

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de
Telecomunicación

*Aplicación web para la educación mediante
gamificación: Integración modular y mejoras
funcionales de los módulos de Administración y
Comunicaciones*

Autor: José Manuel Martínez Delgado
Tutor: Fco. Javier Muñoz Calle

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024



Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación web para la educación mediante gamificación: Integración modular y mejoras funcionales de los módulos de Administración y Comunicaciones

Autor:
José Manuel Martínez Delgado

Tutor:
Francisco Javier Muñoz Calle
Profesor Colaborador

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Trabajo Fin de Grado: Aplicación web para la educación mediante gamificación: Integración modular y mejoras funcionales de los módulos de Administración y Comunicaciones

Autor: José Manuel Martínez Delgado

Tutor: Francisco Javier Muñoz Calle

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia
A mis maestros

Agradecimientos

Tras varios años estudiando en la ETSI, cierro una etapa de mi vida llena de buenos recuerdos y experiencias que han moldeado al niño introvertido que entró por las puertas hace 6 años hasta convertirlo en un adulto con curiosidad por lo desconocido, sin miedo a afrontar retos desafiantes y con seguridad en sí mismo respaldado por los aprendizajes obtenidos durante estos años.

No fue fácil, y quien diga lo contrario se estaría engañando a sí mismo. La pasión por lo que haces es un pilar fundamental a la hora de conseguir el éxito, pero nada de esto hubiera sido posible de no ser por mi familia, a quienes dedico estas palabras.

A mi padre José Manuel, que siempre se encargó de recordarme que nada es imposible si se trabaja lo suficiente y que la perseverancia siempre supera al talento.

A mi madre Amparo, quién siempre ha creído en mí y me ha apoyado en los momentos difíciles, haciendo lo imposible para ayudarme.

A mi hermana Marina, por su apoyo incondicional y por servir de referente como hermana mayor siempre.
A mis abuelos, que jamás imaginaron con mi edad que alguno de sus nietos llegaría a estudiar en la Universidad, ni mucho menos llegar a ser ingeniero.

A mi tutor Javier por su paciencia y comprensión. A pesar de haber visto limitada mi disponibilidad para este proyecto, siempre me ha dado la mayor flexibilidad posible y me ha ayudado.

A Jesús, autor del Proyecto AJDA y colaborador del Proyecto Gamifica por su ayuda en el trascurso del proyecto y sus comentarios motivadores.

*José Manuel Martínez Delgado
Sevilla, 2024*

La gamificación es una metodología de aprendizaje que se integra muy bien en las aulas de las nuevas generaciones que van de la mano con la tecnología desde muy pequeños. El Proyecto de Aplicación de Juegos Didácticos en el Aula (AJDA) [26] fue creado precisamente como una herramienta de gamificación que pueda ayudar a los profesores en el presente y el futuro bajo el marco de la RED Descartes [27]. Con el fin de mejorar la comunicación y las funcionalidades de AJDA, desde el Departamento de Telemática nació el Proyecto Gamifica, el cual lleva varios años en desarrollo y es el protagonista de esta memoria.

Este documento tiene como objetivo presentar los resultados del trabajo realizado como colaborador del proyecto, cuyo objetivo final es hacer progresar el Proyecto Gamifica mediante la integración de los diferentes módulos existentes en un solo proyecto, la implementación de mejoras en algunos módulos, la creación de un entorno de desarrollo unificado y el despliegue de un servidor en la nube que permitirá garantizar la alta disponibilidad de nuestra aplicación.

Como consecuencia de estos años de trabajo, he conseguido agilizar los procesos de desarrollo, testeo y despliegue del código lo cual facilitará el progreso del proyecto en el futuro. Además, los módulos de administración, multijugador y ficheros quedan totalmente integrados entre sí permitiendo lanzar partidas desde cero totalmente configuradas gracias a la comunicación entre dichos módulos y dejando de funcionar como microservicios independientes.

Gamification is a learning methodology that integrates very well in the classrooms of the new generations that go hand in hand with technology from a very young age. The Application of Didactic Games in the Classroom Project (AJDA) [26] was created precisely as a gamification tool that can help teachers in the present and the future under the framework of the Descartes Network [27]. To improve the communication and functionalities of AJDA, from the Department of Telematics was born the Gamifica Project, which has been under development for several years and is the protagonist of this report.

This document aims to present the results of the work done as a collaborator of the project, whose goal is to advance the Gamifica Project by integrating the different existing modules into a single project, implementing improvements in some modules, creating a unified development environment and deploying a cloud server that will ensure high availability of our application.

As a result of these years of work, I have managed to streamline the processes of development, testing and deployment of the code which will facilitate the progress of the project in the future. In addition, the modules of administration, multiplayer and files are fully integrated with each other allowing to launch games from scratch fully configured thanks to the communication between these modules and leaving functions of the game.

Agradecimientos	7
Resumen	9
Abstract	11
Índice	12
Índice de tablas	14
Índice de figuras	16
1 Introducción	19
1.1 Motivación	19
1.2 Descripción del problema	19
1.3 Solución propuesta	20
1.4 Estructura de la memoria	20
2 Tecnologías utilizadas	21
2.1 Java EE	21
2.2 Apache Maven	21
2.3 Java Beans	21
2.4 Java Server Faces (JSF)	22
2.5 PrimeFaces	22
2.6 CSS	22
2.7 WebSockets: Java + JavaScript	22
2.8 Java Persistence API (JPA)	23
2.9 Logs: Simple Logging Facade for Java (SLF4J)	23
2.10 Gestión de idiomas: ResourceBundles	23
3 El entorno de desarrollo	24
3.1 Vagrant	24
3.1.1 Creación del entorno	24
3.1.2 Vagrantfile	24
3.1.3 Personalización de la imagen base	25
3.1.4 Empaquetado y subida	25
3.2 Repositorio GIT	26
3.2.1 El nuevo repositorio	26
3.2.2 Código Java	26
3.2.3 Código JSF/Primefaces	27
3.2.4 Código JSP/HTML	27
3.2.5 Código XML	27
3.2.6 Código CSS y JavaScript	27
3.2.7 Juegos	28
3.2.8 Otros recursos	31
3.2.9 La base de datos	31
4 Mejoras en los módulos	34
4.1.1 Tecnologías y buenas prácticas de desarrollo	34
4.1.2 Mejoras en el módulo de administración	36
4.1.3 Mejoras en el módulo de comunicaciones	40
5 Entorno en la nube	43
5.1 El servidor	43
5.2 El proveedor	43

5.3 Preparación del entorno cloud	44
5.4 Automatización de subidas a la nube	44
5.5 Despliegue de versiones en la nube	45
5.6 Portal de administración Wildfly	46
6 Plan de pruebas	47
6.1 Pruebas unitarias	47
6.2 Pruebas de integración	51
7 Caso de uso real	52
7.1 Clase de biología	52
8 Documentación	58
8.1 Gestión del Proyecto Gamifica	58
8.2 Repositorio	58
9 Conclusiones y mejoras	59
9.1 Conclusiones	59
9.2 Posibles mejoras	59
9.2.1 Migración del repositorio a GitLab/GitHub	59
9.2.2 Empaquetado del código	60
9.2.3 Módulos de administración y comunicaciones	60
9.2.4 Uso de inteligencia artificial	61
10. Referencias	62

Índice de tablas

Tabla 1: PU-1	47
Tabla 2: PU-2	47
Tabla 3: PU-3	47
Tabla 4: PU-4	48
Tabla 5: PU-5	48
Tabla 6: PU-6	48
Tabla 7: PU-7	48
Tabla 8: PU-8	49
Tabla 9: PU-9	49
Tabla 10: PU-10	49
Tabla 11: PU-11	49
Tabla 12: PU-12	50
Tabla 13: PU-13	50
Tabla 14: PU-14	50
Tabla 15: PU-15	50
Tabla 16: PU-16	50
Tabla 17: PU-17	51
Tabla 18: PU-18	51
Tabla 19: PI-1	51

Índice de figuras

Ilustración 1: Repositorio Vagrant del Proyecto Gamifica	25
Ilustración 2: Crear imagen Vagrant	25
Ilustración 3: Añadir proveedor para la imagen Vagrant	26
Ilustración 4: Código Java en el repositorio	26
Ilustración 5: Juego test	28
Ilustración 6: Juego test con cifras	29
Ilustración 7: Juego artificieros	29
Ilustración 8: Juego bombilla	30
Ilustración 9: Juego batalla naval	30
Ilustración 10: Diagrama entidad relación 1	31
Ilustración 11: Diagrama entidad relación 2	31
Ilustración 12: Diagrama entidad relación 3	32
Ilustración 13: Constantes en base de datos	35
Ilustración 14: Gestión de idiomas con Bundles	35
Ilustración 15: Nomenclatura en el repositorio	35
Ilustración 16: Mejora 1, botones de confirmación	36
Ilustración 17: Mejora 2, botón de borrar todos los recursos	36
Ilustración 18: Mejora 2, confirmación	36
Ilustración 19: Mejora 3, configuración final de la partida	36
Ilustración 20: Mejora 4, panel de información de usuario	37
Ilustración 21: Mejora 5, integración con otros módulos	37
Ilustración 22: Mejora 6, interfaz de parámetros	38
Ilustración 23: Mejora 7, interfaz de partidas jugadas	38
Ilustración 24: Mejora 8, subida de ficheros de preguntas	39
Ilustración 25: Super usuario modifica la contraseña de otro usuario	39
Ilustración 26: Mejora 9, códigos de acceso	40
Ilustración 27: Mejora 9, control de accesos	40
Ilustración 28: Mejora 9, control de accesos 2	41
Ilustración 29: Mejora 11, panel de control	41
Ilustración 30: Mejora 9, control de accesos 3	41
Ilustración 31: Mejora 10, estado de conexión activa	42
Ilustración 32: Mejora 10, estado de conexión terminada	42
Ilustración 33: Mejora 12, interfaz del profesor	42
Ilustración 34: Firewall instancia cloud	43
Ilustración 35: Acceso a la aplicación, rama José Manuel	45
Ilustración 36: Acceso a la aplicación, rama master	45
Ilustración 37: Acceso a la aplicación, rama Álvaro	45
Ilustración 38: Administración Wildfly, creación de usuario admin	46
Ilustración 39: Caso de uso, configuración partida	52
Ilustración 40: Caso de uso, selección de juego	53
Ilustración 41: Caso de uso, configuración parámetros	53
Ilustración 42: Caso de uso, configuración equipos y grupos	53
Ilustración 43: Caso de uso, registro de jugadores	54
Ilustración 44: Caso de uso, trascurso de la partida	54
Ilustración 45: Caso de uso, tiempo detenido	55
Ilustración 46: Caso de uso, jugador respondiendo	55
Ilustración 47: Caso de uso, respuesta incorrecta	56
Ilustración 48: Caso de uso, equipo no supera el factor de corrección	56

Ilustración 49: Caso de uso, el equipo no supera el factor de corrección 2
Ilustración 50: Caso de uso, fin de la partida

57
57

1 INTRODUCCIÓN

*“La tecnología nos promete volvernos dueños de un mundo que podamos controlar con un botón.”
- Volker Grassmuck -*

Hace unos años, nadie creería que sus hijos estudiarían en aulas con ordenadores, proyectores y pantallas digitales en vez de utilizar libros o tomar notas de las lecciones del profesor. La tecnología ha llegado para quedarse en nuestras vidas y debemos ser capaces de aprovechar sus ventajas.

El objetivo de este proyecto es desarrollar una aplicación que sirva de apoyo al profesorado en las aulas para mejorar la enseñanza a través de la gamificación. Con el uso de juegos educativos, los profesores tendrán una valiosa herramienta que les permitirá evaluar el grado de aprendizaje de sus alumnos en clase al mismo tiempo que da lugar a un espacio donde se fomentan la adquisición de habilidades, el trabajo en equipo y el aprendizaje.

1.1 Motivación

Hoy en día los jóvenes nacen con un dispositivo electrónico bajo sus brazos y algunos incluso aprenden a manejarlos antes que a escribir. Dejando de lado las cuestiones morales que esto acarrea, la educación debe apoyarse en las tecnologías de la información para aprovechar sus ventajas y optimizar los procesos de enseñanza en las aulas. Ya somos testigos de este cambio: las aulas tienen proyector, pizarra digital e incluso ordenadores, ¿por qué no ir un paso más allá?

La propuesta del Proyecto AJDA es ofrecer juegos educativos para ser aplicados utilizando metodologías de gamificación en las aulas. El objetivo final es que el profesorado obtenga un feedback de su alumnado sobre cuánto han aprendido en clase, al mismo tiempo que ameniza a los estudiantes el proceso de aprendizaje con una aplicación que refuerce los conocimientos impartidos en clase.

1.2 Descripción del problema

El proyecto lleva varios años en desarrollo, pero se estaba llevando a cabo de forma fragmentada, creando módulos para implementar funcionalidades concretas, pero sin comunicación entre ellos. Esto estaba generando una deuda técnica considerable en el proyecto, ya que teníamos las piezas del rompecabezas, pero faltaba ensamblarlas.

Para contextualizar un poco más la situación, expondré dos de los principales problemas que presentaba la aplicación antes de la integración modular. Teníamos un módulo que permitía administrar las cuentas de los profesores, así como crear plantillas de partidas y lanzarlas, pero para ello necesitaba conectarse con el módulo de comunicaciones. También contábamos con un módulo que permitía generar ficheros de preguntas, pero necesitaba poder comunicarse con el módulo de administración para que leyera estos ficheros y los usara en sus plantillas.

Por otro lado, cada módulo se había desarrollado con tecnologías, librerías y versiones diferentes. Esto se debe a que cada módulo era tratado como un proyecto independiente por lo que cada desarrollador utilizaba una base de datos diferente, gestores de dependencias distintos e incluso entornos de desarrollo diferentes.

Por último, todo el conocimiento sobre el proyecto se encontraba repartido entre los distintos TFG que se han ido elaborando a lo largo del tiempo y no existía ningún punto centralizado con información sobre todo el proyecto, lo cual dificultaba muchísimo a aquellas personas que necesitaban aprender cómo funcionan los módulos y el proyecto en general.

1.3 Solución propuesta

Viendo la situación en la que se encontraba el proyecto, nacieron distintas líneas de trabajo con el fin de darle una solución a los problemas descritos en el apartado anterior:

1. Creación de un entorno de trabajo centralizado para todos los desarrolladores, eliminando las posibles incompatibilidades tecnológicas y problemas de versiones.
2. Integración de todos los módulos en un único proyecto donde cada módulo realiza su función apoyándose en los demás, ofreciendo así una aplicación funcional y consistente.
3. Creación de un repositorio de trabajo GIT para trabajar por ramas.
4. Creación de una biblioteca con documentación y vídeos detallados sobre el funcionamiento del proyecto.

Por otro lado, y con el fin de darle más valor a este proyecto, he estado trabajando en mejorar las funcionalidades de los módulos de administración y comunicación implementando nuevas funcionalidades que le dan valor al proyecto. Además, he creado una solución para desplegar todo el código en la nube teniendo la aplicación siempre disponible y pudiendo desplegar varias ramas al mismo tiempo para que aquellas personas que no sean desarrolladoras puedan probar los cambios que se realizan en cada rama sin que un desarrollador pueda dificultar el trabajo de otro.

A lo largo de la memoria, iré detallando cada apartado apoyándome en recursos que he creado en la web del proyecto, principalmente vídeos explicativos y documentación, los cuales estarán siempre disponibles para quién lo necesite. Al final de la memoria se encontrarán todas las referencias a estos recursos.

