CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Informática

Universidad Politécnica de Madrid
Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Diseño e implementación de una página web para la comparación de algoritmos de optimización

Autor: David Gutiérrez Baeza
Director: Antonio Latorre de la Fuente

MADRID, JULIO 2017

1. INDICE

2. RESUMEN

Español:

Como resumen, he codificado una web con autenticación de diferentes tipos de usuarios, cuyo propósito principal es dar a éstos la posibilidad de subir sus resultados de algoritmos ejecutados sobre diferentes benchmarks y posteriormente, compararlos mostrando los resultados mediante diferentes gráficas que detallan las ventajas y desventajas de un algoritmo frente a otro u otros de diferentes usuarios. Por otro lado, a través de diferentes paneles de gestión pueden darse de alta en el sistema nuevos benchmarks, funciones, tags, usuarios y validar, en caso de ser usuario administrador, los resultados pendientes de confirmación de usuarios.

Se ha necesitado la elaboración y seguimiento de un plan de trabajo, el uso de diferentes editores, lenguajes de programación y tecnologías cuyo estudio ha debido ser llevado a cabo para finalizar satisfactoriamente la realización de este TFG. Por otro lado, en cuanto a las fuentes o repositorios de ayuda, se ha utilizado en mayor medida que otras stackoverflow.com, página especializada, en su mayor parte, en consultoría sobre programación.

Finalizando, el proyecto ha sido llevado a cabo en su totalidad en una máquina virtual con sistema operativo Linux, versión: 16.04 (Xenial Xerus).

Inglés:

As a summary, I have coded a website with authentication for different types of users, which main interest is to allow them to upload their algorithm results that were executed on different Benchmarks and, later, compare them showing the results through different kinds of graphics that explain the advantages and disadvantages of an algorithm compared to those of the algorithms of other users. Furthermore, through different management panels, new benchmarks, functions, tags and can be registered in the system and the algorithms scores with confirmation pending can be validated by the administrator user.

The elaboration and follow up of a work plan has been a requirement, as well as the use of different editors, programming languages and technologies which study has been carried out to complete this TFG successfully. In addition, in terms of help sources or repositories, the one that has been most frequently used was stackoverflow.com, a specialized website, mainly, in programming consultancy.

To conclude, the project has been conducted in a virtual machine Linux OS v.16.04.

NOTA: A lo largo de la memoria se exponen imágenes que apoyan y clarifican lo que en el texto se indica. Las fechas o tamaños de los diferentes ficheros fuente que puedan observarse en ellas pueden no coincidir con el resultado final. Esto es debido a que la memoria se ha ido llevando a cabo a la vez que se programaba y codificaba el proyecto. No obstante, toda la jerarquía, organización y nombres se conservan y por ello las imágenes conservan su importancia.

3. INTRODUCCION Y OBJETIVOS

3.1 INTRODUCCION

En este capítulo se presentará la información y motivación vinculada al desarrollo del actual proyecto de fin de grado, el cual radica en elaborar una web que permita mostrar los resultados procedentes de diferentes benchmarks que comparan las propiedades, eficiencia, eficacia y rendimiento de diferentes algoritmos de optimización.

Los algoritmos de optimización son técnicas de Soft Computing [1], conjunto de metodologías con bases en la biología, psicología y lingüística, que buscan dar una solución a problemas caracterizados por interactuar con sistemas complejos y con información incompleta. Las técnicas de Soft Computing prueban la tolerancia de la precisión y la verdad total, a diferencia de las técnicas de Hard Computing que estudian la verdad parcial y la inexactitud.

Los algoritmos de optimización son implementados para resolver problemas reales de diferente temática que suelen ser de una magnitud elevada, a través de diferentes operaciones ejecutadas de forma lógica y ordenada.

Relacionado con lo anterior, podemos afirmar que, aunque estos últimos años la tecnología ha evolucionado muy rápido permitiendo a los ordenadores llevar a cabo una inmensa cantidad de operaciones por segundo, actualmente existen multitud de problemas que podrían tardar años en resolverse hasta que se consiguiese llegar a una solución válida y óptima. Por esta razón principalmente, se ha observado un auge en cuanto a la investigación, desarrollo y avance tanto de los algoritmos de optimización como de las ramas que los estudian: la optimización real y la optimización combinatoria.

Desarrollar algoritmos de optimización no es una tarea fácil, ya que en la mayoría de las ocasiones se deben tener en cuenta tanto las características como las condiciones necesarias para poder diseñar el algoritmo de forma que consiga llegar a la solución de forma eficaz, eficiente y de la forma más sencilla siempre que sea posible.

Algunas de las aplicaciones [2] donde los algoritmos de optimización tienen gran importancia son:

- Ingeniería química y bioquímica.
- Biología.
- Diseño, optimización estructural e ingeniería.

- Estimación de parámetros.
- Finanzas y economía.
- Comunicación y red.
- Problemas matemáticos.

Los algoritmos evolutivos engloban un conjunto de técnicas de optimización que permiten la búsqueda estocástica de soluciones óptimas, tomando como base los principios de la evolución biológica. Éstos forman parte de la rama de Inteligencia Artificial. Este tipo de algoritmos se configuran y construyen por software intentando reproducir la selección natural y el entrecruzamiento de los seres vivos por medio de mutación y recombinación genética. Para ello, un conjunto de entidades se toma como posibles soluciones, combinándolas entre sí para saber cuál de ellas se adecua más al problema en cuestión y así garantizar la supervivencia de la misma que es el objetivo principal. De esta forma, su evolución a lo largo del tiempo permitirá crear una población de soluciones mucho más adaptada que la anterior. Es importante destacar que estos algoritmos pueden optimizan varias funciones relativas a la supervivencia al mismo tiempo, pero normalmente se especializan en una (función objetivo). Estos algoritmos tienen una disposición característica que les facilita ser aplicados en problemas diferentes. Una de las ramas que más peso tiene en la investigación de estos algoritmos es la LSGO (Large Scale Global Optimization).

Uno de los métodos de búsqueda que usan los algoritmos evolutivos es la Evolución Automática, que se modeliza de acuerdo a ideas que hacen referencia a los cambios evolutivos de las especies, para poder encontrar las mejores condiciones de supervivencia de un ser vivo (mayor posibilidad tanto de habituarse al medio como de reproducirse) y asegurar que éstas se transfieren de generación en generación. De esa manera se garantiza una mayor probabilidad de supervivencia respecto al medio. Por ello se dice que esta rama se encarga de investigar la optimización de la supervivencia.

Las etapas de un algoritmo evolutivo se pueden dividir en: Seleccionar aleatoriamente una población con un número determinado K de sujetos. Posteriormente, elegir la función objetivo que permitirá optimizar la supervivencia, y entonces repetir con cada generación de estos sujetos las siguientes operaciones: escoger, utilizando un método de selección parte de los K sujetos que forman la población, llevar a cabo sobre los mismos un conjunto de operadores genéticos para así formar descendencia, determinar si los descendientes han mejorado con respecto a sus antecesores en la función objetivo y por último se sustituyen los sujetos con menor probabilidad de supervivencia de la población K por los nuevos.

Con todo lo anterior queda reflejado qué son, por qué son importantes, cómo se implementan y cuáles son las aplicaciones de los algoritmos de optimización.

3.2 OBJETIVOS

La web que se tiene como objetivo refleja la comparación de estos algoritmos de optimización a través de un conjunto de benchmarks cuyos resultados se almacenan en una base de datos que se consulta, para después mostrar los resultados procesados en el sitio web de forma gráfica. Por ello, a continuación, se va a hablar sobre qué son los benchmarks y qué aspectos importantes se necesitan comprender de ellos para así entender, gracias a este apartado, el propósito del desarrollo de la página web.

El término benchmark, aunque es aplicable en muchas áreas, se podría entender como aquella aplicación diseñada para calcular el rendimiento de un computador o de algún componente del mismo. Para llevarlo a cabo, se le hace al ordenador someterse contra una serie de cargas de trabajo con el objetivo de medir su respuesta ante ellas. Así se puede predecir contra qué tareas el computador es fiable y eficaz. Lo mismo ocurre con el Benchmarking de algoritmos de optimización. De esta manera se podrá conocer qué algoritmo es mejor que otro en términos de rendimiento para realizar determinadas tareas. Con el Benchmarking también se permite conocer las debilidades que tiene un algoritmo para poder solventarlas y mejorarlas en procesos posteriores de desarrollo.

Es importante conocer que existen diferentes tipos de benchmarks, entre los cuales dos de los más importantes son:

- Medidas de rendimiento bajo cargas de trabajo: se encargan de medir el rendimiento de alguna parte determinada o de una función específica dentro de la misma del algoritmo de optimización.
- Medida de rendimiento por simulación de escenario: se encargan de medir el rendimiento sometiendo al algoritmo a tareas habituales de un campo de aplicación determinado.

Los objetivos propuestos para este trabajo de fin de grado han sido los siguientes:

- Legitimar, registrar y probar todas las actividades que contemplan el desarrollo del TFG, mediante documentación apropiada.
- Codificar la base de datos de tal manera que sea lo más autónoma posible en cuanto a disparadores se refiere, dentro de los límites de las funciones que están disponibles desde el front-end de la web, la web permite añadir benchmarks,

funciones y tags (están codificados, entre otros, triggers de eliminación de benchmarks y actualización de las funciones que componen un benchmark, que permiten respectivamente, eliminar la tabla que almacena los resultados de los diferentes algoritmos del benchmark eliminado, o actualizar el campo funciones de un benchmark si se ha eliminado o añadido una nueva función, cambio que desembocará en una adición o eliminación de la función en la tabla de resultados de ese Benchmark). Por lo que si en un futuro se desarrollan nuevos paneles en el front-end que permitan llevar a cabo estas operaciones, la base de datos no deberá ser modificada, pues ya están contemplados.

- Diseño de la web intuitivo, correcto y eficaz a través de las directrices dadas por el tutor.
- Implementar el back-end y el front-end del sitio web:
    - Usando HTML, CSS y JavaScript.
    - Utilizando R y el paquete Shiny.
    - Empleando LaTeX para la generación de documentos recopilatorios de resultados en formato PDF.
- Comprobar y validar los objetivos anteriores mediante pruebas/diagnósticos.

Dicho lo anterior queda explicada la importancia y la causa de porqué es importante el proyecto en cuestión.

4. REQUISITOS DE LA APLICACIÓN

En cuanto a los requisitos de la aplicación, éstos se podrían definir en:

RE1. Diseñar, estructurar e implementar una aplicación web interactiva, en su mayor parte, con los diferentes tipos de usuarios.

RE2. El contenido y presentación de la misma debe ser lo más sencillo, claro y fluido posible para el usuario ya sea administrador o encargado de subir resultados para su posterior comparación.

RE3. Delimitación de funciones de usuarios administrador e invitado.

RE3.1. Administrador:

RE3.1.2. Añadir benchmarks.

RE3.1.3. Añadir funciones.

RE3.1.4. Añadir tags.

RE3.1.5. Gestionar las validaciones de los resultados.

RE3.1.6. Subir resultados.

RE3.1.7. Comparar.

RE3.2. Registrado:

RE3.2.1. Añadir benchmarks.

RE3.2.2. Añadir funciones.

RE3.2.3. Añadir tags.

RE3.2.4. Subir resultados.

RE3.2.5. Comparar.

RE3.3. Invitado:

RE3.3.1. Subir resultados.

RE3.3.2. Comparar.

RE4. El sitio web debe ser ajustable a la ventana del navegador.

RE5. Uso de una tecnología que permita satisfacer el requisito anterior: Bootstrap en mi caso.

RE6. Uso de HTML para estructurar el contenido de la web y CSS para especificar el formato de ese contenido (utilización de clases Bootstrap).

RE7. Estructurar las diferentes páginas del sitio web.

RE7.1. Página principal:

RE7.1.2. Visualización del departamento, universidad y tecnologías fundamentalmente usadas en el desarrollo del proyecto, así como una breve descripción del mismo.

RE7.2. Páginas de alta y registro de usuarios:

RE7.2.1. Formulario breve en ambos casos.

RE7.2.2. Login o alta:

RE7.2.2.1. Nombre de usuario y contraseña.

RE7.2.3. Registro:

RE7.2.3.1. Nombre de usuario, contraseña y tipo de usuario.

RE7.2.3.2. (Opcional) Ofrecer la posibilidad de cambiar contraseña y tener un panel de control a cada usuario registrado.

RE7.3. Página de altas de datos (benchmarks, funciones y tags):

RE7.3.1. Todas con un botón de confirmación.

RE7.3.2. Widgets que sustentan los campos de las diferentes tablas en la BD, usando el tipo más óptimo para cada caso. Checkboxes, desplegables con caja de detalle, etc.

RE7.3.3. Barra lateral (soporte JS) para moverte por este tipo de páginas de subida de información.

RE7.4. Página de validación:

RE7.4.1. Widget que permita seleccionar el algoritmo y benchmark a validar.

