Gamification Tool for higher education

Juan Menéndez Buitrago

Abstract– In this project I have developed a basic gamification web-based tool that helps teachers apply gamification elements to their courses activities and keep track of them in an easy way. The development is focused on the back-end features, mainly the database design and the relations between as many gamification elements as possible. The goal of this tool is to serve as the foundation for further developments and therefore it is not intended to be a final product but a starting point for more elaborate solutions.

Keywords– gamification, education, learning, web application, dashboard, incentive, merit, narrative, level, point, PHP, MySQL, Laravel, MVC, motivation

1 INTRODUCTION

"GAMIFICATION is using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning and solve problems"[1], or as some people would say in a somewhat simplistic way, it is about making people want to do things that they wouldn't feel like doing otherwise.
As far as higher education is concerned, gamification is helping teachers engage students in learning activities making the process more attractive and effective at the same time, leveraging the potential that Internet and the always-connected portable devices such as smartphones, laptops and tablets offer which brings the classroom to virtually everywhere.
Although gamification foundations applied to learning are basically no different from classic educational psychology and techniques (giving points, feedback and encouraging collaboration), the current generation of students are used to having things presented to them in a different way, and what gamification does is presenting content in that way that they have been immersed to by playing video games since an early age, educating and motivating them at the same time.
Using games to teach new concepts and procedures is not new. The new actors in this scenario are the means, the tools made available by the spread of Internet and connectivity and the rising awareness of the value of games in the learning process.
But "gamifying" education is not an easy task. It is not only about giving points and badges and rewards but also about building a meaningful narrative with activities that successfully mimic real situations in which students can see the real value of the concepts that they are learning, so the result will produce a higher retention , effective recall and accelerate the learning curve. It is about stimulating the student's intrinsic motivation[3], the willing to learn for the pure pleasure of learning and social recognition, through the use of extrinsic motivators like rewards and merits.

This project's objective in its early stages was twofold: to learn about gamification, what it is and how it fits in an academic context, and to develop a tool that implemented those gamification components, mechanics and dynamics[2], keeping the UI simple and easy to use, with the only limitation of the time available. The resulting tool falls short in some areas but is a starting point on top of which a more elaborate solution can be built.

This document describes the development process with the following structure: firstly, there's a brief state-of-the-art section in which some of most popular applications are mentioned along with a short description. Secondly there's the bulk of the article in which the analysis, design and development processes are explained. Lastly, come the conclusions where the future work is suggested along with the personal opinions and self-evaluations.

2 STATE OF THE ART

Because of the benefits that gamification brings to learning processes, this sort of applications are gaining popularity as well as interest among academic institutions. There are several solutions available in the form of web applications that range from the category of "educational games" to "gamified platforms". Coming up next is a brief list of some of the most popular solutions divided in those two categories: "educational games" and "platforms".

• Contact e-mail: juan.menendez.buitrago@gmail.com
• Mention:Information Technology
• Directed by: Sergi robles (DEIC)
• Academic year 2016/17

2.1 Games

2.1.1 Cerebriti

[6] Cerebriti is an Spanish-speaking community in which users create and publish games about a particular subject, some of them more educational than others. These games are in the form o multiple-choice questions and users accumulate points from one game to another. As those test are designed by the community there is a big variety of subjects, not all of them interesting.

2.1.2 Socrative

[7] This is a gamification platform that allows teachers to create activities for students to do, without having to leave the platform and also to keep track of the students progress. Activities range from simple tests, multiple-choice quizzes to competitions between students in what they call "space race" in which there's a time limit to answer the questions.

2.1.3 Kahoot

[5] Kahoot brings the game inside the classroom by enabling students to play against each other individually or in groups in trivia games designed by their teachers using their smartphones. The most interesting thing from the teacher's perspective is the possibility to export the results to a spreadsheet for later processing and keep track of the students' progress and engagement.

2.1.4 Brainscape

[10] This is a digital version of the typical flashcards. Students or teachers can create a deck of cards on a particular subject and then, students can go through every card answering the questions and "flipping" the card to see the answer. Students then self-evaluate themselves and if they they consider that they haven't answer correctly, they give the card a low score so the same card will have more chances to show up again in the future. On the contrary, if the user says that she have answered the question correctly she would give it a high score that would make the card have less chances to show up again.
It is a great tool for exercising memory skills and to identify the main concepts of a particular subject.

2.2 Platforms

2.2.1 Classcraft

[9] Of all the gamification platforms/ frameworks/ tools mentioned here, this is the most elaborated. It implements many of the usual gamification components, mechanics and dynamics such as points, incentives, merits, randomness, feedback and ranks. Its design is inspired in the classic role games with characters like warriors, wizards, healers. It is highly configurable and provides graphs to keep track of the students' progress.

2.2.2 Class Dojo

[8] Class Dojo solution is focused on creating virtual classrooms in which the teacher can keep a communication channel with both students and parents as well as assign students merits and keep track of their progress in a limited way. It doesn't hold a place to do activities, so they have to take place off-line and reflect the results somehow via badges or notifications.

To sum up, there are many degrees of applying gamification to the learning process but in the end all of these solutions previously mentioned are based on the fact that the simple action of playing a game, regardless of its simplicity, often results in learning.

3 Methodology

In order to complete this project I had to firstly set up a planning and decide what type of development I would use, given the circumstances. Also I got the chance to learn how to use some tools that were new to me like PHPStorm and that proved to be of invaluable help, and I also became more proficient in the use of others such as Sublimetext and Git.

3.1 Planning

Due to the lack of previous experience, the limited time available and in order to be flexible during the development process and to be able to adapt to unexpected requirements and last-minute changes, the decision was to use an iterative development in which with every iteration there was a little improvement introduced. The Gantt diagram in appendix A.5 show the initial planning proposed which turned out to be insufficient, because even though the tasks were completed with little or no delays, the final product fell short of meeting some of the requirements. It worked well for developing the basic gamification components and mechanics, but I underestimated the time necessary for building the front-page with which the users would interact with the tool.