Una parte importante de mi trabajo en el proyecto ha sido ofrecer soporte a los distintos colaboradores que han ido trabajando en la aplicación. Al tener un conocimiento más extendido sobre el proyecto y su alcance, he orientado a los desarrolladores y resuelto dudas técnicas que les han ido surgiendo mediante reuniones virtuales y correos.

1.4 Estructura de la memoria

Dada la diversidad de los objetivos que comprenden la defensa, he pensado que lo mejor será dividirla por ámbitos y de manera cronológica por lo que la memoria desarrollará los siguientes puntos:

1. Creación del entorno de trabajo para los desarrolladores: se explicarán las tecnologías utilizadas para evitar futuros problemas de dependencias o inconsistencias en el desarrollo del proyecto.
2. Integración modular del proyecto: veremos cómo se unificaron todos los módulos ya desarrollados previamente, cada uno con tecnologías diferentes, en un único proyecto con tecnologías unificadas.
3. Creación y configuración del entorno en la nube: proveedor cloud elegido, consideraciones especiales a la hora de trabajar en la nube con el proyecto y automatización de subidas de código.
4. Listado pruebas realizadas contra la aplicación para comprobar sus funcionalidades.
5. Introducción a la web del proyecto, donde se encuentran todos los recursos y la documentación necesaria para empezar a trabajar con el repositorio.

2 TECNOLOGÍAS UTILIZADAS

En este apartado hablaremos sobre las tecnologías que se han utilizado para la creación del entorno de desarrollo y en los propios módulos integrados de la aplicación. Cabe destacar que los módulos de generación de ficheros y LTI no han sido adaptados a estas tecnologías una vez integrados.

Esto quiere decir que todas las tecnologías descritas a continuación aplican únicamente a los módulos de administración y comunicación, quedando la adaptación de las tecnologías de los otros módulos fuera del alcance de este proyecto. Para conocer en detalle las tecnologías utilizadas, recomendamos acudir a la documentación elaborada para ello [2] [13].

2.1 Java EE

Siguiendo los estándares que utilizaban los módulos anteriormente basados en modelo vista controlador (MVC), los módulos utilizan Java [28] como lenguaje base de programación orientada a objetos. Lo más importante es que Java EE permite la integración con tecnologías muy útiles que veremos en apartados posteriores como JPA, o Java Beans.

2.2 Apache Maven

Apache Maven [29] es una herramienta de gestión y comprensión de proyectos de software, basada en el concepto de un modelo de objeto de proyecto (POM, por sus siglas en inglés). Maven proporciona una manera uniforme de construir proyectos, una gestión de dependencias y un sistema de gestión de versiones.

Es una herramienta esencial para la gestión de proyectos Java, proporcionando una manera coherente y automatizada de construir, gestionar dependencias y desplegar aplicaciones.

```
<dependency>
  <groupId>org.primefaces </groupId >
  <artifactId>primefaces</artifactId>
  <version>6.2</version>
</dependency>
```

2.3 Java Beans

Un JavaBean [30] es una clase Java que sigue ciertas convenciones de nomenclatura y estructura, como tener un constructor sin argumentos, así como sus métodos getter y setter para acceder a sus variables desde el código JSF. Estas clases son muy útiles puesto que permiten la interacción entre backend y frontend, es decir, ofrece un mecanismo de control del flujo de la aplicación cuando el usuario interactúa con la interfaz web.

```
@ManagedBean
public class RegistroBean implements Serializable{
    .
    .
    .
}
```

2.4 Java Server Faces (JSF)

JavaServer Faces (JSF) es un framework de Java [31] para la creación de interfaces de usuario basadas en componentes para aplicaciones web. Está construido sobre el Modelo-Vista-Controlador (MVC) y facilita el desarrollo de interfaces web robustas y escalables mediante la reutilización de componentes de UI [12].

JSF se integra muy bien con otras tecnologías que utilizamos en el proyecto como es JPA (que explicaremos más adelante) o Primefaces .

```
<h:form id="temporizadorForm">
    <p:outputLabel id="tiempoCronometro" />
</h:form>
```

2.5 PrimeFaces

PrimeFaces es una biblioteca de componentes de interfaz de usuario (UI) para JavaServer Faces (JSF) [32]. Proporciona una amplia colección de componentes UI ricos y avanzados que facilitan el desarrollo de interfaces web atractivas y funcionales. PrimeFaces es conocido por su facilidad de uso y su capacidad para mejorar significativamente la apariencia y funcionalidad.

```
<p:chart id="grafica" type="pie"
    Model="vistaRegistroJugadores.pieModel1" />
```

2.6 CSS

CSS (Cascading Style Sheets) [33] es un lenguaje de hojas de estilo utilizado para describir la presentación de un documento. CSS define cómo se deben mostrar los elementos HTML en una página web, incluyendo aspectos como el diseño, los colores, las fuentes y otros atributos visuales.

En nuestro caso, CSS nos permite darle formato al estilo de la web puesto que las etiquetas JSF y Primefaces se traducen a etiquetas HTML.

2.7 WebSockets: Java + JavaScript

Los WebSockets [34] son una tecnología que permite establecer una conexión bidireccional persistente entre un cliente y un servidor, lo que permite una comunicación en tiempo real entre ambos. En Java, la API de WebSockets está incluida en la especificación Java EE (Enterprise Edition), y se puede utilizar para desarrollar aplicaciones web que requieran una comunicación bidireccional y en tiempo real.

En el lado del servidor se ha implementado la lógica de control de los WebSockets con Java mientras que por la parte del cliente las conexiones con el servidor websocket se realiza a través de JavaScript. JavaScript es un lenguaje de programación de alto nivel, interpretado y orientado a objetos. Es ampliamente utilizado en el desarrollo web para crear contenido dinámico, interactividad del usuario y aplicaciones web completas, aunque también se utiliza en otros ámbitos como es el nuestro a la hora de coordinar comunicaciones con WebSockets.

```
@ServerEndpoint("/websocket")
public class WebsocketController implements Serializable{
    .
    .
    .
}
```

2.8 Java Persistence API (JPA)

JPA es una especificación de Java [35] que describe una interfaz común para mapear objetos Java a tablas en una base de datos relacional y viceversa. Esta tecnología permite trabajar con datos de una base de datos utilizando objetos de Java de manera más sencilla, simplificando el acceso y la manipulación de datos en una base de datos relacional y abstrayendo gran parte de la complejidad asociada con el almacenamiento y recuperación de datos.

En nuestra aplicación, los accesos a la base de datos se realizan mediante la creación de clases DAO donde se implementan las funciones que interactúan con la base de datos a través de la librería JPA. Cada tabla utilizada en la base de datos es referenciada por una entidad en el código Java, donde se definen sus columnas y claves primarias para que la librería sepa interpretar los datos a la hora de interactuar con dicha tabla.

2.9 Logs: Simple Logging Facade for Java (SLF4J)

Interfaz de registro de eventos para Java [36], que proporciona una API de registro de eventos simple y unificada, permitiendo a los desarrolladores escribir código de registro independiente del framework de registro de eventos subyacente.

2.10 Gestión de idiomas: ResourceBundle

Los bundles [37] son clases que representan un conjunto de recursos localizables, como cadenas de texto, imágenes o archivos de propiedades, que pueden ser utilizados por una aplicación para admitir la internacionalización y la localización. Permiten a una aplicación adaptar su contenido y comportamiento según las preferencias culturales y lingüísticas del usuario.

Estos recursos se definen en el fichero faces-config.xml:

```
<resource-bundle>
  <base-name>bundles.TextoBundle</base-name>
  <var>msg</var>
</resource-bundle>
```

Una vez definidos, se pueden utilizar en los ficheros de código JSF a través de la variable definida para ello:

```
<title>#{msg.bienvenido}</title>
```

3 EL ENTORNO DE DESARROLLO

Este proyecto lleva varios años en desarrollo y durante ese tiempo distintos desarrolladores han implementado mejoras en los módulos, cada uno utilizando las herramientas que consideraban y generando una deuda técnica importante debido a la disparidad de tecnologías que había presentes en el proyecto hace unos años. Algunos desarrolladores habían optado por utilizar Eclipse como entorno de desarrollo, otros usaban Netbeans, unos módulos tenían gestionado la resolución de librerías con Apache Maven mientras que otros descargaban los archivos JAR manualmente y los incluían en el proyecto.

Hacia falta un cambio en la metodología de desarrollo, un entorno unificado que ofreciera a los desarrolladores un conjunto de herramientas para que todos avancen en la misma línea tecnológica. Después de valorar varias opciones, decidí implementar una solución basada en máquinas virtuales con Vagrant [10].

3.1 Vagrant

Vagrant [38] es una herramienta de software que proporciona una forma fácil y eficiente de crear y gestionar entornos de desarrollo virtualizados. Utiliza máquinas virtuales (VMs) y contenedores para configurar entornos de desarrollo consistentes y reproducibles. Vagrant es especialmente útil para desarrolladores, ya que permite definir entornos de desarrollo en un archivo de configuración y luego compartir esos entornos con otros miembros del equipo.

En nuestro caso, hemos optado por el proveedor de VirtualBox ya que es gratuito, aunque también existen otros proveedores de pago como VMware. Las imágenes (también conocidas como boxes) que se utilizan como base para desplegar las máquinas virtuales se encuentran ubicadas en un registro público [1].

3.1.1 Creación del entorno

La idea era generar una máquina virtual en la que ya se encuentren instaladas todas las herramientas que los desarrolladores necesitarán al mismo tiempo que permita actualizarse de manera sencilla si hubiera que añadir o quitar algo de la imagen.

Como imagen base se ha utilizado Ubuntu 20.04 ya que se trata de una versión LTS (Long Term Support) y se han ido instalando todas las herramientas que se detallarán después. Una vez instalados Vagrant y VirtualBox en el ordenador, hay que crear un directorio de trabajo donde iniciará Vagrant y se creará automáticamente el fichero de configuración llamado Vagrantfile. Además, dicho directorio se sincronizará automáticamente con el directorio `/vagrant` de la máquina teniendo así un directorio compartido entre el host anfitrión y virtualizado.

3.1.2 Vagrantfile

El Vagrantfile es el fichero de configuración de Vagrant. Se trata de un archivo de Ruby en el que se define y controla el entorno virtualizado. Admite un amplio repertorio de configuraciones, siendo las más interesantes:

- Proveedor: qué software de virtualización va a desplegar la imagen, este puede ser VirtualBox, VMware (de pago), Docker o Hyper-V.
- Box: imagen que se va a utilizar para desplegar la máquina virtual.
- Memory: memoria RAM que va a tener el entorno virtualizado

A continuación, un ejemplo del Vagrantfile utilizado para desplegar el entorno de desarrollo:

```
Vagrant.configure("2") do |config|
  config.vm.box = "proyectogamifica/gamificación"
  config.vm.version = "1.2.1"
  config.vm.provider "virtualbox" do |vb|
    vb.gui = true
    vb.memory = "4096"
  end
end
```


3.1.3 Personalización de la imagen base

Una vez desplegada la máquina, se instalarán una serie de aplicaciones y herramientas que facilitarán a los desarrolladores en sus tareas más rutinarias:

1. Idioma del teclado: por defecto, la imagen tiene configurado el teclado americano.
2. Interfaz de usuario: al principio lo único que ofrece la imagen oficial de Ubuntu es una consola por lo que se instalará un escritorio que se ajuste a nuestras necesidades de recursos en la máquina.
3. IDE: hemos optado por utilizar Eclipse y se configuran los metadatos para que funcione de manera óptima en la máquina.
4. Base de datos: se instala el agente de PostgreSQL.
5. Servidor de aplicaciones: Wildfly 22
6. Repositorio del proyecto: clonamos del repositorio la carpeta del proyecto para que cuando un nuevo desarrollador acceda al entorno solamente tenga que traerse los nuevos cambios.
7. Gestores de bases de datos: para facilitar el trabajo con la base de datos, se instalarán las herramientas PhpPgAdmin, PgAdmin3 y SQL Workbench.
8. Gestor GIT: para aquellos que lo necesiten, se instalará el software GitKraken que permite gestionar el trabajo con git de forma sencilla y visual. Es un programa de pago, pero con el paquete de estudiantes de GitHub regalan una licencia gratuita de un año.

3.1.4 Empaquetado y subida

Una vez creada y personalizada la imagen hay que hacer uso de Vagrant para construir una imagen basada en la máquina virtual que hemos configurado desde la imagen base de Ubuntu. Para ello basta con situarse en el directorio de trabajo y ejecutar el siguiente comando:

```
Vagrant package --output gamificación.box
```

Esto generará un fichero en nuestro directorio con la extensión *box* que contendrá nuestra imagen configurada. Una vez hecho esto subiremos la imagen al repositorio oficial de Vagrant para que los desarrolladores no tengan que ejecutar todos los pasos anteriores y puedan configurar el Vagrantfile directamente con el nombre de la imagen que vamos a subir. Es necesario tener una cuenta en el repositorio de Vagrant [1], en nuestro caso tenemos la cuenta para el proyecto donde subimos las nuevas versiones de la imagen.

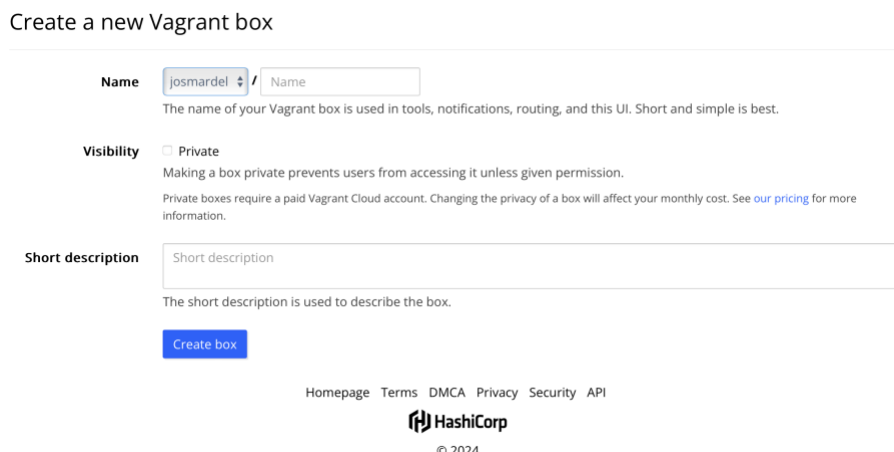


proyectogamifica/gamificacion 2.0.1	virtualbox	Downloads 52	Released 11 months ago
-------------------------------------	------------	-----------------	---------------------------

Vagrant box que contiene el entorno de desarrollo del proyecto de gamificación. Entorno basado en Ubuntu 20.04 LTS.

Ilustración 1: Repositorio Vagrant del Proyecto Gamifica

Para subir la imagen basta con indicar el nombre de la imagen, que irá precedido del nombre de la cuenta, ajustar los permisos y crear una nueva versión indicando el proveedor y seleccionando el fichero generado anteriormente en nuestro ordenador.



Create a new Vagrant box

Name: josmardel / Name

The name of your Vagrant box is used in tools, notifications, routing, and this UI. Short and simple is best.

Visibility: Private

Making a box private prevents users from accessing it unless given permission.

Private boxes require a paid Vagrant Cloud account. Changing the privacy of a box will affect your monthly cost. See our pricing for more information.