RE7.4.2. Posibilitar la visualización de esos datos aún sin validar con una tabla que a su vez facilite el filtro de los mismos.

RE7.4.3. Enlaces de descarga de la documentación o código que habilitan la confiabilidad del algoritmo.

RE7.5. Página de subida de datos:

RE7.5.1. Widget para seleccionar la plantilla del benchmark del cual tenemos resultados para subir información.

RE7.5.2. Descarga de dicha plantilla.

RE7.5.3. Subir los resultados de dicha plantilla seleccionando el benchmark (por omisión es el mismo que se marca en la opción de descargar plantilla).

RE7.5.4. (Opcional) Adición de ficheros de documentación y/o código.

RE7.5.5. Caja de información con el código (descripción) del resultado de esa operación.

RE7.6. Página de comparación:

RE7.6.1. Widgets para seleccionar los benchmarks, dimensiones y tiempos de evaluación de los algoritmos que se almacenan en dichas pruebas de rendimiento.

RE7.6.2. Dar información, mediante código HTML dinámico, antes de la tabla o gráfica que detalla cada análisis.

RE7.6.3. Ofrecer enlaces de descarga de las tablas y gráficas generadas de cada una de las pruebas.

RE7.6.4. Descargar con la ayuda de un botón un fichero comprimido con todas las tablas y graficas generadas, resultado de la comparación.

RE8. Hacer uso del lenguaje R para codificar las distintas aplicaciones incrustadas en la web.

RE9. Utilizar el paquete Shiny para dotar a R de un conjunto de funciones que permiten generar interfaces web de usuario y procesar las interacciones en el lado servidor de la aplicación.

RE10. Hacer uso de un SGBD con soporte para vectores y que su tratamiento sea claro y efectivo.

RE11. Diseño minimalista de la web.

Por último, desarrollaré un diagrama de interacción entre los diferentes sistemas que intervienen en el funcionamiento de la aplicación.

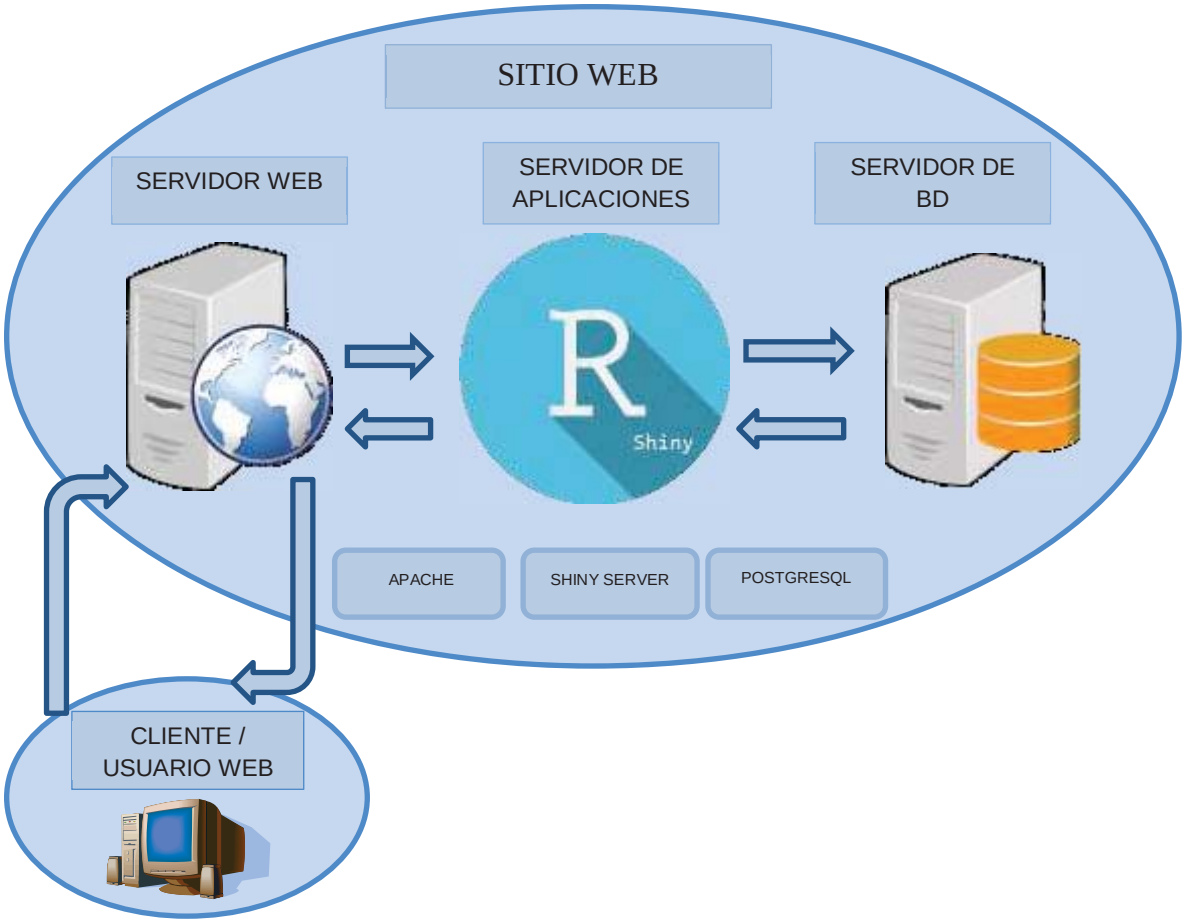


Figura 1

En los siguientes capítulos de la memoria haré una breve descripción de aplicaciones existentes que tienen la optimización como base, comentaré las diferentes pruebas, funciones y algoritmos sobre los que se tiene pensado orientar el proyecto, así como de los diferentes sistemas que componen la aplicación (Figura 1), citaré además las dificultades encontradas en el transcurso del proyecto y las conclusiones. Por último, mostraré una serie de resultados a modo de prueba de cada una de las aplicaciones desarrolladas.

5. TRABAJOS PREVIOS DE LA MEMORIA

A lo largo de las dos secciones que abarcan este capítulo se contaran las tareas que se llevaron a cabo antes de comenzar a desarrollar cualquiera de los sistemas que componen la aplicación. La primera de ellas consiste en investigar en la web un conjunto de herramientas que hagan uso de métodos y técnicas de optimización y ver qué comportamientos o problemas intentan optimizar. La segunda sección trata de introducir y visualizar los elementos y la estructura de una base de datos que va a recoger información de acuerdo a algoritmos, benchmark y funciones, tomando como referencia algunos ejemplos de diferentes sesiones que son para la LGSO (rama que investiga los algoritmos evolutivos -Large Scale Global Optimization-) fuentes importantes de información.

5.1 EJEMPLOS DE APLICACIONES RELACIONADAS

En este apartado se hará referencia a aquellos trabajos similares al proyecto actual, por lo que se estará buscando en la web, aplicaciones que permitan algún tipo de comparación de algoritmos. Una de las primeras tareas aparte de documentarme sobre los temas que conciernen al presente proyecto fue investigar acerca de estos trabajos relacionados.

Lo primero que hay que destacar es que en sí mismo, no existe ningún proyecto que tenga las mismas funcionalidades que el presente. Actualmente, a través de las diferentes búsquedas que he realizado (“algorithm comparison web application, app to compare algorithms, database comparison algorithm app”, entre otras), he podido encontrar cuatro aplicaciones web que guardan cierta relación con el tema que nos ocupa, las cuales listaré un poco más adelante.

Casi todas las búsquedas daban resultados sobre documentos y artículos que trataban de investigaciones en las que se exponían e interpretaban las características de cada algoritmo. Además, éstos eran en su mayoría de aprendizaje automático.

A continuación, citaré, de menor a mayor, aquellas aplicaciones que más parecido guardan con la temática que nos ocupa:

- En esta primera herramienta de visualización [3], se puede ver la forma de operar de diferentes estructuras de datos y algoritmos, en su mayoría, de ordenación. Esta utilidad está escrita en JavaScript y HTML5 que además es compatible con dispositivos iOS.
- Con respecto a la anterior, esta herramienta [4] es bastante parecida, la cual ilustra la eficacia con la que diferentes conjuntos de datos desde diferentes puntos de comienzo pueden ser ordenados usando diferentes algoritmos.

- La tercera [5] y cuarta aplicación son las que mayor relación presentan con el proyecto. En cuanto a la tercera, está escrita en R y trata de medir el desempeño de los algoritmos disponibles comparando los puntos de datos sin clasificar de diferentes clasificadores como se predijo en el conjunto de entrenamiento (50%) del conjunto de datos "Iris". Se detallan, por cada uno de los algoritmos, los puntos de datos que no fueron clasificados correctamente con respecto a los que sí lo fueron.
- Por último, con esta plataforma web [6] se puede verificar los resultados obtenidos de los algoritmos de aprendizaje atribuyendo las pruebas estadísticas a los experimentos, lo cual permite apoyar el proceso de toma de decisiones (el algoritmo más adecuado para cada caso). Esta aplicación permite subir a los usuarios sus propios datos para realizar diferentes test con distintos parámetros. Es de destacar que también permite subir más de un grupo de datos.

5.2 INFORMACION RELEVANTE DEL PROYECTO ACTUAL

A continuación, describiré algunos de los algoritmos, los test o benchmarks, las funciones de las que se componen, y algunas gráficas comparativas, consultando para ello el documento: [7].

La motivación, de la cual se ha hablado levemente en el apartado anterior, reside en que actualmente se llevan a cabo estudios de algoritmos sobre un benchmark en particular, permitiendo conocer de éstos características relativas a las funciones de ese benchmark, pero que nos llevan a un desconocimiento sobre otras posibles cualidades de los algoritmos si se ejecutaran sobre más benchmarks especializados.

Para la LSGO, una de las principales fuentes de información son las sesiones especiales que cuentan con sus propios benchmarks que se llevan a cabo cada cierto tiempo, ejemplos de estas sesiones son las que reciben el nombre: CEC (2008, 2010, 2012, 2013) y SOCO (2011), donde se testearon alrededor de 20 algoritmos distintos para los mismos benchmarks aunque aparecieron algunos más en los que también fueron medidos, lo cual puede provocar fragmentación e inseguridad en los resultados de los diferentes algoritmos.

5.2.1 BENCHMARKS

En la siguiente sección hablaré brevemente sobre los Benchmarks más relevantes de la actualidad y que participaron en estas sesiones, a modo de entender las características y cualidades que deben poder ser almacenadas en una BD destinada a guardar información de algoritmos que se someten a estos test o pruebas.

5.2.1.1 CEC2010 [8]

El benchmark CEC2010 está compuesto por veinte funciones continuas de optimización diferenciadas en cuatro grupos y todas ellas definidas con dimensión 1000:

- Funciones separables.
- Funciones parcialmente separables, donde una parte de la función está formada por variables dependientes y otra por variables independientes.
- Funciones totalmente no separables.

5.2.1.2 SOCO2011 [9]

Este benchmark está compuesto de un menor número de funciones continuas de optimización que el anterior (en concreto 19), todas ellas completamente escalables. A SOCO2011 se le fueron incorporando funciones desarrolladas en otras sesiones o proyectos (las primeras 11). Las últimas funciones fueron construidas combinando dos funciones a lo largo de las 11 primeras.

5.2.1.3 CEC2013 [10]

Este benchmark se compone de 15 funciones, divididas al igual que el primero, en cuatro grupos diferentes:

- Funciones totalmente separables.
- Funciones parcialmente separables.
- Funciones con solapamiento de subcomponentes (dos tipos de subcomponentes –conformes y conflictivos-).
- Una función no separable.

5.2.2 EJEMPLO: GRAFICA COMPARATIVA

A continuación mostraré una imagen del artículo publicado en el diario ElServier [7] que muestra un promedio de los resultados obtenidos por los diferentes grupos de funciones para todos los algoritmos:

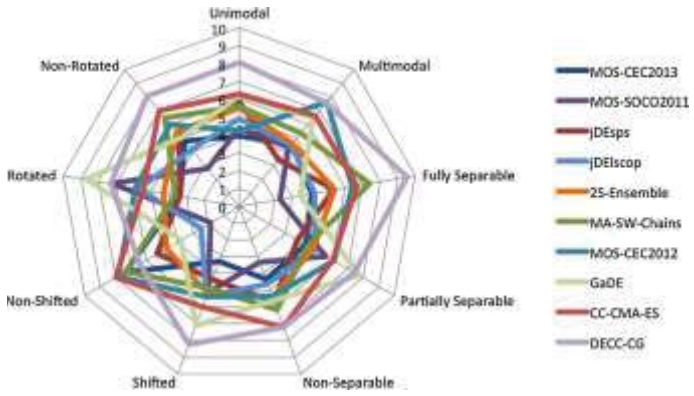


Figura 2

Este tipo de gráfica será una de las que se tiene como objetivo implementar en una de las aplicaciones, que se encarga de cotejar y generar tablas y gráficas comparativas entre diferentes algoritmos, ejecutados sobre distintos benchmarks.

