Desarrollo de gamificación para apoyo de aprendizaje online

Francisco Javier Pacheco Herranz

Javier Romero Pérez

GRADO EN INGENIERÍA INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin de Grado en Ingeniería Informática

Madrid, septiembre de 2018

Director: Jorge Gómez Sanz

Índice

Índice de ilustraciones

Índice de abreviaturas

API	*Application Programming Interface*
Bash	*Bourne again Shell*
CLOC	*Count Lines Of Code*
CSV	*Comma-separated Values*
MOOC	*Masive Online Open Course*
PHP	*Hypertext Preprocessor*
SQL	*Structured Query Language*

Resumen

La inclusión de la gamificación en la educación es una opción en auge. Se presenta como una herramienta educativa con impacto positivo en el desempeño académico de los estudiantes. En este proyecto se propone un *software* para *Bolotweet*, desarrollado por docentes de la facultad y basado en el servidor de microblogueo *Statusnet*. El nuevo software se construye sobre el juego de ordenador de código abierto *Robocode*. La gamificación consiste en la ejecución de batallas entre tanques robot. El trabajo de los alumnos en *Bolotweet*, donde tendrán que completar tareas que posteriormente serán evaluadas por su profesor, se verá recompensado con un robot cuya inteligencia se ajustará a los resultados obtenidos en estos ejercicios. Dichos robots se enfrentarán posteriormente en batallas, de las que se registrarán los resultados para mantener un ranking interno e incentivar así a los estudiantes a seguir completando las tareas de la plataforma. Cuantas más tareas completadas, más poderoso será el robot. El trabajo aporta también unos experimentos para ajustar la potencia de los robots de tal forma que no desincentive el seguir trabajando en *Bolotweet* porque sea imposible ganar a ciertos compañeros.

Palabras clave: Metodología docente, Microanotaciones, Gamificación, Educación, Microblogueo, Evaluación continua, Videojuegos, e-learning

Abstract

Gamification in education is an educational methodology which is attracting lots of atention. It is expected to bring a positive impact on the students' academic performance. In this project, a gamification extension is going to be developed for a microblogging teaching support tool named Bolotweet. The extension is based on the open source computer game Robocode, where robot tanks programmed by students fight each other. The integration of both elements will be guided by the student performance in Bolotweet. Students will be assigned robot tanks in Robocode whose behaviour will be more capable as their owners improve their grades in Bolotweet. These robots will face each other in battles, and the results will be recorded in a ranking. This ranking will serve to engage students to keep completing the platform's tasks. Determining some degree of fairness in the battles has been challenge too. There is always a chance that students will lower grades win over higher grades. The Project includes a case study that illustrates expected evolution of the gaming performance with respect to the progress in Bolotweet.

Keywords: Teaching method, Micro-annotations, Gamification, Education, Micro-blogging, Continuous evalutation, Videogames, e-learning

1. Introducción

1.1. Motivación

La plataforma *Bolotweet* ha sido utilizada durante varios años en algunas asignaturas de la facultad. Se basa en un sistema de microblogueo: los alumnos han de escribir textos breves, sintetizando en estos algo aprendido de la asignatura. Posteriormente, estos mensajes son evaluados por el profesor y los alumnos reciben un *feedback* en el que se les indican los errores contenidos en sus mensajes [22].

El uso de la plataforma resulta positivo porque requiere hacer un ejercicio de síntesis de la información que provoca que el alumno retenga con más facilidad los conocimientos impartidos en la clase.

Ese ejercicio de redacción obliga al alumno a repasar todo lo que ha sido explicado, analizar dicha información y seleccionar qué considera más importante. Se espera que la consecuencia de realizar este esfuerzo sea un refuerzo del aprendizaje y una forma de indicar al profesor cómo progresa la clase. Por tanto, de cara al examen, si el alumno ha realizado un buen trabajo en la plataforma, necesitaría hacer un esfuerzo menor al de otros que hayan tenido dificultades para expresarse.

Aunque el impacto de este trabajo es positivo, se quiere valorar el efecto de la gamificación y conseguir una mayor participación.

1.2. Objetivos

El proyecto busca aumentar el porcentaje de alumnos que completen con asiduidad las tareas que les son mandadas en *Bolotweet*, además de enriquecer y amenizar dicha labor. Para ello, se quiere ofrecer una experiencia que sea suficientemente atractiva para los estudiantes. Se considera que el uso de un juego puede motivar a los alumnos a participar en la herramienta, pues puede producir un "cambio en la actitud del usuario sin la necesidad de usar la coerción o el engaño" [3]. Así, se persigue aumentar su motivación intrínseca, que les atraiga a participar en actividades de *Bolotweet*.

El juego considerado para este trabajo es Robocode [28], que es un *battle royale* donde tanques robot, programados en Java, luchan entre sí. Se pretende combinar las funcionalidades de las plataformas *Bolotweet* y *Robocode*, donde los tanques representen a los estudiantes y su desempeño en *Bolotweet*. La combinación de ambos elementos se llamará *Bolocode*.

El juego, como se ha venido diciendo, ha de estar bien diseñado. K. Kapp define un juego bien diseñado como "un sistema en el que los jugadores se ocupan en un desafío abstracto, definido mediante reglas, interactividad y retroalimentación que resultan en un resultado cuantificable provocando a menudo una reacción emocional" [7].

Esta reacción emocional ha de estimularse midiendo la inteligencia de los avatares de los alumnos, ajustando sus comportamientos para que las partidas resulten entretenidas

e igualadas. Es importante que la diferencia entre los comportamientos de los avatares no sea tal que las partidas entre dos robots de distinto nivel las gane siempre el de mayor nivel, pues las partidas perderían emoción y los alumnos con peores puntuaciones podrían verse desalentados [5]. Se pretende que la experiencia de visionar una partida incite a los alumnos a seguir trabajando sobre la plataforma.

Además de proponer mecánicas que estimulen y motiven a los estudiantes, otro factor que puede influir es la estética, pues se considera importante a la hora de atraer la atención de los usuarios [11]. Otra de las razones por la cual en el proyecto se decantó por utilizar *Robocode* fue que el juego permite modificar la apariencia de los robots; más concretamente, permite cambiar los colores en varias partes del cuerpo de los avatares.

Por último, se procura reducir al máximo la carga de trabajo que recaiga sobre el profesor; todo el proceso de creación de avatar y ejecución de una partida ha de ser lo más automático posible, todo lo que la tecnología posibilite.

Por todas estas razones, se proponen las siguientes mejoras que serán desarrolladas en el resto de la memoria:

- Automatizar la ejecución de un juego multijugador con parámetros personalizados, especialmente los relacionados con la construcción de avatares y las condiciones de la partida.
- Integrar la plataforma *Bolotweet* con dicho juego, pudiendo recabar de esta la información suficiente acerca del desempeño de los alumnos para poder construir los avatares.
- Desarrollar avatares con diferentes niveles de inteligencia para poder establecer un sistema de niveles.
- Interpretar la información recogida de *Bolotweet* y transformarla en asignación de habilidades para los avatares. para ejecutar partidas entretenidas y estimulantes. En dichas partidas, además, se pretende que los avatares de mayor nivel ganen con asiduidad, pero permitiendo que los avatares de niveles inferiores puedan dar alguna sorpresa y obtener una victoria, sin que sea esta situación un hecho frecuente.
- Importar a *Bolotweet* los resultados de las partidas para mantener un seguimiento de las puntuaciones obtenidas por los alumnos participantes.

1.3. Método de trabajo

El proyecto es desarrollado por dos personas, por lo que ha sido posible paralelizar el desarrollo del mismo.

Por un lado, fue necesario comprender el funcionamiento de los robots de *Robocode*, desarrollado en Java. A lo largo del desarrollo del proyecto, el comportamiento de estos robots ha sido ajustado según los resultados obtenidos en las pruebas que iban siendo realizadas. Estas pruebas eran desplegadas de manera independiente de *Bolocode*, con

robots de prueba que no estaban asociados a ningún alumno y, por tanto, los resultados de las batallas no inferían en la clasificación de los estudiantes.

Cuando el rendimiento de un robot se consideraba adecuado, se exportaba al sistema para continuar siendo evaluado a largo plazo. Esto significa que quería comprobarse que la evolución de la posición ocupada por un alumno en la clasificación variaba de forma acorde a las puntuaciones obtenidas en sus mensajes. Se buscaba que los robots de los alumnos con mejor trayectoria en *Bolotweet* ocupasen las primeras posiciones.

Por otro lado, se instaló *Bolotweet* para ser probado localmente. Las primeras pruebas iban dirigidas a comprender el funcionamiento de los plugin de la plataforma, y entender cómo se relacionaban tanto con la entrada del usuario como con la base de datos. Una vez se comprendió su funcionamiento, se desarrolló el plugin que, primeramente, funcionaba de manera independiente al *Robocode*; consistía en el mantenimiento de un ranking, y las puntuaciones eran introducidas manualmente.

Una vez la actualización del ranking funcionaba de manera correcta, y era posible asociar rankings a grupos de clase, se trabajó en la integración de ambas plataformas mediante scripts de administración.

Para ello fue necesario adaptar el código fuente de *Bolotweet*, escrito en PHP y que usa la base de datos *MySQL*. Hubo que añadirle funcionalidad extra: almacenamiento de resultados en ficheros legibles por el sistema y carga de robots desarrollados en plataformas externas.

Cuando todos estos elementos estuvieron preparados, el trabajo consistió en ajustar el funcionamiento de estos y su integración para dotar al sistema de robustez: contemplar casos en los que el usuario introducía parámetros erróneos, desaparición de ficheros de configuración, caídas del sistema imprevistas, etc.

La codificación final del sistema puede consultarse en el enlace de *Github* provisto en la bibliografía [29].

1.4. Organización del documento

El documento se compone de 6 capítulos y 2 apéndices descritos a continuación.

Este primer capítulo sirve a modo de introducción, planteando la idea del proyecto y los objetivos que se persiguen alcanzar.

En el segundo capítulo se introduce el estado del arte. Primeramente se mostrarán ejemplos de gamificación aplicados en distintas ámbitos educativos, desde grupos de alumnos de edades más tempranas hasta estudiantes universitarios. Para cada ejemplo, se indicará cómo se ha aplicado la gamificación y cómo ha influido en los desempeños de los alumnos en los que ha sido aplicada. También se observará algún caso de gamificación aplicada en un ámbito no educativo. Después, se ilustrará el funcionamiento de las plataformas *Bolotweet* y *Robocode* por separado.

En el tercer capítulo se describe la propuesta *Bolocode*. Se introducirá su funcionamiento y qué elementos de cada plataforma serán aprovechados, además de enlazarlo con los objetivos propuestos. También se detallarán una serie de directrices generales y un escenario de uso que ilustrará el funcionamiento global del sistema.

En el cuarto capítulo se describe el desarrollo de *Bolocode*. En primer lugar se presentan los casos de uso para cada uno de los actores del sistema, también descritos. Después se detalla la arquitectura de la solución. Para ello, se presenta el plugin desarrollado en *Bolotweet* y los scripts de administración necesarios para el correcto funcionamiento del sistema. A continuación, mediante diagramas de estado, se muestra el funcionamiento del sistema para, en último lugar, explicar el funcionamiento interno de *Robocode.*

En el quinto capítulo se realiza una evaluación del trabajo. En este apartado se pretende demostrar que el sistema funciona correctamente y que, además, la gamificación funciona. Para ello, se simulará un escenario de uso real en un grupo de alumnos que irán completando tareas y recibiendo calificaciones, a la vez que el profesor va ejecutando batallas y los rankings son actualizados. Se pretende demostrar que los alumnos con mejores desempeños en *Bolotweet* ganan más puntos y están más arriba en la clasificación. En ese capítulo también se hará una evaluación del esfuerzo realizado mediante métricas de ingeniería del software y se mostrará una aproximación al esfuerzo que requirió la instalación y configuración de las plataformas. Por último, se detallará cuál ha sido la labor de cada integrante del proyecto dentro de este.

En el sexto capítulo se presentan las conclusiones del proyecto y el cumplimiento de los objetivos propuestos.

En el Apéndice A se proporciona un manual de uso para el profesor, detallando en este todas las funcionalidades que proporciona el sistema y cómo ejecutarlas.

En el Apéndice B se proporciona un manual de uso para el administrador. Presenta todas las herramientas disponibles para gestionar el sistema, desde el alta de usuarios hasta la activación del plugin para un grupo concreto.

2. Estado del arte

En el estado del arte se revisarán los efectos de la gamificación en general y, concretamente, en el ámbito educativo.

Primeramente se introducirá el concepto de gamificación y se expondrán una serie de pautas a seguir para aplicar una gamificación correcta. Después se citarán ejemplos de gamificación de los que se extraerán una serie de conclusiones aplicables posteriormente a la propuesta.

Por último, se detallará el estado actual de *Bolotweet* y se introducirá el videojuego *Robocode*, sus mecánicas y por qué es la opción elegida como herramienta de gamificación.

2.1. Introducción a la gamificación

Es interesante saber que la gamificación no solo es aplicable al ámbito educativo, sino que existen otros ejemplos como Nike+ [18], las escaleras musicales de la estación de tren de Odenplan [18, 20] y un radar en una calle de Estocolmo que, al respetarlo, permitía al conductor participar en una lotería cuyo premio era el dinero acumulado del pago de las multas por parte de aquellos conductores que no respetaban el límite de velocidad [18, 21].

Kapp define la gamificación como "el uso de mecánicas basadas en juegos, estéticas y pensamiento de juego para enganchar a la gente, motivar la acción, promover el aprendizaje y resolver problemas" [7].

Lee Sheldon, integrante del programa *Interactive Media and Game Development*, del Instituto Politécnico de Worchester, y autor del libro *The Multiplayer Classroom: Designing Coursework as a Game*, opina que la educación actual está mal enfocada [13].

"Las clases se convierten en rutina. Los exámenes están estandarizados. Las medidas se vuelven más importantes que el conocimiento. El fracaso se penaliza con una "F" grande. ¿Pero qué estamos midiendo exactamente? ¿La habilidad de un niño para aprender? ¿Nuestra habilidad para educar?" Sheldon está en contra de, llegada una edad, paralizar el aprendizaje natural de un niño; aquel que se basa en experimentar con la naturaleza, aprendiendo de los errores. Es por ello que aboga por el uso de los videojuegos como herramienta de apoyo en la educación.

A lo largo de su obra propone mecánicas propias de los videojuegos que aplicadas de manera correcta en una clase pueden provocar un impacto positivo en el desempeño de los alumnos y, por ende, en sus calificaciones. Estas mecánicas serán descritas en la sección 2.2.

Para entender cuándo una mecánica está aplicada correctamente, es necesario definir dos tipos de motivaciones [1, 10]. La motivación intrínseca es el interés que tienen los

estudiantes en lo que están aprendiendo y en el proceso en sí, mientras que la motivación extrínseca es el interés en el aprendizaje para lograr un fin, relativamente desasociado del contenido y de la materia de aprendizaje.

Los diversos estudios realizados, y que a lo largo del documento irán siendo citados, apuntan a que una gamificación bien ejecutada produce que los estudiantes participen porque les motiva hacerlo. Esto es una potenciación de la motivación intrínseca, que es el objetivo que un educador se marca cuando gamifica una asignatura [5]. Sin embargo, es posible que estos alumnos puedan ver aumentada su motivación extrínseca por cómo se aplica la mecánica de un juego [12].

En relación con la herramienta *Bolotweet*, lo que se busca es que los estudiantes utilicen con regularidad la plataforma no como un método para aprobar, sino que les resulte entretenido y motivante utilizarla, siento esta motivación potenciada la motivación intrínseca. El grado de motivación extra a alcanzar se basará en mejoras para su avatar en un juego multijugador.

Sin embargo, es preciso medir cuidadosamente estas recompensas. Varios autores han señalado errores relacionados con estas que pueden devaluar el resultado de una gamificación [5].

Uno de los errores es conocido como el efecto *Overjustification* [6, 9]. Este efecto se da al mitigar la motivación intrínseca mediante motivaciones extrínsecas innecesarias, particularmente cuando la tarea es interesante y beneficial para el usuario [15]. En el libro *Punished by reward*s [9], de Alfie Kohn, se citan varios estudios en los que este efecto queda demostrado:

- Uno de los estudios se centró en el grupo de estudiantes que participaba en el periódico de su universidad. Primeramente trabajaron voluntariamente y, a medida que el tiempo avanzaba, la calidad de los titulares aumentaba y necesitaban menos tiempo para escribirlos. Después, se seleccionó a un grupo de ellos y se les comenzó a pagar por cada titular que conseguían. El resultado fue que los estudiantes pagados se estancaron y no mejoraron sus titulares, mientras que los alumnos que no habían sido pagados seguían mejorando. En este caso, la motivación extrínseca –la recompensa económica- devaluó la motivación intrínseca –el trabajar en un periódico por querer informar al resto de estudiantes-.
- Otro de los estudios propuso a unos alumnos de cuarto grado (en EEUU, niños de 9-10 años) regalarles un caramelo o un juguete por sus calificaciones. El resultado fue que dichos niños bajaron su rendimiento. En este caso, de nuevo una motivación extrínseca –la recompensa en forma de regalos- devaluó la motivación intrínseca –estudiar porque están interesados en aprender-.
- El último estudio interesante dio como resultado que un grupo de niños a los que les mandó dibujar unos cuantos dibujos y a los que no se les pagó dibujaron obras de mayor calidad que otro grupo de niños que sí fueron pagados. De

nuevo, la motivación extrínseca –la recompensa económica- devaluó la motivación intrínseca –dibujar por placer-.

Por tanto, puede deducirse que no solo vale con recompensar un trabajo bien hecho, sino que hay que medir qué recompensas afectarán de manera positiva a la motivación intrínseca de los estudiantes [5].

El efecto *Overjustification* fue estudiado por Thom, Millen y DiMicco en 2012. El estudio consistió en eliminar todas las recompensas que se ofrecían en una red social de una empresa estadounidense. Esta red social contaba con una completa colección de mecanismos propios de los videojuegos, tales como insignias y una tabla de clasificación basada en puntos entregados a los usuarios que posteaban fotos y hacían comentarios.

El estudio comenzó con un experimento: durante seis meses se eliminaron dichas recompensas a la mitad de los usuarios del sistema. Cuando finalizó este periodo de tiempo, se advirtió que de los usuarios que se registraron tras el comienzo del mismo, los que ganaban puntos tenían una participación más activa dentro del sitio; sin embargo, la proporción de usuarios recién registrados que participaban activamente en la red era la misma en ambos grupos.

Diez meses más tarde se volvió a eliminar todo elemento de gamificación, esta vez a todos los usuarios. Se monitorizaron durante cuatro semanas su actividad: las dos últimas semanas previas a la eliminación del sistema y las dos primeras semanas tras el evento.

Primeramente se analizaron cuantitativamente las aportaciones a la red social y el resultado arrojado fue que la participación descendió significativamente. Después, se procedió a un análisis cualitativo de las mismas. Durante el período en el cual los usuarios ganaban puntos, los comentarios se dividían entre aquellos que posteaban los usuarios en sus propios perfiles, a modo de saludo –mensajes simples como "*hola*" o "*¿Qué tal va el trabajo*"-, y aquellos que eran posteados en las publicaciones ajenas –mensajes como "*Bonita foto. ¿Estás disponible para quedar un día y hablar?*"-. Tras la eliminación de los puntos, descendieron todo tipo de comentarios, pero en mayor medida aquellos que eran de la naturaleza de los primeros expuestos.

Se estudiaron después, con más detalle, las dos poblaciones con más usuarios activos en la red de la empresa: Estados Unidos, país de origen de esta; y la India, donde tienen externalizado parte de su servicio técnico.

Se observó que en ambos segmentos la participación se redujo tras la eliminación de los puntos, pero había una tendencia por parte de los usuarios de la India que no era compartida en los estadounidenses: la mayoría de las interacciones de los usuarios de la India eran con publicaciones de usuarios de Estados Unidos. Esto responde a la motivación intrínseca de querer empaparse de la cultura del país de origen de la empresa para la que se trabaja y, además, poder labrarse una reputación.

Por tanto, como afirma Groh [6], el uso de recompensas es prescindible si el juego ofrece al usuario una motivación para interactuar. Dicha motivación se consigue a través de una conexión entre lo que el juego ofrece y los intereses del usuario. Y además, no solo es necesario que se atrape a un jugador por sus intereses sino que estos han de ser comunes con los del resto de usuarios que conforman la comunidad con la que va a interactuar. Así pueden producirse situaciones en las que un usuario comparte un logro y dicho hecho es relevante para el resto de la comunidad, mientras que en una comunidad heterogénea dicho logro no significaría nada [6].

Otro hecho estudiado por varios autores apunta a que, con la edad, la motivación intrínseca se reduce mientras que la extrínseca aumenta. Sin embargo, no existe un consenso acerca de en qué medida evoluciona cada una; una parte de los autores afirma que, con la edad, los estudiantes dejan de divertirse durante el proceso de aprendizaje y, además, los incentivos para mantener a estos estudiantes concentrados en sus estudios no compensan esta falta de motivación. Otros autores opinan que la motivación extrínseca disminuye pues los estudiantes toman conciencia de su responsabilidad y no requieren de una figura autoritaria que los mantenga centrados [10].

2.2. Gamificación en la educación

En esta sección se aportan varios ejemplos de gamificación cercanos a lo propuesto en el trabajo.

Just Press Play

Uno de los ejemplos de mayor magnitud es el proyecto *Just Press Play* [2, 6, 13, 16]. A 400 alumnos del campus del Instituto de Tecnología de Rochester (RIT) se les ofreció participar en un juego que consistía en adquirir insignias virtuales a través de retos completados en la vida real. Estos retos fomentaban la vida universitaria y el compañerismo, pues eran de la índole de "mantén una reunión con alguno de tus profesores" o "asiste a una de las conferencias realizadas en el campus".

No todos los retos estaban orientados a acometer acciones íntimamente relacionadas con actividades académicas, sino que también buscaban que los alumnos del campus estrechasen sus relaciones. Por ejemplo, otro de los retos proponía que trece alumnos se reuniesen en un restaurante para comer, y se sacasen una foto para después compartirla en la plataforma. Y no solo eran las relaciones entre alumnos las que se buscaba potenciar; otro reto, "*For the Lawls*" consistía en hacer reír a Elizabeth Lawley, profesora de la universidad y una de las propulsoras del proyecto [16].

Ciertos retos produjeron una mejora notoria en las calificaciones de los alumnos. Uno en concreto, "*Undying: The return*", premiaba a todos los alumnos de un grupo si aprobaba una asignatura el 90% de la clase. Esto generó una ola de compañerismo reflejada en sesiones de estudio grupales fuera del horario lectivo y más implicación por parte de los alumnos más aventajados con aquellos que llevaban peor la materia. Finalmente, el 92% de los alumnos aprobaron la asignatura. Y ese hecho no fue la única

consecuencia positiva derivada del reto; los alumnos pidieron continuar haciendo sesiones de estudio para el resto de las asignaturas del curso [2].

Por último, menciona un hecho que hay que tener en cuenta a la hora de diseñar el sistema, y es que la naturaleza del juego es voluntaria. Esto conduce a otra de las aserciones de Lawley: "Uno de los riesgos de ponerlo en funcionamiento es hacer que parezca más trabajo, y si parece más trabajo los estudiantes no van a hacerlo. La mayoría de ellos no tienen apenas tiempo suficiente para completarlos […] se trata más de reconocer las actividades han realizado que de intentar que hagan un puñado de cosas nuevas".

The Multiplayer Classroom

Lee Sheldon estableció para su asignatura *Theory and Practice of Game Design*, en la Universidad de Indiana, un sistema que llamó *Multiplayer Classroom* y que consistía en transformar el sistema de calificaciones y de exámenes tradicional por un sistema ambientado en un videojuego de rol multijugador masivo.

Su metodología ha sido utilizada por varios docentes y en el libro *The Multiplayer Classroom: Designing Coursework as a Game* [13] ha recogido la experiencia de dichas gamificaciones, exponiendo los resultados y analizando la metodología aplicada.

La primera técnica que aplicó fue gamificar el plan de estudios. Sustituyó las competencias y los resultados de aprendizaje por tareas a completar a lo largo del curso, utilizando un lenguaje basado en los videojuegos. Por ejemplo, en vez de utilizar el verbo redactar para referirse a la composición de un trabajo, utilizaba el verbo *craft* (crear, masivamente utilizado en los videojuegos). Además, a cada tarea le otorgaba una puntuación basada en puntos de experiencia.

Sustituyó el tradicional sistema de puntuación basado en notas (en EEUU las calificaciones son letras que van desde la F hasta la A) por un sistema de niveles. Sheldon estableció doce niveles que iban siendo alcanzados según los alumnos completaban las tareas expuestas en la guía docente y recibían por ello puntos de experiencia. Por tanto, el sistema no se enfocaba en las calificaciones sino en el progreso de los estudiantes. Para adaptarlo aún más al sistema de un videojuego, los puntos de experiencia necesarios para alcanza el siguiente nivel aumentaban conforme lo hacían los propios niveles. Sin embargo, Sheldon advirtió que, de alguna manera, tenía que compensar el acusado incremento de la dificultad que generaba esta implementación de niveles; decidió por tanto otorgar cantidades de puntos de experiencia mayores conforme la tarea a completar era más difícil.

Un último apunte interesante es la manera con la que proponía actividades extras. El término *farmear* es un anglicismo que se puede traducir, en palabras de Sheldon, como "matar al mismo enemigo, usualmente de un nivel muy inferior al del jugador, una y otra vez como la manera más eficiente de ganar experiencia y botín". Basándonos en la jerga instanciada por el autor, matar a un enemigo es completar una tarea. Por tanto,

ciertas actividades extras realizadas de manera continua suponían un aumento de los puntos de experiencia de los alumnos y, por ende, en subidas de nivel. Una de las actividades extra consistía en encontrar fallos en los libros de texto que proponía [13].

Fantasy Geopolitics

En el año 2009, en la *North Lakes Academy Charter School* de Forest Lake, Minnesota, fue desarrollada una herramienta llamada *Fantasy Geopolitics.*

Esta herramienta estaba dirigida a los alumnos de noveno grado de la escuela, para la asignatura de ciencias sociales. Se les propuso formar grupos y, mediante un *draft*, cada grupo escogía tres países. Su tarea consistía en mostrar en clase todas las noticias que encontrasen en las que sus países eran mencionados, y por cada una recibían un punto. Además, también podían comerciar con dichos países, intercambiándolos con otros grupos.

La idea era acostumbrar a los estudiantes a que leyesen noticias, estuviesen al día de cuanto acaecía en el mundo y además motivarlos a estudiar la asignatura.

La herramienta gozó de un éxito tal que posteriormente fue utilizada en un curso cívico de política exterior, y en otros cursos de la propia escuela. En 2014, recaudó $12,706 a través de *Kickstarter*, plataforma de financiamientos para proyectos creativos [18].

Actualmente cuenta con tres juegos: el original *Geopolitics*; *Politics*, que es igual que el anterior pero con los estados de Estados Unidos; y *Olympic Challenge*, que está inspirado en los juegos olímpicos y ayuda a aprender la cultura de los países participantes a través de una mecánica de predecir cuántas medallas ganará cada uno [23].

Passport

La Universidad de Purdue desarrolló una plataforma online de insignias, creadas y diseñadas por profesores para ser entregadas a los alumnos [18, 27]. El valor de las insignias es explicado por Kyle Bowen, director de informática de la *Informatic Technology at Purdue*: “Las insignias se han convertido en una manera de reconocer el aprendizaje en todas sus formas. *Passport* provee una plataforma para cualquiera que quiera entregar credenciales de aprendizaje.” La propia plataforma está construida para permitir a los educadores construir desde cero la insignia que pretenden emitir, “desde la creación del reto hasta la creación de la propia imagen de la insignia, y después una manera de mostrar las insignias ganadas.”