Short description: Short description

The short description is used to describe the box.

Create box

Homepage Terms DMCA Privacy Security API

HashiCorp

© 2024

Ilustración 2: Crear imagen Vagrant

New Box Provider

Provider:
Vagrant will need to know how to use this provider.

File Hosting: Upload to Vagrant Cloud
 External URL

Checksum type:

Checksum:

Box architecture:
 Default provider architecture

[Continue to upload](#)

Ilustración 3: Añadir proveedor para la imagen Vagrant

3.2 Repositorio GIT

Como mencioné en los problemas existentes, cada módulo constituía un proyecto independiente que dificultaba la comunicación entre ellos. Es por esto por lo que la estructura del proyecto ha sufrido algunos cambios, el primero y más significativo es la unificación de todos los módulos en un único directorio de trabajo (en vez de cuatro directorios). Por otro lado, las comunicaciones con la base de datos se realizaban a través de JPA en un proyecto independiente que desplegaba una API Rest y manejaba las llamadas a los endpoints para realizar las consultas SQL. Esto está bien si queremos externalizar los accesos a la base de datos, pero no era nuestro caso por lo que todo ese proyecto ha sido eliminado y las funciones que gestionan los accesos a la base de datos han sido integradas en cada módulo. [22]

3.2.1 El nuevo repositorio

El nuevo repositorio se ha creado en base a una plantilla de proyecto Maven en Eclipse, el cual utiliza los ficheros *pom.xml* para definir su estructura además de otras funcionalidades como son la gestión de librerías o el empaquetado del código en ficheros JAR o WAR. Dentro encontraremos un directorio *target* donde Eclipse guarda todo el código que compila: clases Java, la interfaz web empaquetada en formato WAR, etc... [11]

3.2.2 Código Java

Todos los ficheros Java se utilizan principalmente para controlar el backend de la aplicación, aunque también sirven para definir constantes o parámetros de configuración como veremos más adelante [11]. Estos ficheros se ubican en el directorio *java*, el cual está dividido en subdirectorios donde se agrupa el código por módulos. Aquí encontraremos recursos que implementan funcionalidades distintas: accesos a la base de datos con JPA, ManagedBeans, Pojos, controlador de Bundles y clases Java de todo tipo (constantes, parámetros de configuración, etc...).



Ilustración 4: Código Java en el repositorio

3.2.3 Código JSF/Primefaces

Estos ficheros con extensión *.xhtml* se encuentran en el directorio *webapp* e implementan las interfaces de usuario mediante el uso del framework JSF y Primefaces, así como la comunicación con las clases Java gracias a los ManagedBeans, que controlan el flujo de la aplicación. Como resultado de la integración modular del proyecto, el directorio está organizado por carpetas que agrupan el código por módulos [11].

3.2.4 Código JSP/HTML

Dentro del directorio mencionado anteriormente se encuentra el código del módulo LTI, el cual escapa al alcance de este TFG, pero cabe mencionar que su código no ha sido migrado a JSF aún y sigue utilizando ficheros JSP y HTML [11].

3.2.5 Código XML

Estos ficheros se utilizan principalmente para definir la configuración de recursos como la base de datos o la web que se va a desplegar [11]. Podemos distinguir principalmente cuatro tipos de ficheros XML en el proyecto:

1. *pom.xml*: define la estructura del proyecto Maven y permite gestionar las librerías y sus versiones de una manera eficiente. Además, se pueden configurar otros parámetros como el nombre del fichero que resulta de compilar o empaquetar el código (*.JAR / .WAR*).
2. *persistence.xml*: fichero de configuración donde se define la conexión con la base de datos: driver, usuario, contraseña, puerto, entidades JPA.

A continuación, un ejemplo del fichero de configuración de JPA con los parámetros de acceso a la base de datos:

```
<properties>
  <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://URL" />
  <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver" />
  <property name="javax.persistence.jdbc.user" value="dit" />
  <property name="javax.persistence.jdbc.password" value="dit" />
</properties>
```

3. *faces-config.xml*: fichero de configuración de JSF donde se definen los ManagedBeans y Bundles.
4. *web.xml*: fichero de configuración que utiliza Java para desplegar webs, principalmente utilizado para definir los servlets.

3.2.6 Código CSS y JavaScript

Los estilos de la web se personalizan en un fichero llamado *gamificacion.css* el cual se encuentra ubicado en el directorio *resources*. En lo referente al código JavaScript, es de los ficheros de código más importantes puesto que definen la comunicación websocket entre el cliente y el servidor [11]. Es importante conocer para qué sirve cada archivo de código y cómo están organizados dentro de la carpeta *javascript*:

- *crearpartida.js*, *formCheck.js* y *funciones.js*: coordinan la comunicación entre los jugadores y el servidor.
- *juego-js*: directorio donde está el fichero que coordina la comunicación con los juegos (no se toca).
- *lib-js*: directorio donde está el fichero que coordina la comunicación con el proyecto Descartes (no se toca).
- *servidor-js*: directorio donde están los ficheros que coordinan la comunicación entre el servidor (profesor) y los clientes (jugadores).

3.2.7 Juegos

Merece la pena reservar un apartado para hablar de los juegos que tenemos disponibles e integrados en la aplicación [11]. Actualmente tenemos un total de cinco juegos disponibles, cada uno tiene sus peculiaridades y utiliza componentes distintos del proyecto.

Los juegos agrupan a los jugadores en grupos y las respuestas finales se componen del total de respuestas enviadas por cada participante del grupo. Esto se ha implementado así con el fin de tener más jugadores conectados al mismo tiempo, evitando el envío de un elevado número de respuestas al juego que pueda llegar a afectar en la latencia de la partida. Además, existe un elemento configurable muy importante en cada partida llamado factor de corrección que es importante conocer.

3.2.7.1 Factor de corrección

Este parámetro indica el porcentaje mínimo de respuestas que deben enviar los participantes para que la opción marcada se envíe como respuesta del grupo al juego [25]. Esto es completamente configurable desde el menú de administración para aquellos juegos que soporten este parámetro. En el caso de los juegos con cifras, hay un parámetro similar que indica cuántas cifras puede diferir la respuesta dada de la respuesta correcta para darla como válida.

Gracias al factor de corrección podemos tener una configuración de jugadores por equipos que permite la conexión de más jugadores sin que se vean afectada la latencia de la partida.

3.2.7.2 Test

Es el juego más básico, con un formato preguntas y respuestas sencillo que te da un tiempo para responder la pregunta que aparece por pantalla y cuando acaba el tiempo o el profesor lo fuerza, el servidor manda las respuestas al juego y devuelve una puntuación [17].

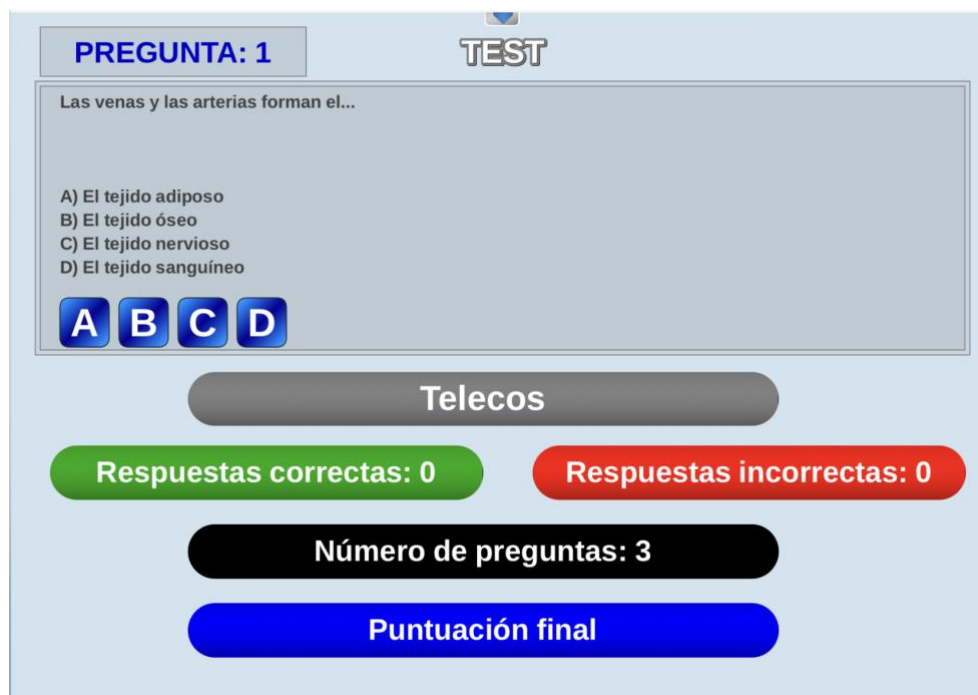


Ilustración 5: Juego test

3.2.7.3 Test con cifras

Sigue la misma estructura del juego test normal pero las respuestas son cifras que deben escribir los jugadores desde sus dispositivos. La respuesta enviada al juego será la media aritmética de las respuestas del grupo [18].

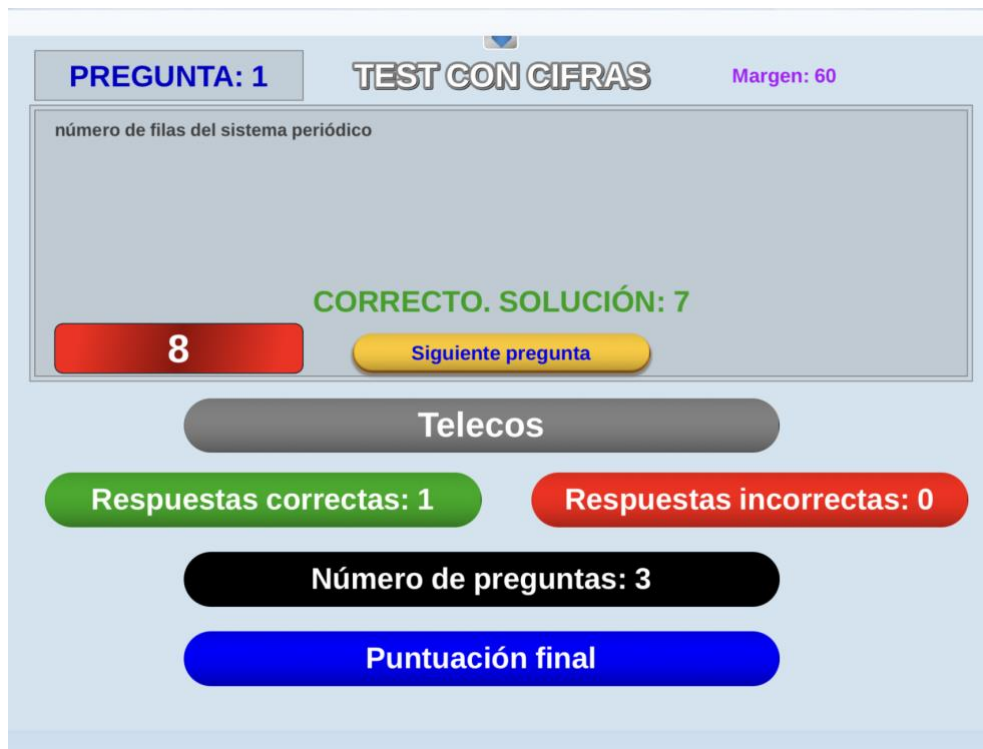


Ilustración 6: Juego test con cifras

3.2.7.4 Artificieros

El objetivo del juego es desactivar una bomba con seis cables conectados, siendo uno de ellos el que evita la explosión. Tras cada pregunta respondida correctamente, el equipo tendrá que votar qué cable cortan hasta que den con el correcto o explote la bomba [19]. Si ningún jugador vota una de las opciones, el servidor automáticamente cortará un cable aleatorio.



Ilustración 7: Juego artificieros

3.2.7.5 Bombilla

Competición entre jugadores para ver quién consigue responder cinco preguntas seguidas antes que el resto. Si un jugador falla una pregunta, su contador se reiniciará a cero de nuevo [21].



Ilustración 8: Juego bombilla

3.2.7.6 Batalla naval

Competición entre dos equipos que se encuentran en buques distintos. Cada pregunta acertada por un equipo dispara una bala de cañón al buque del equipo contrario, el primer buque que reciba cinco disparos quedará hundido y perderá la partida [20].



Ilustración 9: Juego batalla naval

3.2.8 Otros recursos

Además de los recursos ya mencionados, los cuales son el esqueleto de la aplicación, existen otros recursos que merece la pena mencionar [11]:

- config.properties: archivo del proyecto LTI donde se configura la conexión a la base de datos SQLite.
- Imágenes: recursos de imágenes que utiliza la interfaz web.
- TextosBundles.properties: ficheros donde se definen los recursos para el uso de Bundles.

3.2.9 La base de datos

Hasta el momento, la aplicación utiliza una base de datos PostgreSQL en todos sus módulos [14] a excepción del módulo de accesos LTI, el cual queda fuera del alcance de este TFG. A continuación, se mostrarán unos diagramas entidad relación con la estructura de la base de datos y posteriormente se explicará para qué se utiliza cada tabla en la aplicación.

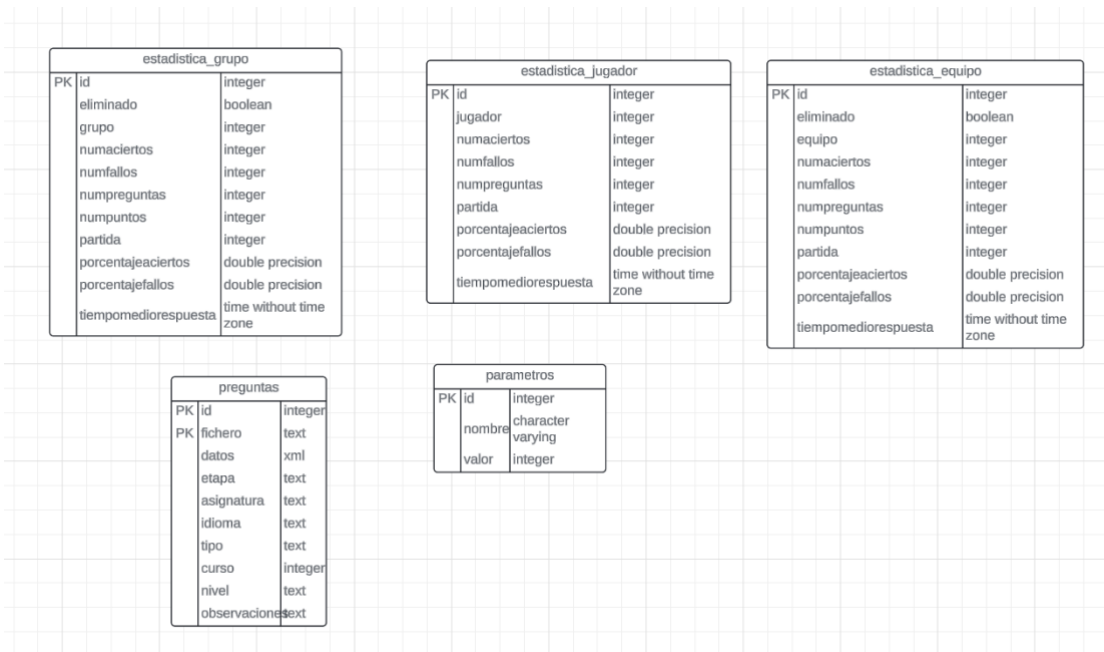


Ilustración 10: Diagrama entidad relación 1

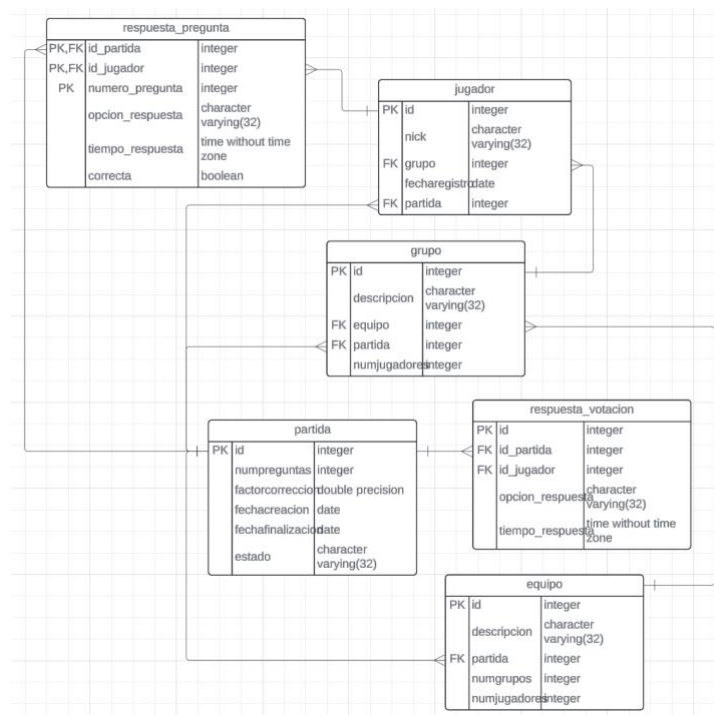


Ilustración 11: Diagrama entidad relación 2

3.2.9.1 Estadísticas grupo

Almacena métricas para un grupo, sin uso en el estado actual del proyecto.

