Universidad de Valladolid
Máster universitario
Ingeniería Informática

Trabajo Fin de Máster

Definición y automatización de pruebas de carga y escalabilidad para una aplicación web colaborativa

Realizado por **Manuel Alda Peñafiel**

✠✠✠

Universidad de Valladolid

15 de septiembre de 2022

Tutor: Yania Crespo González-Carvajal

Universidad de Valladolid

Máster universitario en Ingeniería Informática

D. Yania Crespo González-Carvajal, profesora del departamento de Departamento de Informática, área de Lenguaje y Sistemas Informáticos.

Expone:

Que el alumno D. Manuel Alda Peñafiel, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado "Definición y automatización de pruebas de carga y escalabilidad para una aplicación web colaborativa".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Valladolid, 15 de septiembre de 2022

Vº. Bº. del Tutor:

D. Yania Crespo González-Carvajal

Agradecimientos

A mi familia y amigos, por estar siempre a mi lado en los buenos y en los malos momentos.

A mis compañeros del máster no presencial, en especial a Darío y Jesús. Gracias por ayudarme a sacar lo mejor de mi mismo todos los días.

A mi tutora Yania, por la gran dedicación que ha mostrado durante todo el proyecto y por toda la ayuda y consejo que he recibido en este tiempo.

Al grupo de Crossroads, en especial a David, María y David Crespo. Sin vuestra colaboración y apoyo no habría sido posible terminar este proyecto.

Al GIR GEEDS de la UVa, en especial a Luis Miguel y José María, por la confianza depositada en mi para continuar con el desarrollo de Crossroads 2.0 y por la disponibilidad mostrada a la hora de probar la aplicación y realizar sugerencias de mejora.

A la Fundación Española de Ciencia y Tecnología (FECYT) y al proyecto código 16138 titulado ENCRUCIJADA-MUNDO: ECOHERRAMIENTAS LÚDICAS PARA LA TRANSICIÓN ENERGÉICA gracias al cual he podido disfrutar de un contrato a tiempo parcial durante los meses de diciembre hasta junio. Gracias también al proyecto RethinkingAction H2020 cod. 101037104, por la financiación del contrato durante el mes de julio. Gracias por la gran oportunidad de poder formarme dentro del ámbito de la investigación en la universidad.

Resumen

El cambio climático y el desarrollo sostenible del planeta son dos de los principales retos a los que se enfrenta la humanidad actualmente. Como respuesta a estos retos surge Crossroads 2.0, una aplicación web educativa y gamificada que busca concienciar a los ciudadanos acerca del impacto de las decisiones políticas en el medio ambiente y la sostenibilidad del planeta.

El propósito de este trabajo de fin de máster es desarrollar una segunda versión de esta aplicación, a partir de las sugerencias de mejora obtenidas a través de las pruebas de usabilidad realizadas con la primera versión del software, para después poder evaluar el rendimiento y la escalabilidad de la aplicación.

Como resultado se ha obtenido una nueva versión de Crossroads 2.0, la cual está disponible para su uso a través de Internet y se ha elaborado una propuesta de mejora del rendimiento que se deberá abordar en sucesivos trabajos para poder aumentar las prestaciones del software desarrollado.

Descriptores

Crossroads 2.0, evaluación del rendimiento, aplicación web educativa

Abstract

Climate change and the sustainable development of the planet are two of the main challenges facing humanity today. In response to these challenges, Crossroads 2.0 arises, an educational and gamified web application that seeks to make citizens aware of the impact of political decisions on the environment and the sustainability of the planet.

The purpose of this master's thesis is to develop a second version of this application, based on the improvement suggestions obtained through the usability tests carried out with the first version of the software, in order to later be able to evaluate the performance and scalability of the application.

As a result, a new version of Crossroads 2.0 has been obtained, which is available for use through the Internet, and a performance improvement proposal has been prepared that must be addressed in successive works in order to increase the performance of the developed software.

Keywords

Crossroads 2.0, performance testing, educational web application

Índice general

Índice de figuras

Índice de tablas

1: Introducción

1.1. Contexto

En la actualidad el cambio climático es uno de los grandes retos que afectan a nuestra sociedad. Como respuesta a este reto, han surgido numerosas propuestas que buscan concienciar a la población sobre los impactos de nuestras acciones en el medio ambiente para tratar de revertir la tendencia actual del cambio climático.

Una de estas propuestas es **Crossroads 2.0**, que es una aplicación web educativa gamificada cuyo fin es mostrar el impacto que tienen las decisiones políticas en el medio ambiente y el desarrollo sostenible del planeta. Esta propuesta se encuentra dentro del marco del proyecto europeo LOCOMOTION [20], proyecto que busca diseñar un conjunto de IAM (Modelos de evaluación integrados) que proporcionen una manera de evaluar la viabilidad, efectividad, costos y ramificaciones de diferentes opciones de políticas de sostenibilidad. La aplicación web nace como una adaptación de una actividad educativa definida en el grupo GEEDS llamada Crossroads [12], que se apoya en el IAM desarrollado en un proyecto predecesor de LOCOMOTION: el proyecto MEDEAS [23].

Una vez concluido el desarrollo de una primera versión funcional de esta aplicación [2], el proyecto se encuentra en una nueva fase en la que, tras la realización de diferentes pruebas de usabilidad con usuarios potenciales, se está desarrollando una segunda versión funcional de la aplicación, a partir de las sugerencias de mejora realizadas por los participantes en dichas pruebas. Estas pruebas de usabilidad se definieron y llevaron a cabo en otro TFM [18], por lo que el presente proyecto es una continuación de dicho trabajo predecesor. En esta nueva etapa hay otro objetivo principal que se desea alcanzar, que consiste en determinar el rendimiento de la aplicación en condiciones de carga elevada de usuarios, para poder comprobar el grado de cumplimiento de algunos requisitos no funcionales de la aplicación.

Por tanto, el problema que se pretende resolver con este proyecto es doble. Por un lado, el proyecto busca **completar el desarrollo de la segunda versión funcional de Crossroads 2.0**, mientras que por otro se busca **evaluar el rendimiento de la**

aplicación, mediante la realización de pruebas de carga. Para ello, se parte de una versión estable de la aplicación y se espera que, tras la realización del proyecto, se haya obtenido una nueva versión de la aplicación y se hayan extraído conclusiones acerca de la capacidad y la escalabilidad de esta nueva versión de la aplicación.

Este TFM se encuentra bajo el paraguas del proyecto de la Fundación Española para la Ciencia y Tecnología (FECYT) código 16138 titulado ENCRUCIJADA-MUNDO: ECOHERRAMIENTAS LÚDICAS PARA LA TRANSICIÓN ENERGÉTICA.

1.2. Motivación

Uno de los principales aspectos que hay que cuidar durante el desarrollo de cualquier proyecto software es la calidad del producto final. Dicha calidad depende de múltiples factores, de entre los cuales destaca el tiempo de respuesta del sistema desarrollado, ya que cualquier demora por parte del sistema hará que el usuario se canse de su uso y termine por abandonarlo. Este aspecto es más importante si cabe en el caso de las aplicaciones web, pues dada su naturaleza ubicua estas pueden llegar a soportar gran cantidad de peticiones en un espacio corto de tiempo. Por este motivo, es de vital importancia abordar la evaluación de la capacidad y escalabilidad de este tipo de aplicaciones, mediante la realización de pruebas de carga en las cuales la aplicación se somete a estas condiciones de estrés. Por tanto, este proyecto busca evaluar el rendimiento de la aplicación Crossroads 2.0, con el objetivo de poder identificar aquellos aspectos de la aplicación que es necesario modificar para así poder mejorar la experiencia de los usuarios de la misma y así poder extender su uso.

La aplicación deberá permitir un uso en grupos de alumnos de hasta 100 usuarios concurrentes, por ejemplo, mediante la realización de seminarios en la universidad. Además, en estos grupos podrán participar usuarios de habla inglesa, ya que la aplicación deberá estar completamente internacionalizada.

1.3. Objetivos

Los objetivos que se pretende haber alcanzado al término de este trabajo de fin de máster son los siguientes:

1. Desarrollar una nueva versión estable y funcional de la aplicación, a partir de las sugerencias de mejora obtenidas en las pruebas de usabilidad.
2. Realizar un proceso de internacionalización de la aplicación que permita los idiomas inglés y español.
3. Evaluar el rendimiento de la segunda versión de Crossroads 2.0 y extraer conclusiones acerca del grado de cumplimiento de los requisitos de rendimiento enunciados en la sección anterior.

4. Realizar propuestas que permitan incrementar las prestaciones de la aplicación.

1.4. Estructura de la memoria

A continuación, se presentan los capítulos que conforman la estructura de la presente memoria, indicándose para cada uno de ellos su temática principal:

- **Capítulo 2: Acerca de Crossroads 2.0.** Descripción del contexto de Crossroads 2.0, lo que se concreta en los antecedentes y el estado en el que se encontraba la aplicación antes de comenzar este Trabajo de Fin de Máster.
- **Capítulo 3: Desarrollo de la segunda versión de Crossroads 2.0.** En este capítulo se comentan los aspectos relacionados con la planificación, el desarrollo y el seguimiento del miniproyecto asociado al desarrollo de una nueva versión de la aplicación que ya se ha mencionado anteriormente.
- **Capítulo 4: Estado del arte en pruebas de carga.** Presentación del procedimiento de evaluación del estado del arte en el ámbito de las pruebas de carga, así como de los resultados y conclusiones obtenidas tras la realización del mismo.
- **Capítulo 5: Evaluación del rendimiento de Crossroads 2.0.** Descripción del entorno de pruebas, los escenarios de prueba y análisis de los resultados obtenidos tras la ejecución de los mismos.
- **Capítulo 6: Conclusiones y trabajo futuro.** En este capítulo final se indican todas las conclusiones que se han alcanzado en el momento de finalización del proyecto y se plantea el trabajo que deberá ser abordado en el futuro en una posible continuación del proyecto.

2: Acerca de Crossroads 2.0

En este Capítulo se realiza una introducción a los principales aspectos que atañen a Crossroads 2.0. Dichos aspectos incluyen los antecedentes de esta aplicación, en forma de actividad educativa, la dinámica de uso de la aplicación por parte de los distintos tipos de usuarios y los resultados de los procedimientos de prueba realizados sobre la aplicación.

2.1. Antecedentes

Crossroads nace como una actividad educativa en el seno del GIR (Grupo de Investigación Reconocido) GEEDS, de la Universidad de Valladolid [12]. Dicha actividad consiste en un juego cooperativo que se realiza en grupos presenciales, de tal forma que cada grupo tiene que rellenar un formulario en papel, con el objetivo de tomar un conjunto de decisiones políticas que influirán en el desarrollo sostenible y el cambio climático. Una vez rellenado el formulario, cada equipo se lo entrega al moderador que dirige la partida, el cual se encarga de introducir las respuestas del grupo en un simulador desarrollado con Vensim [57]. Tras un tiempo de espera de entre 5 y 10 minutos, el simulador devuelve unas gráficas que representan la evolución de los principales parámetros que determinan el estado del planeta al final del período de simulación como, por ejemplo, la temperatura media global o el PIB per cápita. Con esta versión de la actividad, cada vez que un grupo finaliza su formulario el moderador tiene que introducir las respuestas en el simulador, ejecutar la simulación, esperar los resultados, teniendo como consecuencia largos tiempos de espera. Así la actividad se alarga bastante en el tiempo perdiendo ritmo y dinámica que debe ser suplida por la acción de personas dinamizadoras. Por este motivo, se decidió realizar su informatización dando origen así a Crossroads 2.0 [9].

2.2. Crossroads 2.0: la aplicación web

Tras conocer el origen de la idea de Crossroads 2.0, es momento de presentar la materialización de dicha idea. En esta sección se especificarán algunos aspectos relacionados con la aplicación como su dinámica de uso, las pruebas funcionales que se le han realizado a

lo largo del tiempo y las pruebas de usabilidad, a partir de cuyas conclusiones y resultados da comienzo parte de este TFM.

Dinámica de uso

Para poder explicar adecuadamente la dinámica de uso de la aplicación, es necesario mencionar que existen dos tipos de usuarios, que utilizan la aplicación de forma distinta. Esta es la descripción de los dos tipos de usuarios:

- **Usuario moderador**. Este usuario tiene una cuenta registrada en la aplicación, lo que le permite crear salas de partida y controlar y monitorizar las mismas.
- **Usuario jugador**. Este otro tipo de usuario accede a las salas de partida a través de un código de sala y se junta con otros jugadores para formar grupos. Una vez asignado a un grupo, el jugador deberá ponerse de acuerdo con sus compañeros en las respuestas al formulario, con el objetivo de poder generar unos resultados que indiquen el estado final al que ha llegado el planeta tras aplicar las decisiones políticas seleccionadas.

Teniendo en cuenta esta distinción, la dinámica de usuario de ambos tipos de jugadores se define de la siguiente forma:

- **Moderador**.
  1. El usuario que quiere registrarse como moderador accede, a través de la pantalla principal (Figura 2.1), a la pantalla de registro (Figura 2.2), en la que se incluye un formulario que va solicitando los datos de la cuenta que se va a crear.
  2. Tras introducir todos los datos pedidos y hacer click en el botón de **Crear cuenta**, el sistema inicia una nueva sesión y muestra al nuevo usuario registrado la pantalla de menú (Figura 2.3). Este inicio de sesión se realizará a partir del momento en el que el usuario ya está registrado desde una pantalla específica (Figura 2.4), accesible desde la pantalla principal.
  3. A través de la pantalla de menú, el moderador puede acceder a la pantalla de creación de la sala de partida (Figura 2.5). En esta pantalla, el usuario puede configurar los distintos parámetros de la partida que se va a jugar. Los parámetros que están implementados por el momento son los siguientes: **Nombre de sala**, **Número máximo de rondas**, **Número de grupos** y **Tamaño de cada grupo**. Una vez configurada la sala, se puede finalizar la creación de la misma, a través del botón correspondiente. Esta acción cambiará la pantalla que se muestra al usuario, pasando el moderador a estar en la sala de espera (Figura 2.6). Como su propio nombre indica, en esta pantalla el moderador espera a que todos los jugadores vayan uniéndose a los distintos grupos de la partida, momento en el cual podrá dar comienzo a la misma, a través del botón **Iniciar partida**.

4. Una vez iniciada la partida, el moderador accederá a la sala de monitorización de la partida (Figura 2.7), a través de la cual podrá controlar todo lo que pasa en la partida. Las acciones principales que se pueden realizar son las siguientes: ver el estado de un grupo (Figura 2.8), ver el estado de todos los grupos durante una ronda (Figura 2.9), iniciar una nueva ronda, finalizar la partida e imprimir un informe que resume el estado de cada grupo a lo largo de toda la partida. Este informe no podrá obtenerse hasta que no se haya finalizado la partida.
5. Tras finalizar la partida, el moderador podrá obtener el informe de la misma, para después abandonar la aplicación.

- **Jugador**.

1. El jugador que quiere entrar a una sala de la partida accede, a través de la pantalla principal (Figura 2.1) a la pantalla de acceso a las salas (Figura 2.10). En esta pantalla, el jugador deberá introducir el código de la sala, que previamente le habrá pasado el moderador.
2. Una vez introducido un código de sala, la aplicación cargará un formulario, que el jugador deberá rellenar con sus datos (Figura 2.11). Los datos solicitados son un alias identificativo para el jugador, la edad y el país. Estos dos últimos datos se solicitan con el objetivo de poder elaborar estadísticas de uso en un futuro. Tras completar el formulario, el jugador pasará a ser miembro de la partida y se le mostrará una pantalla con información relevante acerca del juego (Figura 2.12). Una vez examinadas las instrucciones sobre el funcionamiento del juego, el jugador podrá acceder a la sala de espera (Figura 2.13).
3. En la sala de espera, el jugador deberá buscar un grupo en el que haya plazas disponibles, para poder unirse a dicho grupo. Además, el jugador deberá seleccionar un rol de entre cuatro posibles. Estos roles representan a especialistas en diferentes materias (economía, energía, ecología y sociología) y sirven para contextualizar a los jugadores en la temática de la aplicación, pero no tienen influencia alguna en el desarrollo de la partida. Una vez dentro de un grupo, el jugador deberá esperar a que el moderador inicie la partida.
4. Tras el inicio de la partida, el sistema cargará el formulario que deberá ir respondiendo el jugador (Figura 2.14). En este formulario cada pregunta representa una hipótesis, un objetivo o una medida política y las opciones asociadas a cada pregunta representan diferentes posibilidades para definir cada hipótesis, objetivo o medida. Para poder responder a cada pregunta, el jugador deberá introducir un argumento que sustente su elección (Figura 2.15). Para fomentar el debate entre los jugadores, puesto que estos deberán ponerse de acuerdo en sus respuestas, esta pantalla presenta un chat, en el que los jugadores podrán introducir mensajes a favor, en contra o neutros relacionados con una opción de una pregunta en concreto.
5. Una vez respondidas todas las preguntas, el jugador accede a una pantalla en la que puede comprobar el estado de progreso de sus compañeros, así como la

presencia o no de conflictos en las respuestas del grupo (Figura 2.16). En el caso de que exista un conflicto en la respuesta (Figura 2.17), el jugador podrá acceder a una pantalla de resolución de conflictos, haciendo click en el icono correspondiente a la pregunta que se encuentra en la fila inferior de la tabla.

6. En la pantalla de resolución de conflictos (Figura 2.18), el jugador podrá ver el número de mensajes que ha recibido cada opción de respuesta a la pregunta, así como la opción que ha elegido cada jugador. Además, el jugador podrá seguir haciendo uso del chat, con el objetivo de ponerse de acuerdo con sus compañeros a la hora de solucionar el conflicto. Desde esta pantalla, el jugador puede volver a la pantalla anterior en todo momento, haciendo uso del botón correspondiente.
7. Una vez alcanzado el consenso en las respuestas de todos los jugadores del grupo, el sistema mostrará a cada uno de ellos los resultados que han obtenido en la ronda actual (Figura 2.19). En esta pantalla se mostrarán las gráficas de temperatura y PIB que indican el desempeño del grupo durante la ronda. Estas gráficas representan una proyección obtenida con el simulador para estas dos magnitudes hasta el año 2100. Además, y como indicador complementario a las gráficas, se muestra la puntuación que ha obtenido el grupo con respecto a 4 baremos, que miden la bondad del resultado obtenido con las decisiones políticas tomadas. Estos cuatro baremos son los siguientes: **Precisión**, que mide el grado de proximidad de los resultados finales de las gráficas con los objetivos fijados, **Ecológico**, que valora que los objetivos fijados por el grupo sean respetuosos con el medio ambiente, **Equilibrio**, que valora la consecución de un resultado balanceado entre economía y medio ambiente y **Participativo**, que valora la coordinación entre los miembros del equipo para elaborar la propuesta. Otro aspecto importante de esta pantalla reside en una caja de texto, que contiene una pequeña recomendación para que el grupo pueda mejorar su resultado en la siguiente ronda de partida.
8. En este punto, el moderador puede dar comienzo a una nueva ronda, siempre y cuando no se haya iniciado el número máximo de rondas, en cuyo caso, este deberá finalizar la partida.
9. Tras la finalización de la partida, la aplicación mostrará los resultados finales (Figura 2.20), que no son otros que los de la ronda en la que el grupo ha obtenido una mejor puntuación en el baremo de **Equilibrio**. Además, en la parte inferior se muestran varios podios con los tres grupos que han obtenido una mejor puntuación en cada uno de los cuatro baremos. Además, si el jugador hace click en el botón con forma de lupa podrá visualizar las gráficas del grupo que mejores resultados ha obtenido (Figura 2.21), para que así puedan comparar estos resultados con los suyos propios.
10. En este punto, la partida ya ha finalizado, por lo que el jugador podrá abandonar la aplicación.

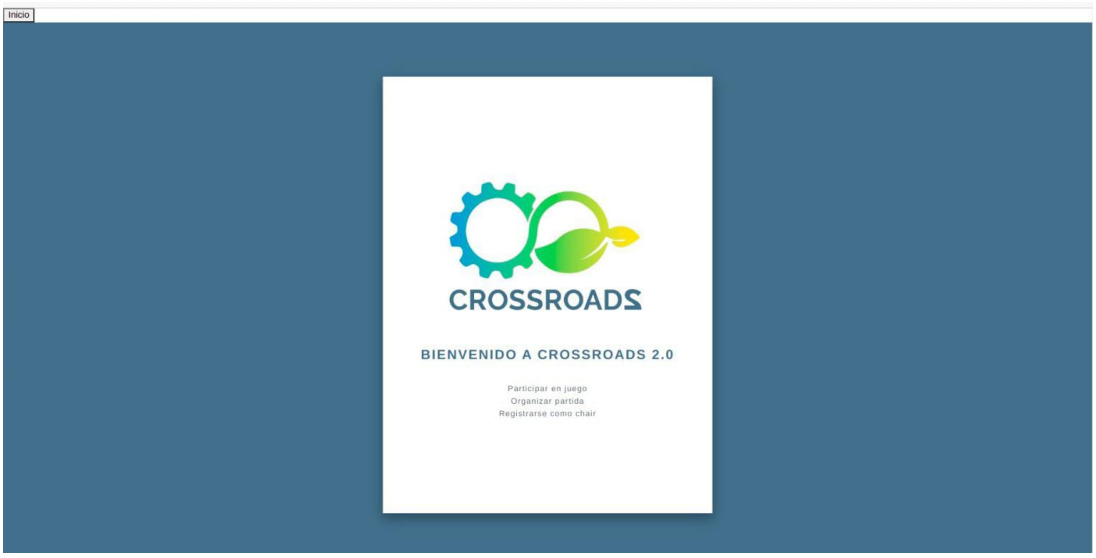

Figura 2.1: Imagen de la pantalla principal de la aplicación

Figura 2.2: Imagen de la pantalla de registro de la aplicación

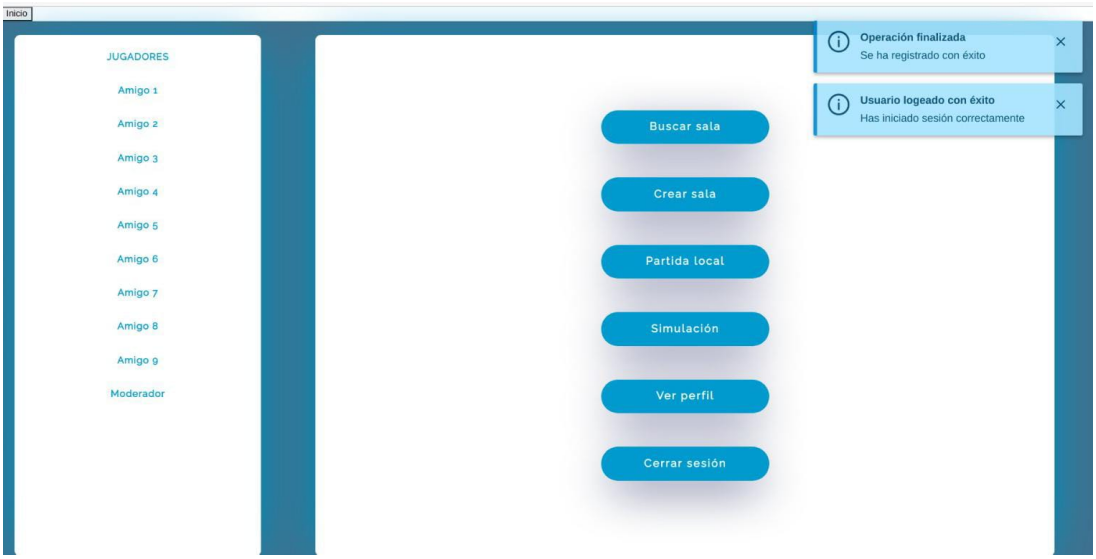

Figura 2.3: Imagen del menú de la aplicación

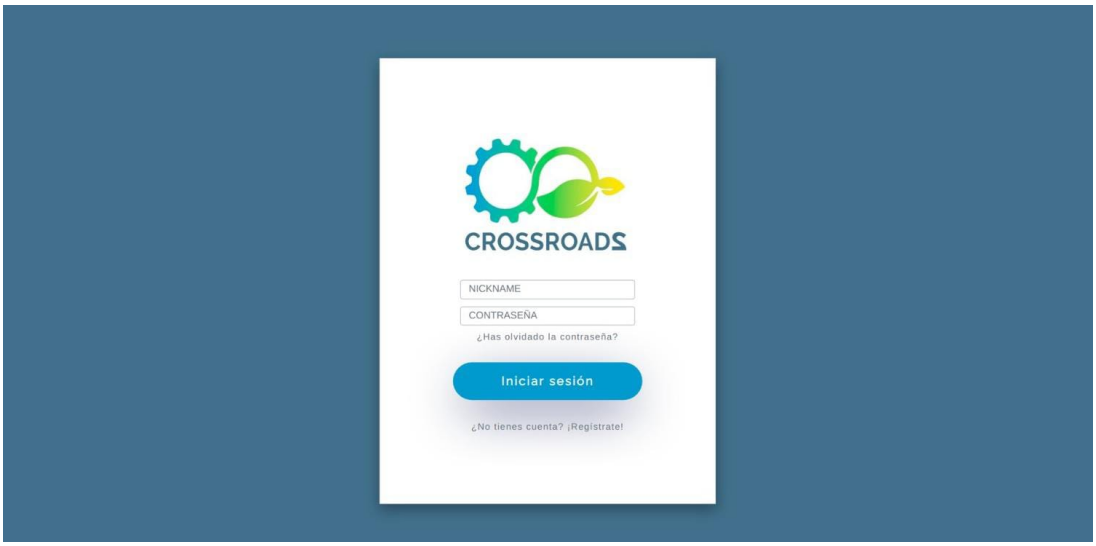

Figura 2.4: Imagen de la pantalla de inicio de sesión de la aplicación

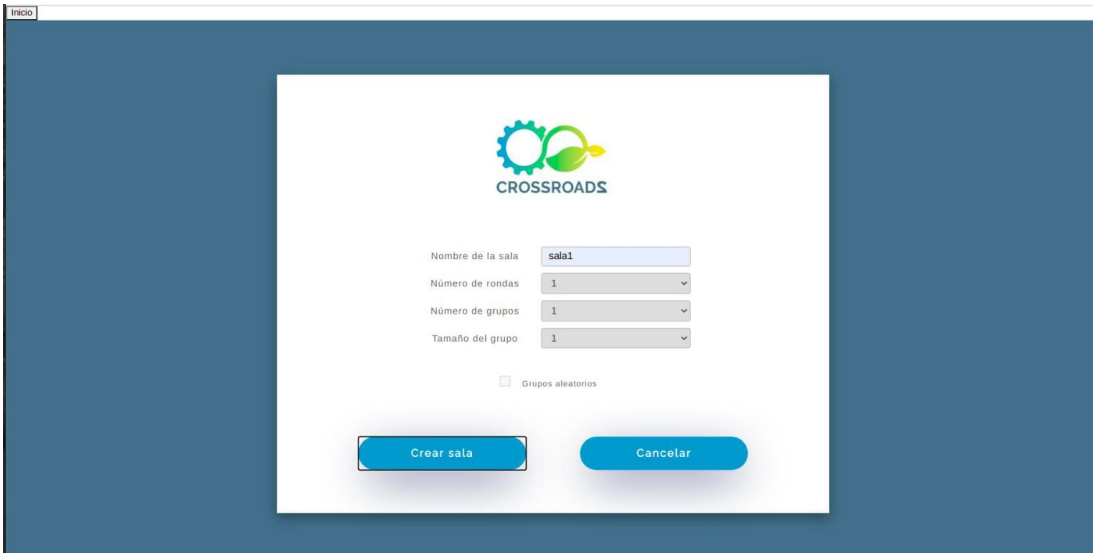

Figura 2.5: Imagen del menú de la pantalla de creación de una sala

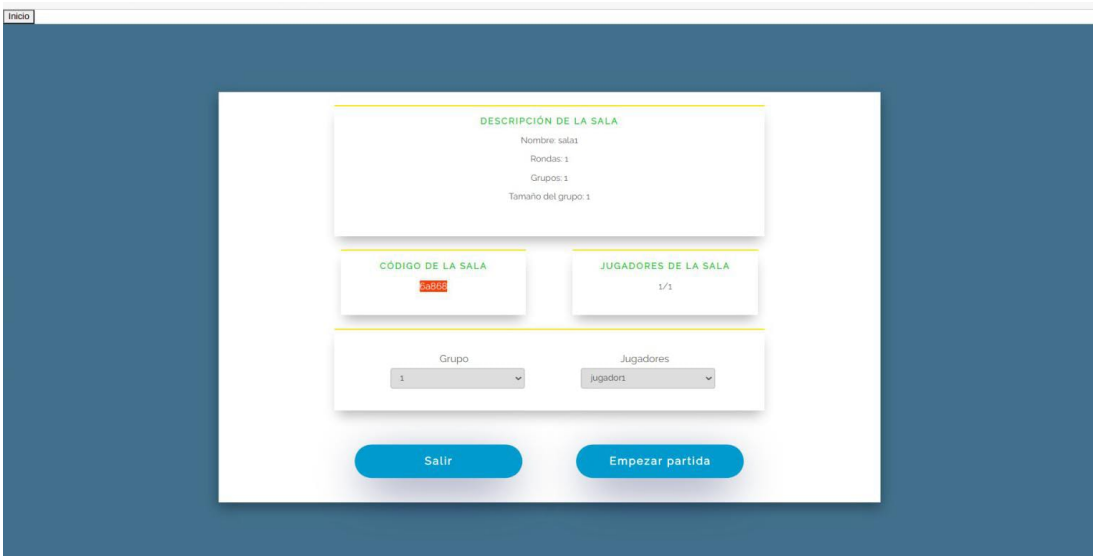

Figura 2.6: Imagen de la sala de espera del moderador

Figura 2.7: Captura de pantalla de la sala de monitorización del moderador

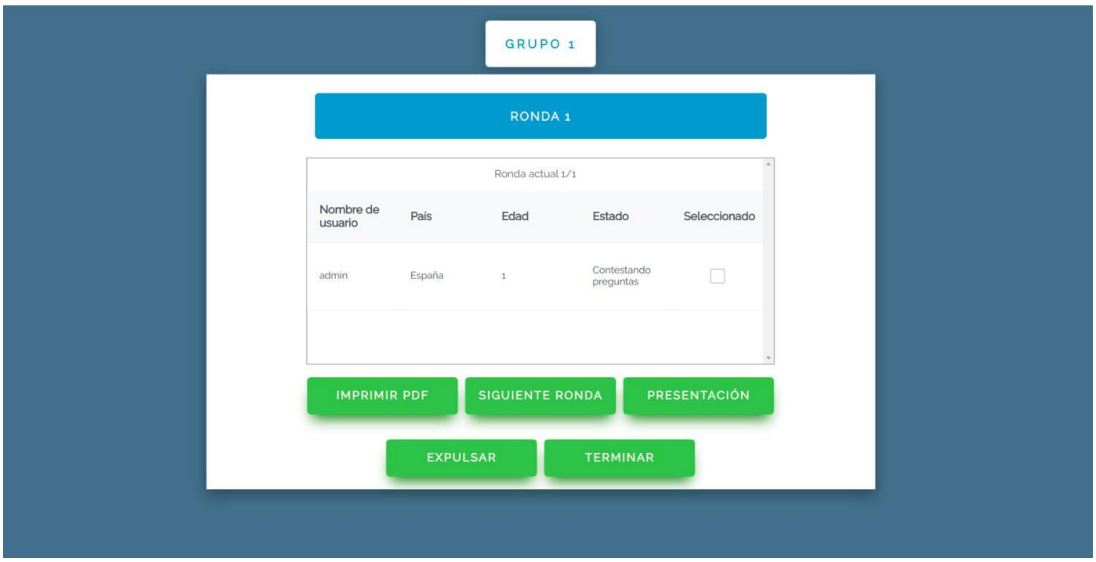

Figura 2.8: Captura de pantalla de la sala de monitorización de la partida cuando el moderador selecciona el grupo 1

