view contents change accordingly. When the role of the template is ROLE_SINGLE, only the viewer dropdown appears, whereas when the role is ROLE_INTERACTION, appears the user and viewer roles, as we can see in the Figure 12. By having it in the view editor, it allows the user to manage the aspects of a template without leaving the view editor. Besides, it makes possible to the user see which aspect he/she is seeing.

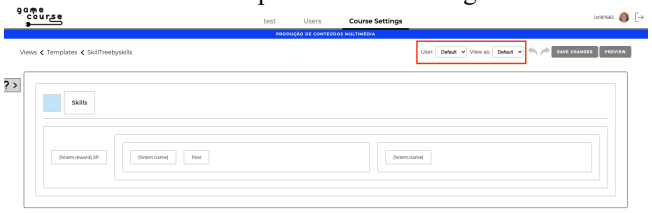


Figure 12. Aspects dropdowns for user and viewer roles.

To create new aspects for a view, we decided to add a new tool to the toolbar of each part – the Manage Aspects tool. At the end of the toolbar, we also added a new label that shows which role will see the selected view.

We created a new modal that is shown after clicking the Manage Aspects tool (Figure 13). And here we can:

- Change the aspect we want to see for that specific view (without changing the global role, whichis the one selected on the dropdown on top). If the role is the same as the global, the view willappear with a solid border. However, if the role we are seeing does not correspond to the globalone, that view will appear with a dashed border.
- Add a new aspect for this view. By clicking the "Add Aspect" button, a form appears above to selectfor which role we want to create an aspect and how. The latter has 2 options: create from scratch and create with the selected aspect as a basis.

The preview section also needed some improvements since we could not see exactly how the pagewill look like. The preview section showed the views inside the boxes as it happens in the view editor while editing, and it did not position the elements as they will be in the page. Therefore, we improved this section to give us the view as it would appear when see in a page.

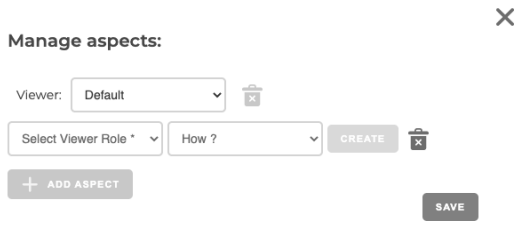


Figure 13. Manage Aspects Modal.

Edit Part Modal

While building a view resorting to the view editor, there are properties of the parts that we want to change or add. For this, in the toolbar, we have the edit part button. If the user clicks it, the edit part modal opens. In this modal, the user can define the function for the loop data, set new variables and their value, set events, change the visibility, add styling, among others specific for each part. Some of these properties must be defined using the EL. In those, there was an algorithm that gives suggestions, provides auto complete and a green light sign regarding what the user write and whether it was rightly written or not. However, only if the users start to write, this auto complete appeared.

Although there is a button that takes the user to the documentation page, in which there are some guidelines on how to write using the EL, the live support was not helpful. To tackle this problem, we expanded the modal (Figure 14), and used the CodeMirror framework.

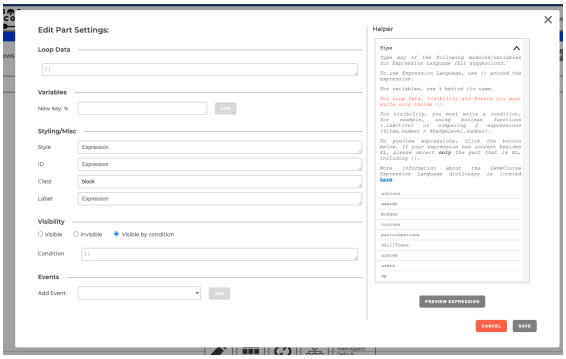
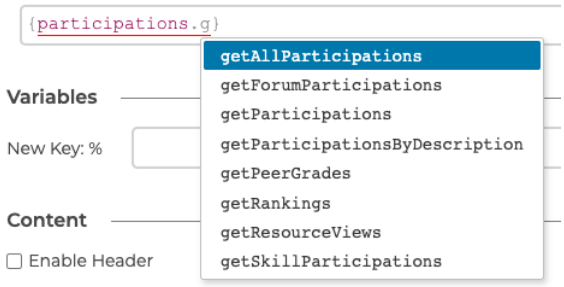


Figure 14. New edit part modal.

