UNIÓN EUROPEA
Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa

GENERALITAT VALENCIANA

IVACE
INSTITUT VALENCIÀ DE COMPETITIVITAT EMPRESARIAL

INESCOP
INSTITUTO TECNOLÓGICO DEL CALZADO Y CONEXAS

EXPEDIENTE	IMDEEA/2017/15
ACRÓNIMO	PASOCON
PROGRAMA	Proyectos de I+D de carácter no económico realizados en cooperación con empresas
TÍTULO DEL PROYECTO	GENERACIÓN DE COMBINADORES DE MATERIALES VIRTUALES PARA CALZADO BASADOS EN LA WEB 2.0

Entregable E3.1.

INFORME TÉCNICO CON LOS RESULTADOS DE LAS PRUEBAS DE ESTRÉS Y LAS SIMULACIONES REALIZADAS

ÍNDICE

1. Descripción del entregable

Durante la fase de desarrollo de la interfaz e implementación del prototipo de pruebas, se van realizando diferentes pruebas de funcionamiento y rendimiento, de manera que se garantice una implementación óptima para la plataforma.

No obstante, esta implementación se centra en tareas concretas del desarrollo y no prueba de manera óptima la viabilidad de la solución implantada, además de que tampoco aporta datos cuantificables que permitan determinar las posibilidades que ofrece, así como los límites de explotación.

Para poder aportar claridad e información valiosa sobre la capacidad de rendimiento y explotación que ofrece el prototipo de pruebas desarrollado, se han realizado una serie de pruebas de estrés y simulación en entornos reales que han permitido la elaboración del siguiente entregable, así como la obtención de información muy valiosa para la validación y análisis de conclusiones del proyecto.

2. Trabajo realizado

Una vez se dispone ya de parte del prototipo desarrollado, se puede empezar a plantear y organizar la fase de pruebas y validación, ya que algunas de las funcionalidades más importantes a validar se pueden probar sin necesidad de tener todo el desarrollo completado.

Así pues, se comienza realizando un estudio sobre las diferentes técnicas utilizadas para la realización de pruebas de estrés del software, orientadas a la validación de aplicaciones web. Se analizan las plataformas más relevantes en esta área, para determinar cuál es la que mejor se adapta a la problemática y por lo tanto es la más idónea para utilizar en el proyecto. Es importante destacar, que se han buscado soluciones de código abierto y por tanto de libre utilización, ya que se trata de un concepto muy extendido y habitual en los desarrollos web.

Posteriormente, con la plataforma a utilizar ya definida, se han establecido las pruebas que se iban a realizar, donde primero había que identificar los puntos más importantes y con más probabilidad de uso/carga por parte de los usuarios, para luego establecer los parámetros a probar en cada uno de ellos. Esto ha permitido obtener la información resultante que nos proporcionan los datos reales sobre los límites de carga y utilización del prototipo, así como la confirmación de que tanto el desarrollo software como la tecnología hardware y software utilizadas en el servidor de pruebas, son los idóneos para llevar a cabo una solución de esta índole.

Por último, cabe destacar que todas estas pruebas y validación de la arquitectura propuesta, se han podido llevar a cabo en el servidor de pruebas, simulando condiciones totalmente reales a como sería una implantación de una empresa como negocio particular. Por lo que los datos obtenidos, son perfectamente extrapolables a esa futura implantación en un entorno de producción real.

3. Estudio de la tecnología a utilizar

Una primera fase dentro del desarrollo de la tarea de pruebas y validación, consiste en la realización de un estudio de las diferentes herramientas y plataformas más comunes para la realización y automatización de las pruebas.

En esta fase, primero de todo había que diferenciar entre los diferentes tipos de desarrollo de aplicaciones software existentes, ya que no se utiliza el mismo procedimiento, tipo de pruebas, ni herramientas, para una aplicación software de escritorio que para una aplicación o plataforma web 2.0. Por lo tanto, el primer filtrado de búsqueda fue por el tipo de aplicación, estudiando solamente aquellas herramientas destinadas a la validación y pruebas de estrés de plataformas web, que fueran compatibles con la tecnología base utilizada en el proyecto, es decir, aplicaciones PHP, HTML5, JavaScript, etc.

Puesto que la tecnología planteada para el desarrollo del proyecto es la más extendida en el ámbito de las soluciones web, se encontraron numerosas herramientas e información referente a esta problemática, además gran parte de ellas de código abierto y por tanto, sin ningún coste asociado la utilización de las mismas. Herramientas como [TheGrinder] ofrece un framework de pruebas de carga basado en Java y de código abierto, por lo que la comunidad va mejorando la herramienta, no obstante, la interfaz gráfica es muy limitada y basa todo su desarrollo en el manejo de la consola de Grinder. Otra herramienta similar, se puede encontrar en [Gatling], que también dispone de una interfaz limitada, pero que por otra parte, genera informes detallados de las pruebas realizadas.