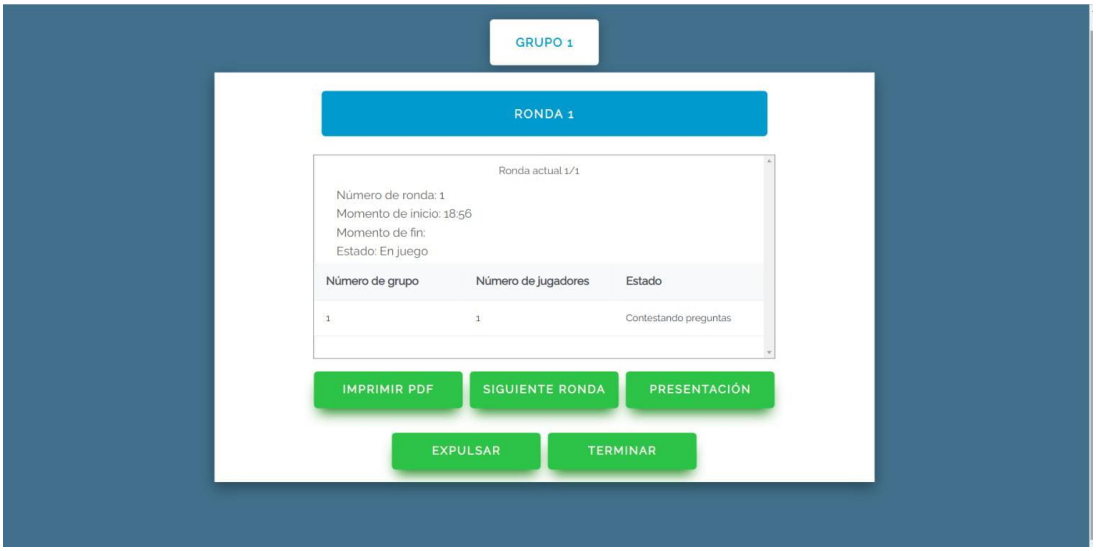

Figura 2.9: Captura de pantalla de la sala de monitorización de la partida cuando el moderador selecciona la ronda 1

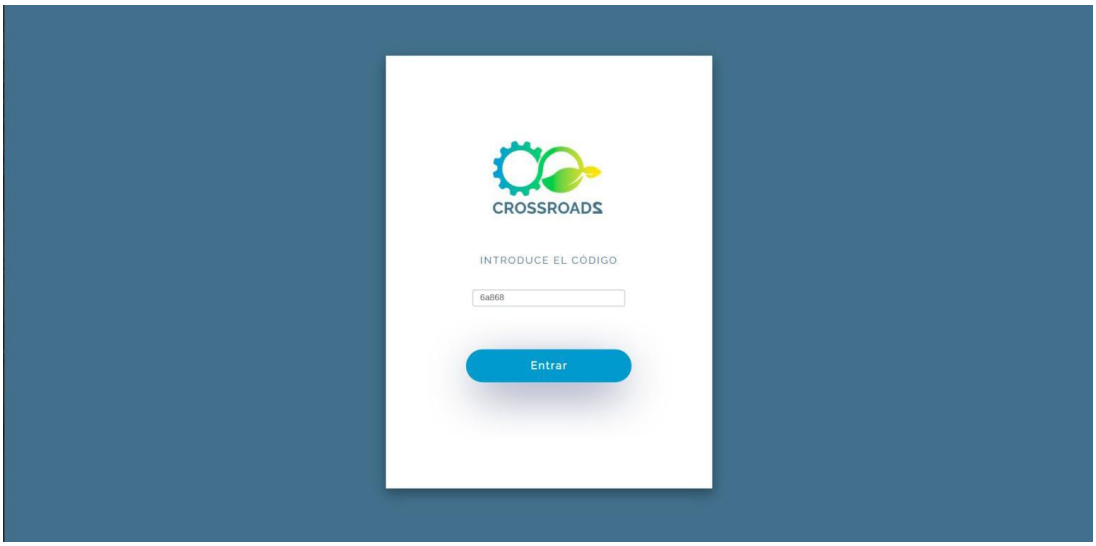

Figura 2.10: Imagen de la pantalla de acceso a las salas de partida para los jugadores

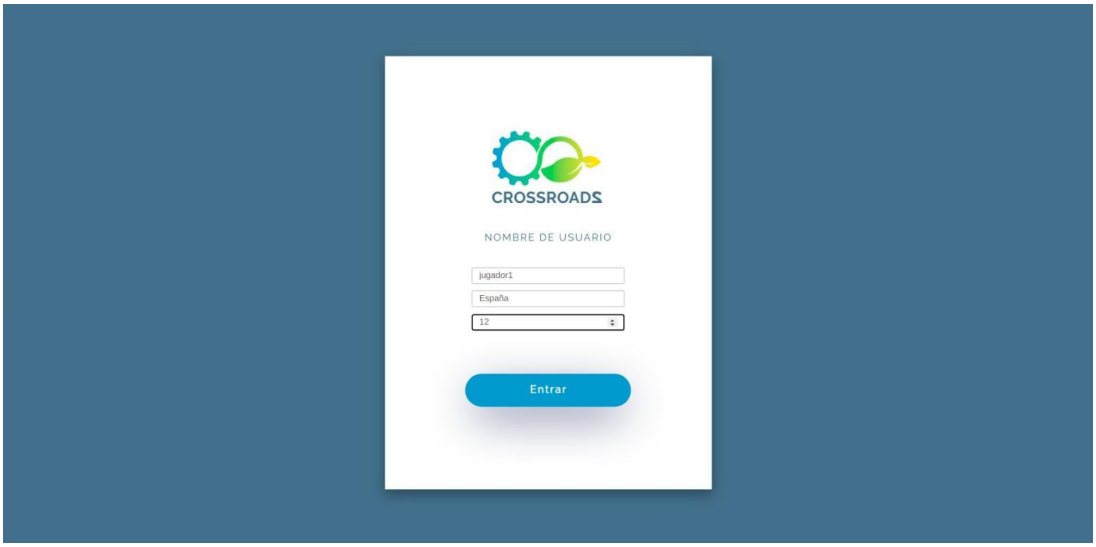

Figura 2.11: Imagen del formulario a través del cual el jugador introduce sus datos

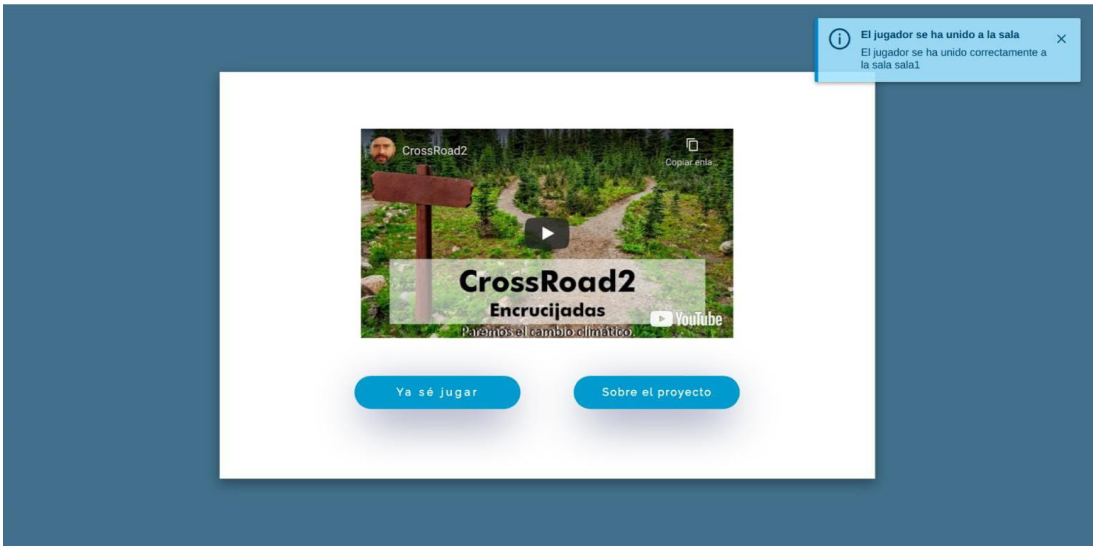

Figura 2.12: Captura de la pantalla de instrucciones sobre el juego

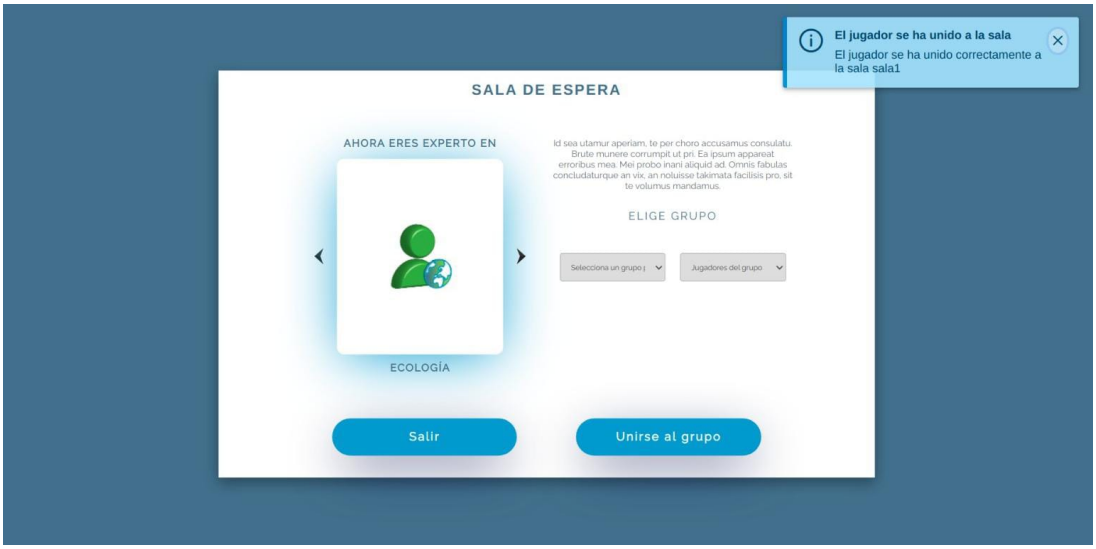

Figura 2.13: Captura de pantalla de la sala de espera para los jugadores

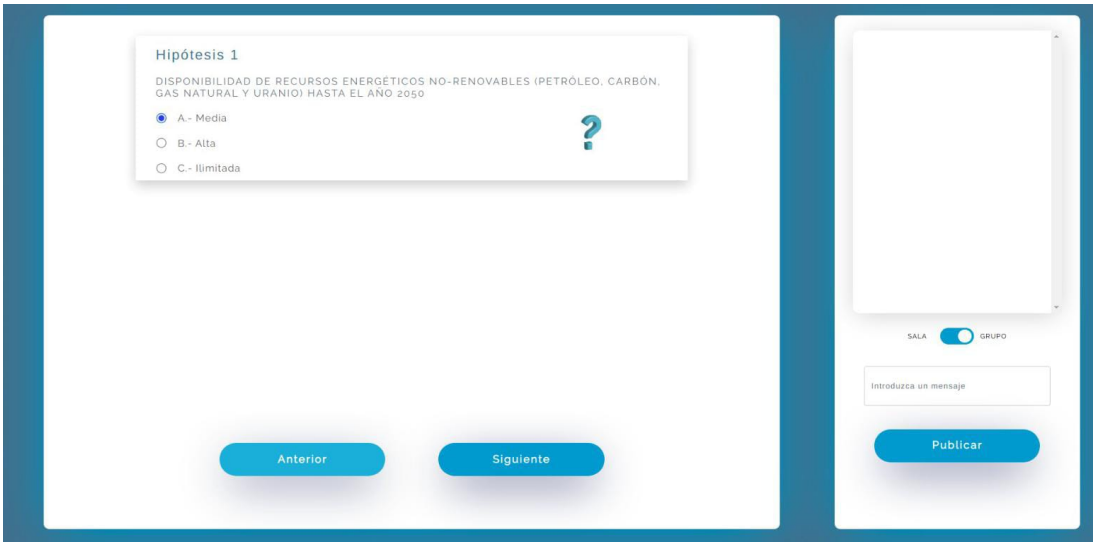

Figura 2.14: Imagen de la pantalla que contiene el formulario de preguntas que cada jugador deberá responder

Figura 2.15: Imagen de la pantalla de preguntas con el pop-up que solicita el argumento

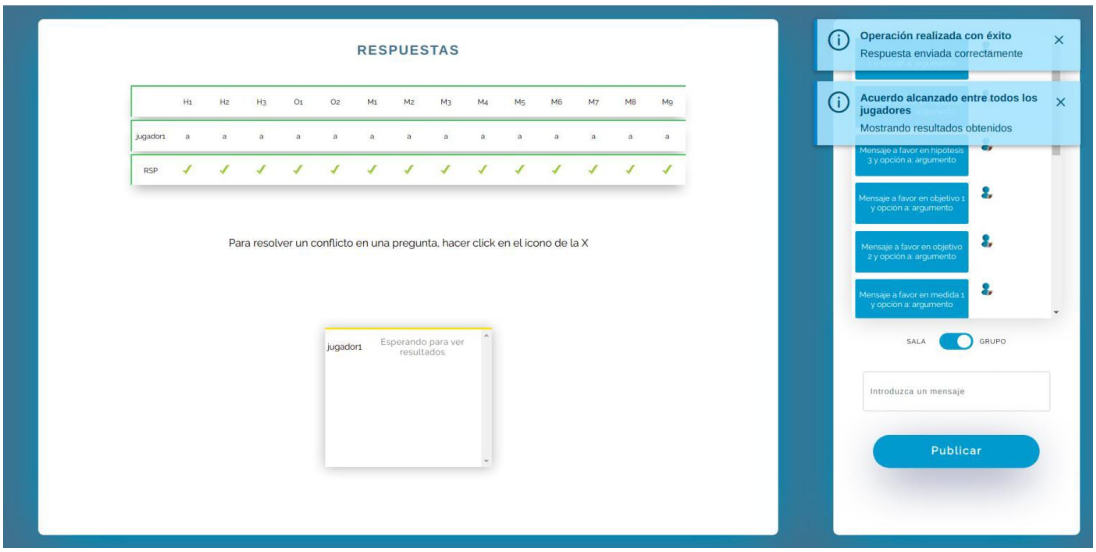

Figura 2.16: Imagen de la pantalla de comprobación de conflictos (sin conflictos)

Figura 2.17: Imagen de la pantalla de comprobación de conflictos (con conflicto en la pregunta 1)

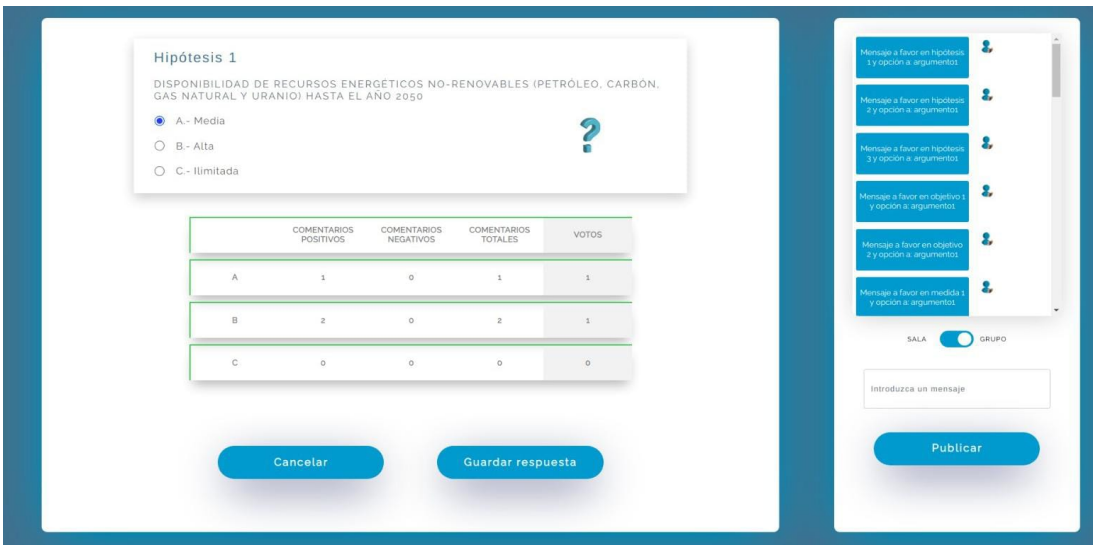

Figura 2.18: Imagen de la pantalla de resolución de conflictos para la pregunta 1

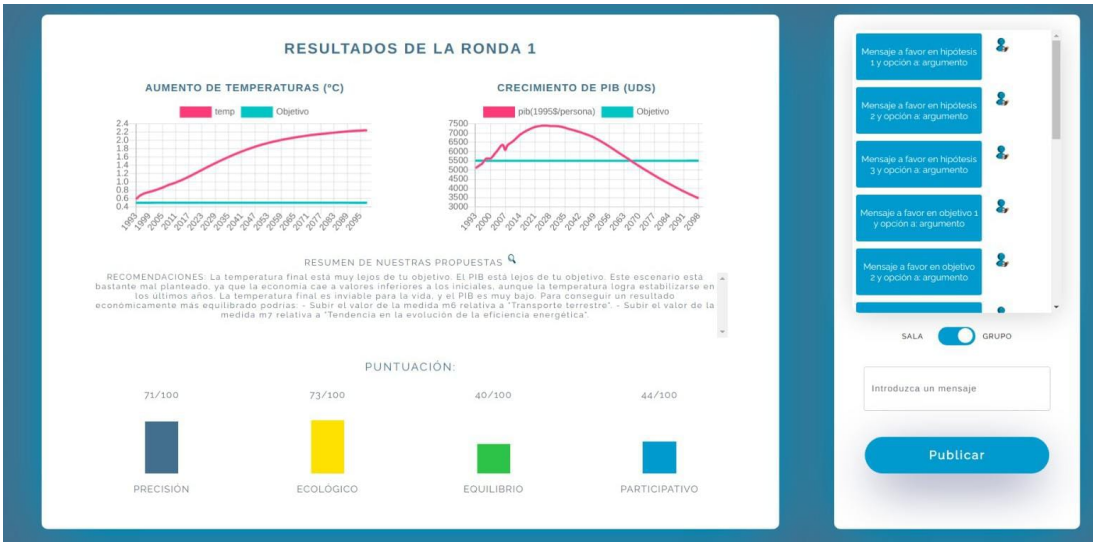

Figura 2.19: Imagen de la pantalla de resultados para la ronda 1

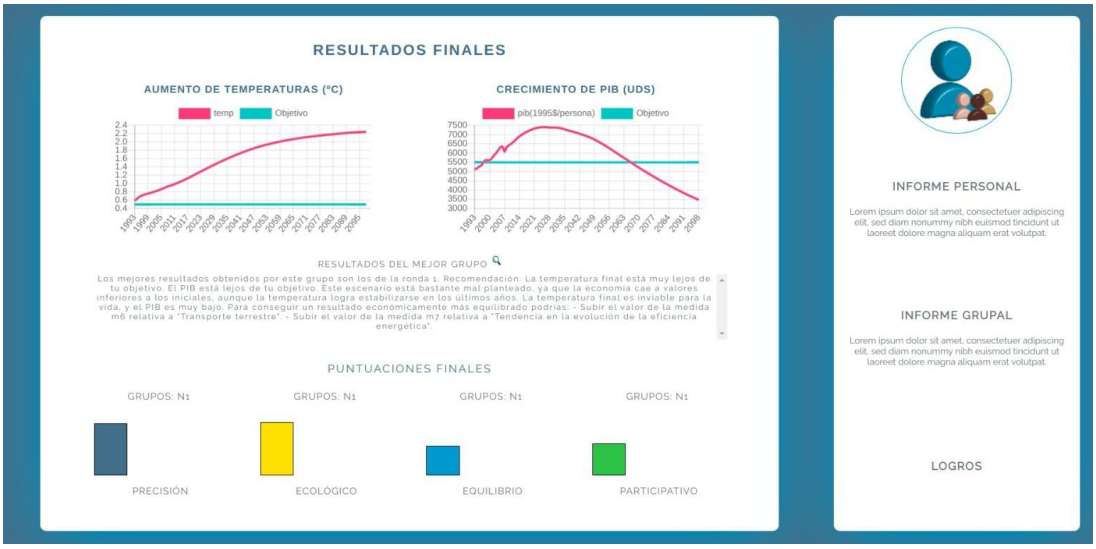

Figura 2.20: Imagen de la pantalla de resultados finales

Figura 2.21: Imagen de la pantalla de resultados finales con el pop-up de las gráficas del mejor grupo

Pruebas funcionales

Para comprobar la corrección de la funcionalidad desarrollada en esta primera versión de la aplicación, se elaboró una batería automatizada de pruebas e2e (*end-to-end*) [53], con el objetivo de detectar posibles errores. Dicha batería de pruebas consta de 135 casos de prueba, agrupados en ficheros JavaScript, donde cada fichero se encarga de almacenar los casos de prueba de cada pantalla o funcionalidad.

El proceso de testing tuvo un resultado positivo, pues permitió sacar a la luz errores en la aplicación. En el momento de inicio del presente proyecto, el porcentaje de éxito de los tests es del 100 %, es decir, todos los tests obtienen el resultado esperado para ellos, mientras que el análisis de la cobertura de los tests (Figura 2.22) indica valores superiores al 95 % en cobertura de sentencia y superiores al 80 % en cobertura de rama (condición-decisión), lo que es un buen indicador de la calidad del procedimiento de testing realizado.

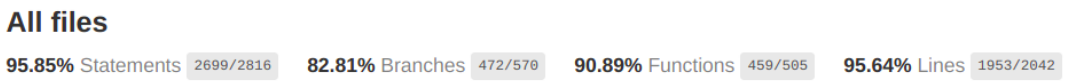

Figura 2.22: Informe de cobertura del código

Pruebas de usabilidad

El diseño y la conceptualización de las diferentes pruebas de usabilidad que se han realizado para Crossroads 2.0 están definidas en otro TFM, desarrollado por la alumna

María Galindo Gómez [18]. Además, en dicho trabajo se presentan los resultados obtenidos de todas esas pruebas y las principales recomendaciones de mejora que es necesario abordar. Dichas recomendaciones (y algunas otras realizadas en pruebas posteriores a la finalización del TFM de María) se recogen en el siguiente listado:

- Permitir la reconexión del jugador en caso de que este cierre el navegador o cambie de página web.
- Mejora de los mensajes proporcionados al usuario cuando comete un error a la hora de registrarse en el sistema.
- Vaciar la caja de texto del chat una vez enviado el mensaje.
- Mostrar los mensajes más recientes del chat con cada actualización del mismo.
- Indicar a los jugadores el estado en el que se encuentra el grupo durante la partida. Este estado debe incluir el número de ronda que se está jugando en un momento determinado y las preguntas que ha ido respondiendo cada miembro del grupo, junto con la respuesta que se ha dado a cada pregunta.
- Mover el envío de los argumentos al chat, en lugar de hacer uso de una caja de texto extra para realizar dicha tarea.
- Modificación de los diferentes podios que se muestran en la pantalla de resultados finales, pues estos son poco intuitivos y a los jugadores les cuesta identificar si su grupo aparece en dichos podios y la posición que ocupa el grupo en los mismos.
- Simplificar los textos de las preguntas y mejorar la explicación del juego, ya que la carga cognitiva necesaria para poder jugar una partida es elevada.
- Incluir una versión de todos los textos en inglés, que permita realizar el cambio de idioma de la aplicación.
- Permitir la navegación por el formulario sin necesidad de introducir una respuesta para cada pregunta.
- Filtrar los mensajes del chat, para mostrar solo los mensajes relativos a la pregunta en la que se encuentra el jugador en cada momento.
- Incluir ayudas a modo de pistas para cada pregunta, de tal forma que el jugador pueda disponer de una guía para responder de manera más adecuada la pregunta correspondiente.
- Incluir un mecanismo que permita expulsar a un jugador cuando el moderador detecte un comportamiento poco adecuado por su parte.
- Incluir explicaciones más exhaustivas de algunos elementos de la aplicación, como pueden ser los distintos baremos que se utilizan para calcular la puntuación del grupo tras finalizar una ronda.

- Modificar el comportamiento de los roles de los jugadores para que estos tengan influencia en el desarrollo de la partida. Esta influencia se verá reflejada en las pistas a las que podrá acceder cada jugador, ya que estas dependerán del rol seleccionado por el mismo.

Todas estas sugerencias se tomarán como punto de partida para el desarrollo de la segunda versión de Crossroads 2.0

3: Desarrollo de la segunda versión de Crossroads 2.0

En este Capítulo se resume aquellos aspectos relacionados con el desarrollo de la segunda versión de la aplicación Crossroads 2.0. Para ello, se presentan los detalles del proyecto que se ha llevado a a cabo dentro de este TFM para completar esta parte del mismo. Los principales temas tratados por el capítulo son aquellos relacionados con la metodología, la planificación, los requisitos y la implementación y modificación de la aplicación.

3.1. Marco de trabajo

Para la gestión del proyecto se ha decidido hacer uso de **Scrum** [59], que es un marco de trabajo que define una forma de llevar a cabo la gestión del proyecto. Se encuentra dentro del marco de los llamados procesos de desarrollo ágiles, cuyo objetivo es reducir la complejidad en el desarrollo de proyectos mediante la mejora de la comunicación entre las partes implicadas en el desarrollo, así como la colaboración de los mismos.

Dadas las características especiales de este proyecto, debido a su naturaleza de TFM y el contexto del mismo, es necesario aplicar un conjunto de adaptaciones para poder aplicar Scrum, las cuales se resumen en el siguiente listado:

- La tutora de este TFM desempeñará los roles de *Scrum Master* y *Product Owner*, este último junto con el resto de miembros del equipo del proyecto FECYT.
- El alumno desempeñará el rol de *Equipo de desarrollo*.
- La duración de los sprints será de una semana. Teniendo en cuenta que la dedicación semanal fijada para el proyecto FECYT es de 16 horas y que en la planificación (Sección 3.3) se ha estimado un número de sprints igual a 30, la duración del proyecto deberá ser de 480 horas.

- En cuanto a los eventos de Scrum, los viernes de cada semana se realizará una reunión, que servirá de *Sprint review* y *Sprint retrospective*. En esta reunión se realizará el *Sprint planning* del siguiente sprint. El *Daily Scrum* se suprimirá debido a las diferencias de horario de todos los participantes en el proyecto FECYT.

3.2. Product Backlog inicial

Partiendo de las sugerencias de mejora realizadas en el TFM de María Galindo [18], cuyo resumen se presenta al final del Capítulo 2 de la presente memoria, se ha elaborado una tabla con las principales historias de usuario que conforman la primera versión del *Product Backlog* del proyecto. La descripción de cada una de las historias se muestra en la Tabla 3.1.

Además, aparte de todas estas historias de usuario, el *Product Backlog* incluye todas aquellas tareas que son necesarias para realizar la corrección de errores, el *refactoring* de la aplicación y la implementación de las mejoras en la usabilidad de la aplicación que se han ido identificando a lo largo de la vida del proyecto. También se incluye una tarea en la que se aborda el cambio de la *skin* de la aplicación por una nueva, cuyo diseño será realizado por la empresa Ondeuev.

Identificador	Descripción
US1	Como jugador quiero poder reconectarme a la sala de partida cuando la abandone accidentalmente
US2	Como jugador quiero poder ver los mensajes más recientes del chat con cada actualización del mismo
US3	Como jugador quiero poder ver las respuestas que están seleccionando mis compañeros de grupo
US4	Como usuario quiero poder utilizar Crossroads 2.0 tanto en inglés como en español
US5	Como jugador quiero poder tener pistas que me ayuden en la selección de la respuesta para cada pregunta
US6	Como moderador quiero poder expulsar a un conjunto de jugadores de la sala cuando su comportamiento no sea adecuado
US7	Como moderador quiero poder crear una partida con grupos que se formen de forma completamente aleatoria

Tabla 3.1: Historias de usuario del *Product Backlog*

3.3. Planificación

Teniendo en cuenta la duración de los sprints fijada en la Sección 3.1 y que las fechas de inicio y de fin del proyecto son, respectivamente, el 3 de enero de 2022 y el 30 de junio de 2022, se ha elaborado la Tabla 3.2, que contiene la planificación temporal de cada sprint. Además, se han planificado 4 sprints de refuerzo en el mes de julio, que se emplearán en caso de necesidad por la aparición de retrasos y otros contratiempos.

ID	Fecha inicio	Fecha fin	Observaciones
S1	03/01/2022	09/01/2022	
S2	10/01/2022	16/01/2022	
S3	17/01/2022	23/01/2022	
S4	24/01/2022	30/01/2022	
S5	31/01/2022	06/02/2022	

S6	07/02/2022	13/02/2022	
S7	14/02/2022	20/02/2022	
S8	21/02/2022	27/02/2022	
S9	28/02/2022	06/03/2022	
S10	07/03/2022	13/03/2022	
S11	14/03/2022	20/03/2022	
S12	21/03/2022	27/03/2022	
S13	28/03/2022	03/04/2022	
S14	04/04/2022	10/04/2022	
S15	11/04/2022	17/04/2022	
S16	18/04/2022	24/04/2022	
S17	25/04/2022	01/05/2022	
S18	02/05/2022	08/05/2022	
S19	09/05/2022	15/05/2022	
S20	16/05/2022	22/05/2022	
S21	23/05/2022	29/05/2022	
S22	30/05/2022	05/06/2022	
S23	06/06/2022	12/06/2022	
S24	13/06/2022	19/06/2022	
S25	20/06/2022	26/06/2022	
S26	27/06/2022	03/07/2022	
S27	04/07/2022	10/07/2022	Sprint de refuerzo
S28	11/07/2022	17/07/2022	Sprint de refuerzo
S29	18/07/2022	24/07/2022	Sprint de refuerzo
S30	25/07/2022	31/07/2022	Sprint de refuerzo

Tabla 3.2: Planificación inicial de los sprints

3.4. Tecnologías utilizadas

Para el desarrollo e implementación de esta segunda versión de Crossroads 2.0 ha sido necesario utilizar un conjunto de tecnologías. Las mismas se muestran a continuación:

- **Node.js y NPM**. Node.js [17] es un entorno de ejecución para JavaScript construido con el motor V8 de Google Chrome. Aunque está pensado para ejecutar código del lado del servidor, también puede ejecutar código de la parte de cliente. Node Package Manager [38] (NPM) es un gestor de paquetes desarrollado en JavaScript que permite añadir nuevos módulos y paquetes a las aplicaciones NodeJS que estemos desarrollando. Estas dos tecnologías se han utilizado para el desarrollo del código del front-end Angular y la gestión de las dependencias del mismo.
- **Angular**. Angular [3] es un framework para crear aplicaciones web y móviles desarrollado en TypeScript y mantenido por Google. Se utiliza para crear aplicaciones

de una sola página (SPA [56]), en la que todas las vistas se encuentran en la misma página y todo el código se carga de una sola vez, o de forma dinámica, cuando las acciones realizadas por el usuario requieran de la carga de ciertos recursos. De esta manera, la aplicación contiene un único fichero HTML llamado index.html en el que se van insertando el código de las vistas de manera estática o dinámica, según lo establezca el desarrollador en el código. El front-end de la aplicación está desarrollado con este framework.

- **Cypress**. Cypress [11] es un framework que permite la realización de tests unitarios y e2e (end-to-end) de forma sencilla, automática y muy amigable para el desarrollador. El testing e2e [53] es un tipo de testing en el cual se emula el comportamiento del usuario a la hora de manejar la aplicación, realizando las acciones que realizaría dicho usuario en la interfaz de la app. El nombre end-to-end proviene del hecho de que, con estos test, no se prueban partes individuales de la aplicación, sino la aplicación entera, tanto el front-end, como el back-end. La batería de pruebas funcionales de la aplicación se ha elaborado haciendo uso de este framework.