6. DESARROLLO

En este capítulo se comentará todo aquello relacionado con la implementación de los diferentes sistemas que permiten el funcionamiento de la web: base de datos, front-end y back-end, cada uno de los cuales tendrá su propia sección.

6.1 BASE DE DATOS

Para poder desarrollar una web que permita, como principal función, la comparación de diferentes algoritmos de optimización será necesario, poder almacenar determinada información, la cual explicaremos en detalle en los siguientes apartados, relativa a benchmarks o pruebas de rendimiento, funciones por las que éstos se componen y los propios resultados de los algoritmos que se ejecutan contra dichos test.

6.1.1 ESPECIFICACION DE REQUISITOS DE LA BASE DE DATOS

El diseño de la base de datos debe permitir guardar toda la información que desde el front-end, ya sea si se es un usuario administrador, un usuario registrado, o un invitado (anónimo), se pueda introducir (también sería necesario almacenar la identificación de cada usuario, de lo cual hablaremos en el siguiente apartado). Teniendo todo lo anterior en cuenta, los requisitos de la base de datos a codificar serían:

- Definición de benchmarks:
    - Nombre del benchmark.
    - Descripción de lo que permite evaluar el benchmark según sus funciones e instantes de evaluación.
    - Instantes de evaluación sobre cada función. Es razonable que, cuando se define un nuevo benchmark, se especifiquen aquellos instantes de evaluación en donde se va a determinar la cercanía del algoritmo a la búsqueda de la solución.
    - Número mínimo de repeticiones que se ejecutará el benchmark para cada algoritmo. Esta información nos permite conocer la sensibilidad del algoritmo (variaciones en la búsqueda de una solución óptima por parte del algoritmo en las diferentes repeticiones).
    - Capacidad para relacionar el benchmark con todos los resultados del mismo (tabla resultados).

- Funciones:

    - Identificador de la función.
    - Nombre de la función.
    - Siglas.
    - Descripción de la función que permita conocer qué tarea o tareas realizan la misma con respecto al benchmark.
    - Valor óptimo al que el algoritmo debe acercarse para corroborar su eficacia.
    - Dimensión mínima y dimensión máxima que especifiquen la magnitud de los arrays de valores con los que trabaja la función.
    - Rango mínimo y rango máximo: limitan los valores que pueden tomar las diferentes posiciones del vector con el que trabaja la función.
    - Tags (etiquetas) o características que permiten agrupar a las funciones.

- Resultados:

    - Identificador del resultado.
    - Siglas del algoritmo sobre el que se ha ejecutado el benchmark.
    - Nombre del algoritmo sobre el que se ha ejecutado el benchmark.
    - Dimensión de las funciones.
    - Instante de evaluación (ya se ha explicado su significado).
    - Siglas de cada una de las funciones que componen el benchmark, donde se ubicarán los valores que ha obtenido el algoritmo sobre las diferentes funciones con una dimensión determinada (dentro del rango) y en un instante determinado.
    - Id de la sesión del usuario que sube resultados. Permite diferenciar distintos usuarios anónimos que suben resultados temporales a la web. Además de usarlo como parte de la ruta, que la hace única, en donde se almacenan diferentes archivos que el usuario usa o al que van orientados (plantilla de resultados, archivos opcionales al subir resultados y ubicación temporal de las gráficas y tablas resultantes de una comparación realizada por dicho usuario).
    - El resultado está o no validado.
    - Ruta donde se han almacenado los ficheros opcionales: código y/o paper.

- Posibilitar las diferentes comparaciones:

    - Comparar dos o más algoritmos frente a un benchmark.
    - Comparar dos o más algoritmos frente a dos o más benchmarks.
        - Si se tienen dos benchmarks con 8 y 21 funciones respectivamente, entonces se comparan los resultados obtenidos por cada algoritmo para las 8+21 funciones, pudiendo comparar también los algoritmos a través de los resultados obtenidos para un grupo determinado de funciones que tengan unas características u otras (tags o características de la función).

- La comparación debe habilitarse únicamente cuando dos o más algoritmos de entre dos o más benchmarks son seleccionados.
- Recoger la información pertinente al usuario: nombre, contraseña (se almacena cifrada) y rol (tipo de usuario: administrador o registrado –si no está en la BD es invitado-).
- Las tablas que almacenan los resultados de cada benchmark se crean cuando el usuario administrador o registrado dan de alta uno nuevo, con las columnas pertinentes a las características de ese benchmark.
- Cuando un usuario sube resultados de un algoritmo que ha ejecutado sobre un benchmark, éstos se añadirán como nuevas filas a la tabla Resultados de ese benchmark, guardando los parámetros de ejecución de ese resultado y el comportamiento obtenido por el algoritmo.

6.1.2 DISEÑO DE LA BASE DE DATOS

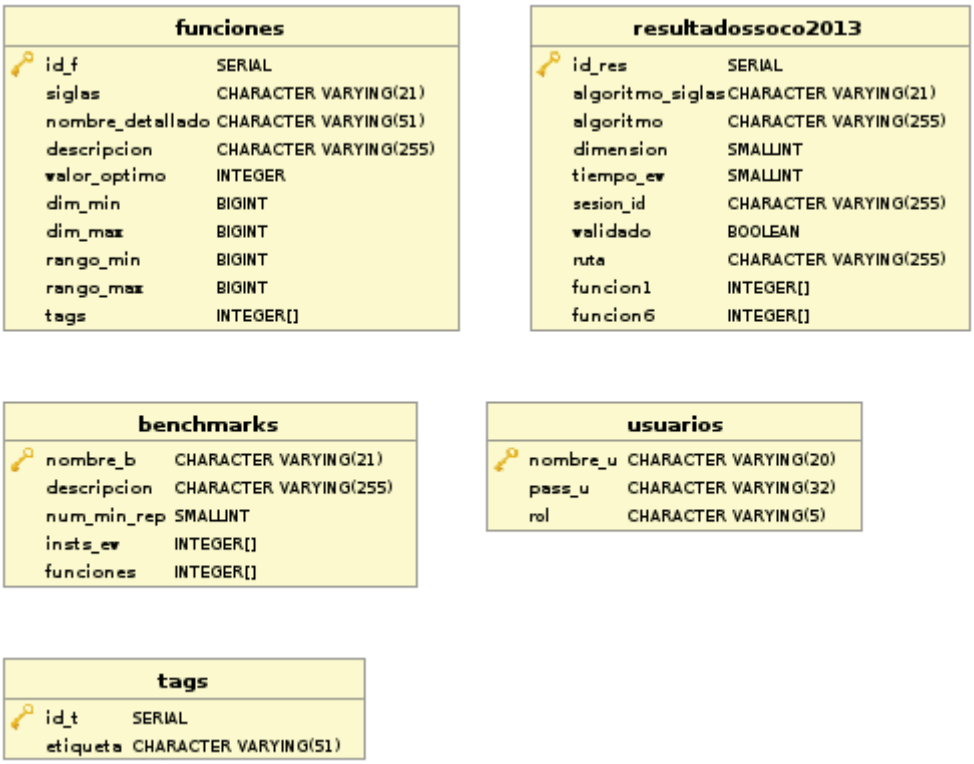


Figura 3

En esta figura se pueden contemplar los cinco tipos de tablas definidas en la BD: benchmarks, funciones, tags, resultados (en este caso del benchmark SOCO2013) y funciones.

NOTA: el tipo de dato SERIAL es un contador que permite no tener que definir dicho campo en la sentencia INSERT. Así se evita, entre otras cosas, colisiones entre identificadores.

6.1.2.1 PROBLEMAS ENCONTRADOS

A continuación, describiré brevemente algunos de los problemas más relevantes que han determinado la modificación de diseños previos de la base de datos.

- Diseño inicial de las tablas que almacenan los resultados de los algoritmos poco eficiente. Este diseño estaba orientado a columnas con valores únicos, por lo que

para almacenar resultados de un algoritmo en las diferentes funciones que conforma un benchmark, era necesario insertar nxm filas, tantas como funciones e instantes de evaluación tenga ese benchmark. Este diseño no es eficaz, tanto por la cantidad de tamaño con información redundante necesaria para identificar cada resultado, como por la velocidad a la que se encuentra un resultado con determinados parámetros.

- Inconveniente similar al anterior por el uso de campos que sólo pueden almacenar un valor. Las funciones pueden estar compartidas por más de un benchmark, y debido al requisito de identificar qué funciones contiene un benchmark determinado, es necesario introducir un campo en la tabla Funciones que permita llevar a cabo esta acción. El hecho de tener una función (fila) replicada porque otro benchmark también la usa no es óptimo.
- Por las anteriores dificultades y problemas derivados, se definieron campos de tipo vector, que permitieran recoger información sobre las funciones que contiene un benchmark o las etiquetas (tags) asignadas a una función de una manera mucho más eficiente. Este requisito fue trascendental a la hora de escoger PostgreSQL [11], un gestor de base de datos que maneja vectores.

6.1.2.2 DISPARADORES ASOCIADOS A LAS TABLAS

Un trigger o disparador consiste en un conjunto de acciones que se definen sobre una tabla de la base de datos, y que entran en ejecución ante determinadas modificaciones de la misma. Estos disparadores llaman a una función que realiza dichas acciones.

Este apartado queda reservado para describir los triggers asociados a cada tabla en el diseño anteriormente mostrado.

Triggers asociados a la tabla benchmarks:

- comprobaract: Después de insertar un benchmark se crea la tabla de resultados correspondiente. Llama a la función: creartabla().

- comprobarelim: Después de eliminar un benchmark se elimina su tabla de resultados. Llama a la función: eliminartabla().

- comprobarfunciones: Antes de insertar un benchmark se comprueba si las funciones que se le están asignando al benchmark existen en la tabla funciones. Ejecuta la función: checkeafunciones().

- comprobarfuncionesact: Antes de actualizar (update) la tabla benchmarks, se comprueba si existen las nuevas funciones asignadas al mismo. Tras una segunda implementación de la función a la que llama este trigger: "comprasingfunciones" (la más laboriosa de todas), se dotó al sistema de grandes funcionalidades haciéndolo casi autónomo: cuando se actualiza la fila correspondiente a uno o varios benchmark,s este trigger se lanza y permite comprobar si el campo funciones ha sido modificado. Si no lo ha hecho (se ha modificado otro campo) no realiza comprobación alguna. En cambio, si se ha eliminado una o más funciones se procede a la eliminación de las columnas correspondientes a éstas en la tabla de resultados del benchmark modificado. Por otro lado, si se han añadido nuevas funciones, se comprueba si éstas existen y además se insertan en la tabla resultados del benchmark en cuestión y así poder almacenar los resultados correspondientes a estas nuevas funciones. Este update (borrado), puede ser generado al eliminar una función desde la tabla de funciones, ya que el trigger que se lanza en la misma elimina de la tabla benchmarks todos aquellos que poseían esa función.

RESTRICCION: La función asociada a este trigger solo borra de uno en uno, mientras que es capaz de añadir más de una función.

Triggers asociados a la tabla funciones:

- comprobartags: Antes de insertar una función se comprueba si los tags que se le asignan a ésta existen en la tabla tags. Ejecuta la función: checkeatags().

- comprobartagsact: Antes de actualizar (update) la tabla funciones se comprueba si existen los nuevos tags asignados a la función. Llama a la función: checkeatags().

- eliminarfuncs: Después de eliminar una función se eliminará ésta en todos aquellos benchmarks que la tenían asociada. Llama a la función: eliminarfuncs().

Trigger asociado a la tabla tags:

- eliminartags: Después de eliminar un tag éste se eliminará en todas aquellas funciones que lo poseían. Llama a la función: eliminartags().

6.2 FRONT-END Y BACK-END

Las dos secciones que siguen: front-end y back-end, describirán de la forma más clara posible los componentes que los forman. En el caso del primero (front-end), mostraré los elementos web que dan diseño a las páginas, la relación de los diferentes ficheros HTML y la aplicación Shiny que la complementa (en caso de que esa página tenga una asociada) y algunos aspectos destacados que merecen ser mencionados para algunas de las páginas que conforman la web. En lo que respecta al back-end me limitaré a explicar la función o funciones que se llevan a cabo en cada fichero R junto con aquellos ficheros PHP que también realizan operaciones características de back-end.

Por enfatizar, vale la pena decir que el front-end (lado cliente) [12] se refiere a todo aquello que se presenta al cliente y con lo que éste interactúa a través del navegador. Éste se encarga de procesar el código de diferentes tecnologías, principalmente tres: HTML, CSS y JavaScript, incluyendo en esta última la biblioteca JQuery, la cual está adquiriendo actualmente mucha relevancia. Por ello, los principios de diseño y de estructuración de la página, gestionados a través de estas tecnologías, permiten establecer un alto grado de usabilidad de la web, lo que lleva al desarrollo del front-end a ser una parte primordial en la que concentrarse para que el usuario se sienta cómodo con ella y no abandone el uso de la misma precipitadamente.