Por otro lado, Gerry McCartney, vicepresidente de Tecnologías de la Información de Purdue, presenta otra ventaja en relación con el uso de las insignias: “además de las clases teóricas y de los deberes, los estudiantes pasan tiempo en laboratorios y haciendo trabajo de campo; tiempo invertido en proyecto de servicio e internados; y experiencia que recogen de las asociaciones estudiantiles. La aplicación *Passport* proporcionará otra

manera de reconocer y validar las habilidades de los estudiantes a las facultades interesadas y a los asesores".

Un ejemplo ilustrativo es haber participado en una organización estudiantil, proporcionando ayuda y servicio a los estudiantes. Dicha labor no es reconocida de manera oficial en las calificaciones de la facultad y, sin embargo, es una labor que brinda de gran experiencia a aquellos que la realizan. Este sistema de insignias sí reconocerá tangiblemente el trabajo efectuado [19].

Nació en 2012, para entregar insignias a los estudiantes que completasen un MOOC en nanotecnología, aparte de otros logros relacionados con otras materias y departamentos, todo dentro del marco de actividades de la Universidad de Purdue. Uno de los profesores cuya participación fue clave a la hora de desarrollar la herramienta, Bill Watson, afirma que "las insignias ayudan a los profesores a alentar a los estudiantes a demostrar cómo han alcanzado objetivos de aprendizaje muy específicos a través de un rendimiento real".

El sistema se integra con *Mozilla Open Badges Infraestructure* [26], un sistema que permite incluir a las insignias metadatos tales como quién emitió la insignia, cómo y cuándo se consiguió.

Actualmente sirve como portfolio online, permitiendo a los usuarios demostrar sus capacidades y competencias. Además, estas insignias pueden ser exportadas a redes sociales como Facebook y LinkedIn. Es utilizada por instituciones tales como el MIT, la Universidad de California y la NASA.

Speculative Design

El Doctor Carman Neustaedter, profesor ayudante en la *School of Interactive Arts & Technology* de la Universidad Simon Fraser, implantó un mecanismo de gamificación a su asignatura, *Speculative Design* [14].

Creó una página web para su curso, donde colocó una tabla de resultados. En dicha tabla aparecían los 70 alumnos de la asignatura, ordenados por las calificaciones que iban obteniendo a lo largo del curso. Para incrementar la motivación de los estudiantes, estableció un sistema de niveles que se iban alcanzando según la suma total de sus puntuaciones aumentaba. Además, también asignaba a cada alumno un rango definido mediante un título que describía las aptitudes de los mismos; por ejemplo, algunos títulos rezaban "*Junior Artist*", "*Artistic Intern*" y "*Grand Master Speculative Designer*". Esta mecánica responde a la premisa llamada *Storytelling*. En palabras de Kapp, "la gente puede aprender hechos mejor cuando los hechos están embebidos en una historia y no en un listado" [7].

Semanalmente proponía acertijos, una mecánica similar a la propuesta en *Bolotweet*. Neustaedter aplicó un concepto conocido como *Just in Time Teaching* [4, 13]. Dicho concepto se define como recopilar aspectos de la asignatura al comienzo de la clase, si el profesor considera oportuno abordarlos. Esta mecánica permite variar la planificación

de la clase si se considera necesario hacer más hincapié en algún concepto teórico. Neustaedter mostraba en sus clases respuestas a los acertijos que creía representativas de un fallo generalizado en los estudiantes, para corregirlo y aclarar ideas a sus alumnos.

En la web mencionada también se mostraban los hitos a alcanzar a lo largo del curso, junto a unos puntos de experiencia. Se mostraban los puntos máximos alcanzables por hito, y los que el alumno había obtenido por su desempeño en dicha actividad. La tabla es similar a la que puede encontrarse en el Campus Virtual, donde aparecen las tareas y exámenes realizados junto a la nota obtenida y la puntuación máxima alcanzable.

Neustaedter eliminó una mecánica que incluía interacción con el mundo real. Inventó la historia de un secuestro y, por cada nivel de experiencia alcanzado por un alumno, le entregaba una palabra clave que, introducida en la web del curso, revelaba una parte de una fotografía. Dicha fotografía mostraba un lugar del campus de la facultad; y en dicho lugar, se escondía la última palabra clave. Esta última palabra revelaba la identidad del secuestrador. Sin embargo, la mecánica fue eliminada porque Neustaedter no encontraba relación alguna entre la tarea de buscar el lugar del campus y revelar la identidad del secuestrador con un aprendizaje real de los conceptos teóricos de su asignatura. Decidió, por tanto, transformar los puntos de experiencia en calificaciones reales.

Uso de Kahoot! en asignaturas de la facultad

Kahoot! es uno de los ejemplos de gamificación implementados en la facultad [25].

Se trata de una herramienta online que permite a los educadores crear test (o utilizar los que se encuentran disponibles en el repositorio) que los alumnos pueden contestar interactivamente, pudiendo ver en tiempo real las respuestas a las preguntas y la clasificación de los participantes, que reciben puntos por contestar correctamente a las preguntas y por hacerlo en más o menos tiempo.

Esta plataforma fue utilizada en alguna asignatura, de tal manera que el profesor, al inicio de la clase, proponía realizar un test en el que las preguntas correspondían a la parte del temario de la asignatura que había sido impartido hasta el momento. Los alumnos que quedaban en las tres primeras posiciones recibieron puntuación extra en sus calificaciones finales.

Ciberguerra en Redes y Seguridad e insignias

La asignatura de Redes y Seguridad tiene un importante componente práctico. Semanalmente, los alumnos han de completar tareas y ejercicios durante las dos horas de laboratorio, además de otros retos que son lanzados a través del campus virtual y que tienen que ver con el descifrado de mensajes, investigación y resolución de acertijos.

Para motivar a los alumnos a participar activamente en los retos y a completar las prácticas de laboratorio, en el curso 2016/2017 se instauró una competición llamada

"Ciberguerra". Consistía en una competición por puntos que eran asignados a cada uno de los dos grupos de la asignatura, y había dos maneras de conseguirlos.

La primera vía para que la clase obtuviese un punto era completar un reto. Los retos eran completados cuando se hallaba la solución al acertijo que se proponía. En el foro del campus virtual había una sección destinada a estos retos, contando con un hilo para cada uno. En dicho hilo podían participar todos los estudiantes de los dos grupos, y el que quería aportar su granito de arena posteaba un mensaje con la información que había recabado o lo que pensaba que podía ser la solución. Cuando finalmente un alumno daba con la respuesta correcta, se asignaba un punto al grupo al que pertenecía.

La segunda vía era resolver en menos tiempo la práctica semanal. Los profesores del laboratorio anotaban el tiempo que había tardado la primera pareja de cada grupo en completar la práctica. El punto era asignado al grupo cuya pareja había sido más rápida.

La Ciberguerra contaba con varios mecanismos inspirados en los juegos. Por ejemplo, se conseguía que la inmersión fuese un tanto mayor pudiendo la clase elegir bajo qué nombre querían participar en la guerra. También se propuso que alguien diseñase un logo que podía ser avistado en la clasificación que era mostrada en la página principal del campus de la asignatura.

Una tabla de clasificación es una mecánica que sustenta un principio definido por Kapp como *Rapid feedback*. Afirma que los estudiantes han de ser conscientes en todo momento de cuál es su estado dentro del juego; mediante una retroalimentación constante y dirigida, su progreso es mayor. Una de las mejores maneras para acometer esta tarea es mediante un marcador, donde un estudiante puede observar si su rendimiento está siendo el adecuado o no [7, 8, 14].

Otro elemento clave en la gamificación es el uso de insignias, que manifiestan que un alumno ha completado o participado en un reto. En el ya mencionado *Just Press Play*, los alumnos que completaban los retos recibían al momento una insignia virtual, que era dispuesta en su perfil junto a las otras insignias que el alumno hubiera recibido con anterioridad [16]. Las insignias también se establecieron en el marco de la asignatura de Redes y Seguridad. Todo alumno que participaba en alguno de los retos lanzados, fuese él o no quien diese con la respuesta correcta, recibía una insignia que demostraba que había participado en el reto. Dichas insignias se mostraban en los perfiles de los alumnos en el Campus Virtual.

2.3. La actualidad de *Robocode* y *Bolotweet*

Bolotweet

Bolotweet se sustenta en la plataforma de microblogueo de código abierto *Statusnet*. Esta plataforma ofrece un sistema de plugins que permite dotar de funcionalidad extra a la herramienta. *Bolotweet* ha sido desarrollado por profesores de la Facultad de Informática de la Universidad Complutense de Madrid, que mediante el desarrollo de

plugins han enfocado la plataforma al método docente propuesto en su trabajo de investigación [17].

Actualmente, *Bolotweet* se basa en mensajes enviados por los estudiantes y que son corregidos por el profesor, que puede otorgar una nota del 0 al 3. Adicionalmente, los mensajes pueden ser respondidos tanto por el profesor como por otro alumno. Esta mecánica permite, por ejemplo, que el profesor indique al alumno en qué punto su mensaje no es correcto.

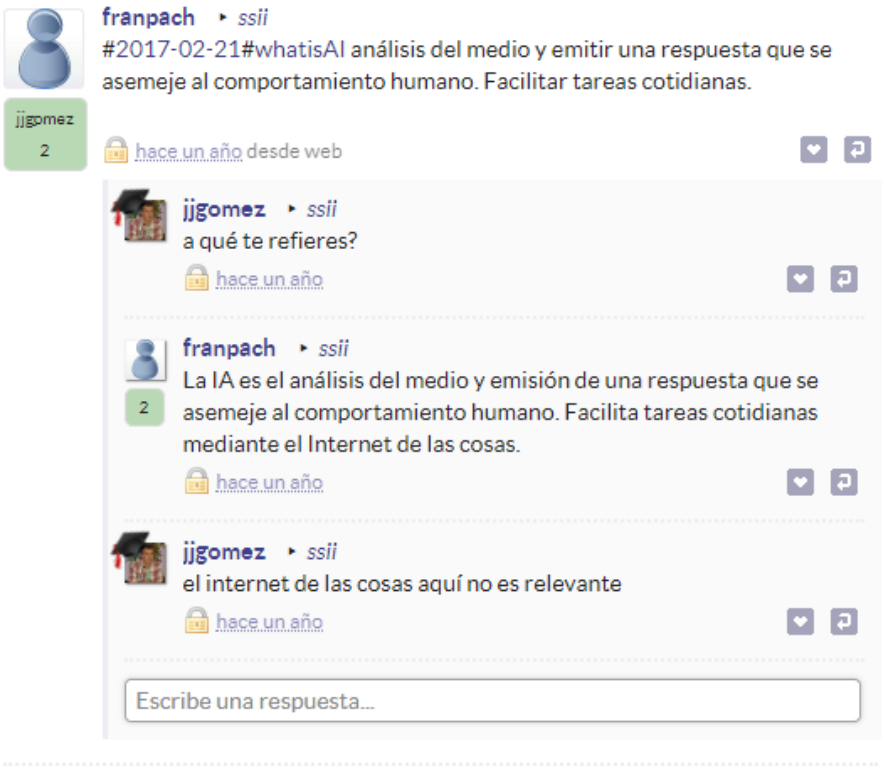


Ilustración 1- Ejemplo de interacción en Bolotweet entre alumno y profesor

El profesor de un grupo posee el rol *grader*. Dicho rol le permite poder puntuar los mensajes que son enviados al grupo por parte de los estudiantes. Un profesor puede ser *grader* de aquellos grupos de los que es profesor.

Además, *Bolotweet* ofrece la posibilidad de descargar, en forma de apuntes, los mensajes del sitio. Puede filtrarse dicha descarga por asignatura y, dentro de ella, descargarlos automáticamente o filtrando por *hashtag*.

Robocode

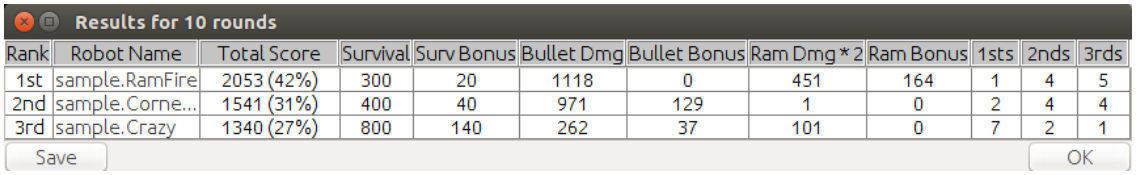
Se ha escogido el juego de código abierto *Robocode* [28] como herramienta de gamificación. Este juego ejecuta batallas entre robots controlados por una inteligencia artificial totalmente programable por el usuario. Tanto la batalla como los robots están codificados en lenguaje Java.

A continuación van a ser explicadas las características más importantes y relevantes de *Robocode*, en relación al funcionamiento de las batallas.

Una partida es ejecutada en un campo de batalla de dimensiones definidas a la hora del lanzamiento de la misma. El tipo de batalla más común, y el que se proporcionará en el sistema, es una batalla de "todos contra todos". Una batalla puede tener el número de rondas que se desee, y cada ronda finaliza cuando solo queda un robot con vida.

Un robot puede atacar a otro de dos maneras: disparándolo o arrollándolo. En el capítulo 4 se especificará cuánto daño produce un ataque. Por otra parte, un robot puede perder vida al ser atacado por otro robot, o al chocar con los muros que delimitan el campo de batalla.

Una vez la batalla finaliza, se recogen los resultados:


Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	sample.RamFire	2053 (42%)	300	20	1118	0	451	164	1	4	5
2nd	sample.Corne...	1541 (31%)	400	40	971	129	1	0	2	4	4
3rd	sample.Crazy	1340 (27%)	800	140	262	37	101	0	7	2	1

Save OK

Ilustración 2 – Resultados de una batalla

- ***Total score:*** La suma de todas las puntuaciones que a continuación van a explicarse.
- ***Survival:*** Un robot recibe 50 puntos por cada vez que otro robot muere y él sigue con vida.
- ***Survival Bonus:*** Un robot recibe 10 puntos por cada robot eliminado durante una ronda si consigue ser el último en pie.
- ***Bullet damage:*** Un robot recibe un punto por cada punto de daño que hace a los enemigos disparándolos.
- ***Bullet bonus:*** Un robot recibe un 20% adicional del daño producido a un enemigo al que ha eliminado disparándolo.
- ***Ram damage:*** Un robot recibe dos puntos por cada punto de daño que hace a los enemigos arrollándolos.
- ***Ram bonus:*** Un robot recibe un 30% adicional del daño producido a un enemigo que ha eliminado arrollándolo.
- ***1sts, 2nds, 3rds:*** Número de veces que el robot ha quedado en primer, segundo y tercer lugar.

Robocode ordena a los participantes según su puntuación total, asignando el primer puesto al que mayor puntuación ha obtenido.

2.4. Crítica al estado del arte

El objetivo común de las propuestas descritas era alentar a los alumnos a mantener una participación activa en las actividades de clase, algunas de ellas específicamente diseñadas para aumentar la motivación de estos alumnos.

En algunos casos, sin embargo, las actividades propuestas diluían la verdadera dificultad que entraña dominar un concepto teórico. La gamificación ha de estar

totalmente separada de las evaluaciones formales, y además tampoco ha de verse de forma aislada a otras herramientas y métodos [5]. Por tanto, nunca debe sustituir a la metodología ya instaurada.

En el caso de *Fantasy Geopolitics*, toda actividad relacionada con la asignatura en la que fue aplicado se sustituyó por el juego descrito. ¿Es dicho juego suficiente para que los alumnos aprendan todo lo que se espera que aprendan sobre geopolítica? No es posible conocer hasta qué puntos los alumnos aprendieron los conceptos que la asignatura proponía, pero el hecho de utilizar únicamente la gamificación es una práctica rechazada por alguno de los autores [5].

Por otro lado, a la hora de gamificar una asignatura, el juego establecido no ha de simplificarse en exceso, pues puede provocar que los estudiantes dejen de mostrar interés en la asignatura [2]. Los ejemplos mencionados a lo largo de la memoria, sin embargo, sí abordan correctamente este punto, pues en todos los juegos no solo se aplicaban mecánicas extraídas de los juegos sino que, además, todo se envolvía en un contexto desarrollado con la intención de atraer a los alumnos. Solo en el caso del uso de *Kahoot!* se simplifica la utilización de un juego; las partidas eran ejecutadas de manera esporádica y no formaban parte de la metodología continua de la asignatura. Sí se entiende la simplificación de la gamificación en este caso, pues su uso estaba enfocado al repaso de los conceptos estudiados y no en el aprendizaje de nuevos.

Otros ejemplos, como *Just Press Play*, pretendían abarcar muchos más aspectos más allá del aprendizaje; fomentaban la vida en el campus universitario, el compañerismo, etc. En este caso, era necesaria una colaboración entre todos las facultades, departamentos y entidades de la universidad; una complejidad logística no abarcable en este proyecto.

En relación con la complejidad de los proyectos, tanto este último ejemplo mencionado como otros no han sido llevados a cabo todos los años. Esto se debe a que es necesario realizar un despliegue de medios que no siempre es posible efectuar. Sin embargo, este proyecto no requiere una logística complicada, pues se basa en un código ejecutable y en una actualización de datos automatizada, amén de estar sustentando en una plataforma como *Bolotweet*, que lleva varios años activa y su fiabilidad está más que probada.

De estos ejemplos estudiados se obtienen una serie de principios aplicables a la propuesta presentada:

- Se hará uso de un sistema de puntuaciones que ordene a los estudiantes según su desempeño en la plataforma.
- Los mensajes enviados a *Bolotweet* y su posterior calificación no sustituirán a los métodos comunes de evaluación: prácticas y exámenes.
- La participación en el sistema no requerirá tiempo extra para su uso, puesto que las batallas serán ejecutadas durante las clases teóricas y también se dedicará tiempo de las mismas para la resolución de las tareas.

- La automatización de la actualización de resultados simplifica la puesta en funcionamiento del sistema, reduciendo la complejidad de su uso.
- En general, se busca un aumento de la motivación intrínseca de los estudiantes –que participen en la plataforma porque sienten interés en las batallas y les resulta motivador tener un robot competitivo-, mientras que la extrínseca se vea devaluada –participar por el simple hecho de ver mejorada su nota final-.

2.5. Conclusiones

Los autores que han desarrollado ideas para gamificar sus asignaturas, y posteriormente las han implementado, han observado resultados positivos en el desempeño de sus alumnos.

Los trabajos revisados también concuerdan en que no solo basta con implementar metodologías que incluyan elementos de los videojuegos; es necesario medir cómo implementar estas mecánicas, qué peso final tendrán en la calificación de la asignatura y qué recompensas recibirán los alumnos por completar satisfactoriamente los desafíos propuestos.

Muchos de los ejemplos mencionados son proyectos fruto de un largo proceso de desarrollo, cuya parte más importante es decidir los aspectos que se acaban de mencionar. Muchos de los autores han pensado, largo y tendido, cómo implementar las mecánicas de manera que los alumnos no vean su motivación intrínseca devaluada en virtud de recompensas fáciles de obtener, desafíos repetitivos y contextos que no casen con sus gustos e intereses [1, 5, 6, 7, 8, 12].

3. Requisitos de integración de *Bolotweet* y *Robocode*

Gamificar el uso de *Bolotweet* busca fomentar su uso y hacerlo más ameno, sin alterar o reducir las actividades que hacen que los estudiantes trabajen los conceptos teóricos de las asignaturas en las que es utilizado.

La propuesta principal del proyecto es transformar el desempeño de los alumnos en *Bolotweet* en mejoras para un avatar en el juego multijugador *Robocode*. Los avatares de los alumnos de un grupo interactuarán dentro del juego, donde se desarrollarán partidas entre estos.

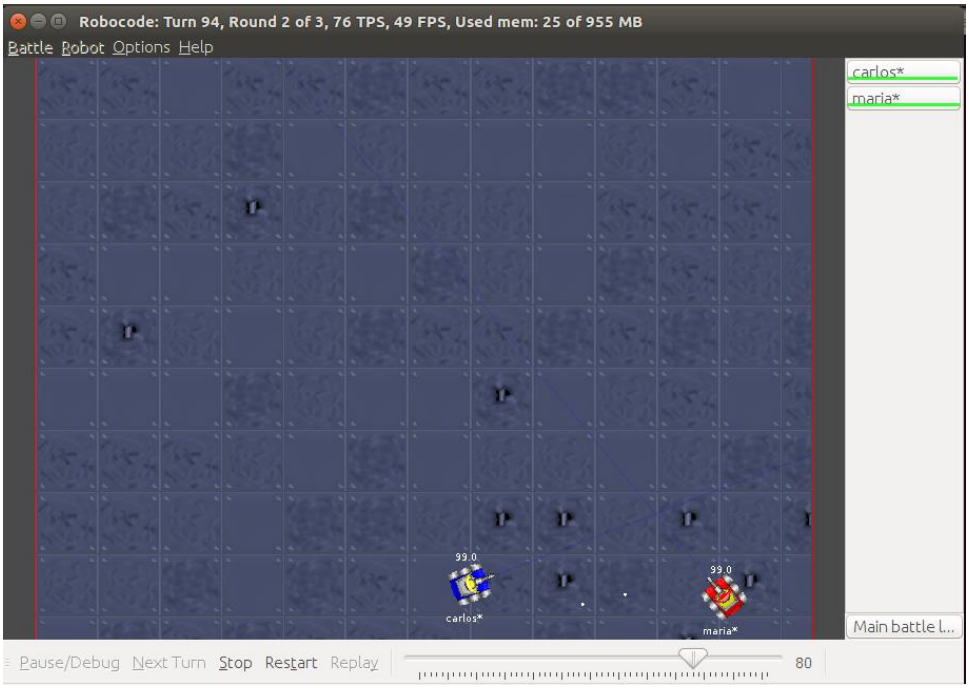


Ilustración 3 – Interfaz de una batalla ejecutada en Robocode

Estas batallas pueden ser de dos tipos:

- *Roborumble*: todos los robots pertenecientes a un grupo batallan a la vez.
- Batalla en la que los robots son elegidos manualmente por el profesor. El número de robots a participar tiene que ser mínimo dos y como máximo, todos los robots del grupo.

Los resultados de estas batallas serán monitorizados y la acumulación de puntos y rondas ganadas podrá verse desde la herramienta *Bolotweet*. Así, en todo momento los estudiantes pueden consultar su posición en el ranking de su grupo correspondiente.

Ilustración 4 – Vista de la clasificación de varios grupos

Mostrar la clasificación en todo momento tiene dos propósitos: alentar a los que están arriba para mantener su posición siguiendo completando las tareas y motivar a los estudiantes que ocupan posiciones más bajas a remontar posiciones mejorando sus robots.

Como se observa en la ilustración 4, si un profesor o alumno pertenece a varios grupos, en la misma página se le mostrarán las clasificaciones de todos los grupos a los que pertenezca.

Las calificaciones obtenidas por los estudiantes en *Bolotweet* serán registradas y el sistema calculará la media de las mismas en un periodo de tiempo indicado por el profesor de la asignatura.

El sistema se basará en un sistema de niveles, asignados a los alumnos según qué media obtengan en las calificaciones de sus mensajes enviados a *Bolotweet.* En función de la magnitud de dicha media, el nivel del robot será mayor o menor. El número de niveles desarrollados es tres, puesto que pueden darse tres principales comportamientos por parte de los alumnos.

El robot de nivel 1 es el más simple, que se asignará a aquellos alumnos que no trabajen en *Bolotweet* o a aquellos cuyos mensajes no cumplan con un mínimo nivel de exigencia. Tiene un desempeño pobre y pocas posibilidades de ganar una partida. Por tanto, un alumno que no trabaje no obtendrá una buena recompensa en el juego.

El robot de nivel 2 mejora en parte al del nivel 1, siendo algo más competitivo. Está pensado para aquellos alumnos que sí completan las tareas, pero estas no suelen alcanzar el grado de corrección que exige el profesor.

El robot de nivel 3 es ampliamente superior a los otros dos robots, y está pensado para los estudiantes que completan todas las tareas y, además, lo hacen con mensajes cuyo contenido satisface los criterios del profesor.

A continuación se muestra una tabla que indica la relación de asignación de niveles en función de la media obtenida en *Bolotweet*:

Rango de media de puntuaciones obtenidas en *Bolotweet*	Nivel asignado en *Robocode*
[0-1]	1
(1-2]	2
(2-3]	3

Tabla 1- Relación entre niveles asignados y medias obtenidas

El comportamiento de los tres niveles, explicado en lenguaje natural, es el siguiente:

Nivel 1

El robot se dedica a dar vueltas alrededor del campo de batalla, evitando colisionar con los bordes. Dispara únicamente cuando su radar detecta un enemigo, pero al no apuntar deliberadamente, errará la mayoría de los disparos, pues para cuando la bala haya llegado a la posición en la que se detectó al rival, este ya se habrá desplazado.

Cuando colisiona con otro robot, huye de él, desplazándose hacia al lado contrario al que ha impactado con el enemigo. Cuando recibe el impacto de una bala actúa de la misma manera.

Nivel 2

Este robot mantiene la misma trayectoria que el robot de nivel 1. Sin embargo, cuando impacta con un enemigo, en vez de huir, trata de arrollarlo. Con esto, tiene más posibilidades de seguir con vida, pues como se explicará en el capítulo 4, arrollar a otro enemigo proporciona puntos de energía.

Nivel 3

El robot de mayor nivel. Cuando escanea un robot enemigo, dispara. Además, mide la potencia del disparo; las balas de menor potencia viajan más rápido. Por tanto, cuanto más lejos se encuentre del robot enemigo, disparará balas más livianas, pues son más rápidas y, además, le harán perder menos vida. Es una estrategia inteligente, pues hay que tener en cuenta que, al ser mayor distancia, tiene menos probabilidades de acertar. Se compensa así la potencial pérdida de energía debido al constante disparo de balas erradas.

Además, trata de mantener en su radar a un enemigo localizado. Conociendo la dirección a la que se mueve el enemigo, modifica su propia trayectoria para continuar acechándolo y, además, mueve su cañón para que apunte al objetivo.

Por tanto, no se dedica a moverse en círculos alrededor del mapa, sino que persigue a sus rivales y los dispara. Además, cuando choca con otro, también trata de arrollarlo, al igual que el robot de nivel 2.

Finalmente, el funcionamiento del sistema puede resumirse en tres mecánicas:

- El profesor propone tareas a ser completadas por los estudiantes. Estas tareas consisten en la escritura de mensajes que son calificados por el profesor con una puntuación comprendida entre el 0 y el 3. El contenido de los mensajes ha de estar relacionado con los contenidos teóricos que el profesor indique.
- Se calcula la media de las calificaciones, que determina el nivel de inteligencia de un tanque robot. Cada estudiante tiene asociado un robot.
- Los tanques robots participan en batallas, lanzadas por el profesor durante las clases teóricas, de las cuales se registran los resultados que, siendo acumulados, mantienen una clasificación de estudiantes.

3.1. Directrices generales

Es necesario establecer una serie de normas para regular el uso de la herramienta.

- Una batalla solo puede ser ejecutada por el profesor del grupo.
- El profesor puede decidir si implementar o no el uso de este sistema.
- Cada grupo cuenta con una clasificación de alumnos. Esta clasificación será calculada en base a los puntos totales obtenidos durante las batallas. Como se ha podido observar en el apartado 2.3, esta puntuación obtenida por un usuario en una partida es la suma de sus acciones durante una batalla.
- El desempeño de un estudiante se calcula como la media de los *grades* recibidos en sus mensajes enviados en los últimos *n* días únicamente al grupo sobre el que se ejecutan las batallas. El número *n* de días es escogido por el profesor.
- El profesor puede variar en cualquier momento el número de días y las medias de los estudiantes serán recalculadas.
- Los alumnos podrán elegir los colores que tendrá su robot. Estos colores pueden ser aplicados a tres partes del robot: cuerpo, radar y cañón.

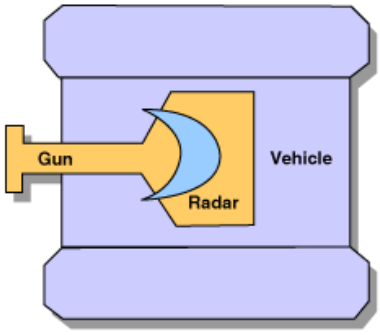


Ilustración 5 – Partes de un robot a las que se les puede aplicar un color

- También serán elegibles las dimensiones del campo de batalla y el número de rondas que tendrá la partida. Además, el profesor podrá guardar esta configuración permanentemente, y cambiarla cuando lo necesite.