- **Spring Boot**. Spring Boot [39] es una versión simplificada del framework Spring, utilizado para crear aplicaciones del lado del servidor en Java, Kotlin y Groovy. La principal ventaja de Spring Boot es que permite crear aplicaciones Spring sin necesidad de realizar todas las configuraciones que habría que realizar si solo se utilizase Spring. El back-end de la aplicación fue desarrollado haciendo uso de Spring Boot con el lenguaje de programación Java.

- **Flask**. Flask [45] es un framework minimalista escrito en Python que permite crear aplicaciones web de manera rápida y sencilla. En este proyecto se ha utilizado para la creación de un servicio externo de recomendación para los jugadores desarrollado por Adrián Manzano, como parte de su Trabajo de Fin de Máster del Máster en Ingeniería Informática de la UVa.

- **MySQL y MongoDB**. MySQL [10] es un gestor de bases de datos relacionales. MongoDB [25] es un gestor de bases de datos NoSQL orientado a documentos, que en lugar de guardar los datos en tablas como en las bases de datos relacionales, esta lo hace en formato BSON, muy similar a JSON. MongoDB se organiza en torno a colecciones. En este proyecto se crean dos bases de datos, una con cada uno de los dos gestores de bases de datos utilizados. La base de datos gestionada con MongoDB se utiliza para guardar los datos de la simulación que conforman los resultados que obtienen los jugadores, mientras que la base de datos gestionada con MySQL guardará el resto de datos relativos al dominio de los usuarios y el juego.

3.5. Análisis

En esta Sección se presentan los cambios que se han realizado en el modelo de dominio de la aplicación. Todos los cambios se muestran con color rojo en el diagrama de la Figura 3.23 y se describen en detalle a continuación:

- Adición de la clase **Pista**, que modela el conjunto de ayudas que puede tener asociadas cada pregunta del formulario.
- Adición de la enumeración **RolJugador**, que modela los distintos roles que puede desempeñar un jugador en su grupo. Las pistas a las que se tendrá acceso dependerán de este rol.
- Adición del atributo **rol** de tipo **RolJugador** a la clase **Jugador**.
- Adición del atributo **nombreCorto** a la clase **Pregunta**. Este atributo modela un resumen del contenido de la pregunta.
- Adición de los atributos booleanos **partidaIndividual** y **gruposAleatorios** a la clase **Sala**. Estos dos atributos modelan dos nuevas características de las salas de partida: el modo individual, de un solo jugador y la partida con grupos aleatorios, en el que cada jugador es asignado a un grupo de manera aleatoria.

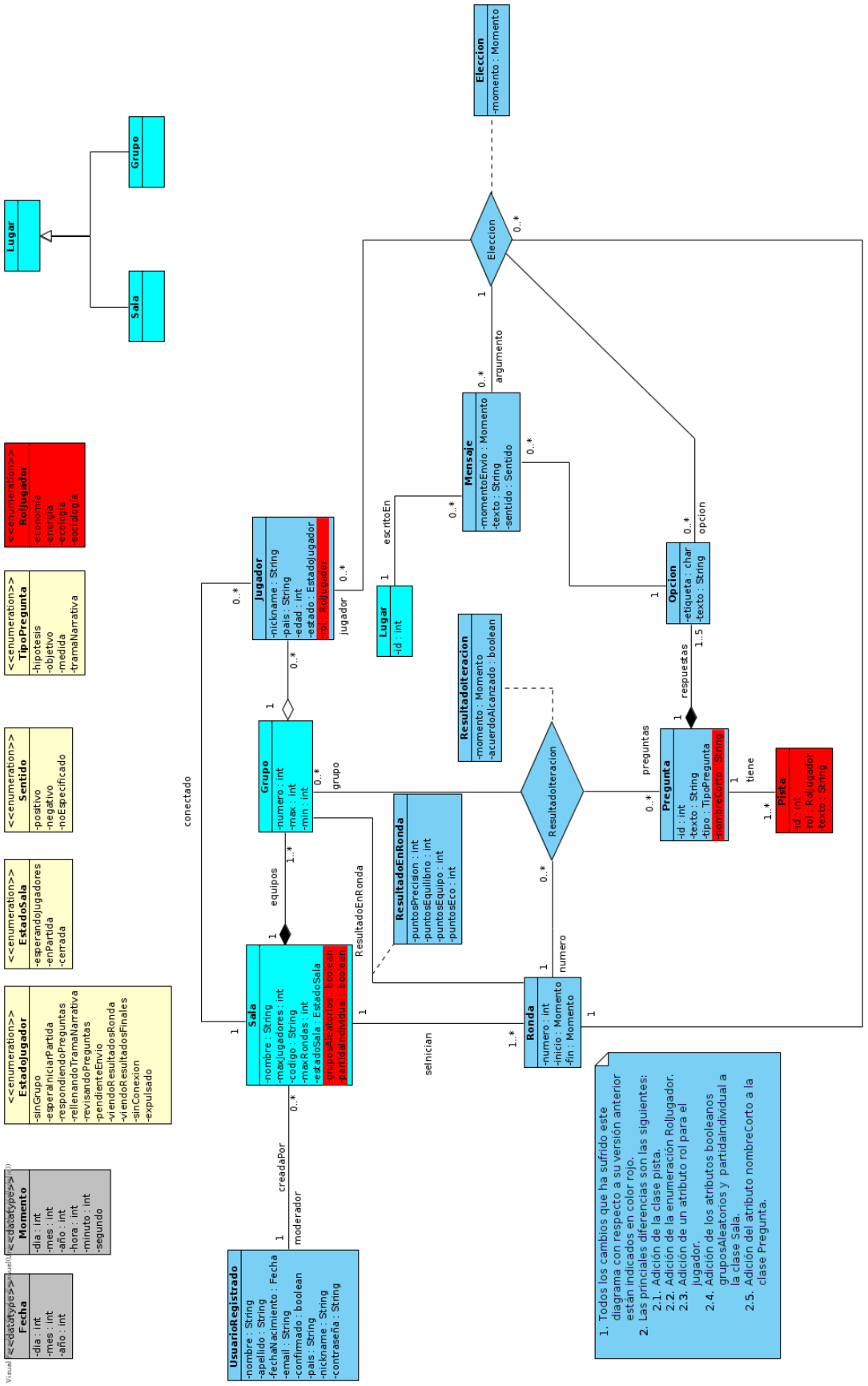

Figura 3.23: Diagrama de clases con las clases del dominio

3.6. Diseño

En esta Sección se presentan todos los cambios aplicados a los diagramas del modelo de diseño de la aplicación. Todos estos cambios se circunscriben a tres diagramas en concreto: el diagrama de componentes del front-end (Figura 3.24), el esquema de la base de datos MySQL (Figura 3.25) y el esquema de la base de datos Mongo (Figura 3.26). A continuación, se describen los cambios en cada diagrama:

- **Diagrama de componentes del front-end**. En este diagrama se ha añadido un nuevo componente, llamado **ConflictMatrix**. Este componente modela una matriz en la que se van mostrando las distintas respuestas que va seleccionando cada jugador del grupo para cada pregunta, así como la existencia o la ausencia de conflicto en las respuestas de cada pregunta. Este componente es utilizado por dos componentes, el componente **Quest**, que se encarga de la gestión del formulario de preguntas que cada jugador debe responder, y el componente **ConflictCheck**, que se encarga de la gestión del estado de la propuesta de respuesta al formulario que está elaborando el grupo. Este componente se ha extraído del componente **ConflictCheck** con el objetivo de reutilizar código, pues se hizo necesario mostrar información de este componente en el componente **Quest**, para que cada jugador tuviese información acerca del estado del formulario de sus compañeros de equipo, recomendado a partir de las pruebas de usabilidad de la primera versión de la aplicación.
- **Esquema de la base de datos MySQL**. Los principales cambios que ha sufrido este diagrama se muestran en el siguiente listado:
  - Adición de la columna **role** a la tabla **player**.
  - Adición de las columnas **single_game** y **random_groups** a la tabla **room**.
  - Adición de las columnas **objective_pib** y **objective_temperature** a la tabla **result_in_round**. Estas columnas almacenan las etiquetas de las respuestas que ha dado el grupo a los dos objetivos del formulario.
  - Adición de la tabla **hint**.
  - Adición de las tablas **question_translation**, **answer_translation** y **hint_translation**. Estas tablas almacenan todos los textos que se encontraban en las tablas **question**, **answer** y **hint** y han surgido como parte de la internacionalización de los textos de la base de datos. De esta forma, cada tabla guarda los textos en el idioma correspondiente y una etiqueta que identifica dicho idioma al que pertenecen los textos de cada tabla.
- **Esquema de la base de datos Mongo**. De este diagrama se han eliminado las colecciones **DescriptionScenario** y **MutualInformation**, pues estas han dejado de ser necesarias por el microservicio de recomendación de la aplicación.

Todos estos cambios están indicados en color rojo en cada diagrama, excepto los de la Figura 3.26).

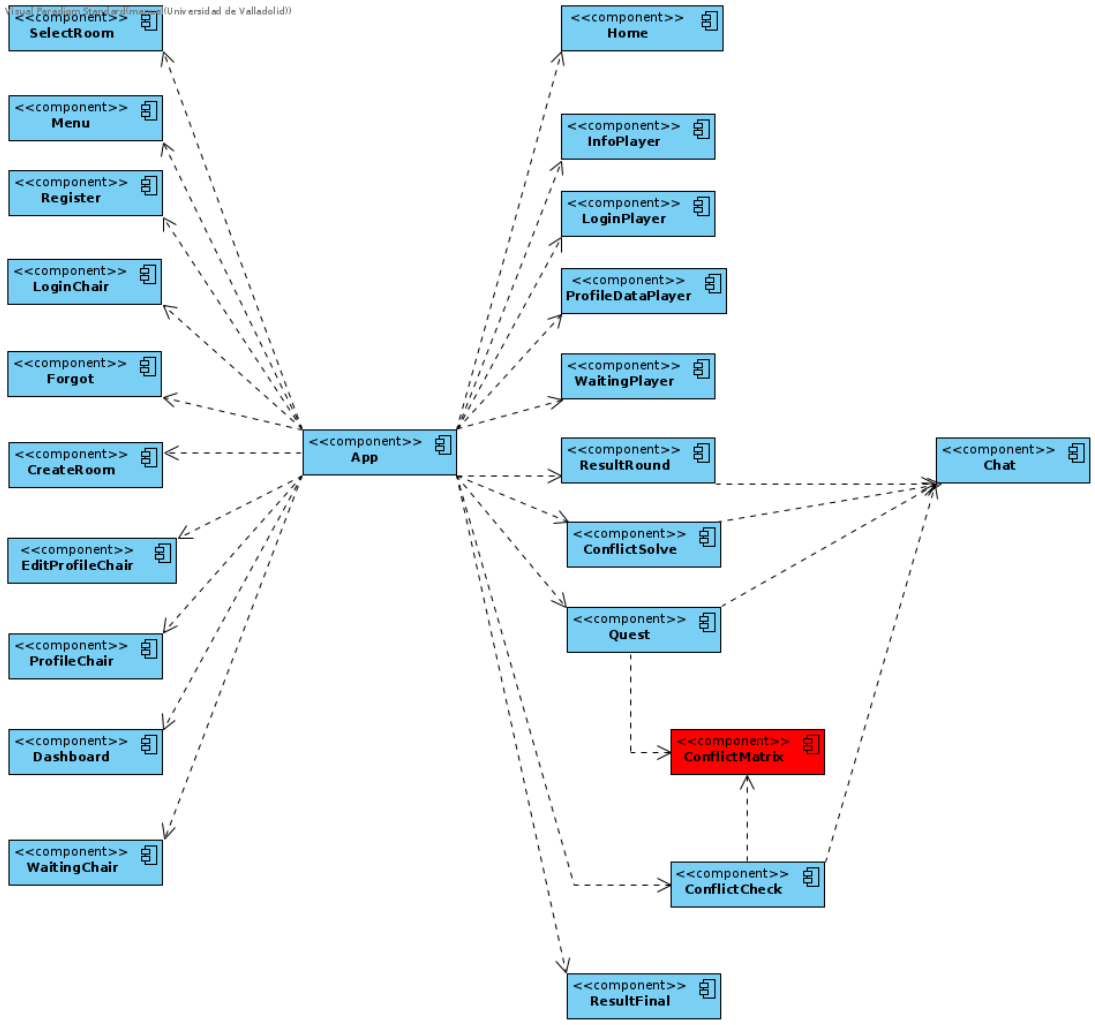

Figura 3.24: Estructura actualizada de componentes del front-end

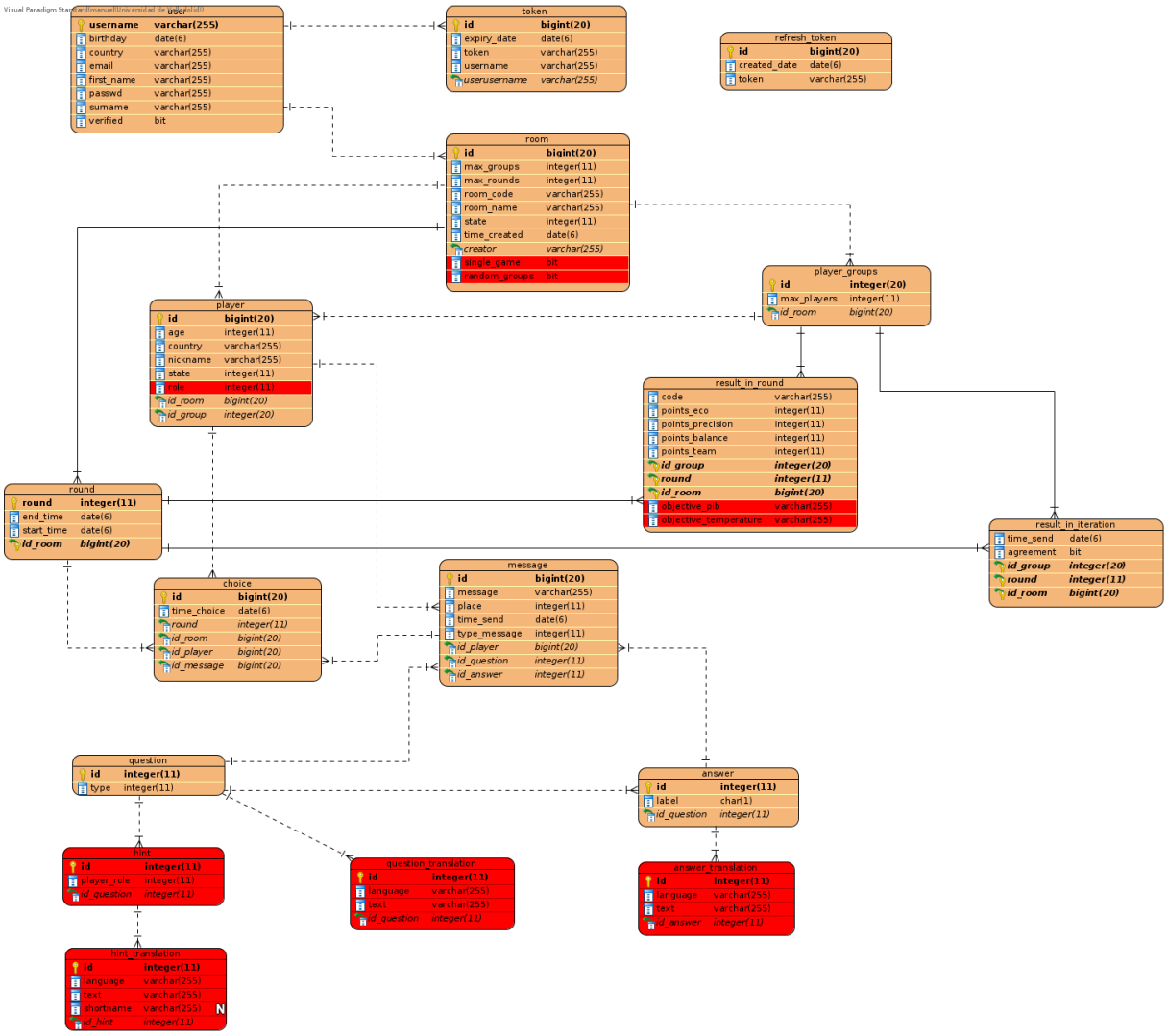

Figura 3.25: Esquema actualizado de la base de datos MySQL

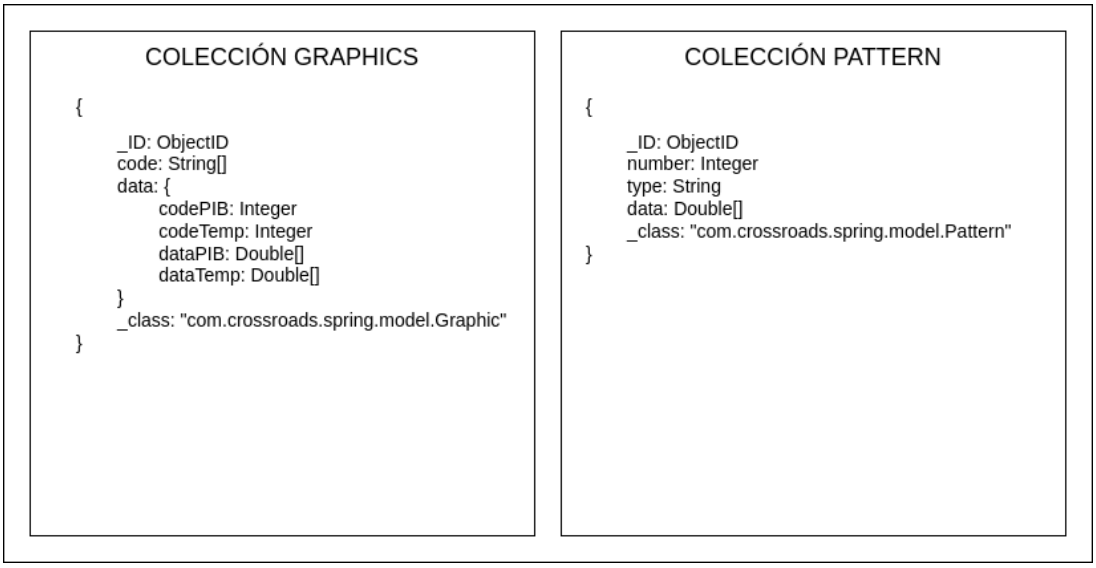

Figura 3.26: Esquema actualizado de la base de datos Mongo

3.7. Implementación

Modificaciones realizadas en el back-end

Para poder desarrollar la nueva versión de Crossroads 2.0 ha sido necesario actualizar el back-end de la aplicación, mediante la modificación de métodos ya existentes y la implementación de nueva funcionalidad. En concreto, se han añadido 7 nuevas operaciones a la API:

- **getHintsByQuestionAndPlayerRole**. Permite recuperar el texto de una ayuda para una pregunta a partir del identificador de la pregunta y del rol que tiene en su grupo el jugador que solicita la ayuda.
- **saveAllChoices**. Permite guardar todas las respuestas al formulario de una única vez. Esta operación solo está disponible cuando la partida es individual.
- **getPlayerChoicesInRound**. Permite recuperar la lista de elecciones que ha realizado un jugador para el formulario en una ronda dada.
- **getVideoHintsLinks**. Permite recuperar la lista de hipervículos a unos vídeos explicativos de cada pregunta. Esta operación solo está disponible cuando la partida es individual.
- **expelPlayers**. Modifica el estado de un conjunto de jugadores de la sala a *Expulsado*, bloqueando de esta manera la participación de estos jugadores en la sala de partida.
- **getPlayerState**. Consulta el estado en el que se encuentra un jugador de la partida en un momento dado de la misma.
- **createSingleGame**. Permite la creación de una sala de partida individual. Dichas salas de partida se caracterizan por disponer de un único grupo, con un único jugador sin rol y un número máximo de rondas de partida igual a 10.

En las Figuras 3.27 a 3.35 se presenta la documentación Swagger de cada una de las nuevas operaciones.

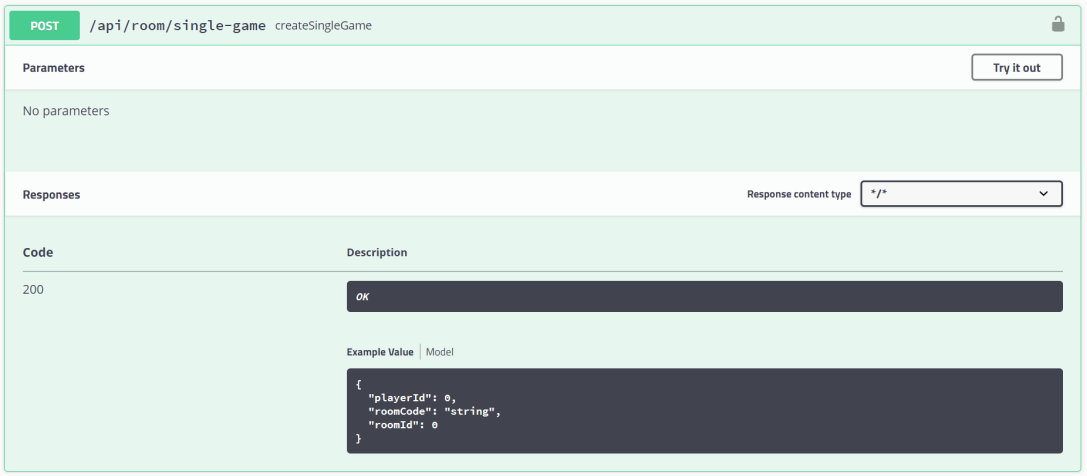

Figura 3.27: Documentación Swagger de la operación **createSingleGame**

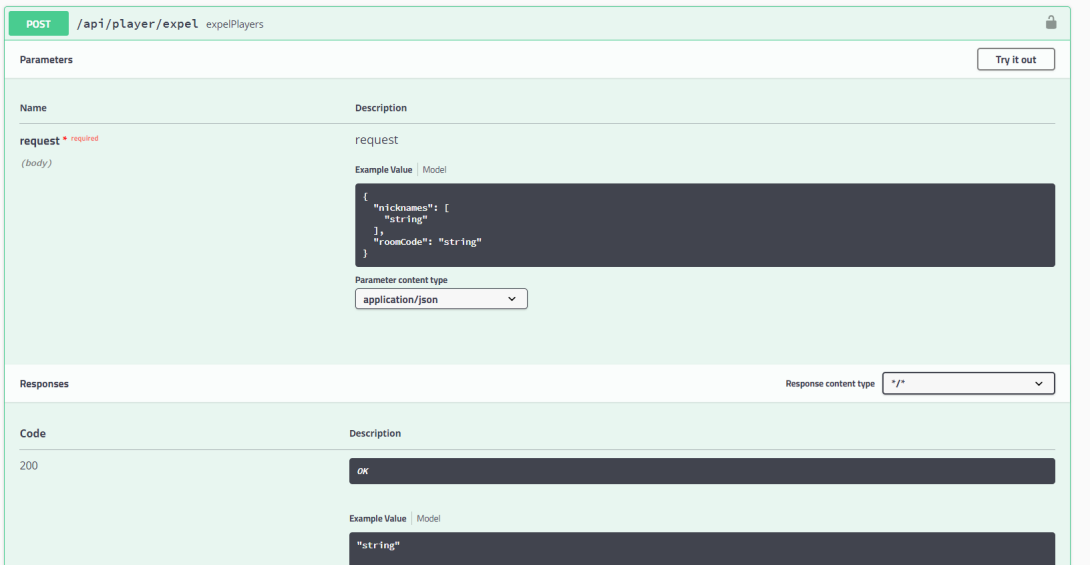

Figura 3.28: Documentación Swagger de la operación **expelPlayers**

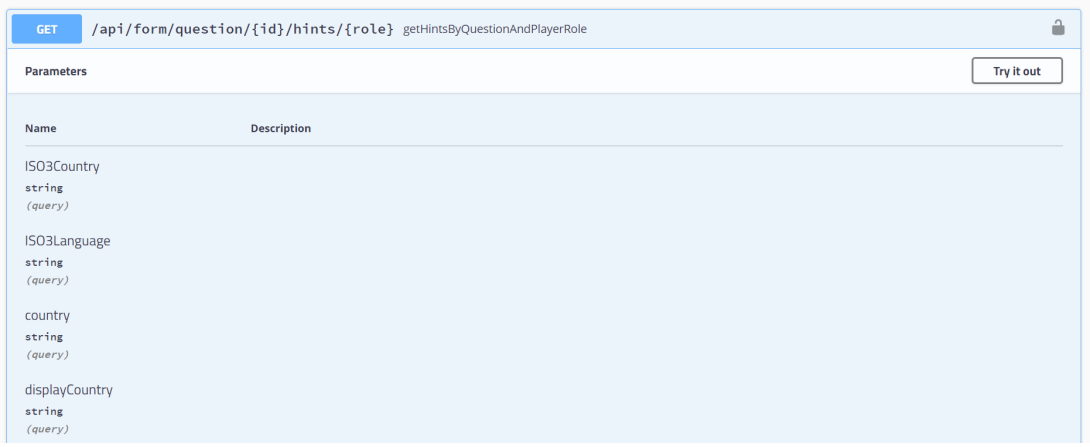

Figura 3.29: Documentación Swagger de la operación **getHintsByQuestionAndPlayerRole** (parte 1)

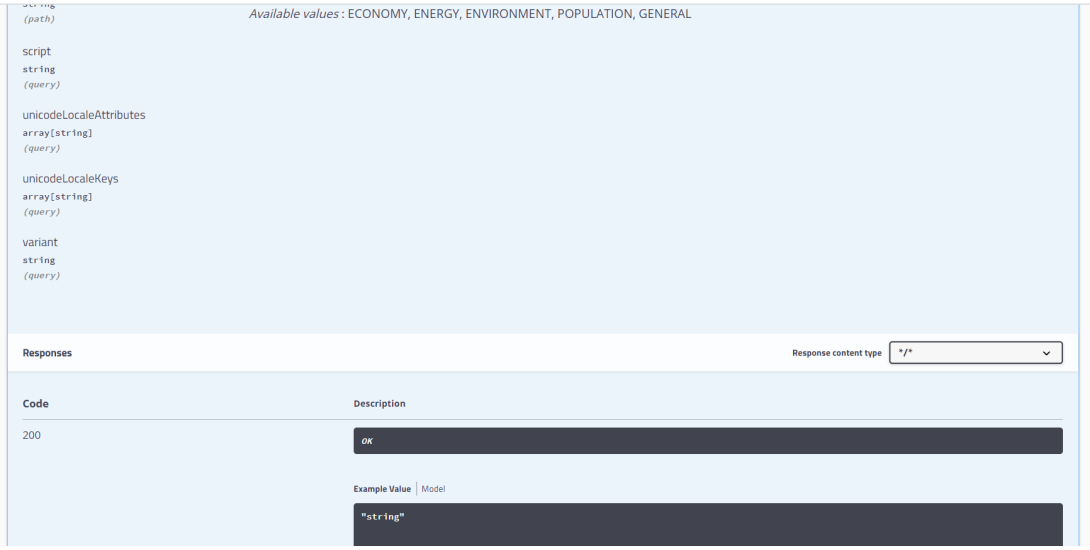

Figura 3.30: Documentación Swagger de la operación **getHintsByQuestionAndPlayerRole** (parte 2)

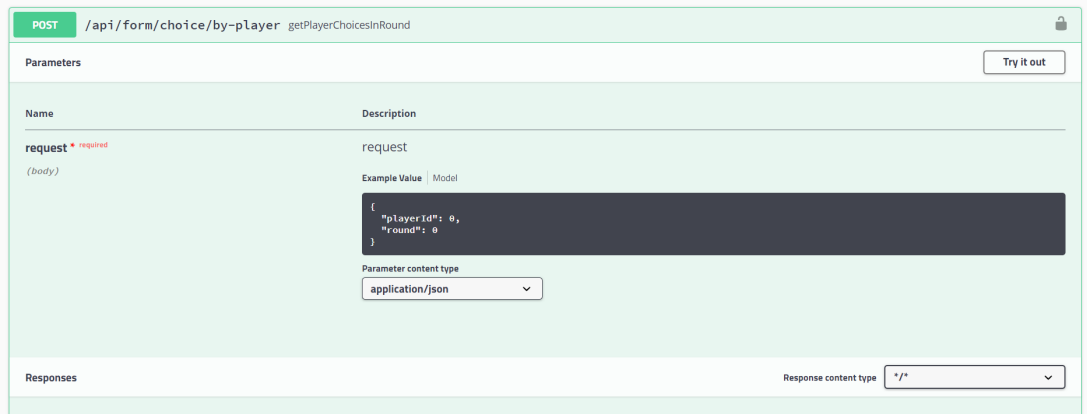

Figura 3.31: Documentación Swagger de la operación **getPlayerChoicesInRound** (parte 1)

Figura 3.32: Documentación Swagger de la operación **getPlayerChoicesInRound** (parte 2)

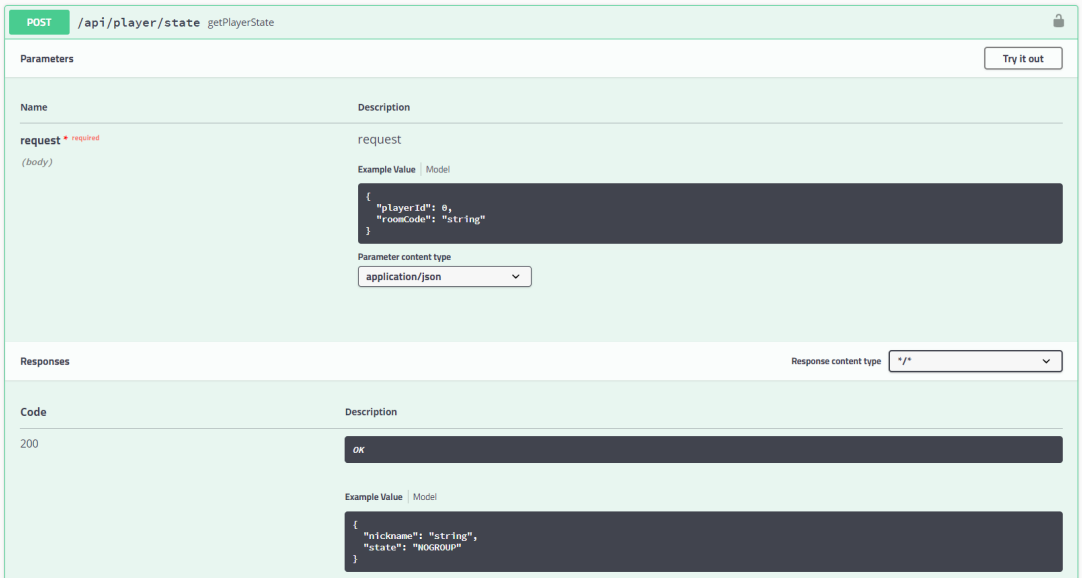

Figura 3.33: Documentación Swagger de la operación **getPlayerState**

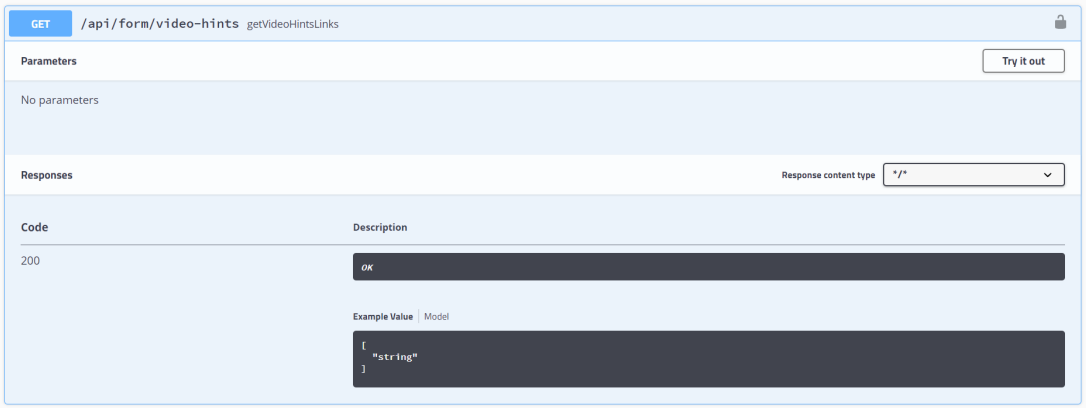

Figura 3.34: Documentación Swagger de la operación **getVideoHintsLinks**

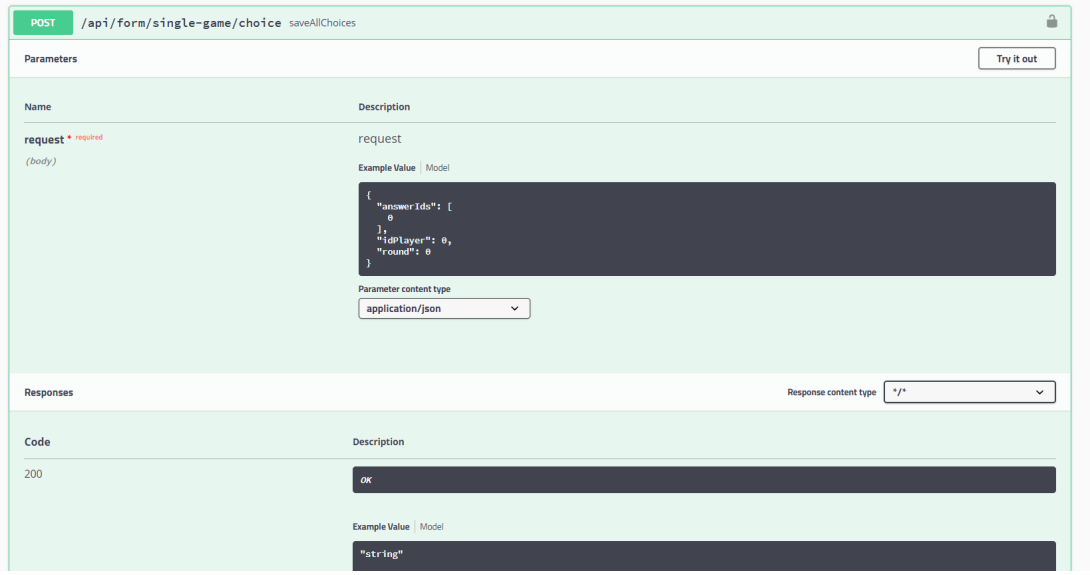

Figura 3.35: Documentación Swagger de la operación **saveAllChoices**

Internacionalización

La internacionalización completa de la aplicación se ha abordado en cada uno de los componentes de la arquitectura, por separado, de tal manera que cada componente oculte al resto la complejidad de la implementación de esta característica y ofrezca al resto una interfaz para hacer uso de la misma. El idioma por defecto que utilizará la aplicación será el inglés y soportará, al menos en esta versión, el español como idioma alternativo, aunque cada componente permite integrar nuevos idiomas de manera casi automática, a partir del uso de diferentes ficheros de texto con diferentes formatos cada uno de ellos. En cada una de las siguientes subsecciones se presenta el detalle de cómo se ha ejecutado esta tarea.

