TFG del Grado en Ingeniería Informática

Medidor estadístico de metajuego Magic the Gathering

Presentado por Eric Berlinches López

en Universidad de Burgos – 20 de Julio de 2020

Tutores: José Manuel Galán Ordax y José Ignacio Santos Martín

D. José Manuel Galán Ordax, profesor del departamento de Ingeniería de Organización, área de Organización de Empresas

Expone:

Que el alumno D. Eric Berlinches López., con 71313412 A, ha realizado el Trabajo final del GºIng.Informática titulado: Medidor estadístico metajuego Magic TheGathering.

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

En Burgos a 20 de julio de 2020

I- Índice de contenido

[APÉNDICE A: PLANIFICACIÓN DEL PROYECTO SOFTWARE 8](#_Toc46044737)

[A.1 Introducción 9](#_Toc46044738)

[A.2 Planificación temporal 9](#_Toc46044739)

[A.3 Viabilidad del producto 19](#_Toc46044740)

[APÉNDICE B: ESPECIFICACIÓN DE REQUISITOS FUNCIONALES Y CASOS DE USO 22](#_Toc46044741)

[B.1 Introducción 23](#_Toc46044742)

[B.2 Catálogo de requisitos 23](#_Toc46044743)

[B.3 Casos de uso 26](#_Toc46044744)

[APÉDICE C: DOCUMENTACIÓN TÉCNICA PARA EL USUARIO (MANUAL DE USUARIO) 35](#_Toc46044745)

[C.1 Explicación de contenidos 36](#_Toc46044746)

[C.2 Manual: Funcionalidades de la aplicación 36](#_Toc46044747)

[1- REGISTRO Y ACCESO AL SISTEMA 36](#_Toc46044748)

[2- MODIFICACIÓN DE PERFIL Y PERMISOS 40](#_Toc46044749)

[3- ASGINACIÓN DE BARAJAS 41](#_Toc46044750)

[4- INTRODUCIR DATOS DE UN TORNEO 42](#_Toc46044751)

[5- HISTORIAL DE TORNEOS 44](#_Toc46044752)

[6- ESTADÍSTICAS DE BARAJAS 45](#_Toc46044753)

[7- GESTIONAR BARAJAS (Administrador) 47](#_Toc46044754)

[7.1- AÑADIR BARAJAS 48](#_Toc46044755)

[7.2- MODIFICAR Y BORRAR BARAJAS 48](#_Toc46044756)

[8- CONCEDER PERMISOS 49](#_Toc46044757)

[9- INTRODUCIR RESULTADOS 50](#_Toc46044758)

[APÉNDICE D: DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN (MANUAL DEL DESARROLLADOR) 52](#_Toc46044759)

[D.1- Introducción: ¿qué hay en este anexo? 53](#_Toc46044760)

[D.2- Elementos, instalación y configuración 53](#_Toc46044761)

[D.3 Cómo importar el proyecto y continuar el desarrollo 61](#_Toc46044762)

[D.4 Ejecución fuera del IDE (desde el escritorio) 67](#_Toc46044763)

[D.5 Estructura del código 72](#_Toc46044764)

[D.6 Extra: procesos que requieren autenticación 73](#_Toc46044765)

[APÉNDICE E: ESPECIFICACIÓN DE DISEÑO 75](#_Toc46044766)

[E.1 Introducción 76](#_Toc46044767)

[E.2 Glosario de términos 76](#_Toc46044768)

[E.3 Computación de los datos 78](#_Toc46044769)

[E.5 Tablas y campos 79](#_Toc46044770)

II- Índice de figuras

**FIGURAS DEL ANEXO A:**

[Figura A01: tareas del sprint inicial 9](#_Toc45711256)

[Figura A02: tareas del segundo sprint 10](#_Toc45711257)

[Figura A03: tareas del tercer sprint 11](#_Toc45711258)

[Figura A04: Tareas del cuarto sprint 11](#_Toc45711259)

[Figura A05: Tareas del quinto sprint 12](#_Toc45711260)

[Figura A06: Tareas del sexto sprint 13](#_Toc45711261)

[Figura A07: Tareas del séptimo sprint 13](#_Toc45711262)

[Figura A08 Tareas del octavo sprint 14](#_Toc45711263)

[Figura A09: Tareas del noveno sprint 15](#_Toc45711264)

[Figura A10: Tareas del décimo sprint 15](#_Toc45711265)

[Figura A11 Tareas del undécimo sprint 16](#_Toc45711266)

[Figura A12 Tareas del duodécimo sprint 17](#_Toc45711267)

[Figura A13: Tareas del decimotercer sprint 18](#_Toc45711268)

[Figura A14: Tabla de análisis de servicios de hosting 20](#_Toc45711269)

[Figura A15: Detalles del servicio 1&1 20](#_Toc45711270)