- Cada alumno tendrá su robot. Si un alumno pertenece a varios grupos, en el sistema solo se registrará un robot. El nivel del mismo se actualizará en el momento en el que un profesor ejecute el sistema, ajustándose así dicho nivel al desempeño del estudiante en ese grupo. Los colores del robot serán los mismos para cada grupo.

3.2. Directrices para profesores

El profesor de un grupo es el encargado de ejecutar las batallas, así como de definir los parámetros de las mismas.

Es el único usuario con potestad para ejecutar el script *bolocode.sh*, que soporta la funcionalidad total del sistema.

En el apéndice A de este documento puede consultarse un manual de uso para profesor completo, en el que se detallan las acciones que se han mencionado con anterioridad.

3.3. Directrices para alumnos

El trabajo de los alumnos dentro de *Bolotweet* no varía respecto a la versión anterior; su único cometido es completar las tareas propuestas por el profesor.

Sin embargo, en relación al sistema han de tener en cuenta que cuando un profesor registra a los alumnos de su grupo, lo hace indicando su nombre y los colores que los propios estudiantes han elegido. Si un alumno es registrado en otro grupo estando ya registrado en uno, y le indica al profesor una serie de colores, esos colores se aplicarán a su robot para todo grupo al que pertenezca.

3.4. Descripción de un escenario de uso

En este apartado va a representarse un escenario para mostrar el funcionamiento global del sistema.

Para ello, se va a suponer que la rutina establecida por el profesor se ajusta a lo sugerido en el desarrollo de *Bolotweet 2.0*. [17]. Esto es, que de 50 minutos de clase, 10 se dedican al uso de la plataforma. O lo que es lo mismo: de cada 50 minutos de clase se propone una tarea a ser completada por los alumnos.

Suponiendo su uso en una asignatura de cuatro horas semanales divididas en dos días con dos clases seguidas, el profesor propone una tarea para ser completada durante el descanso entre las dos clases, y otra para el final de la segunda.

En cuanto a la ejecución de las batallas, al final de la segunda clase, además de proponer la tarea correspondiente, ejecutará varios *roborumbles* de tres rondas cada uno. En la sección 5 de la memoria se proponen alternativas, tales como torneos. Sin embargo, para esta demostración solo se ejecutarán *roborumbles*.

Además, esta monitorización va a realizarse con el curso ya empezado y habiéndose ejecutado varias batallas, una vez se han estabilizado las tendencias de uso de los estudiantes.

El profesor del grupo EDA propone la una tarea, que posteriormente es completada por los alumnos en el plazo de tiempo asignado para ello.

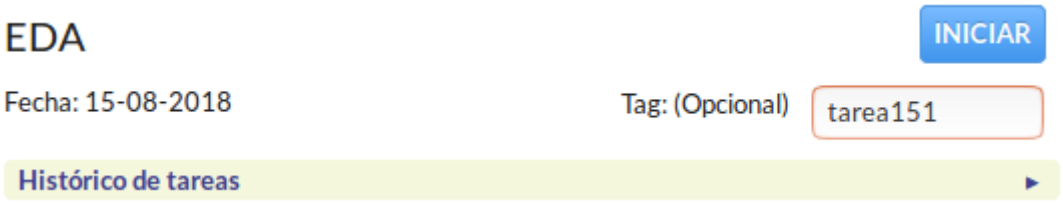


Ilustración 6 – Formulario para el profesor para lanzar una tarea

Estas tareas son completadas mediante el siguiente formulario:

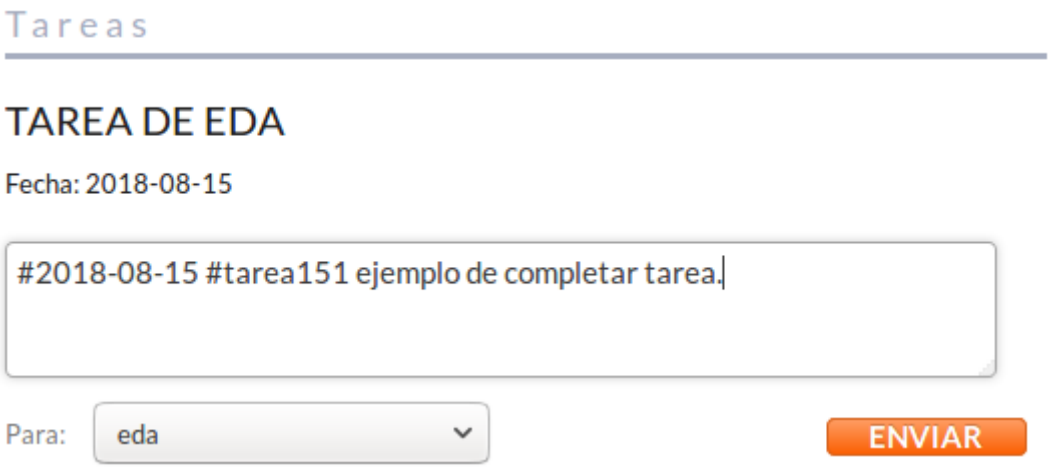


Ilustración 7 – Formulario para completar una tarea

El profesor corrige los sendos mensajes que van siendo enviados al grupo.

Ilustración 8 – Timeline del grupo donde el profesor puede corregir las tareas

Cuando acaba la segunda clase, de nuevo se propone una segunda tarea y, además, se ejecutan varios *roborumbles*.

Como puede apreciarse en la ilustración 10, el sistema, al iniciarse, actualiza la media de los alumnos de los mensajes enviados últimos *n* días. Este número de días es configurable en el archivo de configuración *roboconfig.cfg* o dentro de la propia aplicación, como se aprecia en la ilustración 9.

```
--------------------------------------------------------------------------------
Elige una de las siguientes opciones:
1.-Modificar opciones de notas/grupo.          2.-Modificar/crear robots.
3.-Modificar opciones de batalla.              4.-Generar batalla.
0.-Salir.
--------------------------------------------------------------------------------
Opción:
1
--------------------------------------------------------------------------------
Elige una de las siguientes acciones:
1.-Modificar grupo de trabajo.          2.-Modificar periodo de medias.
0.-Volver atrás.
--------------------------------------------------------------------------------
```

Ilustración 9 – Diálogo para modificar el periodo de días para el cálculo de las medias

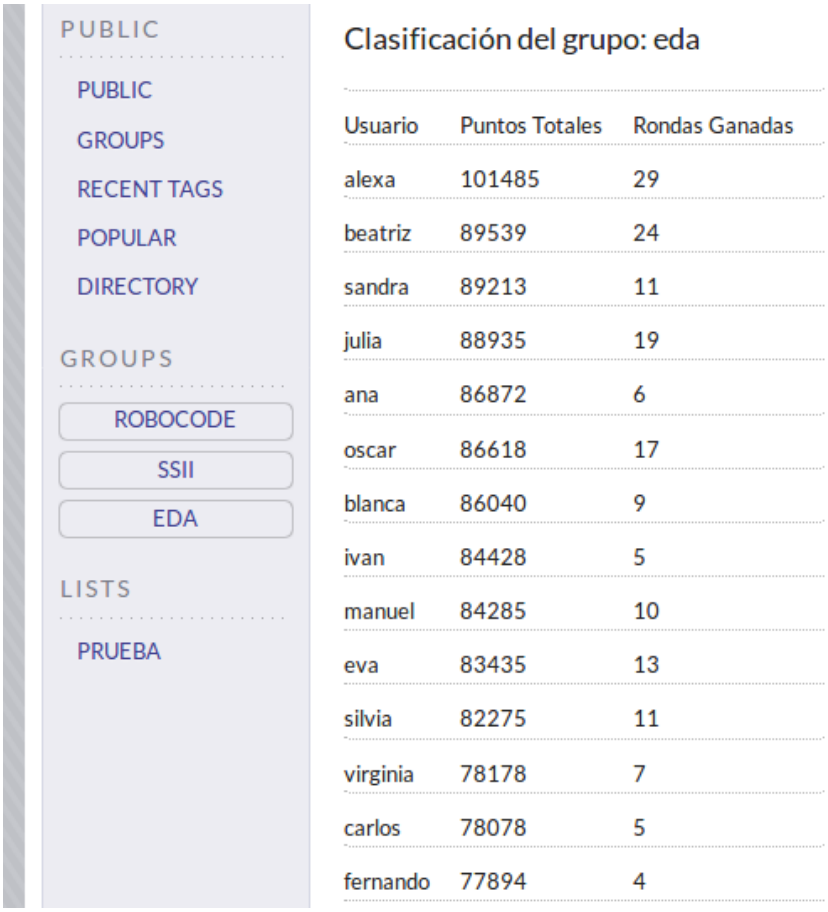
```
¡Hola!
¡Bienvenido al generador de batallas Robocode!
Leyendo la configuración...
¿A qué grupo perteneces?
Escribe cancel para cancelar.
eda
Las medias de los usuarios del grupo 'eda' han sido actualizadas.
ACTUALIZANDO ROBOTS...
carlos 2.5556
silvia 2.4286
ana 2.4286
manuel 2.4286
david 1.5714
beatriz 2.7143
oscar 1.8571
pablo 2.3333
juan 1.8571
julia 2.4286
blanca 2.5714
mario 2.2857
jaime 1.8571
alexa 2.4286
elena 2.1429
ivan 2.0000
fernando 2.4286
eva 2.2857
raquel 2.0000
alberto 2.0000
jesus 1.5000
ruben 1.0000
miriam 3.0000
daniel 1.0000
adriana 0
victor 0
virginia 2.5000
cintia 2.0000
```

Ilustración 10 – Inicio del sistema y actualización de medias de los estudiantes

Es interesante poder modificar el periodo de medias porque así el profesor puede ajustar con más exactitud el criterio que aplica: puede tener en cuenta el rendimiento a lo largo de todo el curso o el de los últimos días –promoviendo así un uso continuo de la plataforma, sin dejar de completar tareas-.

El profesor ejecuta los *roborumbles,* cuyos resultados acumulados pueden apreciarse en la pestaña *Robocode* de *Bolotweet*.

Ilustración 11 – Ejecución de un roborumble



PUBLIC

PUBLIC
GROUPS
RECENT TAGS
POPULAR
DIRECTORY

GROUPS

ROBOCODE
SSII
EDA

LISTS

PRUEBA

Clasificación del grupo: eda

Usuario	Puntos Totales	Rondas Ganadas
alexa	101485	29
beatriz	89539	24
sandra	89213	11
julia	88935	19
ana	86872	6
oscar	86618	17
blanca	86040	9
ivan	84428	5
manuel	84285	10
eva	83435	13
silvia	82275	11
virginia	78178	7
carlos	78078	5
fernando	77894	4

Ilustración 12 – Resultados acumulados en Bolotweet

Además, el profesor puede cambiar los parámetros de una batalla desde la aplicación, si lo considera necesario:

```
--------------------------------------------------------------------------------
Elige una de las siguientes opciones:
1.-Modificar opciones de notas/grupo.           2.-Modificar/crear robots.
3.-Modificar opciones de batalla.               4.-Generar batalla.
0.-Salir.
--------------------------------------------------------------------------------
Opción:
3
--------------------------------------------------------------------------------
Elige una de las siguientes acciones:
1.-Modificar tamaño de tablero.         2.-Modificar número de rondas.
0.-Salir.
--------------------------------------------------------------------------------
Opción:
```

Ilustración 13 – Opciones para parametrizar las batallas

Adicionalmente, un alumno puede solicitar cambiar los colores de su robot, tarea que puede llevarse a cabo desde el sistema.

```
--------------------------------------------------------------------------------
Elige una de las siguientes opciones:
1.-Modificar opciones de notas/grupo.           2.-Modificar/crear robots.
3.-Modificar opciones de batalla.               4.-Generar batalla.
0.-Salir.
--------------------------------------------------------------------------------
Opción:
2
--------------------------------------------------------------------------------
Elige una de las siguientes acciones:
1.-Listar robots existentes.            2.-Crear nuevo robot.
3.-Modificar colores de un robot.       0.-Salir.
--------------------------------------------------------------------------------
Opción:
3
Introduce el nombre del robot que quieres modificar
Escribe cancel para cancelar.
sandra
¿Quieres elegir los colores de tu robot o usar los de por defecto? (S/N) (S: Elegir colores)
Escribe cancel para cancelar.
s
Los colores disponibles son:
black blue green orange pink red white yellow
Elige: colorBody colorGun colorRadar
black green green
```

Ilustración 14 – Diálogo para cambiar los colores de un robot

El resto de funcionalidades que el sistema tiene implementadas se detallan en el apéndice A de la memoria.

Lo que se espera que suceda a lo largo del curso es analizado en profundidad en la sección 5 de la memoria.

4. Desarrollo de la solución

Toda la funcionalidad provista por el sistema va a ser representada y explicada mediante casos de uso. A continuación se van a listar y detallar los casos de uso para dos de los actores de la plataforma: profesores y estudiantes.

Ver clasificación: Esta acción la puede llevar a un usuario genérico, tanto profesor como estudiante. Pueden consultarse los resultados acumulados de todas las batallas disputadas entre usuarios de un mismo grupo. Un estudiante solo puede consultar los resultados de los grupos a los que pertenece y tienen el plugin activado, y un profesor aquellos grupos en los que tiene el rol *grader*.

Ejecutar juego: Acción que solamente un profesor puede realizar consistente en el lanzamiento del script que inicia la lógica para desplegar una batalla, cambiar los parámetros de la misma y gestionar robots.

Gestión de grupos: El profesor puede gestionar los grupos en los que tiene el rol de *grader*. Puede pasar de uno a otro simplemente indicando el ID o el *nickname* del grupo en cuestión, o modificar el número de días a tener en cuenta para el cálculo de las medias.

Cambiar grupo de clase: Si el profesor tiene varios grupos registrados en el sistema, puede cambiar de grupo sobre el que trabajar, simplemente indicando el nuevo grupo al que cambiarse.

Elegir nuevo periodo de días: El profesor puede elegir el periodo de tiempo durante el cual se van a recoger las calificaciones obtenidas por un estudiante, desde el día actual hacia atrás. Esto es, si por ejemplo se indica que el período de notas es de tres días, se calculará para cada estudiante la media de las notas recibidas en los mensajes enviados en los últimos tres días.

Gestionar parámetros de batalla: Interfaz que ofrecerá al profesor la opción de elegir qué parámetros de la batalla quiere cambiar. Estos parámetros son las dimensiones del tablero y el número de rondas que lucharán los robots.

Cambiar número de rondas: Relacionado con el caso de uso anterior, se pedirá al profesor que introduzca el número de rondas que tendrá la batalla.

Cambiar tamaño del tablero: Relacionado con el caso de uso anterior, se pedirá al profesor que introduzca el ancho y el alto del tablero.

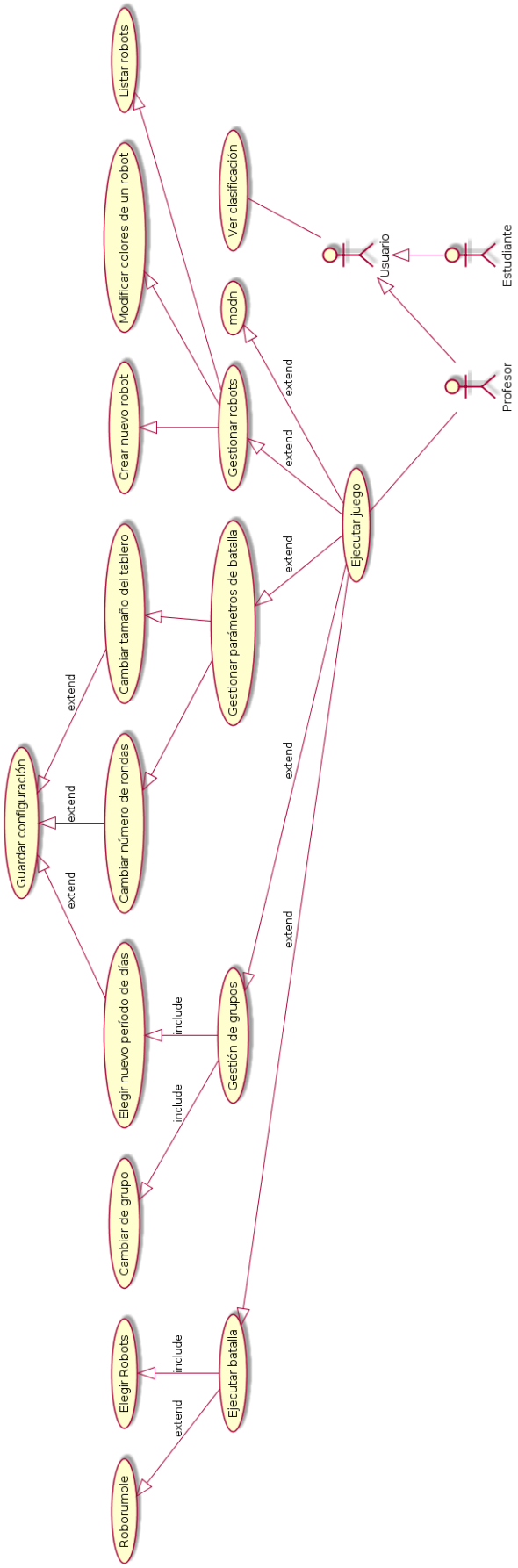


Ilustración 15 – Casos de uso del sistema para profesores y alumnos

Crear robot: Relacionado con el caso de uso anterior, un profesor podrá crear un robot para un alumno, en caso de no haberlo registrado hasta el momento. Podrá indicar el nombre y los colores del mismo, de entre una serie de opciones a elegir, siendo estos aplicables a tres partes del mismo: cuerpo, radar y cañón.

Modificar colores de un robot: Relacionado con el caso de uso anterior, para un robot existente, el profesor podrá seleccionar sus nuevos colores.

Listar robots: Se mostrarán por pantalla todos los usuarios del grupo que tengan su robot registrado.

Ejecutar batalla: Lanzará el script que permite ejecutar una batalla. Las condiciones de la batalla serán o bien las que contenga el fichero de configuración o las temporales indicadas por el profesor, en caso de que haya decidido modificar alguna de estas sin guardar la configuración.

Elegir robots: Relacionado con el caso de uso anterior, el profesor tendrá que registrar los robots que vayan a participar en la batalla. No podrá registrar un robot dos veces, para que un usuario no pueda luchar contra sí mismo, y mínimo tienen que participar dos robots.

Roborumble: Relacionado con el caso de uso anterior, el profesor puede optar por seleccionar todos los robots del grupo para que luchen en una misma batalla.

A continuación se muestran los casos de uso proporcionados por los scripts de administración. Será el usuario administrador el encargado de la gestión del sistema.

Ilustración 16 – Casos de uso para el administrador

Activar o desactivar Robocode: El profesor de un grupo puede optar por hacer uso del juego o no. Para hacer la gestión del sistema más eficiente, evitando hacer consultas sobre grupos que no tienen el plugin activado, se ha instanciado una tabla en la base de datos que indica para cada grupo si el plugin está activado o no. El administrador puede activar o desactivar el plugin para un grupo, a petición del profesor. Este es el único script cuyos permisos de ejecución recaen exclusivamente en el administrador.

Actualizar resultados de batallas: Este script proporciona la funcionalidad necesaria para actualizar las puntuaciones de los estudiantes. Recorrerá el directorio donde se almacenan los resultados de todas las batallas, leerá los ficheros y actualizará los resultados que posteriormente son mostrados en la página principal del plugin.

Actualizar ficheros de medias: El administrador puede, manualmente, modificar los archivos de las medias de los alumnos de un grupo. Para ello, ha de introducir como parámetro al script PHP el grupo a actualizar y el número de días a tener en cuenta.

Crear robots: El administrador recibe del profesor un archivo .csv en el que aparecen los *nicknames* de los alumnos pertenecientes a su asignatura junto a los colores correspondientes a sus robots. Hace uso de un script, *crearrobots.sh*, que lee dicho fichero y crea en el sistema los robots que aparecen en él.

4.1. Arquitectura de la solución

La solución consiste en la integración de la plataforma *Bolotweet* con las batallas lanzadas por *Robocode*.

El sistema se articula en torno a los siguientes elementos:

- Un plugin para *Bolotweet*. Enlaza con *Robocode* y muestra los resultados acumulados de las batallas ejecutadas en esta plataforma, ordenando para cada grupo a los estudiantes en función de los puntos totales obtenidos a lo largo de todas las batallas.
- Scripts de administración. Estos scripts se encargan de enlazar las dos herramientas, de manera que el recuento de puntos y la actualización de medias sea automático.
- Scripts para ejecutar una batalla. Ejecutan *Robocode* externamente para poder parametrizar las batallas de manera más cómoda y para poder recoger los resultados de estas automáticamente.

Primeramente se va a detallar un diagrama de despliegue para poder observar dónde se aloja cada componente del sistema.

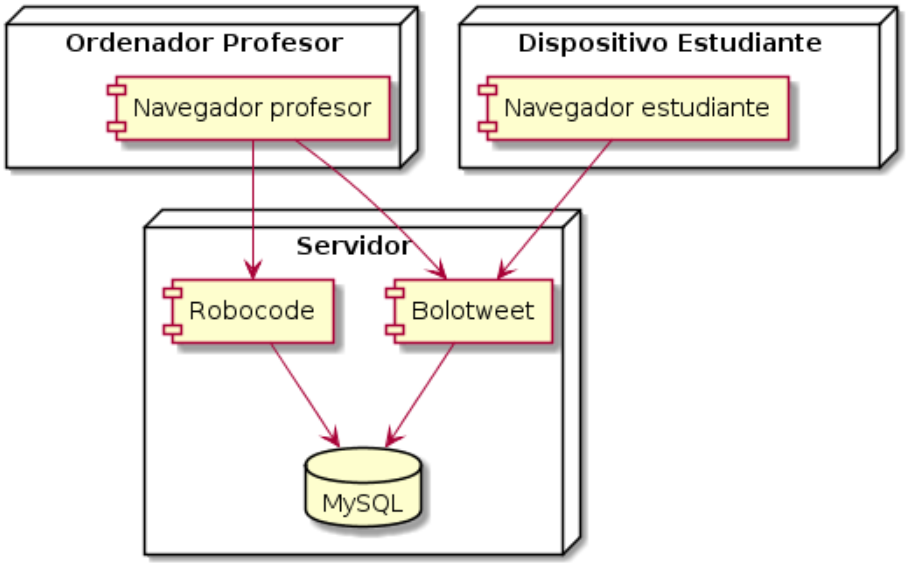


Ilustración 17 – Diagrama de despliegue del sistema

En el servidor se alojará tanto *Bolotweet* como *Robocode*, donde se realizarán todas las tareas de mantenimiento necesarias.

Tanto el profesor como los estudiantes podrán acceder a *Bolotweet* desde sus dispositivos. En el caso de los alumnos, para completar las tareas; los profesores, para corregirlas y, en caso de ser necesario, contestar dichos mensajes con las correcciones que crean oportuno indicar.

Adicionalmente, solo el profesor podrá acceder a *Robocode*, para ejecutar las batallas pertinentes.

4.1.1. Diseño del plugin Robocode

El primer paso fue crear un plugin para el sistema *StatusNet*. Este plugin se encarga de recoger los datos de las batallas ejecutadas y actualizar los resultados mostrados en *Bolotweet*, así como exportar las medias actualizadas de los alumnos de un grupo.

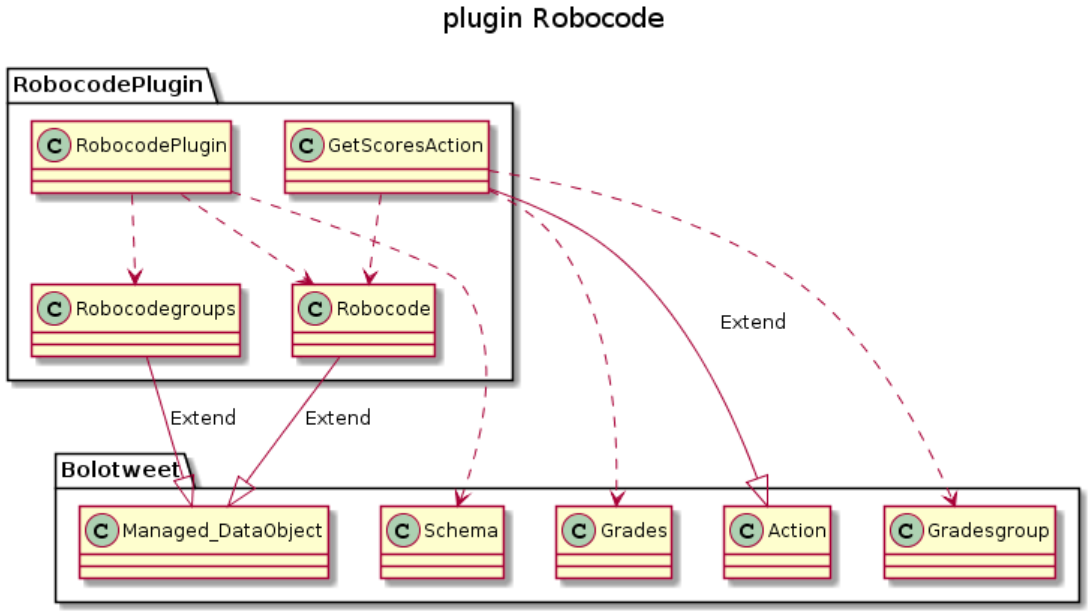


Ilustración 18 – Diagrama de clases del plugin Robocode

Clase RobocodePlugin: Se trata de la clase principal que extiende la clase plugin. Se encarga de cargar los ficheros necesarios para el correcto funcionamiento del plugin, y de relacionar sus funciones con el *core* a través de eventos. Gracias a la clase *Schema* se asegura la estructura de las tablas de la base de datos propias del plugin, y hace uso de las funciones de las clases *Robocode* y *Robocodegroups* para la definición las tablas.

Clase GetscoresAction: Esta clase extiende a la clase *Action* y se encarga de mostrar los resultados acumulados de las batallas que han librado todos los usuarios que pertenecen al mismo grupo que el usuario que está visualizando la página y también actualiza la media de sus notas. Para la primera tarea, hace uso de la clase *Robocode* para obtener la puntuación total y el número de rondas ganadas de todos los usuarios del grupo para después mostrarlos en formato tabla. También hace uso de esta misma clase para

actualizar dichos valores cuando el sistema registra el resultado de una batalla. Para la segunda tarea, se hace uso de la clase *Grades* para volcar en un fichero .csv las medias de las notas de los usuarios del grupo. La clase *Gradesgroup* permite comprobar si el usuario que está visualizando la página principal del plugin es un *grader*. En caso de serlo y de tener su o sus grupos el plugin activado, mostrará sus clasificaciones.

Clase Robocode: Extiende la clase *Managed_DataObject* y representa la tabla *robocode_scores*. Esta tabla contiene las puntuaciones de todos los usuarios que pertenecen a uno o varios grupos con el plugin activado. Registra su puntuación total y número de rondas ganadas para cada grupo al que pertenece. Proporciona todas las funciones necesarias para comunicarse con la tabla de la base de datos.

Clase Robocodegroups: Extiende la clase *Managed_DataObject* y representa la tabla *robocode_groups*. Esta tabla contiene información acerca de qué grupos tienen activado el plugin, información necesaria a la hora de la gestión de actualizaciones de puntuaciones, registro de nuevos grupos, etc. Proporciona todas las funciones necesarias para comunicarse con la tabla de la base de datos.

4.1.2. Diseño de scripts de soporte para administración

Para la integración de ambas plataformas se requieren scripts que se encarguen de la comunicación de los datos entre estas. Por ejemplo, cuando una batalla ejecutada por *Robocode* finaliza, el resultado de la misma ha de ser almacenado en la base de datos de *Bolotweet*. De manera inversa, antes de lanzar una batalla, han de construirse los robots en función de su nivel, que viene indicado por su desempeño en *Bolotweet*, por lo que hay que acceder a esta para recuperar las medias de las calificaciones de los estudiantes.