3.2 Tools

As for the tools used for this project, I worked with PHPstorm+xdebug[16][20] mainly for debugging, Sublimetext[19] for coding and as version control manager I used Git[17] pushing the commits to my personal Github[18] account as a back-up measure.
The use of PHPStorm for debugging was a real time-saver when dealing with complex issues that would have been cumbersome to debug with the usual *var_dump()* calls, and the use of Git or any other version control system with a remote back-up system is a life-saver for every developer.

In conclusion, these have been the tools and the planning that helped me during the last few months. Below, I will explain the core of the projet's development.

4 The Gamification tool

In order to describe in detail the development process carried out during the last months I will begin explaining in the analysis section, then I will continue with the the design decisions and describing the development and finally I will conclude with the results obtained.

4.1 Analysis

The first weeks of the project were dedicated to collecting information about what gamification is and how it is applied to learning processes. I had two meetings with the Department of Psychology in the Universitat Autònoma de Barcelona (UAB) in which they explained the main concepts of gamification and how they could fit in an academic context. Those meetings and the bibliography consulted about this subject outlined the most important gamification elements that the project should have and how they related to each other:

- **Narrative**
  Even though the term "gamification" is usually associated with giving points and badges, with rankings and levels, there's one dynamic that is the cornerstone of a successful gamification experience, and this is the narrative that gives a context to all the activities that the users have to complete. This context gives a plot to the activities, a meaning that produces a higher retention of the concepts to be learnt and higher chances to be used correctly in the future or "effective recall".
  In most cases, one narrative per course is enough to host all the activities that teachers need their students to complete. However, for the sake of flexibility, the tool allows more than one narrative per course.

- **Points**
  If the narrative is the cornerstone, points are the basic measure unit that give value and order to levels, merits, coins and incentives. Points have an owner, a category or type and an amount, and this value can be positive or, in some cases like missing a deadline, negative. This means that when one user is given an amount of points, they are not given in the form of many single units but in packs, and this packs can't be divided.
  There can be many different type of points but usually there are at least three categories: experience (XP), health (HP) and karma (KP). Karma points are usually earned by helping others, health points are earned depending on how well the student has made an activity, and experience is earned by simply participating.

- **Coins**
  When a user is given an amount of health or experience points, it also receives the same amount in coins. These coins are used to "buy" incentives or give them to other users that, who can turn them into points if they need them to reach a certain level.

- **Merits**
  Merits reflect that the user has assimilated a certain type of knowledge. In comparison they would be like the badges that the boy-scouts or girl-scouts earn when they learn to light a fire or use a knife. Some merits can be labeled under a certain type so they can be shared among different courses. For instance, there could be a merit tag for "team work" or "public speaking" and a teacher from one course can check if one of his/her students have earned it in another course.

- **Incentives**
  When students have a certain amount of coins they can "buy" an incentive that could be anything in a list that the teacher has made. From some sort of advantage in the following test (extra time, one question less...) to an invitation to a free beer at the cafeteria.

- **Levels**
  Depending on the amount of points (not the amount of coins) that a user has, he/she is tagged of belonging to a given level. Users that need to reach a specific level but don't have enough points, can ask other users for coins that once they're received can be turned into experience points.

Apart from the gamification elements there are other items that are necessary for the development of the gamification tool. Files are needed to store additional information or UI customization. Images that represent merits, levels or incentives, user avatars, pdf file for narrative or activities detailed descriptions, zip files that may hold complementary content, stylesheets and font files for customizing the narrative pages.. etc.
Also user information have to be stored and represented in a proper way, along with roles and permissions. And academic information should be taken in account for administration purposes.

- **Files**
  Files can be associated to any type of element: activities, narratives, users, merits, levels, incentives. In some cases more than one image can be linked to an element and in some other cases only one image can be attached. In the design section this will be explained in detail.

- **Users**
  At least three types of users will use the tool. Students, teachers and administrators, and they will have permissions and a dashboard according to their role.

- **Academic information** Students and teachers should only be concerned about the data regarding their courses, but since this tool is designed to hold data of every degree in the University, some tables like the users tables will hold many thousands of entries and could penalize performance in some situations.

These were all concepts that were explained to me by the faculty of the Department of Psychology and that resulted very helpful in the design of the tool. Coming up, is the explanation of the design elements on top of which has been built the gamification tool.

4.2 Design

The Model-View-Controller[21] (MVC) design pattern is nowadays the most used when developing web applications. It separates presentation layer from data access and business logic and makes development easier because separates concerns and when working in big projects, tasks can be distributed in different teams. There are many frameworks available for many different languages that use the MVC

pattern. There's Rails for Ruby, Django for Python, Java Server Faces for Java and CakePHP, Yii, Zend and Laravel for PHP just to name a few.
Laravel[11] was the option chosen to develop this project because of his popularity, available documentation and active community.

4.2.1 Front-end

As for the front-end, the option chosen was to use Bootstrap[12] for its grid system, components and its ease of use, SASS[13] as CSS preprocessor in order to mantain the CSS rules well-strucured and jQuery[14] as javascript library as it is required by Bootstrap.

4.2.2 Site structure

The site structure is divided in three main branches: student, teacher and admin. Each of them host a dashboard under */home* path from which users can browse through the courses, narratives, activities .. etc. Teacher and administrator branches hang from its own subdomain. In order to understand this design decision, the concept of route groups needs to be understood and they're explained in the development section (section 4.3).

4.2.3 Interaction with the tool

Students, teachers and administrators interact with the tool in very different ways. While teachers have an active role, creating, updating, deleting and reading elements in the courses that they teach, students have more of a passive role. They only get to participate in the activities off-line and read the results once they're uploaded by the teacher, and see their evolution in their status. This is not the desired scenario but the actual one. After more development has been carried out on this tool, students should have more of an active role being able to interact with each other transferring coins and claiming incentives, and interact with the tool uploading handouts or answering tests.
To create or give points to users, for now teachers have the only option of uploading files in *csv* format in which they specify the user NIU and the values for XP and HP. If the upload is successful the tool will parse the file and create the entries in the *points* table accordingly.