Por otra parte, se conoce como back-end (lado servidor) [12], todos aquellos ficheros que engloban un conjunto de operaciones y funciones llevadas a cabo en los momentos precisos, la mayoría de las veces son las propias acciones del usuario a través del front-end las que desencadenan la ejecución de dichas operaciones. Hay multitud de lenguajes de programación y frameworks para desarrollar la parte servidora de la aplicación, entre ellas podemos encontrar:

- Java.
- R (usada en el proyecto actual).
- PHP (usada en el proyecto actual).
- Python.

Además, toda aplicación debe poder almacenar datos de alguna manera por lo que los SGBD y el manejo de los mismos, también forman parte del desarrollo del back-end. Algunas de los SGBD más importantes en la actualidad, son:

- MySQL.
- Oracle.

- PostgreSQL (usado en el proyecto actual, SGBD relacional orientado a filas. Esto es, todas las propiedades de una entidad específica son ubicadas en posiciones contiguas en disco. Mantiene toda la sintaxis válida para MySQL, excepto algunas particularidades del lenguaje SQL apropiadas para este gestor).
- MongoDB (SGBD no relacional. Son aquellos que no se componen de tablas o relaciones).

6.2.1 FRONT-END

Toda la parte de front-end ha sido desarrollada en un software específico libre de la distribución Linux. En concreto el sistema en el que he trabajado es Ubuntu 16.04. También está disponible para Windows de forma gratuita. Este software se llama Brackets y la versión utilizada es la 1.7.0.

6.2.1.1 ESTRUCTRURA DE LA WEB

Seguidamente describiré el propósito de cada una de las vistas que conforman la web acompañado de una captura de la página.

La página de inicio le da al usuario información acerca del contenido de la web y de las tecnologías usadas para su desarrollo. Esta vista puede ser accedida directamente a través de un botón situado en la barra superior horizontal en todo momento. Ver figura 4.

Figura 4

La siguiente vista permite la comparación entre diferentes algoritmos ejecutados sobre dos o más benchmarks, en la cabecera de la página se comentan brevemente los pasos a seguir para realizar una comparación de algoritmos. A esta aplicación que permite ejecutar comparaciones se puede acceder desde las páginas de inicio, validaciones, alta de usuario e inicio de sesión. Es visible por cualquier tipo de usuario. Ver figura 5.

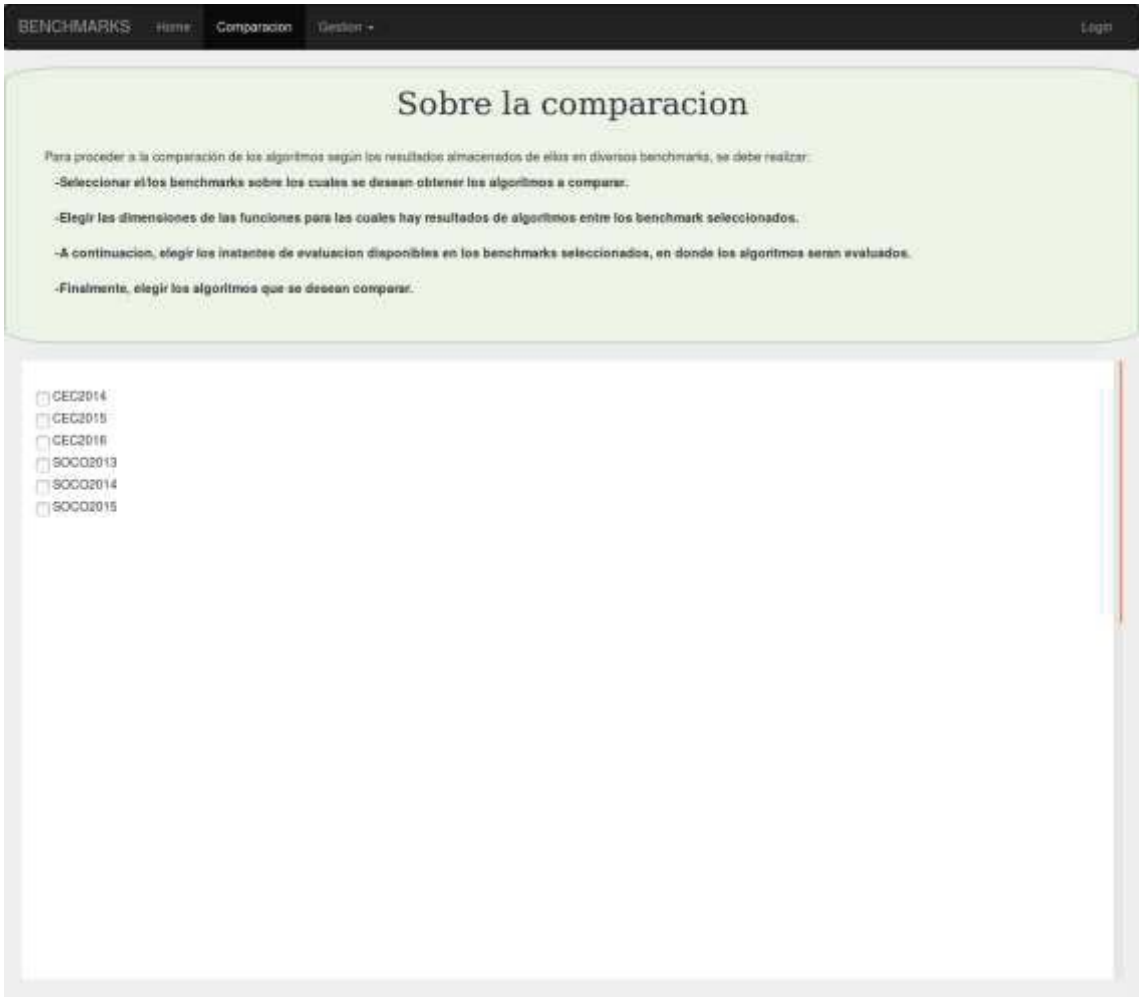


Figura 5

A continuación se muestran los formularios de inicio de sesión, Figura 6, y de alta de usuario, Figura 7. Cuando se inicia sesión, el usuario adquiere un rol y puede ver determinadas páginas de la web, ese rol será: registrado, si es un usuario normal o administrador, si a la hora de registrar un usuario se marca dicha opción. La vista de registro de usuario solamente la pueden ver los administradores, por lo tanto, sólo ellos pueden dar de alta usuarios en el sistema. Se ha omitido la página web completa ya que los formularios perdían bastante calidad.

Figura 6

Figura 7

La siguiente vista es aquella que permite a los administradores validar los resultados de los algoritmos que hayan subido los usuarios, si éstos han suministrado, al menos, uno de los ficheros opcionales: paper o código. Puede ser accedida, si se es usuario privilegiado, desde el último ítem del menú desplegable de la barra superior de la web (Gestión). En esta página, se selecciona a través de un desplegable el algoritmo y el benchmark sobre el que están los resultados. Estos resultados se muestran en una tabla en la parte derecha de la página. La descarga de los ficheros opcionales del algoritmo seleccionado están disponibles justo debajo del botón que ejecuta la validación. Ver figura 8.

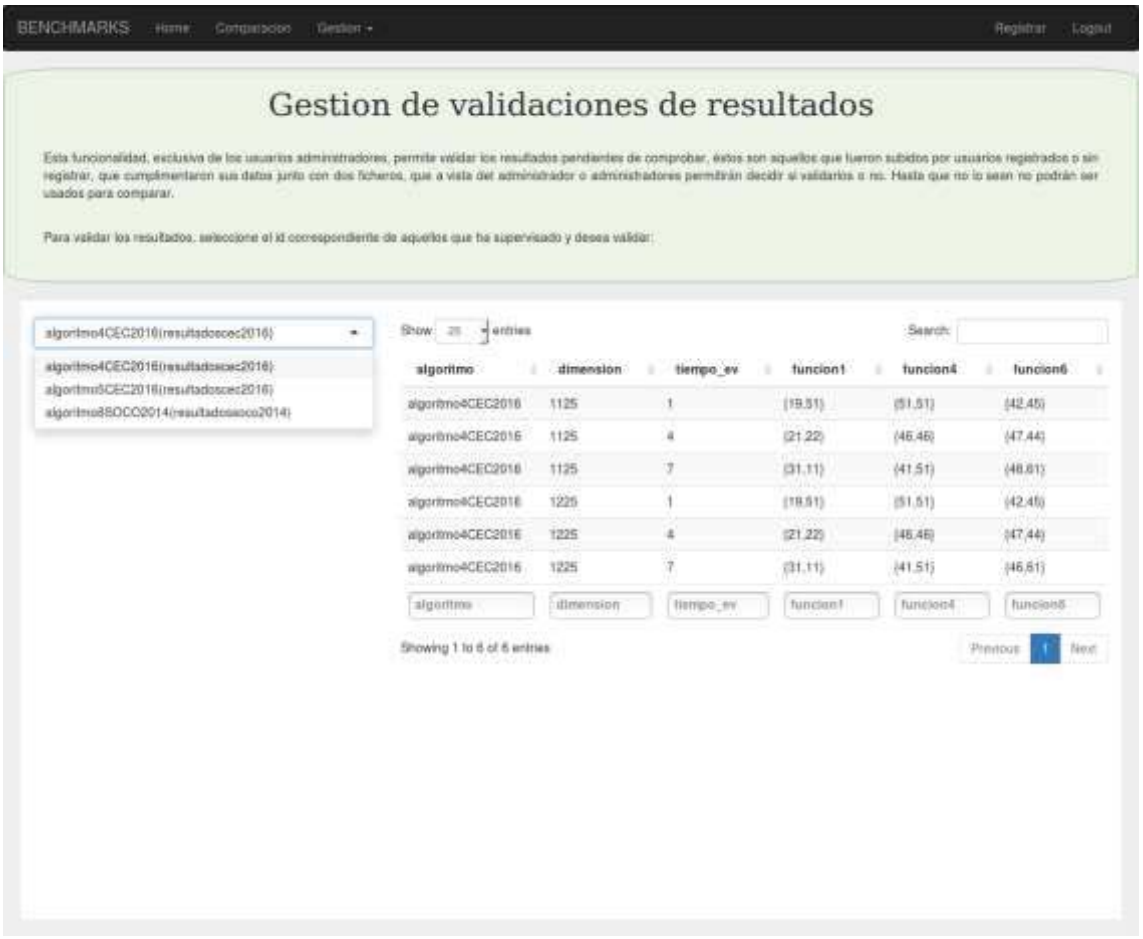


Figura 8

Las siguientes tres y últimas vistas de la web, figuras: 9,10 y 11, posibilitan a los usuarios registrados y administradores dar de alta nuevos benchmarks, funciones y tags en el sistema. Estas vistas pueden ser accedidas por el tipo de usuario que he citado anteriormente y desde la barra superior, en el menú desplegable. Cuando un usuario se encuentra en una de estas páginas, la navegación entre ellas puede hacerse fácilmente a través de un botón ("Mostrar opciones de gestión"), figura 12, que abre una barra lateral para seleccionar si se quiere crear un benchmark, función o tag más, ya que, normalmente, cuando se crea un nuevo benchmark, éste esté lo más seguro compuesto por nuevas funciones que sean caracterizadas por nuevas etiquetas, y de esta forma es más sencillo y rápido la navegación entre estas páginas que permiten llevar a cabo dichas funciones.

Para dar de alta un nuevo benchmark sólo se tienen que rellenar los campos solicitados, el campo instantes de evaluación precisa de ser recordado cada vez que se quiere elegir uno. Para finalizar, se le pulsa al botón confirmar que mostrará en un campo el estado de dicha acción.

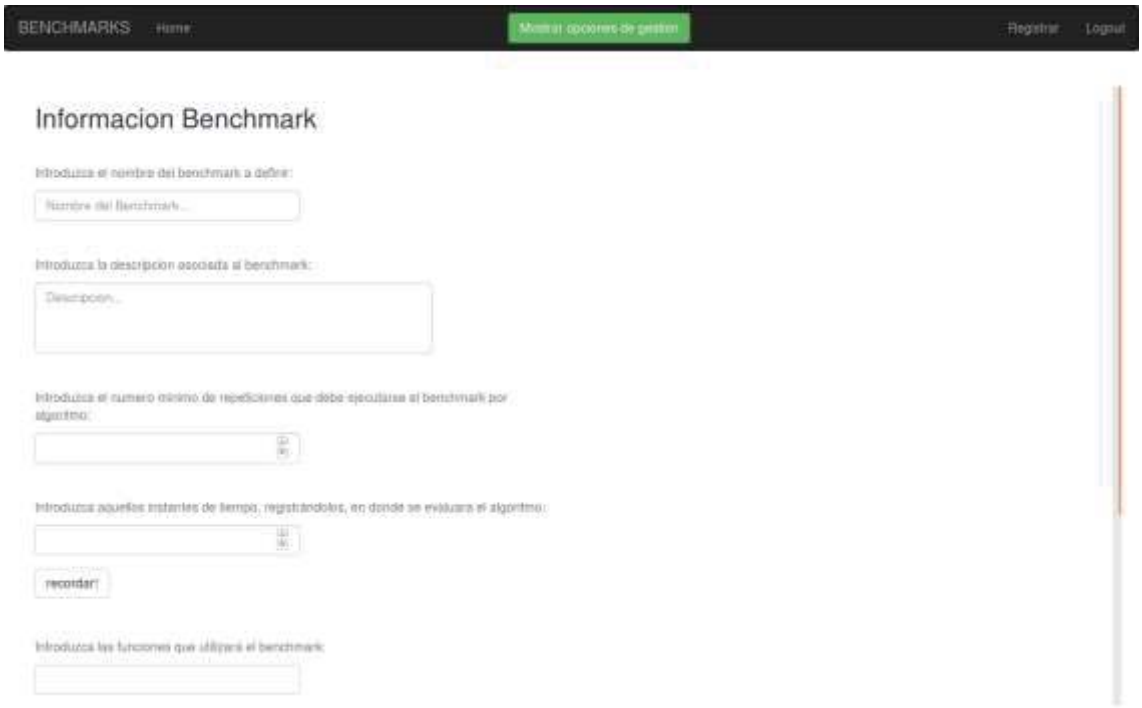


Figura 9

La página que permite subir funciones y tags comparten las características de acceso y visibilidad que la anterior y únicamente se deben rellenar los campos requeridos. Ver figuras 10 y 11.