Estos scripts son usados tanto por el administrador para actualizar las tablas de la base de datos manualmente como por la lógica del sistema, que también ha de trabajar sobre estas tablas.

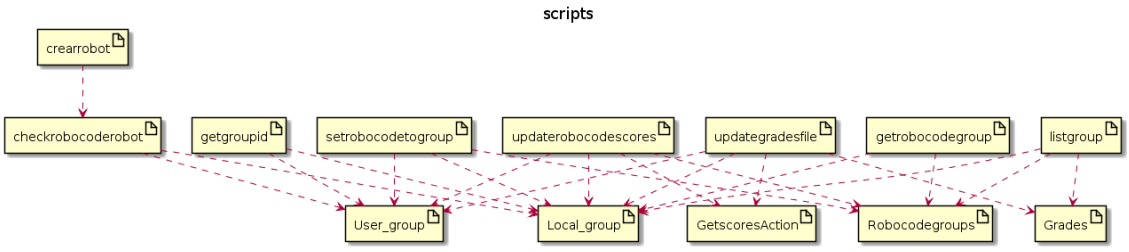


Ilustración 19 – Diagrama de los scripts de administración creados

Checkrobocoderobot: Este script PHP se encarga de comprobar si un usuario pertenece a un grupo concreto. Hace uso de las clases *User_group* y *Local_group* para validar que el grupo introducido existe, para aceptar su identificación mediante tanto ID como alias y para hacer la propia comprobación.

Crearrobot: Este script *Bash* recibe un ID de un grupo y un archivo .csv que contiene los usuarios de ese grupo seguido de sus correspondientes colores e introduce en el

sistema los robots de estos alumnos. Hace uso del script *checkrobocoderobot.php* para comprobar que los usuarios listados en el archivo pertenecen al grupo indicado por parámetro.

GetGroupId: Este script PHP se encarga de devolver, dado un ID de un grupo, su nombre y alias; y viceversa, dado un alias, devuelve su ID. Este script proporciona soporte a todas las operaciones en las que se requiere hacer consultas a la base de datos y la información recibida acerca del grupo varía en relación al formato. Para validar los datos de entrada y consultar la información requerida hace uso de las clases *Local_group* y *User_group.*

Getrobocodegroup: Este script PHP indica si el grupo introducido tiene el plugin *Robocode* activado o no. Hace uso de la clase *Local_group* para recoger la identificación del grupo y de la clase *Robocodegroups* para consultar la información solicitada.

ListGroup: Este script PHP devuelve todos los usuarios pertenecientes a un grupo que se indique por parámetro y que tenga el plugin *Robocode* activado. Hace uso de la clase *Local_group* para recoger la identificación del grupo. Para comprobar que tiene el plugin activado hace uso de la clase *Robocodegroups*. Para devolver la lista que contiene los nombres de todos los usuarios pertenecientes al grupo, recurre a la clase *Grades.*

SetRobocodetogroup: Este script PHP activa o desactiva el plugin *Robocode* a un grupo concreto. Hace uso de las clases *Local_group* y *User_group* para comprobar la existencia del grupo indicado, y de la clase *Robocodegroups* para acometer la acción.

Updategradesfile: Este script PHP actualiza el fichero con las medias de los grades de los usuarios de un grupo. Hace uso de las clases *Local_group* y *User_group* para recoger la identificación del grupo. Para comprobar que tiene el plugin activado hace uso de la clase *Robocodegroups*. Después, mediante la clase *Grades* obtiene los usuarios del grupo indicado y con la clase *GetscoresAction* actualiza dicho fichero. Se mantiene un fichero por grupo existente.

Updategradesfileandshow: Este script *Bash* recibe un grupo y un número de días y actualiza el fichero de medias del grupo. Su propósito es proporcionar la lógica necesaria para discernir entre el ID y el alias de un grupo. Llamará al script *updategradesfile.php* activando las opciones necesarias para el correcto funcionamiento de la actualización.

UpdateRobococeScores: Este script PHP actualiza la tabla *robocode_scores* mediante el archivo que contiene los resultados de una batalla. Hace uso de las clases *Local_group* y *User_group* para recoger la identificación del grupo. Con la clase *Robocodegroups* comprueba que el grupo indicado tiene el plugin activado, y mediante la clase *GetscoresAction* realiza la actualización de los resultados.

UpdateAllRobocodeScores: Este script *Bash* recorre el directorio donde se almacenan los ficheros que contienen los resultados de las batallas y para cada fichero, llama al script *UpdateRobocodeScores.php* para actualizar la tabla *robocode_scores.*

Adicionalmente, se han modificado algunos de los scripts ya existentes para la versión anterior de *Bolotweet*, para adaptarlos a las nuevas funcionalidades del sistema.

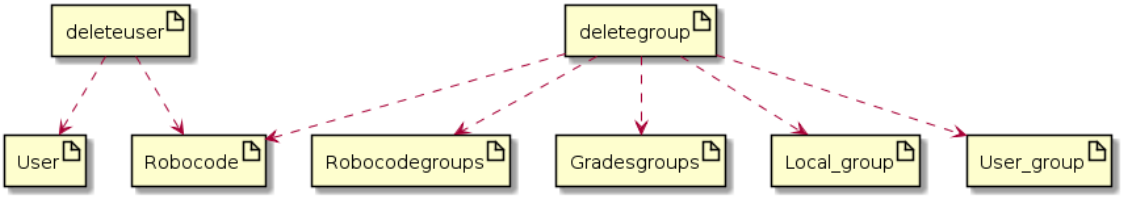


Ilustración 20 – Diagrama de los scripts de administración modificados

Deletegroup: Este script PHP elimina del sistema el grupo que se le indique por parámetro. Se han añadido sentencias para que elimine todos los usuarios registrados en la tabla *robocode_scores*, para no almacenar resultados de usuarios pertenecientes a un grupo que ha dejado de existir. También se elimina la fila del grupo eliminado en la tabla *robocode_groups*. Por último, también se elimina la fila del grupo y su *grader* en la tabla *grades_group*. Esta tabla indica, para cada grupo, qué usuarios poseen el rol *grader*.

Deleteuser: Este script PHP elimina del sistema el usuario que se le indique por parámetro. Se ha añadido una sentencia para que elimine al usuario de la tabla *robocode_scores*, para dejar de almacenar los resultados de un usuario no existente.

4.1.3. Scripts para la batalla

A su vez, la generación de una batalla se sustenta en dos scripts.

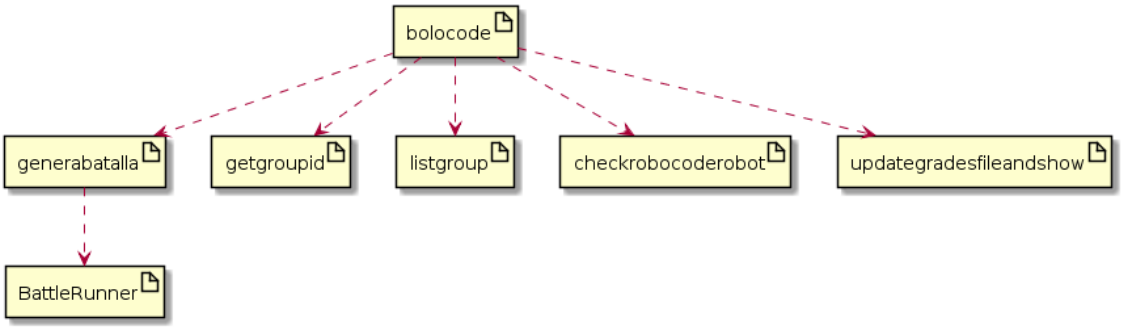


Ilustración 21 – Diagrama de clases del script generador de una batalla

Bolocode: Es el script general del sistema, que se encarga desde el inicio de sesión de un grupo hasta el lanzamiento de una batalla. Hace uso del script *getgroupid.php* para poder aceptar, a la hora de iniciar sesión, tanto el ID de un grupo como su *nickname*. El script *listgroup.php* le permite recuperar el listado de usuarios pertenecientes al grupo, y con el script *checkrobocoderobot.php* comprueba que un robot pertenece al grupo. Es útil cuando el sistema pide introducir el nombre de un robot, ya sea para modificar su

color o para introducir uno nuevo. El script *updategradesfileandshow.sh* es invocado cada vez que se inicia la ejecución del sistema o se modifica el número de días de cálculo de la media, para que el nivel de los robots siempre esté lo más actualizado posible.

GeneraBatalla: Es invocado por *bolocode.sh* y es el encargado de generar la ejecución de una batalla. Para ello, hace uso de la clase *BattleRunner* que, como veremos más adelante, es la encargada de conectar con *Robocode* para el lanzamiento de una batalla.

4.2. Diagramas de secuencia

Para mostrar con mayor detalle el funcionamiento del sistema, se van a ilustrar a continuación los diagramas de secuencia para los casos de uso mostrados con anterioridad, cuya funcionalidad se sustenta en el plugin *RobocodePlugin* y los scripts de administración también mencionados y explicados.

Mecánica general

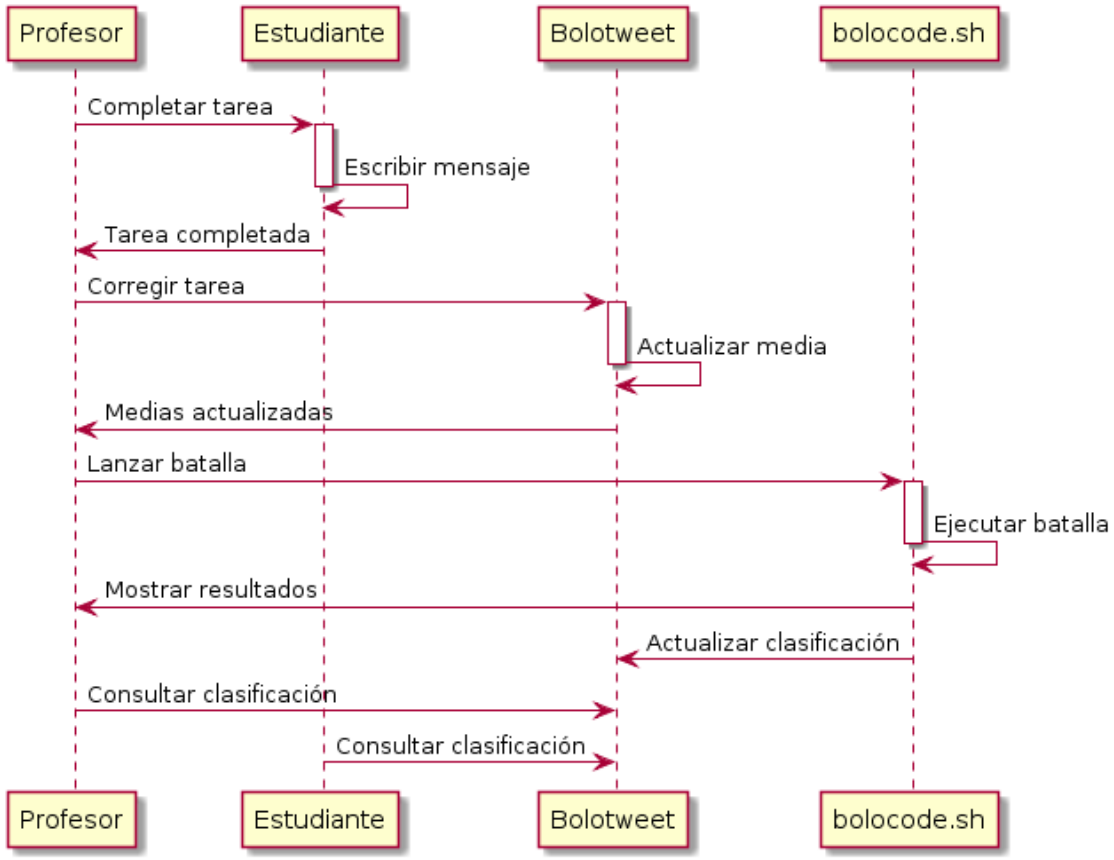


Ilustración 22 – Diagrama de secuencia que muestra el funcionamiento general del sistema

El profesor propone tareas a ser completadas. Estas tareas consisten en escribir mensajes sintetizando alguna idea vista durante la clase teórica.

Los alumnos completan estas tareas y el profesor las corrige. Entonces, al lanzar el sistema *Bolocode*, este actualiza las medias de los alumnos con las nuevas tareas

corregidas. El profesor lanza una batalla con los parámetros que desee y el sistema muestra los resultados tras finalizar esta. El propio sistema, además, actualiza la clasificación que puede ser consultada en *Bolotweet*, más concretamente en el plugin *Robocode*, tanto por el profesor como por los estudiantes.

Ver clasificación

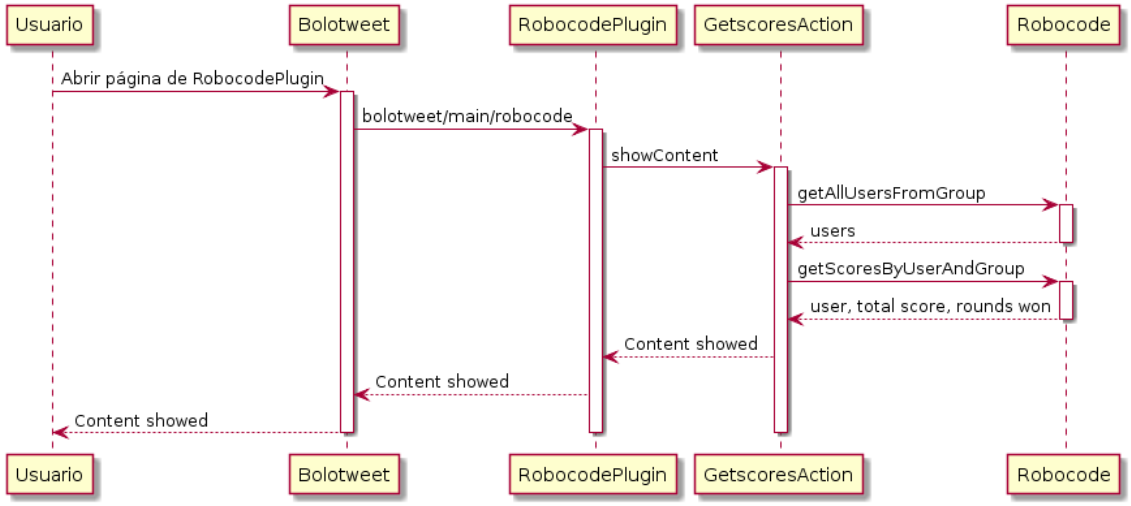


Ilustración 23 – Diagrama de secuencia para el caso de uso Ver Clasificación

El usuario se conecta a la página de *Bolotweet* y accede al plugin mediante el enlace del menú lateral izquierdo. Este enlace es apreciable en la ilustración 12. Automáticamente, se le desplegarán las tablas con las clasificaciones de todos los grupos a los que pertenece. Esta tarea recae en la clase *GetscoresAction*, que consulta a la clase *Robocode* los usuarios pertenecientes al grupo o grupos del usuario, y para cada usuario obtenido consulta sus puntos y rondas ganadas. Esta información será desplegada en forma de tabla y mostrada en la página del plugin.

Elegir nuevo período de días

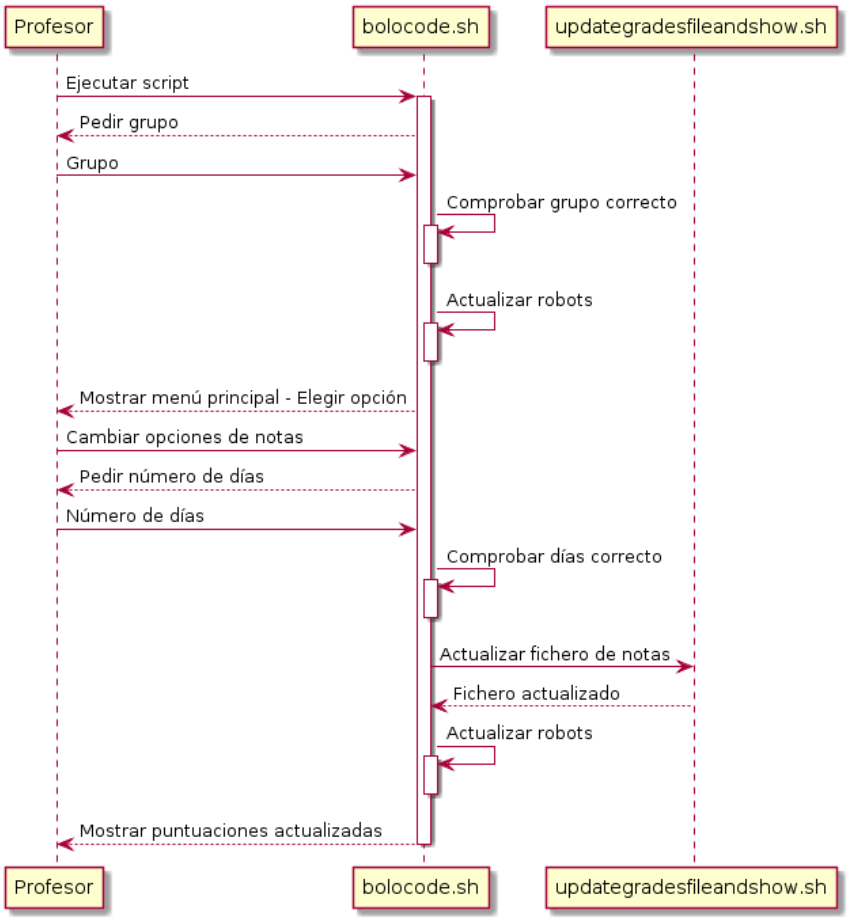


Ilustración 24 – Diagrama de secuencia para el caso de uso Elegir nuevo periodo de días

El profesor ejecuta el script *robobatalla.sh*, que contiene la funcionalidad necesaria para lanzar una batalla. Lo primero que hace el script es solicitar el grupo de pertenencia. Cuando comprueba que el número de grupo es correcto, actualiza el nivel de los robots con el archivo .csv de grades y solicita qué acción realizar. En los próximos diagramas, se omitirá esta interacción por cuestión de redundancia. El profesor indica que quiere modificar el número de días, y el sistema le pide que introduzca un número. Tras comprobar que lo introducido es un número entero, el script llama al script de administración *updategradesfileandwhow.sh*, que actualiza el fichero del grupo que se le indique por parámetro, junto al número de días a contar. Posteriormente, actualiza los códigos de los robots del grupo acorde a la media obtenida por el usuario y, finalmente, muestra las nuevas medias al profesor.

El proceso de guardar la configuración se mostrará más adelante, pues es común a la actualización del número de rondas y del tamaño del tablero.

Cambiar número de rondas

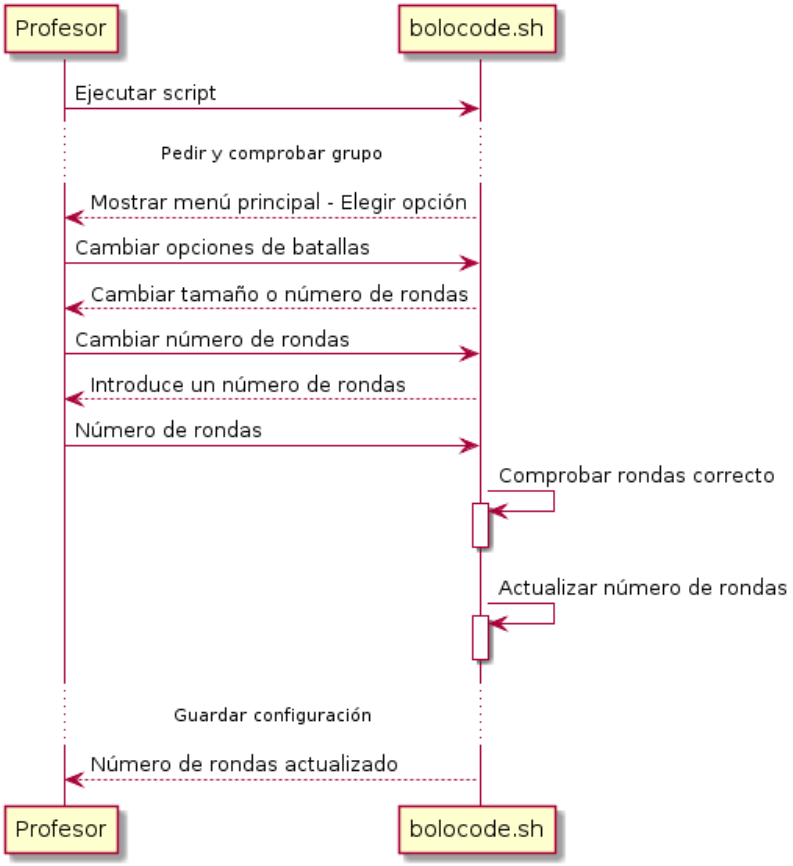


Ilustración 25 – Diagrama de secuencia para el caso de uso Cambiar número de rondas

Tras ejecutar el script e iniciar sesión con un grupo correctamente, le pide al profesor qué acción realizar. Al indicar que quiere cambiar los parámetros de una batalla, se le ofrece el poder cambiar el número de rondas o el tamaño del tablero. Tras indicar que desea cambiar el número de rondas, se le pide introducir un número. Tras comprobar que es un número correcto, actualiza el valor del número de rondas para, en la ejecución de una batalla, se coja como parámetro el valor actualizado.

Cambiar tamaño del tablero

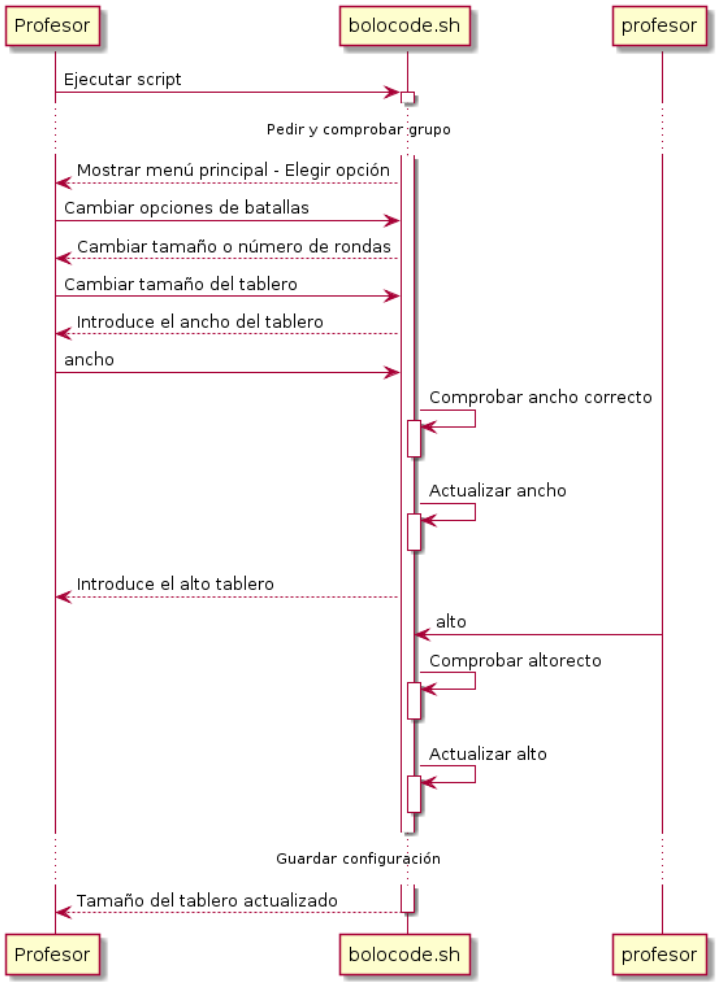


Ilustración 26 – Diagrama de secuencia para el caso de uso Cambiar tamaño del tablero

Tras ejecutar el script e iniciar sesión con un grupo correctamente, le pide al profesor qué acción realizar. Al indicar que quiere cambiar los parámetros de una batalla, se le ofrece el poder cambiar el número de rondas o el tamaño del tablero. Tras indicar que desea cambiar el tamaño del tablero, primeramente el sistema le pide introducir el ancho. Tras la pertinente comprobación y actualización, se le pide introducir el alto, para posteriormente repetir el proceso.

Guardar configuración

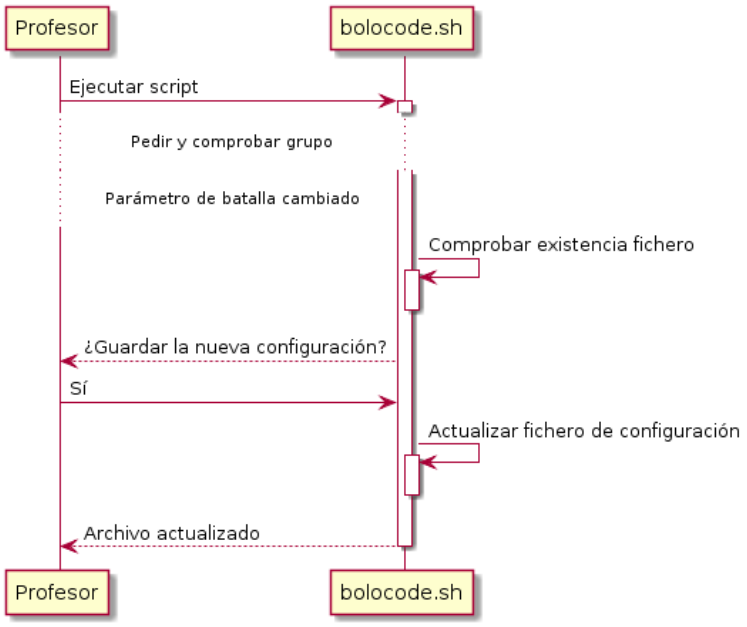


Ilustración 27 – Diagrama de secuencia para el caso de uso Guardar configuración

Una vez se ha cambiado algún parámetro y se ha registrado de forma temporal, se inicia el proceso de guardado. Primeramente se comprueba que existe un fichero y, si no existe, se crea uno con una configuración predeterminada. Se le pregunta al usuario si desea guardar los cambios y, en caso positivo, se actualiza el fichero con el valor recientemente introducido.

Notar que, si el usuario indica que no quiere guardar la configuración, el proceso termina.

Crear nuevo robot

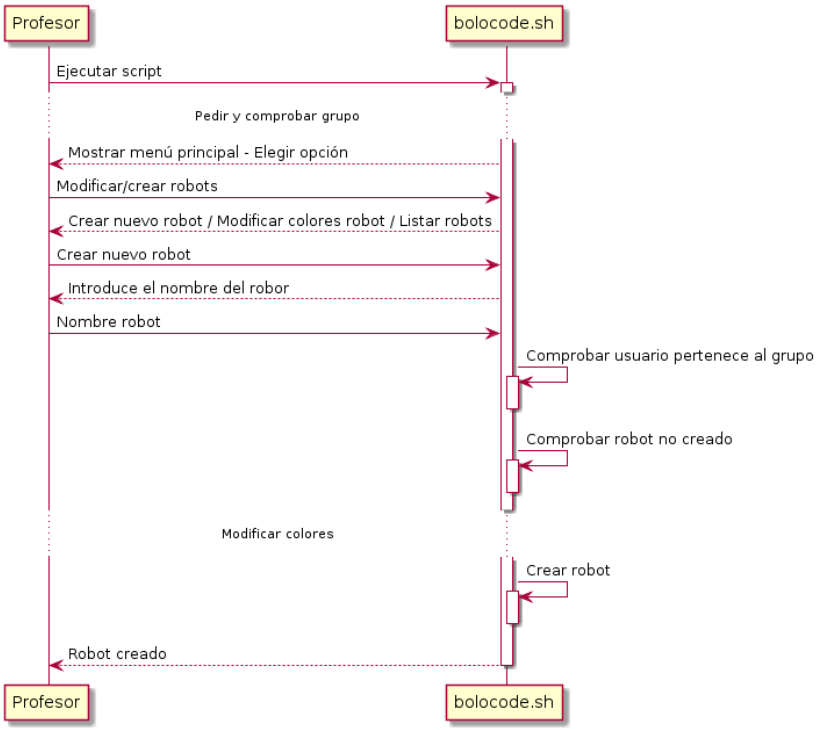


Ilustración 28 – Diagrama de secuencia para el caso de uso Crear robot

El sistema solicitará al usuario introducir el nombre del robot. Al recibirlo, comprobará primeramente que la entrada es una cadena de caracteres correcta, para después comprobar que el nombre introducido pertenece al de un usuario del grupo sobre el que se está trabajando. Después de estas comprobaciones, se comprobará también si el robot ya está creado o no. En caso de estarlo, se preguntará al profesor si desea sobrescribirlo. En caso positivo, se continuará con la ejecución y se seguirán pidiendo datos. Si no desea sobrescribirlo, se le pedirá otro nombre. Cuando el robot a crear (o modificar) está decidido, comienza el proceso de aplicar colores al robot. Este proceso se explicará en el siguiente punto. Tras esto, se comprobará qué nivel tiene el robot. Para ello se recurre al fichero que contiene la media de los *grades* recibidos en *Bolotweet*. Si resulta que no ha participado en la plataforma, no aparecerá en el fichero, por lo que se le asignará el menor nivel posible.