4.2.4 database tables relationships

A couple of things need to be explained about how files data is stored in the database. Some elements such as the merits, the levels and the incentives have the option of showing a little picture to describe its content. A merit, for instance, can be shown in the UI as a badge. In these cases, only the file name is required to be saved and therefore can be stored in the same table as the rest of the merit data.
Some other elements can be linked to multiple files. For example, a narrative can have attached a pdf, an image, a font file ... etc. For these type of cases in which there's a one-to-many relationship (one element can have many files) the usual approach is to have the file data stored in the *files* table with a foreign key pointing to the primary key of the element that has the file attached. However due to the fact that in this tool there are more than one type of element that can have files attached to it, the *files* table needs to store one more piece of information, the type of element that the file belongs to (Fig 1).

Regarding the database design, the full table and relationships layout is in appendix A.2

4.3 development

Some parts of the project like the route groups mentioned in the design subsection (4.2) need deeper explanation and this is what this section is for. I will go through some concepts typical in MVC frameworks like the front-controller design pattern, two-step-view pattern and friendly URLs, and other more specific to the Laravel framework. I will also mention the tools used during the development, the problems that I encountered and the decisions I had to take to overcome them.

4.3.1 Front-page design pattern

Most if not all of the frameworks in use nowadays for web designing use this design pattern[25] in which every request received by the server is redirected to a single script that:

- Loads the configuration options.
- Bootstraps the application based on the configuration.
- Redirects the execution flow to desired controller according to the request parameters and the configuration of the application's router (more information on this in subsection 4.3.2).

In order to make the HTTP server redirect every request to a single file, it is necessary to have enabled some module that does this job. For *Apache* this module is called *mod_rewrite* and needs an *.htaccess* file to be included in the project's *public* folder with the following contents:

```
<IfModule mod_rewrite.c>
   <IfModule mod_negotiation.c>
      Options -MultiViews
   </IfModule>

   RewriteEngine On

   # Redirect Trailing Slashes If Not A Folder...
   RewriteCond %{REQUEST_FILENAME} !-d
   RewriteRule ^(.*)/$ /$1 [L,R=301]

   # Handle Front Controller...
   RewriteCond %{REQUEST_FILENAME} !-d
   RewriteCond %{REQUEST_FILENAME} !-f
   RewriteRule ^ index.php [L]

   # Handle Authorization Header
   RewriteCond %{HTTP:Authorization} .
   RewriteRule .* - [E=HTTP_AUTHORIZATION:
      %{HTTP:Authorization}]
</IfModule>
```

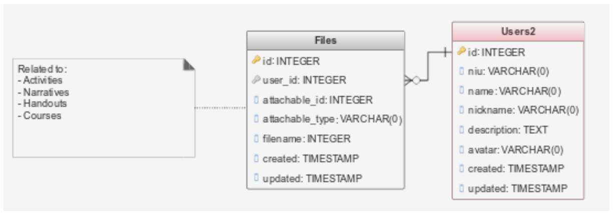


Fig. 1: files table detail

The explanation of this file's contents is beyond the scope of this article so I will only say that what this file does is redirect every request to the *index.php* file if and only if the request target isn't an actual file (an image file, a javascript file, an stylesheet file... etc) hanging from the *public* folder..

4.3.2 Two-step-view design pattern

Another pattern frequently used when working with MVC is the two-step-view pattern[26]. What this pattern does is divide the rendering of the final view in two steps one for the layout, which contains the common elements shared across most of the pages in the site like the header, footer and navigation bar, and one for the specific contents that every controller generates that eventually, and only if necessary, get embedded in the layout and sent back to the browser.
Laravel uses Blade as its template engine implements this pattern by adding an *@extends* directive at the top of each view file and including one or more *@section* directives to define blocks of content to be passed to the layout.
Once in the layout file, there is a *@yield* directive for each block of code that the layout expects to receive from the view.

4.3.3 Routing requests

As I mentioned in the front-page-controller design pattern, every request that doesn't point to an existing file under the *public* directory is redirected to the *index.php* file.
Friendly URLs therefore must not point to an existing file and they usually have a RESTful[24] format like:

```
http://(host)/(resource)/(id)
```

Then the tool, uses the URL of the request (the *resource* and the optional *id*) and the verb (GET,PUT/PATCH,POST,DELETE) and matches it to a set of rules that bind each possible URL to a controller and a method. This is how the tool knows what to do next with the data received in the request.
Moreover, this list of routes is used to compose the URL of the links that are used inside the tool, thanks to the helper function *route()*. The following snippet will create the URL admin.tfg.local/school/10

```
<a href="{{ route('admin.school.show',['id'=> 10]) }}">
   ETSE
</a>
```

An excerpt of the routes file with a little explanation can be found in the appendix A.1.

4.3.4 Route groups

The reason for this is that Laravel framework offers the possibility to organize the routes in groups, assigning subdomains, namespaces and authorization rules among many other options for the routes inside each group and thus keeping the code organized and separating responsibilities if needed. In this case, since there are three clear user roles with different permissions it made sense to separate the code like this even though it means that, since the cde will be sread along three different namespaces, the number of files would increase. An excerpt of the *routes/web.php* file can be found in appendix A.1

4.3.5 Form validation and authorization

Most frameworks encapsulate the data regarding requests and responses in *Request* and *Response* objects. Through those objects, the application can access the data in the POST request, the requested URL, the referrer IP and so on in the case of a request, and they can inject data to be returned to the browser, turn data into JSON format, or redirect among other options in the case of a response. Laravel is no different but there's also a subclass of the *Request* class, called *FormRequest* in which validation rules and authorization can take place.
Using *FormRequest* objects, there's no need to place validation or authorization tests inside the controllers actions, leaving much more cleaner and organized code.
Authorization rules are based on the permissions stored in the database and the current user's role and an extra test to see the ownership of the element to be created/ read/ updated/ deleted.
Digging in the details of how validation and authorization works would take a while to explain but luckily it is rather self-explanatory if you look at the source code, so there's a couple of snippets in appendix A.3.