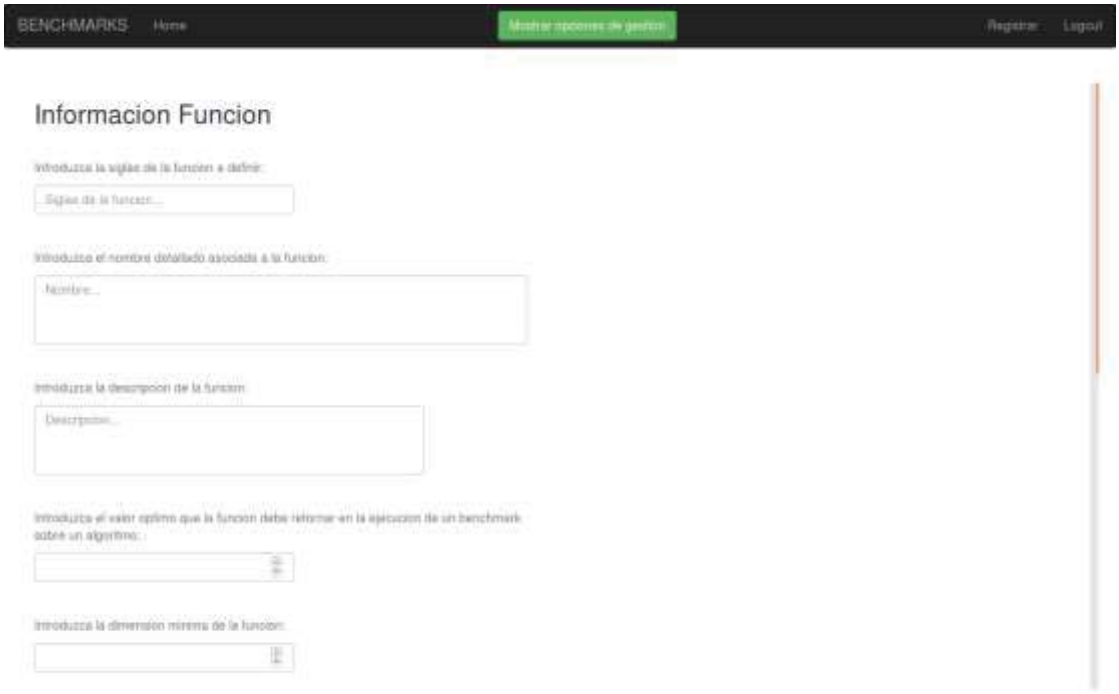


Figura 10

Figura 11

Para finalizar mostraré la barra lateral que permite esconderse y desplazarse desde el lado izquierdo de la pantalla para la navegación entre los diferentes paneles de gestión comentados anteriormente (benchmarks, funciones y tags). Figura 12.

Figura 12

6.2.1.2 ASPECTOS TECNICOS MÁS DESTACABLES

En la cabecera de todos los ficheros que forman el proyecto se enlazan los archivos CSS de bootstrap para permitir que la web sea responsive y se puedan adaptar los diferentes elementos de la misma a diferentes tamaños de ventana del navegador. También se encuentran enlazados los archivos CSS que necesitan las diferentes páginas, en este caso: index.css y barraLateral.css, la cual no es necesaria en este fichero pero que he decidido incluirla en todos para que la web guarde mayor coherencia. Es importante señalar que los ficheros JavaScript que se importan (tanto de bootstrap como de jQuery) son necesarios para que puedan funcionar algunos elementos, como por ejemplo, la contracción en un botón de los elementos de la barra cuando la ventana se hace lo suficientemente pequeña, o los ítems desplegables de la barra de navegación horizontal. Además, estos últimos tienen que proceder de una versión específica, en este caso, la 1.12.4, ya que en caso contrario no funcionarían.

Un aspecto a destacar también es que el formato de los ficheros es PHP y no HTML. Esto es así porque en todas las páginas, existen fragmentos de código PHP, por pequeños que sean, que el servidor debe interpretar y por ello, se usa esta extensión. Estos pequeños fragmentos de código PHP se corresponden con el hecho de crear, reanudar o mantener una sesión en todas las páginas que el usuario va visitando, permitiendo que su sesión sea continuada en todas ellas. En aquellas páginas dónde una aplicación Shiny se encuentra encapsulada en un iFrame, el código PHP permite obtener la ip dónde está alojada dicha aplicación, leyéndola un fichero de configuración alojado en una de las carpetas del proyecto (carpeta configuración).

Las rutas a los ficheros, especificadas tanto en Brackets como en rStudio, son relativas, lo que permite que, si no se modifica la estructura de directorios del proyecto, éste funcionará en cualquier sistema u máquina. Si se desea realizar cualquier cambio, como, por ejemplo, cambiar la ip dónde las aplicaciones Shiny se encuentran, los directorios donde se dejan las plantillas o ficheros de los usuarios, etc., éstos pueden llevarse a cabo modificando las líneas específicas del fichero de configuración.

6.2.2 BACK-END

Para codificar el back-end del proyecto se ha usado principalmente el entorno de desarrollo integrado (IDE): RStudio, versión 0.99, junto con PHP. RStudio [13] tiene numerosas facultades entre las que destacan: un notable editor de código en R, herramientas para depurar el código en busca de posibles fallos y de visualización. Una de las librerías o módulos más populares del mismo es el que se ha empleado en este proyecto: Shiny [13], el cual permite la creación de aplicaciones web interactivas con

las que visualizar información. También son utilizadas en este proyecto funciones que permiten dibujar gráficos a través de diferentes puntos. Para ello se ha utilizado la librería: highcharter.

A continuación, se mostrarán las diferentes aplicaciones Shiny y los ficheros PHP que toman parte en el back-end, para posteriormente explicar su función o funciones de la forma más clara y concisa posibles.

Todas las aplicaciones Shiny están constituidas por los mismos ficheros: ui.r, server.r y global.r.

- ui.r: interfaz gráfica de la aplicación con la que el usuario interactuará en la web.

- server.r: operaciones llevadas a cabo tras una acción determinada del usuario a través de la interfaz. Realizarán una o varias funciones.

- global.r: fichero usado para aquellos datos que vayan a compartirse entre las partes cliente y servidor: ui.r y server.r. Todas las aplicaciones excepto una: algOptimRegBenchs tienen un enlace simbólico al fichero global.r de dicha aplicación, ya que todos los datos declarados e inicializados en él sirven para todas.

Los objetivos de las diferentes aplicaciones son:

- algOptimRegBenchs: presenta un formulario al usuario con la información pertinente para dar de alta un nuevo benchmark en el sistema. La parte servidora recopila esa información y comprueba si es correcta para, a continuación, subirla a la base de datos.

- algOptimRegFuncs: de la misma forma que el anterior y el siguiente se encarga de ofrecer las diferentes opciones al usuario para poder dar de alta una nueva función en el sistema, comprobando el estado de la misma y dándola de alta en la base de datos en caso de que ésta sea correcta.

- algOptimRegTags: como los dos anteriores, se encarga de registrar un nuevo tag, etiqueta o característica de una función en el sistema.

- algOptimComp: esta aplicación permite al usuario comparar los resultados de dos o más algoritmos ejecutados en dos o más benchmarks, suministrándole información al mismo con gráficas que aclaran las diferencias entre los mismos.

- algOptimResultados: el usuario puede subir, a través de esta aplicación, sus propios resultados de un algoritmo, descargándose para ello la plantilla adecuada dependiendo del benchmark sobre el cual haya ejecutado su algoritmo. Acto seguido, se cumplimenta la plantilla correctamente y se sube. Esta información será permanente en el sistema si es validada por el administrador o administradores. Para ello, es necesario que el usuario adjunte un paper en formato PDF o código en formato .zip que demuestren la autenticidad de dichos datos. En caso contrario, si no se ha acompañado a los resultados de alguno de estos ficheros, esta información será temporal y un script que se ejecuta automáticamente cada hora se encargará de eliminarla si la sesión del usuario ha caducado y así lo demuestra la base de datos (la entrada con la ruta del script a ejecutar –borraResultados.php- se encuentra en el crontab del sistema -/etc/cron.d -).

- algOptimVal: Panel de gestión disponible para los administradores que permite validar los resultados de los algoritmos subidos por los usuarios que se encuentran pendientes de comprobación.

Una vez descritas las funciones de las diferentes aplicaciones, explicaré de igual manera la de los ficheros PHP que tengan un propósito determinado. Por lo tanto, aquellos que principalmente se encarguen de encapsular una aplicación no serán desarrollados:

- borraResultados.php: corresponde al código que permite obtener todas las sesiones activas PHP. También obtiene los identificadores de las diferentes sesiones de los usuarios que han subido información de sus resultados a cualquier tabla resultados$ de la base de datos (de cualquier benchmark), compararlas entre sí, identificar las sesiones caducadas (aquellas que están en la base de datos y no entre las sesiones activas), y borrar toda la información de la BD relacionada con estas sesiones.

- login.php: ofrece el formulario de inicio de sesión al usuario.

- loginS.php: realiza todo el proceso de login en el sistema de un usuario, lo que lleva a procesar los diferentes datos que éste introduce y de llamar al código del fichero errorLog.php en caso de que suceda algún problema.

- errorLog.php: notifica al usuario si el nombre y contraseña que introdujo para identificarse en el sistema no se corresponden con ningún par usuario-contraseña existente en la BD.

- registro.php: muestra el formulario de alta de usuario en el sistema.

- registroSroot.php: es parecido al anterior, lo que le diferencia es que en caso de ser un usuario administrador el que está registrando un usuario, éste será dado de alta como usuario administrador. El principio anterior sugiere que inicialmente en la base de datos debe estar dado de alta un usuario admin por defecto.

- errorReg.php: tiene el mismo objetivo que el anterior, es decir informar o mostrar al usuario un error, pero en este caso le advierte sobre que el nombre de usuario escogido ya existe y por tanto, no está disponible.

6.2.2.1 CONJUNTO DE LIBRERIAS USADAS

Seguidamente, citaré las librerías instaladas y vinculadas a RStudio que han permitido dotar a éste de las funcionalidades necesarias para este proyecto:

- Shiny: es la librería principal del proyecto y aquella que permite construir aplicaciones interactivas con el lenguaje de programación R. La asociación automática y reactiva entre Widgets de entrada y salida permiten construir aplicaciones web vistosas y fluidas.
- DBI: una interfaz que ofrece funciones para poder establecer la comunicación de R como cliente de un SGBD, en este caso PostgreSQL.
- Highcharter: ofrece la librería de JavaScript Highcharts, a través de un conjunto de funciones que permiten tomar como parámetros los campos de dicha librería. Las gráficas se van construyendo a través de una sintaxis especial de “pipes” que permiten recoger la salida de la anterior función y tomarla como entrada para la siguiente y así añadir o modificar alguna característica a dicha gráfica.
- DT: permite renderizar como tablas seleccionables los objetos de datos de R, como por ejemplo los DataFrame, utilizando la librería de JavaScript DataTables.
- Xtable: se compone de diferentes funciones para poder manejar y convertir a código Latex objetos de R como, por ejemplo, DataFrames.
- Xlsx: librería para generar código xlsx (Excel) a partir de DataFrames.
- ViridisLite: contiene funciones para permiten solicitar un número determinado de colores, expresados en código hexadecimal.

NOTA: En el propio código dónde se hace uso de estas librerías se indica, como comentarios, los comandos “install.packages()” que permiten la instalación en el sistema de cada una de las anteriores librerías. En la mayoría de las ocasiones este comando recibe como parámetro el mismo que se utiliza para importar las librerías al proyecto: “library()”, pero existen dependencias con otras librerías que no se usan de forma directa y por ello también deben ser instaladas. También se señalan en el código.