Para crear el código de un robot, se ha recurrido al uso de plantillas. Estas plantillas tienen escrito el comportamiento de un robot. Existe una plantilla por cada nivel de robot existente. Se volcará dicha plantilla en el archivo .java del robot (que contiene su comportamiento) y se cambiará el nombre de la clase por el del nombre del robot. Finalmente, compilará el código.

Modificar colores de un robot

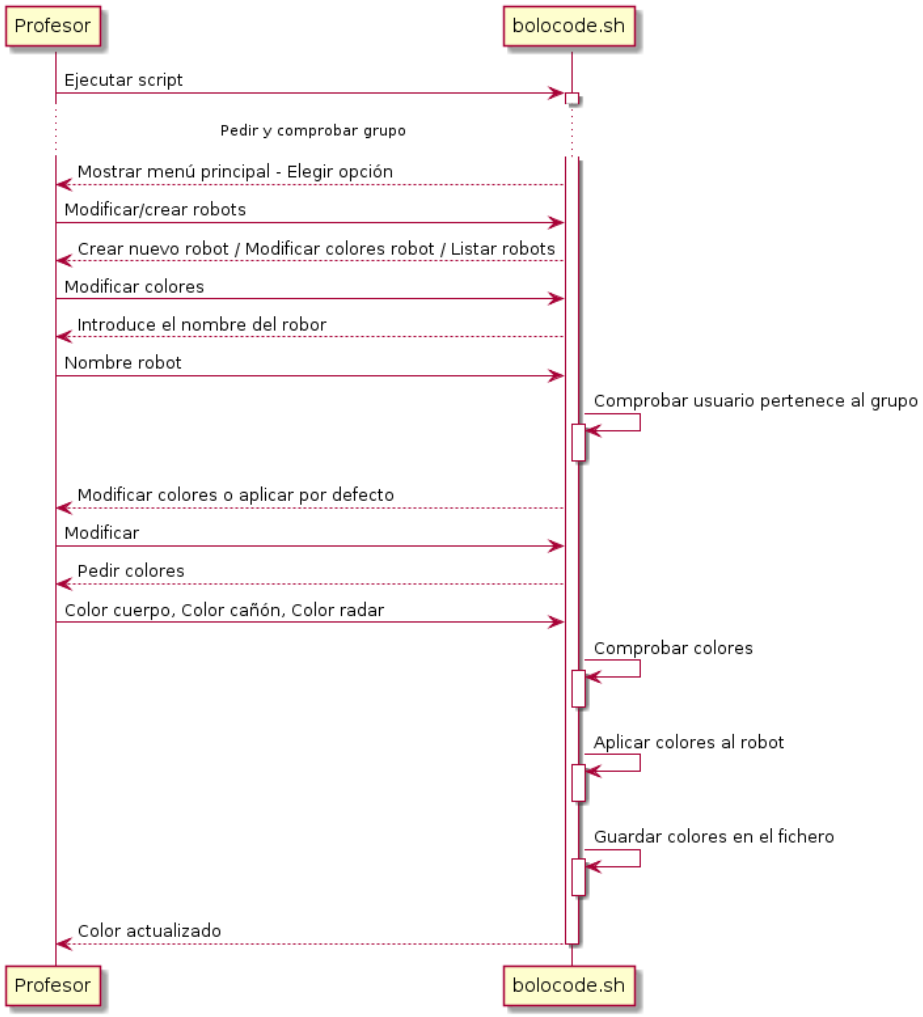


Ilustración 29 – Diagrama de secuencia para el caso de uso Modificar colores de un robot

Los colores de un robot pueden ser modificados bien si el profesor así lo indica desde el menú del script, o bien si al seleccionar la opción crear robot, introduce un robot ya existente y señala que desea sobrescribirlo. Entonces, se le pregunta al profesor si desea seleccionar sus propios colores o aplicar los colores por defecto. Si quiere elegirlos, se le pide introducir dichos colores. El profesor puede elegir de entre una gama de colores predeterminada:

```
Los colores disponibles son:
black blue green orange pink red white yellow
Elige: colorBody colorGun colorRadar
black orange orange
```

Colores aplicables a un robot

Si los colores introducidos no son correctos, el sistema aplicará el color por defecto asignado a la parte del cuerpo mal introducida. Posteriormente, aplicará los colores al robot plasmándolos en su código, para después registrar esos colores en un fichero que mapea los colores de todos los robots del sistema.

Listar robots

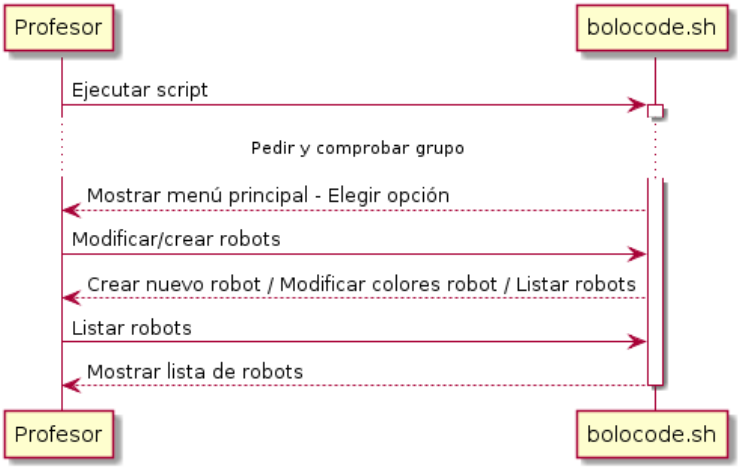


Ilustración 30 – Diagrama de secuencia para el caso de uso Listar robots

De nuevo en el menú para crear, modificar o listar los robots, el profesor indica esta última opción. Se le devolverá la lista completa de los usuarios del grupo que tengan un robot creado en el sistema.

Ejecutar una batalla eligiendo los robots

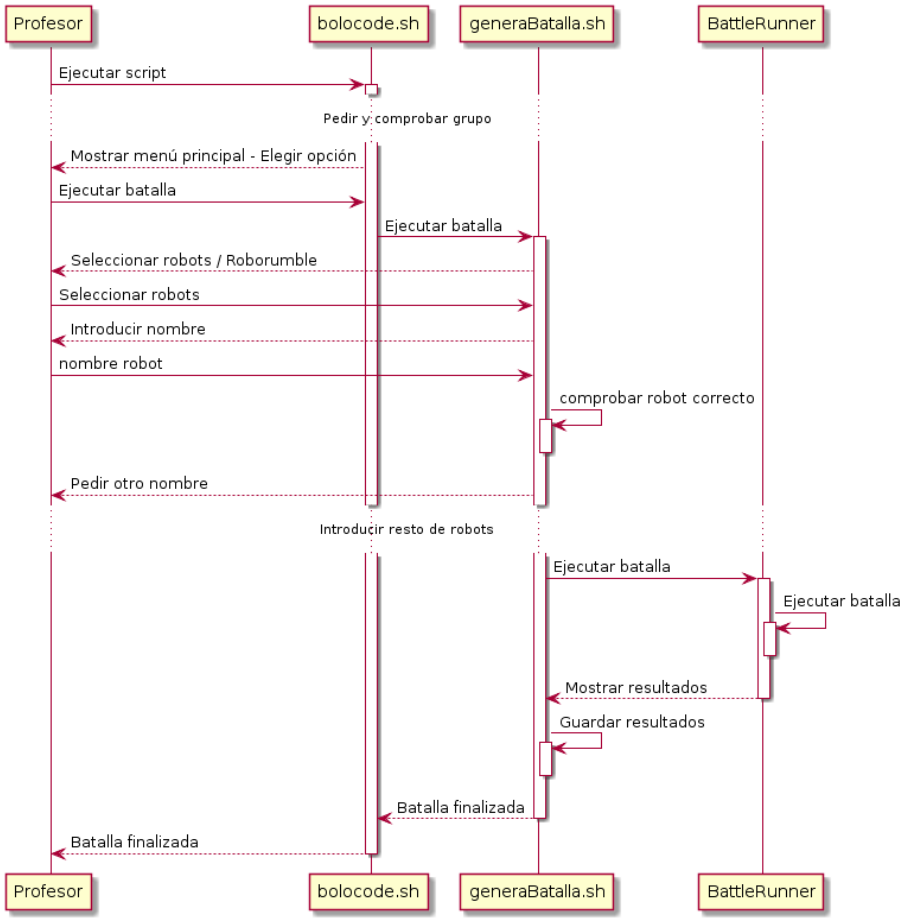


Ilustración 31 – Diagrama de secuencia para el caso de uso Ejecutar una batalla eligiendo los robots

El profesor selecciona la opción de generar una batalla. El script *generaBatalla.sh* recibe la lista de robots del grupo y la configuración de la misma, consistente en el tamaño del tablero y el número de rondas. Pide al profesor que indica si quiere elegir los robots manualmente o ejecutar un *roborumble*. Al elegir los robots manualmente, le pide introducir un nombre. Comprueba que dicho robot está en la lista del grupo y, si lo está, lo añade a la lista de robots participantes en la batalla. Dicho proceso vuelve a repetirse hasta que el profesor no desea añadir más robots. Mínimo ha de introducir dos robots y ninguno puede repetirse.

Tras elegir los robots, ejecuta la batalla. Para ello, el script recurre a la clase *BattleRunner* que, como se verá más adelante, es la clase encargada de ejecutar una batalla. Esta clase devuelve los resultados de la batalla para ser mostrados por consola y, además, también crea un fichero .csv temporal que los almacena. El script *generaBatalla.sh* es el encargado de guardar el contenido del fichero temporal en un fichero que el sistema reconoce para poder almacenarlo en la base de datos de *Bolotweet*. El sistema lo reconoce pues se codifica el nombre del fichero para que indique el grupo, los participantes y la fecha y hora de la batalla.

Finalmente, el control vuelve al profesor sobre el script *bolocode.sh*.

Ejecutar un roborumble

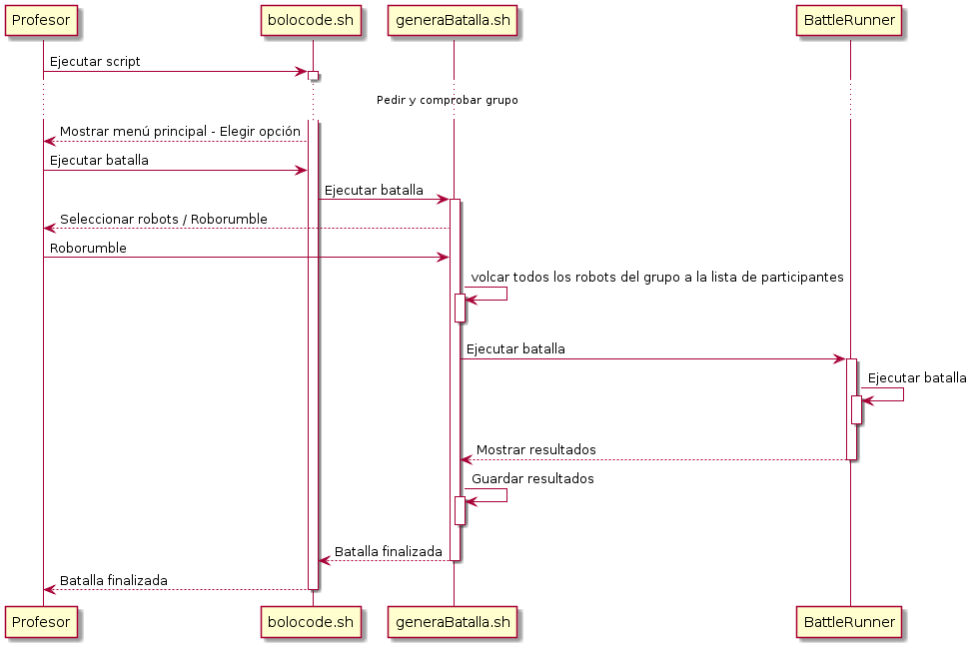


Ilustración 32 – Diagrama de secuencia para el caso de uso Ejecutar un Roborumble

La única diferencia respecto al caso de uso anterior radica en que el sistema vuelca todos los usuarios del grupo con robot a la lista de participantes, en vez de ser el profesor quien introduce manualmente los nombres de los robots que van a batallar.

4.3. *Robocode*

En esta sección se presentan las modificaciones necesarias en la parte de *Robocode*.

4.3.1. Batalla

Robocode es un juego de código abierto, por lo que se podría haber hecho uso de todo el potencial que ello supone. Sin embargo, también ofrece una API que brinda la posibilidad de ejecutar una batalla desde una aplicación externa. Se ha optado por aprovechar esta funcionalidad para simplificar y facilitar la construcción de la clase que ejecute una batalla, así como su integración con la herramienta *Bolotweet*.

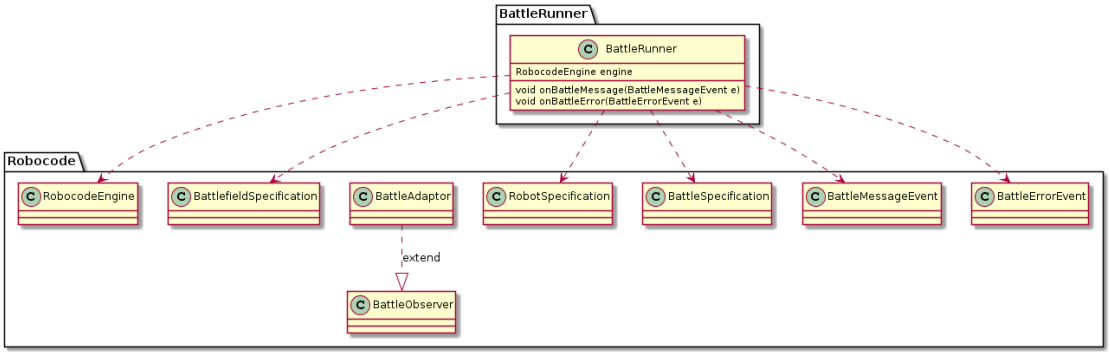


Ilustración 33 – Diagrama de clases de la clase BattleRunner

Clase BattleRunner: Clase principal que se encarga de la creación y ejecución de una batalla tras la lectura y parseo de los argumentos que reciba, que serán los parámetros de la batalla. Hace uso de la clase *RobocodeEngine* para poder instanciar una batalla. Las clases *BattlefieldSpecification*, *RobotSpecification* y *BattleSpecification* son las encargadas de, respectivamente, crear un campo de batalla con las dimensiones indicadas, una lista con los nombres de los robots participantes y un objeto que contenga la información mencionada junto con el número de rondas. Con las clases *BattleMessageEvent* y *BattleErrorEvent* gestiona los mensajes que serán siendo mostrados por la consola.

Clase RobocodeEngine: Proporcionada por la API de *Robocode*, es la interfaz que permite ejecutar una batalla en *Robocode* desde una aplicación externa. Cuenta con funciones que permiten crear una batalla con las especificaciones indicadas, ejecutarla y recoger los resultados.

Clase BattlefieldSpecification: Clase que define el tamaño de un campo de batalla.

Clase RobotSpecification: Clase que define las propiedades de un robot, que son obtenidas desde el repositorio local. Para la creación de una batalla, se proporciona una lista con objetos de esta clase, representando cada objeto a un robot participante en la batalla.

Clase BattleSpecification: Clase que define la configuración de una batalla. Entre otros, define el tamaño del campo de batalla, los robots participantes y el número de rondas, que son los parámetros que se necesitan manejar en el marco de este proyecto.

Clase BattleMessageEvent: Clase que recoge el evento accionado cuando el juego envía un mensaje. Muestra dicho mensaje por consola.

Clase BattleErrorEvent: Clase que recoge el evento accionado cuando el juego envía un mensaje de error. Muestra dicho mensaje por consola.

Clase BattleObserver: Clase que extiende la clase abstracta *BattleAdaptor* y que recibe los eventos del juego. Interesa el evento accionado cuando una batalla finaliza, para poder así registrar los resultados de la misma.

4.3.2. Robots

Sistema de niveles

Antes de explicar cómo han sido diseñados los distintos niveles asignables a un robot, es necesario detallar cómo funciona el sistema de puntos de una batalla y cómo funcionan los propios robots.

La posición de un robot se define mediante coordenadas cartesianas, donde el punto (0, 0) es la esquina inferior izquierda del campo de batalla. A su vez, la dirección que toma se mide en grados centígrados o radianes, se ofrecen operaciones para ambos sistemas de medida. Se utiliza una convención de dirección a la derecha, en el sentido de las agujas del reloj, donde el norte es el ángulo 0:

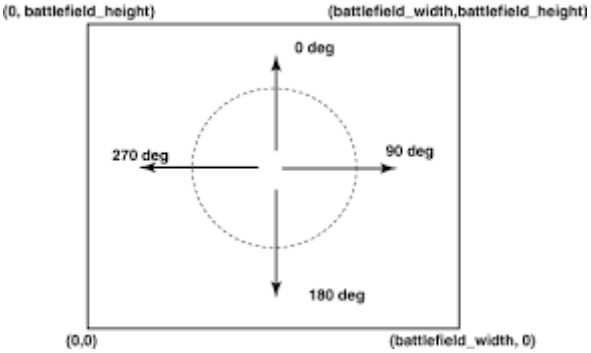


Ilustración 34 – Coordenadas del campo de batalla y dirección del robot

Respecto a la aceleración y velocidad que puede tomar un robot, estas medidas no han sido tomadas en cuenta a la hora de programar su comportamiento.

Un robot pierde vida al disparar. En concreto, pierde los mismos puntos de vida que la potencia imprimida al disparo, que es codificable. Esto es, si el robot dispara con potencia igual a uno, perderá un punto de vida. Sin embargo, si acierta al enemigo, este perderá 4 veces la potencia del disparo recibido y, además, el que dispara recibe tres

veces la potencia imprimida. En el caso anterior, de acertar, el que dispara recuperaría tres puntos de vida y el que recibe el disparo perdería cuatro.

Esta consideración ha de tenerse en cuenta pues no es una estrategia inteligente disparar sin asegurarse una mínima posibilidad de acertar al enemigo. Los primeros robots desarrollados en el proyecto disparaban sin seguridad de acertar, por lo que muchas rondas quedaban eliminados al haber perdido toda su energía disparando.

En cuanto a arrollar a los enemigos, los dos robots que colisionan pierde 0.6 puntos de vida. Por lo tanto, solo es una buena idea arrollar a un rival si el nivel de energía del mismo es suficientemente inferior al del propio robot.

En caso de chocar con un muro, pierde *velocidad*0.5 – 1,* donde *velocidad* es la velocidad que llevaba en ese momento.

Como se ha explicado con anterioridad, los robots creados en este proyecto están diseñados para evitar los muros en todo momento. Se ha desarrollado un código que calcula a qué distancia se encuentra el muro más próximo que esté en la dirección que toma el robot en todo momento y, si dicha distancia es menor que un mínimo definido, gira para evitar el choque.

Las funciones que dotan a los robots de funcionalidad pueden dividirse en tres grupos:

- ***De movimiento:*** Permiten al robot desplazarse a lo largo de todo el campo de batalla. Permiten al robot moverse hacia adelante, hacia atrás, girar su cuerpo un número de grados hacia la derecha o izquierda, y también girar el cañón o el radar individualmente. Además, son las que permiten disparar una bala imprimiéndole la potencia que se desee.
- ***De eventos:*** Permiten al robot definir su comportamiento según qué suceda durante la partida. Puede saber cuándo ha chocado contra un muro, contra otro enemigo, cuándo ha escaneado a un rival, cuándo ha sido impactado por un misil enemigo, etc.
- ***De reconocimiento del campo:*** Permiten al robot conocer las circunstancias de la partida. Pueden saber el tamaño del tablero, su posición en este, su propio tamaño, en qué dirección se mueve él u otro enemigo, la energía que le queda, el número de enemigos con vida, etc.

Estructura de un robot

Un robot tiene definido su comportamiento en un fichero .java, que conforma una clase cuyo nombre corresponde al nombre del robot.

Los robots funcionan por eventos; a cada evento sucedido durante el desarrollo de la batalla, el robot reacciona según esté programado. *Robocode* provee la clase abstracta *_RobotBase*, que a su vez es extendida por una serie de subclases que amplían sus funciones. Estas funciones dotan al robot de herramientas que le permiten moverse y estudiar el campo de batalla y sus circunstancias. Todas estas clases ya vienen

desarrolladas, por lo que a la hora de crear un robot, solo es necesario crear su propia clase y hacer uso de la funcionalidad ya proporcionada por el sistema.

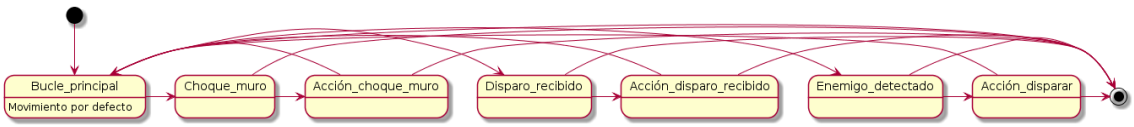


Ilustración 35 – Diagrama de estados que representa el funcionamiento de un robot

En el diagrama de estados de la ilustración 35 se ven representados los eventos que han sido tenidos en cuenta a la hora de desarrollar el funcionamiento de los robots. Para cada evento, se codifica una acción a realizar por parte del tanque robot.

El estado final, por su parte, representa el final de la actividad del robot. A este momento, el robot puede llegar vivo –significa que ha ganado-, o muerto.

El bucle principal es el que se ejecuta mientras no suceda ninguno de los eventos que activen las diferentes acciones que realiza el robot.

La estructura de las clases que definen todo el comportamiento de un robot es la siguiente:

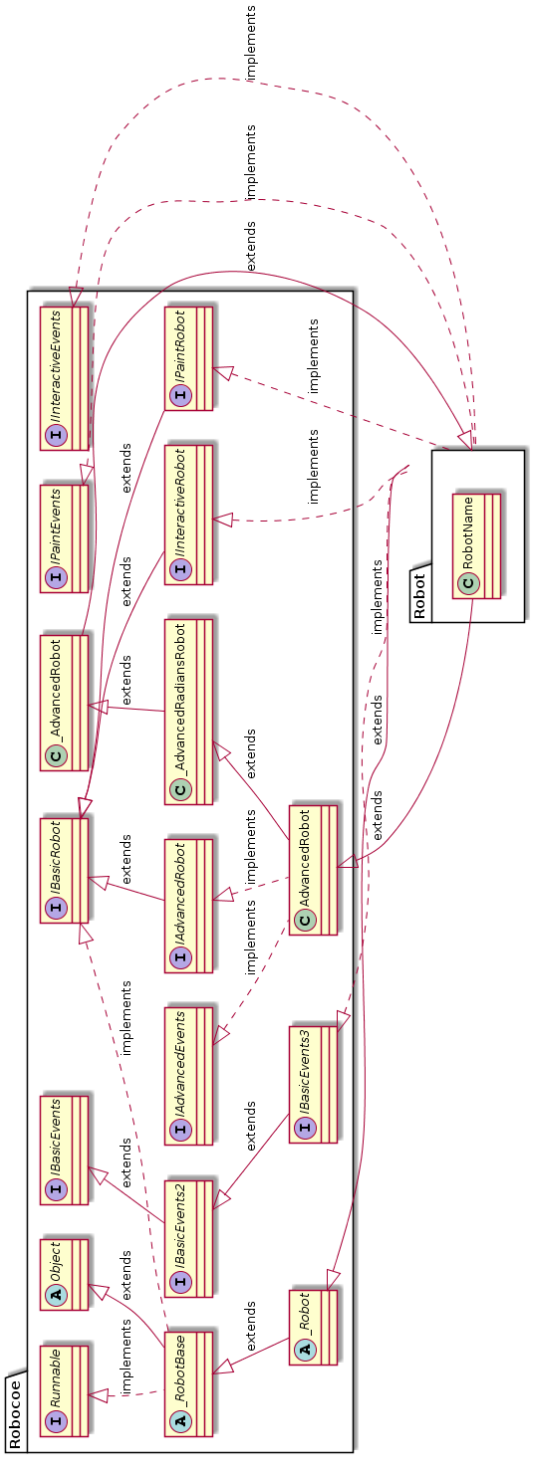


Ilustración 36 – Diagrama de clases para un robot

Estas clases son las que proveen las funciones explicadas en el apartado anterior.

5. Evaluación del trabajo

En esta sección se va a valorar el trabajo realizado desde dos puntos de vista: primeramente se va a comprobar que el sistema cumple los objetivos propuestos, analizando tanto el comportamiento de los robots como la evolución de las puntuaciones de los usuarios en relación con su desempeño en *Bolotweet*; por otro lado, se va a comprobar cuánto trabajo se ha invertido de manera cuantitativa, mediante métricas de ingeniería del software.

5.1. Valoración del funcionamiento del sistema

Hasta ahora han sido explicadas la arquitectura de la solución y una serie de directrices básicas que describían el funcionamiento del sistema.

A lo largo de la memoria se han ido exponiendo las soluciones desarrolladas y su finalidad. Por ejemplo, cuando se ha definido el comportamiento de los robots, se ha indicado que la idea reside en que el robot de mayor nivel gane con suficiencia las partidas, pues se considera justo que el alumno que ha realizado un mejor trabajo en *Bolotweet* reciba una bonificación que justifique el esfuerzo que ha realizado. Sin embargo, también se pretende que ciertas batallas, a priori desniveladas, presenten resultados más ajustados. Así, estas batallas generarán más interés en los estudiantes, motivándoles a prestar atención a la metodología aplicada a la asignatura.

En este apartado se va a comprobar que:

- El sistema actualiza correctamente los resultados de las batallas en *Bolotweet*, para que los alumnos puedan consultar siempre que quieran su posición en la clasificación general.
- La solución desarrollada responde al planteamiento inicial del sistema. Para ello, van a ser planteados diferentes escenarios con las siguientes características:
    - ***Primer caso:*** Se pretende estudiar el resultado de las batallas con robots de distintos niveles, para comprobar que los distintos niveles están equilibrados. Para ello, van a ejecutarse 20 batallas, de tres rondas cada una, entre dos robots, uno de nivel 2 y otro de nivel 3. Posteriormente va a repetirse esta mecánica con los robots de niveles 1 y 2. Por último, se ejecutarán 20 *roborumbles* con robots de niveles variados.
    - ***Segundo caso:*** Se pretende estudiar el funcionamiento global del sistema en un grupo, manteniendo un ranking de usuarios y realizando enfrentamientos regularmente, como sucedería en un grupo real de clase.

5.1.1. Primer caso

El resultado esperado es aquel en el que la inmensa mayoría de las batallas, pudiendo ser todas, son ganadas por el robot de mayor nivel por bastante diferencia. Sin embargo, también se espera que en unas pocas batallas esta diferencia sea mínima e, incluso, sea el de menor nivel el robot ganador.