4.3.6 Middleware

Middlewares provide a mechanism for filtering HTTP requests. For example, There's a middleware that verifies if the user is authenticated, there's another that tests if the cross-site-request-forgery token doesn't match, and other middlewares that verify if the user is an administrator, a teacher or a student. If a request passes the middleware's verification, the execution contnues and if it doesn't pass any of the tests an action is carried out depending on how the middleware is defined (usually a redirection or an exception is thrown).
More than one middleware can be applied to any request and they take place in the same other they are declared in the *app/Http/Kernel.php* file.

4.3.7 Authentication

Authentication can be tested again local database data or against a single-sign-on service such as the CAS provided by Universitat Aotonoma de Barcelona.
Switchin from one option to the other is as easy as changing the middleware in charge of the authentication. Local authentication is a built-in featur in Laravel and to provide CAS authentication we can take advantage of the large Laravel community and install the package phpCAS package by subfission[23].

4.3.8 File Storage

There are two approaches to storing files when working with Laravel. One, is storing files under the public folder which makes them publicly accessible to everyone who knows the corresponding URL, the other one is using laravel's built-in filesystem abstraction based on Frank de Jonge's *Filesystem*[22]. This is a very powerful tool that allows us to work with a local filesystem as well as with

remote one like Dropbox, Google drive, Amazon Web Services, Amazon Cloud Drive and others with very little configuration.
In our case the tool works with a local filesystem and every file is private and can only be downloaded if the user has the necessary permissions.
A proposal for a filesystem structure is the following.

```
storage
`- app
    |- avatars
    |  |- file
    |  |- ...
    |  `- (user niu)
    |  `- file
    `- courses
        `- (course code)
            |- file
            |- ...
            |- merits
            |  |- file
            |  `- ...
            |- incentives
            |  |- file
            |  `- ...
            |- levels
            |  |- file
            |  `- ...
            `- narratives
                `- (narrative id)
                    |- file
                    |- ...
                    |- activities
                       `- (activity id)
                           |- file
                           `- ...
```

4.3.9 CSRF, XSS and SQL-injection protection

One of the requirements established in the analysis phase was to ensure that the tool would be protected agains Cross-Site-Scripting (XSS), Cross-Site-Request-Forgery (XSRF) and SQL-Injection attacks. Basic measures to achieve those objectives are:

- Preventing output from user-generated content to contain javascript source code: in order to ensure this, Laravel has two modes of inserting output in the views. One of them displays the data passed to the view after applying the *htmlspecialchars()* function on them, and the other one displays the data as-is.

```
{{ $result }} // escaped
{!! $result !!} //unescaped
```

- Sending a token with every POST request and match it against a session variable: to prevent attackers from faking a POST request, Laravel has a middleware and a view helper defined just for that and this validation is carried out behind the scenes. If both values do not match, it throws a TokenMissmatchException.

```
// in the view file
{{ scrf_field() }} // inserts the hidden
                   // input with the token
```

- Using parameterized SQL queries: To take care of this, Laravel parameterizes every query created either using Eloquent ORM or an SQL Builder which are the most frequent ways to crate SQL queries along with raw SQL.

4.3.10 Problems and decisions

During the last four months of development there have been a few obstacles that have slowed down the progress of the project. Some of them were due to bad planning and some were unpredictable and had nothing to do with the project itself. In the end, for one reason or another some decisions had to be made:

- **Prioritize functionality before visual design**
  There were times during the project in which I found myself focusing too much and spending too much time on the visual appearance of a single element in the UI that offered no real improvement to the tool. It was just "bells and whistles" that happened a few times and almost without even noticing, but resulted in hours of precious development time wasted. The decision that came out of this was to leave all visual improvements until the final stages, and only if the main tasks are finished.

- **Focus on the teacher's dashboard**
  At the beginning of the project there was the need to populate the database tables with mock data and create the HTML forms to create and update the elements required by the tool.Because of this, the first dashboard to work on was the administrator's. The problem was that this is the biggest dashboard and the most complex as it dealt with each and every one of the elements in the tool, but as far as gamification is concerned it had little to do with it. It was consuming too much time and producing little outcome so i decided to focus on the teacher's dashboard as it reuses some of the stuff in the administrator's dashboard but works a lot more on the gamification side of the project.

- **Leave AJAX request for improvements phase**
  The amount of data that has to be rendered in some cases is calling for some requests to be made via AJAX. However and since I would considered this an improvement I decided to first make all request via regular HTTP requests and later, turn those best candidates to AJAX.

5 Results

The result of the work carried out during these months is a web-based tool that can be used as underpinnings of a more complex and complete web application. It provides the authentication and authorization capabilities as well as the implementations of the most used gamification components, mechanics and dynamics. It provides a dashboard for the three roles involved in an academic environment: students, teachers and administrators, as well as *csv* file uploading and processing.
Although in the initial planning it was considered the chance to test the tool in a controlled group of real users, this hasn't been possible because the tool has not reached the desired level of development.
Further improvements should be implemented in order to have a tool that could be deployed and used in a real-case scenario and there's a list of which ones i consider the most urgent in section 6.2 A summarized site-map of the tool can be found in appendix A.4

6 Conclusions

Having explained the development details, I will go through the conclusions. In the first place I will enumerate the fulfilled and unfulfilled goals. Then I will explain what I consider should be the next steps to take, and finally I will present my personal thoughts on this project.

6.1 Fulfilled goals

At the beginning of this project there was a lit of requirements and objectives. As it turned out, it was an overoptimistic list and not all of the items could be fulfilled due to the lack of time and bad planning as described in the development subsection (4.3).