7. RESULTADOS

En este capítulo se comentará, utilizando como soporte capturas, los procedimientos a seguir para: crear un tag, una función y un benchmark, el cual tendrá otras funciones asignadas ya creadas, y posteriormente subir resultados de un algoritmo que compararé con otros tres ya existentes en la base de datos. Serán resultados con ámbito permanente, por lo que se mostrará también la fase de validación, pues sin ésta el algoritmo no se podría comparar al no estar visible en la aplicación de comparación. Tendremos que ser administradores para realizar este último paso comentado, y registrado, al menos, para dar de alta el benchmark, función y tag.

En primer lugar crearemos un tag, por lo que se dirigirá a la aplicación que permite crearlo y se introducirán los datos que se muestran a continuación, figura 13. Si la operación es correcta, recibiremos el mensaje: “Datos enviados”, en caso contrario, si existe una etiqueta con ese nombre recibiremos un mensaje de error: “Error, tag con esta etiqueta ya definida”.

Figura 13

A continuación, dirigiéndose a la página que permite dar de alta una nueva función rellenamos los campos adecuadamente como se muestran en las figuras 14 y 15.

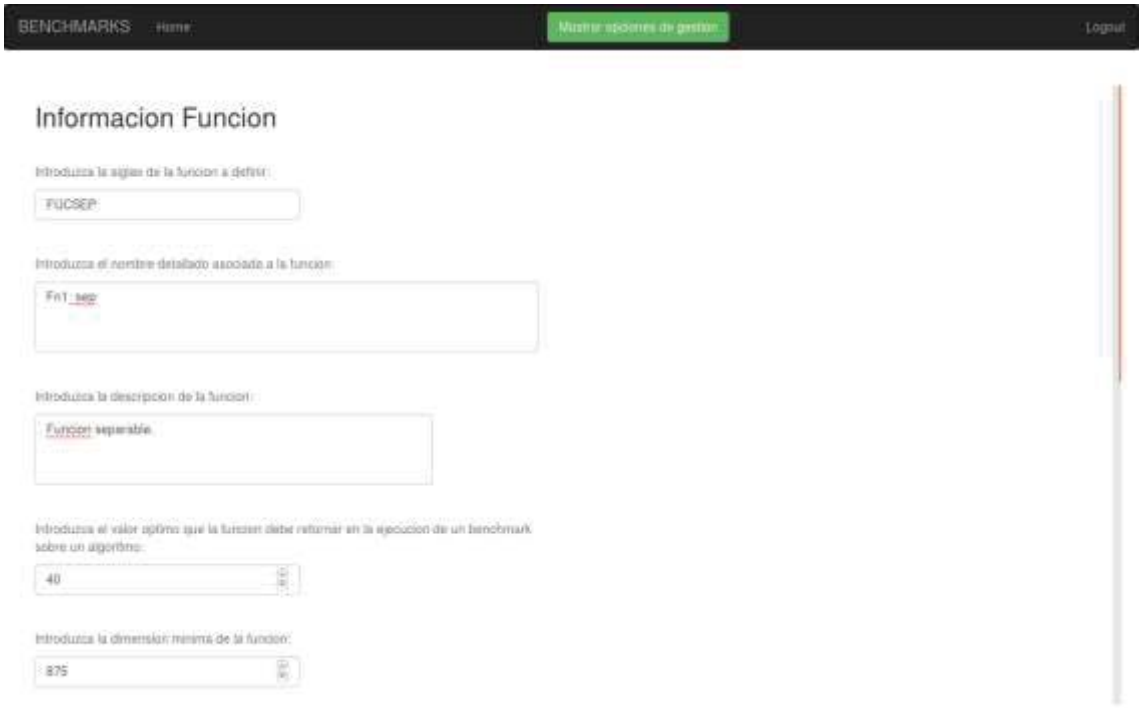


Figura 14

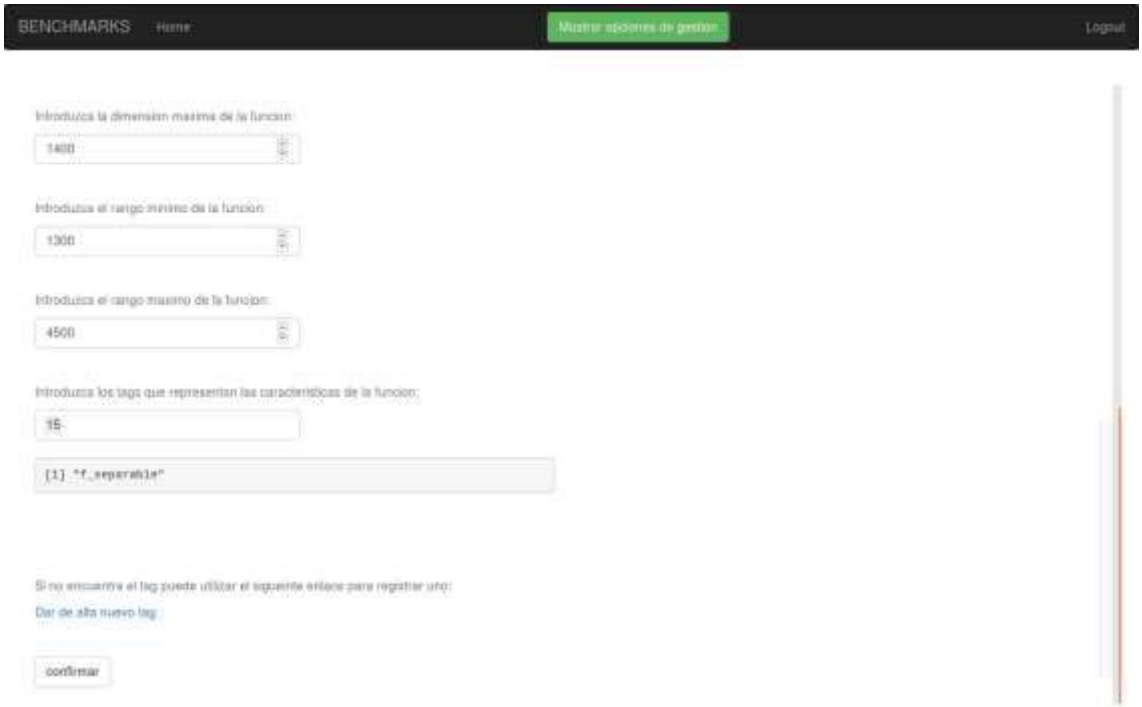


Figura 15

Seguidamente, daremos de alta el benchmark que incluye la función creada, ver figura 16.

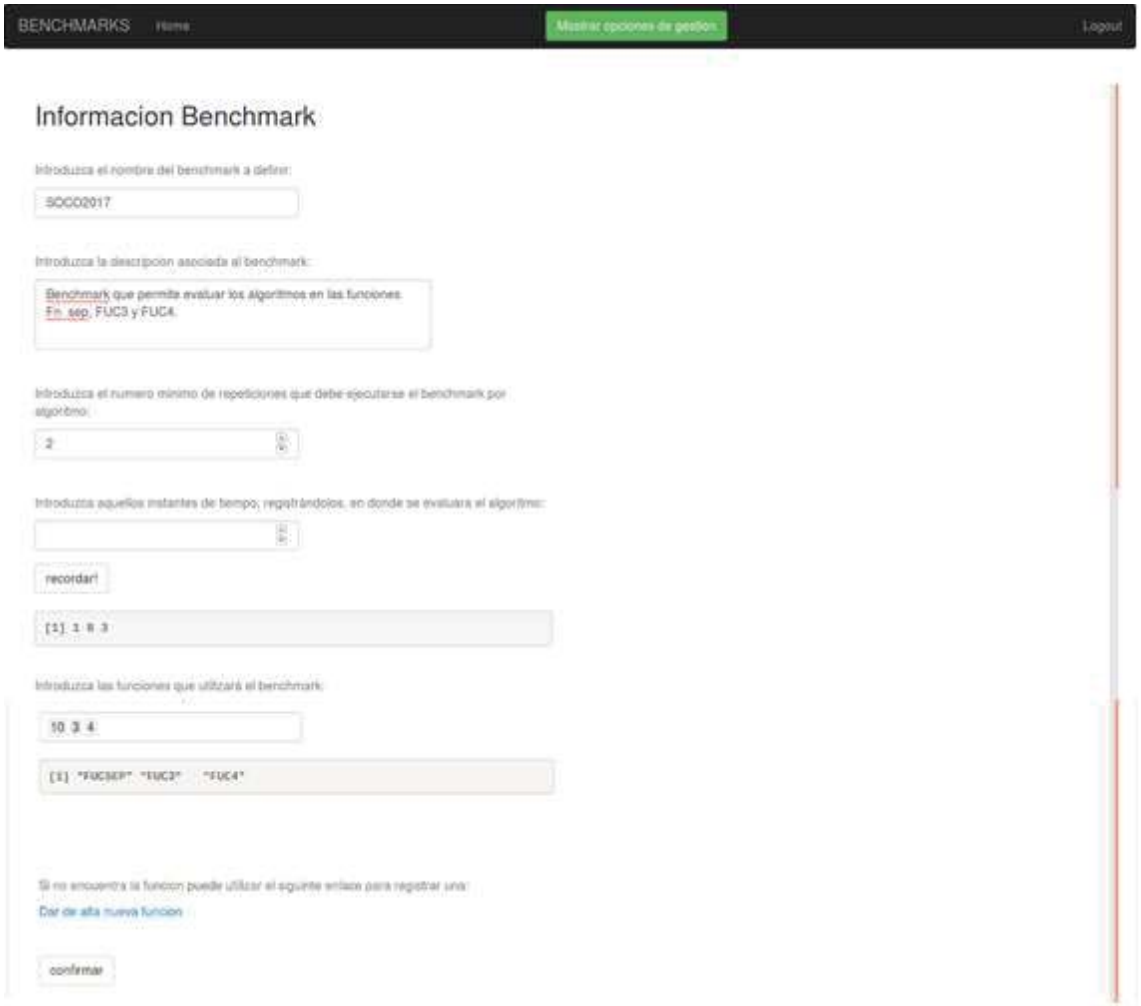


Figura 16

De nuevo, y esto se repite en todas las aplicaciones que permiten subir benchmarks, funciones y tags, si el procedimiento en el back-end se ejecuta correctamente, un mensaje cuyo contenido es: “Datos enviados”, aparecerá en pantalla, debajo del botón continuar.

En el paso siguiente procederé a subir resultados de un algoritmo, que llamaré algoritmo4SOCO201320152017, esta terminación se debe a los benchmarks sobre los que hay resultados de dicho algoritmo. No tiene mayor importancia, es para visualizar más rápidamente dichos algoritmos cuando lo estaba programando, y es la regla que he seguido. Ver figura 17.

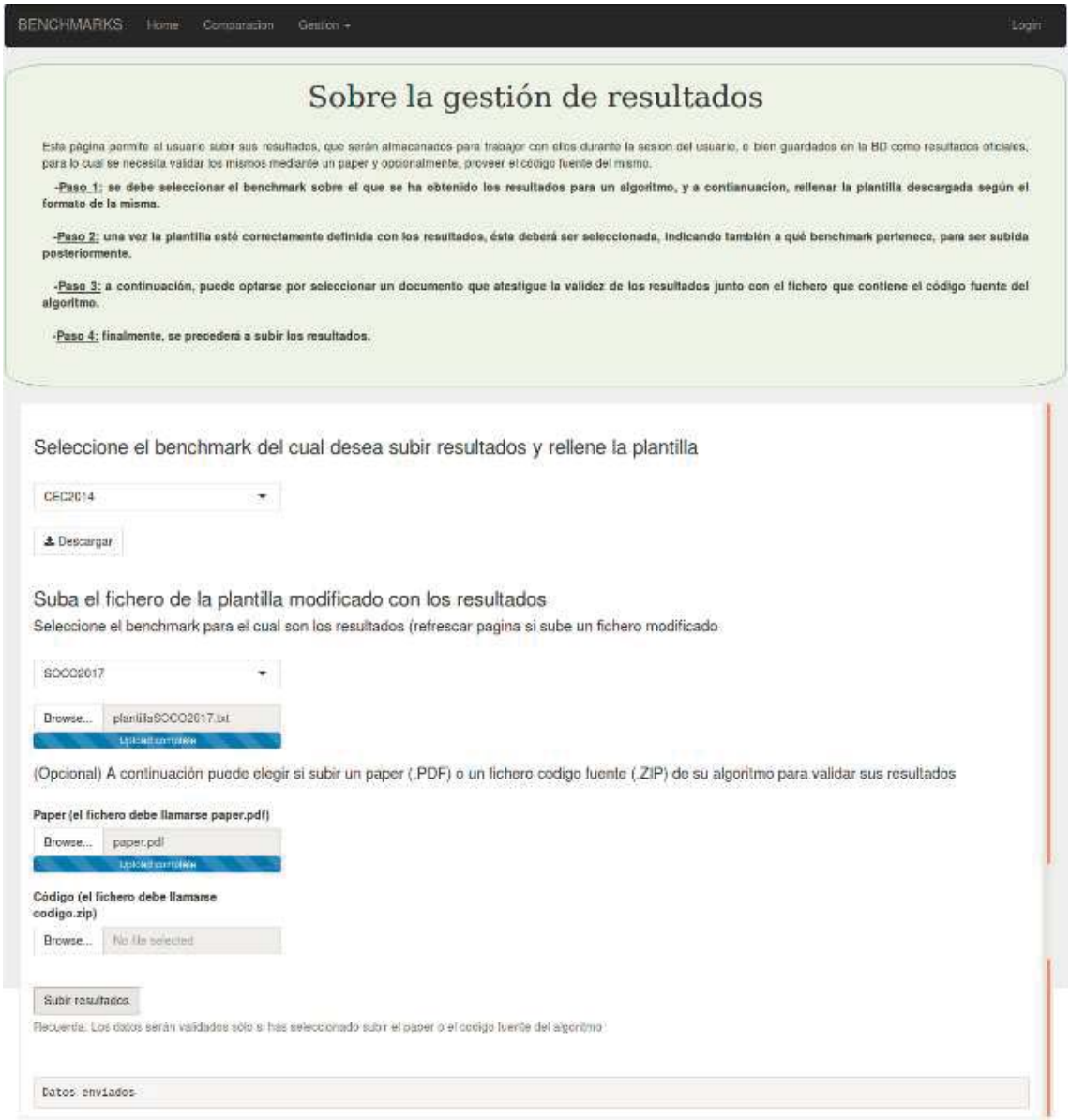


Figura 17

Hasta este punto tenemos los resultados en la base de datos, pero están sin validar, deberemos validarlos para poder empezar a realizar comparaciones con ellos, esto es lo que se muestra en la siguiente imagen, figura 18.