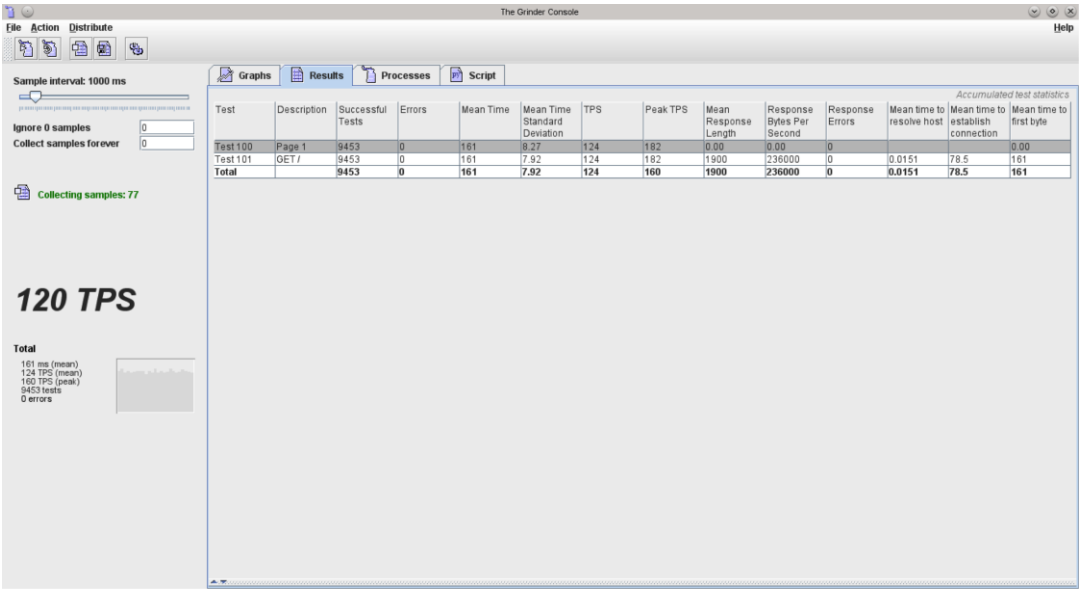

Figura 1: Interfaz gráfica de la herramienta [TheGrinder].

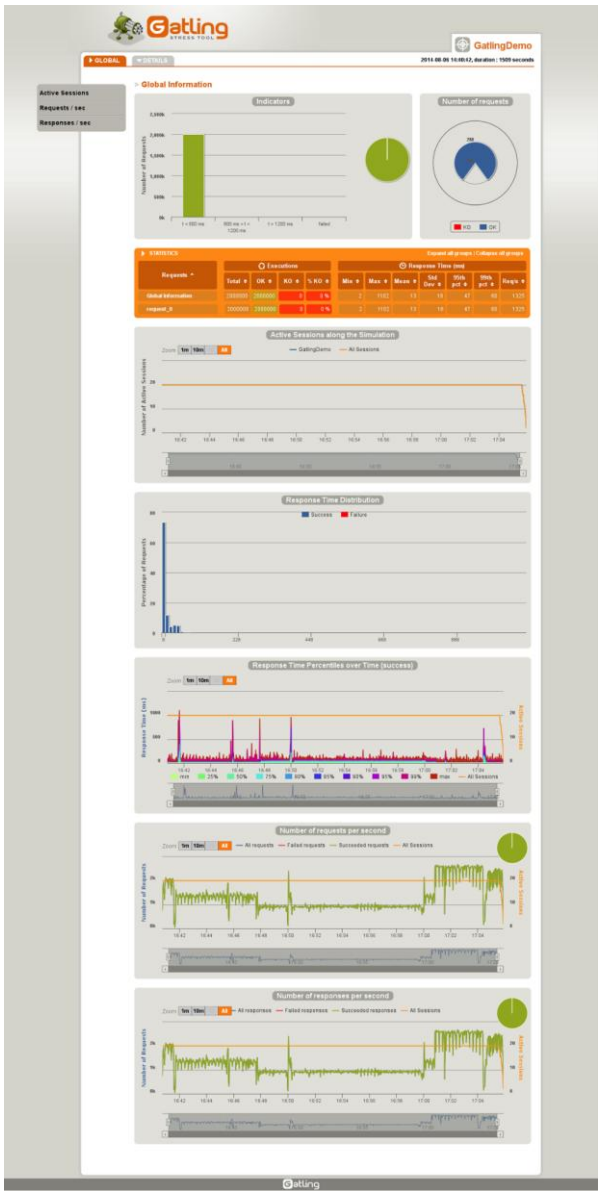

Figura 2: Informe de resultados de la herramienta [Gatling].

Aunque estas herramientas ofrecen una interfaz gráfica limitada y un poco pobre en cuanto a diseño se refiere, al menos permiten abstraer en cierta medida al usuario de la realización 'manual' de las pruebas, no es el caso de [TSung] que a pesar de poder realizar todo tipo de pruebas y obtener los respectivos informes resultantes, no dispone de interfaz gráfica, lo que hace que su uso sea más complicado y menos versátil, por ello, ya de ante mano, esta solución quedó descartada. De la misma manera, [Locust] ofrece una potente herramienta basada en Python, pero que carece de interfaz gráfica, de manera que para definir las pruebas se utiliza un lenguaje de script que permite realizar prácticamente cualquier test que se desee.

```
~$ tsung
Usage: tsung <options> start|stop|debug|status
Options:
    -f <file>     set configuration file (default is ~/.tsung/tsung.xml)
                   (use - for standard input)
    -l <logdir>   set log directory where YYYYMMDD-HHMM dirs are created (default is ~/.tsung/log/)
    -i <id>       set controller id (default is empty)
    -r <command>  set remote connector (default is ssh)
    -s            enable erlang smp on client nodes
    -p <max>      set maximum erlang processes per vm (default is 250000)
    -m <file>     write monitoring output on this file (default is tsung.log)
                   (use - for standard output)
    -F            use long names (FQDN) for erlang nodes
    -w            warmup delay (default is 1 sec)
    -v            print version information and exit
    -6            use IPv6 for Tsung internal communications
    -x <tags>     list of requests tag to be excluded from the run (separated by comma)
    -h            display this help and exit
```

Figura 3: Interfaz de consola de la herramienta [TSung].

```
from locust import HttpLocust, TaskSet, task

class SimpleLocustTest(TaskSet):

      @task
      def get_something(self):
          self.client.get("/")

class LocustTests(HttpLocust):
      task_set = SimpleLocustTest
```

Figura 4: Ejemplo de script para la herramienta [Locust] y su informe de resultados.