This framework was already used in the rules editor, so to keep it consistent, we decided to use it too in this section. For every input box that is filled resorting to the Expression Language, we replaced them with a CodeMirror box. This framework provides a suggestion box and an auto complete functionality, according to what is written (Figure 15). These suggestions are given by the system. We check what is written and what is accepted andmanage them accordingly. For example, in the loop data, the

expression should return a collection, so the user should write (inside the curly braces) is whether a variable or a module followed by function(s). The function that deals with the suggestions, detects them using Regular Expressions. For variables, itdetects with the % symbol. For



functions, it detects if it has a module name followed by a dot or modulename followed by a dot, a function, and a dot again. Variables can have properties, which are detectedwith the % symbol followed by the variable name followed by a dot.

Figure 15. Suggestions box provided while the user types.

Besides, on the right side, we placed a section for help. On the top of this section, we have a box, in which we have:

- **Tips section**: here are some tips to guide the user before/when he/she is writing. This list can becollapsed using the arrow on the top right.
- **Suggestions section**: this section is dynamic, and it changes according to what is written. Thissection will only have content is the user clicks one of the input boxes. If a function of a module, ora variable, is selected, its properties, such as arguments (for functions), return type and description appear in this section. Otherwise, it will appear the possible functions.

On the bottom, we have a preview button. By clicking it, it will open a modal to preview the selected expression if one is selected.

Course Settings Page

There were two tasks that was related to the course settings – the change of order of pages on navigationbar and the override of the default style with a personalized CSS file. We decided to use the global course settings to place these two new features. First, we added a section to change theorder of the pages on the navigation bar - Navigation. This section contains a table with all the pages but users and course settings. To manage the order of each page, we added a column to the page table, *seqId*, which indicates the order in which each page appear in the navigation bar.

Regarding the theme feature, we created another section – Styling. If the course has not a file yet, a button "Create style file" appears. Once the user clicks it, a CSS file is created and saved in the course folder. The styling in the file will only apply to the respective course and it will override the styles of every page of that course. When the course already has a style file, this can be modified through the User Interface, in an editor generated with CodeMirror. With this feature, GameCourse leverages the power of the styling, since it allows the customization of every view, even every aspect, which will impact the way users see each element, the system in general, and mainly howthey will accept gamification.

EVALUATION

We conducted the user tests with a total of 20 users, the minimum number for a meaningful analysis. Users were asked to perform 13 tasks using the system, during which we gathered meaningful metrics such as whether the task was successfully completed, the number of errors and the time expended in each task.

We gathered some details about our users, such as their age, programming experience and whether they had previous knowledge of the GameCourse system. Then we provided them with some context provided regarding GameCourse system, followed by a limited time for exploring the systems before performing the tasks:

1. On the PCM course create a new Page.

2. Configure the module "XP and levels" by adding a new level.

3. Configure the module skills by adding a new skill.

4. Configure the module badges by setting the overlay for extra as theredborder.png image by uploading it from the computer.

5. Go to your info and change your nickname to "Mariana".

6. Now we want that the PCM course looks differently. We want to change the look of some elements. For that, define a new style for this course.

7. On the PCM course create a new view template.

8. Add a text part to the outside block.

9. Create a new aspect, from scratch, for role Teacher for the outside block.

10. Create a new aspect, from scratch, for the text box for which the content is "PCM" for role Student.

11. I want the textbox for which the content is "IPM" to have another content: Student number of the user with the id=1.

12. Add the class blue_border to the outside block. (use "; blue_border;").

13. I want this view to go through all the active courses. How would you perform this?

These tasks were given randomly so there is no direct influence of the learning curve of the system. We decided to set a time limit for each task as a safeguard - **3min 30sec** for the hard tasks (tasks 11 and 13) and **1min 30sec** for the

easier ones (the remaining tasks). After the tests, we calculated the success rate as well as analyzed the other measures we made, which can be seen in Table 1.