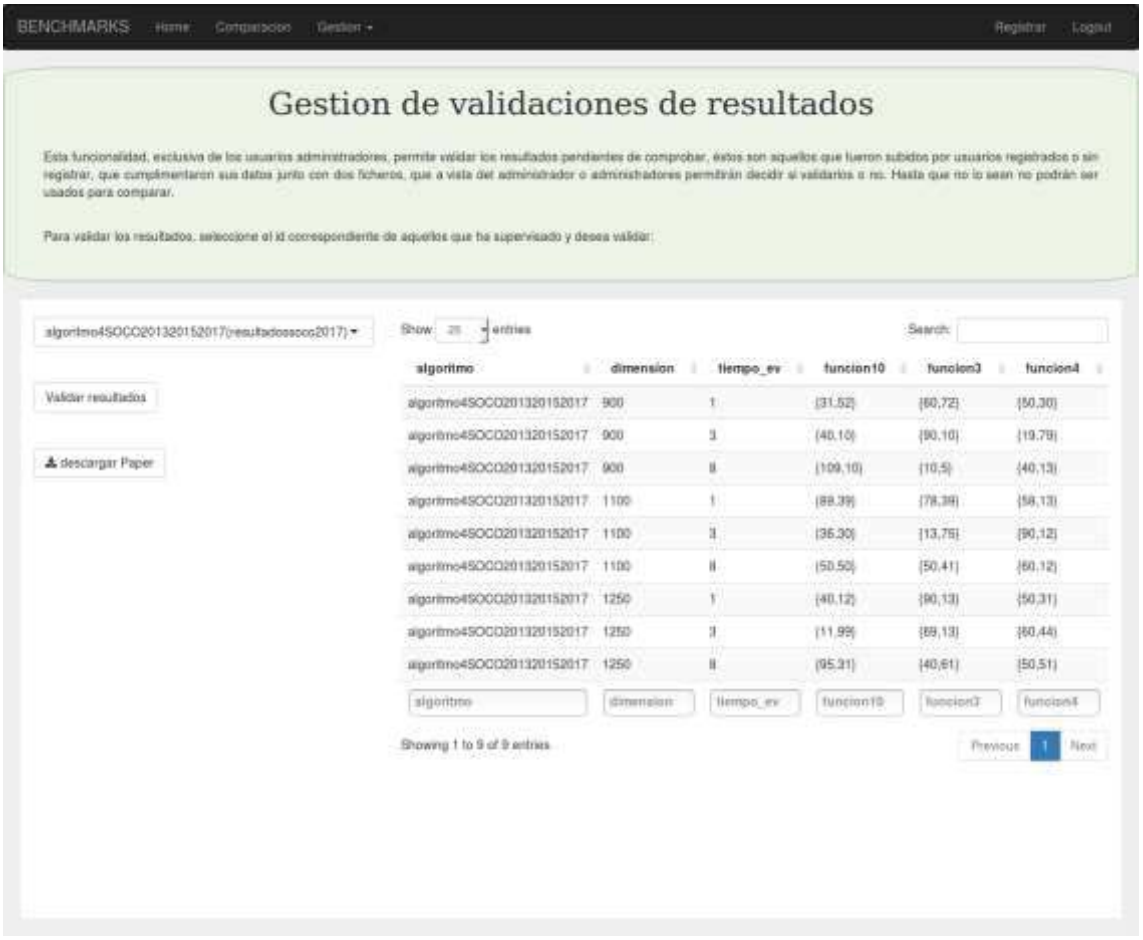


Figura 18

Por último, mostraré las tablas y algunas gráficas comparativas tras realizar la comparación entre cuatro algoritmos presentes en tres benchmarks.

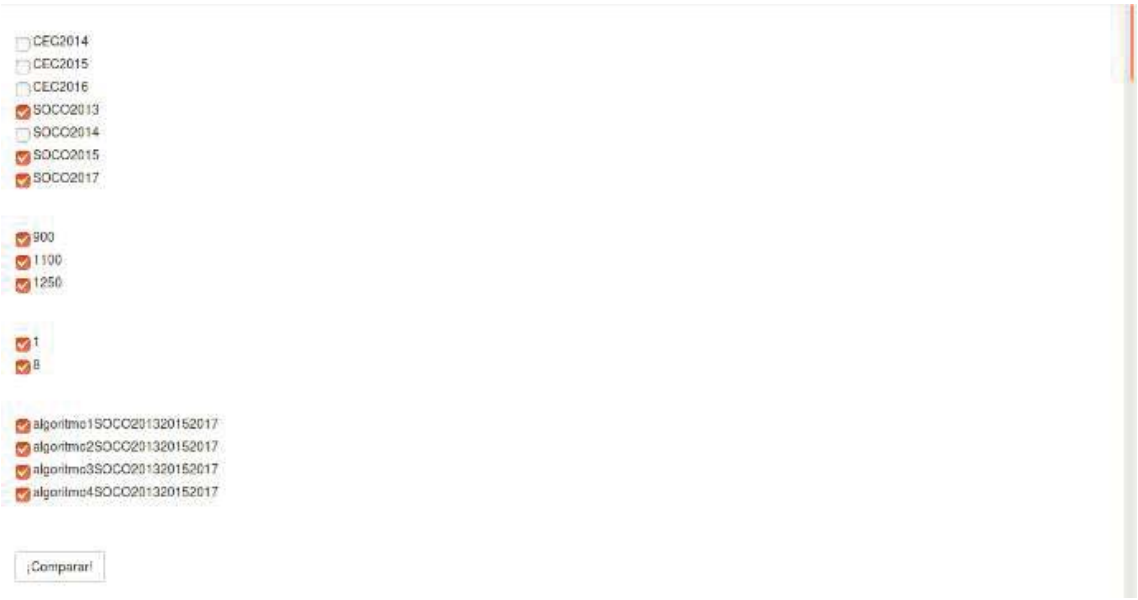


Figura 19

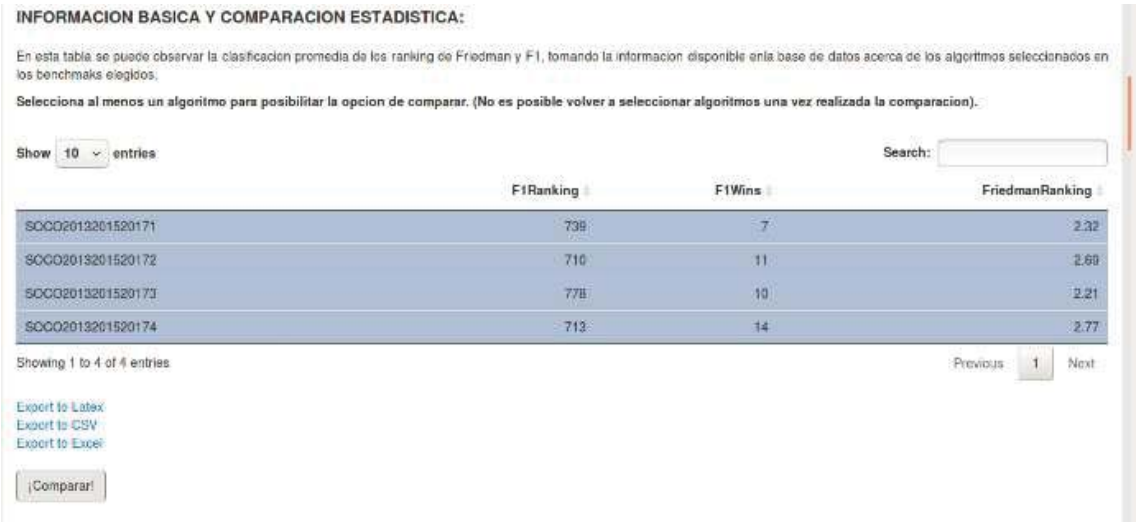
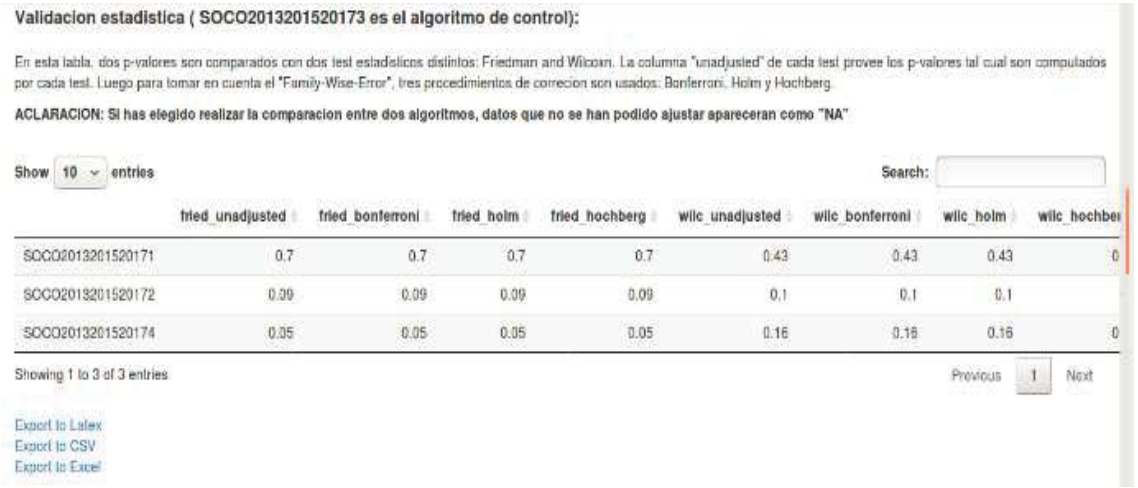


Figura 20

En la siguiente prueba, imagen 21, uno de los algoritmos, el que menor puntuación consigue en el anterior test, se escoge como algoritmo de referencia contra los demás.



Validacion estadistica (SOCO2013201520173 es el algoritmo de control):

En esta labla, dos p-valores son comparados con dos test estadisticos distintos: Friedman and Wilcoxn. La columna "unadjusted" de cada test provee los p-valores tal cual son computados por cada test. Luego para tomar en cuenta el "Family-Wise-Error", tres procedimientos de correcion son usados: Bonferroni, Holm y Hochberg.

ACLARACION: Si has elegido realizar la comparacion entre dos algoritmos, datos que no se han podido ajustar apareceran como "NA"

Show 10 entries Search:

	fried_unadjusted	fried_bonferroni	fried_holm	fried_hochberg	wilc_unadjusted	wilc_bonferroni	wilc_holm	wilc_hochber
SOCO2013201520171	0.7	0.7	0.7	0.7	0.43	0.43	0.43	0
SOCO2013201520172	0.09	0.09	0.09	0.09	0.1	0.1	0.1	
SOCO2013201520174	0.05	0.05	0.05	0.05	0.16	0.16	0.16	0

Showing 1 to 3 of 3 entries Previous 1 Next

Export to Latex
Export to CSV
Export to Excel

Figura 21

El test de Bonferroni-Dunn’s, figura 22, calcula una diferencia crítica que nos permite identificar diferencias significantes entre los algoritmos. Tomando como referencia el algoritmo con valor más bajo en el ranking de Friedman, figura 20.