Finalmente, la última herramienta a detallar del análisis y que ha sido la elegida para el desarrollo, es [JMeter] de Apache. Se trata de una aplicación de escritorio basada en Java, pero con una interfaz mucho más potente y fácil de usar que los casos vistos anteriormente. Permite la realización y depuración de tests de carga y estrés de forma mucho más sencilla, y se trata del sistema más utilizado y popular entre las alternativas de código libre válidas para el trabajo con proyectos basados en la web.

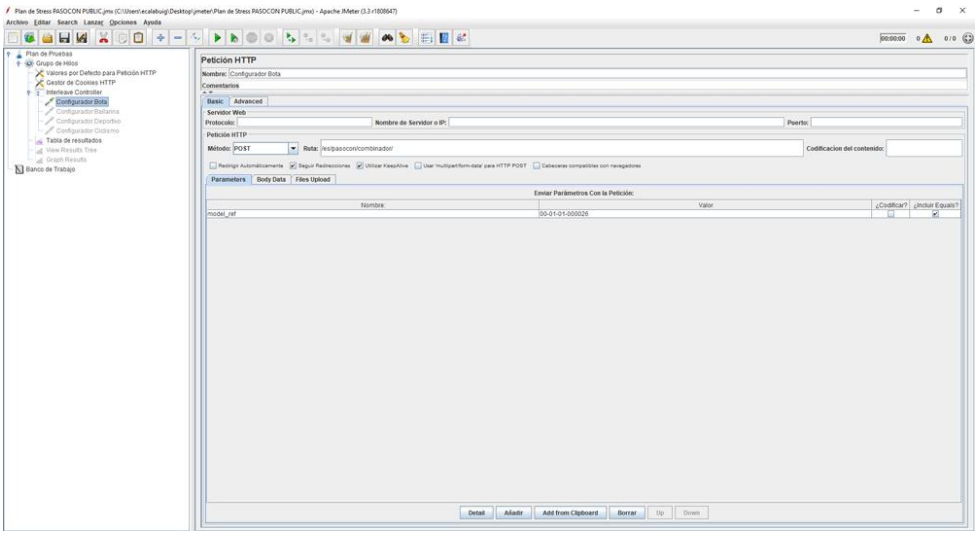

Figura 5: Herramienta [JMeter] con las pruebas de carga utilizadas para los modelos de pruebas del prototipo desarrollado.

Esta potente herramienta, junto con el comando *htop* de Linux, y las herramientas para desarrolladores de los navegadores web, han permitido realizar un completo estudio sobre el comportamiento de la plataforma web 2.0, los requerimientos del servidor, la cantidad de datos transferidos por modelo, tiempos de carga de la información, etc.

4. Definición y ejecución de las pruebas de estrés

Una vez establecida la herramienta de pruebas a utilizar para esta fase del proyecto, se definen las pruebas a realizar para determinar el comportamiento y límites de la tecnología propuesta. En este apartado, se van a definir las características consideradas más relevantes, y que por tanto serán el principal objetivo de los tests, además se describirán las pruebas realizadas en cada caso, analizando convenientemente los resultados obtenidos.

Primero de todo, gracias al sistema de virtualización implantado en el servidor de pruebas, se podrán variar las capacidades del mismo en función de las pruebas que se van a realizar. Por lo tanto, se realizarán tres simulaciones diferentes del servidor, de manera que se podrá apreciar para cada una de ellas, el nivel de carga y saturación de los procesadores, siendo éste, el elemento más importante en cuanto a rendimiento se refiere. Gracias a esta diferenciación, se podrán obtener datos referentes a las distintas características de servidor y así poder determinar que capacidades son las más adecuadas para cada caso particular.

Configuración Servidor 1	2 CPU's	4 Núcleos (2.4 Ghz x 4)
Configuración Servidor 2	2 CPU's	10 Núcleos (2.4 Ghz x 10)
Configuración Servidor 3	2 CPU's	20 Núcleos (2.4 Ghz x 20)

Tabla 1: Configuraciones del servidor para pruebas de estrés y validación.

Además de esta configuración básica, también se podrá variar la capacidad de almacenamiento o incluso memoria RAM de cada servidor virtualizado. Sin embargo, para este cometido, estos dos parámetros permanecerán constantes puesto que no alteran los resultados, aunque sí que se aportarán datos y pruebas de los límites de almacenamiento y procesamiento de memoria RAM para diferentes operaciones del prototipo.