Internacionalización del back-end

La internacionalización del back-end tiene dos vertientes: la internacionalización de todos los mensajes de excepción y textos que devuelve este componente y la internacionalización de los textos almacenados en la base de datos.

- Para la internacionalización de todos los mensajes y textos del back-end se han creado dos *beans* [51], uno que se encarga de recuperar el idioma preferido por el usuario a partir de la cabecera *Accept-Language* de las peticiones HTTP y otro que se encarga de generar el mensaje en el idioma correspondiente. Para ello, este segundo elemento hace uso de un conjunto de ficheros de mensajes, cada uno de los cuales guarda las traducciones a un idioma determinado de todos los strings que aparecen en el código del back-end, de tal forma que este componente devuelve estos strings (desde el fichero correspondiente) según el idioma indicado por el primer *bean*. En la Figura 3.36 se presenta un extracto del fichero utilizado para el inglés.

De esta manera, para introducir un nuevo idioma en el back-end basta con generar una copia de estos ficheros en el idioma que se quiere introducir.

- Para la internacionalización de la base de datos se han modificado las tablas **Question**, **Answer** y **Hint**, para que la información traducible de estas tablas pase a estar en las tablas **Question_translation**, **Answer_translation** y **Hint_translation**. En estas tablas auxiliares se almacenan las traducciones de los elementos de las tablas correspondientes, junto con el idioma correspondiente a la traducción que almacenan. Estas traducciones se pueden recuperar a través de los mecanismos que implementa Spring Boot para el acceso a la base de datos, por lo que los usuarios de este componente pueden acceder a esta información a través de peticiones HTTP, igual que se hacía anteriormente. Esta modificación de la base de datos ya se indicó en la Sección 3.6, por lo que la estructura final de la base de datos se puede encontrar en los diagramas de la misma, concretamente en la Figura 3.25.

```
# Mensajes de la clase RoomController
room.start-game.success=Game started successfully
room.end-round.success=Round ended successfully
room.next-round.success=Next round started successfully
room.end-game.success=Game ended successfully

# Mensajes de la clase AuthService
auth.activate-account-email.subject=Please activate your account
auth.activate-account-email.body=Thank you for signing up to our social game Crossroads, please click on the below url to activate your account:\n http://localho
auth.retrieve-account-email.subject=Account credential retrieval
auth.retrieve-account-email.body=It seems like you forgot your credentials! Don't worry, we searched them for you! \n\t Username: {0}\n\tPassword: {1}

# Mensajes de la clase GraphicService
graph.save.success=Graphics saved successfully

# Mensajes de excepción de diferentes clases
exception.answer.translation-not-found=No answer translation with answer id: {0} and language: {1}
exception.question.translation-not-found=No question translation with question id: {0} and language: {1}
exception.auth.username-already-in-use=The username {0} is already in use. Try another one!
exception.auth.email-already-in-use=The email is already in use. Try another one!
```

Figura 3.36: Extracto del catálogo de mensajes en inglés del back-end

Internacionalización del front-end

La internacionalización del front-end requiere identificar todos los textos estáticos que se muestran al usuario durante la utilización de la aplicación, generar una copia de esos textos en el idioma destino y, finalmente, utilizar la versión de los textos correspondiente al idioma que el usuario tiene seleccionado en su navegador. Este uso del idioma predeterminado del navegador deberá hacerse de manera transparente al usuario, por lo que se realizará sin necesidad de intervención por parte del usuario. Además, las peticiones al back-end que lo requieran deberán reflejar en sus cabeceras el idioma en el que se desea recibir la respuesta. Para implementar todos estos requisitos se ha hecho uso de los mecanismos que ofrece el propio *framework* para ello. A continuación, se describe cada uno de los mecanismos implementados:

- Según se indica en la guía de internacionalización de Angular [4], la gestión de las traducciones se ha realizado mediante la adición de marcas a los textos del código que se desean traducir, la extracción del fichero de mensajes por defecto mediante

el uso de la herramienta **extract-i18n** del CLI de Angular y la traducción de ese fichero al español, según los pasos especificados por la guía. En la Figura 3.37 se muestra un extracto del fichero de traducciones al español de los textos del front-end.

- Para el envío de las peticiones al back-end, se ha añadido un nuevo elemento al front-end, de tipo *HttpInterceptor* [5]. Este componente captura todas las peticiones HTTP que se envían desde el front-end y permite modificar dichas peticiones. En este caso concreto, se ha utilizado este elemento para modificar las peticiones que se envían al back-end y así poder añadir la cabecera *Accept-Language* a las peticiones, para que el back-end haga uso de esta cabecera para devolver la respuesta de las peticiones en el idioma correspondiente.
- Para la detección automática del idioma en el navegador, se ha modificado la configuración del servidor Apache en el que se encuentra desplegado el front-end. Para ello, se hace uso de las preferencias definidas por el usuario, a través de la cabecera *Accept-Language*. Además, el servidor se ha configurado para que, si el usuario hace uso de un idioma no soportado por la aplicación, se haga uso del inglés como idioma por defecto. En la Figura 3.38 se muestra una captura de pantalla de la nueva configuración del servidor Apache.

```
<trans-unit id="8693043701637118233" datatype="html">
  <source>All the fields must be filled-in following the required format</source>
  <target>Debes rellenar todos los campos obligatoriamente, siguiendo el formato adecuado.</target>
  <context-group purpose="location">
    <context context-type="sourcefile">src/app/pages/chair/register/register.component.ts</context>
    <context context-type="linenumber">167</context>
  </context-group>
</trans-unit>
<trans-unit id="74964856291507309" datatype="html">
  <source>Incorrect password</source>
  <target>Contraseña incorrecta</target>
  <context-group purpose="location">
    <context context-type="sourcefile">src/app/pages/chair/register/register.component.ts</context>
    <context context-type="linenumber">204</context>
  </context-group>
  <context-group purpose="location">
    <context context-type="sourcefile">src/app/pages/chair/register/register.component.ts</context>
    <context context-type="linenumber">213</context>
```

Figura 3.37: Extracto del catálogo de mensajes en español del front-end

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        ServerName crossroads.geeds.eu
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/crossroads-frontend
        <Directory "/var/www/html/crossroads-frontend">
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted

                RewriteEngine on
                RewriteBase /
                RewriteRule ^../index\.html$ - [L]

                RewriteCond %{REQUEST_FILENAME} !-f
                RewriteCond %{REQUEST_FILENAME} !-d
                RewriteRule (..) $1/index.html [L]

                RewriteCond %{HTTP:Accept-Language} ^es [NC]
                RewriteRule ^$ /es/ [R]

                RewriteCond %{HTTP:Accept-Language} !^es [NC]
                RewriteRule ^$ /en/ [R]
        </Directory>
```

Figura 3.38: Configuración del servidor Apache para la detección automática del idioma del usuario

Internacionalización del recomendador

El recomendador está internacionalizado mediante el empleo de la biblioteca **Flask-Babel** [44]. Esta biblioteca funciona de manera similar a como lo hace la internacionalización en Angular, mediante el uso de un conjunto de ficheros, que actúan como catálogo de mensajes para un determinado idioma. De esta forma, y gracias al contenido de la cabecera *Accept-Language* de las peticiones HTTP recibidas por el recomendador, la biblioteca empleará en la respuesta los mensajes del fichero correspondiente al idioma solicitado por el usuario en la cabecera de la petición.

En la Figura 3.39 se muestra un extracto del fichero que guarda las traducciones de los textos en inglés al español.

```
#: recomendador.py:275
msgid ""
"CONGRATULATIONS, you have reached a solution that can be considered "
"sustainable in terms of economy and global temperature"
msgstr ""
"ENHORABUENA, has llegado a una solución que puede considerarse sostenible"
" en términos de economía y temperatura global"

#: recomendador.py:277
msgid "is in very positive values"
msgstr "está en valores muy positivos"