Fulfilled

- **Access control**
  Users are able to log in the application and access their dashboard according to the role they're given: either student, teacher or administrator. However, the goal was to use the single-sign-on service provided by the University and that can't be done yet.
- **Permission system**
  Based on the role that every user is given, a set of permission are assigned to them. Basically there are four actions for every model: create, read, update and delete.
- **Administration dashboard**
  Every user with the administrator role has all the permissions necessary to create, update and delete every type of element in this tool. This was the most time-consuming part of the project.
- **Teacher dashboard**
  For every teacher, there's a dashboard in which he/she can manage the course, its students and the gamification items related to the course.
- **Student dashboard**
  Up to now, student can only use their dashboard to read their status, their point balance, level and list of incentives and merits.
- **User profile**
  A basic user profile is available for every student and teacher. It holds little information but could be easily extended if needed.
- **Points, levels, incentives, merits**
  The core of this project are these gamification elements and how they relate to each other. Everything revolves about them. How teachers rate their students, the value that every activity has and how students see their progress compared to others.

Unfulfilled

- **Statistical charts**
  To reflect the evolution of students and the effectiveness of the activities designed by the teachers on their engagement, it was an objective to provide both of them with visual tools like charts.
- **Coins**
  Coins are a gamification element that was included in the list of objectives from the beginning. However, since coins could be derived from points I decided to leave its implementation for future developments.
- **Ranks**
  Ranks help students find their place and measure their progress in relation to the rest of the class. It provides the gamification with a competitive side that, although it might motivate some type of players, it is not a fundamental feature.
- **Notifications**
  Giving feedback to users is a crucial part of the gamification process and the quicker the feedback, the better. This feature should be on top of the list for the future developments.
- **File management**
  Although files can be uploaded and downloaded to and from the tool, there's not a dashboard from where to have a general overview of the files uploaded by the teachers.

.

6.2 Future work

Future work on the tool should begin with some of the unfulfilled goals while continuing improving and fine-tuning what is already done. Eventually, the tool should not only process activity data imported in the form of *csv* files and the like, but should also allow to host at least simple activities such as multiple-choice and single choice tests.
As for the unfulfilled goals, the most important one is a notification system that would send messages to user when their status is updated in any way. when points are given, when handouts are uploaded (should this feature be implemented)... etc, because giving feedback to users is as important as providing other gamification elements or designing good narratives and activities.
A pleasing design and enabling customization of the UI depending on the narrative that give a context to the activities would also be a big improvement for the user experience and user engagement. Also, and related to the UI, some of the synchronous HTTP requests could be turned into AJAX thus improving the application performance, specially when dealing with large amounts of data, such as querying the database for a users list or points list, which happen to be the most populated database tables.
In conclusion, further development should focus on providing a notification and feedback system as well as improving the graphic design of the UI, and then continue with the list of unfulfilled goals.

6.3 Personal conclusions

Despite the disappointment for not having fulfilled part of the expectations I am very satisfied with the overall process, specially all the knowledge about gamification and learning that probably I wouldn't have acquired if i hadn't done this project. I would definitely continue reading about gamification and maybe develop a simpler tool as a personal project

in my free time.
This experience has helped me identify some of my biggest mistakes when developing: wasting time on details instead of focusing on the essential and then work up from there, making time estimations more accurately taking in account that "knowing how to do something" can make me feel too optimistic about the time it will take to implement it. Keep an open communication channel with the "client" even if there are no updates worth mentioning. And the most important in my opinion: "get things done" and not spend a lot of time learning before starting to code, because most of the problems don't need to be an expert to start working on them and it is always better to write a first draft and improve it later, than waiting until reaching a certain degree of expertise.
Finally, I find that the project was not particularly difficult and that the courses that I had taken during the degree in the mention of Information Technology were good foundations for working on it. Had I organized myself better and with a little more time the results would have been more useful, but I am still very proud of the overall process.

References

[1] K. Kapp, The gamification of learning and instruction. San Francisco: Pfeiffer, 2012.

[2] "Gamification Mechanics, Dynamics and Components," University of Wisconsin-Platteville. [Online]. Available: https://www.uwplatt.edu/ttc/gamification-mechanics-dynamics-and-components. [Accessed: 27-Jun-2017].

[3] "Motivation," Wikipedia, 25-Jun-2017. [Online]. Available: https://en.wikipedia.org/wiki/MotivationIncentive_-theories:_intrinsic_and_extrinsic_motivation. [Accessed: 27-Jun-2017].

[4] M. Stauffer, Laravel: up and running: a framework for building modern PHP apps. Sebastopol, CA: OReilly, 2017.

[5] "Kahoot! — Learning Games — Make Learning Awesome!", Kahoot!, 2017. [Online]. Available: https://kahoot.com/. [Accessed: 24- Jun- 2017].

[6] "Cerebriti - Demuestra lo que sabes", Cerebriti.com, 2017. [Online]. Available: http://www.cerebriti.com/. [Accessed: 24- Jun- 2017].

[7] "Socrative", Socrative.com, 2017. [Online]. Available: https://www.socrative.com/. [Accessed: 24- Jun- 2017].

[8] "Learn all about ClassDojo", ClassDojo, 2017. [Online]. Available: https://www.classdojo.com/. [Accessed: 24- Jun- 2017].

[9] "Classcraft - Make learning an adventure", Classcraft, 2017. [Online]. Available: http://www.classcraft.com/. [Accessed: 24- Jun- 2017].

[10] "The Best Online and Mobile Flashcards App — Brainscape", Brainscape.com, 2017. [Online]. Available: https://www.brainscape.com/. [Accessed: 24- Jun- 2017].

[11] "Love beautiful code? We do too.," Laravel - The PHP Framework For Web Artisans. [Online]. Available: https://laravel.com/. [Accessed: 25-Jun-2017].

[12] "Bootstrap · The world's most popular mobile-first and responsive front-end framework.," Bootstrap · The world's most popular mobile-first and responsive front-end framework. [Online]. Available: http://getbootstrap.com/. [Accessed: 25-Jun-2017].

[13] "Syntactically Awesome Style Sheets," Sass. [Online]. Available: http://sass-lang.com/. [Accessed: 25-Jun-2017].

[14] "jQuery," jQuery. [Online]. Available: http://jquery.com/. [Accessed: 25-Jun-2017].

[15] "DataTables — Table plug-in for jQuery", Datatables.net, 2017. [Online]. Available: https://datatables.net/. [Accessed: 26- Jun- 2017].