Por otra parte, en cuanto a las pruebas se refiere, se van a realizar pruebas para análisis de tiempos de carga, pruebas para evaluar el volumen de datos descargados y pruebas que determinan el rendimiento del servidor sometido a diferentes situaciones de carga. En el caso de las dos primeras, se han realizado con una configuración básica del servidor, puesto que no miden las capacidades del servidor, sino que proporcionan información referente a un proceso determinado sobre los modelos de prueba propuestos. Sin embargo, en el caso de los terceros tipos de pruebas, sí que se comprobarán según las diferentes configuraciones de servidor comentadas, ya que el comportamiento y rendimiento podrá variar entre ellas.

Finalmente, como base de datos de prueba se utilizarán los modelos que se han diseñado y renderizado para el propósito de este proyecto. Aunque esta información ya se especificó en el entregable *E2.1* del presente proyecto, se detallan a continuación los modelos y sus características con el fin de facilitar la lectura y seguimiento del entregable:

Modelo	Descripción	Materiales (Nº)	Escenas (Nº)	Resolución (AnchoxAlto)
	**Bota** con alto grado de configuración e imágenes de resolución media	169	3	1024x576 (PAL WS)
	**Bailarina** con grado medio de configuración e imágenes de resolución muy alta	85	1	3000x1863
	**Zapatilla ciclismo** con grado medio de configuración e imágenes de resolución media	43	2	1024x576 (PAL WS)
	**Deportivo** con poco grado de configuración e imágenes de resolución alta	38	4	1920x1080 (Full HD)

Tabla 2: Características de modelos diseñados para pruebas y validación.

4.1 Pruebas de tiempos de carga

En este tipo de pruebas se ha analizado el tiempo que conlleva la petición y carga de un modelo cuando el usuario está interactuando con la aplicación. Gracias a la información obtenida con este tipo de pruebas, se podrá determinar como afectan las diferentes características de los modelos virtuales al tiempo de carga de los mismos al usuario final. En este caso de pruebas, no se tienen en cuenta la velocidad de conexión y se utiliza la *Configuración Servidor 1*, ya que no se trata de analizar la velocidad de la operación en general, sino de estudiar cómo afectan los datos y características del modelo a los tiempos de carga del mismo.

En cuanto a las operaciones analizadas para esta prueba de carga, se pueden resumir en dos, la carga del modelo inicial y el cambio de un material de un modelo ya cargado. Se utilizan estas dos operaciones porque serán las más comunes que se realizarán en la plataforma, ya que el resto de operaciones (gestión de usuarios, preferencias, modelos, etc.) se realizan en un momento puntual, cuando el administrador o usuario indicado esté personalizando y poniendo en marcha la herramienta. Por el contrario, los posibles casos de uso exhaustivo de la plataforma se darán en la mayoría de los casos en la carga de modelos e interactuando con ellos (cambiando materiales).

Una vez definidas las operaciones a utilizar y las pruebas con el sistema JMeter, se procede con la ejecución de las mismas y obtención de los datos. En este caso, también se hace uso de la herramienta para desarrolladores del navegador Google Chrome ya que permite ver los tiempos de descarga de información asíncrona a la carga del modelo (por ejemplo, la descarga de las imágenes que realiza el navegador) y que no puede ser trazada por JMeter. Así pues, los datos obtenidos durante la ejecución de estas pruebas fueron los siguientes:

MODELO	CARGA DE MODELO (Seg.)	CAMBIO DE MATERIAL (Seg.)
BOTA	2,52	1,61
BAILARINA	2,08	1,09
ZAPATILLA CICLISMO	1,62	0,87
DEPORTIVO	1,78	0,95

Tabla 3: Tiempos de carga y cambio de material de los modelos de prueba.

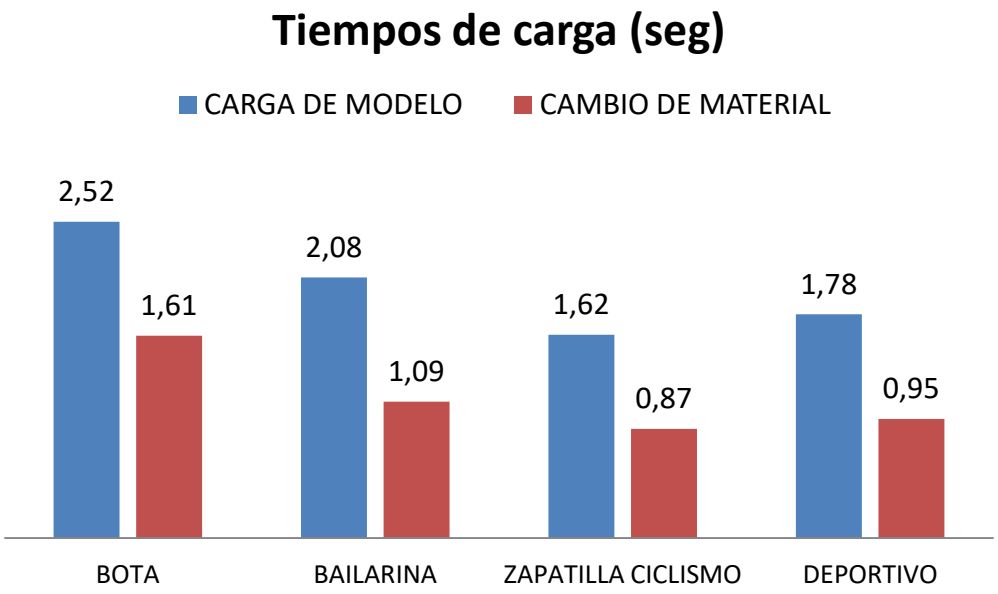

Gráfico 1: Comparativa de los tiempos de carga y cambio de material entre los modelos tipo de prueba.