3.2.9.2 Estadísticas jugador

Almacena métricas sobre un jugador, sin uso en el estado actual del proyecto.

3.2.9.3 Estadísticas equipo

Almacena métricas para un equipo, sin uso en el estado actual del proyecto.

3.2.9.4 Parámetros

Tabla donde se guardan los parámetros de configuración del proyecto.

3.2.9.5 Preguntas

Entidad para almacenar preguntas, utilizada por el módulo de gestión de ficheros.

3.2.9.6 Respuestas pregunta

Histórico de respuestas a preguntas, indicando el jugador que respondió, la partida en la que estaba registrado, la opción que respondió y si lo hizo correctamente o falló.

3.2.9.7 Respuesta votación

Análogo a la tabla anterior, pero para las respuestas a votaciones (en el juego artificieros, por ejemplo).

3.2.9.8 Jugador

Entidad que define al jugador de una partida. Almacena datos como el nombre, la partida que ha jugado y el grupo en el que lo hizo además de la fecha en la que se registró.

3.2.9.9 Grupo

Define un grupo en la aplicación. Para ello, esta entidad cuenta con un identificador y un nombre además de otros parámetros como la partida en la que se encontraba y el número máximo de jugadores que admitía.

3.2.9.10 Equipo

Análogo a la tabla para los grupos.

3.2.9.11 Partida

Entidad con información básica sobre las partidas. Lo más interesante de esta tabla son el factor de corrección configurado, el número de preguntas que tenía la partida, cuándo se creó y su estado actual.

3.2.9.12 Juegos

Entidad que define un juego. Contiene mucha información de configuración que después utiliza el módulo de administración para crear las partidas, pero lo más interesante de esta entidad es la columna *jsondatos* ya que contiene los datos iniciales por defecto que se le pasan al módulo de multijugador al arrancar la partida.

3.2.9.13 Partidas

Entidad con información adicional sobre una partida, lo más importante es el fichero de preguntas elegido para esta partida junto con el código de la partida y el tiempo de respuesta configurado.

3.2.9.14 Profesores

Define la entidad de los usuarios con toda la información que tenemos de ellos, siendo las columnas más importantes el rol del profesor, la fecha de registro y contraseña.

3.2.9.15 Configuración partidas

Entidad que contiene toda la información sobre las plantillas de configuración de partidas que posteriormente utiliza el módulo de administración para crear partidas. Es la entidad más grande puesto que la aplicación admite múltiples configuraciones que deben recogerse en la entidad.

4 MEJORAS EN LOS MÓDULOS

Una parte de mi trabajo ha consistido en, una vez unificado el entorno y hacerlo funcional [9], llevar a cabo mejoras en los módulos de administración y comunicación. En este contexto, hay una serie de adaptaciones que se han llevado a cabo en los dos módulos por igual, dejando de lado tecnologías obsoletas o con falta de documentación por otras más reconocidas y utilizadas en el mundo del desarrollo software.

4.1.1 Tecnologías y buenas prácticas de desarrollo

Tomando como referencia los criterios previamente mencionados, elaboré una lista de tecnologías sobre las que se debía basar el proyecto con todos sus módulos [13]. Me he encargado de ajustar los dos módulos del TFG para que sigan las reglas que listaré a continuación:

1. **Los accesos a la base de datos deben hacerse a través de JPA**, teniendo que rediseñar los accesos para el módulo de administración que se basaban en Spring.
2. **Las librerías se gestionarán con Apache Maven**, teniendo que crear un fichero pom.xml que gestiona las dependencias de cada módulo como ya se mostró anteriormente y eliminando las librerías .jar que se estaban añadiendo a mano.
3. **Las URL deben ser dinámicas [23], esto significa que no debe haber nada escrito a mano**. En nuestro caso, tuve que definir una clase Java para obtener la IP dinámicamente y no escribirla a mano. Esta clase hace uso de un proveedor externo que nos proporciona la dirección IP pública del host que realiza la petición

```
public Class Ip{
    String ip;
    ...
    public String getIp(){
        URL url = new URL(https://ipinfo.io/ip);
        ...
        return ip;
    }
}
```

4. Rutas relativas, evitando hacer referencias a rutas absolutas. Por ejemplo, para referenciar los recursos de los juegos (carátulas o capturas), usaremos rutas relativas:

```
this.setCaratulaSrc("../resources/juegos/"...);
```

5. Estilo web responsive, para ello instalé un tema de Bootstrap para Primefaces que instalaba una amplia lista de configuraciones CSS.
6. La gestión de los logs se llevará a cabo con SLF4J, instalando la librería y sustituyendo todas las trazas por la clase Logger. Su uso es muy sencillo, basta con instanciar la clase con el nombre de la clase Java en la que se va a utilizar y llamar a sus métodos:

```
Logger logger = LoggerFactory.getLogger(nombreClase.class);
logger.info("Traza de información");
logger.error("Traza de error");
logger.debug("Traza de debug");
```

7. Constantes y parámetros de configuración aislados del código, teniendo dos fuentes de datos disponibles: ficheros de constante java y base de datos con parámetros. En el módulo de administración utilizamos constantes que están definidas en ficheros Java, por ejemplo, las rutas relativas donde se descomprimen los juegos:

```
public Class Rutas {
    public static final String rutaZip = "../../../webapp/resources/juegos";
}
```

Por otro lado, tenemos los parámetros de configuración más sensibles que se encuentran en una tabla de la base de datos.

id	nombre	valor
1	ncodigo	5
2	secreto_correo	6279973
3	proyecto.gamifica@gmail.com	28438092

Ilustración 13: Constantes en base de datos

8. Portal web multi-idioma con el uso de los recursos Bundle [13].

Ilustración 14: Gestión de idiomas con Bundles

9. Nomenclatura común para las clases, ficheros y directorios, evitando el uso de caracteres como la barra horizontal o la barra baja y usando mayúsculas para distinguir los nombres compuestos.

```
src/main/java/administracion
src/main/java/lti
src/main/java/multiJugador
src/main/java/generadorFicheros
```

Ilustración 15: Nomenclatura en el repositorio

10. Uso de anotaciones Javadoc para la generación de documentación. Haciendo uso de las anotaciones predefinidas, documentamos cada clase para que Javadoc sea capaz de generar la documentación una vez terminemos la implementación. Todas las clases llevan las anotaciones básicas de autor y versión:

```
/**
 * Nombre de la clase Bean que controla el comportamiento de un fichero
 * JSF
 * @author José Manuel Martínez Delgado
 * @version 1.0 (28/09/2023)
 */
```

11. Código web basado en el framework JSF, adaptando todos aquellos ficheros de código que estaban escritos en HTML o JSP.

4.1.2 Mejoras en el módulo de administración

En primer lugar, se corrigieron algunos errores residuales de la integración que habían roto algunas funciones básicas como la recuperación de contraseñas o la subida de juegos a la plataforma. Dejando eso a un lado, se han llevado a cabo mejoras en el funcionamiento del módulo, así como nuevas funcionalidades [15] [22] que le permiten integrarse con el módulo de comunicación:

- Implementación de botones de confirmación en todos los procesos que ejecuten una acción con consecuencias para el proyecto. Aquí podemos destacar el botón de desconexión o los botones para borrar recursos.

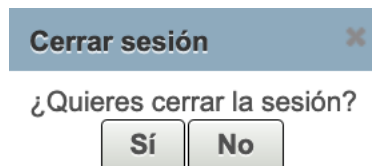


Ilustración 16: Mejora 1, botones de confirmación

- Botón para eliminar todas las plantillas y todas las partidas en curso, con su correspondiente doble verificación.



Ilustración 17: Mejora 2, botón de borrar todos los recursos

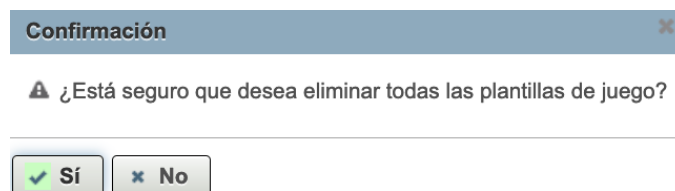


Ilustración 18: Mejora 2, confirmación

- A la hora de lanzar una plantilla puedes elegir arrancar la partida o simplemente crearla para empezarla más adelante. Además, puedes personalizar el título de la partida, aunque siempre llevará un prefijo con la hora y fecha en que se creó.

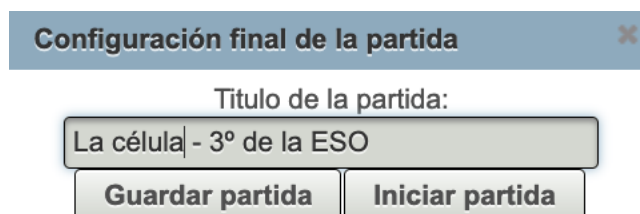


Ilustración 19: Mejora 3, configuración final de la partida

- Interfaz web (Perfil) que muestra la información sobre la cuenta del usuario y te permite cambiar la contraseña.

Datos del usuario

Nombre: Super
Apellidos: Admin
Correo: super@admin.com
País: España
Rol: SUPER
Fecha de registro: Sun Jun 25 00:00:00 UTC 2023
Nueva contraseña *

Repetir contraseña: *

Cambiar la contraseña

Ilustración 20: Mejora 4, panel de información de usuario

- Enlace a los módulos de ficheros y LTI desde un menú desplegable.

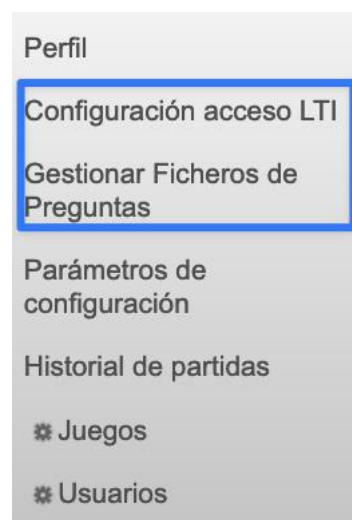


Ilustración 21: Mejora 5, integración con otros módulos

- Interfaz web (Parámetros de configuración) que te muestra valores de configuración que se encuentran guardados en la base de datos como el número de cifras que tendrá el código de acceso del cual hablaremos más adelante.

Parámetro	Valor	
ncodigo	5	
secreto_correo	6279973	
proyecto.gamifica@gmail.com	28438092	

Ilustración 22: Mejora 6, interfaz de parámetros

- Interfaz web (Historial de partidas) que muestra el histórico de partidas jugadas con información básica como su ID, quién la creó, el resultado o la fecha en la que se creó. Además, cada entrada de la tabla tiene un botón que en un futuro sacará estadísticas gracias a la integración de este módulo con otro que está actualmente en desarrollo.

<input type="checkbox"/>	ID	Título	Profesor	Fecha de creación	Fecha de update	¿Partida finalizada?	Juego	Resultados
<input type="checkbox"/>	913	Test normal	Super Admin	2024-03-17T22:46:57	2024-03-17T22:48	<input checked="" type="checkbox"/>	Test	<input type="button" value="Ver"/>
<input type="checkbox"/>	915	Test normal	Super Admin	2024-03-17T23:01:48	2024-03-17T23:02:40	<input checked="" type="checkbox"/>	Test	<input type="button" value="Ver"/>
<input type="checkbox"/>	1084	Test normal	Super Admin	2024-05-27T23:52:42	2024-05-27T23:53:46	<input checked="" type="checkbox"/>	Test	<input type="button" value="Ver"/>
<input type="checkbox"/>	917	Test normal	Super Admin	2024-03-25T18:23:17	2024-03-25T18:28	<input checked="" type="checkbox"/>	Test	<input type="button" value="Ver"/>
<input type="checkbox"/>	1020	Test artificieros v4	Super Admin	2024-05-09T19:29:46	2024-05-09T19:34:37	<input checked="" type="checkbox"/>	Artificieros	<input type="button" value="Ver"/>
<input type="checkbox"/>	1021	Test artificieros v4	Super Admin	2024-05-09T19:34:40	2024-05-09T19:36:26	<input checked="" type="checkbox"/>	Artificieros	<input type="button" value="Ver"/>
<input type="checkbox"/>	922	Test cifras	Super Admin	2024-03-25T19:02:25	2024-03-25T19:05:05	<input checked="" type="checkbox"/>	Test con cifras	<input type="button" value="Ver"/>
<input type="checkbox"/>	1049	Test artificieros v4	Super Admin	2024-05-11T15:58:04	2024-05-11T15:59:18	<input checked="" type="checkbox"/>	Artificieros	<input type="button" value="Ver"/>
<input type="checkbox"/>	925	Test cifras	Super Admin	2024-03-25T19:33:46	2024-03-25T19:35:35	<input checked="" type="checkbox"/>	Test con cifras	<input type="button" value="Ver"/>
<input type="checkbox"/>	1043	Test artificieros v4	Super Admin	2024-05-10T17:27:02	2024-05-10T17:28:09	<input checked="" type="checkbox"/>	Artificieros	<input type="button" value="Ver"/>

Ilustración 23: Mejora 7, interfaz de partidas jugadas

- Eliminación de la API REST, que se desplegaba con Spring en otro puerto para gestionar los accesos a la base de datos. Este elemento era innecesario ya que la base de datos se encuentra en la misma máquina que el resto del proyecto por lo que únicamente estaba ocupando espacio y elevando la latencia en la ejecución del código.
- Integración con el módulo de comunicación WebSockets. Cuando se crea una partida desde la web de administración, automáticamente se abre una nueva pestaña con la interfaz web de gestión de la partida del profesor (esto ya pertenece al módulo websocket). Por detrás, se ha ajustado el código para que entre módulos haya una comunicación ya que se necesita información generada en la web de administración para el correcto funcionamiento del módulo websocket.
- Se han eliminado parámetros innecesarios para algunos juegos y en la configuración de plantillas.
- Implementación de un botón que permite subir un fichero con la batería de preguntas que se lanzarán posteriormente en el módulo de los juegos. Este será el punto donde se debe integrar el módulo de ficheros en el futuro.