[16] "PhpStorm IDE :: JetBrains PhpStorm," JetBrains. [Online]. Available: https://www.jetbrains.com/phpstorm/. [Accessed: 27-Jun-2017].

[17] "Git," Git. [Online]. Available: https://git-scm.com/. [Accessed: 27-Jun-2017].

[18] "Build software better, together," GitHub. [Online]. Available: https://github.com/. [Accessed: 27-Jun-2017].

[19] "Sublime Text," Sublime Text: The text editor you'll fall in love with. [Online]. Available: http://www.sublimetext.com/. [Accessed: 27-Jun-2017].

[20] D. R. derick@xdebug.org, "Xdebug - Debugger and Profiler Tool for PHP," Xdebug - Debugger and Profiler Tool for PHP. [Online]. Available: http://xdebug.org/. [Accessed: 27-Jun-2017].

[21] "Model–view–controller," Wikipedia, 25-Jun-2017. [Online]. Available: https://en.wikipedia.org/wiki/Model-view-controller. [Accessed: 27-Jun-2017].

[22] T., "thephpleague/flysystem," GitHub, 08-Jun-2017. [Online]. Available: https://github.com/thephpleague/flysystem. [Accessed: 27-Jun-2017].

[23] S., "subfission/cas," GitHub, 07-Mar-2017. [Online]. Available: https://github.com/subfission/cas. [Accessed: 27-Jun-2017].

[24] "Representational state transfer," Wikipedia, 26-Jun-2017. [Online]. Available: https://en.wikipedia.org/wiki/Representational_-state_transfer#Relationship_between_URL_and_-HTTP_methods. [Accessed: 27-Jun-2017].

[25] “Front Controller,” P of EAA: Front Controller. [Online]. Available: https://martinfowler.com/eaaCatalog/frontController.html. [Accessed: 27-Jun-2017].

[26] “Two Step View,” P of EAA: Two Step View. [Online]. Available: https://martinfowler.com/eaaCatalog/twoStepView.html. [Accessed: 27-Jun-2017].