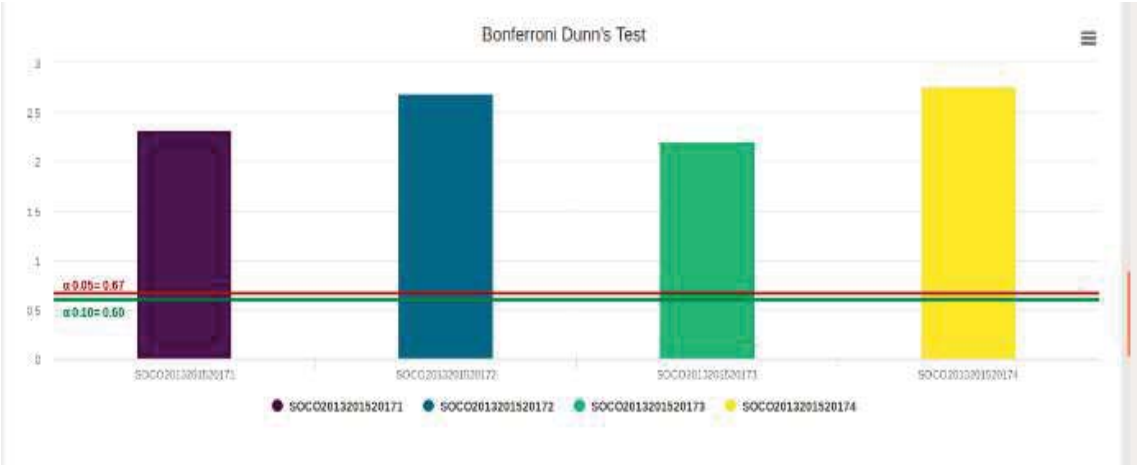


Figura 22

En las siguientes dos figuras veremos el ranking de Friedman por grupo de funciones (tags) entre los diferentes algoritmos comparados, la primera, figura 23, en formato tabla y la segunda en un gráfico de tela de araña, figura 24.

Ranking de Friedman por grupo de funciones

Esta tabla contiene el ranking de Friedman por grupo de funciones.

Show 10 entries Search:

	Tag1	Tag2	Tag3	Tag5	Tag4	Tag6	f_separable
SOCO2013201520171	2.3	2.33	2.38	2.38	2	2	2.17
SOCO2013201520172	2.58	2.58	2.96	2.96	2.5	2.5	2.5
SOCO2013201520173	2.2	2.83	2.25	2.25	1.67	1.67	2.17
SOCO2013201520174	2.92	2.25	2.42	2.42	3.83	3.83	3.17

Showing 1 to 4 of 4 entries Previous 1 Next

Export to Latex
Export to CSV
Export to Excel

Figura 23

Figura 24

Por último pondré un par de gráficos de tarta, los cuales representan la proporción de funciones, con una característica en común (tag), para las que cada algoritmo obtiene su mejor resultado (en términos de media de error), ver figura 25.

Figura 25

8. CONCLUSIONES Y DIFICULTADES ENCONTRADAS

Tras haber finalizado este proyecto de fin de grado he conseguido adquirir notables conocimientos sobre diferentes tecnologías que hasta ahora no había puesto en práctica.

Lenguajes de programación como plpgsql, usado para programar los diferentes triggers de la base de datos, junto con el manejo del gestor de bases de datos PostgreSQL, utilizado por sus características funcionales, resultaron de gran ayuda para conseguir llevar a cabo satisfactoriamente el sistema a desarrollar.

HTML, CSS y JavaScript han servido para desarrollar gran parte del front-end de la web. De ellas, las dos primeras se emplearon en menor medida el año pasado en la asignatura Diseño de Aplicaciones Web, debido a que la situación era distinta. Por un lado, dicho proyecto era compartido por un compañero que permitía aliviar la carga de trabajo, además de poder utilizar una plantilla web de estilo ya configurada, modificarla y conseguir que se adaptara a las necesidades pertinentes. Por otro lado, la mayor parte del back-end era llevado a cabo a través de Spring MVC, un framework especializado que proporciona un modelo vista – controlador, y componentes que pueden ser usados para desarrollar aplicaciones web flexibles con Java.

La otra gran parte corresponde a R, Shiny y DBI. El primero de ellos usado como lenguaje base para codificar todo el back-end, junto con PHP, y además permitiendo la interacción con los otros dos paquetes: Shiny, una librería que permite dotar a la web de widgets de entrada con los que interactuar, y de salida, para mostrar resultados en forma de texto, gráficas etc… y DBI, que contiene las funciones necesarias para hacer cualquier tipo de consulta (query) a la base de datos desde R.

Finalizando mis conclusiones, he conseguido implementar el sistema objetivo satisfactoriamente obteniendo conocimientos valiosos que seguro me servirán en mi futuro profesional. La ayuda del profesor prestada en tutorías y reuniones ha sido un recurso que me ha logrado aclarar en muchas ocasiones tiempo y malentendidos que posteriormente, si se hubieran acumulado, podrían haber resultado en grandes problemas. Es relevante destacar que como soporte o medio de respaldo he usado gmail, el almacenamiento de la máquina virtual y el disco duro de la máquina física, teniendo diferentes versiones siempre del fichero .zip que engloba el proyecto en las localizaciones indicadas.

Por otro lado, continuaré citando una lista de las complicaciones encontradas en este proyecto y trataré de explicar, por cada una de ellas, cómo las he afrontado.

- La primera de ellas hace referencia a la elección de este trabajo como proyecto de fin de grado. Una gran parte de las propuestas tenían como tema principal la

minería de datos, programación móvil, matemática discreta… temas que son importantes en la informática pero que no me resultan tan atractivas como pudieran ser el estudio de las tecnologías para desarrollar videojuegos, seguridad de empresas, ciberseguridad, diseño de aplicaciones web… En éstas últimas tengo cierto conocimiento previo debido a mi elevado interés por ellas, pero a excepción de la última sobre la cual tengo una idea práctica, como ya he mencionado, con las demás no me atreví por temas de tiempo que me podría llevar entender y finalizar el proyecto. Por estos motivos escogí el presente trabajo.

- A excepción de HTML y algo de CSS (y debido a la asignatura previamente indicada) no disponía de conocimiento previo de los lenguajes de programación ni de las librerías necesarias. El modo de desarrollar la aplicación en Spring y Java junto con HTML me sirvió para entender los conceptos y así poder continuar con este proyecto.
- Al principio me supuso cierta dificultad identificar el orden concreto de desarrollo de los diferentes sistemas que intervienen en la aplicación, pero decidí diseñar e implementar siempre de la forma más precisa y fiel que creía en cada momento y de acuerdo a los requisitos de la aplicación, permitiendo así que los sistemas, y con ello la aplicación, sean susceptibles a realizar un mayor número de modificaciones. En sí, cuando piensas más allá y te preguntas cómo se va a relacionar un sistema con los demás, los resultados son más eficaces y eficientes. El orden efectivo que seguí para la implementación fue: SGBD (PostgreSQL), Frontend (HTML y CSS), Backend (PHP), Frontend (Shiny), Backend (Shiny). He citado efectivo ya que en algunas ocasiones estaba diseñando o implementando un sistema que se apoya en uno anterior y me daba cuenta de que en éste alguna especificación o funcionalidad no estaba creada o funcionaba de forma incorrecta, por lo que tenía que volver a este otro sistema y corregirlo.
- El concepto de reactividad de Shiny. El hecho de que al principio creía que iba a tener que generar el código html de los widgets y controlar esta reactividad me supuso dificultades, por lo que me tuve que documentar con las lecciones del manual de la página para entender cómo funcionaba y cuáles eran las bases de esta librería. El hecho de que haya funciones reactivas, provistas por esta librería, que se ejecutan automáticamente al cambiar de valor alguno de los widgets de entrada sobre los que depende cada widget de salida para actualizarlos me resultaba un poco extraño. El hecho de controlar en una de las aplicaciones (algOptimComp) multitud de widgets de salida (botones, tablas, texto HTML dinámico, gráficas...) me resultó complicado de manejar y resolver con sentencias de control de flujo.

- La librería highcharter para R que permite, a través de una serie de funciones ofrecer la sintaxis JavaScript de la librería Highcharts y así poder generar gráficas con multitud de funcionalidades.

9. LINEAS FUTURAS

Para concluir, citaré una posible mejora y una observación acerca de qué funciones añadir con mayor trascendencia a la web.

En cuanto a la mejora, se podría incluir un script que se ejecute de forma automática cada x tiempo y permita la eliminación de aquellos directorios que se les asigna a cada usuario para almacenar sus ficheros opcionales a la hora de subir sus resultados a la web, esta acción se podría llevar a cabo por cada usuario, cuando los resultados se validaran por los administradores. Llevar a cabo una acción similar a la anterior contra los directorios que se usan para almacenar las plantillas que se les ofrecen y que utilizan para subir sus resultados, y también para los que almacenan, aunque al descargarlos en formato ZIP se eliminan, los ficheros e imágenes resultantes de una comparación.

Por otro lado, las primeras funciones que añadiría a la web serían las correspondientes a la eliminación y modificación tanto de los benchmarks, funciones y tags que están dados de alta en la base de datos. Los disparadores ya están codificados, a falta de hacer las modificaciones pertinentes en el front-end (HTML, CSS y ui.r) y en el back-end (server.r). También sería interesante, aunque de este tipo de aportes se podrían pensar cantidades de ellos, un nuevo panel, visible a todos los usuarios, con información acerca de los benchmarks, funciones y algoritmos registrados en la base de datos.

Sería aconsejable que en la subida de resultados se hiciera una comprobación más: la dimensión de los resultados sobre un benchmark que se están dando de alta debe estar dentro del rango dimensión mínima y dimensión máxima de todas las funciones que conforman el benchmark. Relacionado a lo anterior, verificar, antes de crear un benchmark, que las funciones que se le están asociando son todas válidas entre ellas (dimensión mínima y dimensión máxima contenidas).

Por último, en la comparación de algoritmos, el comportamiento del widget que permite seleccionar los algoritmos situados en el ranking de Friedman debe ser optimizado, si se selecciona un número menor de algoritmos que del total que hay en la misma, el código debería de actualizar la tabla con los algoritmos seleccionados, hasta que el usuario decida realizar la comparación de todos los presentes en dicha tabla. Actualmente, dicha actualización funciona cuando se deciden comparar dos de tres algoritmos presentes, pero si hay más de tres no permite seleccionar menos algoritmos, no actualiza bien. Por lo que para asegurar un correcto funcionamiento, se deben comparar los algoritmos seleccionados desde las checkboxes.

10. BIBLIOGRAFIA

[1] Wikipedia. (-). Soft computing [online]. Avaiable: https://en.wikipedia.org/wiki/Soft_computing

[2] Weise, T. (2009). Global optimization algorithms-theory and application. Self-Published [online]. Avaiable: http://scholar.googleusercontent.com/scholar?q=cache:qc9jL8awqJ8J:scholar.google.com/

[3] David Galles –Compute Science University of San Francisco-. (2011). Visualizing algorithms [online]. Avaiable: https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

[4] Toptotal. (-). Sorting algorithms animations [online]. Avaiable: https://www.toptal.com/developers/sorting-algorithms

[5] Edmund Julian L. Ofilada, RPubs. (2016, April 11). Comparing algorithms [online]. Avaiable: https://rpubs.com/DocOfi/170100

[6] I.Rodrigez-Fdez, A.Canosa, M.Mucieentes, A.Bugarín. (2015). STAC: Stadistical tests for algorithms comparison [online]. Avaiable: http://tec.citius.usc.es/stac/index.html

[7] Antonio LaTorre, Santiago Muelas, Jose-María Peña -. (Avaiable online, 2 October 2014). A comprehensive comparison of large scale global optimizers [article, Journal Homepage]. www.elsevier.com/locate/ins

[8] 2010 IEEE World Congress on Computational Intelligence. (2010, July 18-23). Special Session on Large Scale Global Optimizaction [online]. Avaiable: https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC10-LSO/CEC10.htm

[9] 2011 IEEE World Congress on Computational Intelligence. (2011). SOCO Special Issue on Large Scale Continuous Optimization Problems [online]. Avaiable: http://sci2s.ugr.es/sites/default/files/files/TematicWebSites/EAMHCO/editorial.pdf (enlace directo). http://sci2s.ugr.es/EAMHCO

[10] 2013 IEEE Congress on Evolutionary Computation. (2013, June 20-23). CEC 2013 LSGO Competition [online]. Avaiable: https://titan.csit.rmit.edu.au/~e46507/cec13-lsgo/competition/

[11] PostgreSQL. (-). PostgreSQL: The world's most advanced open source database [online]. Avaiable: https://www.postgresql.org/

[12] campusmvp. (-). Desarrollador web: Front-end, back-end y full stack [online]. Avaiable: http://www.campusmvp.es/recursos/post/Desarrollador-web-Front-end-back-end-y-full-stack-Quien-es-quien.aspx

[13] rstudio. (-). Main page [online]. Avaiable: https://www.rstudio.com/

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Fri Jul 07 21:05:36 CEST 2017
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)