Configuración Partida
✕

General

Juego

Configuración Partida

Grupos y Equipos

Configuración Partida

- Fichero de preguntas

act-3eso-celula1.txt

+ Cambiar fichero

➔ Subir

⊗ Cancelar

Gestionar Ficheros de Preguntas

Factor de corrección:

Tiempo de respuesta (segundos):

Nº de preguntas:

¿Preguntas aleatorias?

¿Opciones de respuesta aleatoria?

Jugadores por equipo (max):

← Atrás

→ Siguiente

Ilustración 24: Mejora 8, subida de ficheros de preguntas

- Mejoras en el panel de administración de usuarios. El profesor con rol super administrador tendrá a su disposición, aparte de las funcionalidades previas para modificar los datos y roles de un usuario, la posibilidad de modificar la contraseña de otro usuario sea cual sea su rol.

	Mail	Contraseña	Apellidos:
<input type="checkbox"/>	<input style="width: 90%;" type="text"/>	<input style="width: 90%;" type="text"/>	<input style="width: 90%;" type="text"/>
<input type="checkbox"/>	adjamail52@gmail.com	<input style="width: 90%;" type="text"/>	Profesor
<input type="checkbox"/>	administrador@admin.com	<input style="width: 90%;" type="text"/>	Admin
<input checked="" type="checkbox"/>	josmardel8@alum.us.es	Martínez
<input type="checkbox"/>	proyecto.gamifica@gmail.com	<input style="width: 90%;" type="text"/>	NO BORRAR
<input type="checkbox"/>	super@admin.com	<input style="width: 90%;" type="text"/>	Admin
<input type="checkbox"/>	test@gmail.com	<input style="width: 90%;" type="text"/>	
<input type="checkbox"/>	test2@gmail.com	<input style="width: 90%;" type="text"/>	TEST

Ilustración 25: Super usuario modifica la contraseña de otro usuario

4.1.3 Mejoras en el módulo de comunicaciones

El módulo de comunicaciones tiene como funcionalidad principal la preparación de una sala virtual (similar a un servidor) a la que se unen jugadores (clientes) para posteriormente lanzar un juego que requiere de una comunicación entre cliente y servidor gestionada por este módulo [16]. El problema que había al principio era que la información referente a la configuración de la partida no venía del módulo de administración, sino que se rellenaban a mano, lo cual no es nada eficiente. Al implementar la comunicación entre módulos, ahora los parámetros de configuración vienen de la plantilla de la partida creada en el módulo de administración.

Con estos dos módulos conectados ya podíamos simular un caso de uso real en el que un profesor crea una plantilla y lanza una partida en una ventana nueva que utiliza las funcionalidades del módulo de comunicaciones para llevar a cabo una partida totalmente configurada y sin ninguna acción manual. Con el fin de darle valor al proyecto, he implementado una serie de mejoras en este módulo que detallaré a continuación:

- Sistema de generación de códigos para acceder a las partidas. Antes se accedía con el ID de la partida, que era simplemente el índice que la base de datos autogeneraba al crear una nueva entrada en la tabla. Ahora, automáticamente se genera un código de X números (por defecto 5 cifras) cuya extensión puede ser configurada en un parámetro de configuración ubicado en la base de datos.



Ilustración 26: Mejora 9, códigos de acceso

- Sistema de control de accesos. El módulo comprueba cada vez que un jugador intenta acceder a una partida si su grupo está lleno o no. En caso de estar lleno, comprueba si los otros grupos también lo están y en caso afirmativo bloquea la pantalla del jugador que quiere entrar y cuando queda un hueco libre le avisa desbloqueando la pantalla. En caso de que hubiera otro grupo con hueco, lo indica en la pantalla y le permite seleccionar otro grupo.



Ilustración 27: Mejora 9, control de accesos

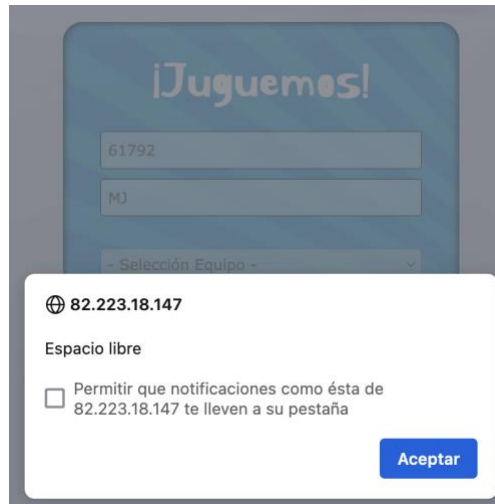


Ilustración 28: Mejora 9, control de accesos 2

- Al comenzar el juego, la pantalla queda totalmente limpia ya que el menú con la información de los jugadores registrados se minimiza para que el juego quede totalmente a la vista.
- Implementación de un pequeño menú de control para el profesor en el que se puede parar, iniciar o reiniciar el tiempo de cada pregunta o votación. Este menú también permite forzar la respuesta, mandando únicamente al juego las respuestas recibidas hasta el momento. Por último, el profesor tiene a su disposición un botón que finaliza la partida sin importar el estado en el que se encuentre.

Panel de control

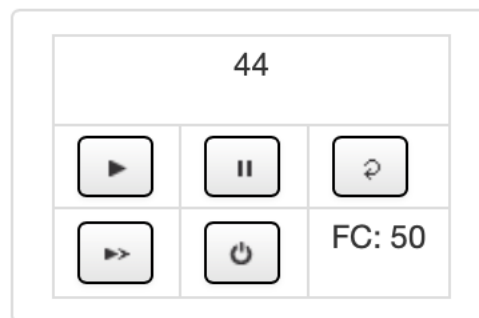


Ilustración 29: Mejora 11, panel de control

- Sistema de notificaciones al profesor y alumno. Cuando un jugador se desconecta, el profesor es notificado y el menú de jugadores registrados se actualiza dinámicamente. En el caso de los jugadores, si el profesor los expulsa del juego son notificados. El profesor recibe además notificaciones adicionales como la respuesta nula por parte de los jugadores al no haber llegado al factor de corrección.

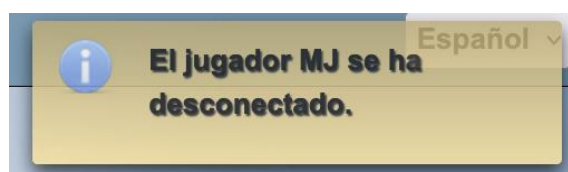


Ilustración 30: Mejora 9, control de accesos 3

- En la interfaz web de los jugadores, se ha implementado un indicador de conexión que se mostrará en verde si la conexión con el servidor es correcta o en rojo si dicha conexión websocket se ha cerrado.



Ilustración 31: Mejora 10, estado de conexión activa

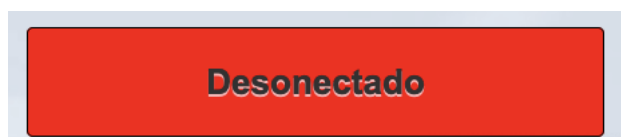


Ilustración 32: Mejora 10, estado de conexión terminada

- Mejoras en la interfaz de gestión del profesor, con control absoluto sobre la partida y visión de los jugadores registrados.

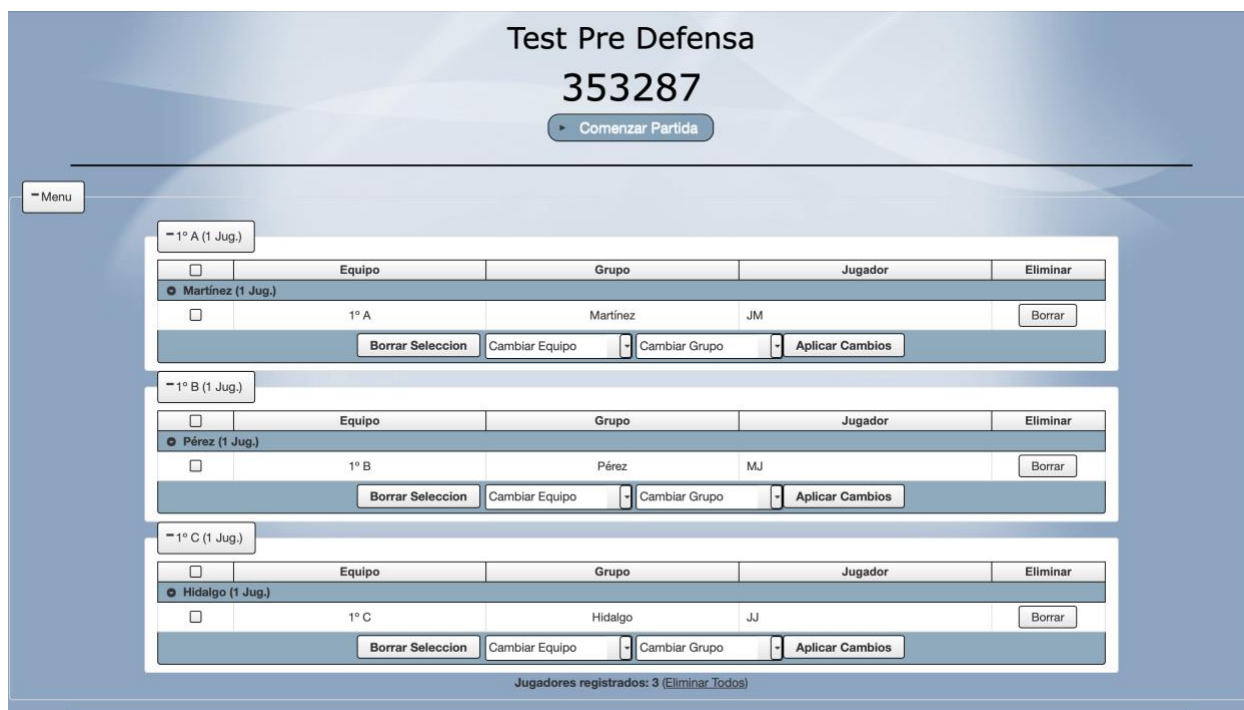


Ilustración 33: Mejora 12, interfaz del profesor

5 ENTORNO EN LA NUBE

Como aplicación educativa, este proyecto debe contar con alta disponibilidad durante todo el día y ser accesible para todos sus usuarios. Con el fin de darle aún más valor a la aplicación, he implementado una solución cloud para desplegar el proyecto en un servidor dedicado a través del proveedor IONOS [6] [24].

5.1 El servidor

La máquina virtual que nos ofrece el proveedor cuenta con las siguientes especificaciones:

- Dirección IPv4 pública: 82.223.18.147
- 2 GB de memoria RAM
- Procesador de 2 núcleos
- 80 GB de memoria no volátil (SSD)
- Sistema operativo Ubuntu 20.04

5.2 El proveedor

La plataforma IONOS nos permite configurar una serie de recursos que resultan de interés de cara a la administración del servidor. La principal configuración de interés es el firewall, donde podemos crear nuevas políticas de acceso para los servicios que requieran comunicación con el servidor (principalmente SSH).

Entrada

Acción ▾	IP permitida ▾	Protocolo ▾	Puerto(s) ▾	Descripción ▾	
Permitir	Todas	TCP	22	SSH	 
Permitir	Todas	TCP	80	Tráfico HTTP	 
Permitir	Todas	TCP	443	Tráfico HTTPS	 
Permitir	Todas	TCP	8443	Puerto alternativo Tomcat (deprecado)	 
Permitir	Todas	TCP	8447	Plesk	 
Permitir	Todas	ICMP		Pruebas de conexión ping	 
Permitir	Todas	TCP	9990	Administración de Wildfly	 

Ilustración 34: Firewall instancia cloud

Con la licencia contratada actualmente también nos ofrecen un dominio gratuito para evitar escribir la dirección IP en el buscador cada vez que queramos acceder. El último servicio gratuito que nos ofrecen es relativo a la administración de la instancia, pudiendo forzar un reinicio o apagado de la máquina si fuera necesario o abrir una consola que se conecta con el servidor por si no se pudiera acceder por SSH u otros medios.

5.3 Preparación del entorno cloud

La finalidad de esta instancia es mantener la aplicación activa en todo momento. Para ello, es necesario desplegar el proyecto en el servidor Wildfly y configurar la base de datos para cargar los archivos. En el caso de la instalación de Wildfly es muy sencillo puesto que no requiere de configuración adicional una vez instalado y arrancado en la instancia. Para la base de datos, diseñé unos scripts de código SQL que generan toda la estructura de la base de datos automáticamente y carga algunos datos básicos como son algunos parámetros por defecto del módulo de administración o ficheros de pregunta por defecto que los profesores tienen disponibles desde el primer momento.

Por otro lado, creé algunos usuarios en la instancia con funcionalidades específicas:

- Usuario wildfly: encargado de desplegar el código empaquetado .WAR en el servidor.
- Usuario ditssh: usuario que utiliza el script de despliegue para las transferencias de datos.
- Usuario ditpostgres: tareas de comunicación con la base de datos.

5.4 Automatización de subidas a la nube

Con el fin de tener un procedimiento de subida de código a la nube automatizado para todos los desarrolladores, he creado un script para que automáticamente empaquete el código, exporte la base de datos local y despliegue los cambios en la instancia. Este script se ejecuta desde el ordenador local del desarrollador y en cuestión de unos 3 o 4 minutos el código está desplegado en el entorno cloud. ¿Cómo funciona el script?

En primer lugar, es necesario empaquetar todo nuestro código en un archivo que tiene el formato WAR. Un archivo WAR es un archivo JAR utilizado para distribuir una colección de JavaServer Pages, servlets, clases Java, archivos XML, bibliotecas de tags y páginas web estáticas que juntos constituyen una aplicación web. Para generarlo desde una CLI (interfaz de línea de comandos) haremos uso de la API que ofrece Apache Maven para compilar el proyecto y generar el fichero que necesitamos.

```
cd ~/gamificación/Gamificacion/webAdministracion
mvn clean install &>/dev/null; echo OK
```

Una vez creado el fichero WAR, vamos a exportar la base de datos mediante la herramienta *pg_dump* y subiremos dicho fichero al entorno cloud con otra herramienta llamada *sshpass*, la cual permite realizar una autenticación SSH para posteriormente poder ejecutar la copia con *scp*.

```
pg_dump -U dit gamificación-$1 > ~/$pgfilename; echo OK
sshpass -p $sshpwd scp ~/$pgfilename ditssh@host:/etc/bd/$1
```

Cabe destacar que la mayoría de los valores están parametrizados por lo que a la hora de ejecutar el script habrá que indicarle como parámetro de llamada el nombre de la rama de trabajo del desarrollador para poder desplegar varias ramas al mismo tiempo en el entorno como veremos en apartados posteriores.

Una vez exportada la base de datos y enviada al entorno cloud, hay que parar el servicio postgresql y el despliegue actual en wildfly para poder iniciar de nuevo la base de datos con la información actualizada. Una vez arrancada, se llevará a cabo una conexión remota al servidor haciendo uso de un script que proporciona wildfly (*jboss-cli.sh*) para desplegar la nueva versión de la aplicación.