The 13 tasks we asked to be performed had different levels of difficulty and different character. We asked users to rate each task regarding their perception of difficulty in a scale between **1 (Very Easy)** and **7 (Very Difficult)** after they had completed it.

We start by analysing the tasks related to the basic interactions with the views page, **tasks 1 and 7**, which revealed to be the easiest tasks with an average rating below 2.0 in the difficulty scale. These tasks had a 100% of success rate and only thetasks 1had some insignificant errors, mainly, becausesome users did not remember where pages are and ended up looking for the pages in other sections.

Then, **tasks 2, 3, and 4**, which are related to the modules configuration, also had a average ratingin the difficulty scale between 1.0 and 2.5. We noticed that some users were confused with the names of the modules and templates. **Task 2** was considered the easiest task and every user performed itwithout any trouble. **Task 3** required to add a new skill in the module Skills. The system also had thetemplate Skills in the views page. Therefore, one user was confused and thought the task should beperformed in the template, but as they re-read the task, they realized they were in the wrong place, andwent to the modules page and completed the task successfully. **Task 4** asked to configure the moduleBadges, and that confusion also happened to another user, but they ended up completing the task. We noticed a great increase in the interaction with modules configuration. The users that made that mistakeperformed the tasks in the very beginning. Then, when performing other configurations in the modules,they were already aware on where they should go.

Task 5 asks users to change the nickname of the logged in user and had an average perceiveddifficulty below 2.0. This task was specially interesting since some users did not see the icon of the useron the top right corner of the page. This issue led users to either not complete the task or use the Userspage to do it, which was not the goal, since this page will not be able for non-admin users. On the otherhand, most users that performed successfully this task, by going to the My Information Page, did it in a short time, and immediately saw the icon. This can be because of its location, which matches thelocation in many websites with profiles such as social networks, for example, Facebook, Instagram and LinkedIn.

We analyse **tasks 6 and 12** simultaneously, since both may require extra knowledge of CSS and, specially, the term “class” in this context in order to understand what is asked. Nevertheless, one of the users without this knowledge completed both. They were both rated with an average difficulty between 2.5 and 3.5.

Task 12, which asked to add a class to a view, was affected essentially by the misunderstanding of the real meaning of an element having a class and how they could change it. Users without any programming experience found it extra hard to perform however 2 of the 3 of this group could complete it since they could find the class label in the edit part modal. Regarding **task 6**, which asked to add new styling to a course, was not completed only by 2 users with no programming experience.

Tasks	Success Rate	Avg. completion time	Avg. number of errors
1	100%	00:33	0.3
2	100%	00:29	0
3	100%	00:59	0.3
4	100%	00:53	0.15
5	85%	00:38	0.6
6	90%	00:54	0.6
7	100%	00:23	0
8	100%	00:38	0.85
9	100%	00:38	0.6
10	95%	00:51	0.6
11	30%	03:02	-
12	90%	01:09	0.7
13	30%	03:10	-

Table 1. Results of the tests.

We then analyse the **tasks 8, 9 and 10** that were concerned about the tasks within the view editor. **Task 8** asked the users to add a new part to the outside block. This task has a perceived difficulty of around 2.5, mainly because users struggled with the icons in the edit toolbar and what they could do in which one. Regarding **tasks 9 and 10**, where users must create new aspects, the main struggle was,once again, find the right icon - the aspects icon. These last two tasks had an average rating between 2.0 and 3.0. In all these 3 tasks, users tended to first click on the edit icon to open the edit part modal. Then, they realized that the tasks could not be performed in that modal and tried to explore more the view editoruntil reach the correct icon. The main struggle of the users was regarding the aspects icon and theicon for edit layout that were not understandable or intuitive. However, after the first task of creating na aspect, users performed the second one easily. Also, a user suggested having a label for each icon thatis visible on hover. Although we already have labels, they do not appear immediately when hovering and we need to wait a few seconds until they appear.