A partir de los datos obtenidos, se puede deducir la conclusión evidente de que los tiempos de carga iniciales del modelo son superiores a los cambios de material que se realizan posteriormente. Esto es debido a que inicialmente se descarga toda la información del modelo, mientras que en las posteriores peticiones (cambios de materiales) sólo se accede a la información afectada por el cambio solicitado, gracias en gran medida a la estructura de datos definida y a la optimización del desarrollo.

En cuanto a la comparativa entre los diferentes modelos de pruebas, se puede apreciar que los tiempos de carga del modelo *BOTA* son ligeramente mayores al resto, puesto que dispone de una configuración de grupos y reglas de combinación complejas y por lo tanto, de un mayor volumen de información. Esto afecta no sólo a la cantidad de información descargada del servidor, sino también al coste de computación para realizar las operaciones de lectura de modelos y aplicación de reglas de combinación. De la misma manera, en el resto de modelos se aprecia un comportamiento que hace justicia a los datos y complejidades de cada uno de ellos, siendo el modelo *BAILARINA* el siguiente en cuanto a tiempos de carga, al contar con las imágenes de render de mayor resolución, mientras que el modelo *ZAPATILLA DE CICLISMO* presenta los resultados más óptimos al tener una configuración sencilla a la vez que imágenes de render de una resolución más óptima para su trabajo en la web.

Como conclusión a este tipo de pruebas, se puede determinar que aunque el grado de resolución de las imágenes foto-realistas de los modelos influye en los tiempos de carga y cambio de materiales, también habrá que tener en cuenta el número de materiales y complejidad de combinación del modelo.

4.2 Pruebas de volumen de carga

El segundo caso de pruebas realizado, consiste en el análisis del volumen o cantidad de datos descargados durante la interacción con la plataforma. Al igual que en las pruebas anteriores se

analizará este proceso para las operaciones de carga de modelo y cambio de material, por considerarlas críticas en el proceso y además porque son las que mayor flujo de información requieren. De la misma manera, se utilizará para el servidor la *Configuración Servidor 1* establecida.

Con la ejecución de este tipo de pruebas, se va a determinar el volumen de datos que se necesita en el proceso normal de combinación de un modelo, lo que permite realizar una estimación más detallada sobre el ancho de banda que se puede llegar a utilizar para un gran número de consultas o accesos a los modelos virtuales. Gracias a las diferentes características estos modelos, también se podrá evaluar cómo se comportan cada uno de ellos en relación con este campo.

Por lo tanto, para la ejecución de estas pruebas, no fue necesario definir ninguna prueba específica en JMeter, puesto que los datos de carga completos se analizan con más detalle desde la propia herramienta para desarrolladores del navegador. Una vez definido el proceso de prueba, se realizaron las diferentes operaciones, recopilando los datos proporcionados por la herramienta:

MODELO	CARGA DE MODELO (MB.)	CAMBIO DE MATERIAL (MB.)
BOTA	1,60	0,56
BAILARINA	5,90	0,55
ZAPATILLA CICLISMO	1,30	0,10
DEPORTIVO	3,30	0,24

Tabla 4: Información de datos descargados en las operaciones de carga y cambio de material para los modelos tipo.

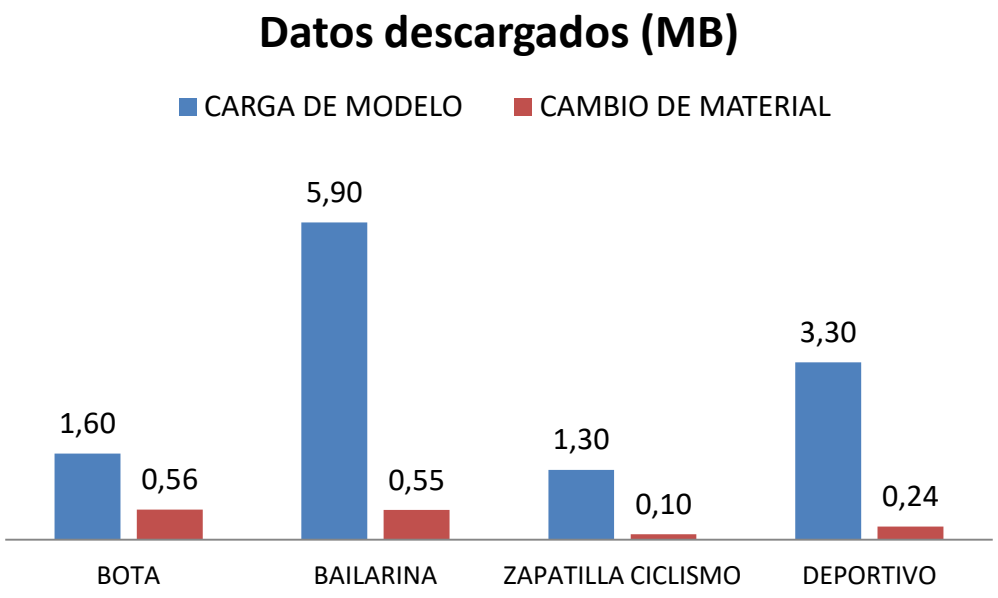

Gráfico 2: Comparativa del volumen de carga entre los diferentes modelos tipo de pruebas.