El inicio, apagado y reinicio de la base de datos se llevan a cabo con scripts preparados para ello que se encuentran ya subidos a la instancia cloud. Una vez despliegue el servidor wildfly ya habrá terminado la subida de código a la nube.

```
sshpass -p $password ssh root@host "$script_reiniciar_servidor & sleep 45;
```

```
$wildfly/jboss-cli.sh --connect --controller=$host:9990 --user=wildfly \  
--password=$wildflypwd --command="deploy --force $WAR"
```

5.5 Despliegue de versiones en la nube

Hemos creado un entorno de desarrollo unificado porque el proyecto avanza en varias líneas de trabajo, distintos desarrolladores trabajan en sus propias ramas dedicadas a un módulo en concreto para no entorpecer el trabajo de los demás. A la hora de trabajar en la nube se nos presenta un problema, ¿qué versión se sube al entorno? ¿qué pasa si queremos tener desplegado en la nube varias versiones para que los tutores revisen los avances de cada desarrollador?

La solución a este problema nos la dio, una vez más, Apache Maven. Cuando empaquetamos toda la aplicación web en el fichero WAR, existe la posibilidad de personalizar el nombre del fichero generado, pudiendo así subir a la nube diferentes despliegues, cada uno con el nombre de la rama del desarrollador como prefijo. Para fijar el nombre con el que se genera el WAR hay que incluir la etiqueta *build* dentro de nuestro fichero de configuración de Maven, *pom.xml*.

```
<build>  
  <finalName>gamificacion-tfg_JoseManuel_Martinez</finalName>  
</build>
```

El segundo problema que se presenta es la base de datos, pues en diferentes versiones su arquitectura puede sufrir cambios. Para evitar posibles conflictos, se creará una base de datos cuyo nombre irá precedido del nombre de la rama del desarrollador. Es importante que cada desarrollador cambie en su rama las referencias a la base de datos que, principalmente, se encuentran en el fichero de configuración de JPA: *persistence.xml*


```
<properties>  
  <property name="javax.persistence.jdbc.url" value="URL-tfg_nombre_rama" />  
  .  
  .  
</properties>
```

Con todo esto, tenemos un entorno en la nube configurado para trabajar con varias ramas de desarrollo y un procedimiento de subida de código a la nube totalmente automatizado. Para acceder a cada versión del proyecto en la nube tan solo habrá que incluir el nombre de la rama de desarrollo después del dominio.



82.223.18.147/gamificacion-tfg_JoseManuel_Martinez/login.xhtml

Ilustración 35: Acceso a la aplicación, rama José Manuel



82.223.18.147/gamificacion/

Ilustración 36: Acceso a la aplicación, rama master



82.223.18.147/gamificacion-tfg_AlvaroSantiago_Cuenca/

Ilustración 37: Acceso a la aplicación, rama Álvaro

5.6 Portal de administración Wildfly

Antes de automatizar las subidas a la nube, las realizaba a través de la web de administración de Wildfly. Este portal ofrece numerosas herramientas muy interesantes para el futuro del proyecto por lo que detallaré en este apartado cómo acceder y qué nos ofrece Wildfly.

La URL es la misma que utilizamos para acceder a los despliegues, pero debemos acceder por el puerto **9990**. Para entrar necesitaremos un usuario que podemos generar con un script que nos proporciona el propio servidor en su directorio **bin** llamado **add-user** con una versión para ejecutarlo en Windows (.bat) y otra en formato **.sh** para ejecutarlo desde la terminal.

```
What type of user do you wish to add?
a) Management User (mgmt-users.properties)
b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : dit
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
- The password should be different from the username
- The password should not be one of the following restricted values {root, admin, administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric
```

Ilustración 38: Administración Wildfly, creación de usuario admin

Dentro de la web encontraremos varios apartados:

- Deployments: permite gestionar las aplicaciones desplegadas. Con el script de subidas no necesitamos administrar nada en este apartado.
- Runtime: permite obtener métricas del servidor, su estado y variables de entorno.
- Patching: permite instalar parches o actualizaciones del servidor.
- Configuration: configuración general del servidor.
- Access control: gestión de accesos y usuarios en el servidor, permite definir grupos y roles.

6 PLAN DE PRUEBAS

En este apartado se detallarán las pruebas que se han realizado para el proyecto, indicando los requisitos de cada una y su resultado. Llevaremos a cabo principalmente dos tipos de pruebas: unitarias y validación. Con el siguiente listado de pruebas se recogerán las principales funcionalidades de la aplicación, así como las nuevas mejoras implementadas tras el trabajo realizado en este TFG.

6.1 Pruebas unitarias

Nombre	PU-1
Objetivo	Creación de plantilla de partida con fichero de preguntas.
Requisitos	Disponer de un usuario en la aplicación y un fichero de preguntas.
Procedimiento	Desde el menú de administración, el usuario selecciona la opción ' <i>Nueva plantilla</i> ' y rellena el formulario. En uno de los pasos deberá subir desde su ordenador un fichero .txt con la batería de preguntas que lanzará el juego.
Resultado	La aplicación crea una nueva plantilla de partidas configurada por el profesor.
Comentarios	En un futuro, estos ficheros vendrán dados por uno de los módulos del proyecto.

Tabla 1: PU-1

Nombre	PU-2
Objetivo	Borrar todas las plantillas de partidas
Requisitos	Disponer de un usuario en la aplicación.
Procedimiento	El profesor pulsa el botón ' <i>Eliminar todas las plantillas</i> ' y confirma el borrado.
Resultado	Todas las plantillas de partidas generadas por el profesor se borran y se actualiza la lista.
Comentarios	

Tabla 2: PU-2

Nombre	PU-3
Objetivo	Borrar todas las partidas en curso.
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	El profesor pulsa el botón ' <i>Eliminar todas las partidas</i> ' y confirma el borrado.
Resultado	Todas las partidas en curso son eliminadas de la base de datos.
Comentarios	Si la interfaz no se recarga, el profesor tiene a su disposición un botón para recargar la información que lee de la base de datos.

Tabla 3: PU-3

Nombre	PU-4
Objetivo	Creación de una plantilla de partida.
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	Una vez el usuario ha iniciado sesión, seleccionará la opción ' <i>Nueva plantilla</i> ' y rellenará el formulario de configuración de la plantilla que va a crear.
Resultado	El usuario crea una plantilla de partida para un juego, configurada con los parámetros que ha decidido el profesor y el fichero de preguntas seleccionado.
Comentarios	Dependiendo del juego, el profesor tendrá a su disposición unos parámetros de configuración distintos debido al funcionamiento de cada juego.

Tabla 4: PU-4

Nombre	PU-5
Objetivo	Crear una partida para el futuro
Requisitos	Tener una cuenta de usuario en la aplicación y una plantilla de partida creada.
Procedimiento	Una vez el usuario ha iniciado sesión, seleccionará la opción ' <i>Iniciar partida</i> ' en su plantilla de juego e introducirá el nombre de la partida en el menú de configuración final. Seleccionará la opción ' <i>Guardar partida</i> '.
Resultado	La aplicación crea una partida, pero no la inicia.

Tabla 5: PU-5

Nombre	PU-6
Objetivo	Cambio de contraseña
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	Una vez el usuario ha iniciado sesión, accederá al menú ' <i>Perfil</i> ', una vez dentro escribirá dos veces la nueva contraseña y seleccionará la opción ' <i>Cambiar la contraseña</i> '.
Resultado	La aplicación cambiará la contraseña del usuario si ambas contraseñas coinciden y desconectará de la web al profesor para que vuelva a iniciar sesión.

Tabla 6: PU-6

Nombre	PU-7
Objetivo	Personalización del código de acceso.
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	Una vez el usuario ha iniciado sesión, accederá al menú ' <i>Parámetros de configuración</i> ' y modificará el parámetro <i>ncodigo</i> .
Resultado	Las nuevas partidas creadas tendrán un código de acceso de tantas cifras como se hayan configurado en el parámetro.
Comentarios	Dicho parámetro tiene un valor mínimo de 5.

Tabla 7: PU-7

Nombre	PU-8
Objetivo	Ver el histórico de partidas jugadas
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	Una vez el usuario ha iniciado sesión, accederá al menú <i>'Historial de partidas'</i> .
Resultado	El usuario tendrá a su disposición un listado de todas las partidas jugadas en la aplicación con información útil sobre las mismas y un sistema de filtrado y paginación.
Comentarios	En la esquina inferior derecha se pueden configurar cuántos elementos por página queremos ver.

Tabla 8: PU-8

Nombre	PU-9
Objetivo	Borrar el histórico de partidas
Requisitos	Tener una cuenta de usuario en la aplicación.
Procedimiento	Una vez el usuario ha iniciado sesión, accederá al menú <i>'Historial de partidas'</i> , seleccionará las partidas que desea borrar, hará click en la opción <i>'Eliminar partidas seleccionadas'</i> y confirmará su acción.
Resultado	Todas las partidas seleccionadas se borrarán de la base de datos y dejarán de estar disponibles.
Comentarios	Existe la opción de darle click a la esquina superior izquierda para seleccionar todas las partidas.

Tabla 9: PU-9

Nombre	PU-10
Objetivo	Comprobar el estado de la conexión con el servidor
Requisitos	Tener conectividad con la dirección del servidor.
Procedimiento	El usuario accederá al portal de jugadores y podrá ver en la parte superior el indicador de conexión.
Resultado	Si el usuario se ha conectado con el servidor, la barra será de color verde. De lo contrario, será de color rojo.

Tabla 10: PU-10

Nombre	PU-11
Objetivo	Acceder a una partida hueco
Requisitos	Un profesor debe haber creado una partida y hay espacio para jugadores.
Procedimiento	El usuario accederá al portal del jugador e introducirá el código de acceso, indicando su nombre, grupo y equipo.
Resultado	Si el grupo está libre, recibirá un mensaje de registro correcto. De lo contrario, pedirá al usuario que seleccione otro grupo.

Tabla 11: PU-11

Nombre	PU-12
Objetivo	Acceder a una partida llena
Requisitos	Un profesor debe haber creado una partida y todos los grupos se han llenado.
Procedimiento	El usuario accederá al portal del jugador e introducirá el código de acceso, indicando su nombre, grupo y equipo.
Resultado	El jugador será notificado y su pantalla se quedará bloqueada hasta que se libere un hueco.
Comentario	Si el jugador recarga la página podrá empezar de nuevo el proceso de registro a una partida.

Tabla 12: PU-12

Nombre	PU-13
Objetivo	Acceder a una partida con uno o más grupos llenos
Requisitos	Un profesor debe haber creado una partida y al menos uno de los grupos se ha llenado.
Procedimiento	El usuario accederá al portal del jugador e introducirá el código de acceso, indicando su nombre, grupo y equipo.
Resultado	La aplicación pedirá al usuario que seleccione otro grupo puesto que el que ha escogido se encuentra lleno.

Tabla 13: PU-13

Nombre	PU-14
Objetivo	Detener el tiempo en una partida
Requisitos	Un profesor debe haber creado y empezado una partida.
Procedimiento	El profesor pulsará el botón de pausar el tiempo en el menú que tiene en la parte izquierda de la pantalla.
Resultado	El cronómetro se detendrá para todos los jugadores.

Tabla 14: PU-14

Nombre	PU-15
Objetivo	Iniciar el tiempo en una partida
Requisitos	Un profesor debe haber creado y empezado una partida. En algún momento de la partida, ha detenido el tiempo.
Procedimiento	El profesor pulsará el botón de reanudar el tiempo en el menú que tiene en la parte izquierda de la pantalla.
Resultado	El cronómetro volverá a funcionar para todos los jugadores.

Tabla 15: PU-15

Nombre	PU-16
Objetivo	Reiniciar el tiempo en una partida
Requisitos	Un profesor debe haber creado y empezado una partida.
Procedimiento	El profesor pulsará el botón de reiniciar el tiempo en el menú que tiene en la parte izquierda de la pantalla.
Resultado	El cronómetro empezará a contar desde su valor inicial.

Tabla 16: PU-16

Nombre	PU-17
Objetivo	Forzar la respuesta de una pregunta
Requisitos	Un profesor debe haber creado y empezado una partida.
Procedimiento	El profesor pulsará el botón de forzar la respuesta.
Resultado	El servidor enviará al juego las respuestas que ha recibido hasta ese momento, no contabilizando las que faltan por llegar.
Comentario	El tiempo se detiene al pulsar el botón.

Tabla 17: PU-17

Nombre	PU-18
Objetivo	Terminar la partida
Requisitos	Un profesor debe haber creado y empezado una partida.
Procedimiento	El profesor pulsará el botón de terminar la partida.
Resultado	La partida terminará, enviando los datos que tiene actualmente a la base de datos y notificando a los usuarios que el profesor ha finalizado la partida.

Tabla 18: PU-18

6.2 Pruebas de integración

Nombre	PI-1
Objetivo	Crear una partida
Requisitos	Tener una cuenta de usuario en la aplicación y una plantilla de partida creada.
Procedimiento	Una vez el usuario ha iniciado sesión, seleccionará la opción ' <i>Iniciar partida</i> ' en su plantilla de juego e introducirá el nombre de la partida en el menú de configuración final. Seleccionará la opción ' <i>Iniciar partida</i> '.
Resultado	El módulo de administración se comunicará con el de multijugador para pasarle toda la configuración que necesita de cara a iniciar la partida. La aplicación abrirá una nueva pestaña con el panel de gestión del profesor para la partida y el registro de jugadores quedará abierto.
Comentarios	La comunicación entre módulos se efectúa a través de la base de datos, donde el módulo de administración guarda la configuración de la partida y el módulo de multijugador la lee para empezar el juego.

Tabla 19: PI-1

7 CASO DE USO REAL

Como ya adelanté en el cuarto apartado, la aplicación web ya es funcional pudiendo configurar, lanzar y jugar partidas para los cinco juegos que disponemos ahora mismo. En este apartado veremos un caso de uso de interés para comprobar que verdaderamente hemos solucionado los problemas que venía acarreado el proyecto. Además, en la web del proyecto se guardarán vídeos explicativos donde se recrean otros escenarios.

7.1 Clase de biología

En un Instituto de Educación Secundaria, un profesor ha terminado de impartir la lección del día. La clase se había centrado en estudiar las células, sus tipos y características. Al profesor le preocupa este tema ya que todos los años los alumnos fallan estas preguntas en el examen y quiere asegurarse de que lo han entendido bien. Para el profesor lo más importante es saber qué partes de la lección no han entendido los alumnos para organizar su siguiente clase haciendo hincapié en esos conceptos.

A la clase le quedan 15 minutos para acabar y pide a sus alumnos que saquen sus ordenadores portátiles para jugar. Dividirá la clase en dos grupos: Telecom e Informáticos e iniciará sesión en el portal de administración del proyecto con su usuario. Una vez dentro, va a lanzar una partida con la plantilla de configuración que creó el día anterior desde su casa para la clase de ese día, donde ha cargado un fichero de preguntas relacionadas con la célula y ha establecido el número de jugadores por grupo, factor de corrección y tiempo de respuestas.

El juego será *Batalla Naval*, pues sabe que su clase es muy competitiva y harán lo que sea por ganarle al equipo contrario.