Primer enfrentamiento: 20 batallas 1v1 entre robot de nivel 2 vs robot de nivel 3

Batallas	Puntos N3	Rondas ganadas N3	Puntos N2	Rondas ganadas N2	Puntos N3 - Puntos N2
1	460	3	18	0	442
2	339	1	195	2	144
3	572	3	102	0	470
4	448	3	51	0	397
5	537	3	137	0	400
6	371	2	128	1	243
7	314	1	232	2	82
8	357	2	116	1	241
9	323	2	156	1	167
10	318	2	84	1	234
11	437	2	95	1	342
12	384	2	194	1	190
13	292	1	239	2	53
14	376	2	96	1	280
15	398	2	161	1	237
16	226	3	92	0	134
17	352	2	103	1	249
18	533	3	44	0	489
19	526	3	75	0	451
20	487	3	67	0	420

Tabla 2 - Resultados de 20 batallas entre dos robots de niveles 2 y 3

Como puede observarse en la tabla 2, la superioridad del robot de nivel 3 es notoria. En ninguna de las batallas la puntuación del robot de nivel 2 supera al del 3, y en solo tres batallas ha ganado más rondas el robot de menor nivel, siendo además el resultado de estas batallas muy ajustado.

Se han calculado también las diferencias entre las puntuaciones de los robots. De media, el robot de nivel 3 supera en alrededor de 283 puntos al robot de nivel 2. Se considera que es una magnitud aceptable, teniendo en cuenta que un robot recibe 60 puntos por ganar una ronda (50 por ganar + 10 por robot al que sobrevive) y dos puntos por cada punto de daño efectuado, además de varios bonus por la manera en la que ha atacado al rival. Si en la mayoría de casos obtiene 120 o 180 puntos solo por ganar dos o tres rondas, el robot de nivel 3 obtiene alrededor de 100 puntos por daño infringido al rival.

Segundo enfrentamiento: 20 batallas 1v1 entre robot de nivel 1 vs robot de nivel 2

Batallas	Puntos N2	Rondas ganadas N2	Puntos N1	Rondas ganadas N1	Puntos N2 - Puntos N1
1	357	1	191	2	166
2	583	2	157	1	426
3	466	2	105	1	361
4	470	2	106	1	364
5	453	2	99	1	354
6	540	2	164	1	376
7	740	3	40	0	700
8	340	2	172	1	168
9	645	2	163	1	482
10	447	2	85	1	362
11	292	2	98	1	194
12	505	2	134	1	371
13	233	1	226	2	7
14	583	3	280	0	303
15	547	2	124	1	423
16	639	3	92	0	547
17	425	2	107	1	318
18	505	3	24	0	481
19	794	3	38	0	756
20	342	2	80	1	262

Tabla 3 - Resultados de 20 batallas entre dos robots de niveles 1 y 2

Tras enfrentar 20 veces los robots de niveles 1 y 2, se aprecia de nuevo que el robot de nivel 2 gana con solvencia sus enfrentamientos. Solo en dos ocasiones el robot de nivel 1 ha ganado más rondas que el de nivel dos, y los resultados fueron, de nuevo, muy ajustados. Y, además, en ninguna batalla la puntuación del robot inferior supera a la puntuación del robot de mayor nivel.

La media de las diferencias, en este caso, ha sido de 371 puntos, de nuevo un valor aceptable.

Tercer enfrentamiento: 20 roborumbles con robots de niveles variados

En los 20 *roborumbles* ejecutados participaban un robot de nivel 3, dos robots de nivel 2 y tres robots de nivel 1. En este tipo de batallas hay que tener en cuenta que, al haber más robots, los resultados se esperan que sean más abiertos, pues un robot de mayor nivel puede recibir el impacto de una bala perdida, cuyo objetivo original fue errado. Este factor de aleatoriedad, sin embargo, no volverá las partidas imprevisibles; se seguirá observando una superioridad por parte del robot de mayor nivel, aunque no tan acusada como en batallas con menos contrincantes.

Batallas	Puntos N3	Rondas ganadas N3	Puntos N2	Rondas ganadas N2	Puntos N1	Rondas ganadas N1
1	1603	3	1536	0	1210	0
2	1814	3	1240	0	1259	0
3	1807	3	1175	0	1299	0
4	1417	2	1419	0	1473	1
5	1943	3	920	0	1585	0
6	1377	2	1412	0	1547	1
7	1281	2	1429	0	1818	1
8	1334	1	1802	1	1245	1
9	1884	3	1166	0	1314	0
10	1518	1	2160	1	1381	1
11	1556	1	1974	2	1050	0
12	1804	2	1284	0	1266	1
13	1473	3	1403	0	1512	0
14	1643	3	1585	0	1154	0
15	1242	2	2222	0	1088	1
16	1431	2	1658	0	1441	1
17	1797	3	1751	0	833	0
18	987	1	1938	1	1443	1
19	1944	3	1574	0	886	0
20	893	1	1147	0	2373	2

Tabla 4 - Resultados de 20 roborumbles

Como se observa, el nivel 3 vuelve a dominar las batallas. Hay que tener en cuenta que los valores de las columnas de puntuación y rondas ganadas de los niveles 2 y 3 son la suma de las de todos los robots de ese nivel. Esto muestra que un solo robot de nivel 3 puede, en la mayoría de los casos, obtener puntuaciones semejantes a la suma de todos sus contrincantes y, en muchos casos, superarlas.

Además, de un total de 60 rondas, el robot de nivel 3 ha ganado 44, por 5 entre todos los robots de nivel 2 y 11 entre los robots de nivel 1. Que haya más victorias de robots de nivel 1 que de nivel 2 se debe a que había más robots del nivel inferior.

Por tanto, los desempeños individuales de los robots de nivel inferior quedan por debajo del robot de nivel 3. Solo en 5 batallas, el robot de mayor nivel no ha conseguido ganar, al menos, dos de tres rondas. Además, de entre esas 5 batallas, solo en 2 se ha visto superado por algún rival.

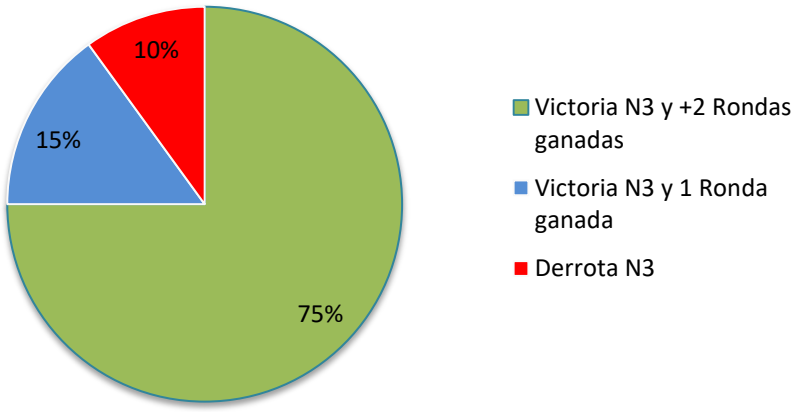


Ilustración 37 – Resultados en porcentaje de 20 roborumbles

Como puede apreciarse en las tablas 5 y 6, en las dos batallas en las que no ha ganado el robot de nivel 3, quedó en ambas en segunda posición.

Posición	Robot	Puntos	Rondas ganadas
1	María	1283	0
2	Andrea	987	1
3	Lucía	655	0
4	Sergio	553	0
5	Paco	461	1
6	Susana	429	1

Posición	Robot	Puntos	Rondas ganadas
1	Paco	1021	1
2	Andrea	893	1
3	Sergio	793	1
4	María	588	0
5	Lucía	559	0
6	Susana	559	0

Tablas 5 y 6 - Resultados de las batallas 18 y 20

Conclusiones

Atendiendo a los enfrentamientos que se han ejecutado, se pueden obtener las siguientes conclusiones:

- En enfrentamientos 1v1, el robot de nivel superior domina y gana con claridad la amplia mayoría de las rondas de una batalla. Cuando no lo hace, el resultado total de la batalla sigue siendo favorable para este.

- En *roborumbles* con robots de niveles dispares es más posible que el robot de mayor nivel no gane. Estas situaciones no son las más comunes. Sin embargo, es un hecho aceptado y que dota a la competición de una imprevisibilidad que aumenta el interés de los alumnos en la mecánica.

5.1.2. Segundo caso

Se va a simular el uso continuado del sistema por parte de 30 alumnos pertenecientes al grupo imaginario llamado *EDA*. Estos alumnos irán respondiendo a las tareas que el profesor vaya proponiendo, serán puntuadas y, en base a las calificaciones obtenidas, la inteligencia de sus robots variará.

Se ejecutarán batallas cada dos tareas respondidas y se comprobará que los resultados de estas se ajustan al rendimiento de los alumnos dentro de la herramienta *Bolotweet*; los alumnos con mejores medias obtendrán mejores resultados en dichas batallas.

Primer roborumble

Para empezar, teniendo todos los robots nivel 1, se realiza un roborumble de 3 rondas con un tablero de dimensiones 800x600.

```
-- Battle has completed --
Battle results:
  silvia*: 4327 0
  david*: 4106 0
  raquel*: 3847 0
  adriana*: 3696 0
  oscar*: 3652 0
  daniel*: 3500 1
  victor*: 3414 0
  juan*: 3408 0
  manuel*: 3345 1
  mario*: 3136 0
  pablo*: 3111 0
  ruben*: 2988 0
  ivan*: 2874 0
  aitor*: 2841 0
  jesus*: 2716 0
  fernando*: 2664 0
  virginia*: 2452 0
  jaime*: 2336 0
  elena*: 2310 1
  cintia*: 2189 0
  sandra*: 2171 0
  carlos*: 2123 0
  miriam*: 2113 0
  julia*: 1959 0
  beatriz*: 1943 0
  blanca*: 1822 0
  alberto*: 1709 0
  eva*: 1688 0
  ana*: 1192 0
  alexa*: 506 0
```

Ilustración 38 – Resultados del primer roborumble

Los resultados visibles en la ilustración 38, que al ser todos los robots de nivel 1 son un tanto aleatorios, son arrojados por el propio sistema. Posteriormente son volcados a la página de *Bolotweet*. Este proceso se realiza de manera automática gracias al sistema *Crontab*, o bien el propio administrador puede acometer esta tarea manualmente.

En la ilustración 39 puede apreciarse que los resultados están a la vista para todos los usuarios pertenecientes al grupo, en la pestaña *Robocode*.

Ilustración 39 – Resultados del primer roborumble en Bolotweet

Segundo roborumble y torneo

Los estudiantes han contestado a su primera tarea, por lo que las medias de sus calificaciones han variado. Como solo se ha respondido a una tarea, la media es igual a la puntuación obtenida en esta primera tarea.

Han respondido 23 de los 30 alumnos. Por tanto, los 7 alumnos que no han contestado mantienen el nivel 1 de sus robots, mientras que los siguientes obtienen mejores robots en caso de que las calificaciones obtenidas superen el 1.

Ilustración 40 – Ejemplo de mensajes ya calificados por el grader Fran

Ejecutando de nuevo el sistema, las medias varían en el caso de los alumnos que hayan obtenido más de un 1 en la calificación de su primera tarea.

Por ejemplo, en las ilustraciones 40 y 41 vemos que Miriam ha obtenido un 3 en su tarea, por lo que le corresponde un robot de nivel 3.

```
agregando: robots/ruben.java(entrada = 4698) (salida = 1325)(desinflado 71%)
El nivel del robot miriam es 3
manifiesto agregado
agregando: robots/miriam.java(entrada = 2837) (salida = 1248)(desinflado 56%)
El nivel del robot daniel es 1
manifiesto agregado
agregando: robots/daniel.java(entrada = 4701) (salida = 1331)(desinflado 71%)
```

Ilustración 41 – Actualización de los robots al comienzo del programa

El profesor decide realizar otro *roborumble*. Posteriormente, los 16 mejores robots participarán en un torneo basado en enfrentamientos 1v1, en el que un robot irá avanzando rondas conforme gane sus batallas, y quedará eliminado al perder.

Tras ejecutar el *roborumble* eliminatorio, se comprueba si los robots de mayor nivel han quedado clasificados para la ronda final o si, por el contrario, alguno ha caído antes de lo previsto.

En la tabla 7 pueden observarse los resultados de este segundo *roborumble*.

Posición	Nombre	Nivel	Puntos
1	Mario	3	3934
2	Carlos	3	3886
3	Julia	3	3784
4	Alexa	3	3738
5	Jesús	1	3406
6	Beatriz	3	3371
7	Silvia	3	3321
8	Miriam	3	3305
9	Fernando	1	3293
10	Alberto	1	2905
11	Cintia	1	2851
12	Sandra	1	2745
13	David	1	2693
14	Elena	1	2684
15	Aitor	1	2683
16	Ana	3	2631
Eliminados			
21	Raquel	3	2380
22	Eva	3	2353

Tabla 7 - Clasificación del segundo roborumble

Solo dos robots de nivel 3 –de diez robots totales- han quedado fuera: Raquel y Eva. Atendiendo a sus puntuaciones, no han estado lejos de clasificarse. En un *roborumble* de 30 robots en un tablero un tanto pequeño, es más factible que se den ciertos resultados inesperados.

En la ilustración 42 puede comprobarse cómo ocupan los espacios los 30 robots participantes:

Ilustración 42 – Roborumble de 30 robots en un tablero de 800x600

Como se ha venido señalando a lo largo de la memoria, los escenarios en los que se producen resultados sorprendentes también ayudan a mantener el interés de los estudiantes en la partida.

Ahora se lleva a cabo el torneo. Se empareja al primer robot con el último, al segundo con el penúltimo y así sucesivamente. Las batallas son de tres rondas salvo la final, que se juega a cinco rondas.

El resultado final es el siguiente:

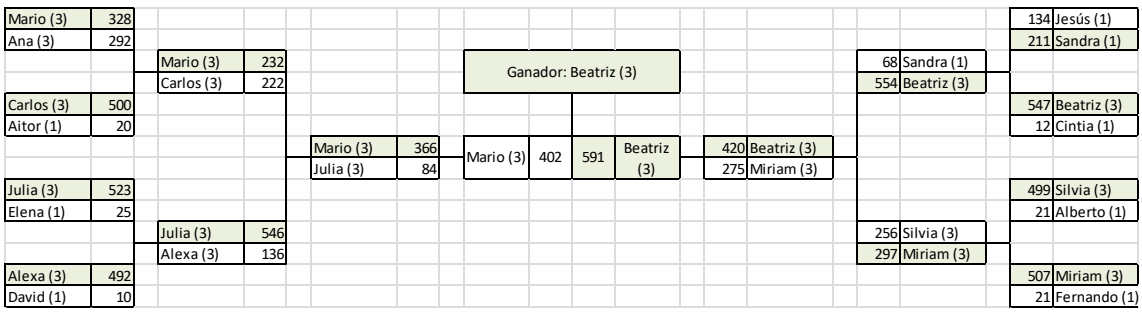


Ilustración 43 – Ejemplo de un torneo de 16 robots

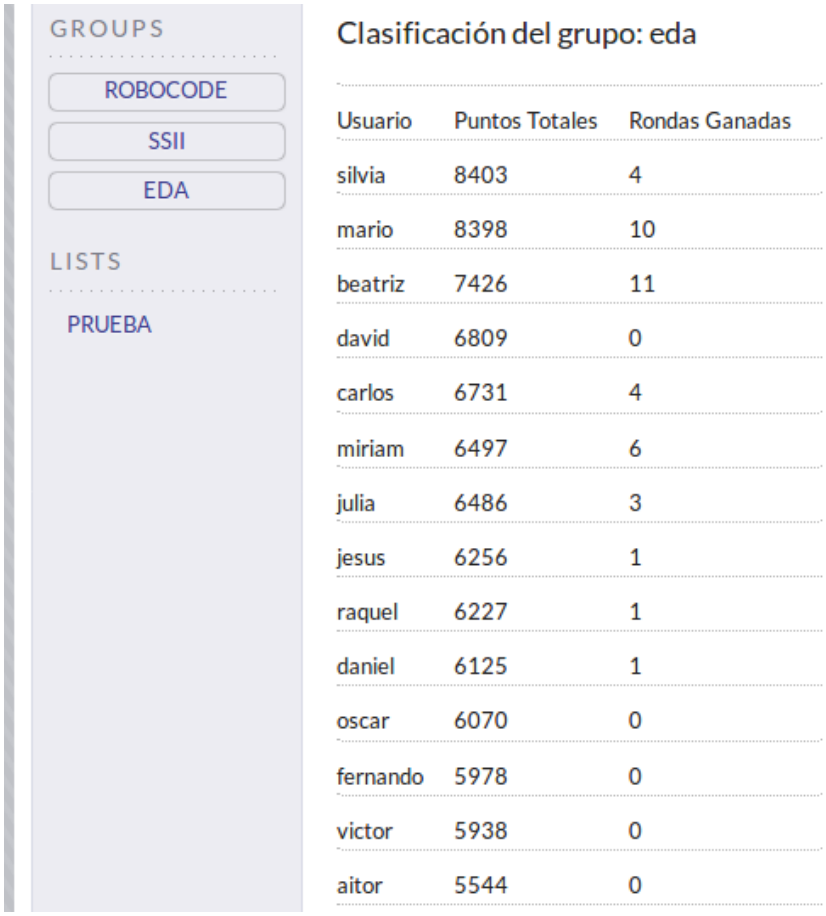
De este resultado pueden obtenerse dos conclusiones:

- La diferencia entre un robot de nivel 3 y otro de nivel 1 es enorme. Gana con mucha solvencia el robot de mayor nivel. Esto premiará a los estudiantes que

trabajen en la plataforma sobre aquellos que no muestren interés en esta herramienta.

- Los enfrentamientos entre robots de nivel 3 son más imprevisibles. Algunos son muy ajustados –como el enfrentamiento entre Mario y Carlos, donde la diferencia ha sido de 10 puntos-, y en otros uno de los robots machaca al enemigo –como en el enfrentamiento entre Julia y Alexa, donde la diferencia ha sido de 410 puntos-. Esto cumple el propósito de ejecutar batallas cuyo resultado, a priori, no es presumible.

Finalmente, se actualizan las puntuaciones de los robots, y estos resultados actualizados aparecen en la página principal del plugin en *Bolotweet.*



Usuario	Puntos Totales	Rondas Ganadas
silvia	8403	4
mario	8398	10
beatriz	7426	11
david	6809	0
carlos	6731	4
miriam	6497	6
julia	6486	3
jesus	6256	1
raquel	6227	1
daniel	6125	1
oscar	6070	0
fernando	5978	0
victor	5938	0
aitor	5544	0

Ilustración 44 – Resultados acumulados tras dos roborumbles y un torneo

Como puede apreciarse en la ilustración 44, los robots que más lejos han llegado en el torneo acumulan más puntos y rondas ganadas.

Mario y Beatriz, los finalistas, destacan por su número de rondas ganadas. Al haber llegado tan lejos, han acumulado más puntos y se han colocado, respectivamente, en segunda y tercera posición.

Sin embargo, Silvia mantiene la primera posición porque en el torneo alcanzó los cuartos de final y aumentó su puntación, que previamente al segundo *roborumble* era la más alta debido a que fue la ganadora del primero.

Por otro lado David se mantiene cuarto, pese a haber caído en octavos de final en el torneo y habiendo obtenido solo 10 puntos. Su alta posición se debe a que en el primer *roborumble* quedó segundo y en el siguiente clasificó entre los 16 primeros.

Primeras tareas y batallas

A lo largo del curso, el profesor inicia sucesivas tareas que son completadas por los estudiantes, y posteriormente calificadas. Las medias de los alumnos siguen una tendencia: los alumnos que completan con asiduidad las tareas tienen medias superiores a los que no utilizan *Bolotweet*.

El profesor ha ejecutado 10 *roborumbles* sin posterior torneo, tras haber propuesto tres tareas. En la tabla 8 se presenta la clasificación de los 16 primeros robots tras haber participado en estas batallas, junto con la media del usuario, el nivel del robot y el número de tareas completadas.

Posición	Nombre	Media del usuario	Nivel	Tareas completadas (sobre 3)	Puntos
1	Alexa	2,33	3	3	42789
2	Mario	2,66	3	3	41619
3	Beatriz	2,66	3	3	41532
4	Pablo	2,33	3	3	38773
5	Raquel	2,33	3	3	38651
6	Miriam	3	3	3	38150
7	Julia	2,66	3	3	37477
8	Manuel	2,33	3	3	37437
9	Silvia	2,66	3	3	36978
10	Blanca	2,66	3	3	36002
11	Eva	2,33	3	3	35781
12	Sandra	2,33	3	3	35742
13	Óscar	1	1	3	32394
14	Ana	2,33	3	3	32116
15	Daniel	0	1	0	30252
16	Carlos	2,66	3	3	30025

Tabla 8- Clasificación tras 12 roborumbles y un torneo

El primer aspecto a destacar es que todos los usuarios con robot de nivel 3 han quedado entre los 16 primeros. Es la recompensa que reciben los estudiantes por completar las tareas propuestas en *Bolotweet*.

Las dos plazas restantes en este top 16 son ocupadas por dos robots de nivel 1. Es una de las sorpresas que puede deparar un *roborumble*. Cabe destacar también que los robots tienen el mismo nivel, pero por distintos motivos: en el caso de Daniel, no ha completado ninguna tarea; sin embargo, Óscar sí ha completado las tres propuestas,

pero sus respuestas no han superado los criterios de evaluación del profesor y, por tanto, la media de estas es de 1.

Entre estos 16 robots se realiza otro torneo con las mismas características que el anterior.

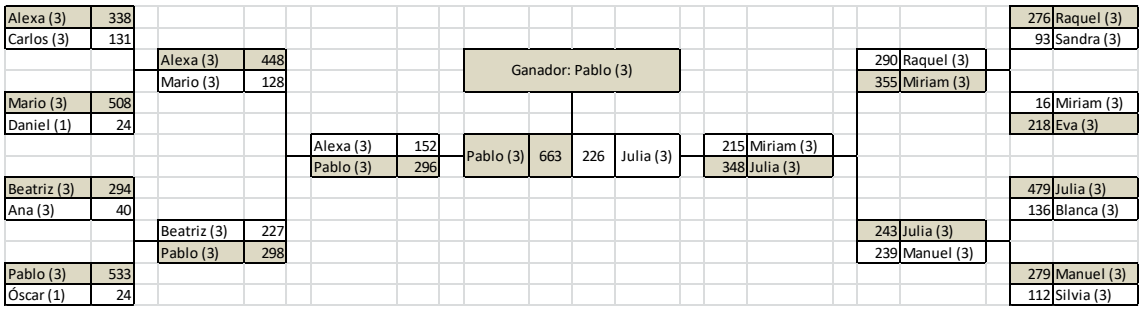


Ilustración 45 – Ejemplo de un torneo de 16 robots

Los resultados son similares respecto al torneo realizado con anterioridad.

Los robots de nivel 3 han vuelto a machacar a los de nivel 1. En cuanto a enfrentamientos de robots de mismo nivel, se han visto batallas muy igualadas –Julia ha superado a Manuel por 4 puntos-, y otras muy desigualadas –Beatriz ha superado a Ana por 254 puntos-.

Ilustración 46 – Resultados acumulados tras 12 roborumbles y 2 torneos

Actualizando los resultados en *Bolotweet*, comprobamos que cada vez destacan más los estudiantes con mejor desempeño en la plataforma –salvo las ya mencionadas excepciones-. Por tanto, los estudiantes con peores notas quedan relegados a la parte baja de la tabla. Sin embargo, como se mostrará a continuación, el sistema permite a los estudiantes volver a tener robots competitivos si retoman el trabajo.

Se entiende que esta tendencia se mantendrá si los estudiantes que más destacan continúan completando las tareas. Como la media se calcula en función de las calificaciones obtenidas en un periodo de días indicado por el profesor, si un alumno deja de participar en la plataforma, su media se verá devaluada y, por tanto, su robot empeorará.

Así, se motiva al estudiante a seguir completando tareas. En el momento en el que este deja de completar tareas, perderá posiciones en la clasificación.

Sucesivas tareas y batallas

Conforme avanza el curso, se siguen completando tareas y batallas.

Puede darse el caso de que el desempeño de ciertos alumnos sea irregular: algunos pueden dejar de completar tareas y, por el contrario, otros pueden comenzar a participar en la herramienta pese a haber dejado sin completar las primeras.

En estos casos, *Bolocode* está diseñado para premiar a este segundo grupo de alumnos. Sus robots mejorarán y, por tanto, podrán competir por las primeras plazas de los *roborumbles* para acceder a futuros torneos. Por otro lado, los alumnos que han dejado de trabajar dejarán de participar en los torneos pues tendrán más difícil clasificarse para estos.

Por ejemplo, las estudiantes Cintia y Virginia, cuyo desempeño ha sido inferior al de la mayoría de los alumnos, comienzan a completar tareas y recibir mejores calificaciones. Por otro lado, Raquel y Pablo, que estaban en el top 5, dejan de trabajar.

Las estadísticas iniciales de estos cuatro alumnos eran las siguientes:

Posición	Nombre	Nivel	Media	Tareas completadas (sobre 3)	Puntos
4	Pablo	3	2,3	3	40563
5	Raquel	3	2,3	3	39034
20	Virginia	1	0	0	28248
21	Cintia	1	1	2	27721

Tabla 9 - Puntuaciones de varios alumnos cuyo rendimiento va a ser controlado

Tras haber sido propuestas cuatro tareas más –sin haber ejecutado ningún *roborumble-*, estas son las nuevas estadísticas de estos cuatro alumnos:

Posición	Nombre	Nivel	Media	Tareas completadas (sobre 7)	Puntos
4	Pablo	1	0	3	40563
5	Raquel	1	0	3	39034
20	Virginia	3	2,5	4	28248
21	Cintia	3	2,5	6	27721

Tabla 10 - Puntuaciones actualizadas de los alumnos controlados

Las medias de Virginia y Cintia aumentan puesto que el número de días que se tienen en cuenta a la hora de calcularlas es 3. Asimismo, como Pablo y Raquel dejan de contestar, en los últimos días no han completado ninguna tarea, por lo que su media es 0.

Se asume que, a partir de este punto, el trabajo de los alumnos se mantiene constante. Tanto aquellos que han venido trabajando y obteniendo buenos resultados como aquellos que no mantendrán sus respectivas dinámicas.

El profesor lanza, a lo largo del curso, quince *roborumbles* más. Así queda la clasificación tras estas batallas:

Posición	Nombre	Nivel	Media	Puntos
1	Alexa	3	2,5	91586
2	Julia	3	2,25	84273
3	Sandra	3	2,25	83974
4	Beatriz	3	2,75	81971
5	Óscar	3	2,5	80559
6	Blanca	3	2,5	80028
7	Iván	3	2,5	79837
8	Manuel	3	2,5	79427
9	Ana	3	2,5	79081
10	Eva	3	2,25	77127
11	Silvia	3	2,25	76234
12	Jaime	3	2,25	73186
13	Virginia	3	2,5	71325
14	Fernando	3	2,75	70838
15	Cintia	3	2,5	70675
16	Pablo	1	0	70514
Eliminados				
17	Carlos	3	2,75	70263
18	Raquel	1	0	70063
20	Miriam	0	0	68487
24	Mario	2	2	63749

Tabla 11 - Clasificación tras 25 roborumbles y dos torneos

Cintia y Virginia han alcanzado el top 16, así como lo ha hecho Pablo, valiéndose de su puntuación previa a los *roborumbles* y el factor suerte existente en estas batallas. Raquel, por otro lado, sí ha caído de este top, así como también lo ha hecho Carlos pese a tener un robot de nivel 3.

También han caído Miriam y Mario, cuyo trabajo ha empeorado en *Bolotweet*.

En la siguiente tabla se presenta la evolución de las puntuaciones de los cuatro alumnos cuyos rendimientos están siendo controlados:

Nombre	Diferencia de posición	Puntos obtenidos
Pablo	-12	29951
Raquel	-13	31029
Virginia	7	43077
Cintia	6	42954

Tabla 12 - Evolución de los alumnos controlados

El profesor decide realizar un tercer torneo. De nuevo, siguiendo las reglas y estructura de los dos anteriores.