Finally, we analyse **tasks 11 and 13**, the more difficult tasks, with an average difficulty rated above 5, which asked

the user to write expressions using EL, which takes time to understand how it works. All users that completed at least one of these tasks already had programming experience and it may be more logical after reading part of the documentation. Even users with programming experience, someonly could complete the task they performed in the second place, after failing the first one. Most usersconsulted the documentation page to try to write the desired expression. But some did not understand it. In **task 11**, we expected the user to write **{users.getUser(1).studentNumber}.** In the documentation, we can see that the properties of the object user are described as%user.studentNumber for the student number, but for users with no programming experience, it was not clear that *%user* was an user object. **Task 13** asked the user to get all the active courses. To write this expression, almost every user started by using the variable *%course* and the property *isActive*, that, by chance, we give as an example for a visibility condition in the tips, and users thought that they were supposed to use this property because of the task asking for active course. Overall, when it comes to write an expression using EL, we noticed some struggle, mainly regarding: variables, Loop Data, Libraries and Functions and Suggestions section.

CONCLUSION

As a successful technique, gamification has been used to increase the motivation of users as well as their performance while performing certain tasks. Its success relies on an appropriate design to ensure its meaningfulness, mostly in the education context. The employment of gamification in education still has some flaws regarding its adaptability since the different learners could have a significant discrepancy in their needs.

For this thesis, we improved the GameCourse system by adding new features and improving the UX of the system: new configuration pages for some modules, with import and export functionalities; in the view editor, we started by restructuring the views logic, and then improve the view editor to make it easier to use. Finally, we developed a suggestion system to give more support to use the EL in the view creation.

All the work we done has contributed to increase the system's flexibility and helped it to achieve the goal of being used in other higher education courses as it has aroused interest in other teacher of IST to use it in their courses this school year.

As future work, regarding the view editor, although it is working, the aspects deal with roles but not with specific users. It would be interesting if it was added the possibility to create an aspect for a specific user, for which we are already working. Also, for a more friendly interaction, the icons in the toolbar should be reviewed to be more comprehensible. Regarding the file picker, it would be very useful if it allows the creation and deletion of folders, instead of just uploading and deleting files. Finally, the GameCourse system already has a documentation page. However, this needs to be updated and reviewed to reflect all the changes that we made.

ACKNOWLEDGMENTS

This work was supported in part by the National Funds through the Fundação para a Ciência e a Tecnologia (FCT) under Project UIDB/50021/2020, and in part by the Project GameCourse, Portugal, under Grant PTDC/CCI-CIF/30754/2017.

REFERENCES

1. Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game designelements to gamefulness: Defining gamification. volume 11, pages 9–15.
2. Pengnate, S. F. and Sarathy, R. (2017). An experimental investigation ofthe influence of website emotional design features on trust in unfamiliar online vendors. Computers in Human Behavior, 67:49–60
3. Mekler, E., Brühlmann, F., Opwis, K., and Tuch, A. (2013). Disassembling gamification: The effects of points and meaning on user motivation and performance. pages 1137–1142
4. Brühlmann, F. (2013). Gamification From the Perspective of Self-Determination Theory and Flow. PhD thesis
5. Amaral, J. (2013). Gamecourse. Master's thesis, Instituto Superior Técnico
6. Garden, C. and Rivera, E. (2018). Putting theory into practice: Gamificationfor student engagement. InEDULEARN18 Proceedings, 10th International Conference on Educationand New Learning Technologies, pages 4563–4570. IATED
7. Landers, R. (2015). Developing a theory of gamified learning. Simulation & Gaming.
8. Kahu, E. (2011). Framing student engagement in higher education.Studies in HigherEducation, 2011
9. Bedwell, W. L., Pavlas, D., Heyne, K., Lazzara, E. H., and Salas, E. (2012). Towarda taxonomy linking game attributes to learning: An empirical study.Simulation & Gaming, 43(6):729–760
10. Monterrat, B., Desmarais, M., Lavoue, E., and George, S. (2015). A playermodel for adaptive gamification in learning environments
11. Barata, G., Gama, S., Jorge, J., and Gonçalves, D. (2013a). Engaging engineering students with gamification. InVS-GAMES 2013, pages 1–8. IEEE.
12. Barata, G., Gama, S., Jorge, J., and Gonçalves, D. (2013b). Improving participation and learning with gamification. In Gamification 2013, pages 10–17.