Configuración Partida ✕

General **Juego** **Configuración Partida** **Grupos y Equipos**

Información general de la partida

Título: *

Etapa:

Curso:

Asignatura:

Tema:

[→ Siguiente](#)

Ilustración 39: Caso de uso, configuración partida

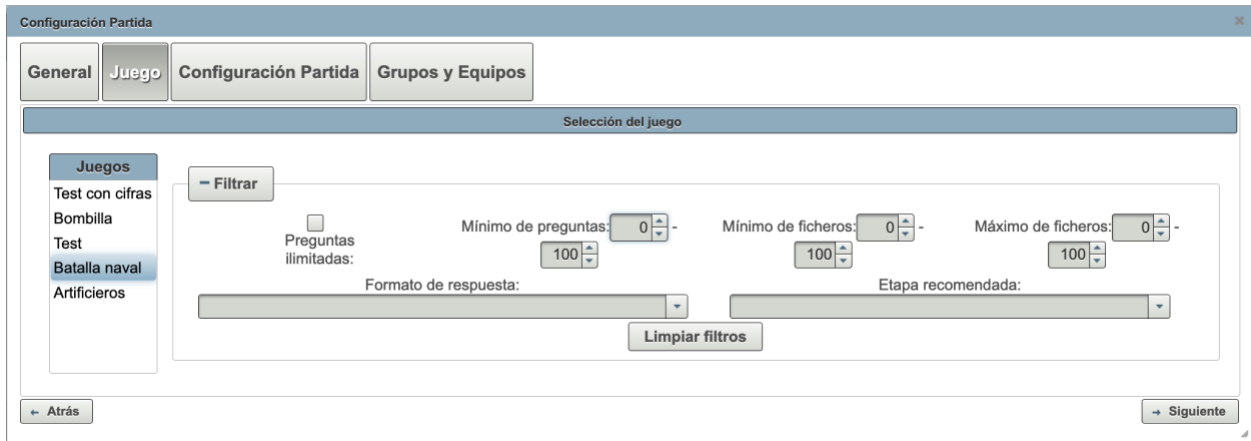


Ilustración 40: Caso de uso, selección de juego



Ilustración 41: Caso de uso, configuración parámetros

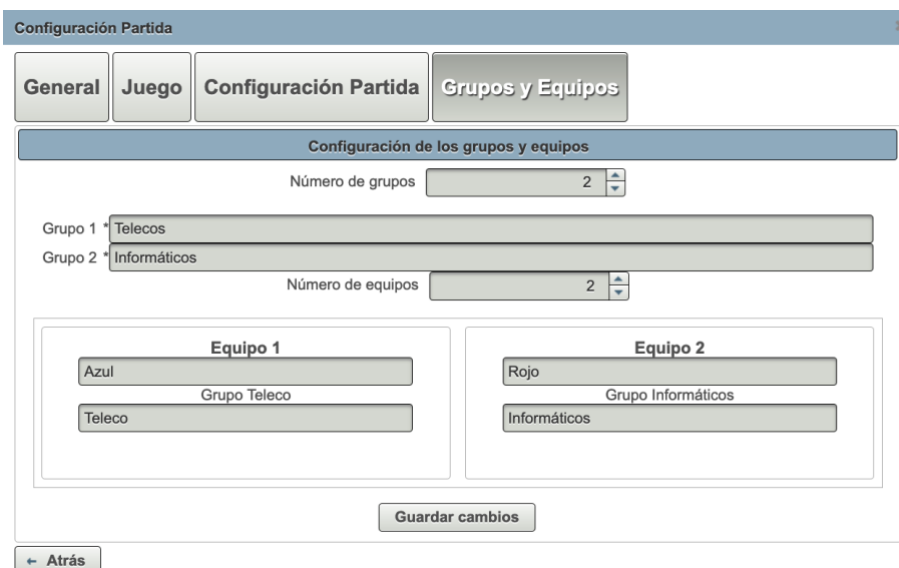


Ilustración 42: Caso de uso, configuración equipos y grupos

El profesor crea la partida y empieza a compartir su pantalla con el resto de la clase, quedando a la vista el código de acceso de la partida **42277**. Como los alumnos ya están divididos en dos grupos, cada uno accede a la partida seleccionando el grupo correcto. Si algún alumno se equivoca, el profesor lo moverá de grupo desde su interfaz de administración.



Ilustración 43: Caso de uso, registro de jugadores

Para simplificar las pruebas, supongamos que en la clase solo hay cuatro alumnos, aunque podrían ser muchos más sin problema. Los jugadores están registrados y el profesor comienza el juego. Empieza respondiendo el equipo Teleco, el cual dispone de 45 segundos para responder a la pregunta tal y como el profesor configuró en la plantilla. Ambos jugadores responden bien la primera pregunta y el buque del equipo Informáticos resulta dañado, parece que el sistema sanguíneo ha quedado claro en clase.

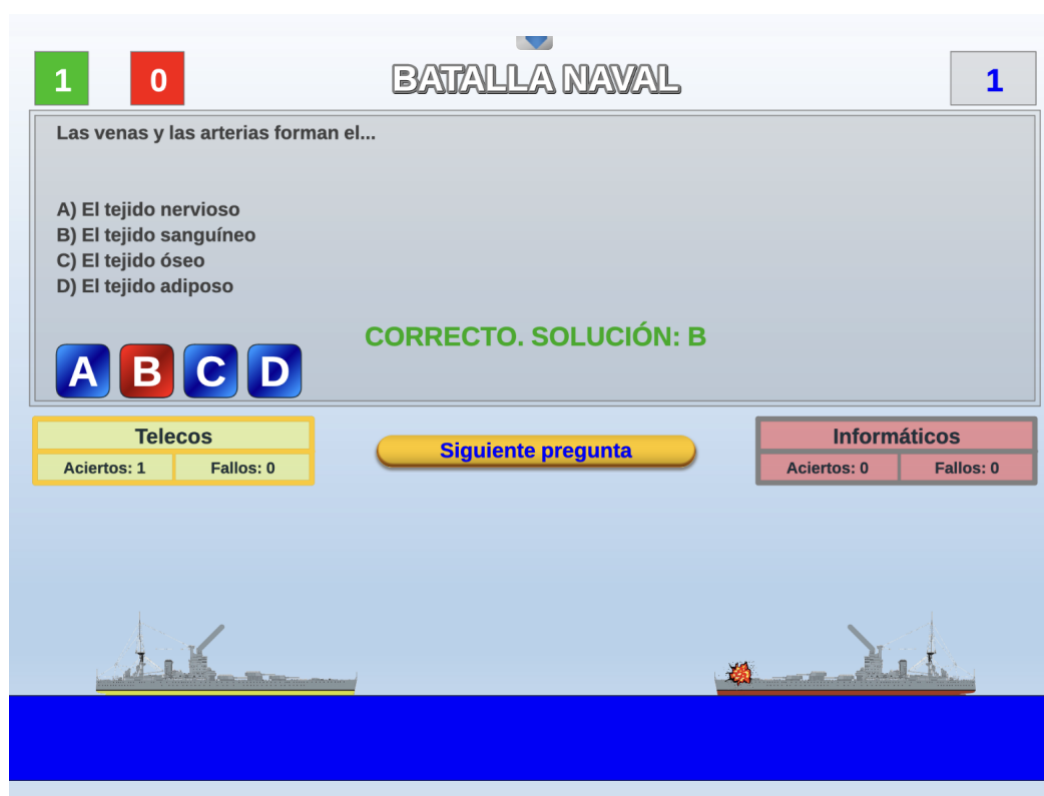


Ilustración 44: Caso de uso, transcurso de la partida

El juego sigue su curso natural y los jugadores siguen respondiendo, algunas bien y otras mal por lo que el profesor va tomando nota de las preguntas que no son capaces de responder pues quiere asegurarse de que queden claros esos conceptos en la siguiente clase. Al rato alguien interrumpe en la clase para hablar con el profesor sobre un tema importante, por lo que el tiempo se pausa hasta que terminan de hablar.



Ilustración 45: Caso de uso, tiempo detenido



Ilustración 46: Caso de uso, jugador respondiendo

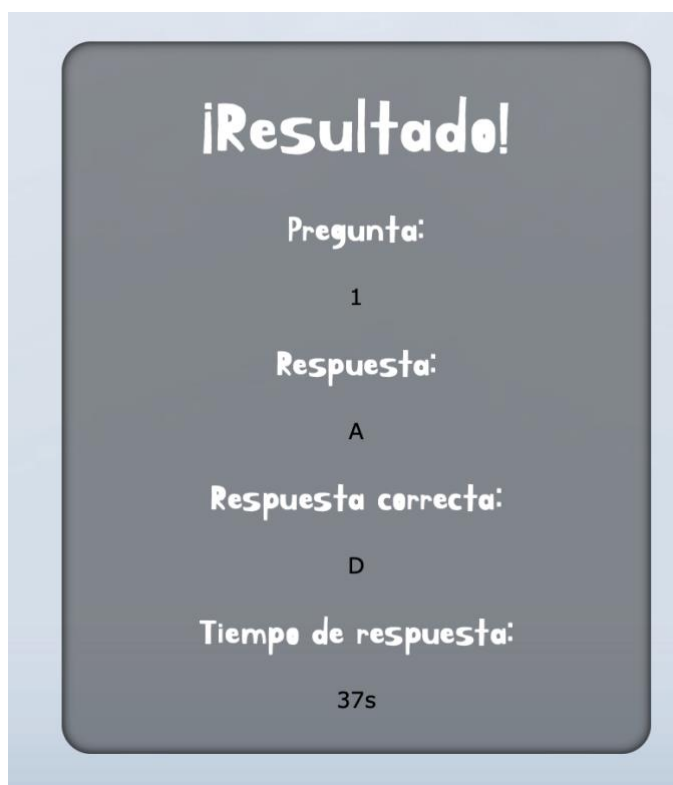


Ilustración 47: Caso de uso, respuesta incorrecta

Al volver a activar el tiempo, el juego continúa. Desafortunadamente, van un poco mal de tiempo y el timbre que indica el cambio de clase es inminente por lo que el profesor ha decidido que va a dar solamente 15 segundos para responder las preguntas, quién no haya respondido en ese tiempo no será contabilizado en las respuestas. Para esto, el profesor dispone de un botón que fuerza el envío de la respuesta cuando quiera, mandando al juego las respuestas recibidas hasta el momento.

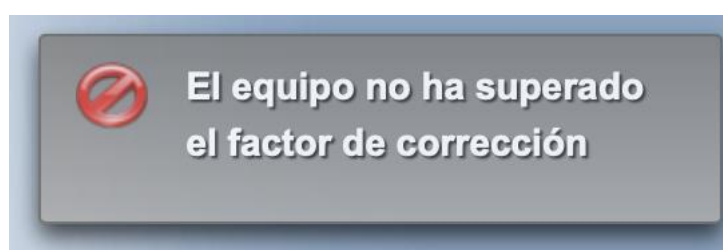


Ilustración 48: Caso de uso, equipo no supera el factor de corrección

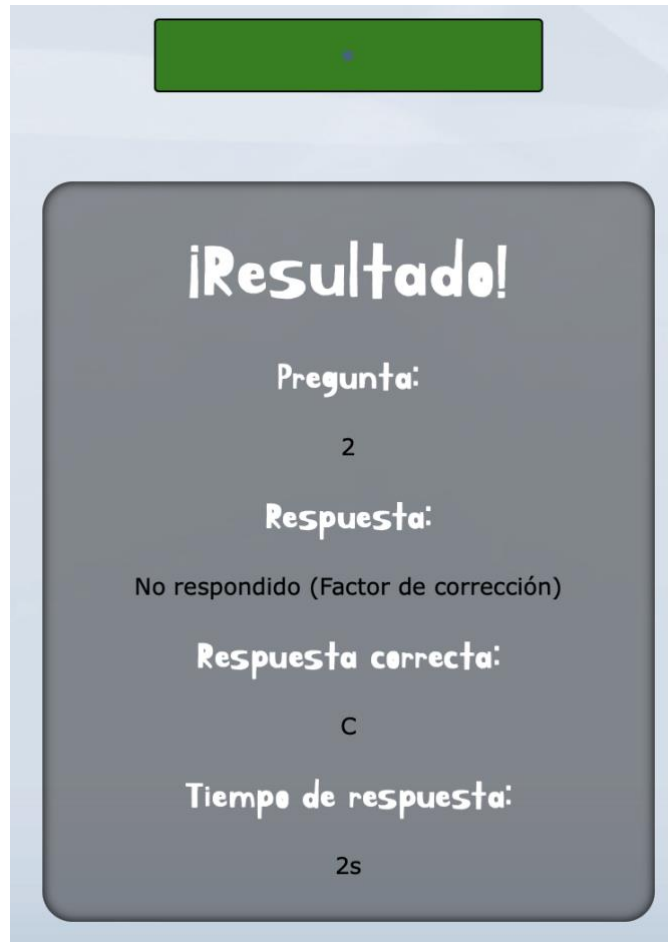


Ilustración 49: Caso de uso, el equipo no supera el factor de corrección 2

Finalmente, el equipo Informáticos acaba ganando la partida y el juego finaliza. El profesor ha tomado notas y en la siguiente clase deberá repasar los conceptos relacionados con la proteína y los aminoácidos.



Ilustración 50: Caso de uso, fin de la partida

8 DOCUMENTACIÓN

Para finalizar con la memoria, es importante hablar de la página web del proyecto [7] [8]. Se trata de un portal donde encontraremos toda la documentación del proyecto: repositorio de código, guías, vídeos y archivos de utilidad. Para acceder a esta web se necesita tener permisos en el Proyecto Gamifica con vuestro usuario de la Universidad de Sevilla.

8.1 Gestión del Proyecto Gamifica

Con el fin de tener un repositorio centralizado de código y la documentación del proyecto lo más detallada posible disponible para que cualquier colaborador, la Universidad de Sevilla ofrece un portal web para la gestión de sus proyectos, incluido el Proyecto Gamifica. Los colaboradores tendrán a su disposición todos los recursos del proyecto una vez tengan permisos para acceder a la web. Dentro, encontraremos lo siguiente:

- Memorias de TFG anteriores: muy útiles para entender el contexto del proyecto y la finalidad de cada módulo. Es importante ser consciente de que algunas cosas pueden estar obsoletas porque se han llevado a cabo mejoras en los módulos con el paso del tiempo.
- Guías: documentación de todo tipo que se han ido referenciando a lo largo de la memoria. Aquí encontraremos información sobre las tecnologías que usamos, cómo está creado el entorno de desarrollo, guías para descargar el repositorio y lanzar por primera vez la aplicación, etc...
- Contactos: principalmente desarrolladores que han participado en el proyecto anteriormente y puedan resolver dudas que surjan en el futuro.
- Vídeos: explicando las funcionalidades de la aplicación, la estructura del repositorio y el contexto general del proyecto.
- Gráficas: esquemas visuales de cómo funciona a bajo nivel la comunicación websocket, los tipos de mensaje que se envían y reciben, así como las combinaciones de parámetros que se pueden utilizar para provocar un comportamiento del juego u otro.
- Scripts: automatizaciones de procesos recurrentes que facilitan el trabajo de los desarrolladores. Aquí tenemos principalmente dos, uno para la subida del código a la nube y otro para la creación de nuevas ramas de trabajo Git.
- Sala virtual de reuniones: donde se realicen charlas para resolver dudas o hacer el seguimiento del proyecto.
- Noticias: apartado que actualmente no está en uso, pero puede ser interesante implementar notas de tipo “*Changelog*” donde se indican los cambios de las últimas actualizaciones o subidas a la nube.
- Actividad: sección que proporciona un historial de acciones realizadas en la web. Aquí quedan registradas las subidas de código al repositorio o los ficheros que se suben al menú de archivos.