El resultado de este torneo es el siguiente:

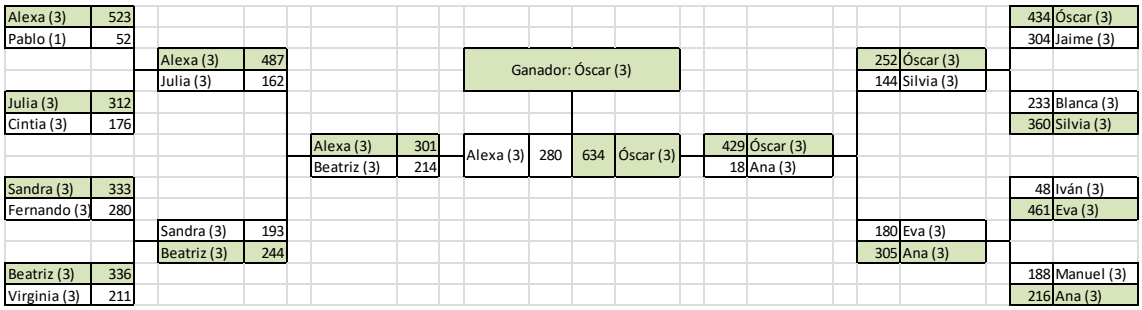


Ilustración 47 – Ejemplo de un torneo de 16 robots

Vuelve a haber superioridad del nivel 3 y, al enfrentarse entre ellos, los resultados finales son dispares. Por ejemplo, Ana ha superado a Manuel por 28 puntos, mientras que Eva ha hecho lo propio contra Iván por 413 puntos.

Por último, se comprueban las puntuaciones acumuladas actualizadas:

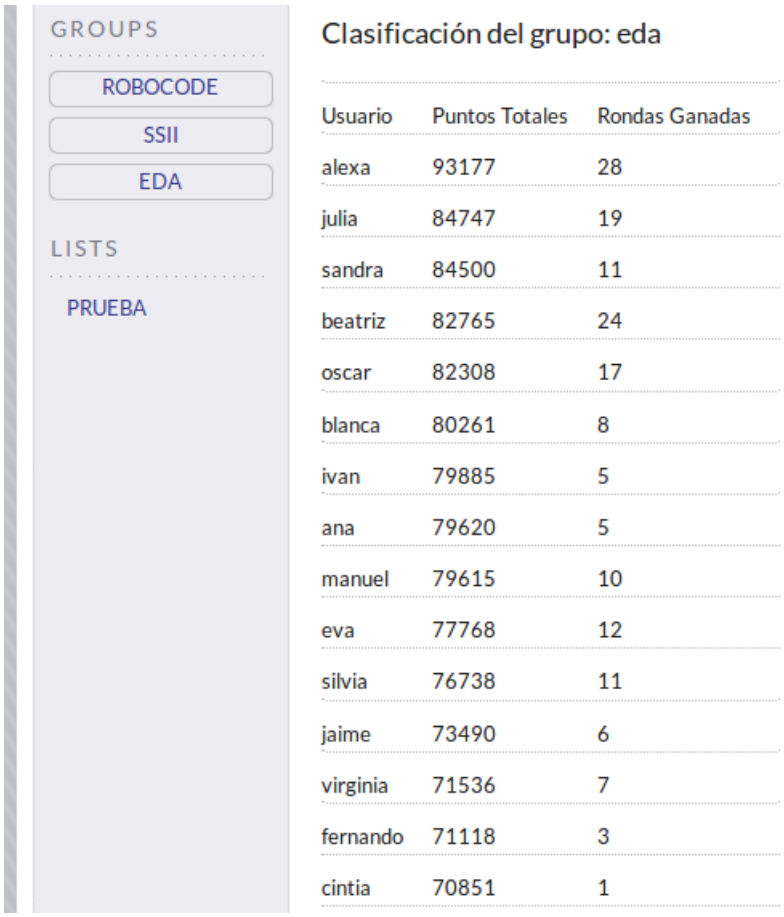


Ilustración 48 – Resultados acumulados tras 27 roborumbles y tres torneos

Virginia y Cintia han visto recompensado su trabajo con una participación en el torneo. Sin embargo, al haber caído en octavos de final, no han podido escalar posiciones.

En el siguiente gráfico puede apreciarse la evolución de los alumnos que han quedado entre los 16 primeros a lo largo de la monitorización del sistema.

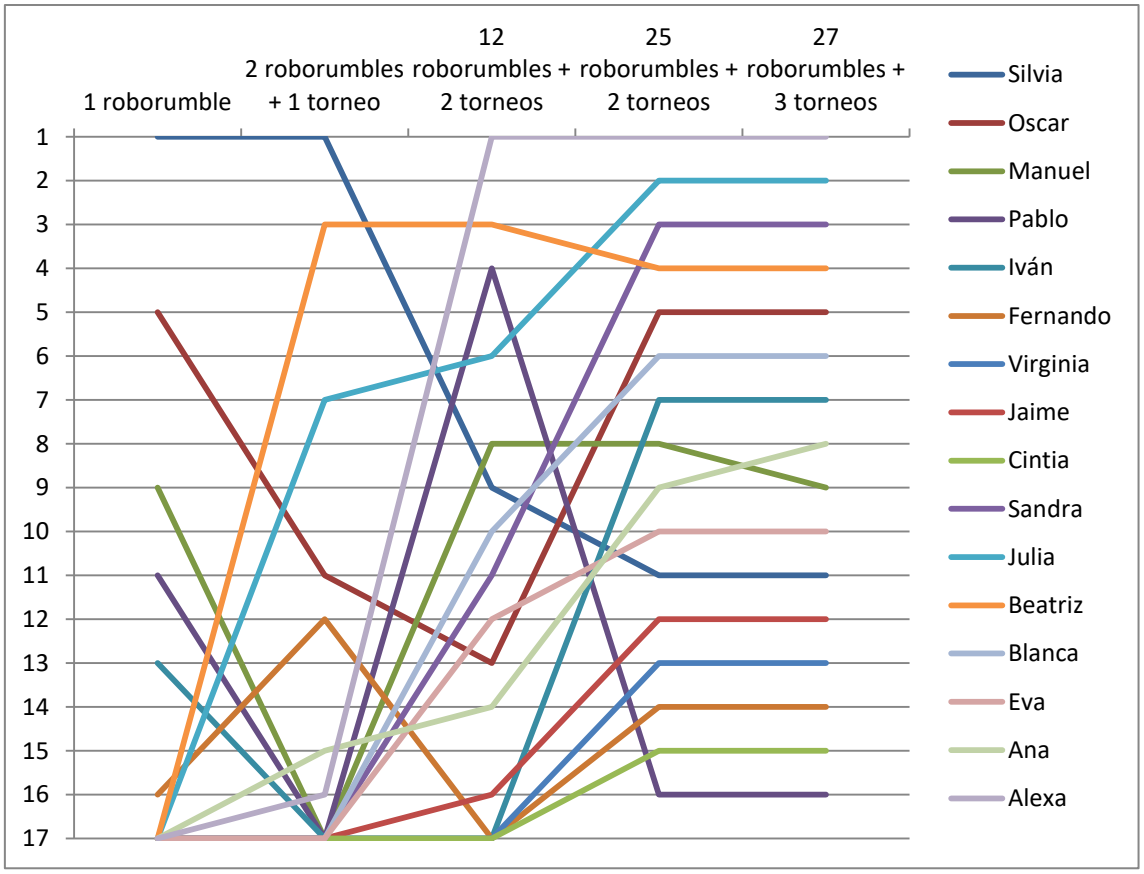


Ilustración 49 – Evolución de los puestos de los 16 mejores estudiantes

Como puede apreciarse en la gráfica anterior, la posición que ocupa un usuario en el ranking fluctúa según qué calificaciones obtiene en sus tareas, pudiendo tanto ascender en caso de ir mejorando el contenido de sus mensajes como descender en caso de recibir peores notas.

Este hecho prueba que, para tener un buen puesto al final del curso, es necesario participar de forma regular en la plataforma y, además, que nunca es tarde para reengancharse. El sistema premia a los estudiantes con los mejores desempeños más recientes, en detrimento de aquellos que comienzan el curso obteniendo buenas calificaciones para posteriormente abandonar el uso de *Bolotweet.*

Conclusiones

Observando el desarrollo del uso del sistema y cómo han evolucionado las puntuaciones de los alumnos pertenecientes a un grupo cuyo comportamiento puede compararse al de un grupo real, las conclusiones son las siguientes:

- El sistema premia a los estudiantes que más y mejor trabajan en la plataforma; no solo importa completar las tareas, sino que el contenido de las mismas ha de superar los criterios del profesor.
- El sistema permite a los estudiantes reengancharse a la lucha por quedar arriba en la clasificación. Cuando acumulan una serie de calificaciones positivas, su

desempeño en los *roborumbles* y combates individuales aumenta, provocando que su puntuación acumulada aumente en mayor medida de lo que solía hacer cuando el nivel del robot era inferior.

- El aumento no es instantáneo; es necesario mantener un ritmo de trabajo constante durante cierto tiempo para poder avanzar y destacar en la clasificación. Como se ha visto, han sido necesarios 15 *roborumbles* para que varios estudiantes puedan clasificarse para el torneo. Dependiendo del número de tareas que un profesor mande, y también del número de batallas que ejecute, un estudiante necesitará más o menos tiempo para avanzar.
- Las batallas entre robots del mismo nivel pueden deparar resultados muy alternos; es igual de posible que el resultado final sea muy ajustado como que uno de los dos enemigos aplaste al rival.
- El sistema actualizar correctamente en *Bolotweet* los resultados acumulados de las puntuaciones de los usuarios.

5.2. Valoración del esfuerzo realizado

Bolocode se sustenta en *Bolotweet*, un sistema ya existente sobre el cual se ha añadido una nueva funcionalidad consistente en el mantenimiento de los marcadores de las batallas ejecutadas en *Robocode*.

Para ello, se ha desarrollado un nuevo *plugin*, *RoboPlugin*, el cual consta de los siguientes códigos PHP:

- *GetScoresAction.php*
- *Robocode.php*
- *Robocodegroups.php*
- *RoboPlugin.php*

Ha sido necesario desarrollar una serie de scripts para administrar el sistema y para sustentar la integración de las dos plataformas base. Esto scripts son:

- *Checkrobocoderobot.php*
- *GetGroupID.php*
- *Getrobocodegroup.php*
- *ListGroup.php*
- *Setrobocodetogroup.php*
- *Updateallrobocodescores.sh*
- *Updategradesfile.php*
- *Updategradesfileandshow.sh*

Otros scripts y *plugins* ya existentes han sido modificados. Sin embargo, no han sido contabilizados dado que la herramienta de medición no puede medir trozos concretos de un archivo.

También se han desarrollado tres scripts para proporcionar funcionalidad a todo el sistema:

- *Bolocode.sh*
- *Crearrobots.sh*
- *GeneraBatalla.sh*

Para poder ejecutar batallas en *Robocode* de manera externa a la aplicación y con parámetros personalizados, ha sido necesario crear un programa Java que se valiese de la API que provee el sistema para generar batallas:

- *BattleRunner.java*

Por último, ha sido necesario desarrollar el código de los robots que participan en las batallas:

- *Nivel1.txt*
- *Nivel2.txt*
- *Nivel3.txt*

Se ha hecho uso del programa CLOC [23] para medir cuantitativamente todo el código generado. Los resultados obtenidos son los siguientes:

Lenguaje	Archivos	En blanco	Comentarios	Código
Bourne Shell	5	83	55	852
PHP	10	130	133	594
Java	4	94	149	340
Total	19	307	337	1786

Tabla 13 - Análisis cuantitativo del trabajo realizado

Otro aspecto a tener en cuenta es la configuración de las plataformas. La instalación de *Bolotweet* en un ordenador con sistema operativo Ubuntu 16.04 ha supuesto un esfuerzo inesperado a causa de las complicaciones surgidas a lo largo de dicha instalación.

Bolotweet se sustenta en la plataforma *Statusnet*, la cual dejó de existir para convertirse en *GNU Social*. Por tanto, el mantenimiento de esta ha sido abandonado y obtener ciertos recursos, tales como librerías o paquetes, no resultó una tarea sencilla. Fue necesario, entre otras cosas, acudir a repositorios no oficiales e instalar estos paquetes manualmente.

La plataforma funciona con la versión 5.6 de PHP y la 5.5.59 de *MySQL*. Estas versiones no son las más actuales que pueden obtenerse vía consola de comandos en Ubuntu, por lo que fue necesario realizar una instalación manual, descargándolas desde sus páginas oficiales e instalándolas mediante el comando *dpkg*. Además fue necesario adquirir paquetes de PHP que se encontraban en repositorios manejados por usuarios

independientes de la plataforma, dado que en la página oficial de PHP no aparecían para la versión 5.6. Los paquetes instalados fueron los siguientes:

- *Curl.*
- *XMLWriter.*
- *MySQL.*
- *GD.*
- *Mbstring.*

La instalación de PHP 5.6 requería ciertas dependencias que se solucionaban instalando más paquetes que, como ocurrió con los anteriormente mencionados, no se encontraban en la página oficial. Estos paquetes fueron:

- *Libapache2-mod.*
- *Fpm.*
- *Cgi.*
- *Common.*
- *Libssl1.1.*

Estos dos últimos paquetes no aparecían en el repositorio del cual se obtuvieron el resto, pero sí aparecieron en el repositorio oficial de Ubuntu.

Además, la instalación de estos paquetes requería una activación manual en el archivo de configuración de PHP.

Respecto a *MySQL*, la instalación manual requirió modificar el código fuente de *Bolotweet*, más concretamente el de la librería externa *mysqli*, pues era necesario indicar explícitamente el puerto y el *socket* al que se conecta el sistema.

Además, cabe destacar que antes de configurar correctamente las herramientas con las versiones mencionadas, se intentaron instalar las versiones actuales de *MySQL* y PHP. Existió un proceso paralelo en el que se intentó hacer funcionar el sistema con estas versiones, pero finalmente se descartó en cuanto se consiguió instalar el sistema con las versiones adecuadas para su funcionamiento.

También se probó a instalar la versión 14.04 de Ubuntu, con el fin de observar si las versiones antiguas de *MySQL* y PHP tenían mejor compatibilidad que con la versión 16.04 del sistema operativo, pero igualmente su uso quedó descartado.

Por último, como se ha mencionado, la extinta plataforma *Statusnet* se convirtió en *GNU Social*. También se probó a utilizar esta nueva plataforma, aunque hubiese sido una versión incompatible con *Bolotweet* y el problema fue solucionado antes de avanzar más en esta alternativa.

5.3. Participación de cada estudiante

Francisco Javier Pacheco Herranz

Ha realizado conjuntamente la instalación del sistema junto con Javier Romero: todo el proceso de estudio de la herramienta *Bolotweet*, obtención de recursos y configuración de los mismos.

Para la configuración de esta plataforma se estudiaron varias alternativas, como se ha mencionado en el apartado anterior. Francisco Javier se centró en la configuración que finalmente ha sido utilizada.

Fue el encargado de investigar repositorios hasta encontrar las librerías PHP que requería *Bolotweet* y que no se podían encontrar en su página oficial, instalarlas y comprobar qué dependencias existían respecto a otras librerías. También modificó el código fuente de *Bolotweet* para compatibilizarlo con la versión de *MySQL*. Para ello, depuró la ejecución de la instalación del sistema, estudió los mensajes que este proceso lanzaba, localizó desde dónde se lanzaban y así pudo observar dónde se encontraba el fallo.

Una vez fue instalado *Bolotweet* en una versión local, junto con Javier estudió su funcionamiento. Realizaron conjuntamente una serie de pruebas con el fin de tener un plugin funcional.

El primer plugin se basaba en un formulario que contenía un botón; dicho botón, al ser pulsado, aumentaba en uno un contador cuyo valor se alojaba en la base de datos del sistema, en una tabla creada específicamente para esa tarea. Tanto la codificación del plugin como las primeras consultas SQL fueron realizadas conjuntamente.

Codificó este plugin para adaptarlo a los resultados de las batallas ejecutadas en *Robocode*. Originalmente, la tabla contenía un usuario, sus puntos acumulados y el número de rondas que había ganado. Posteriormente, cada usuario contaba con una entrada en la tabla por cada grupo al que pertenecía, para diferencias sus puntuaciones según a qué grupo pertenecía la batalla.

También se encargó de desarrollar el código para mostrar los rankings en la página del plugin, ordenándolo por puntos acumulados. Se encargó, además, de codificar todas las consultas SQL necesarias para el correcto funcionamiento del sistema.

Tuvo que modificar parte del código de otros plugins, más concretamente del plugin *Grades*, para adaptarlo a las necesidades del nuevo. Hubo que añadir más consultas a la tabla *Grades* para obtener información pertinente para el cálculo del nivel asignado a los robots de los alumnos.

Generó un primer script para ejecutar batallas en *Robocode* de manera externa a la aplicación, pudiendo seleccionar qué robots participaban. Era un script primitivo que, por ejemplo, no permitía modificar el tamaño del campo de batalla, ni el número de

rondas. Se utilizó para almacenar los primeros resultados de las batallas y comprobar que se actualizaban correctamente en la base de datos de *Bolotweet*.

Este script fue actualizado por Javier, que le dotó de mayor funcionalidad, como se explicará en su correspondiente apartado. Cuando el script estuvo más avanzado, Francisco Javier se encargó de depurarlo y corregir todos los errores existentes, dotándole de robustez para evitar caídas del sistema en caso de darse errores tales como falta de archivos de configuración, introducción de entradas erróneas por parte del usuario, etc.

También realizó la investigación perteneciente al Estado del Arte. Buscó ejemplos de gamificación, estudió su puesta en funcionamiento, qué aspectos eran mejorables, cuáles eran inabarcables, etc. Estudió a los autores que han sido yendo mencionados a lo largo de la memoria, anotando directrices para realizar una buena gamificación y apuntado qué errores han de evitarse. De todo ello obtuvo una serie de conclusiones que han servido para estipular las bases del proyecto.

Ha desarrollado gran parte de los scripts de administración, necesarios para integrar los funcionamientos de ambas plataformas. También modificó los scripts ya presentes para adaptarlos a las necesidades del proyecto.

Por último, ha redactado gran parte de la memoria, centrándose en la introducción, el Estado del Arte, el desarrollo de la solución, las conclusiones y gran parte del tercer y quinto apartado, aparte de haber participado en la redacción de los manuales de uso, pertenecientes a los apéndices de esta memoria.

Conjuntamente desarrollaron el caso de uso disponible en el apartado 5. Ambos podían observar errores, tanto pertenecientes al comportamiento de los robots como al funcionamiento global del sistema. Si el error estaba relacionado con el primer caso, Javier se encargaba de solucionarlo. En cambio, si estaba relacionado con el segundo caso, era Francisco Javier.

De la memoria también se ha encargado de mantener la bibliografía, enumerar los trabajos citados y mencionarlos cuando era pertinente, además de mantener la correcta enumeración de ilustraciones y tablas.

Javier Romero Pérez

Ha realizado conjuntamente la instalación del sistema junto con Francisco Javier Pacheco: todo el proceso de estudio de la herramienta *Bolotweet*, obtención de recursos y configuración de los mismos.

Para la configuración de esta plataforma se estudiaron varias alternativas, como se ha mencionado en el apartado anterior. Javier investigó y desarrolló versiones alternativas que podrían haber sido utilizadas en caso de no haber funcionado la versión finalmente presentada, no sin antes haber participado en la configuración inicial probada que, finalmente, fue la versión utilizada a lo largo de todo el proyecto.

Probó a instalar el sistema en una máquina virtual con Ubuntu 14.04, comprobando la compatibilidad tanto de las versiones más antiguas de *MySQL* y PHP como de las versiones actuales. También encontró otros repositorios que ofrecían varias de las librerías requeridas, y probó los códigos proporcionados.

También hizo uso de un contenedor software *Docker* que contenía una versión ya operativa de *GNU Social*. Sin embargo, al no poder acceder al código fuente de la plataforma y, por tanto, siendo imposible su modificación para añadir los plugins necesarios, se descartó su uso.

El trabajo de Javier se centró más en la codificación de los robots. Fue el encargado de estudiar más a fondo el funcionamiento del sistema, cómo los robots se movían a lo largo y ancho del mapa, cómo se orientaban, cómo calculaban desde qué dirección eran atacados, etc.

Realizó gran cantidad de pruebas para testar su comportamiento: desde unos códigos primitivos en los que el robot daba vueltas alrededor del mapa y disparaba cuando su radar detectaba un enemigo hasta los actuales comportamientos en los que el robot reacciona a lo que sucede en su entorno: esquiva los muros que delimitan el campo de batalla, intenta alejarse de la trayectoria de las balas que le impactan, etc.

Documentó todo este proceso de análisis, ajustando los comportamientos hasta obtener unos resultados que se ajustaban a los objetivos del proyecto. Cabe recordar que uno de los objetivos propuestos era nivelar las batallas de tal manera que no estaban completamente cerradas antes de ser ejecutadas, permitiendo a los robots más débiles poder ganar alguna ronda, siendo este hecho algo esporádico.

Las pruebas que realizó eran variadas, desde enfrentamientos 1v1 entre robots de iguales y distintos niveles hasta *roborumbles* con una gran cantidad de participantes, estudiando así qué tamaño de tablero era adecuado para soportar estas batallas. Para cada prueba realizada modificaba el comportamiento de los robots si era necesario.

Para facilitar la tarea de codificación y posterior prueba del robot, adaptó el script original creado para poder seleccionar de forma más accesible qué robots participaban en una batalla y en qué condiciones. Codificó, por tanto, el sistema de menús existente en la versión final de dicho script, que permitía cambiar el tamaño del campo de batalla, el número de rondas y el número de días a tener en cuenta para el cálculo de las medias de los alumnos. Además, también se encargó de codificarlo de tal manera que podía elegirse qué grupo estaba activo, para poder seleccionar los robots pertenecientes al mismo.

Automatizó la ejecución de dicho script mediante el software *Expect*, que permite parametrizar la entrada por consola que recibe un script *Bash* según qué salida produce este. Así optimizó los procesos de prueba de comportamiento de robots.

Ha desarrollado gran parte de los apéndices –ambos manuales de uso-, y el caso de uso descrito en el apartado 3 de la memoria. Ello también le sirvió para comprobar de

manera más global el funcionamiento de los robots, viendo cómo evolucionaba el ranking a largo plazo. Detectó errores en los scripts de administración que él mismo se encargó de subsanar.

El trabajo de desarrollar y documentar un caso de uso cumple con los dos propósitos: estudiar el sistema e ilustrar su funcionamiento.

Conjuntamente desarrollaron el caso de uso disponible en el apartado 5. Ambos podían observar errores, tanto pertenecientes al comportamiento de los robots como al funcionamiento global del sistema. Si el error estaba relacionado con el primer caso, Javier se encargaba de solucionarlo. En cambio, si estaba relacionado con el segundo caso, era Francisco Javier.

6. Conclusiones

Bolocode nace con la idea de proveer a los alumnos de un incentivo para mantener una participación activa en *Bolotweet.*

Según los estudios y autores consultados, aplicar métodos y herramientas características de los videojuegos aumenta la motivación de los usuarios. El uso de un juego como *Robocode*, junto con elementos tales como tablas de clasificación, materializa la gamificación de la asignatura, que era el objetivo principal del proyecto.

Otro objetivo propuesto era premiar a los alumnos con mejores calificaciones en *Bolotweet* con mayores ventajas dentro del juego, sin perder el interés de los estudiantes que tengan un mal rendimiento. Estas ventajas debían materializarse en batallas cuyo resultado fuese positivo para el robot de mayor nivel. Se ha conseguido, como se ha podido observar en el apartado anterior, que un buen desempeño en *Bolotweet* deriva en un puesto alto en la clasificación general, y que si un alumno no obtiene buenos resultados pero comienza a mejorar su rendimiento en el sistema puede escalar posiciones y reengancharse en la lucha por alcanzar un buen puesto.

Como aspecto negativo, cabe resaltar que el sistema no ha podido ser probado con usuarios reales. Un problema en el desarrollo del sistema retrasó su puesta en funcionamiento y todas las pruebas han sido realizadas con usuarios ficticios. Por tanto, no han podido obtenerse opiniones de usuarios reales, aunque sí ha podido estudiarse el funcionamiento del sistema a largo plazo.

En general, la experimentación muestra que las batallas están equilibradas, sin ser totalmente previsibles: el robot de mayor nivel obtendrá mejores resultados, pero es posible que de vez en cuando se vea superado por un rival inferior. Esto facilita el cumplimiento del segundo objetivo mencionado.

Esta experimentación se ha llevado a cabo tanto con batallas sacadas de contexto, con el fin de estudiar qué tendencia en los resultados se produce -número de victorias del robot de mayor nivel y diferencia de puntos respecto al resto de enemigos-, como con un estudio continuado del uso de la plataforma, teniendo en cuenta la acumulación de puntos y el uso de un ranking de usuarios.

Por último, para casos futuros, dos aspectos de la actual versión del trabajo son interesantes de abordar.

- Por un lado, habría de ejecutarse la automatización de la herramienta para ser lanzada desde un servidor.
- Por otro lado, sería interesante añadir complejidad al sistema de niveles. En futuras versiones habría más parámetros a tener en cuenta a la hora de asignar habilidades a los robots. Por ejemplo, no solo calcular la media de *n* días sino la progresión de los alumnos–cuánto han mejorado sus medias, número de tareas totales que han completado-. Habría que tener en cuenta el número de tareas que

han sido lanzadas en el período de *n* días y cuántas de esas tareas ha completado el alumno. Si en el periodo de días estipulado han sido propuestas tres tareas y un alumno ha completado solo una obteniendo un 3, se valoraría de manera diferente a otro alumno que haya completado las tres tareas con una media de 2,5.

Por tanto, hay que concluir que:

- **Existe una relación proporcional entre las calificaciones en *Bolotweet* y los resultados en *Bolocode*:** Un alumno con buen desempeño en *Bolotweet* va a tener un robot más competitivo y ganará más batallas, mientras que un abandono de las tareas supone tener un robot poco competitivo.
- **Las batallas pueden deparar resultados inesperados:** Como varios autores han señalado, es necesario mantener el interés de los usuarios. Por tanto, permitir que en una batalla pueda saltar la sorpresa provoca que estos alumnos sientan interés en el sistema.
- **La carga de trabajo para el profesor no aumenta en exceso:** Originalmente, el profesor se limitaba a lanzar tareas y posteriormente corregirlas. Ahora, ha de ejecutar las batallas. Sin embargo, se desentiende de la tarea de actualizar las puntuaciones en *Bolotweet* y las medias de los usuarios, pues todo es realizado de manera automática por el sistema.
- **El profesor tiene libertad a la hora de ejecutar las batallas:** No hay estipulada ninguna manera de uso del sistema. El profesor puede elegir qué tipo de batallas ejecutar, quiénes participan, las características del entorno, etc. Puede utilizar el criterio que desee a la hora de elegir participantes.

7. Conclusions

Bolocode was born with the idea of giving students an incentive to keep an active participation in *Bolotweet*.

Following the researchs and authors consulted, aplying methods and tools from videogames increases users' motivation. Using a game like *Robocode*, together with elements like rankings, builds the gamification of a subject, which was the main purpose of the project.

Another objective was to reward students with higher grades in *Bolotweet* with better advantages in the game, without losing interest from those students whose grades were lower. These advantages should materialize in battles whose results were positive for the stronger robot. It has been reached that a good performance in *Bolotweet* means a top position in the ranking, and also that students whose grades are low can reach top positions if they improve their performance.

As a negative aspect, it has not been possible to test the system with real users. There was a problem during its development and all the tests have been performed with fictional users. Therefore, there is no feedback from real users, but it has been possible to study the system's long-term performance.

In general, experimentation shows that battles are balanced, and there is a chance that an unexpected result comes along: stronger robot will get better results, but it is possible that this robot can be defeated by a weaker one. This supplies the second objective introduced.

This experimentation has been performed with battles taken out of context –just to analyze what kind of results these battles report-, and also it has been tested the system's long-term performance, considering the points earned by robots and the general ranking.