En cuanto a la información proporcionada por las pruebas de volumen de datos, se puede apreciar también un aspecto claramente diferenciador, éste es el volumen de carga del modelo *BAILARINA*, debido en gran medida a las características de las imágenes utilizadas. Como se puede apreciar en la definición de este modelo tipo, la resolución de las imágenes de render utilizadas es bastante elevada comparado con el resto, lo que le aporta un elevado grado de realismo y detalle, pero que a su vez, lo hacen muy pesado para su utilización en plataformas web que quieran ofrecer un servicio de acceso multi-dispositivo, donde las conexiones no siempre serán las más óptimas. En cuanto al resto de modelos, se puede apreciar claramente en la operación de carga del modelo, cómo influye la resolución de las imágenes de render en el volumen de datos descargados de cada uno de ellos, independientemente de la complejidad de materiales y reglas de combinación del modelo (al contrario de cómo sucedía en las pruebas de tiempos de carga con el modelo *BOTA*).

Otro aspecto importante a destacar, es la reducción drástica de los datos descargados en la operación de cambio de material, gracias a la optimización de la estructura de datos desarrollada e información que proporciona el sistema CAD, puesto que para esta operación, únicamente se descarga la pequeña parte del modelo que se ve afectada por el cambio. Esto repercute directamente en el volumen de datos descargados y en los tiempos de carga (como también se podía apreciar en la *Gráfica 1*).

Finalmente, gracias a estas pruebas se ha podido certificar la importancia que tiene la resolución de las imágenes de render del modelo virtual, por lo que éste puede ser un aspecto crucial a definir a la hora de diseñar los modelos y su posterior explotación en la web. Estas pruebas, también ponen de manifiesto el acierto en el modelo de datos desarrollado, consiguiendo una mejora más que notable en las posteriores operaciones sobre el modelo, una vez descargado inicialmente.

4.3 Pruebas de rendimiento del servidor

Por último, se van a analizar los resultados obtenidos de las pruebas enfocadas a la evaluación del rendimiento del servidor, donde la definición de pruebas en JMeter y la interacción simultánea con el comando *htop* de Linux en el servidor, han sido los elementos fundamentales en la obtención de los datos que se presentan.

Por lo tanto, en cuanto a la realización de los plantes de prueba en JMeter, se han aprovechado las pruebas definidas en las fases anteriores, pero además, se han desarrollado para poder simular diferentes conexiones simultáneas de un gran número de usuarios y permitiendo la selección de un modelo o varios a la vez. Gracias a esto, se han podido establecer pruebas para ver el rendimiento del servidor con diferentes cargas de usuarios, aplicadas en los distintos modelos por separado y con todos a la vez. En las siguientes tablas se muestran los resultados obtenidos:

Configuración Servidor 1				
MODELO	**USUARIOS**	**CARGA CPU's (%)**	**TIEMPO (seg)**	**ERRORES**
BOTA	50	19,30	10	0
	100	38,60	10	0
	1000	100,00	31	321
BAILARINA	50	6,80	10	0
	100	12,80	10	0
	1000	100,00	13	0
ZAPATILLA CICLISMO	50	6,30	10	0
	100	11,90	10	0
	1000	100,00	12	0
DEPORTIVO	50	6,30	10	0
	100	11,80	10	0
	1000	100,00	13	0
TODOS	50	10,80	10	0
	100	19,70	10	0
	1000	100,00	25	8

Tabla 5: Información de rendimiento del servidor con Configuración Servidor 1 en pruebas de carga.

Configuración Servidor 2				
MODELO	**USUARIOS**	**CARGA CPU's (%)**	**TIEMPO (seg)**	**ERRORES**
BOTA	50	8,10	10	0
	100	16,60	10	0
	1000	100,00	25	5
BAILARINA	50	2,90	10	0
	100	4,70	10	0
	1000	51,70	10	0
ZAPATAILLA CICLISMO	50	4,10	10	0
	100	11,20	10	0
	1000	52,10	10	0
DEPORTIVO	50	3,20	10	0
	100	4,60	10	0
	1000	52,40	10	0
TODOS	50	4,10	10	0
	100	8,20	10	0
	1000	90,00	10	0

Tabla 6: Información de rendimiento del servidor con Configuración Servidor 2 en pruebas de carga.

Pese a la gran cantidad de datos que presentan estas tablas, se pueden deducir varios aspectos que ya revelan cómo afectan las pruebas al rendimiento del servidor. El primero de ellos es la

saturación del servidor cuando se realizan simulaciones masivas, de hasta 1000 usuarios, accediendo exactamente al mismo tiempo y al mismo recurso. En todos estos casos, para la *Configuración Servidor 1*, se somete la carga de la unidad de computación (CPU) del servidor al 100% de su capacidad, por lo que se experimentan tiempos de respuesta mayores, incluso como también se puede observar, en ocasiones se producen errores de espera de la petición al servidor, debido a esta saturación del servidor. En cuanto a la *Configuración Servidor 2*, se aprecia también esa sobrecarga del servidor, aunque sólo se llega a sobrecargar en el caso del modelo de prueba *BOTA*, que como se demuestra en las pruebas anteriores, es el que más requerimientos exige al servidor.