8.2 Repositorio

Como he comentado anteriormente, en el portal web tenemos a nuestra disposición un apartado para acceder al repositorio Git a través de una interfaz de usuario que nos permite filtrar por ramas y ver los cambios que se han ido haciendo entre versiones. En comparación con otros gestores de repositorios Git, las funcionalidades que ofrece son escasas, pero su interfaz de usuario nos ofrece una buena visión general del estado del proyecto.

9 CONCLUSIONES Y MEJORAS

Cuando me ofrecieron participar en este proyecto, acepté porque me gustó mucho la idea de llevar la gamificación a las aulas. Como estudiante que ha pasado muchos años en las aulas, creo que este proyecto aportará un gran valor al sistema educativo cuando esté completamente terminado.

Hablamos de una aplicación fácil de usar para profesores y alumnos, con una proyección de futuro impresionante debido a todas las funcionalidades que se pueden llegar a implementar y la amplia variedad de juegos que podrían estar disponibles para los profesores. A título personal, opino que el uso de estos juegos aumenta la motivación y el compromiso de los alumnos gracias a los distintos elementos lúdicos que utilizamos (puntuaciones, niveles, cronómetro), los cuales incentivan la participación de los usuarios y mejoran su rendimiento.

9.1 Conclusiones

Los resultados obtenidos en el caso de uso real documentado anteriormente demuestran el impacto positivo que puede llegar a tener esta aplicación en las aulas, no solo por el hecho de captar la atención de los alumnos al final de la clase (algo que es complicado muchas veces), sino porque el profesor indirectamente recibirá mucha información con los resultados de la partida que puede utilizar para reforzar aquellos conceptos mostrados en clase que no han terminado de interiorizar sus alumnos.

En lo referente a la parte técnica, todas las tareas realizadas han supuesto un verdadero reto. En primer lugar, la falta de documentación fue un gran contratiempo en la integración modular, ya que en las defensas anteriores había muchos aspectos técnicos que no quedaban cubiertos. Esto me obligó a estudiar la arquitectura de cada módulo en profundidad hasta entender cómo estaba montado cada uno. Por otro lado, la creación del entorno de desarrollo me llevó a aprender nuevas tecnologías de virtualización, como Vagrant, y algunas tareas de administración en entornos Linux, como la instalación de un escritorio, la configuración del idioma del teclado y la creación de usuarios y grupos.

Con la elaboración de este TFG, hemos pasado de tener un conjunto de proyectos con funcionalidades distintas a una aplicación web unificada y funcional a pesar de que aún no están completamente comunicados los módulos. Ahora el Proyecto Gamifica cuenta con una extensa documentación donde se explican en detalle todos los componentes de la aplicación con guías y vídeos que permitirán a los desarrolladores descargar el repositorio sabiendo lo que van a encontrar dentro. Sin duda, ha sido un antes y un después en la vida del proyecto. Estoy seguro de que a partir de ahora la aplicación va a avanzar mucho más rápido gracias a las herramientas que tendrán a su disposición los futuros colaboradores.

9.2 Posibles mejoras

Conociendo todo el stack tecnológico del proyecto y prácticamente todas las funcionalidades de la aplicación, me gustaría listar una serie de mejoras para futuros desarrolladores que le darán un gran valor al proyecto. Todo lo que voy a listar a continuación es en base a lo aprendido durante el proyecto y en parte influenciado por mis experiencias en el mundo laboral como ingeniero DevOps, por lo que es posible que haya algunas cosas que escapen al alcance del proyecto.

9.2.1 Migración del repositorio a GitLab/GitHub

El repositorio que nos ofrece la Universidad de Sevilla se queda muy corto en cuanto a funcionalidades comparado con gestores de repositorios gratuitos como GitLab o GitHub y explicaré por qué. El repositorio que tenemos actualmente admite subidas y descargas de código, pero más allá de eso solo ofrece una interfaz web de lectura.

Con el uso de proveedores como GitLab o GitHub se podrían montar pipelines de integración continua (CI). Estos pipelines se componen de tareas o **jobs** que ejecuta el repositorio cuando detecta subidas de código, definidos en ficheros **.yaml** y con una interfaz que te permite hacer el seguimiento de la ejecución de dichos jobs. Creo que el desarrollo del proyecto ganaría una calidad inmensurable con la migración del repositorio a uno de estos proveedores y la elaboración de un pipeline con una estructura que ejecute los siguientes pasos:

1. Análisis estático del código para mejorar la calidad de las implementaciones y buscar posibles vulnerabilidades de seguridad con programas de software libre como **SonarQube**.
2. Empaquetado del código y generación del **artefacto** WAR con Apache Maven.
3. Subida del código a la nube realizando los pasos que ya se hacen en el script de subida.

Todo esto podría estar parametrizado con variables de entorno que te permitan distinguir ramas o versiones de código.

9.2.2 Empaquetado del código

En mi opinión, creo que debemos “*dockerizar*” el despliegue de la aplicación. Actualmente se está utilizando Apache Maven para generar el fichero WAR con todo el código empaquetado, pero creo que sería interesante utilizar Docker para crear un contenedor que despliegue la aplicación, os explicaré algunas de las razones que me han llevado a pensar esto:

1. Con contenedores Docker tienes el control de cuántos recursos vas a consumir en la instancia cloud. Actualmente, si la aplicación empezara a consumir recursos por alguna anomalía (inyección SQL en el código, por ejemplo), el servidor acabaría colapsando. Con un contenedor, estás creando una capa de abstracción que podría evitar este problema.
2. Si trabajamos con contenedores se podría empezar a trabajar con Kubernetes, una de las tecnologías más demandadas para la orquestación de contenedores garantizando así la alta disponibilidad de la aplicación.
3. Portabilidad, da igual qué ordenador estés usando, su sistema operativo, drivers y recursos. Si tienes instalado Docker puedes desplegar la aplicación en segundos.
4. Control de versiones, al tener un repositorio (Dockerhub, por ejemplo) al que subir tu imagen cuentas con un control de versiones y etiquetado que te permite en unos pocos segundos desplegar la versión de hace unos años en un puerto y la versión actual en otro, por ejemplo.

9.2.3 Módulos de administración y comunicaciones

Entrando a valorar los módulos con los que he trabajado, hay una serie de funcionalidades que también le darían un gran valor a la aplicación pero que, por falta de tiempo, no puedo implementar. En lo referente al módulo de administración, destacaría:

1. Subida de juegos en el entorno de producción (nube). Los juegos están subidos desde el ordenador local y posteriormente empaquetados en formato WAR, cuando se suben los juegos desde la versión desplegada en la nube, la aplicación no es capaz de descomprimir el juego en la carpeta destino porque se trata del directorio de la aplicación desplegada (WAR) y no el repositorio del proyecto.
2. Integración con el módulo de estadísticas en el listado de partidas jugadas.
3. Integración con el módulo de ficheros en la creación de plantillas de juegos.
4. Integración con el módulo LTI para la creación de partidas en Blackboard o Moodle.
5. Pagar las listas de partidas en progreso, puesto que al haber demasiadas partidas la ventana crece hacia abajo y rompe la estética, sería necesario implementar el paginado de esa lista.
6. Mejorar el funcionamiento de la recuperación de contraseñas.
7. Añadir más funcionalidades a los usuarios, implementando por ejemplo algún foro de profesores para que funcione como una comunidad que comparte conocimiento.
8. En relación con lo anterior, completar los perfiles de usuario añadiendo la opción de tener una foto y otros atributos personalizables como la materia que imparte o el centro en el que lo hace.
9. Reestructuración de las entidades en la base de datos. Para las partidas se utilizan dos tablas distintas con información común que podría unificarse.

Hablamos principalmente de implementaciones que mejoran la experiencia de usuario al interactuar con la aplicación, siendo la más importante la integración con el módulo de ficheros para evitar errores humanos a la hora de subir los archivos de configuración a la web de administración y poder hacer uso de funcionalidades que nos ofrecen los juegos como son las preguntas aleatorias.

El módulo de comunicaciones es el más completo en mi opinión, pero existen varias líneas de trabajo en las que creo que vale la pena trabajar:

1. Persistir las conexiones websocket, una vez se recarga la página el objeto websocket pierde su asociación con el servidor central y no es capaz de recuperarla siendo imposible continuar partidas.
2. Implementar mejoras en el panel del profesor, obteniendo métricas en tiempo real de los jugadores como el tiempo que ha tardado cada uno en responder o la nota media de cada uno conforme avanza el juego.
3. Guardar las partidas para continuarlas más adelante. Esto va estrechamente relacionado con la primera mejora, le daría un gran valor a la aplicación el hecho de poder guardar una partida para seguir jugándola otro día o en otro momento.
4. Mejoras visuales, por ejemplo, crear una interfaz con forma de reloj para el profesor donde se pueda ver dentro el tiempo que va pasando.
5. Nuevos juegos, esto no depende realmente de los desarrolladores, pero una de las cosas básicas que pueden aportar al proyecto es la adaptación de nuevos juegos para la aplicación.

Todas estas mejoras tratan de satisfacer posibles casos de uso reales que se puedan dar en un aula como el guardado de partidas para continuar otro día. Como comenté anteriormente, la implementación de nuevos juegos es una línea de mejora sin límites que siempre aportará un gran valor a esta aplicación.

9.2.4 Uso de inteligencia artificial

Por increíble que parezca, la inteligencia artificial ha dejado de ser un concepto para pasar a ser una realidad que se encuentra presente en nuestro día a día. A pesar de no ser ningún experto en este campo, la información que obtenemos tras cada partida es muy valiosa y creo que se podría utilizar para algo más que generar métricas y puntuaciones. Toda esa información sería útil para entrenar modelos de inteligencia artificial que permitan identificar patrones de comportamiento en los alumnos. Esto no es imposible de hacer, de hecho, muchas empresas ya están utilizando la información de sus clientes para generar modelos de detección de fraude en el sector de los seguros o previsión de seísmos en las empresas de sismología.

Algunos de los casos de uso en los que había pensado son:

1. Identificar áreas del conocimiento que resultan más complicadas de aprender para los alumnos, categorizado por nivel educativo y localización geográfica.
2. Identificar patrones de respuestas para las preguntas tipo test.
3. Ajuste automático de la dificultad de un juego en función de la habilidad y progresión de los usuarios.
4. Detección y prevención del abandono escolar mediante modelos que estudien las respuestas de los alumnos a formularios de bienestar. Este caso de uso puede estar fuera del contexto del proyecto, pero dichos formularios se podrían integrar sin ningún problema en nuestra aplicación con ficheros de preguntas nuevos.

Lo más importante en este caso es elegir un proveedor que nos permita crear y entrenar modelos de inteligencia artificial. Existen diferentes proveedores, aunque a título personal recomendaría una empresa que proporcione un servicio de Plataforma como Servicio (PaaS) en la nube ya que habrá que almacenar mucha información. En este caso, tenemos disponibles varios proveedores de nube pública:

1. Amazon Web Services (AWS): cuenta con servicios de IA que permiten la creación, entrenamiento y despliegue de modelos de inteligencia artificial como AWS SageMaker o AWS Bedrock.
2. Microsoft Azure: tiene el servicio Azure AI cuyas funcionalidades son similares a las de AWS.
3. Google Cloud Platform: este proveedor cuenta principalmente con dos servicios:
 - a. Vertex AI: similar a AWS SageMaker
 - b. Google Gemini: software que se encuentra aún en desarrollo de tipo “chatbot”, cuenta con una API que te permite entrenar modelos.

10. Referencias

- [1] *Repositorio Vagrant*, creado por el Proyecto Gamifica en 2023: <https://app.vagrantup.com/boxes/search>
- [2] *Guía del Desarrollador 1 - Preparación y Uso del Entorno de Desarrollo*, creado por José Manuel Martínez Delgado en 2023. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [3] *Guía del Desarrollador 2 – Tecnologías y Buenas Prácticas de Desarrollo*, creado por José Manuel Martínez Delgado en 2023. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [4] *Guía del Administrador 1 – Administración del sitio web*, creado por José Manuel Martínez Delgado en 2023. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [5] *Guía del Administrador 2 – Construcción del entorno de desarrollo*, creado por José Manuel Martínez Delgado en 2023. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [6] *Instancia cloud en IONOS*, Creado por José Manuel Martínez Delgado en 2023. Accesible desde: <https://my.ionos.es/product-overview>
- [7] *Web del Proyecto Gamifica*, Creado por Fco. Javier Muñoz Calle y Fco. José Fernández Jiménez. Accesible desde: <https://gestionproyectos.us.es/projects/gamificacion>
- [8] *Vídeo 1: La web del proyecto*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [9] *Vídeo 2: La aplicación, acceso y preparación de una partida*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [10] *Vídeo 3: El entorno de desarrollo*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [11] *Vídeo 4: El repositorio, estructura y tecnologías*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [12] *Vídeo 5: JSF en el proyecto Gamifica*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [13] *Vídeo 6: Otras tecnologías usadas*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [14] *Vídeo 7: La base de datos*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [15] *Vídeo 8: El módulo de administración*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [16] *Vídeo 9: El módulo de comunicaciones*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>

- [17] *Vídeo 10: El juego test*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [18] *Vídeo 11: El juego cifras*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [19] *Vídeo 12: El juego artificieros*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [20] *Vídeo 13: El juego batalla naval*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [21] *Vídeo 14: El juego bombilla*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [22] *Vídeo 15: El repositorio del proyecto*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [23] *Vídeo 16: Gestión de IPs dinámicas*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [24] *Vídeo 17: El entorno cloud*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [25] *Vídeo 18: El factor de corrección*, creado por José Manuel Martínez Delgado en 2024. Accesible desde la documentación del proyecto en <https://gestionproyectos.us.es>
- [26] *Proyecto AJDA*, Jesús Muñoz Calle en 2007. Accesible desde la página web oficial del proyecto: <https://proyectodescartes.org/newton/juegosdidacticos/index.php>
- [27] *RED Descartes*, 2013. Página web: <https://proyectodescartes.org/descartescms/>
- [28] *Java Enterprise Edition*, Oracle: <https://www.oracle.com/es/java/technologies/java-ee-glance.html>
- [29] *Apache Maven*, Apache Software Foundation: <https://maven.apache.org>
- [30] *Java ManagedBeans*, Oracle: <https://docs.oracle.com/javase/7/tutorial/jsf-develop001.htm>
- [31] *Java Server Faces*, Sun Microsystems: <https://www.oracle.com/java/technologies/javaserverfaces.html>
- [32] *Primefaces*, PrimeTek Informatics: <https://www.primefaces.org>
- [33] *CSS*, Hakum Wium Lie y Bert Bos: <https://developer.mozilla.org/es/docs/Web/CSS>
- [34] *Protocolo WebSocket*, web de interés: <https://www.ibm.com/docs/es/was/9.0.5?topic=applications-websocket>
- [35] *Java Persistence API*, Comunidad de Java: <https://docs.oracle.com/javase/7/tutorial/persistence-intro.htm>
- [36] *SLF4J*, Apache Software Foundation: <https://slf4j.org>
- [37] *Java Resource Bundles*: web de interés: <https://docs.oracle.com/javase/8/docs/api/java/util/ResourceBundle.html>
- [38] *Vagrant*, HashiCorp: <https://www.vagrantup.com>