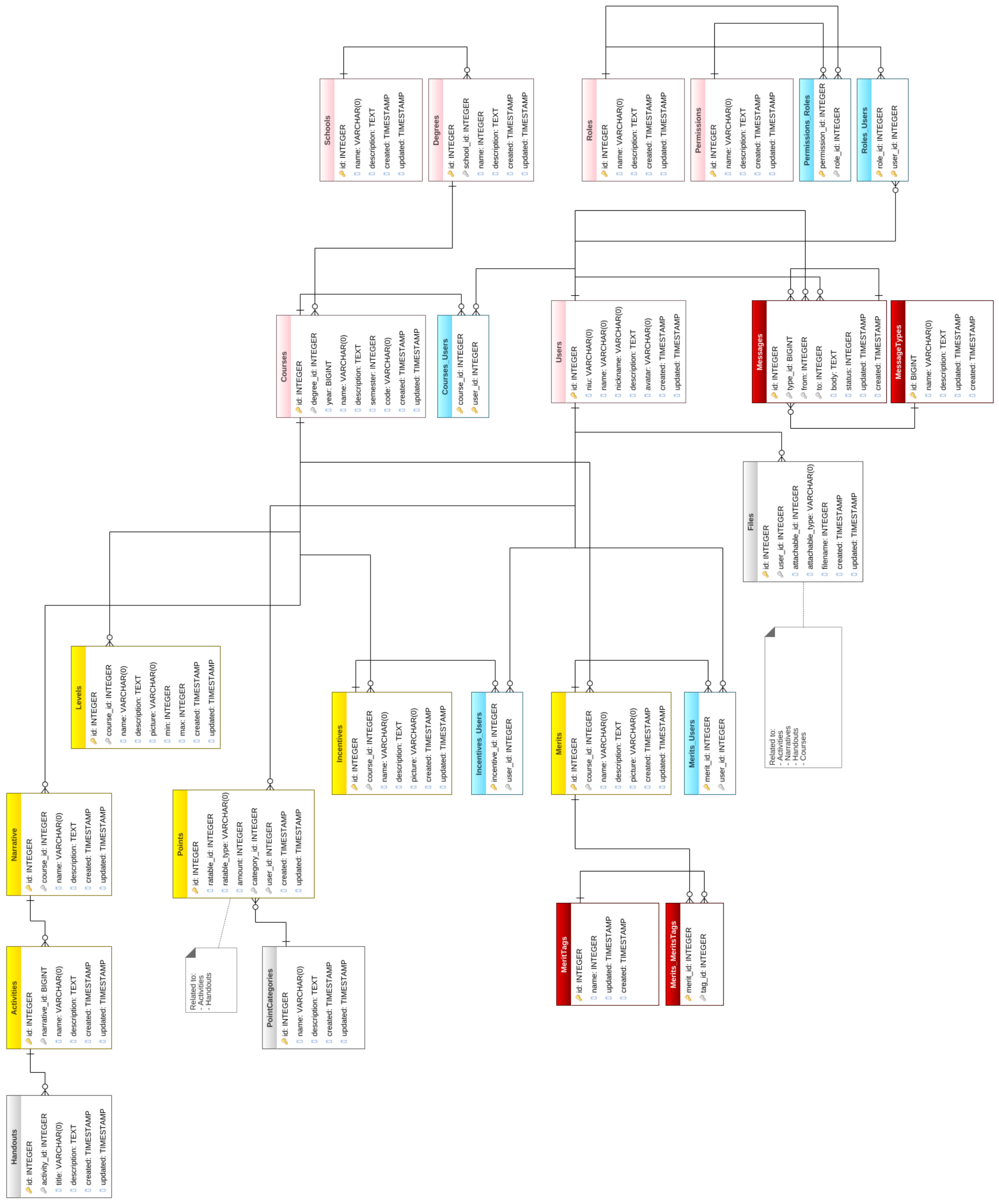
APPENDIX

A.1 Route groups

```php
<?php

use App\PointCategory;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

///
/// Resources
///
Route::group([
      'namespace' => 'Admin',
      'middleware' => 'admin',
      'domain' => 'admin.tfg.local',
      'as'=>'admin.'
   ], function () {
   Route::get('home', 'SchoolController@index')->name('home');
   Route::resource('school', 'SchoolController');
   Route::resource('degree', 'DegreeController');
   Route::resource('role', 'RoleController');
   Route::resource('permission', 'PermissionController');
   Route::resource('user', 'UserController');
   Route::resource('course', 'CourseController');
   Route::resource('level', 'LevelController');
   Route::resource('merit', 'MeritController');
   Route::resource('activity', 'ActivityController');
   Route::resource('point', 'PointController');
   Route::resource('incentive', 'IncentiveController');
   Route::resource('pointCategory', 'PointCategoryController');
   Route::resource('file', 'FileController');
   Route::resource('narrative', 'NarrativeController');
   Route::resource('handout', 'HandoutController');
   Route::get('avatar/{filename}','ImageController@profilePicture')->name('avatar.get');
});

Route::group([
      'namespace' => 'Teacher',
      'middleware' =>'teacher',
      'domain' => 'teacher.tfg.local',
      'as'=>'teacher.'
   ], function () {
   Route::get('home', 'HomeController@index')->name('home');
   Route::resource('file', 'FileController');
   Route::resource('handout', 'HandoutController');
   Route::resource('narrative','NarrativeController');
   Route::resource('activity','ActivityController');
   Route::resource('course','CourseController');
   Route::resource('merit','MeritController');
   Route::resource('level','LevelController');
   Route::resource('incentive','IncentiveController');
   Route::resource('student','StudentController');
});

Route::group(['
      domain' => 'tfg.local',
      'middleware' => 'student'
   ], function(){
   Route::get('home', 'HomeController@index')->name('home');
   Route::get('avatar/{filename}','ImageController@profilePicture')->name('avatar.get');
   Route::get('course/{course}','CourseController@show')->name('course.show');
   Route::get('narrative/{narrative}','NarrativeController@show')->name('narrative.show');
   Route::get('activity/{activity}','ActivityController@show')->name('activity.show');
   Route::get('school/{school}', function(){
      redirect('home');
   })->name('school.show');
   Route::get('degree/{degree}', function(){
      redirect('home');
   })->name('degree.show');
   Auth::routes();
});

Route::get('/', function () {
   return view('welcome');
});

Route::get('/merit/image/{merit}/{filename}', 'ImageController@meritPicture')->name('merit.image');
Route::get('/auth/login', function(){
   cas()->authenticate();
});
```

A.2 Database diagram

In red, tables that are not implemented but should be in future work for the notification system and to enable the transverse competences tagging system.



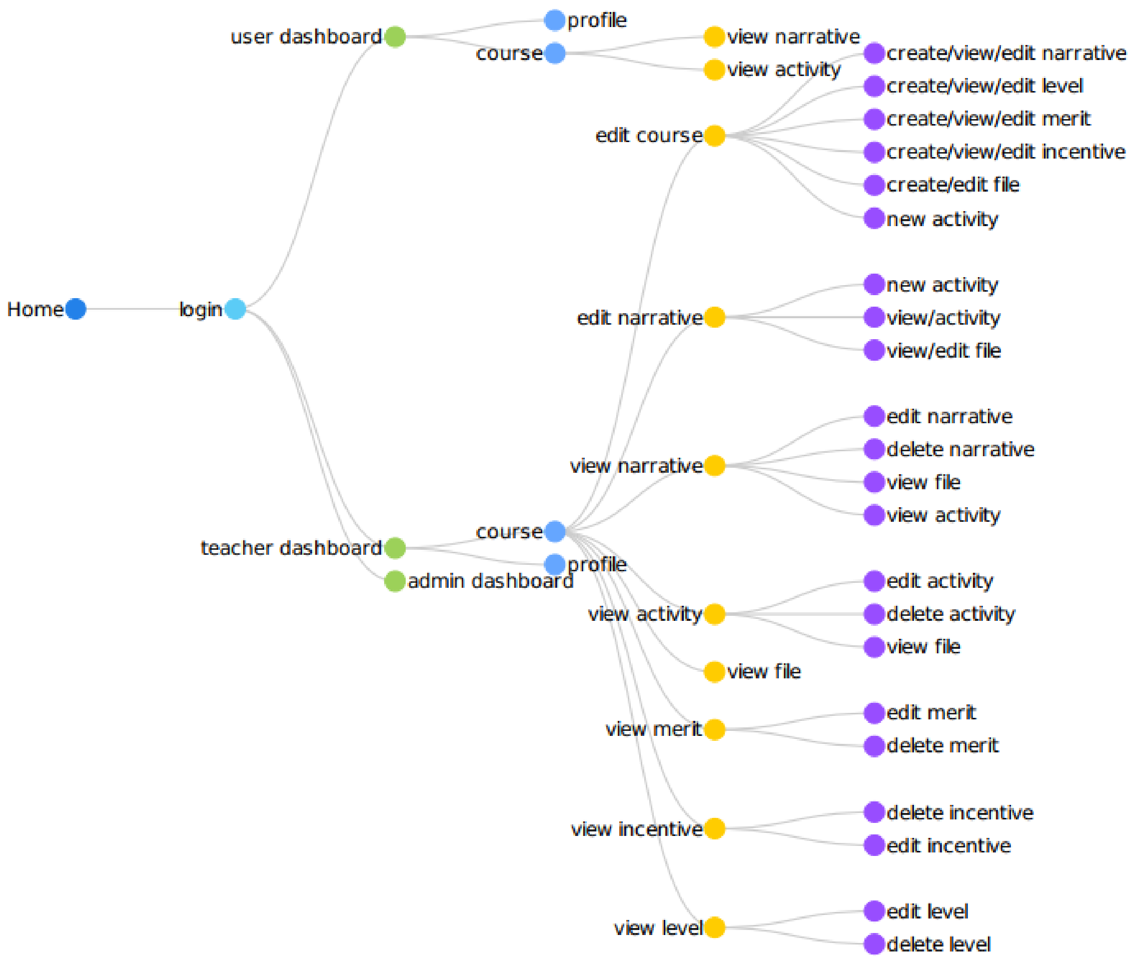
A.3 Validation and authorization

This is the *FormRequest* that performs the validation and authorization when someone tries to update a narrative. The *authorize()* method tests if the current user has the 'update-narrative' permission, and the second parameter passes the narrative model that it is being updated so the tool can see if this narrative belongs to this user. The logic behind this test is defined elsewhere, in what it is called the *AuthServiceProvider*, which is loaded during bootstrap.
The *rules()* method is quite easy to understand too. It returns a set of key-value pairs. The key is the name of the value received in the request, and the value is a string that defines the validation functions and optional parameters. In this example both name and *course_id* values are required, the name must be less than 256 characters and the *course_id* must exist in the *courses* table in the *id* column.