#: recomendador.py:278
msgid "moves away from the values agreed at the Paris summit on Climate Change"
msgstr "se aleja de los valores acordados en la cumbre de Paris sobre Cambio Climático"
```

Figura 3.39: Extracto del catálogo de mensajes en español del recomendador

Capturas de pantalla de la aplicación

En las Figuras 3.40 a 3.65 se muestran diferentes capturas de pantalla del estado actual de la aplicación web, tras aplicar todas las modificaciones indicadas en esta Sección. A continuación se muestran algunos comentarios acerca de los puntos de las capturas en los que puede apreciarse nueva funcionalidad:

- En la Figura 3.45 puede observarse que se ha modificado el formulario para incluir la opción de que la sala tenga grupos aleatorios. En el caso de crear una sala de este tipo, el grupo se asignará automáticamente a los jugadores y estos solo tendrán que seleccionar el rol que desempeñarán en el grupo. Esta última característica se muestra en al Figura 3.55.

- La reconexión de los jugadores a la sala es difícil de mostrar solo con capturas de pantalla, pues requiere de la realización de una acción en diferentes pasos. De todas formas se ha incluido en la memoria una captura del mensaje que se le muestra al jugador cuando se reconecta a la partida (Figura 3.66).

- En la Figura 3.67 se muestra la selección del jugador que se quiere expulsar y el mensaje que se le muestra al moderador tras hacer click en el botón de expulsión. En la Figura 3.68 se muestra la acción de expulsión desde el punto de vista del jugador. Como se puede comprobar por el mensaje que aparece en la captura, el jugador expulsado ha perdido la capacidad de enviar respuestas al formulario y continuar participando en la sala.

- En la Figura 3.56 pueden apreciarse algunos elementos nuevos de funcionalidad. Por un lado, se muestra la tabla resumen con el estado del formulario de cada jugador del grupo. Dicha tabla se irá actualizando de manera automática con las respuestas que vaya dando cada jugador. También puede apreciarse un indicador del rol del jugador en diferentes elementos como el formulario o los mensajes del chat. En el chat también pueden apreciarse los mensajes de los distintos jugadores, apareciendo el mensaje más reciente en la parte inferior del mismo.

- El uso de las pistas asociadas a una pregunta puede verse en la Figura 3.58. Tras hacer click en el botón correspondiente, se muestra un desplegable con el texto asociado a la pista de la pregunta.
- En la Figura 3.63 se puede ver un desplegable con el resumen de la propuesta de respuestas que ha dado el grupo para el formulario. En dicho resumen se hace uso de los nombre cortos de las preguntas.
- La internacionalización no puede apreciarse en las capturas de pantalla, pues esta se produce de manera transparente al usuario y no hay ningún elemento visible que haga perceptible el cambio de idioma.
- En todas las Figuras puede apreciarse el cambio de la *skin* de la interfaz.

Figura 3.40: Captura de la nueva pantalla de inicio de de la aplicación

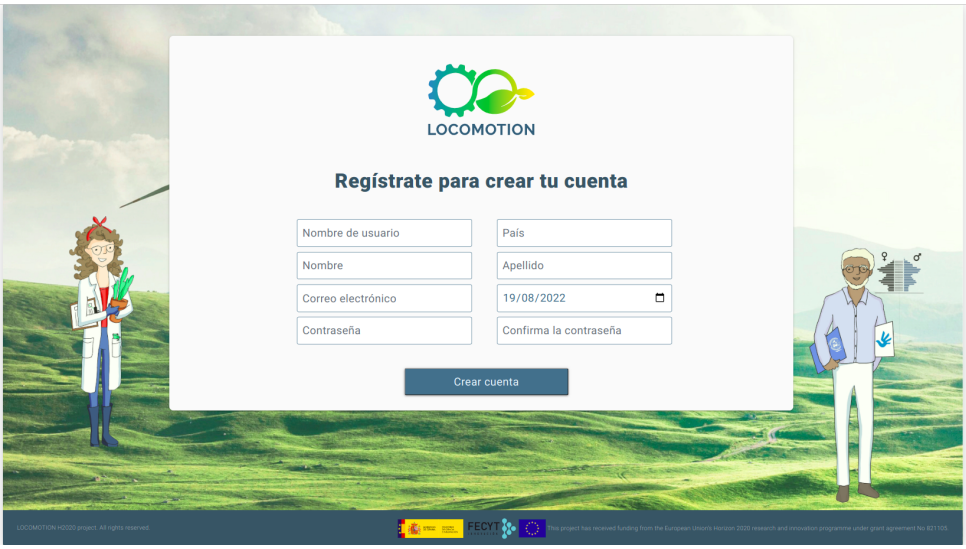

Figura 3.41: Captura de la nueva pantalla de registro

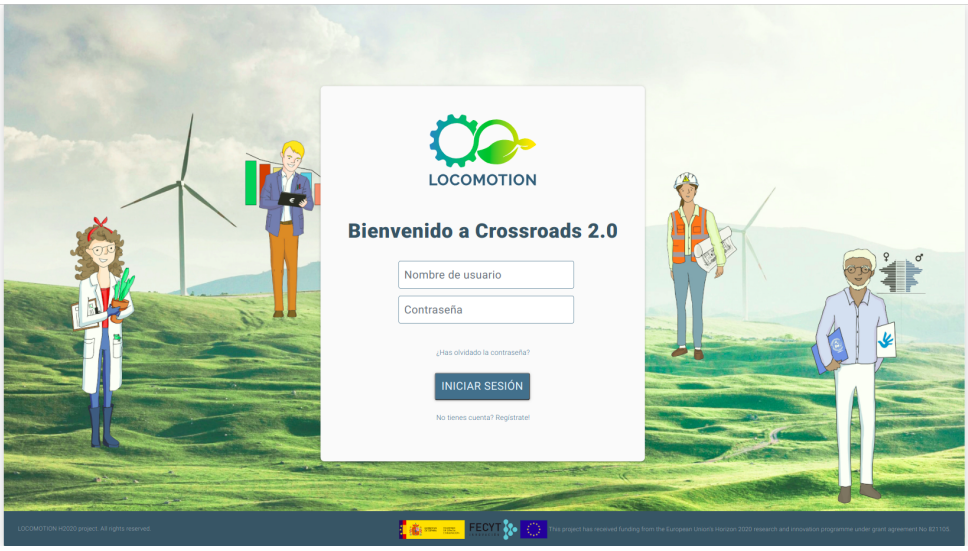

Figura 3.42: Captura de la nueva pantalla de inicio de sesión para moderadores

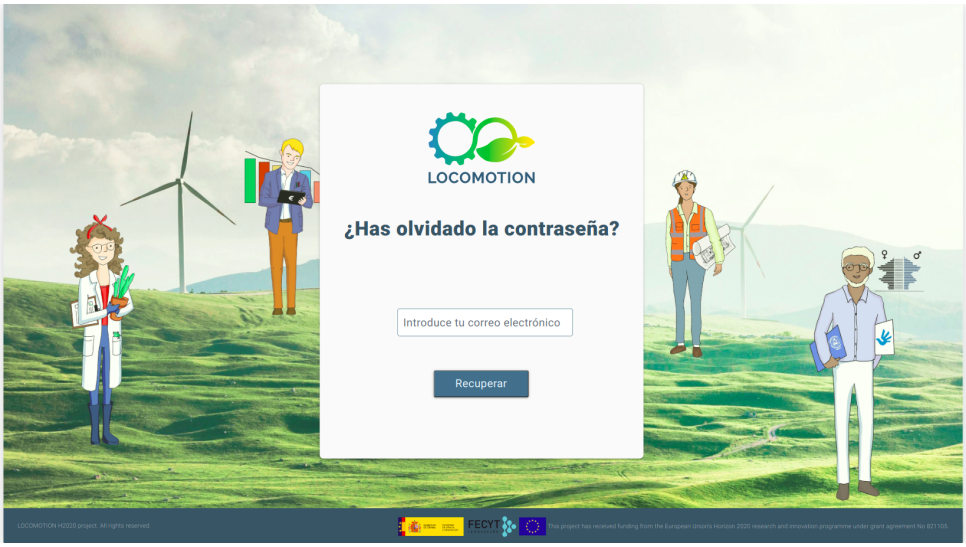

Figura 3.43: Captura de la nueva pantalla de recuperación de la contraseña

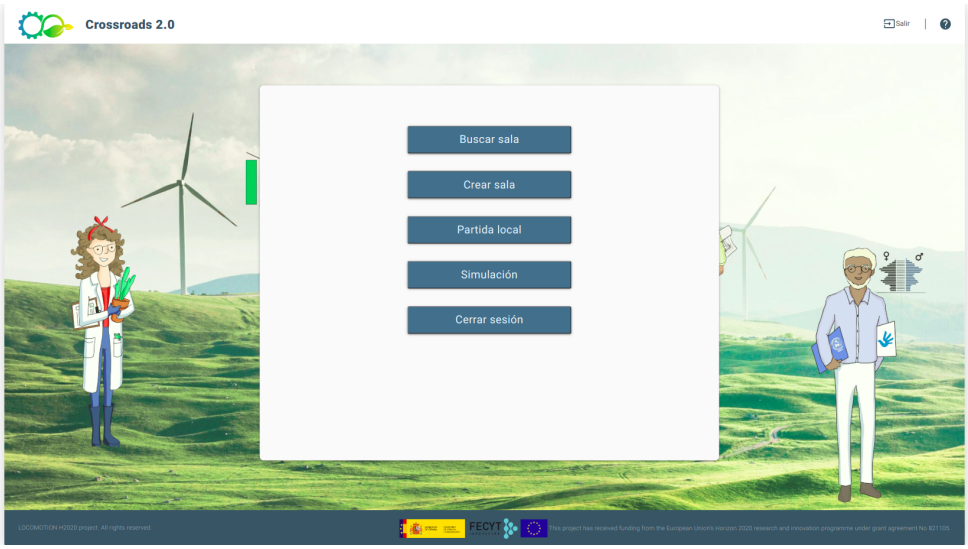

Figura 3.44: Captura del nuevo menú del usuario registrado en la aplicación

Figura 3.45: Captura del nuevo formulario de creación de salas de partida

Figura 3.46: Captura de la sala de espera del moderador

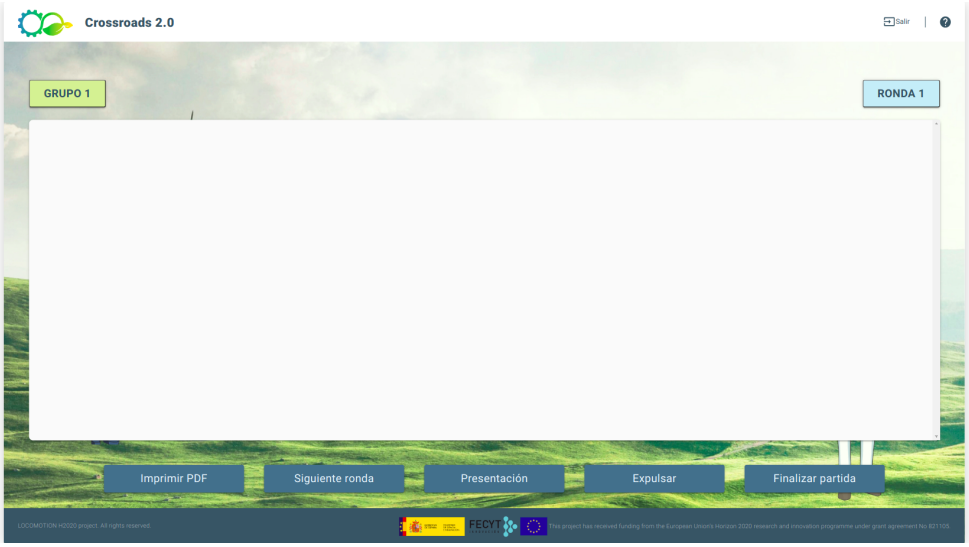

Figura 3.47: Captura del panel de monitorización de la partida del moderador

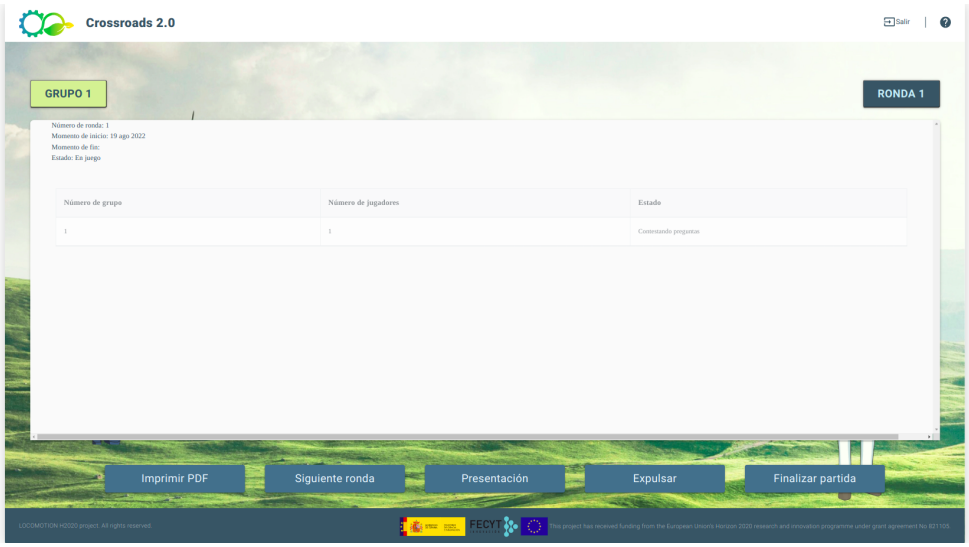

Figura 3.48: Captura del panel de monitorización de la partida del moderador (comprobación del estado de la ronda)

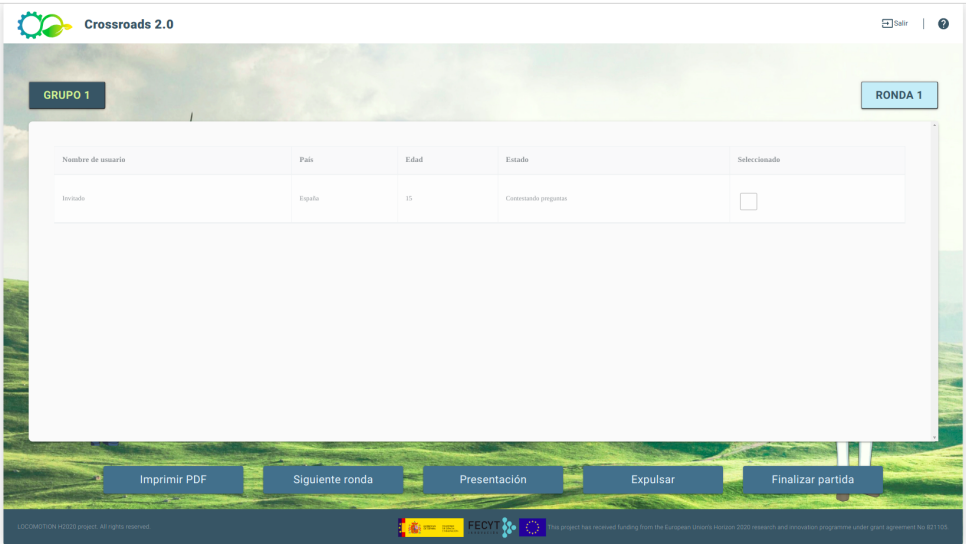

Figura 3.49: Captura del panel de monitorización de la partida del moderador (comprobación del estado de los jugadores del grupo)

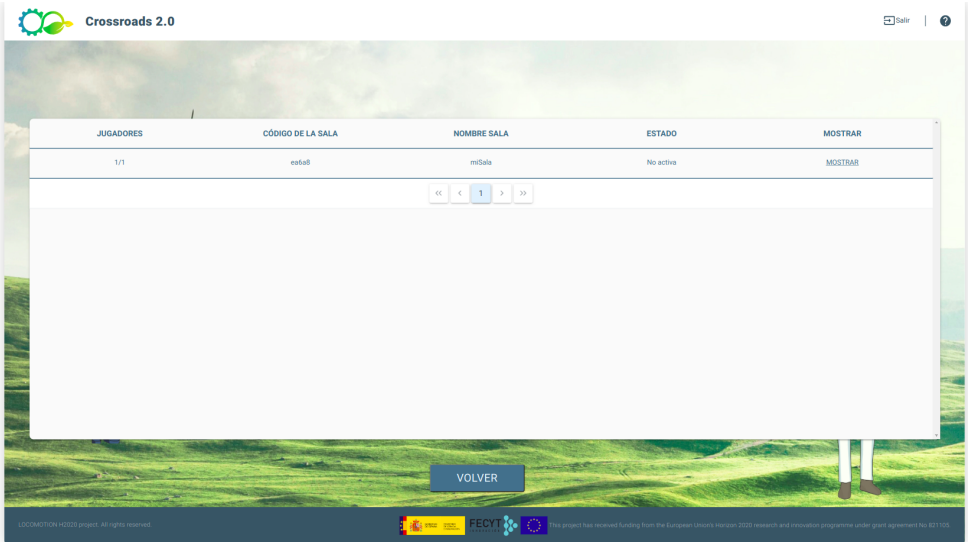

Figura 3.50: Captura de la pantalla de reconexión del moderador a una sala

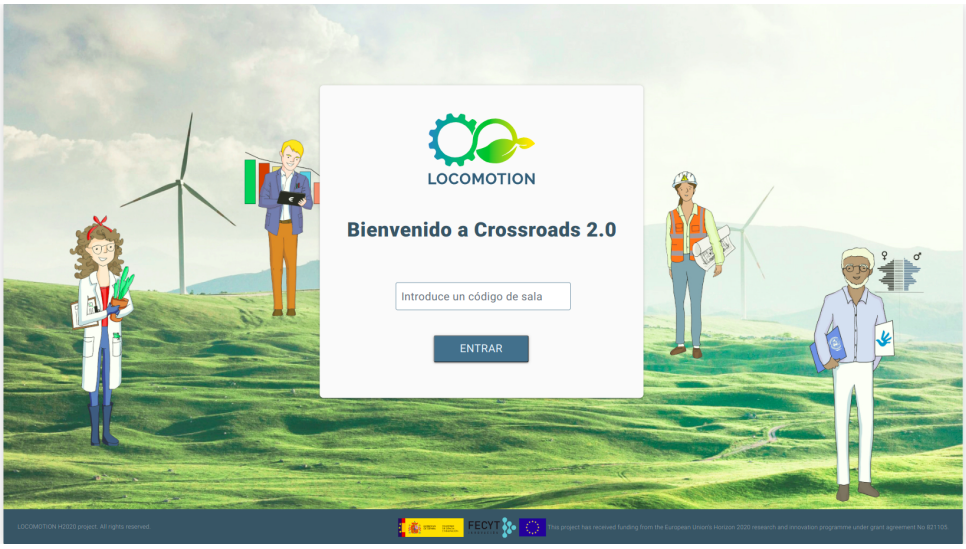

Figura 3.51: Captura de la pantalla de acceso a las salas de partida para los jugadores

Figura 3.52: Captura del formulario que los jugadores deberán rellenar con sus datos

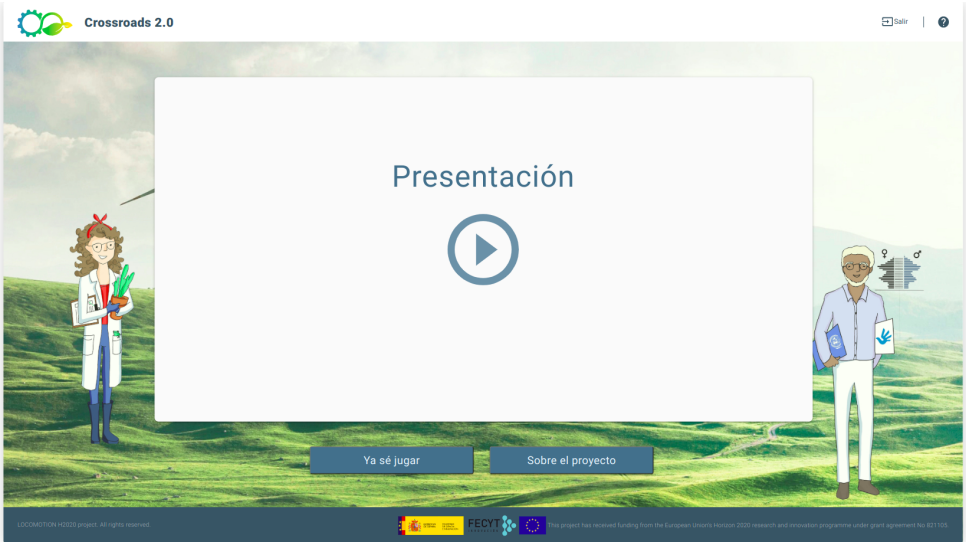

Figura 3.53: Captura de la pantalla de información e instrucciones acerca del juego

Figura 3.54: Captura de la sala de espera de los jugadores

Figura 3.55: Captura de la sala de espera de los jugadores. En esta partida los grupos se han formado de manera aleatoria, por lo que el jugador solo tiene que seleccionar rol

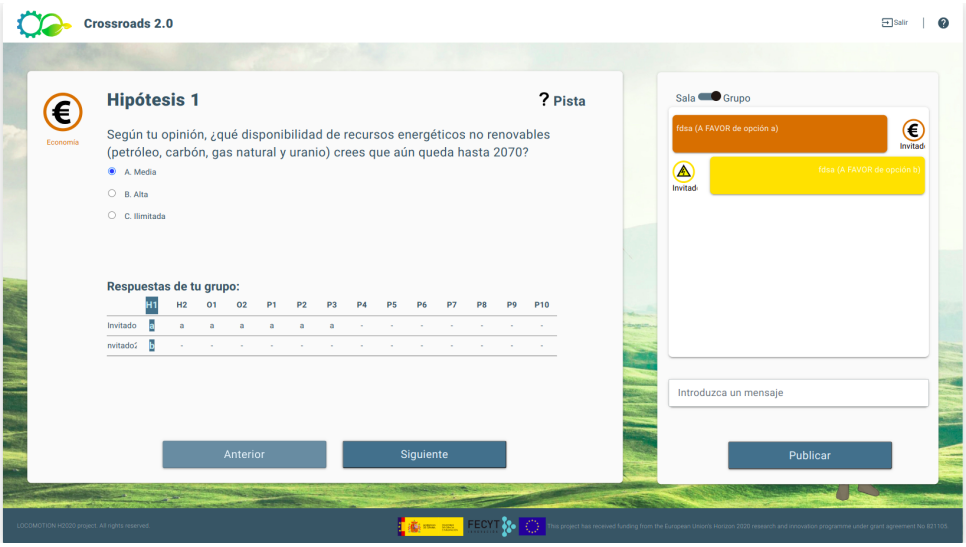

Figura 3.56: Captura del formulario de preguntas que cada jugador deberá responder

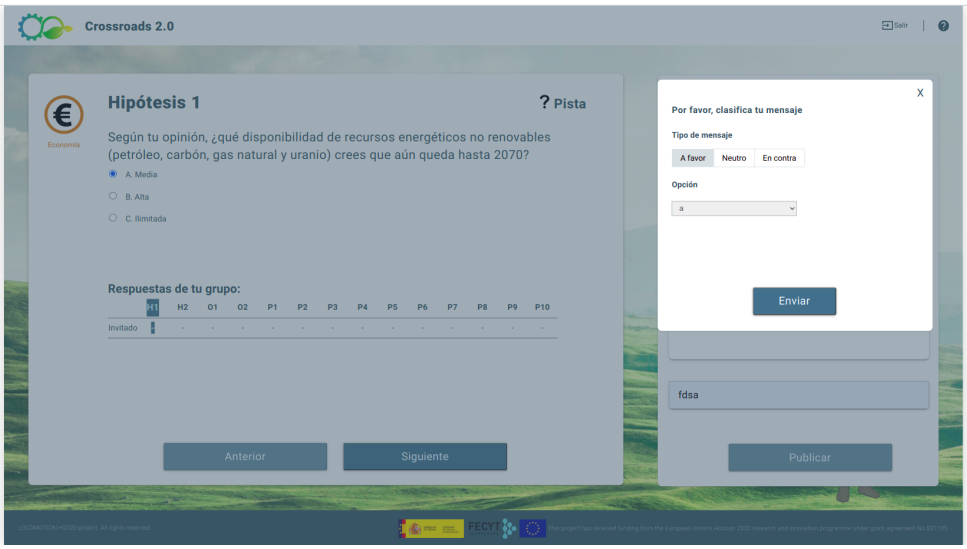

Figura 3.57: Captura del formulario de preguntas que cada jugador deberá responder (vista del pop-up con los datos del mensaje que se va a enviar a través del chat)

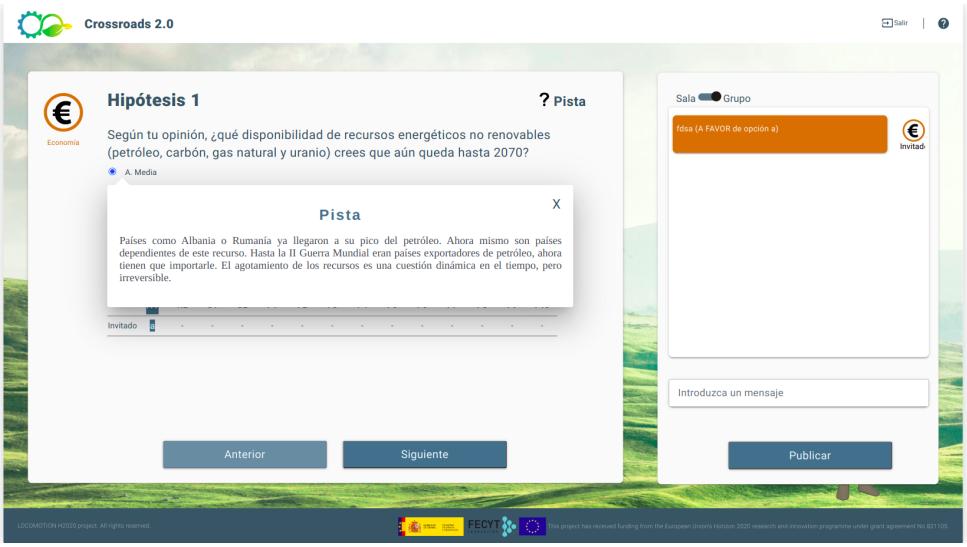

Figura 3.58: Captura del formulario de preguntas que cada jugador deberá responder (visionado de una pista para la pregunta)

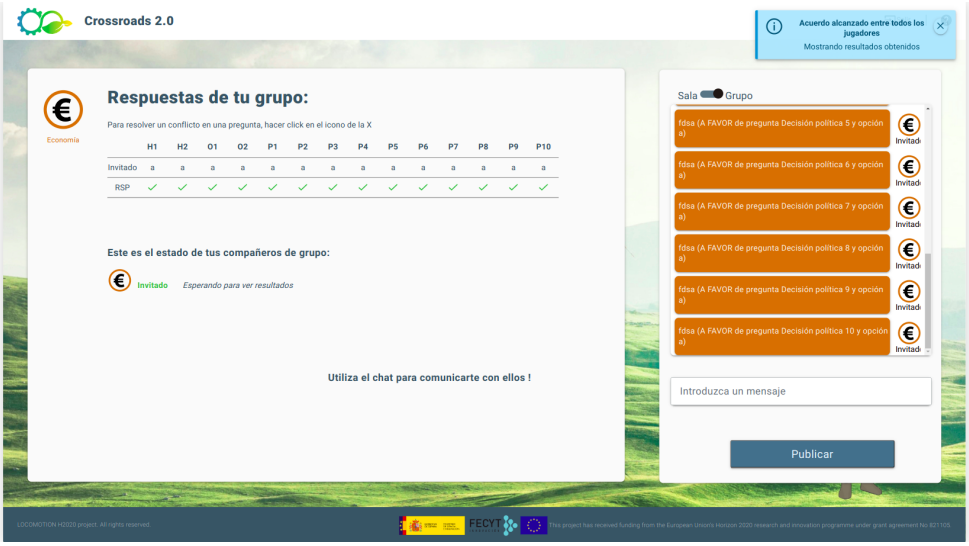

Figura 3.59: Captura de la pantalla de comprobación de conflictos (sin conflictos)

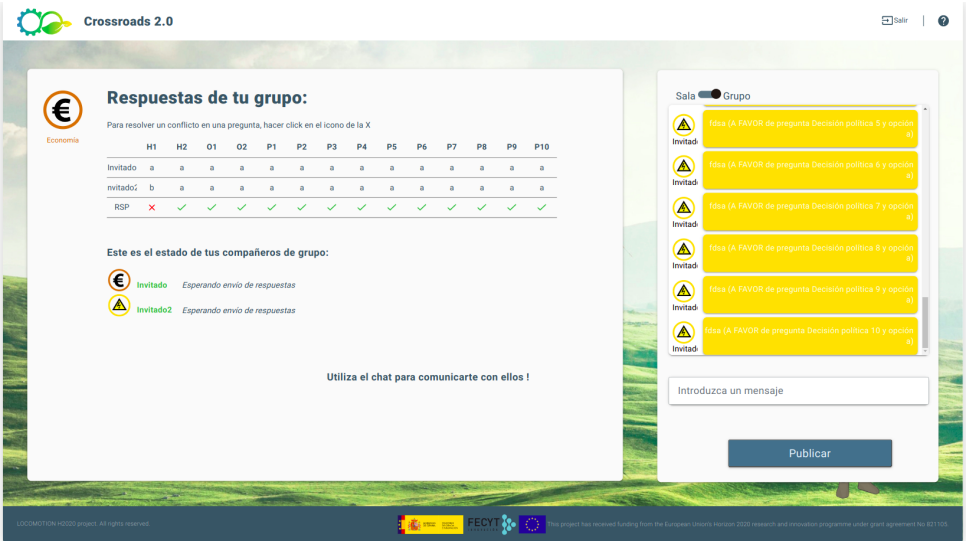

Figura 3.60: Captura de la pantalla de comprobación de conflictos (conflicto en la primera pregunta)

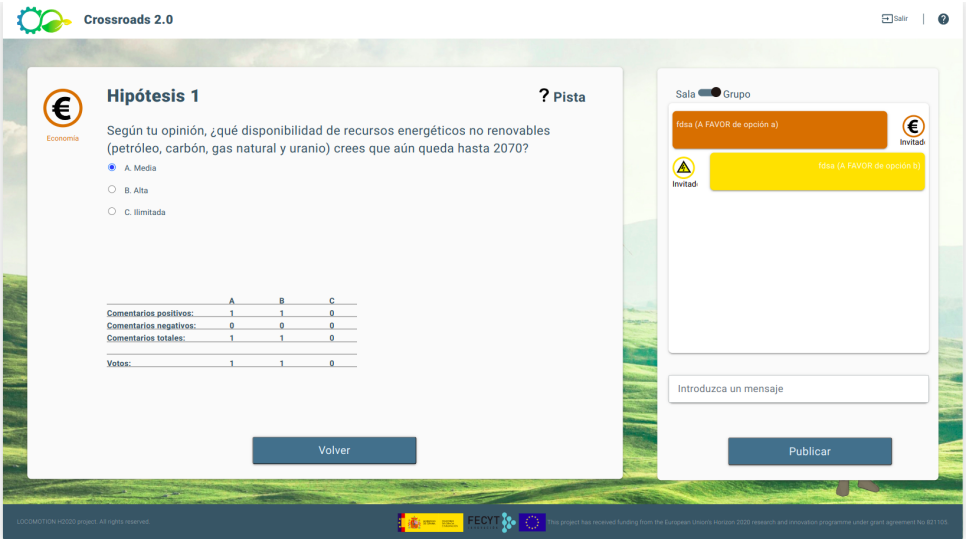

Figura 3.61: Captura de la pantalla de resolución de conflictos para la primera pregunta

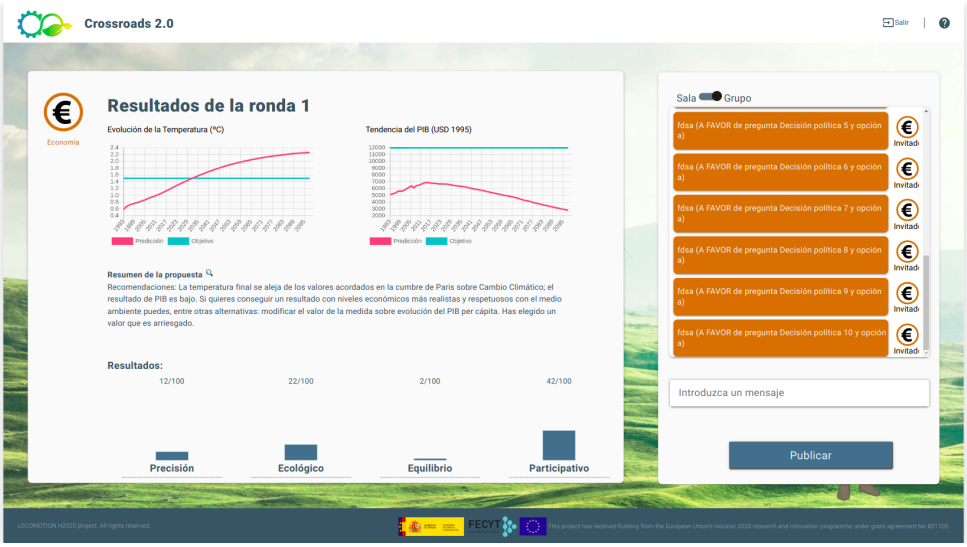

Figura 3.62: Captura de la pantalla de resultados de ronda

Figura 3.63: Captura de la pantalla de resultados de ronda (visionado del resumen de la propuesta seleccionada por el grupo)

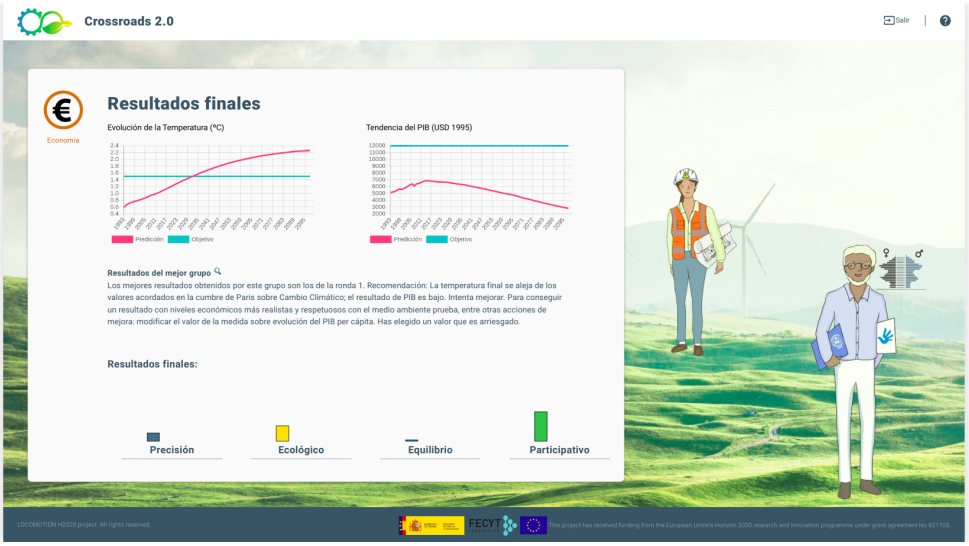

Figura 3.64: Captura de la pantalla de resultados finales

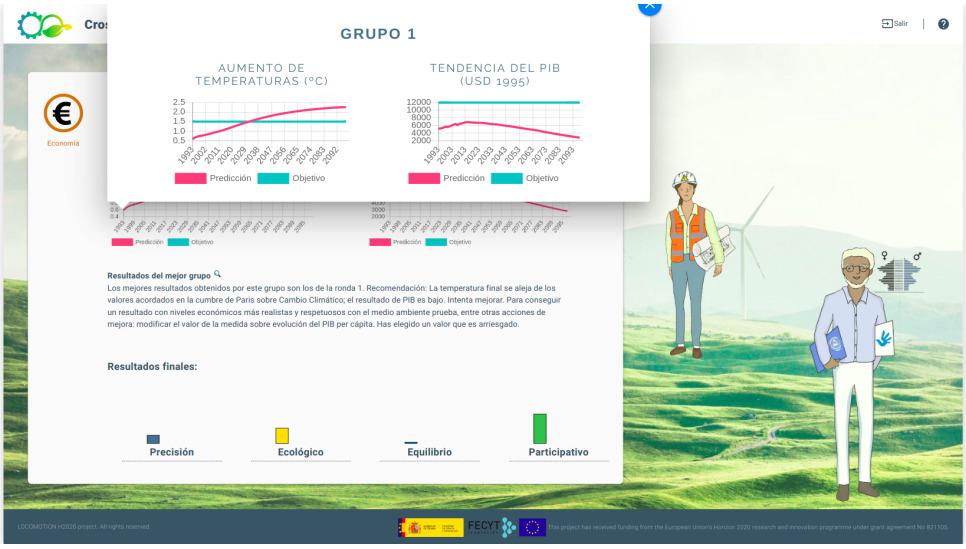

Figura 3.65: Captura de la pantalla de resultados finales (resultados del mejor grupo)

Figura 3.66: Captura del mensaje mostrado a los jugadores cuando se reconectan a la sala de partida en la pantalla del formulario de preguntas

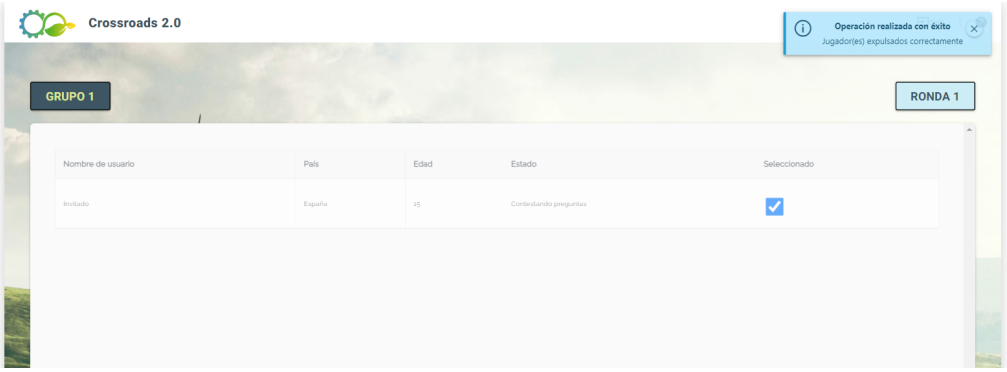

Figura 3.67: Captura de pantalla de la expulsión de un jugador de un grupo

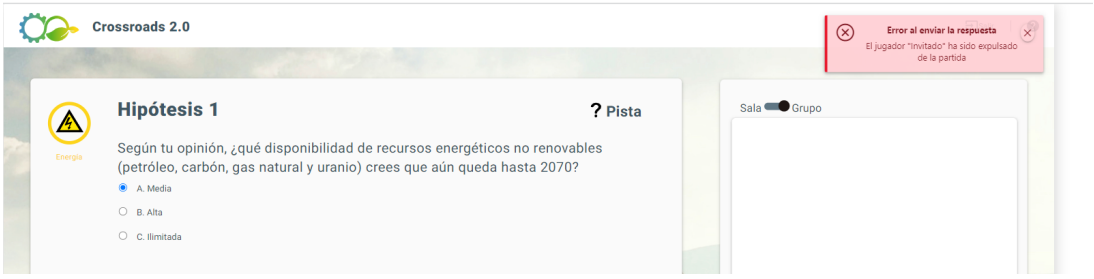

Figura 3.68: Captura de pantalla del mensaje mostrado al jugador expulsado cuando trata de enviar una respuesta

Pruebas funcionales

Para probar toda la funcionalidad desarrollada como parte de este proyecto se ha modificado la base de casos de prueba de la que se partía inicialmente, mediante la actualización de aquellos tests que se han quedado desfasados y la adición de nuevos casos de prueba. En el momento de finalización del proyecto, se dispone de 183 casos de prueba, agrupados en 18 ficheros de código JavaScript y el porcentaje de éxito de los mismos es del 100 %, es decir, todos los tests obtienen su resultado esperado.

En cuanto a la cobertura de código de los tests (Figura 3.69), se ha comprobado que esta se mantiene en valores próximos al 95 % en cobertura de sentencia y próximos al 80 % en cobertura de rama (condición-decisión), por lo que se puede garantizar con bastante confianza el buen funcionamiento de la aplicación.

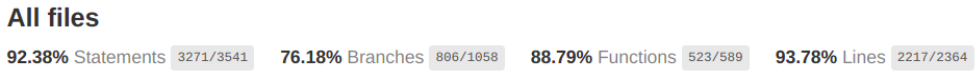

Figura 3.69: Nuevo informe de cobertura del código

3.8. Seguimiento del proyecto

En esta última Sección de este Capítulo se presenta un resumen de los tiempos que ha llevado la realización de cada parte del desarrollo de esta segunda versión de la aplicación. Dicho resumen se muestra a continuación:

- **Internacionalización**: 300 horas.
- **Modificación y mantenimiento del back-end**: 227 horas
- **Modificación y mantenimiento del front-end**: 270 horas.
- **Testing de la aplicación**: 80 horas

Tras sumar todos los tiempos indicados en el anterior listado, tenemos que para la realización de todo el trabajo asociado han sido necesarias 877 horas de trabajo.

En cuanto a la calendarización de los trabajos realizados, se ha seguido el plan temporal establecido durante la planificación (Sección 3.3), por lo que las fechas de inicio y fin de este miniproyecto coinciden con las especificadas en dicho plan. Teniendo en cuenta la estimación del esfuerzo en número de horas realizada en la Sección 3.1, la finalización de estas tareas ha conllevado un sobrecoste en número de horas, superándose las 480 horas estimadas en 397 horas.

Teniendo en cuenta el sobrecoste en horas de esfuerzo que se ha producido, es necesario analizar las causas del mismo. La principal tarea que ha llevado más tiempo del esperado es la internacionalización completa de la aplicación, pues fue necesario elaborar diferentes prototipos de aplicaciones internacionalizadas antes de poder abordar la internacionalización de Crossroads 2.0. También hubo problemas con el chat de la aplicación, pues se detectó un bug a la hora de recuperar los mensajes desde el back-end. Este bug se producía por un error con la fecha y la hora de la máquina servidor, pues esta no estaba bien actualizada y no coincidía con la fecha y hora indicadas en la petición para recuperar los mensajes. Este bug fue difícil de depurar, pues solo se producía en el entorno de producción. Otra tarea que fue ardua y que consumió más tiempo del esperado fue la extracción de un nuevo componente con la matriz de conflictos de las preguntas (Sección 3.6), pues hubo que decidir que código se podía reutilizar y que código habría que mantener en el componente original, a parte de definir la interfaz del mismo, para que este fuese lo más independiente posible de los componentes que pasarían a utilizarlo. La última tarea que me gustaría remarcar como costosa en tiempo y esfuerzo sería la implementación de la reconexión del jugador a la sala, pues se llegaron a implementar varias versiones de esta funcionalidad que hubo que descartar, ya que no se cubrían adecuadamente todos los casos de reconexión posibles, lo que desembocaba en múltiples fallos y pequeños problemas de seguridad en el uso de esta funcionalidad.

4: Estado del arte en pruebas de carga

En este Capítulo se presentan los resultados del procedimiento de investigación realizado para comprobar el estado del arte en el ámbito de las herramientas para la realización de las pruebas de carga. Dicho procedimiento ha tenido dos vertientes principales, de acuerdo con la fuente de información consultada. En primer lugar, se ha realizado una búsqueda de literatura académica que trate el tema. En segundo lugar, se ha consultado la literatura gris de Internet, pues en el ámbito de la Ingeniería del Software no se puede desdeñar la opinión de los profesionales que tienen una amplia experiencia en el sector. En relación con esta búsqueda de información se ha realizado una encuesta a diferentes profesionales que trabajan en el ámbito de las pruebas de rendimiento, con el objetivo de obtener algunas directrices que orienten la forma de realizar la evaluación del rendimiento de Crossroads 2.0. Los resultados de todos los procesos de búsqueda y consulta se presentan en las siguientes secciones.

4.1. Búsqueda en literatura académica

Metodología

Para realizar la consulta de artículos académicos, se ha decidido realizar una búsqueda por palabras clave en el buscador especializado **Web of Science** (WoS) [58]. La búsqueda a partir de palabras clave se basa en identificar en cada documento un conjunto de términos o frases que estén relacionados con la temática de la búsqueda. De esta forma (y mediante el uso de conectores lógicos) se puede elaborar una cadena de búsqueda con todas las palabras clave que recoja todas las posibles combinaciones de términos que puedan aparecer en un artículo. La cadena de palabras clave que se ha elaborado para esta búsqueda es la siguiente:

- TS=((“load testing” OR “scalability testing”) AND “collaborative web applications”)

De la cadena de búsqueda anterior es necesario destacar el uso del prefijo TS, que indica que la identificación de palabras clave se realizará en la temática del artículo, el *abstract* o

resumen del mismo y en la lista de términos clave especificada por los autores del artículo. Además, se ha incluido la palabra clave *collaborative*, pues una de las características principales de la aplicación que se va a probar es su naturaleza colaborativa, puesto que los jugadores de cada grupo tienen que colaborar para poder completar la partida.

En cuanto a la forma de procesar los artículos devueltos por el buscador, se realizará una lectura en diagonal de cada uno de ellos, lo que incluye el título, el *abstract*, la introducción y las conclusiones. A través de esta lectura, se determinará la temática principal del artículo y se comprobará si está relacionada en realidad con el dominio del problema que se pretende resolver. Para todos aquellos artículos cuya temática esté estrechamente relacionada con el problema, se realizará una lectura completa de los mismos.

Resultados obtenidos

Tras utilizar la cadena de búsqueda en WoS, se pudo comprobar que esta no devolvía ningún resultado, debido a la inclusión del término *"collaborative"* en la cadena, por lo que se decidió eliminar dicho término y realizar una inspección de todos los artículos devueltos por la nueva cadena de búsqueda.

Con el uso de la nueva cadena de búsqueda se han obtenido 24 artículos en total. De esos 24 artículos, 7 no estaban relacionados con la evaluación del rendimiento de aplicaciones web, por lo que se han descartado. Además, de los 17 artículos restantes, ninguno presenta relación estrecha con la realización de pruebas de carga de aplicaciones web colaborativas. Los principales temas tratados por estos artículos se recogen a continuación:

- Optimización de los recursos hardware empleados por una aplicación desplegada en el cloud.
- Modelado de la carga de trabajo que se utilizará para las pruebas, de tal forma que esta represente de manera realista el comportamiento de los usuarios finales.
- Metodología para la realización de las pruebas de carga.
- Uso de la tecnología cloud para la generación de carga de trabajo.

Además, es de especial interés el artículo llamado *"A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications"* [40], pues en este se realiza una comparativa de dos herramientas de uso extendido, como son Apache JMeter y Locust, con el objetivo de determinar aquella herramienta que ofrece un mayor rendimiento a la hora de ejecutar las pruebas de carga en diferentes sitios web. La principal conclusión de ese artículo es que Locust tiene mayor rendimiento que JMeter, ya que presenta un tiempo de ejecución de las pruebas menor que JMeter.

El listado completo de artículos devueltos por la cadena de búsqueda modificada se muestra en la Tabla 4.4.

4.2. Búsqueda en la literatura gris de Internet

Metodología

La búsqueda en la literatura gris de Internet consistirá en consultar diferentes artículos y sitios web en los que se presenten herramientas con las que se realicen pruebas de carga de aplicaciones web.

Para realizar esta tarea, se hará uso del buscador Google y de cadenas de búsqueda como *"load testing tools"*, *"open source load testing tools"* o *"collaborative web application load testing"*. Como la cantidad de resultados que puede devolver Google con consultas de este tipo es demasiado grande, se seleccionarán las herramientas que aparezcan más veces repetidas en las diferentes páginas consultadas, hasta obtener, al menos, 25 herramientas.

Además, para cada una de las herramientas se identificará si esta es *open source*, el lenguaje de programación empleado para la creación de los test, la existencia de versiones gratuitas de la misma y la explotación que hace la herramienta de la tecnología cloud, si procede.

Resultados obtenidos

Tras la consulta de más de 100 sitios web, entre artículos que recopilan varias herramientas y páginas web asociadas a una sola herramienta, se ha elaborado un listado de 29 herramientas de uso extendido. Dicho listado se muestra en la Tabla 4.3, ordenado alfabéticamente por nombre de herramienta.

De dicho listado se desprende que hay gran variedad de herramientas que tienen características de lo más dispares, pues hay herramientas de pago, gratuitas, *open source* o no, etc. En cuanto al uso de la tecnología cloud, se han identificado diferentes posibilidades, que se agrupan en dos categorías: ejecución de la herramienta (SaaS) y generación de carga de trabajo a través de máquinas desplegadas en el cloud.

4.3. Consulta a profesionales del sector

Para completar el proceso de investigación sobre el estado del arte en pruebas de carga, se ha decidido realizar una consulta a diversos profesionales del sector. Además, esta consulta tiene otro objetivo, pues se pretende obtener orientación por parte de estos profesionales para realizar las pruebas de carga de Crossroads 2.0.

Metodología

Se ha decidido abordar esta parte de la investigación mediante la elaboración de una encuesta, que será distribuida por correo electrónico a algunas empresas que colaboran de manera estrecha con la escuela y a través de LinkedIn.

La encuesta se creará utilizando Google Forms y deberá contener preguntas relacionadas con la empresa, las aplicaciones y la experiencia de la persona en el sector, el proceso habitual de análisis de rendimiento que sigue la empresa (que incluye las herramientas utilizadas y el modelado de la carga de trabajo) y la existencia de diferencias en la aplicación de dicho proceso para una aplicación de características similares a Crossroads 2.0. El formulario está disponible a través del siguiente enlace.

Resultados obtenidos

Después del transcurso de varias semanas desde que se distribuyó la encuesta por primera vez tan solo se ha obtenido una respuesta a la misma, por lo que los resultados obtenidos a través de esta no son concluyentes. De todas formas, se ha decidido incluir en esta sección el detalle de esa única respuesta, por motivos informativos.

- La persona encuestada realiza su actividad en una empresa grande, en un equipo que realiza el desarrollo y testing de aplicaciones web. Además, en ese equipo de trabajo se realizan pruebas de carga y escalabilidad de las aplicaciones web que desarrollan.
- La aplicación web en la que la persona encuestada tiene experiencia de pruebas de carga tiene más de 100.000 y menos de 1.000.000 de usuarios concurrentes.
- La persona encuestada tiene más de 10 años de experiencia en el campo de la realización de pruebas de carga y escalabilidad de aplicaciones web.
- La herramienta utilizada en el equipo de trabajo es JMeter.
- La metodología empleada para realizar las pruebas de carga y escalabilidad de una aplicación web incluye los siguientes pasos: Planteamiento de los casos de uso, planteamiento de los umbrales de ejecución de las máquinas virtuales y servicios y ejecución de las pruebas.
- En cuanto al modelado de la carga de trabajo, la persona encuestada propone realizar el planteamiento de los casos de uso y grabar los escenarios con la herramienta JMeter.
- Para realizar las pruebas de carga de Crossroads 2.0, el encuestado propone añadir más aleatoriedad a las pruebas y revisar en profundidad los patrones de uso de la aplicación.
- Sugerencias generales: Tener en cuenta los casos de uso y la grabación de los mismos e intentar realizar las pruebas con la mayor aleatoriedad posible para los usuarios simulados, lo que aumentará el realismo de los resultados.

4.4. Conclusiones de la investigación

Tras la finalización del proceso de investigación, se han extraído las siguientes conclusiones:

1. El modelado de la carga de trabajo es un tema muy importante pues, si esta no representa de manera realista el comportamiento de los usuarios, los resultados obtenidos de las pruebas pueden no ser concluyentes.

2. La tecnología cloud juega un papel importante en la realización de pruebas de carga de aplicaciones web. Este papel se manifiesta en distintas vertientes, desde ser la base sobre la que se despliega la aplicación que se desea probar, hasta ser el medio de acceso a gran número de herramientas que permiten realizar pruebas de carga, pasando por la generación de los usuarios virtuales que harán uso de la aplicación durante las pruebas.

3. Los resultados de la encuesta no pueden ser tomados en cuenta, debido al número bajo de respuestas. De todas formas, la única respuesta recibida incide en la importancia de la aleatoriedad al modelar la carga de trabajo y revisar los patrones de uso de la aplicación para realizar esta tarea, por lo que parece reforzar lo comentado en el primer punto de este listado.

4. No parece que existan diferencias a la hora de probar una aplicación web colaborativa, como se desprende de la búsqueda realizada de artículos académicos.

Nombre herramienta	Open source	Plan de prueba/Versión gratuita	Lenguaje de los tests	Posibilidad cloud
Artillery.io	Sí	Versión gratuita	JavaScript	Versión de pago en AWS
Bees with Machine Guns	Sí	Es gratis	Python	Generadores de carga en AWS
BlazeMeter	No	Versión freemium	YAML/JSON	SaaS/Generadores de carga
Boomq.io	No	Versión gratuita muy limitada (10 usuarios y 15 min de test)	Ninguno	Ejecución de los test en cloud
CloudTest from Akamai	No	Prueba de 30 días	JavaScript	Generadores de carga
Eggplant	No	Demo gratuita	DSL propio	Generadores de carga
Flood Element	Sí	Es gratis	TypeScript	Generadores de carga
Fortio	Sí	Es gratis	Línea de comandos + JSON	
Gatling	Sí	Versión gratuita	Scala	Hosting para la versión de pago
HeadSpin	No	Prueba gratuita	Ninguno	
JMeter	Sí	Es gratis	XML	
K6	Sí	Versión gratuita	JavaScript	Versión cloud de pago
LoadFocus	No	Prueba de 14 días	Ninguno	SaaS
LoadNinja	No	Prueba de 14 días	JavaScript (pequeños fragmentos)	Generadores de carga
LoadRunner	No	Prueba gratuita	C	SaaS
Loadster	No	Primeras 50 horas de test gratuitas	JavaScript (pequeños fragmentos)	SaaS
LoadView	No	Versión de prueba de 30 días	Ninguno	SaaS
Locust	Sí	Es gratis	Python	
Neotys Neoload	No	Prueba de 30 días	JavaScript (no es necesario)	SaaS
nGrinder	Sí	Es gratis	Jython o Groovy	
Parasoft Load Test	No	Prueba gratuita	Ninguno	Despliegue del agente en máquinas del cloud
puppeteer-webperf	Sí	Es gratis	JavaScript	
Rational Performance Tester	No	No	Java	SaaS
Siege	Sí	Es gratis	Ninguno	
Taurus	Sí	Es gratis	YAML/JSON	SaaS en el cloud de BlazeMeter
The Grinder	Sí	Es gratis	Jython	
Tsung	Sí	Es gratis	XML	
WAPT	No	Versión con características recortadas	JavaScript (pequeños fragmentos)	SaaS
WebLOAD	No	Prueba gratuita	JavaScript	SaaS/Generadores de carga

Tabla 4.3: Listado de herramientas de pruebas de carga elaborado a partir de los resultados de la búsqueda en la literatura gris. El listado está ordenado alfabéticamente a partir del nombre de la herramienta

Nombre del artículo	DOI	Tema relacionado con las pruebas de carga	Temática principal
ROAR: A QoS-oriented modeling framework for automated cloud resource allocation and optimization [50]	10.1016/j.jss.2015.08.006	Sí	Optimizar el coste de recursos de una app desplegada en el cloud cumpliendo los requisitos de rendimiento
Distributed mashups: a collaborative approach to data integration [52]	10.1108/IJWIS-04-2015-0018	No	Integración de diferentes fuentes de datos para trabajar con ellas de manera colaborativa
A Source Code Independent Reverse Engineering Tool for Dynamic Web Sites [15]	10.1109/CSMR.2005.4	No	Ingeniería inversa de aplicaciones web dinámicas
Modeling a Realistic Workload for Performance Testing [34]	10.1109/EDOC.2008.40	Sí	Modelado de la carga de trabajo para que esta represente de manera adecuada el comportamiento real de los usuarios
Selenium-Jupiter: A JUnit 5 extension for Selenium WebDriver [19]	10.1016/j.jss.2022.111298	No	Extensión de Selenium WebDriver a partir del framework JUnit 5 que facilita la escritura de tests
Spatio-Temporal Web Performance Prediction: Turning Bands Method and Sequential Gaussian Simulation [6]	10.1016/j.procs.2016.08.236	Sí	Predicción del rendimiento de una aplicación web a través de modelos estadísticos
Model-based load testing of web applications [55]	10.1080/02533839.2012.726028	Sí	Modelado de la carga de trabajo para que esta represente de manera adecuada el comportamiento real de los usuarios
Towards Automatic Performance and Scalability Testing of Rich Internet Applications in the Cloud [48]	10.1109/SEAA.2011.33	Sí	Retos a la hora de probar aplicaciones web de tipo RIA
Cloud-based load testing method for web services with VMs management [46]	10.1109/KBEI.2015.7436040	Sí	Modelo de pruebas de carga haciendo uso de la tecnología cloud (Testing as a Service)
Study And Improvement Of A Web Application Load Testing Model [61]	10.4028/www.scientific.net/AMM.321-324.2969	Sí	Modelado de la carga de trabajo para que esta represente de manera adecuada el comportamiento real de los usuarios
Realistic load testing of Web applications [14]	10.1109/CSMR.2006.43	Sí	Modelado de la carga de trabajo para que esta represente de manera adecuada el comportamiento real de los usuarios
LTF: A Model-based Load Testing Framework for Web Applications [60]	10.1109/QSIC.2014.53	Sí	Modelo para la caracterización y generación de la carga de trabajo de las pruebas
Black-box load testing to support auto-scaling web applications in the cloud [8]	10.1504/ijguc.2021.114823	Sí	Optimizar el coste de recursos de una app desplegada en el cloud cumpliendo los requisitos de rendimiento
Research on Web Application Load Testing Model [22]	10.1109/ITNEC.2017.8284961	Sí	Modelado de la carga de trabajo para que esta represente de manera adecuada el comportamiento real de los usuarios
Web based Testing - An Optimal Solution to Handle Peak Load [54]	10.1109/ICPRIME.2013.6496439	Sí	Metodología de las pruebas de carga
Software Testing for Web-Applications Non-Functional Requirements [43]	10.1109/ITNG.2009.209	Sí	Metodología de las pruebas de carga
Dominant Behavior Identification of Load Data [41]	10.1109/CCAA.2015.7148361	Sí	Impacto de la localización del usuario en el rendimiento de la aplicación
A Model-Based System to Automate Cloud Resource Allocation and Optimization [49]	10.1007/978-3-319-11653-2_2	Sí	Optimizar el coste de recursos de una app desplegada en el cloud cumpliendo los requisitos de rendimiento
Investigations on implementation of web applications with different techniques [33]	10.1049/iet-sen.2011.0136	No	Comparativa del rendimiento de dos aplicaciones desarrolladas con .NET y Java
Performance Testing for Web Based Application Architectures (.NET vs. Java EE) [21]	10.1109/NDT.2009.5272178	No	Comparativa del rendimiento de aplicaciones desarrolladas con .NET y con Java a lo largo del ciclo de vida del desarrollo
An Effective Automation Testing Framework for OATS Tool [42]	10.1007/978-81-322-2126-5_59	No	Extensión de la funcionalidad de una herramienta mediante un framework que permite su automatización
Comparative Study of N-Tier and Cloud-Based Web Application Using Automated Load Testing Tool [26]	10.1007/978-981-10-5508-9_23	No	Comparativa de dos tipos de arquitectura de aplicaciones web
Performance of a Java Web Application Running on Amazon EC2 Micro Instance [24]	10.15546/aeei-2013-0046	Sí	Pruebas de carga sobre una aplicación web desplegada en dos tipos de servidores en el cloud
A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications [40]	10.1109/AICAI.2019.8701327	Sí	Comparativa de múltiples herramientas de pruebas de carga

Tabla 4.4: Listado de artículos obtenidos a través de la cadena de búsqueda y resultados obtenidos tras la lectura en diagonal de los mismos

5: Evaluación del rendimiento de Crossroads 2.0

En este Capítulo se realiza una presentación del procedimiento de evaluación del rendimiento de la aplicación Crossroads 2.0. El Capítulo se centra en describir el entorno que se utilizará para realizar las pruebas, el diseño y ejecución de cada escenario de prueba, el análisis de los resultados obtenidos y las conclusiones que se pueden extraer de dicho análisis, lo que permitirá plantear una propuesta que permita mejorar el rendimiento de la aplicación.

5.1. Selección de herramienta

El primer paso que hay que llevar a cabo antes de poder comenzar con la evaluación del rendimiento de la aplicación consiste en seleccionar la herramienta mediante la cual se ejecutarán las pruebas de carga de la aplicación. Para ello, se ha elaborado un cuadro de criterios de selección y requisitos que se busca que cumpla la herramienta para ser seleccionable. En la Tabla 5.5 se muestra el listado que se ha elaborado para esta tarea. Para cada criterio se ha indicado el peso que se le da al mismo a la hora de calcular la puntuación de una herramienta, de tal forma que la herramienta con mayor puntuación será aquella que sea seleccionada finalmente. A continuación, se muestra una pequeña referencia de aquellos aspectos de una herramienta que se valoran con cada requisito o criterio:

- **C1**. Con este criterio se valora que la aplicación disponga de un plan gratuito con la duración y características suficientes para poder implementar los escenarios de prueba y poder ejecutarlos.
- **C2**. Este requisito valora la posibilidad de realizar una propuesta para extender la funcionalidad del software empleado en la realización de las pruebas de carga.
- **C3**. Se valora que el lenguaje de programación empleado para codificar los escenarios de prueba sea conocido por el desarrollador. También se valora el caso de

aquellas herramientas en las que no sea necesario hacer uso de ningún lenguaje de programación.

- **C4**. Se valora la existencia de gran cantidad de tutoriales y ejemplos accesibles a través de Internet. También se valora que la herramienta posea un manual o página web con su documentación.
- **C5**. Se valora la existencia de reseñas en Internet que reflejen que el uso de la herramienta es habitual en el sector.
- **C6**. Se valora que el software sea de tipo SaaS o que la generación de la carga de trabajo (o usuarios virtuales) se lleve a cabo a través de máquinas virtuales desplegadas en un proveedor de cloud. Con cualquiera de estas dos características se elimina la limitación de recursos de hardware empleados en la ejecución de las pruebas, por lo que se consigue acelerar dicho proceso.
- **C7**. Se valora que se haya encontrado alguna referencia en toda la documentación consultada a alguna característica del software que facilite la realización de las pruebas de carga de las aplicaciones web colaborativas. Como este criterio está relacionado con la característica principal de la aplicación que se quiere probar, se le ha asignado el mayor peso.

ID	Criterio/requisito	Peso
C1	Versión gratuita (o plan gratuito)	3
C2	Es open source	4
C3	Lenguaje de programación conocido	3
C4	Existencia de amplia documentación y ejemplos	2
C5	Herramienta de uso extendido en el ámbito de las pruebas de carga	2
C6	Posibilidad de ejecución en el cloud	2
C7	Presenta característica que facilita las pruebas de aplicaciones colaborativas	5

Tabla 5.5: Listado de requisitos y criterios para la selección de la herramienta con la que realizar las pruebas de Crossroads 2.0

Tras evaluar cada una de las herramientas frente a la lista de criterios anterior, se han obtenido las puntuaciones que aparecen en le Tabla 5.6. Como se puede observar, hay un empate entre las herramientas **Artillery.io**, **Flood Element**, **JMeter**, **K6**, **Locust** y **Taurus**. Como para ninguna de estas herramientas se ha certificado el cumplimiento del criterio **C7**, se ha decidido seleccionar la herramienta **JMeter**, pues esta herramienta es conocida por el desarrollador y es la herramienta mencionada en la única respuesta a la encuesta a profesionales del sector. Esta herramienta se ha seleccionado a pesar de lo indicado en [40] pues, aunque Locust tiene mejor rendimiento que JMeter, ha pesado más la experiencia previa en el manejo de JMeter por parte del desarrollador.

Nombre herramienta	**Puntuación final**
Artillery.io	14
Bees with Machine Guns	12
BlazeMeter	12
Boomq.io	5
CloudTest from Akamai	10
Eggplant	2
Flood Element	14
Fortio	10
Gatling	11
HeadSpin	5
JMeter	14
K6	14
LoadFocus	7
LoadNinja	9
LoadRunner	9
Loadster	7
LoadView	12
Locust	14
Neotys Neoload	10
nGrinder	7
Parasoft Load Test	7
puppeteer-webperf	10
Rational Performance Tester	7
Siege	12
Taurus	14
The Grinder	9
Tsung	14
WAPT	8
WebLOAD	7

Tabla 5.6: Puntuaciones obtenidas por cada herramienta tras evaluar los criterios de selección sobre cada una de ellas

5.2. Entorno de pruebas

Antes de poder analizar el rendimiento de la aplicación, es necesario preparar el entorno de pruebas. Este entorno está conformado por dos equipos diferentes, el servidor o SUT (*System Under Test*), que es el sistema en el que está desplegada la aplicación que se va a probar y el cliente, que será la máquina que ejecutará la herramienta de pruebas de carga y que generará todos los usuarios concurrentes. Las características técnicas de cada equipo se presentan a continuación:

- **Servidor**. Este equipo es una máquina virtual con Debian 10 como sistema operativo. Además, dispone de una CPU con 4 cores, una memoria RAM de 16 GB y 30GB de espacio libre en disco, sobre 50GB del total de espacio en el disco. El servidor se encuentra disponible en la siguiente dirección IP: 157.88.62.117.
- **Cliente**. La máquina cliente es un ordenador HP Pavilion con Ubuntu 20.04 como sistema operativo. La CPU dispone de 4 cores y el equipo tiene 8GB de memoria RAM y un disco SSD de 512GB de espacio, de los cuales 385GB están libres.

5.3. Diseño de la prueba

En esta Sección se presentan todos los aspectos relativos al diseño de las pruebas de carga que se van a realizar. Estos aspectos abarcan la descripción de los casos de uso o escenarios que se van a probar, la carga de trabajo o número de usuarios concurrentes que se van a emplear en cada prueba y las métricas que se van a recopilar durante la ejecución de las mismas.

Descripción de los escenarios de prueba