Finally, for future cases, there are two aspects from the current version of the work that are interesting to approach:

- On the one hand, the tool should be automatized and launched from a server.
- On the other hand, it might be interesting to add more complexity to the level system. In future versions, there might be more parameters to be taken into account when choosing robots' skills. For example, not only calculating the average points earned in Bolotweet in the last n days, but the students' progression –how much have their grades improved, total number of tasks completed-. It should be taken into account the total number of tasks proposed in the last n days and how many of those tasks have been completed by the student. If in the last n days three task kave been proposed and a student has completed just one of them, earning a 3, it should be rewarded differently than other student who has completed the three task, earning a 2,5.

Therefore, it can be concluded that:

- **There is a proportional relationship between grades in *Bolotweet* and performance in *Bolocode*:** The students' robots will be more capable as their performance improves in *Bolotweet*, whilst abandoning the tool means a less competitive robot.
- **Battles might offer unexpected results:** As several authors have pointed, it is necessary to keep students' interest. In order to achieve this objective, battles' results are not always predictable.
- **The effort to run the battles is minimal:** Originally, teacher's work was to propose tasks and evaluate them. Now, the teacher have to run the battles. However, the maintenance of the ranking and the update of results is carried out automatically by the system.
- **The teacher is free to set the battles' parameters:** There is no stipulated way to run a battle. The teacher can choose what kind of battle to run, competitors, the environment's properties, etc.

8. Apéndices

Trabajo de Fin de Grado en Ingeniería Informática

Curso 2017-2018

BOLOCODE

A. Manual de uso – Profesor

Autores: Francisco Javier Pacheco Herranz, Javier Romero Pérez

Índice

1. Introducción. Qué es *Bolocode*

Bolocode es un plugin para *Bolotweet* que gamifica el uso de esta herramienta, proponiendo batallas entre robots controlados por una inteligencia artificial que será construida en base al desempeño de los estudiantes dentro de la plataforma *Bolotweet*. Cuanto mejor sea la calificación media obtenida en los mensajes enviados, el alumno tendrá un robot más competitivo.

Los resultados acumulados de las batallas serán mostrados en la página del plugin, manteniendo un ranking constantemente actualizado.

2. Solicitud de alta de grupo

La solicitud de alta de grupo se realiza de igual manera que para *Bolotweet 2.0* [17].

Sin embargo, adicionalmente, el profesor ha de indicar si quiere activar el plugin. En caso afirmativo, ha de proveer al administrador un archivo .csv en el que se indica, para cada alumno, qué tres colores aplicar a su robot.

Dicho fichero ha de tener el siguiente formato:

	A	B	C	D
1	beatriz	black	yellow	yellow
2	carlos	blue	black	black
3	silvia	white	red	white
4	ana	red	red	red
5	manuel	orange	orange	black
6	david	white	white	blue

Ilustración 50 – Ejemplo de archivo .csv con los colores de los robots

3. Ejecución del programa e inicio de sesión

El código que da soporte a toda la funcionalidad está codificado en *Bash*. Para ejecutarlo, basta con acceder al directorio ./robocode/RoboPlugin/ y, ahí, ejecutar el script *bolocode.sh*.

```
fran@fran:~/robocode/RoboPlugin$ ./bolocode.sh
¡Hola!
¡Bienvenido al generador de batallas Robocode!
Leyendo la configuración...
¿A qué grupo perteneces?
Escribe cancel para cancelar.
eda
Las medias de los usuarios del grupo 'eda' han sido actualizadas.
ACTUALIZANDO ROBOTS...
carlos 0
silvia 0
ana 0
manuel 0
david 0
beatriz 0
El nivel del robot carlos es 1
manifiesto agregado
agregando: robots/carlos.java(entrada = 4700) (salida = 1323)(desinflado 71%)
El nivel del robot silvia es 1
manifiesto agregado
agregando: robots/silvia.java(entrada = 4699) (salida = 1325)(desinflado 71%)
El nivel del robot ana es 1
manifiesto agregado
agregando: robots/ana.java(entrada = 4692) (salida = 1317)(desinflado 71%)
El nivel del robot manuel es 1
manifiesto agregado
agregando: robots/manuel.java(entrada = 4703) (salida = 1325)(desinflado 71%)
El nivel del robot david es 1
manifiesto agregado
agregando: robots/david.java(entrada = 4699) (salida = 1322)(desinflado 71%)
El nivel del robot beatriz es 1
manifiesto agregado
agregando: robots/beatriz.java(entrada = 4704) (salida = 1326)(desinflado 71%)
--------------------------------------------------------------------------------
Elige una de las siguientes opciones:
1.-Modificar opciones de notas/grupo.          2.-Modificar/crear robots.
3.-Modificar opciones de batalla.              4.-Generar batalla.
0.-Salir.
--------------------------------------------------------------------------------
```

Ilustración 51 – Ejemplo de ejecución del script bolocode.sh

Lo primero que pide el script es indicar el grupo de clase sobre el que trabajar. Para esta versión de la plataforma, es suficiente con introducir el ID o el *nickname*.

Una vez el grupo es seleccionado, se cargan los robots y, en base a sus medias, asigna los niveles correspondientes.

Aparece entonces el menú principal, que despliega una serie de opciones que van a ser detallas en los posteriores apartados.

4. Gestión de notas y grupos

Como se aprecia en la ilustración 52, desde el menú principal, seleccionando la opción 1, el profesor puede gestionar sus grupos.

Se le ofrecen dos acciones: cambiar de grupo de trabajo y modificar el número de días a tener en cuenta para calcular la media de las calificaciones.

```
--------------------------------------------------------------------------------
Opción:
1
--------------------------------------------------------------------------------
Elige una de las siguientes acciones:
1.-Modificar grupo de trabajo.          2.-Modificar periodo de medias.
0.-Volver atrás.
--------------------------------------------------------------------------------
```

Ilustración 52 – Selección de la opción 1 en el menú principal de bolocode.sh

Cambiar grupo

En caso de seleccionar la primera opción, el profesor puede cambiar de grupo sobre el que trabajar. Para esta versión del proyecto, es suficiente con indicar un ID o un *nickname.*

```
--------------------------------------------------------------------------------
Opción:
1
Actualmente el grupo activo es el grupo 7
¿A qué grupo perteneces?
Escribe cancel para cancelar.
ssii
Las medias de los usuarios del grupo 'ssii' han sido actualizadas.
ACTUALIZANDO ROBOTS...
El nivel del robot susana es 1
manifiesto agregado
agregando: robots/susana.java(entrada = 4695) (salida = 1320)(desinflado 71%)
El nivel del robot maria es 2
manifiesto agregado
agregando: robots/maria.java(entrada = 4309) (salida = 1242)(desinflado 71%)
El nivel del robot paco es 1
manifiesto agregado
agregando: robots/paco.java(entrada = 4699) (salida = 1322)(desinflado 71%)
El nivel del robot lucia es 2
manifiesto agregado
agregando: robots/lucia.java(entrada = 4309) (salida = 1241)(desinflado 71%)
El nivel del robot sergio es 1
manifiesto agregado
agregando: robots/sergio.java(entrada = 4701) (salida = 1324)(desinflado 71%)
El nivel del robot andrea es 3
manifiesto agregado
agregando: robots/andrea.java(entrada = 2836) (salida = 1246)(desinflado 56%)
susana 1.0000
maria 2.0000
paco 1.0000
lucia 2.0000
sergio 1.0000
andrea 3.0000
--------------------------------------------------------------------------------
```

Ilustración 53 – Cambio de grupo de trabajo

Modificar número de días

En caso de seleccionar la segunda opción, el profesor puede modificar el número de días a tener en cuenta para calcular la media de las calificaciones.

En caso de introducir un número negativo, el sistema rechazará dicha entrada. Si es positivo y, además, supera el número de días desde el primer mensaje enviado, se calculará la media en base a todos los mensajes enviados por el usuario.

```
--------------------------------------------------------------------------------
Opción:
2
Actualmente el periodo de notas es de 5 días
¿Cuántos días de antigüedad quieres que tenga la media de notas?
Escribe cancel para cancelar.
7
Actualizando los niveles de los robots...
Las medias de los usuarios del grupo 'ssii' han sido actualizadas.
El nivel del robot susana es 1
manifiesto agregado
agregando: robots/susana.java(entrada = 4695) (salida = 1320)(desinflado 71%)
El nivel del robot maria es 1
manifiesto agregado
agregando: robots/maria.java(entrada = 4700) (salida = 1322)(desinflado 71%)
El nivel del robot paco es 1
manifiesto agregado
agregando: robots/paco.java(entrada = 4699) (salida = 1322)(desinflado 71%)
El nivel del robot lucia es 1
manifiesto agregado
agregando: robots/lucia.java(entrada = 4700) (salida = 1323)(desinflado 71%)
El nivel del robot sergio es 1
manifiesto agregado
agregando: robots/sergio.java(entrada = 4701) (salida = 1324)(desinflado 71%)
El nivel del robot andrea es 1
manifiesto agregado
agregando: robots/andrea.java(entrada = 4701) (salida = 1322)(desinflado 71%)
¿Quieres guardar esta nueva configuración? (S/N)
n
susana 0
maria 0
paco 0
lucia 0
sergio 0
andrea 0
--------------------------------------------------------------------------------
```

Ilustración 54 – Modificación del número de días para el cálculo de la media

El sistema permite guardar permanentemente la configuración establecida o, por el contrario, aplicarla únicamente a la actual ejecución del sistema.

5. Gestión de los robots

Como se aprecia en la ilustración 55, desde el menú principal, seleccionando la opción 2, el profesor puede gestionar los robots de un grupo.

Se le ofrecen tres acciones: listar los robots pertenecientes al grupo, crear un nuevo robot o modificar los colores de estos.

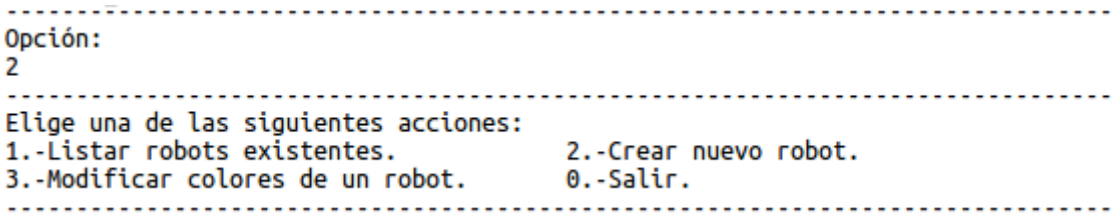


Ilustración 55 – Selección de la opción 2 en el menú principal de bolocode.sh

Listar robots pertenecientes al gupo

En caso de seleccionar la primera opción, el programa listará todos los robots que están listos para participar en una batalla.

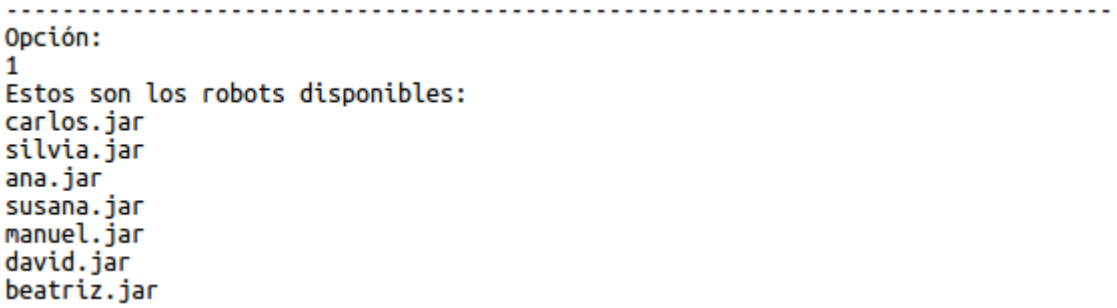


Ilustración 56 – Listado de los robots pertenecientes al grupo

Crear nuevo robot

En caso de seleccionar la segunda opción, el programa permitirá al profesor introducir un nuevo robot.

Esta circunstancia no se espera que sea frecuente pero, en caso de darse, el profesor puede dar de alta a un alumno en el sistema. Sin embargo, solo podrá hacerlo si el alumno está dado de alta en *Bolotweet*, y para ello ha de delegar dicha tarea en el administrador.

```
--------------------------------------------------------------------------------
Opción:
2
Introduce el nombre que le quieres dar al robot:
Escribe cancel para cancelar.
oscar
¡Perfecto! El robot ha sido incluido para pelear.
¿Quieres elegir los colores de tu robot o usar los de por defecto? (S/N) (S: Elegir colores)
Escribe cancel para cancelar.
s
Los colores disponibles son:
black blue green orange pink red white yellow
Elige: colorBody colorGun colorRadar
yellow orange orange
¡Perfecto! Los colores del robot han sido modificados.
¡Vaya! Parece que el usuario oscar no ha participado en Bolotweet. Se le asignará un robot de nivel 1
manifiesto agregado
agregando: robots/oscar.java(entrada = 4703) (salida = 1326)(desinflado 71%)
```

Ilustración 57 – Creación de un nuevo robot

Cuando el profesor introduce el nombre del usuario, se le pregunta por los colores. Puede elegir aplicar los colores por defecto o introducirlos manualmente de entre una serie de colores disponibles. En este segundo caso, ha de introducir los tres colores que se aplicarán, correspondientemente, al cuerpo, cañón y radar.

En último lugar, consulta la media del usuario y le asigna un nivel a la inteligencia del robot.

Modificar colores de un robot

En caso de seleccionar la tercera opción, el programa permitirá al profesor modificar los colores de un robot de su grupo.

```
--------------------------------------------------------------------------------
Opción:
3
Introduce el nombre del robot que quieres modificar
Escribe cancel para cancelar.
carlos
¿Quieres elegir los colores de tu robot o usar los de por defecto? (S/N) (S: Elegir colores)
Escribe cancel para cancelar.
n
Aplicando colores por defecto...
manifiesto agregado
agregando: robots/carlos.java(entrada = 4701) (salida = 1323)(desinflado 71%)
```

Ilustración 58 – Modificación de los colores de un robot

El usuario que introduce el profesor ha de existir y pertenecer al grupo sobre el que se está trabajando. Una vez se ha introducido un usuario correcto, el sistema es igual que en el proceso de creación de un robot: puede asignar los colores por defecto o introducirlos manualmente.

6. Modificación de los parámetros de una batalla

Como se aprecia en la ilustración 59, desde el menú principal, seleccionando la opción 3, el profesor puede modificar los parámetros de una batalla.

Se le ofrece entonces dos parámetros a modificar: el tamaño del campo de batalla y el número de rondas a ejecutar.

```
--------------------------------------------------------------------------------
Opción:
3
--------------------------------------------------------------------------------
Elige una de las siguientes acciones:
1.-Modificar tamaño de tablero.         2.-Modificar número de rondas.
0.-Salir.
--------------------------------------------------------------------------------
```

Ilustración 59 – Selección de la opción 3 en el menú principal de bolocode.sh

Modificación del tamaño del campo de batalla

En caso de seleccionar la primera opción, el profesor indica el nuevo ancho y alto del campo de batalla. Posteriormente, se le pregunta si desea guardar la configuración o no, tanto para el ancho como para el alto.

```
--------------------------------------------------------------------------------
Opción:
1
Actualmente el tamaño del tablero es de 900 x 600
Indica el tamaño del tablero
Ancho:
Escribe cancel para cancelar.
800
Alto:
Escribe cancel para cancelar.
600
¿Quieres guardar esta nueva configuración? (S/N)
s
¿Quieres guardar esta nueva configuración? (S/N)
s
```

Ilustración 60 – Introducción del nuevo tamaño del campo de batalla

Modificación del número de rondas

En caso de seleccionar la segunda opción, el profesor indica el nuevo número de rondas a batallar. De nuevo, se le pregunta si desea guardar la configuración o no.

```
--------------------------------------------------------------------------------
Opción:
2
Actualmente el número de rondas es de 3
¿Cuántas rondas quieres batallar?
Escribe cancel para cancelar.
5
¿Quieres guardar esta nueva configuración? (S/N)
n
```

Ilustración 61 – Introducción del nuevo número de rondas

7. Ejecución de una batalla

Como se aprecia en la ilustración 62, desde el menú principal, seleccionando la opción 4, el profesor puede ejecutar una batalla.

El sistema provee dos tipos de batalla: un enfrentamiento en el que el profesor puede introducir manualmente los rivales, o un *roborumble*, una batalla en la que participan todos los usuarios del grupo.

```
--------------------------------------------------------------------------------
Opción:
4
¿Quieres escoger los robots a batallar o hacer un RoboRumble?
(E/R) E: escoger, R: RoboRumble
```

Ilustración 62 - Selección de la opción 4 en el menú principal de bolocode.sh

Escoger robots

En caso de escoger los robots, el profesor tendrá que introducir los nombres de los usuarios que van a participar en la batalla.

En una batalla han de participar mínimo dos alumnos, y solo puede haber un robot por estudiante.

```
¿Quieres escoger los robots a batallar o hacer un RoboRumble?
(E/R) E: escoger, R: RoboRumble
e
Estos son los robots disponibles para tu grupo
carlos.jar
silvia.jar
ana.jar
susana.jar
manuel.jar
david.jar
beatriz.jar
oscar.jar
Escoge un robot
carlos
Se ha escogido un robot correctamente.
¿Quieres escoger otro robot? (S/N)
s
Estos son los robots disponibles para tu grupo
carlos.jar
silvia.jar
ana.jar
susana.jar
manuel.jar
david.jar
beatriz.jar
oscar.jar
Escoge un robot
beatriz
Se ha escogido un robot correctamente.
¿Quieres escoger otro robot? (S/N)
n
GENERANDO BATALLA
Espere un momento...
```

Ilustración 63 – Selección manual de los robots participantes

Una vez el profesor indica que no quiere más usuarios, y habiendo sido escogidos al mínimo dos, comienza la ejecución de la batalla.

Roborumble

En caso de escoger este tipo de batalla, esta se ejecuta automáticamente seleccionando todos los robots pertenecientes al grupo.

```
¿Quieres escoger los robots a batallar o hacer un RoboRumble?
(E/R) E: escoger, R: RoboRumble
r
GENERANDO BATALLA
Espere un momento...
```

Ilustración 64 – Ejecución de un roborumble

Trabajo de Fin de Grado en Ingeniería Informática

Curso 2017-2018

BOLOCODE

B. Manual de uso – Administrador

Autores: Francisco Javier Pacheco Herranz, Javier Romero Pérez

Índice

1. Introducción. Qué es *Bolocode*

Bolocode es un plugin para *Bolotweet* que gamifica el uso de esta herramienta, proponiendo batallas entre robots controlados por una inteligencia artificial que será construida en base al desempeño de los estudiantes dentro de la plataforma *Bolotweet*. Cuanto mejor sea la calificación media obtenida en los mensajes enviados, el alumno tendrá un robot más competitivo.

Los resultados acumulados de las batallas serán mostrados en la página del plugin, manteniendo un ranking constantemente actualizado.

2. Creación de grupos y activación de la herramienta

El proceso de creación de un grupo y solicitud de ser profesor del mismo ya es explicada en el manual de uso de *Bolotweet* [17].

Adicionalmente, para este nuevo plugin, el profesor ha de indicar que quiere la activación del mismo.

Activar plugin

Se hará uso de un script PHP que permite tanto activar como desactivar el plugin a un grupo ya creado.

```
setrobocodetogroup.php [options]

-G –group-id ID del grupo

-g –group Nickname del grupo

-a --activated Activación (1) o desactivación (0) del grupo
```

Como se ha mencionado, necesariamente el grupo ha de existir, de lo contrario la ejecución finalizaría con un mensaje de error. No es necesario proporcionar tanto el ID como el *nickname*, con uno de los dos identificadores es suficiente.

La ejecución del mismo sería:

```
fran@fran:/var/www/bolotweet/scripts$ php setrobocodetogroup.php -geda -a1
Activando el plugin Robocode al grupo 'eda'
```

Ilustración 65 – Ejemplo de ejecución del script setrobocodegroup

3. Creación de los robots de los alumnos

Una vez el grupo tiene activado el plugin, sus robots han de ser creados dentro del sistema. Así, los alumnos podrán tener su avatar con los colores personalizados que ellos elijan.

Creación de los robots

El administrador recibirá un archivo .csv en el que aparecen los nombres de los alumnos junto los tres colores elegidos para su robot. En el orden en el que los colores aparecen el archivo serán aplicados al cuerpo, cañón y radar.

Para llevar a cabo esta tarea, se le proporciona un script *Bash* que lee del fichero .csv los datos de cada alumno y crea los correspondientes robots.

```
crearobots.sh [options]

nombre Nombre del fichero .csv del que leer los datos

grupo ID del grupo al que registrar los alumnos
```

El script no funcionará en caso de que el grupo introducido no exista o no tenga el plugin activado. En el caso de que un alumno que aparezca en el fichero no esté registrado en el grupo, se obviará y se proseguirá con el siguiente alumno.

La ejecución del mismo sería:

```
fran@fran:~/robocode/RoboPlugin$ ./crearobots.sh robotseda.csv 7
Bienvenido al script generador de robots
Leyendo robots del archivo...
manifiesto agregado
agregando: robots/beatriz.java(entrada = 4704) (salida = 1326)(desinflado 71%)
El robot 'beatriz' ha sido correctamente registrado en el grupo '7'
manifiesto agregado
agregando: robots/carlos.java(entrada = 4700) (salida = 1323)(desinflado 71%)
El robot 'carlos' ha sido correctamente registrado en el grupo '7'
manifiesto agregado
agregando: robots/silvia.java(entrada = 4699) (salida = 1325)(desinflado 71%)
El robot 'silvia' ha sido correctamente registrado en el grupo '7'
manifiesto agregado
agregando: robots/ana.java(entrada = 4696) (salida = 1323)(desinflado 71%)
El robot 'ana' ha sido correctamente registrado en el grupo '7'
manifiesto agregado
agregando: robots/manuel.java(entrada = 4703) (salida = 1325)(desinflado 71%)
El robot 'manuel' ha sido correctamente registrado en el grupo '7'
manifiesto agregado
agregando: robots/david.java(entrada = 4699) (salida = 1322)(desinflado 71%)
El robot 'david' ha sido correctamente registrado en el grupo '7'
```

Ilustración 66 – Ejemplo de ejecución del script crearobots

4. Administrar grupos

Un administrador puede controlar el fichero que contiene la media de las calificaciones obtenidas por los usuarios de un grupo y las puntuaciones obtenidas en las batallas.

Actualizar fichero de medias

Al administrador se le facilita un script *Bash* que permite actualizar el fichero de medias de un grupo, pudiendo proporcionar tanto el ID del grupo como su *nickname*.

```
updategradesfileandshow.sh [options]

grupo ID o nickname del grupo a actualizar

días Número de días a tener en cuenta en el cálculo de la media
```

El script no funcionará en caso de introducir un grupo que no tenga el plugin *Robocode* activado o que, directamente, el grupo no exista.

La ejecución del mismo sería:

```
fran@fran:/var/www/bolotweet/scripts$ ./updategradesfileandshow.sh eda 7
Las medias de los usuarios del grupo 'eda' han sido actualizadas.
```

Ilustración 67 – Ejemplo de ejecución del script updategradesfileandshow

Actualizar resultados de las batallas

Todas las batallas son almacenadas en el directorio /tmp/batallas/. Se le proporciona al administrador un script *Bash* que recorre dicho directorio y, para cada fichero que encuentra, lo lee y actualiza las puntuaciones mostradas en *Bolotweet*.

```
updateallrobocodescores.sh
```

Si el directorio está vacío o no existe, no sucede nada.

La ejecución del mismo sería:

```
fran@fran:/var/www/bolotweet/scripts$ ./updateallrobocodescores.sh
Todos los resultados han sido almacenados.
```

Ilustración 68 – Ejemplo de ejecución del script updateallrobocodescores

Sin embargo, se ha desarrollado también un *Crontab* que ejecuta dicho script periódicamente, para así automatizar la tarea de actualización de resultados.

9. Bibliografía

[1] Buckley, P., & Doyle, E. (2016). Gamification and student motivation. *Interactive Learning Environments*, *24*(6), 1162-1175.

[2] Deterding, S. (2012). Gamification: designing for motivation. *Interactions magazine*, *19*(4), 14-17.

[3] Díaz Cruzado, J., & Troyano Rodríguez, Y. (2013). El potencial de la gamificación aplicado al ámbito educativo. *III Jornadas de Innovación Docente. Innovación Educativa: respuesta en tiempos de incertidumbre (2013)*,.

[4] Gee, J. P. (2008). Learning and games. *The ecology of games: Connecting youth, games, and learning*, *3*, 21-40.

[5] Glover, I. (2013, June). Play as you learn: gamification as a technique for motivating learners. In *EdMedia: World Conference on Educational Media and Technology* (pp. 1999-2008). Association for the Advancement of Computing in Education (AACE).

[6] Groh, F. (2012). Gamification: State of the art definition and utilization. *Institute of Media Informatics Ulm University*, *39*, 31.

[7] Kapp, K. M. (2012). Games, gamification, and the quest for learner engagement. *T+ D*, *66*(6), 64-68.

[8] Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons.

[9] Kohn, A. (1999). *Punished by rewards: The trouble with gold stars, incentive plans, A's, praise, and other bribes*. Houghton Mifflin Harcourt.

[10] Lepper, M. R., Corpus, J. H., & Iyengar, S. S. (2005). Intrinsic and extrinsic motivational orientations in the classroom: Age differences and academic correlates. *Journal of educational psychology*, *97*(2), 184.

[11] Ruzic, I. M., & Dumancic, M. (2015). GAMIFICATION IN EDUCATION/IGRIFIKACIJA U ODGOJU I OBRAZOVANJU. *Informatologia*, *48*(3/4), 198.

[12] Sandusky, S. (2015). Gamification in education.

[13] Sheldon, L. (2011). *The multiplayer classroom: Designing coursework as a game*. Cengage Learning.

[14] Stott, A., & Neustaedter, C. (2013). Analysis of gamification in education. *Surrey, BC, Canada*, *8*, 36.

[15] Thom, J., Millen, D., & DiMicco, J. (2012, February). Removing gamification from an enterprise SNS. In *Proceedings of the acm 2012 conference on computer supported cooperative work* (pp. 1067-1070). ACM.

[16] "Elizabeth Lawley: "Just Press Play" – Adding a Game Layer to the Undergraduate Experience", Vimeo video, 14:19, posted by "Connected Learning Alliance", October 29, 2012.

[17] Ortego, Álvaro. (2014). *Bolotweet 2.0.* Trabajo de Fin de Grado en Ingeniería del Software. Facultad de Informática, Universidad Complutense de Madrid.

[18] ALATech Source Journals. Chapter 4. (2015). *Gamification in Education and Libraries.* https://journals.ala.org/index.php/ltr/article/view/5631/6951?utm_content=buffer19760&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer

[19] Purdue University News. (2012) *Digital Badges show students' skills along with degree*. http://www.purdue.edu/newsroom/releases/2012/Q3/digital-badges-show-students-skills-along-with-degree.html

[20] "Piano Stairs", YouTube video, 1:47, posted by "Rolighetsteorin", October 7, 2009, embedded on The Fun Theory website, www.thefuntheory.com

[21] "The Speed Camera Lottery", YouTube video, 2:08, posted by "Rolighetsteorin", November 12, 2010, embedded on The Fun Theory website, www.thefuntheory.com

[22] Página de *Bolotweet*, http://grasia.fdi.ucm.es/bolotweet/

[23] Página de CLOC, http://cloc.sourceforge.net/

[24] Página de *Fantasy Geopolitics,* https://www.fanschool.org/

[25] Página de *Kahoot!*, https://kahoot.com/

[26] Página de *Mozilla Open Badges*, https://openbadges.org/

[27] Página de *Passport*, https://www.openpassport.org/Account/Login?ReturnUrl=%2f#WhatIsPassport

[28] Página de *Robocode*, http://robowiki.net/wiki/Robocode

[29] Pacheco Herranz, F. J., & Romero Pérez, J. (2018). Código fuente de *Bolocode*. https://github.com/franpach/bolocode