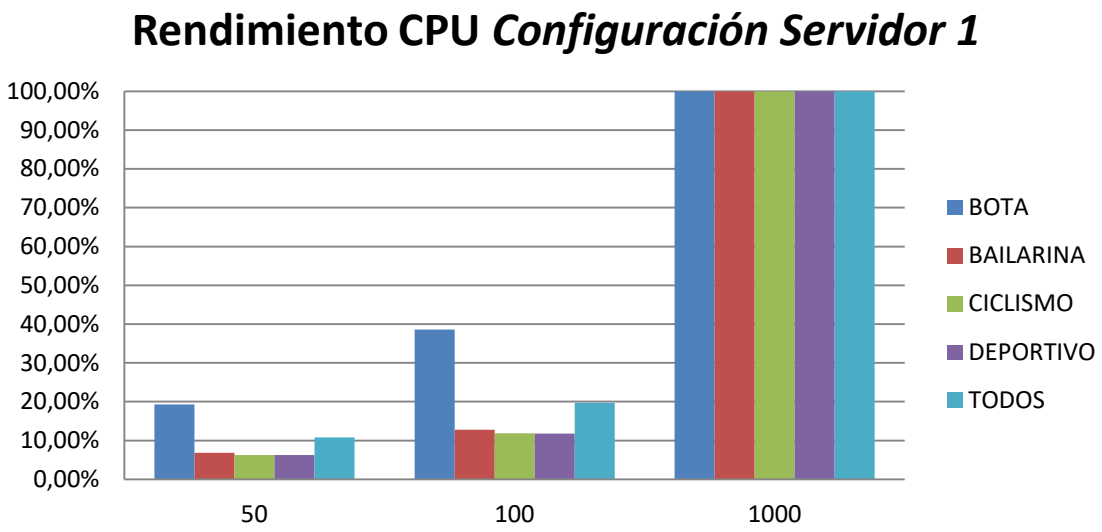

Tabla 7: Comparativa del rendimiento servidor con Configuración Servidor 1 con estrés de carga.

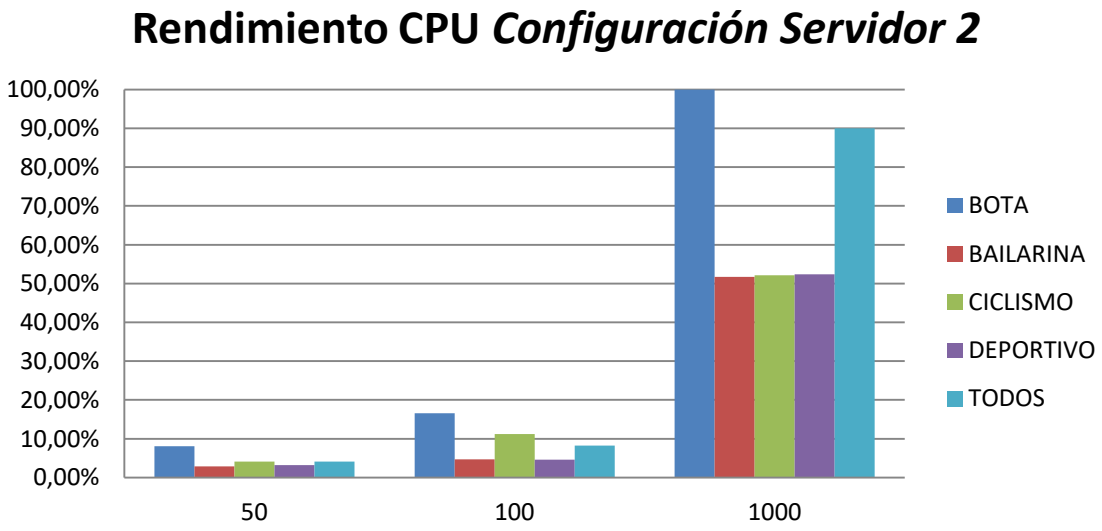

Tabla 8: Comparativa rendimiento servidor con Configuración Servidor 2 con estrés de carga.

Gracias a la representación en gráficos de los datos obtenidos en las pruebas que se han realizado, se puede ver de forma más clara como incide el acceso múltiple de usuarios al rendimiento del servidor. También demuestran la importancia de los datos del modelo virtual utilizado en cada caso. No obstante, como se puede apreciar en la *Tabla 8*, gracias al aumento de las prestaciones del servidor, la carga de sus procesadores se reduce notablemente, lo que repercute directamente en los tiempos de ejecución de las pruebas, consumo de memoria RAM y errores provocados por saturación del servidor.

Finalmente, en lo referente a los procesos de carga del servidor, se han ejecutado las pruebas de estrés para la *Configuración Servidor 3* sólo en el caso del modelo *BOTA*, ya que por sus altas exigencias de procesamiento y cálculo provocaban la saturación del servidor cuando se prueba una tasa elevada de acceso de los usuarios, incluso en la *Configuración Servidor 2* (con el resto de modelos no ha habido ningún problema, por lo que probarlos en esta nueva configuración no aporta ninguna información). Por lo tanto, tras probar con la configuración del servidor con sus prestaciones al completo, se ha podido constatar que aunque se requiere durante algún tiempo del 100% de la capacidad de computación del servidor, se podría afrontar una carga similar a 1000 usuarios conectados simultáneamente en un tiempo menor que los casos anteriores (17 segundos) y sin producirse ningún error en la respuesta a esas conexiones simultáneas.

Por otra parte, también se han realizado una serie de pruebas con el fin de determinar el grado de exigencia y límites de capacidad de servidor en cuanto a memoria RAM se refiere. En toda la batería de pruebas realizada anteriormente este aspecto no era relevante, ya que habría que someter a cargas totalmente irreales para poder llegar a altas exigencias de memoria RAM por parte del servidor.