Se ha decidido seleccionar un único caso de uso, que será el que se evalúe con las pruebas. En este caso de uso, los jugadores accederán a una sala de partida de una única ronda con el código de una sala previamente creada, seleccionarán un grupo para unirse al él y seleccionarán un rol de manera aleatoria. Una vez todos los jugadores han realizado esta tarea, el moderador dará comienzo a la partida y todos los jugadores responderán las preguntas del formulario, de tal forma que no se generarán conflictos entre las respuestas de todos los jugadores de cada grupo. Para cada pregunta, cada jugador realizará una consulta de una de las pistas asociadas. Una vez completado el formulario por todos los jugadores del grupo, estos visualizarán los resultados de ronda y, tras la finalización de la partida por parte del moderador, visualizarán los resultados de la ronda.

Partiendo del caso de uso indicado anteriormente, se realizarán pruebas de red cerrada y pruebas de red abierta sobre el mismo escenario. Según las técnicas de análisis operacional [36], que se sirven para estimar el rendimiento de un sistema informático, las redes cerradas son sistemas que tienen siempre el mismo número de trabajos en ejecución, mientras que en las redes abiertas el número de trabajos en ejecución varía con el tiempo. De esta forma, la red abierta simula mejor el sistema que se quiere probar, pues la carga de trabajo del SUT en un momento dado no es siempre la misma.

Como JMeter solo simula pruebas en sistemas de red cerrada, hay que aplicar algunas modificaciones al plan de pruebas para poder simular una red abierta. Según lo descrito en [7], esta tarea puede realizarse aumentando considerablemente el tiempo de pensar del usuario [35], que es el tiempo que pasa entre que un usuario realiza una interacción con el sistema y la realización de la siguiente.

Métricas

Las métricas de rendimiento que se recopilarán durante las pruebas son las que aparecen en el siguiente listado:

- **Tiempo de respuesta**. Métrica de velocidad que se define como el intervalo de tiempo, en milisegundos, entre el final del envío de una petición y el final de la respuesta correspondiente del sistema.
- **Productividad**. Métrica de velocidad que se calcula como el número de peticiones servidas por segundo.
- **Utilización**. Métrica que se define como el porcentaje de tiempo en el que un componente del servidor está ocupado con respecto al período de observación del sistema. Se calculará la utilización de la CPU y la memoria RAM.
- **Porcentaje de error**. Se calculará el número medio de peticiones respondidas de forma incorrecta y el número medio de peticiones a las que no se ha dado servicio. Además, se clasificarán los errores producidos según su tipo y se determinará la proporción de cada tipo de error sobre el total.

Especificación de la carga de trabajo

La carga de trabajo constará de todas las peticiones HTTP grabadas con JMeter durante la grabación del escenario de prueba (Sección 5.3). Además, se han definido tres tramos de intensidades de carga, en cada uno de los cuales el número de usuarios concurrentes será distinto. Esta decisión se ha tomado con el objetivo de acotar mejor la capacidad de la aplicación en cuanto a niveles de concurrencia se refiere. El número de usuarios empleado en cada tramo será, respectivamente, 30, 50 y 100 usuarios. Además, para aumentar la fiabilidad de los datos recopilados, se realizarán tres ejecuciones con cada uno de los tramos. Otras características relevantes de la carga de trabajo se resumen a continuación:

- **Período de subida** [16]. El período de subida es el tiempo que se tarda en desplegar todos los usuarios virtuales del plan de pruebas. Se utilizará un período de subida de 2 minutos.
- **Tiempo de pensar** [35]. El tiempo de pensar es el período de tiempo que pasa entre dos interacciones del usuario con el sistema. Se utilizará un valor aleatorio de una distribución normal. El valor máximo es de 3 segundos en el caso de las pruebas de red cerrada, mientras que en el caso de las pruebas de red abierta el valor máximo es de 8 segundos.

5.4. Plan de pruebas

Teniendo en cuenta todos los factores descritos en el diseño de la prueba, se ha elaborado un plan de pruebas, parametrizado de tal forma que, mediante el uso de un único plan de pruebas, se permita reflejar todas las combinaciones posibles de la carga de trabajo.

Debido a la extensión de este capítulo, el detalle del plan de pruebas se describe en el Apéndice B.

5.5. Ejecución de las pruebas

Teniendo en cuenta los distintos números de usuarios concurrentes que se van a emplear en las pruebas, que se van a realizar pruebas de red abierta y red cerrada con cada número de usuarios y que para cada combinación de las anteriores se van a realizar tres ejecuciones para aumentar la fiabilidad de las métricas, se obtienen, en total, 18 ejecuciones del plan de pruebas.

Estas 18 ejecuciones se han realizado en tres días distintos, es decir, 6 ejecuciones por día, todas ellas con un número de usuarios fijo. De estas 6 ejecuciones, 3 de ellas se dedicaban a pruebas de red cerrada y las otras 3 a pruebas de red abierta. Las pruebas se han hecho siempre en 3 momentos distintos del día para cada tipo de red, con el objetivo de aumentar la independencia del momento del día de los datos recogidos.

Todas las ejecuciones se han llevado a cabo sin ninguna incidencia, por lo que, a partir de este punto, se da comienzo a la fase de análisis de los resultados obtenidos.

5.6. Análisis de los resultados

Tras la finalización de todas las ejecuciones del plan de pruebas y la recopilación de todos los ficheros con los resultados de las pruebas, se ha generado con JMeter un conjunto de informes en HTML [28] que presentan un resumen de cada fichero de resultados de una manera más legible para el usuario. A partir de esos informes se han obtenido las Tablas 5.7, 5.9, 5.11 y 5.12. Todas estas tablas serán comentadas a continuación.

En primer lugar hay que analizar los resultados de las pruebas en cuanto a tiempo de respuesta y productividad. En las Tablas de los resultados en bruto de las pruebas de red abierta y red cerrada (Tablas 5.7, 5.9) se muestran el número de hilos o número de usuarios concurrentes utilizado para la prueba, el número de peticiones HTTP que se han enviado en cada una de las tres ejecuciones del plan de pruebas y un resumen estadísitico para el tiempo de respuesta y la productividad. Este resumen incluye métricas como la media, la mediana o el percentil 95, entre otros. A partir de estas tablas se han obtenido las Tablas 5.8 y 5.10 que presentan la media de las tres ejecuciones para los valores medios del tiempo de respuesta y la productividad.

Como se puede observar en la Tabla 5.8, el tiempo de respuesta medio va subiendo según sube el número de usuarios utilizados para la prueba. En el caso de los 50 hilos y los 100 hilos este tiempo de respuesta toma valores poco aceptables, pues en el primer caso se acerca al segundo de tiempo de respuesta, mientras que en el segundo este valor asciende a casi 4 segundos. En cuanto a los valores máximos tomados por esta métrica (Tabla 5.7), en la gran mayoría de ejecuciones se supera ampliamente el minuto de tiempo de respuesta, lo cual es completamente inaceptable en una aplicación web, pues perjudica claramente la experiencia de usuario y la fluidez de la aplicación. Con respecto a la productividad, se puede apreciar que el máximo valor medio de las tres ejecuciones se obtiene en las pruebas con 50 usuarios (valor 7,19) contrariamente a lo que cabría esperar, es decir, que el máximo valor se obtuviese en las ejecuciones con 30 hilos. Esto puede deberse a un diferente nivel de carga en el servidor dependiente del día y el momento en el que se han ejecutado esas pruebas, por lo que no hay que darle una mayor importancia. Finalmente, en el caso de 100 usuarios concurrentes la productividad no llega a 4 transacciones/segundo por lo que se vuelve a comprobar que la aplicación no alcanza el requisito de concurrencia marcado inicialmente.

Si comparamos estos resultados con los obtenidos en las pruebas de red cerrada (Tabla 5.10), podemos observar la misma tendencia que en las pruebas de red abierta. Si comparamos los valores individuales obtenidos por cada número de hilos, podemos comprobar que la productividad es mayor que en las pruebas de red abierta, al igual que el tiempo medio de respuesta, que también es superior. Este aumento tanto de la productividad como del tiempo medio de respuesta se debe a la influencia del tiempo de pensar empleado en ambos tipos de pruebas. Como el tiempo de pensar máximo empleado en las pruebas de red cerrada es menor que en en el caso de red abierta, se lanzan y atienden más peticiones cada segundo (aumento de la productividad) lo cual comienza a saturar el servidor de la aplicación lo que conlleva que se tarde más en atender cada petición (aumento del tiempo de respuesta). En cualquier caso, los resultados obtenidos en las pruebas de red cerrada (sobre todo el tiempo de respuesta) son mejorables.

En las Figuras 5.70, 5.71, 5.72 y 5.73 se muestran de manera gráfica los resultados presentados en las Tablas 5.8 y 5.10, respectivamente.

Una vez analizados los resultados relativos al tiempo de respuesta y a la productividad es momento de analizar las métricas de error recopiladas. En las Tablas 5.11 y 5.12 se recogen dichas métricas para las pruebas de red abierta y cerrada. En cada tabla se muestran los tipos de error que han acontencido, el porcentaje medio (entre las tres ejecuciones de cada número de hilos) de peticiones sobre el total que han terminado en error, el nombre de la petición que ha tenido un mayor número de errores de cada tipo entre las tres ejecuciones y el número medio de ocurrencias de este tipo de error para esta petición en concreto.

El tipo de error que más veces se ha producido es el error 500 [37], con una proporción de errores de este tipo sobre el total de peticiones enviadas cercana al 14 %. Este código

de error indica que el fallo se ha producido en el servidor. La aparición de este tipo de errores se debe a que el servidor no ha podido atender algunas de las peticiones enviadas o a que la petición que termina en error depende del resultado de otra petición que no ha podido completarse. Un caso relevante de esta última causa es la petición de nombre **/api/player/update**, a través de la cual se actualiza el estado de los jugadores con el grupo y el rol seleccionados, pues si falla esta petición no se pueden completar otras peticiones que hagan uso de esta información, como puede ser el caso de la consulta de las pistas de cada pregunta.

Continuando con el análisis de las métricas de error es necesario comentar la petición con mayor número de errores de tipo 500, pues en todos los casos esta siempre es la petición **/api/form/status**. Esta petición se realiza desde el front-end de la aplicación con el objetivo de recuperar la información necesaria para comprobar el estado del formulario del resto de jugadores del grupo y es una petición que requiere de bastantes consultas a la base de datos. Esta información se obtiene de manera periódica, por lo que la presencia de tantos errores puede revelar que esta operación constituye un cuello de botella importante en la aplicación.

Para finalizar el análisis de los resultados se va a realizar un comentario acerca de la utilización de los principales recursos del servidor: la CPU y la memoria RAM.

En las Figuras 5.74 a 5.81 se muestran gráficas de utilización de la CPU durante diferentes ejecuciones del plan de pruebas, tanto para red abierta, como para red cerrada. En todas estas gráficas se puede apreciar una gran fluctuación de los valores de utilización de este recurso, encontrándose picos de utilización en los que se pasa de valores cercanos a 0 a valores próximos a 100 en muy poco tiempo. Todos estos picos parecen estar relacionados con los problemas de rendimiento de la aplicación comentados anteriormente, pues reflejan la saturación de este recurso durante la ejecución de las pruebas.

En cuanto a las diferencias entre todas estas gráficas, no parece haber grandes diferencias entre las gráficas de red abierta y red cerrada, habiendo algún pico más en las gráficas de red cerrada. Sin embargo, sí que hay diferencias notables entre las gráficas de las distintas ejecuciones con 50 hilos, notándose una mayor utilización en la CPU en la ejecución número 3 y una menor utilización en las gráficas de las ejecuciones 1 y 2. Esta circunstancia puede deberse a una mayor o menor carga de trabajo en el servidor dependiendo del momento del día.

En las Figuras 5.82 a 5.89 se muestra la contraparte de las gráficas anteriores para la utilización de la memoria RAM. Como se puede observar en las mismas, existe fluctuación en los valores de utilización de este recurso, al igual que pasa con la CPU. Sin embargo, la fluctuación de valores no es tan pronunciada como en el caso de la CPU, pues esta siempre se produce entre los valores de 20 % y 30 % de utilización. Esto parece indicar que este recurso no constituye un problema para la ejecución de la aplicación en el servidor, incluso en el caso de 100 usuarios, por lo que no debe preocuparnos.

		Tiempo de respuesta (ms)					Productividad
Nº hilos	**Nº peticiones**	**Media**	**Min**	**Max**	**Mediana**	**95th pct**	**Transacciones/s**
30	14345	256,39	10,00	63506,00	25,00	215,70	6,31
	14345	396,50	8,00	955768,00	22,00	146,70	4,50
	14345	232,25	8,00	56526,00	22,00	145,70	6,29
50	23905	972,22	9,00	955878,00	25,00	682,90	4,48
	23905	631,05	8,00	947766,00	22,00	718,00	7,20
	23905	487,28	3,00	66209,00	22,00	752,00	9,90
100	47780	3037,32	4,00	7640514,00	1323,50	29013,40	5,29
	47780	4460,33	9,00	13566543,00	719,00	24633,90	3,13
	47780	3964,28	8,00	23393265,00	883,00	27337,45	1,77

Tabla 5.7: Resultados de las ejecuciones del plan de pruebas configurado para realizar pruebas de red abierta

Nº hilos	**Productividad media (trans/s)**	**Tiempo medio de respuesta (ms)**
30	5,7	295,05
50	7,19	696,85
100	3,40	3820,64

Tabla 5.8: Valores medios de las tres ejecuciones realizadas por cada número de hilos en las pruebas de red abierta. Datos obtenidos de la Tabla 5.7

		Tiempo de respuesta (ms)					Productividad
Nº hilos	**Nº peticiones**	**Media**	**Min**	**Max**	**Mediana**	**95th pct**	**Transacciones/s**
30	14345	332,01	9,00	71225,00	25,00	398,00	13,75
	14345	331,01	7,00	67225,00	22,00	321,70	13,60
	14345	334,77	8,00	66526,00	22,00	313,40	13,39
50	23905	1462,4	8,00	7938776,00	38,00	3783,95	2,74
	23905	854,84	7,00	69178,00	42,00	4078,50	17,77
	23905	919,81	7,00	67067,00	42,00	4468,95	17,22
100	47780	4757,61	7,00	3419040,00	3031,00	28631,70	8,44
	47780	5911,67	9,00	5178036,00	1662,50	17595,05	7,06
	47780	6246,96	9,00	13779413,00	2524,50	26755,75	3,09

Tabla 5.9: Resultados de las ejecuciones del plan de pruebas configurado para realizar pruebas de red cerrada

Nº hilos	**Productividad media (trans/s)**	**Tiempo medio de respuesta (ms)**
30	13,58	332,60
50	12,58	1079,02
100	6,20	5638,75

Tabla 5.10: Valores medios de las tres ejecuciones realizadas por cada número de hilos en las pruebas de red cerrada. Datos obtenidos de la Tabla 5.9

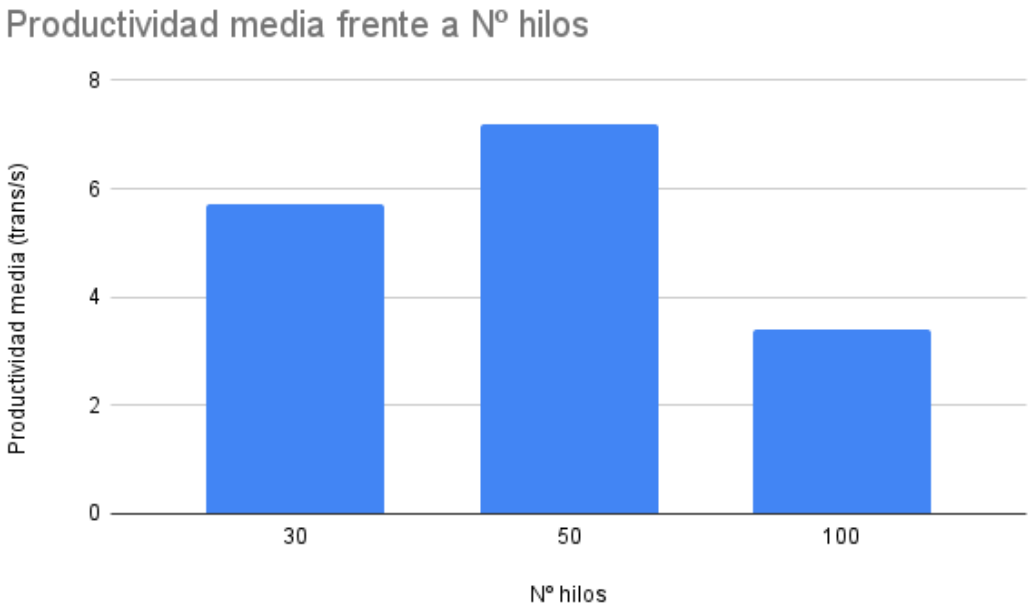

Figura 5.70: Productividad media de las ejecuciones con distinto número de hilos en pruebas de red abierta. Datos obtenidos de la Tabla 5.8

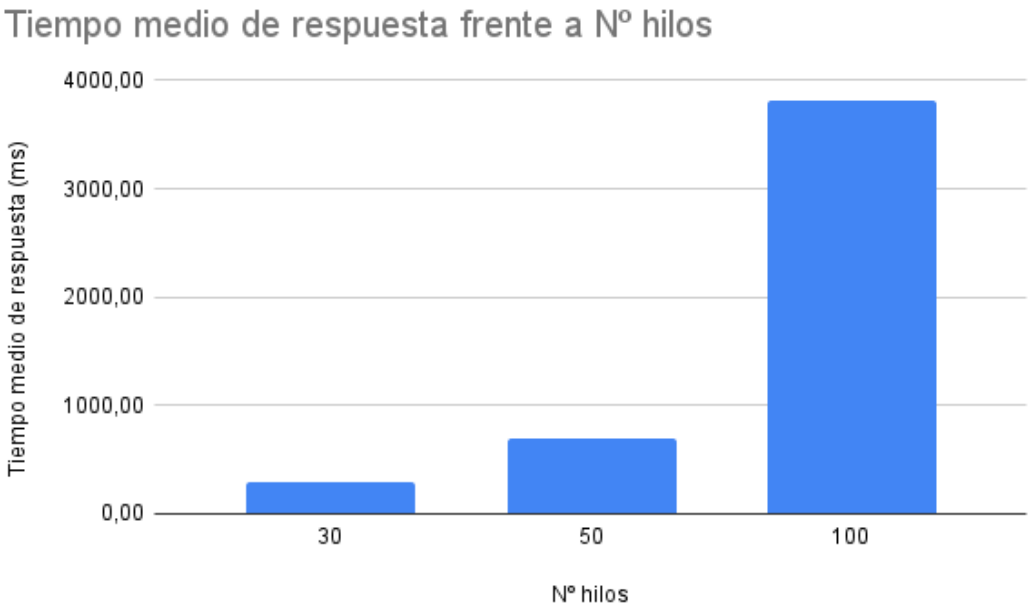

Figura 5.71: Tiempos medios de respuesta de las ejecuciones con distinto número de hilos en pruebas de red abierta. Datos obtenidos de la Tabla 5.8

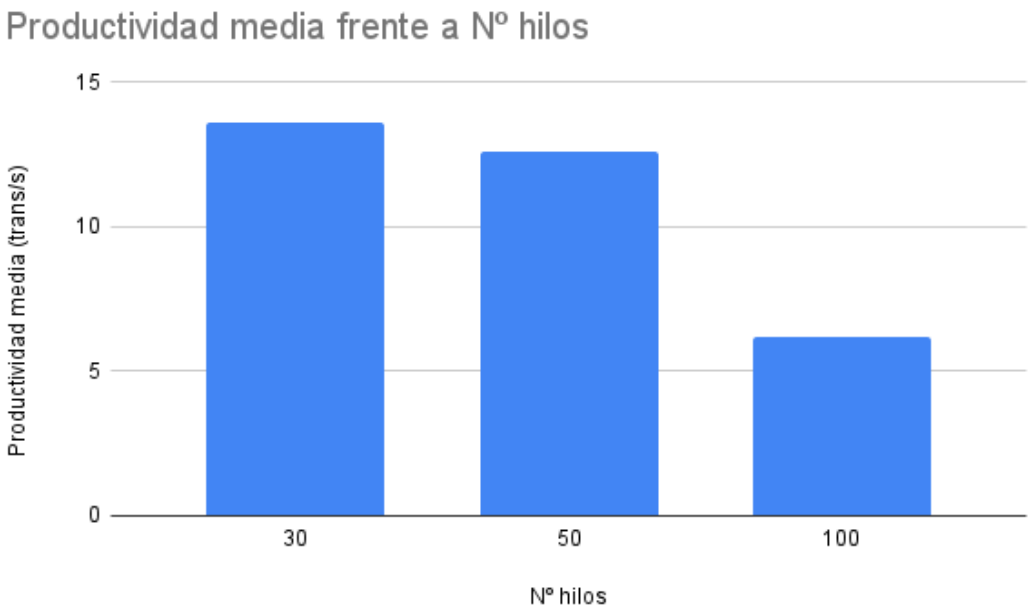

Figura 5.72: Productividad media de las ejecuciones con distinto número de hilos en pruebas de red cerrada. Datos obtenidos de la Tabla 5.10

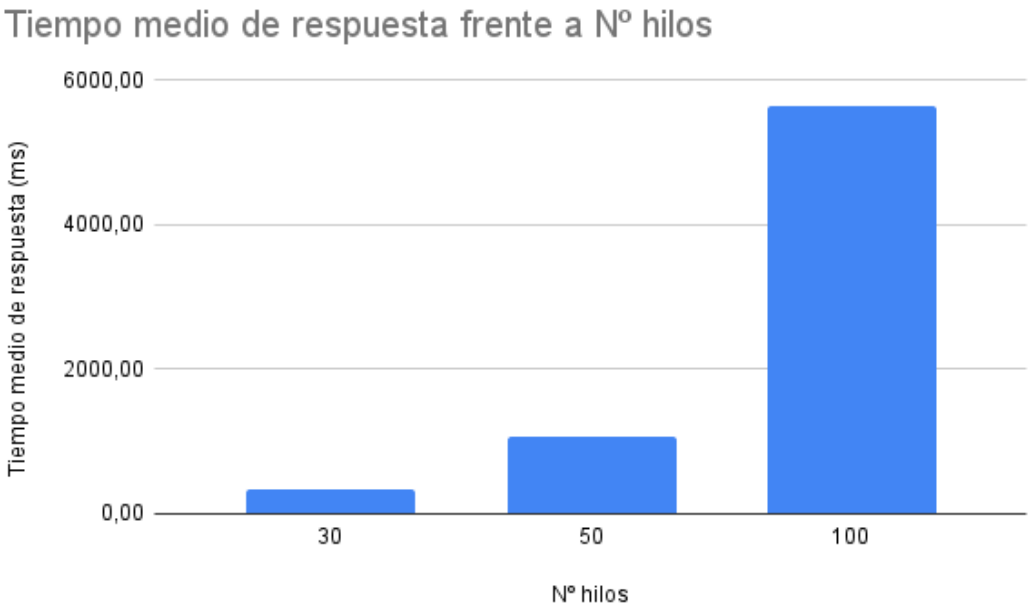

Figura 5.73: Tiempos medios de respuesta de las ejecuciones con distinto número de hilos en pruebas de red cerrada. Datos obtenidos de la Tabla 5.10

Nº hilos	Tipo de error	% medio sobre el total	Petición con mayor nº de errores	
			Nombre	Nº medio de ocurrencias
30	500	13,78 %	/api/form/status	1917
	Expiró el tiempo de conexión	0,003 %	/api/chat/get-chat y /api/room/rounds/68046&1	0,33
50	500	13,91 %	/api/form/status	3194
	Expiró el tiempo de conexión	0,023 %	/api/chat/get-chat	3,00
100	500	14,59 %	/api/form/status	6332,67
	Expiró el tiempo de conexión	0,003 %	/api/chat/get-chat	1,33
	Socket closed	0,003 %	/api/chat/get-chat y /api/room/rounds/68046&1	1,33

Tabla 5.11: Métricas de error obtenidas durante las distintas ejecuciones del plan de pruebas en las pruebas de red abierta

Nº hilos	Tipo de error	% medio sobre el total	Petición con mayor nº de errores	
			Nombre	Nº medio de ocurrencias
30	500	13,97 %	/api/form/status	1917
50	500	14,13 %	/api/form/status	3185
	Socket closed	0,003 %	/api/form/choice	0,667
100	500	14,56 %	/api/form/status	5966,333
	Socket closed	0,03 %	/api/form/status	3,333

Tabla 5.12: Métricas de error obtenidas durante las distintas ejecuciones del plan de pruebas en las pruebas de red cerrada

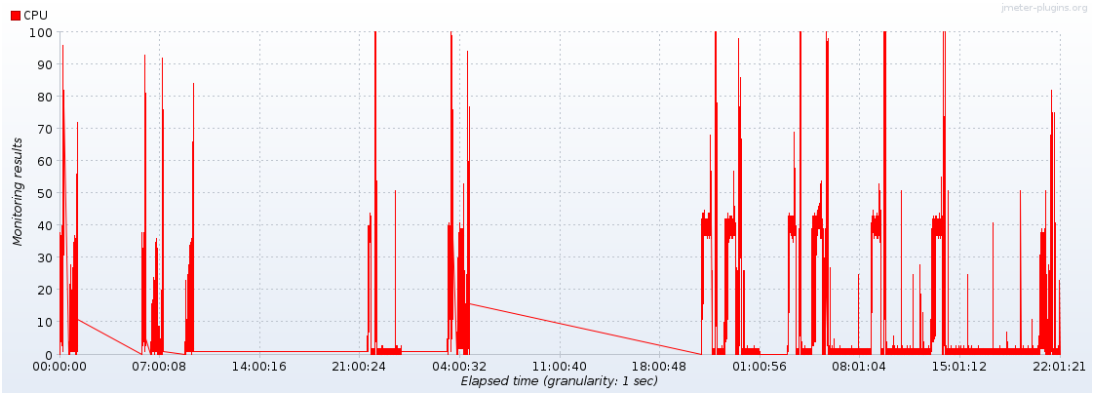

Figura 5.74: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 30 hilos en una prueba de red abierta

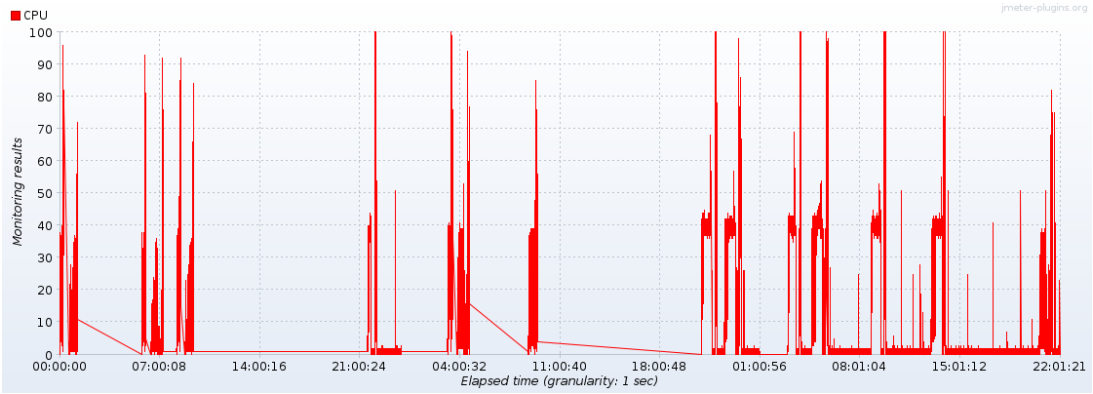

Figura 5.75: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 50 hilos en una prueba de red abierta

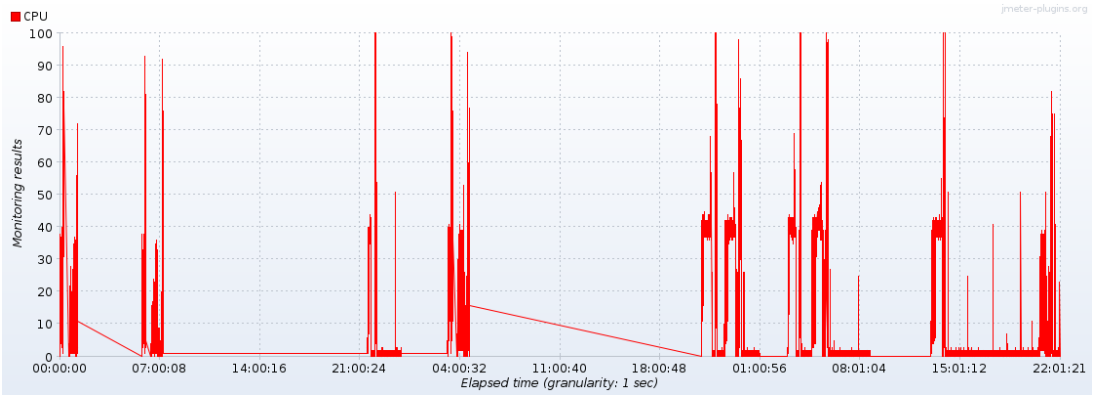

Figura 5.76: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 100 hilos en una prueba de red abierta

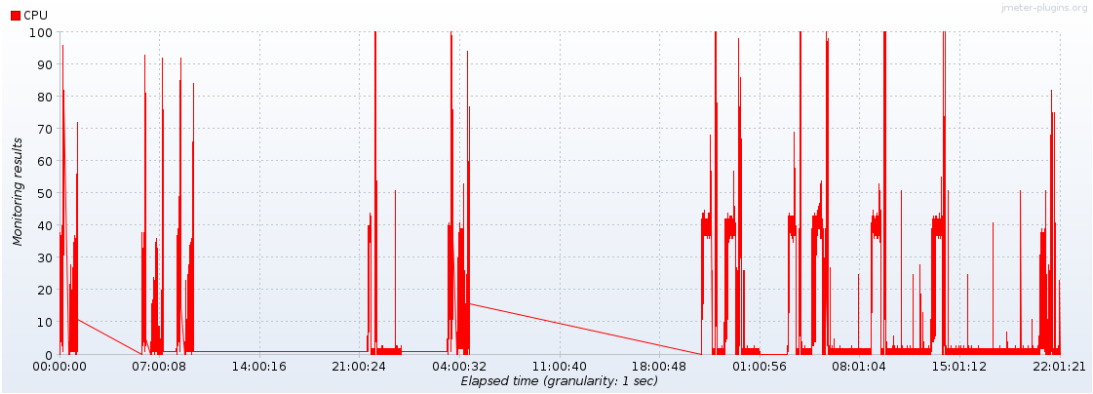

Figura 5.77: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 30 hilos en una prueba de red cerrada

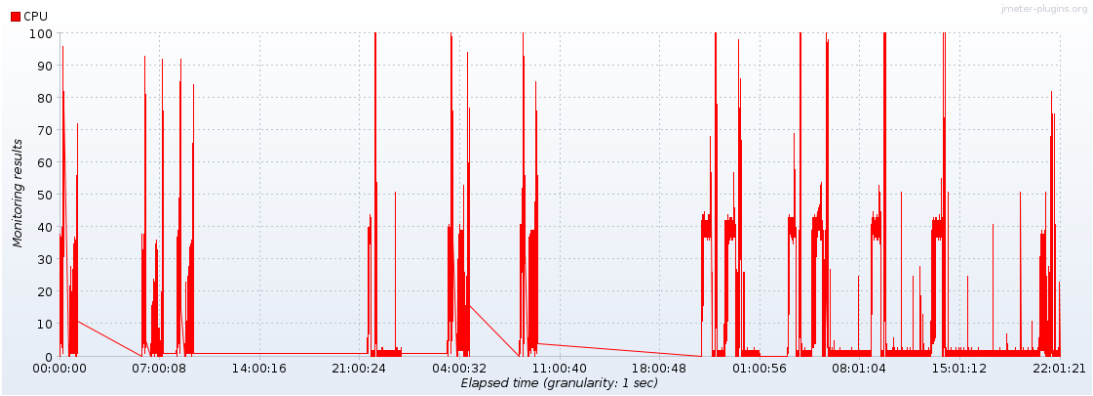

Figura 5.78: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

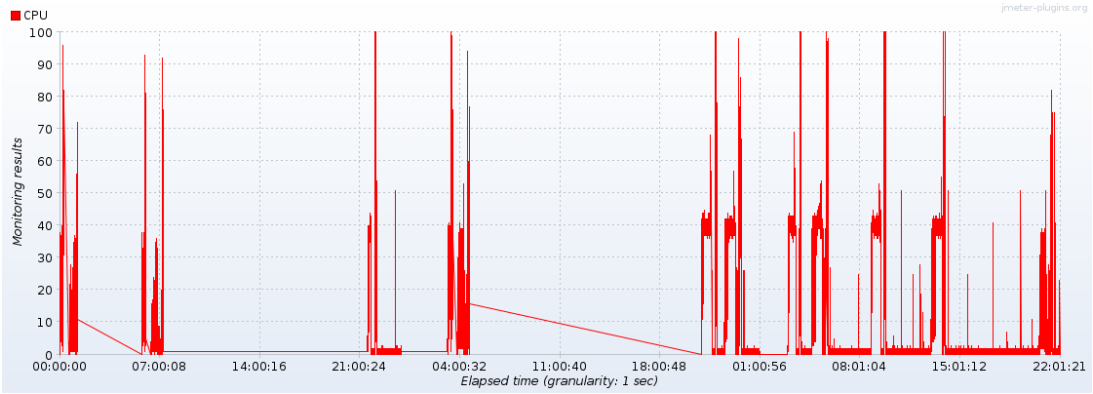

Figura 5.79: Utilización de la CPU durante la tercera ejecución del plan de pruebas con 100 hilos en una prueba de red cerrada

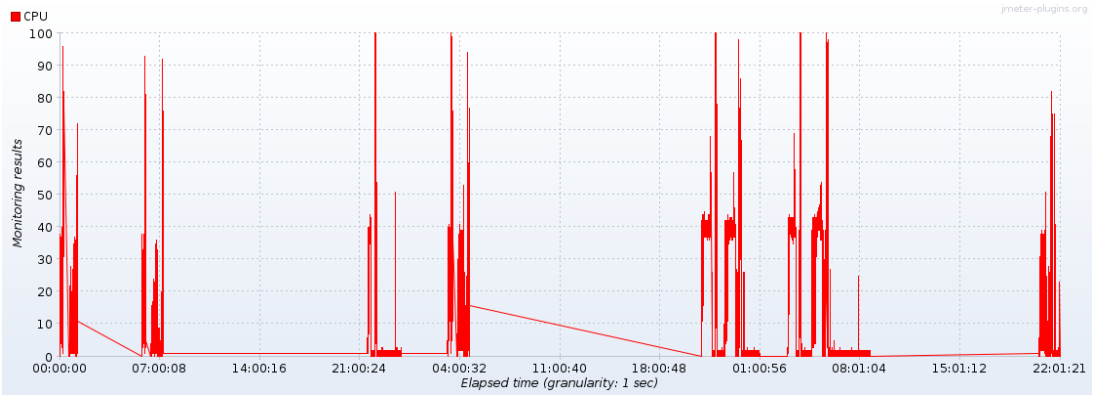

Figura 5.80: Utilización de la CPU durante la segunda ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

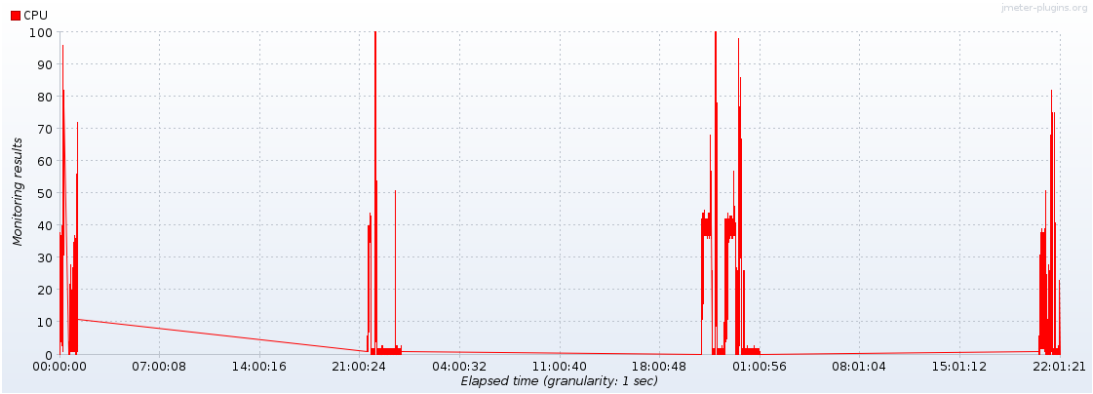

Figura 5.81: Utilización de la CPU durante la primera ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

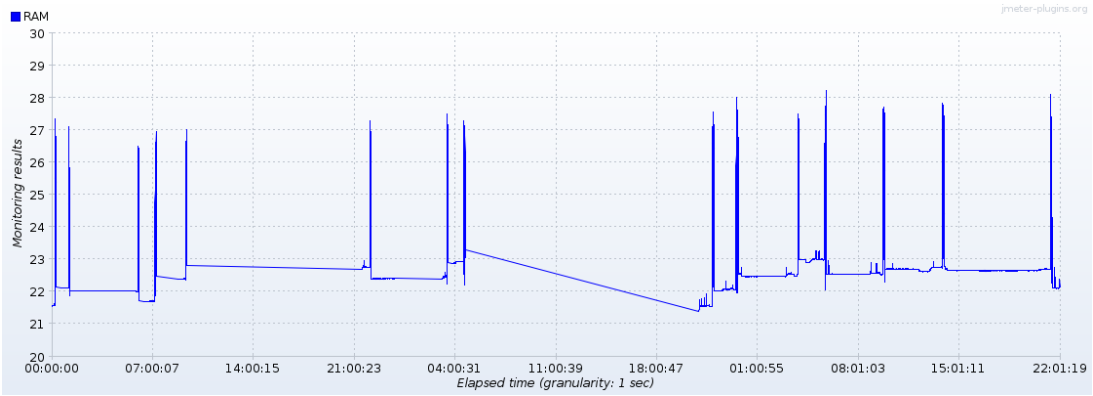

Figura 5.82: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 30 hilos en una prueba de red abierta

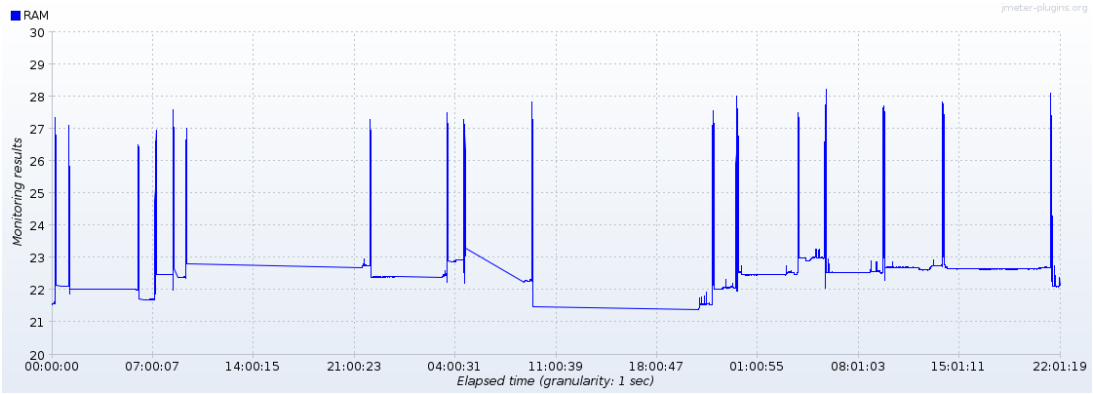

Figura 5.83: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 50 hilos en una prueba de red abierta

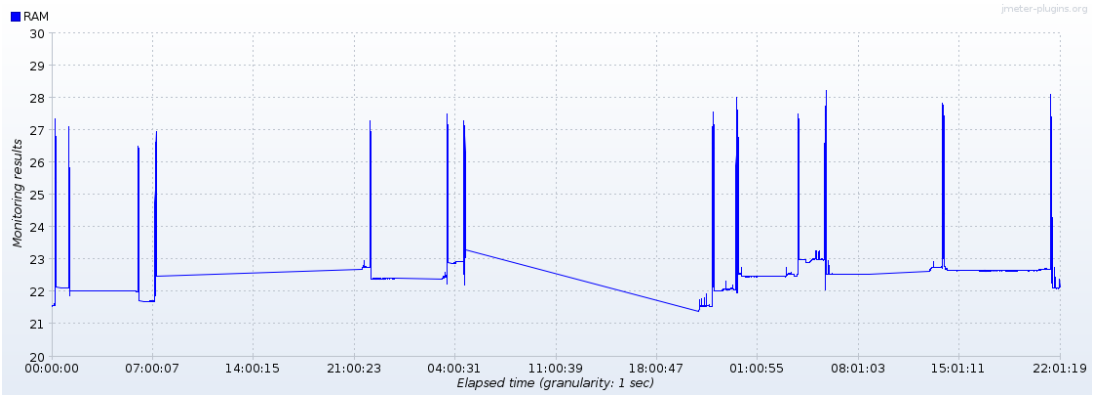

Figura 5.84: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 100 hilos en una prueba de red abierta

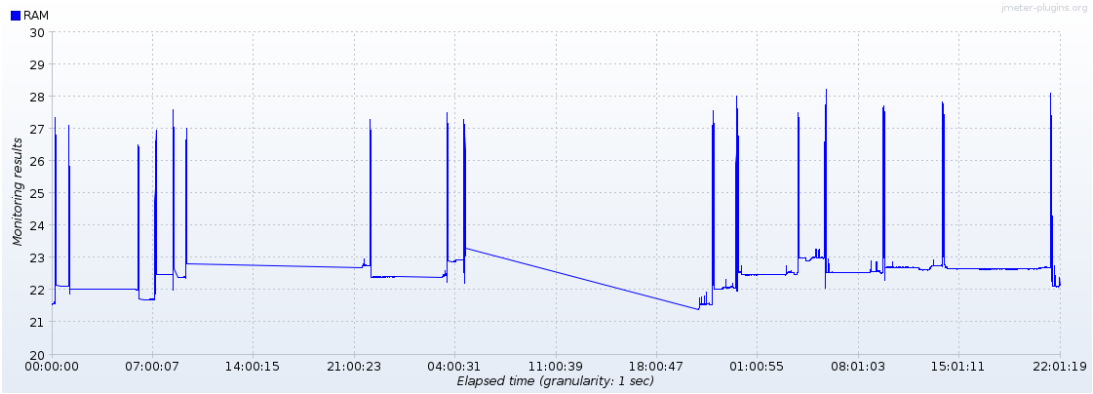

Figura 5.85: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 30 hilos en una prueba de red cerrada

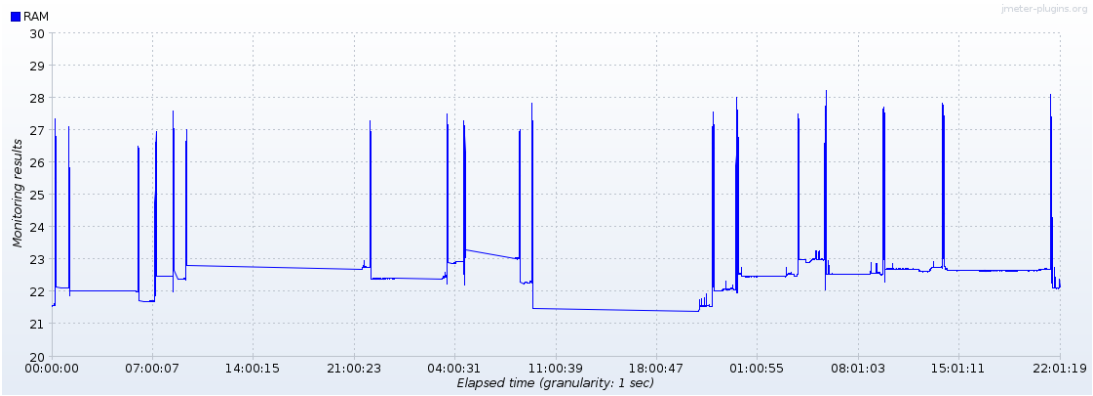

Figura 5.86: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

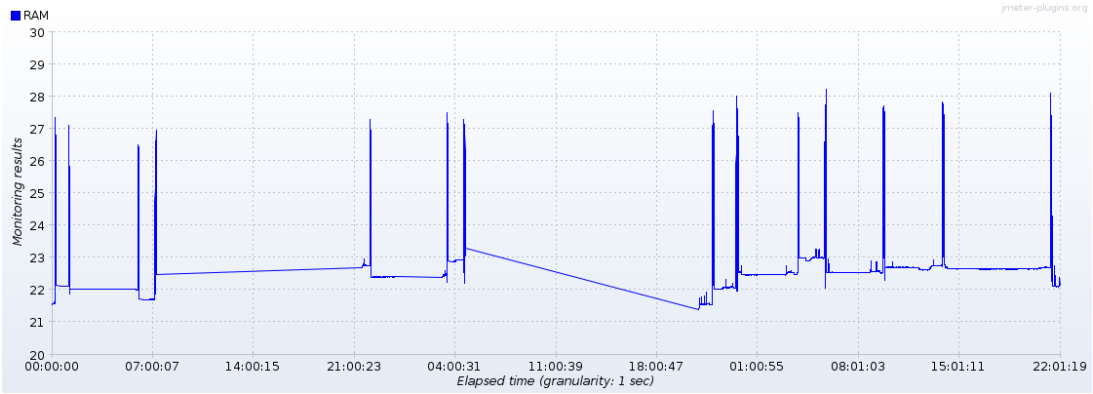

Figura 5.87: Utilización de la memoria RAM durante la tercera ejecución del plan de pruebas con 100 hilos en una prueba de red cerrada

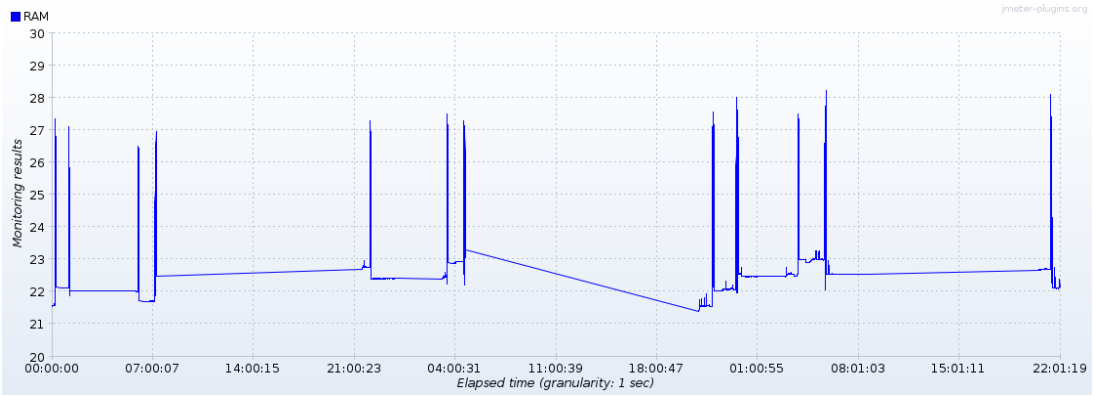

Figura 5.88: Utilización de la memoria RAM durante la segunda ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

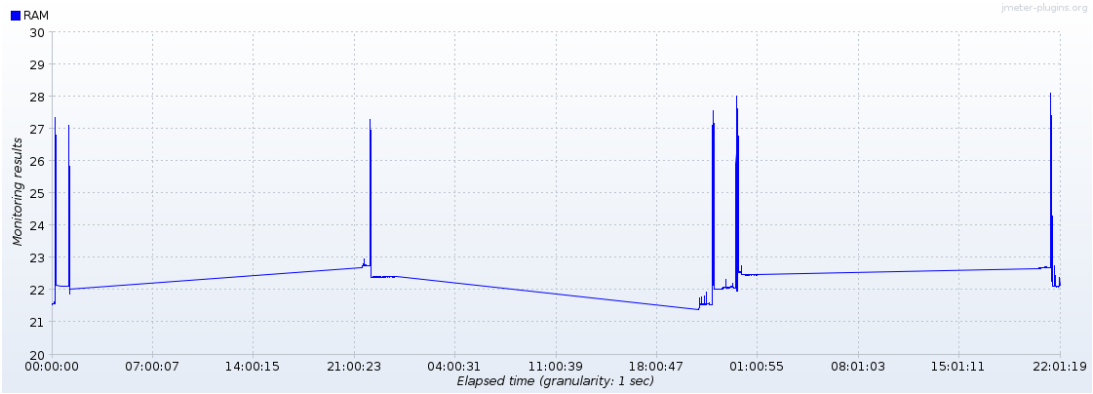

Figura 5.89: Utilización de la memoria RAM durante la primera ejecución del plan de pruebas con 50 hilos en una prueba de red cerrada

5.7. Conclusiones y propuestas de mejora

Tras la finalización del análisis de los resultados se ha elaborado un listado con las principales conclusiones que se pueden extraer del mismo, las cuales serán presentadas a continuación:

- El rendimiento de la aplicación no alcanza el requisito marcado inicialmente de soportar 100 usuarios concurrentes en la misma partida, como indican las métricas recopiladas y la presencia de errores en algunas peticiones.
- Las partes de la aplicación que parecen tener mayores problemas de rendimiento son la pantalla de resolución del formulario y la pantalla de comprobación de conflictos, pues en ellas se hace uso de manera muy frecuente de la operación **/api/form/status**, que constituye un cuello de botella de la aplicación.
- Este uso de algunas operaciones del back-end de manera muy frecuente se realiza para dotar a la aplicación de una sensación de actualización en tiempo real ya que, por la arquitectura cliente-servidor de la aplicación, la única manera de notificar a un cliente el estado en el que se encuentran el resto de clientes es realizando consultas al servidor con cierta periodicidad. Como algunas actualizaciones de este estado de los clientes deben notificarse de manera casi inmediata por los requisitos de la aplicación, se puede concluir que la arquitectura condiciona gravemente el rendimiento de la aplicación en algunos puntos.
- Por tanto, el mecanismo implementado para realizar las actualizaciones en tiempo real convierte en potenciales cuellos de botella a algunas operaciones del back-end que se invocan desde la sala de monitorización del moderador como, por ejemplo, la consulta del estado de la partida. También se convierten en cuellos de botella en potencia la recuperación de mensajes del chat y, en definitiva, cualquier otra operación del back-end que se invoque de manera muy frecuente desde el front-end.

- En cuanto a los recursos hardware que utiliza el servidor de la aplicación, el cuello de botella reside en la CPU, mientras que la memoria RAM no presenta este tipo de problemas, pudiéndose incrementar el uso de este recurso sin ver afectado el rendimiento de la aplicación.

Teniendo en cuenta las conclusiones anteriores, la propuesta de mejora de la aplicación pasa por adaptar la arquitectura de la aplicación en aquellos puntos en los que el modelo cliente-servidor no es adecuado para las características de actualización en tiempo real de la aplicación. Esta adaptación pasa por combinar la arquitectura cliente-servidor con una arquitectura de tipo *Publish-Subscribe* [47], en la que los clientes se suscriben a ciertos eventos del servidor y estos son notificados cuando dichos eventos ocurren. Como paso previo a la implementación de esta mejora, sería interesante analizar la eficiencia de las consultas a la base de datos de las operaciones identificadas como cuello de botella. También sería interesante analizar si el rendimiento de la aplicación puede escalar mediante la replicación de servicios.

6: Conclusiones y trabajo futuro

Una vez expuestos todos los aspectos relativos al desarrollo de este proyecto en todas sus fases es momento de evaluar el grado de cumplimiento de los objetivos marcados al inicio del mismo.

Considero que todos los objetivos se han cumplido de manera satisfactoria pues se ha conseguido desarrollar una nueva versión estable e internacionalizada de la aplicación, que permite a los usuarios utilizarla tanto en inglés como en español y se ha conseguido determinar el rendimiento actual de la aplicación, lo que ha permitido elaborar una propuesta con la que poder mejorar dicho rendimiento en el futuro.

En cuanto a la línea de trabajo futuro que se deberá seguir para continuar con el proyecto, esta tiene varias vertientes, cuyo detalle es el siguiente:

- Implementación de la propuesta de mejora del rendimiento de la aplicación.
- Continuar con la evaluación de la usabilidad de la aplicación, para así poder identificar aquellos elementos de la aplicación que siguen dificultando la experiencia de usuario, con el objetivo de mejorar la aplicación en este aspecto.
- Desarrollo de la funcionalidad asociada a los usuarios con cuenta de la aplicación que no se ha implementado por no ser prioritaria. Esta funcionalidad está relacionada con la recuperación de los credenciales de acceso de los usuarios a la aplicación y la visualización y edición de los datos de perfil de los usuarios registrados.

Apéndices

Apéndice A

Plan de proyecto

En este apéndice se presenta el plan de proyecto que se ha elaborado para la realización de este Trabajo de fin de máster. En dicho plan se contempla la planificación de cada una de las tres fases de las que consta el proyecto global, que son el desarrollo de la segunda versión de Crossroads 2.0, el estudio del estado del arte en el ámbito de las pruebas de carga de aplicaciones web y el análisis del rendimiento de la aplicación Crossroads 2.0. El plan desarrollado para cada una de estas fases se tratará en detalle en las secciones de este anexo, excepto el plan para el desarrollo de la segunda versión de Crossroads 2.0, pues dicho plan ya ha sido tratado en el Capítulo 3.

A.1. Planificación de la fase 2: Estudio del estado del arte en el ámbito de las pruebas de carga

La realización de esta fase del proyecto se ha planificado para llevarse a cabo durante la estancia en grupos de investigación de la asignatura *I+D+i en Informática*. De acuerdo con la guía docente de dicha asignatura [13], la duración de la estancia es de 190 horas, repartidas en 5 semanas. Teniendo en cuenta esta información, se ha elaborado un calendario, en el cual se recogen las tareas que se van a realizar en esas 5 semanas (Tabla A.1).

Semana	Fecha de inicio	Fecha de fin	Tarea
1	09/05/2022	15/05/2022	Búsqueda en literatura gris
2	16/05/2022	22/05/2022	Búsqueda en literatura gris
3	23/05/2022	29/05/2022	Búsqueda en literatura académica
4	30/05/2022	05/06/2022	Búsqueda en literatura académica
5	06/06/2022	12/06/2022	Creación de lista de criterios de selección y selección de la herramienta

Tabla A.1: Planificación temporal de la fase 2 del proyecto

Además de las tareas indicadas en la tabla, en cada una de esas semanas se continúa con el desarrollo de la segunda versión de la aplicación, pues la planificación de ambas fases se solapa.

A.2. Planificación de la fase 3: Análisis del rendimiento de la aplicación

Para la planificación temporal de la fase 3 se han agrupado todas las tareas a realizar en tres partes distintas y se han asignado fechas de inicio y de final a cada una de las tres partes. Además, se ha planificado un tiempo de descanso de 15 días entre la primera y la segunda parte de la fase, en el cual no se realizarán tareas de ningún tipo. El resumen de esta planificación se muestra en la Tabla A.2.

ID	**Fecha de inicio**	**Fecha de fin**	**Tarea**
P1	18/07/2022	31/07/2022	Diseño de la prueba
P2	15/08/2022	31/08/2022	Elaboración del plan de pruebas
P3	01/09/2022	15/09/2022	Ejecución de las pruebas y análisis de resultados

Tabla A.2: Planificación temporal de la fase 3 del proyecto

Apéndice B

Plan de pruebas de carga en JMeter

En este apéndice se muestra el detalle del plan de pruebas JMeter que se ha elaborado para poder analizar el rendimiento de la aplicación Crossroads 2.0, según lo descrito en el Capítulo 5.

B.1. Estructura del plan de pruebas

Como se puede observar en la Figura B.1, el plan de pruebas consta de un conjunto de elementos de configuración para las peticiones HTTP, un elemento de gestión de variables definidas por el usuario (que se pueden utilizar en otros puntos del plan de pruebas) y un grupo de hilos. Los elementos de configuración permiten, entre otras cosas, simular el comportamiento de un navegador web, mediante la gestión de las cookies y de la caché del mismo de manera transparente al usuario. En la Figura B.2 se muestra el contenido del elemento *HTTP Request Defaults* [29], que contiene los valores por defecto que se emplearán para enviar todas las peticiones HTTP durante la prueba. En cuanto al grupo de hilos, este es el elemento encargado de generar toda la carga de trabajo que se aplicará sobre la aplicación a probar. La configuración base del grupo de hilos se encuentra en la Figura B.3. En dicha imagen se puede apreciar la parametrización del número de hilos, definiendo como valor por defecto de la variable dos hilos y la fijación del período de subida o ramp up.

Si descendemos por la estructura de árbol del plan de pruebas, encontramos dentro del grupo de hilos el conjunto de peticiones que lanzará cada hilo durante la prueba. Para la definición de las peticiones que conforman el escenario se ha hecho uso de un plugin llamado *Apache JMeter HTTP(S) Test Script Recorder* [27], que permite grabar, a través de un proxy en el navegador, el contenido de todas las peticiones que se realizan a un determinado sitio web. De esta forma, mediante la ejecución manual del escenario de prueba en el navegador, se recopilan todos los datos de las peticiones enviadas a través del mismo. A cada petición se le ha dado el nombre de la parte de la URL correspondiente al recurso

al que se trata de acceder con la petición, indicando los parámetros correspondientes a la URL en dicho nombre.

Todas estas peticiones están agrupadas haciendo uso de un elemento llamado *Transaction Controller* [31], que permite medir el tiempo que tarda un hilo en completar el contenido del controlador. La excepción la conforman aquellas peticiones HTTP que deben ser realizadas por el moderador de la partida (en la imagen agrupadas bajo los nombres **Login moderador**, **Empezar partida** y **Finalizar ronda y partida**), pues estas solo tienen que ser ejecutadas por un único hilo, por lo que están agrupadas mediante el uso de un *If Controller* [30], que es un elemento que define una condición y cuyo contenido será ejecutado por un único hilo. En este caso, el único hilo que ejecutará dichas peticiones será el primer hilo (Figura B.4).

En cuanto a la gestión del tiempo de pensar de cada hilo, se ha decidido utilizar el elemento *Uniform Random Timer* [32], pues este se adapta a la perfección a las características del tiempo de pensar que se decidió emplear. En la Figura B.5 se muestra la configuración de este elemento. Como se puede apreciar, el tiempo máximo de pensar está parametrizado, para poder reutilizar el plan de pruebas para las pruebas de red abierta.

Para finalizar con la descripción del plan de pruebas es necesario hablar de la monitorización del estado del servidor durante la ejecución del plan de pruebas. Para ello, se ha utilizado un plugin de JMeter llamado *SSHMon* [1]. Este plugin permite realizar conexiones SSH con el servidor durante la prueba para, mediante el uso de los elementos de monitorización presentes en el propio servidor, recabar diferentes métricas sobre el estado del mismo. Para ello, es necesario indicar la dirección del servidor, así como el puerto en el que se establecerá la conexión, los credenciales de acceso a la máquina y el comando que se deberá ejecutar para obtener la métrica deseada. En la Figura B.6 se muestra la configuración que se ha utilizado para este plugin. Por motivos de seguridad, se decidió parametrizar los credenciales de acceso al servidor, por lo que estos no aparecen en la imagen.

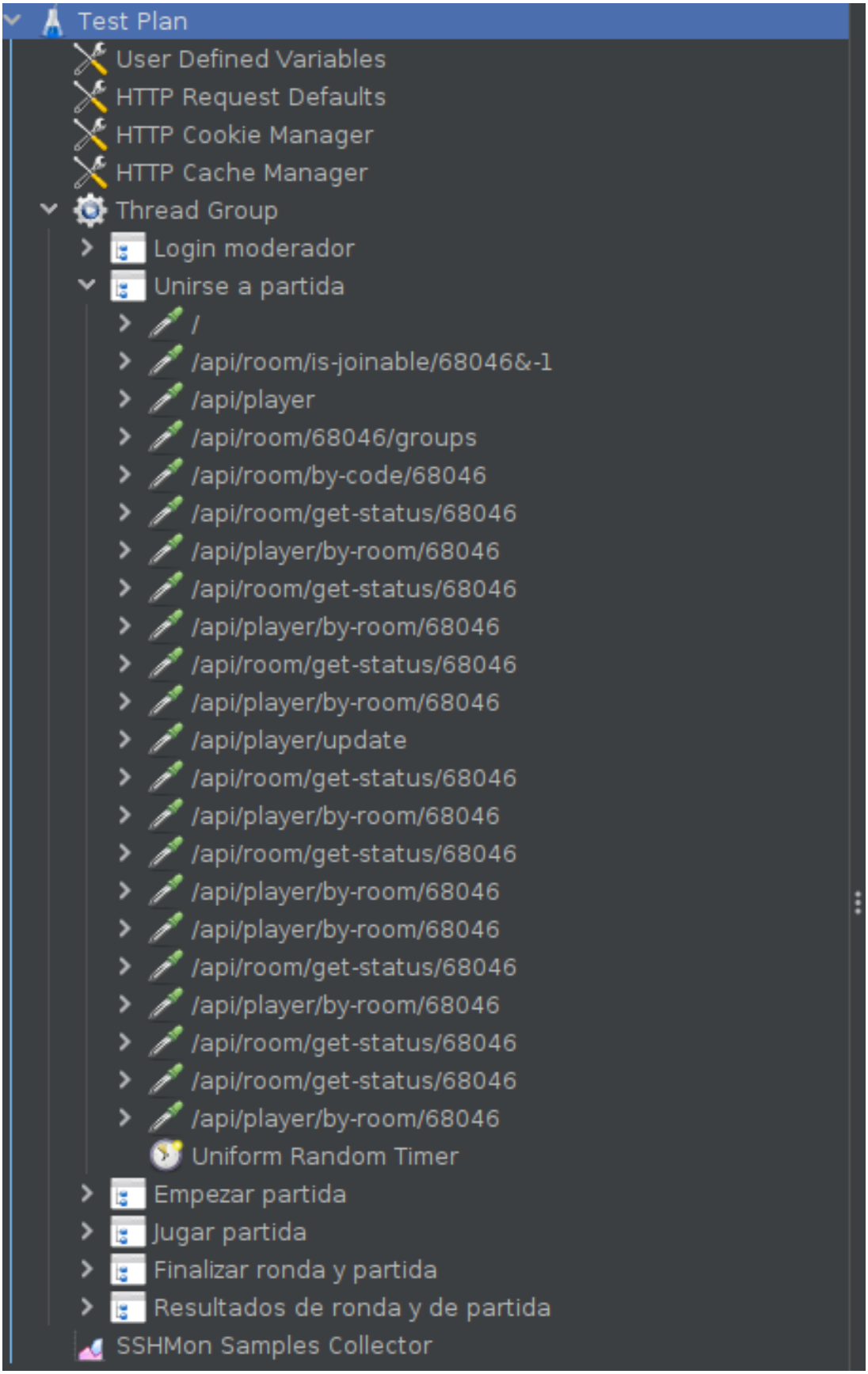

Figura B.1: Estructura general del plan de pruebas

Figura B.2: Configuración por defecto de todas las peticiones HTTP que se lanzan durante el plan de pruebas

Figura B.3: Configuración del grupo de hilos del plan de pruebas

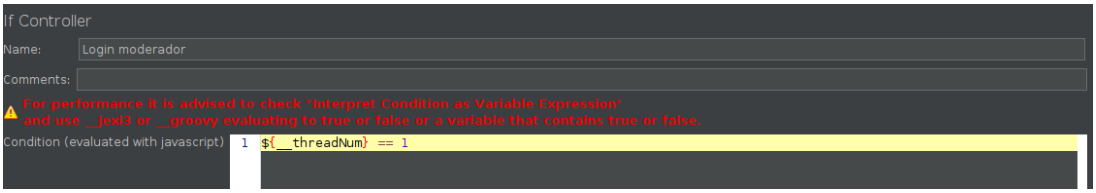

Figura B.4: Configuración del *If Controller* para la petición que se encarga de iniciar la sesión del moderador

Figura B.5: Configuración del *Uniform Random Timer*

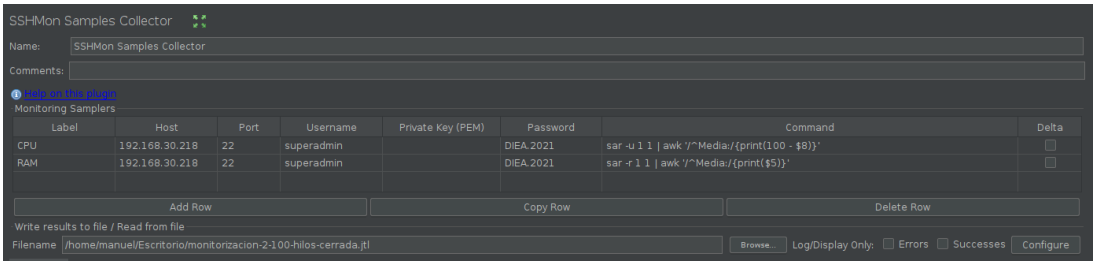

Figura B.6: Configuración del plugin SSHMon

Apéndice C

Paquete de replicabilidad de las pruebas de carga

El objetivo de este apéndice final de la memoria es elaborar una guía detallada que permita evaluar el rendimiento de la aplicación Crossroads 2.0 en el futuro, bajo las mismas condiciones que en este proyecto. En las siguientes secciones se irá describiendo cada uno de los pasos realizados para la preparación del entorno de las pruebas antes de cada ejecución de las pruebas de carga, la forma de puesta en marcha de las mismas y la generación de los informes de resultados, entre otros. Además, se ha creado un repositorio en el Gitlab de la Escuela con todos los elementos y scripts utilizados para esta parte del proyecto. Este repositorio está disponible a través del siguiente enlace.

C.1. Preparación del entorno de pruebas

Antes de poder ejecutar el plan de pruebas de carga es necesario preparar el entorno de pruebas. En primer lugar hay que preparar una base de datos de prueba, pues algunas peticiones que se van a lanzar dependen de que esta base de datos se encuentre en un estado determinado. Para poder reproducir el estado de la base de datos de prueba se ha creado un script SQL con todas las sentencias SQL necesarias. Dicho script se encuentra dividido en las siguientes partes:

- Creación de la cuenta de usuario del moderador.
- Creación de la sala de partida con el código de sala indicado en el plan de pruebas y un número máximo de grupos de 10.
- Creación de los 10 grupos de jugadores.
- Creación de las preguntas y respuestas del formulario y de las pistas asociadas a cada pregunta.

Para facilitar el uso de este script y el borrado del contenido de la base de datos entre dos ejecuciones del plan de pruebas, se ha creado un script bash que automatiza esta tarea. Este script bash se llama `reset.sh`.

El segundo paso de la preparación del entorno de pruebas consiste en utilizar la herramienta OpenVPN para conectarse al servidor VPN que controla el acceso a la red privada en la que se encuentra la máquina virtual del servidor. Este paso es imprescindible, pues la conexión SSH de monitorización del servidor se realizará a través de la IP privada de la máquina virtual. Además, con el objetivo de evitar el reseteo de la conexión con el servidor VPN será necesario abrir una conexión SSH auxiliar, que será distinta de la que utilizará el plugin SSHMon de JMeter.

Finalmente, será necesario modificar el plan de pruebas JMeter, para cambiar la ruta en el sistema de ficheros de la máquina local y el nombre del fichero en el que se guardarán los resultados de monitorizar el servidor de la aplicación.

C.2. Ejecución del plan de pruebas

El lanzamiento de una ejecución del plan de pruebas con JMeter se ha de realizar en modo CLI, desde una consola de terminal. La sintaxis base del comando que hay que ejecutar es la siguiente:

```
jmeter -n -t plan-pruebas.jmx -l <FICHERO_RESULTADOS> -j <
   FICHERO_LOG> -DHILOS=<NUM_HIHLOS> -DUSERNAME=<
   USUARIO_SERVIDOR> -DPASSWORD=<PASSWORD_SERVIDOR> -DTHINK=<
   TIEMPO_DE_PENSAR>
```

El significado de los parámetros indicados en el comando anterior es el siguiente:

- `FICHERO_RESULTADOS`. Fichero JTL en el que se guardarán los resultados de la ejecución del plan de pruebas.
- `FICHERO_LOG`. Fichero de log en el que JMeter irá volcando su propia información de estado durante la ejecución de las pruebas.
- `NUM_HIHLOS`. El número de hilos que se va a utilizar en la ejecución del plan de pruebas.
- `USUARIO_SERVIDOR`. El nombre de usuario de una cuenta en la máquina servidor.
- `PASSWORD_SERVIDOR`. La contraseña correspondiente al nombre de usuario indicado.
- `TIEMPO_DE_PENSAR`. Tiempo de pensar máximo que se utilizará en el plan de pruebas (en milisegundos).

C.3. Generación de los informes HTML

Para la generación automática de todos los informes HTML se ha creado un script bash, llamado `gen-html.sh`. Este script requiere que todos los ficheros JTL de resultados se encuentren en una carpeta en el directorio en el que se encuentra el script. El nombre de esta carpeta es `ejecuciones`.

C.4. Adaptación de los ficheros de monitorización para mostrar bien las gráficas de utilización de la memoria RAM

Antes de poder representar las gráficas de utilización de la memoria RAM es necesario modificar los registros correspondientes en los ficheros con los resultados de monitorización. Esta modificación se debe a que la salida del comando empleado para obtener la utilización de este recurso incluye la coma decimal, en lugar del punto habitual en otras representaciones, por lo que JMeter ignora el símbolo e interpreta mal el resultado. Para facilitar la modificación de todos los ficheros se ha creado un script Python que automatiza esta tarea. El script se llama `get-real-ram.py` y se lanza desde la carpeta en la que están todos los ficheros de monitorización haciendo uso del siguiente comando:

```
python3 get-real-ram.py
```

Bibliografía

[1] Aladev, R. How to monitor server resource utilization with jmeter's sshmon listener. https://dzone.com/articles/how-to-monitor-server-resource-utilization-with-jm. Internet; Accedido por última vez: 03/09/2022.

[2] Alda Peñafiel, M. Desarrollo del front-end y mejoras en el back-end de un juego didáctico multijugador de competición y consenso sobre el cambio climático. https://uvadoc.uva.es/handle/10324/50085. Internet; accedido por última vez 05-julio-2022.

[3] Angular. Angular. https://angular.io/. Internet; Accedido por última vez: 27/08/2022.

[4] Angular. Angular internationalization. https://angular.io/guide/i18n-overview. Internet; Accedido por última vez: 31/08/2022.

[5] Angular. Httpinterceptor. https://angular.io/api/common/http/HttpInterceptor. Internet; Accedido por última vez: 31/08/2022.

[6] Borzemski, L., Danielak, M., and Kamińska-Chuchmała, A. Spatio-temporal web performance prediction: Turning bands method and sequential gaussian simulation. *Procedia Computer Science 96* (2016), 568–576. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 20th International Conference KES-2016.

[7] Brady, J. F., and Gunther, N. J. How to emulate web traffic using standard load testing tools. *CoRR abs/1607.05356* (2016).

[8] Catillo, M., Ocone, L., Villano, U., and Rak, M. Black-box load testing to support auto-scaling web applications in the cloud. *International Journal of Grid and Utility Computing 12*, 2 (2021), 139–148.

[9] CONSORTIUM, L. Global sustainability crossroads ii. https://www.locomotion-h2020.eu/locomotion-models/global-sustainability-crossroads-ii/. Internet; accedido por última vez 06-julio-2022.

[10] CORPORATION, O. Mysql. https://www.mysql.com/. Internet; Accedido por última vez: 27/08/2022.

[11] CYPRESS. Cypress. https://www.cypress.io/. Internet; Accedido por última vez: 27/08/2022.

[12] DAVID ÁLVAREZ ANTELO, I. C.-P., AND MIGUEL, L. J. Global sustainability crossroads: A participatory simulation game to educate in the energy and sustainability challenges of the 21st century. *Sustainability 11*, 13 (2019), 3672.

[13] DE VALLADOLID, U. Proyecto docente de la asignatura i+d+i en informática para el curso 2021-2022. https://albergueweb1.uva.es/guia_docente/uploads/2021/693/55125/1/Documento.pdf. Internet; Accedido por última vez: 02/09/2022.

[14] DRAHEIM, D., GRUNDY, J., HOSKING, J., LUTTEROTH, C., AND WEBER, G. Realistic load testing of web applications. In *Conference on Software Maintenance and Reengineering (CSMR'06)* (2006), IEEE, pp. 11 pp.–70.

[15] DRAHEIM, D., LUTTEROTH, C., AND WEBER, G. A source code independent reverse engineering tool for dynamic web sites. In *Ninth European Conference on Software Maintenance and Reengineering* (2005), pp. 168–177.

[16] FLOOD.IO. Ramp up. https://guides.flood.io/test-execution/test-parameters/ramp-up. Internet; Accedido por última vez: 02/09/2022.

[17] FOUNDATION, O. Node.js. https://nodejs.org/es/. Internet; Accedido por última vez: 27/08/2022.

[18] GALINDO GÓMEZ, M. Elaboración y ejecución de un plan de pruebas de usabilidad para juego educativo crossroads 2.0. https://uvadoc.uva.es/handle/10324/50376. Internet; accedido por última vez 06-julio-2022.

[19] GARCÍA, B., DELGADO KLOOS, C., ALARIO-HOYOS, C., AND MUNOZ-ORGANERO, M. Selenium-jupiter: A junit 5 extension for selenium webdriver. *Journal of Systems and Software 189* (2022), 111298.

[20] H2020, L. Low-carbon society: an enhanced modelling tool for the transition to sustainability. https://www.locomotion-h2020.eu/about-project/overview/. Internet; accedido por última vez 05-julio-2022.

[21] HAMED, O., AND KAFRI, N. Performance testing for web based application architectures (.net vs. java ee). In *2009 First International Conference on Networked Digital Technologies* (2009), IEEE, pp. 218–224.

[22] Huaji, Z., and Huarui, W. Research on web application load testing model. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)* (2017), IEEE, pp. 1175–1178.

[23] Ignacio de Blas and Iñigo Capellán-Pérez and Jaime Nieto and Carlos de Castro and Luis Javier Miguel and Óscar Carpintero and Margarita Mediavilla and Luis Fernando Lobejón and Noelia Ferreras-Alonso and Paula Rodrigo and Fernando Frechoso and David Álvarez Antelo. Medeas: a new modeling framework integrating global biophysical and socioeconomic constraints. *Energy Environ. Sci 13* (2020), 986–1017.

[24] Imre, G., Charaf, H., and Lengyel, L. Performance of a java web application running on amazon ec2 micro instance, 2013.

[25] Inc, M. Mongodb. https://www.mongodb.com/es. Internet; Accedido por última vez: 27/08/2022.

[26] Jailia, M., Agarwal, M., and Kumar, A. Comparative study of n-tier and cloud-based web application using automated load testing tool. In *Information and Communication Technology* (Singapore, 2018), D. K. Mishra, A. T. Azar, and A. Joshi, Eds., Springer Singapore, pp. 239–250.

[27] JMeter. Apache jmeter http(s) test script recorder. https://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html. Internet; Accedido por última vez: 03/09/2022.

[28] JMeter. Generating report dashboard. https://jmeter.apache.org/usermanual/generating-dashboard.html. Internet; Accedido por última vez: 03/09/2022.

[29] JMeter. Http request defaults. https://jmeter.apache.org/usermanual/component_reference.html#HTTP_Request_Defaults. Internet; Accedido por última vez: 03/09/2022.

[30] JMeter. If controller. https://jmeter.apache.org/usermanual/component_reference.html#If_Controller. Internet; Accedido por última vez: 03/09/2022.

[31] JMeter. Transaction controller. https://jmeter.apache.org/usermanual/component_reference.html#Transaction_Controller. Internet; Accedido por última vez: 03/09/2022.

[32] JMeter. Uniform random timer. https://jmeter.apache.org/usermanual/component_reference.html#Uniform_Random_Timer. Internet; Accedido por última vez: 03/09/2022.

[33] Kalita, M., and Bezboruah, T. Investigations on implementation of web applications with different techniques. *IET SOFTWARE 6*, 6 (DEC), 474–478.

[34] Lutteroth, C., and Weber, G. Modeling a realistic workload for performance testing. In *2008 12th International IEEE Enterprise Distributed Object Computing Conference* (2008), IEEE, pp. 149–158.

[35] Microsoft. Edit think times to simulate website human interaction delays in load tests scenarios. https://docs.microsoft.com/en-us/visualstudio/test/edit-think-times-in-load-test-scenarios?view=vs-2022. Internet; Accedido por última vez: 02/09/2022.

[36] Molero, X., Juiz, C., and Rodeño, M. *Evaluación y Modelado del Rendimiento de los Sistemas Informáticos*. Pearson Educación, 2004.

[37] Mozilla. 500 error interno del servidor. https://developer.mozilla.org/es/docs/Web/HTTP/Status/500. Internet; Accedido por última vez: 08/09/2022.

[38] Muradas, Y. Qué es npm y para qué sirve. https://openwebinars.net/blog/que-es-node-package-manager/. Internet; Accedido por última vez: 27/08/2022.

[39] Pahino, R. ¿qué son spring framework y spring boot? tu primer programa java con este framework. https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx. Internet; Accedido por última vez: 27/08/2022.

[40] Pradeep, S., and Sharma, Y. K. A pragmatic evaluation of stress and performance testing technologies for web based applications. In *2019 Amity International Conference on Artificial Intelligence (AICAI)* (2019), pp. 399–403.

[41] Pratibha, and Ashraf, M. Dominant behavior identification of load data. In *International Conference on Computing, Communication & Automation* (2015), IEEE, pp. 142–145.

[42] Ramasamy, G., and Ramalingam, S. An effective automation testing framework for oats tool. In *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems* (New Delhi, 2015), L. P. Suresh, S. S. Dash, and B. K. Panigrahi, Eds., Springer India, pp. 543–550.

[43] Romano, B. L., e Silva, G. B., de Campos, H. F., Vieira, R. G., da Cunha, A. M., Silveira, F. F., and Ramos, A. C. B. Software testing for web-applications non-functional requirements. In *2009 Sixth International Conference on Information Technology: New Generations* (2009), IEEE, pp. 1674–1675.

[44] Ronache, A. Flask-babel. https://pypi.org/project/Flask-Babel/. Internet; Accedido por última vez: 31/08/2022.

[45] Ronacher, A. Flask. https://flask.palletsprojects.com/en/2.0.x/. Internet; Accedido por última vez: 27/08/2022.

[46] Shojaee, A., Agheli, N., and Hosseini, B. Cloud-based load testing method for web services with vms management. In *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (2015), IEEE, pp. 170–176.

[47] Singh, A. System design basics: Pub/sub messaging. https://medium.com/geekculture/system-design-basics-pub-sub-messaging-88dfd98e67b7. Internet; Accedido por última vez: 09/09/2022.

[48] Snellman, N., Ashraf, A., and Porres, I. Towards automatic performance and scalability testing of rich internet applications in the cloud. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications* (2011), IEEE, pp. 161–169.

[49] Sun, Y., White, J., and Eade, S. A model-based system to automate cloud resource allocation and optimization. In *Model-Driven Engineering Languages and Systems* (Cham, 2014), J. Dingel, W. Schulte, I. Ramos, S. Abrahão, and E. Insfran, Eds., Springer International Publishing, pp. 18–34.

[50] Sun, Y., White, J., Eade, S., and Schmidt, D. C. Roar: A qos-oriented modeling framework for automated cloud resource allocation and optimization. *Journal of Systems and Software 116* (2016), 146–161.

[51] Thai, N. N. What is a spring bean? https://www.baeldung.com/spring-bean. Internet; Accedido por última vez: 31/08/2022.

[52] Trinh, T.-D., Wetz, P., Do, B.-L., Kiesling, E., and Tjoa, A. M. Distributed mashups: a collaborative approach to data integration. *International Journal of Web Information Systems 11*, 3 (Jan 2015), 370–396.

[53] Valencia, P. W. Testing end-to-end (e2e). https://programadorwebvalencia.com/cursos/testing/e2e/. Internet; Accedido por última vez: 27/08/2022.

[54] Vani, B., Deepalakshmi, R., and Suriya, S. Web based testing — an optimal solution to handle peak load. In *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering* (2013), IEEE, pp. 5–10.

[55] Wang, X., Zhou, B., and Li, W. Model-based load testing of web applications. *Journal of the Chinese Institute of Engineers 36*, 1 (2013), 74–86.

[56] Wikipedia. Single-page application. https://es.wikipedia.org/wiki/Single-page_application. Internet; Accedido por última vez: 27/08/2022.

[57] Wikipedia. Vensim. https://en.wikipedia.org/wiki/Vensim. Internet; Accedido por última vez: 11/09/2022.

[58] Wikipedia. Web of science. https://es.wikipedia.org/wiki/Web_of_Science. Internet; Accedido por última vez: 11/09/2022.

[59] Y Jeff Sutherland, K. S. La guía de scrum. https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf. Internet; Accedido por última vez: 31/08/2022.

[60] Zhou, J., Zhou, B., and Li, S. Ltf: A model-based load testing framework for web applications. In *2014 14th International Conference on Quality Software* (2014), IEEE, pp. 154–163.

[61] Zhu, H. J., and Wu, H. R. Study and improvement of a web application load testing model. In *Mechatronics and Industrial Informatics* (7 2013), vol. 321 of *Applied Mechanics and Materials*, Trans Tech Publications Ltd, pp. 2969–2973.