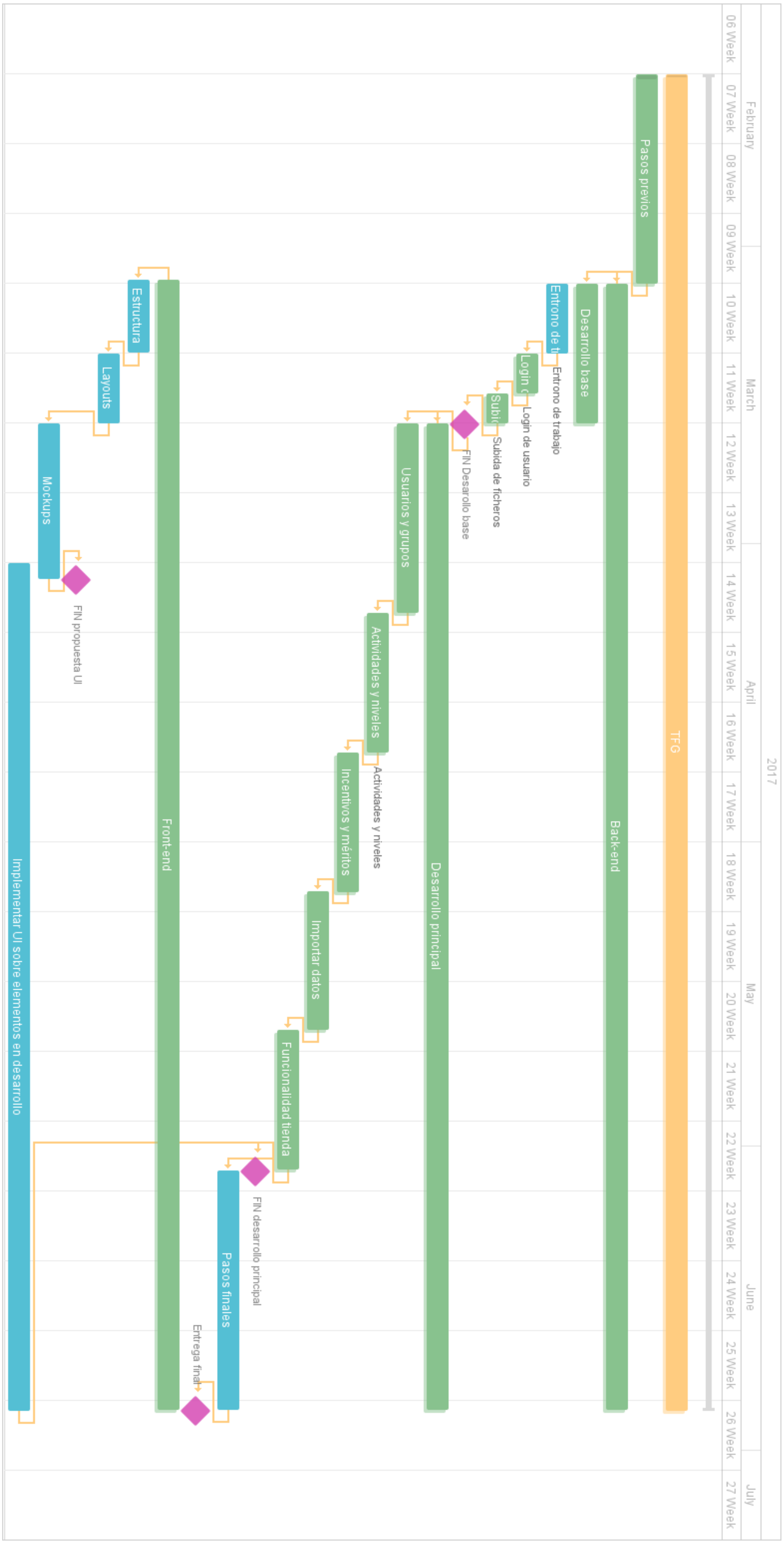
```php
<?php

namespace App\Http\Requests\Narrative;

use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Support\Facades\Gate;

class UpdateNarrativeRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return Gate::allows('update-narrative', $this->route('narrative'));
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'course_id' => 'required|exists:courses,id',
            'name' => 'required|max:255'
        ];
    }
}
```

A.4 Site map

This is the summarized version of the site map. The branch corresponding to de administrator has been pruned for the sake of simplicity and due to the fact that administrators must have access to every level of the tool and therefore it is assumed that there is a section for every component of the tool.



A.5 Gantt diagram


February
March
April
May
June
July
06 Week
07 Week
08 Week
09 Week
10 Week
11 Week
12 Week
13 Week
14 Week
15 Week
16 Week
17 Week
18 Week
19 Week
20 Week
21 Week
22 Week
23 Week
24 Week
25 Week
26 Week
27 Week
TFG
Pasos previos
Back-end
Desarrollo base
Entrono de trabajo
Login de usuario
Subida de ficheros
FIN Desarollo base
Desarrollo principal
Usuarios y grupos
Actividades y niveles
Incentivos y méritos
Importar datos
Funcionalidad tienda
FIN desarrollo principal
Pasos finales
Entrega final
Front-end
Estructura
Layouts
Mockups
FIN propuesta UI
Implementar UI sobre elementos en desarrollo