En consecuencia, para poder analizar esta característica, se realizó un breve análisis de las posibles operaciones que supondrían un mayor uso de memoria RAM por parte del servidor web en las diferentes peticiones realizadas por la plataforma, y se dedujo que la creación o alta de un nuevo modelo virtual en la plataforma suponía la tarea más costosa en cuanto a consumo de memoria RAM se refiere. Para esta operación, el prototipo de plataforma web 2.0 debe descomprimir un fichero CAD que puede tener una gran cantidad de datos (en torno a 190 MB en el caso del modelo *BOTA*) y que además puede requerir un gran número de operaciones de inserción de esta información en la base de datos. Por lo tanto, se procedió a las pruebas de alta de los modelos establecidos, haciendo uso solamente del comando *htop* de *Linux* ya que en este caso no se requería la automatización de ningún proceso, por lo que se realizó de forma manual.

MODELO	USUARIOS	MAX. RAM (GB)
BOTA	1	0,03
BAILARINA	1	0,001
ZAPATILLA CICLISMO	1	0,001
DEPORTIVO	1	0,001

Tabla 9: Datos de utilización de memoria RAM para los diferentes modelos de pruebas.

Como se puede apreciar en la *Tabla 9*, el consumo de memoria RAM requerido por el proceso de alta de un modelo parece no ser muy elevado, siendo de a penas 1MB para casi todos los modelos, excepto para el modelo *BOTA*, que requiere de 30 MB de memoria RAM en esta tarea. Sin embargo, si extrapolamos esta operación a la ejecución de la misma de forma simultánea por diferentes usuarios de la plataforma, se puede determinar que el consumo de memoria RAM se incrementará significativamente. En la *Tabla10* se puede observar esta simulación, que se ha calculado a partir de la primera muestra realizada anteriormente, ya que esta prueba para un gran número de procesos o usuarios, requiere de un ancho de banda que haría imposible la validación simultánea del proceso.

Modelo	USUARIOS	MAX. RAM (GB)
BOTA	50	1,71
	100	3,42
	1000	34,18

Tabla 10: Datos de utilización de memoria RAM para el modelo BOTA.

Por lo tanto, aunque no es una operación que se vaya a repetir constantemente en el uso normal de la plataforma, el alta de un modelo puede llegar a exigir niveles de utilización de memoria RAM elevados. No obstante, no se alcanzan los límites de memoria RAM física de los que dispone el servidor de pruebas, por lo que su rendimiento no se vería afectado en el caso de que se dieran este tipo de situaciones.

5. Conclusiones

Como conclusión al proceso de pruebas de estrés realizadas sobre el prototipo desarrollado y el servidor de pruebas que lo sustenta, se ha podido recopilar y analizar el comportamiento de la plataforma en diferentes situaciones simuladas.

Por un parte, se han simulado los procesos de carga de modelos y cambios de materiales, analizando tanto los tiempos requeridos como los volúmenes de datos descargados según el modelo de pruebas utilizado. Gracias a estas pruebas, se puede determinar a priori como se va a comportar la plataforma en función de las características del modelo virtual que se vaya a utilizar, y así poder decidir como se quiera proceder en cada caso.

Por otra parte, se ha sometido al servidor a situaciones extremas, con el fin de poder analizar cómo responde la plataforma en conjunto ante estas situaciones. Demostrando que cuando se trata de un modelo virtual con una configuración de materiales y reglas de combinación complejas, se puede observar que en los casos más extremos, el servidor se veía sobrecargado, llegando incluso a no poder responder a todas las peticiones que se le solicitaban. Esto es debido a que en este tipo de modelos se realiza un uso exhaustivo de la CPU y acceso a la base de datos del servidor, además de lidiar con todo el proceso de la carga de un gran número de imágenes.

Por consiguiente, se puede concluir que la arquitectura propuesta ofrece unas prestaciones lo suficientemente altas para poder afrontar un proyecto de esta índole, sin que esto suponga un

riesgo elevado sobre la viabilidad de la ejecución del mismo. Además, cumpliendo con una de las premisas planteadas, sobre la escalabilidad de la solución, se ha demostrado en las pruebas cómo la mejora de las prestaciones del servidor de forma dinámica, repercute de forma directa en el rendimiento general de la plataforma, por lo que en el caso de que se llegaran a los límites que se plantean, se podría escalar la plataforma en la medida que se considere necesaria para así garantizar siempre un rendimiento óptimo. De igual modo, aunque no se ha podido verificar en las pruebas del proyecto, sí que es evidente que si el sistema de gestión de base de datos, se separara del servidor web y se incluyera en un servidor de base de datos específico, el rendimiento total del sistema también se vería mejorado. Y para finalizar con las posibles mejoras de los resultados, una vez detectadas las operaciones que más requieren recursos del servidor y gracias a la clasificación de los modelos de pruebas utilizados, también se podría estudiar y analizar como ajustar la estructura y modelo de datos desarrollados, de manera que se mejoren los tiempos y procesos de cálculos de dichas operaciones.

6. Referencias

[TheGrinder] http://grinder.sourceforge.net/

[Gatling] https://gatling.io/

[Tsung] http://tsung.erlang-projects.org/

[Locust] https://locust.io/

[JMeter] http://jmeter.apache.org/