**FIGURAS DEL ANEXO B:**

[Figura B16: Caso de uso 01 27](#_Toc45711271)

[Figura B17: Caso de uso 02 28](#_Toc45711272)

[Figura B18: Caso de uso 03 30](#_Toc45711273)

[Figura B19: Caso de uso 04 31](#_Toc45711274)

[Figura B20: Caso de uso 05 32](#_Toc45711275)

[Figura B21: Caso de uso 06.1 34](#_Toc45711276)

[Figura B22: Caso de uso 06.2 34](#_Toc45711277)

**FIGURAS DEL ANEXO C:**

[Figura C23: Pantalla de login 36](#_Toc45711278)

[Figura C24: Pantalla de registro 37](#_Toc45711279)

[Figura C25: Pantalla home user 38](#_Toc45711280)

[Figura C26: Pantalla home Superuser (cuenta administrador) 39](#_Toc45711281)

[Figura C27: Pantalla modificar datos del perfil 40](#_Toc45711282)

[Figura C28: Pantalla mis barajas 41](#_Toc45711283)

[Figura C29: Pantalla introducir torneo paso 1 42](#_Toc45711284)

[Figura C30: Pantalla introducir torneo paso 2 (introducir una ronda del torneo) 43](#_Toc45711285)

[Figura C31: Pantalla de confirmación del torneo 44](#_Toc45711286)

[Figura C32: Pantalla mis torneos 45](#_Toc45711287)

[Figura C33: Pantalla mis torneos (con un filtro aplicado) 45](#_Toc45711288)

[Figura C34: Pantalla estadísticas globales de las barajas 46](#_Toc45711289)

[Figura C35: Pantalla estadísticas globales de una baraja contra el resto 47](#_Toc45711290)

[Figura C36: Pantalla gestión de barajas 47](#_Toc45711291)

[Figura C37: Pantalla modificar y eliminar baraja 48](#_Toc45711292)

[Figura C38: Pantalla gestionar solicitudes de permisos 49](#_Toc45711293)

[Figura C39: Pantalla introducir resultado individual 50](#_Toc45711294)

[Figura C40: Pantalla introducir resultado individual con el menú de selección de baraja desplegado 51](#_Toc45711295)

**FIGURAS DEL ANEXO D:**

[Figura D41: Descarga de Apache NetBeans 54](#_Toc45711296)

[Figura D42: Descarga de MySQL Server (paso 1) 55](#_Toc45711297)

[Figura D43: Descargas de MySQL Community 55](#_Toc45711298)

[Figura D44: Descarga de My SQL Installer 56](#_Toc45711299)

[Figura D45: Descarga de librerías Vistual Studio C++ Redistributable 56](#_Toc45711300)

[Figura D46: Interfaz principal de MySQL Workbench 57](#_Toc45711301)

[Figura D47: Perfil de conexión a MySQL Server desde MySQL Workbench 58](#_Toc45711302)

[Figura D48: Solicitud de contraseña de MySQL 58](#_Toc45711303)

[Figura D49: Página de trabajo de MySQL Workbench 59](#_Toc45711304)

[Figura D50: Ubicación y contenido del fichero config.properties 59](#_Toc45711305)

[Figura D51: Descarga de Apache Tomcat 60](#_Toc45711306)

[Figura D52: Instalación de Apache Tomcat paso 1 60](#_Toc45711307)

[Figura D53: Instalación de Apache Tomcat: configuración de puertos 61](#_Toc45711308)

[Figura D54: Importación de proyecto en NetBeans paso 1 62](#_Toc45711309)

[Figura D55: Importación de proyecto en NetBeans paso 2 correcto 63](#_Toc45711310)

[Figura D56: Importación de proyecto en NetBeans paso 2 incorrecto 63](#_Toc45711311)

[Figura D57: Seleccionar servidor paso 1 64](#_Toc45711312)

[Figura D58: Seleccionar servidor paso 2 65](#_Toc45711313)

[Figura D59: Seleccionar servidor paso 3 65](#_Toc45711314)

[Figura D60: Importación de proyecto en NetBeans último paso 66](#_Toc45711315)

[Figura D61: Esquema del proyecto importado 66](#_Toc45711316)

[Figura D62: Abrir las variables de entorno paso 1 68](#_Toc45711317)

[Figura D63: Abrir las variables de entorno paso 2 68](#_Toc45711318)

[Figura D64: Modificar las variables de entorno “Home” 69](#_Toc45711319)

[Figura D65: Editar el path 70](#_Toc45711320)

[Figura D66: Editar el path paso 2 70](#_Toc45711321)

[Figura D67: Ruta de Apache Tomcat para colocar el fichero .war 71](#_Toc45711322)

**FIGURAS DEL ANEXO E:**

[Figura E68: Tabla usuarios 79](#_Toc45711323)

[Figura E69: Tabla barajas 79](#_Toc45711324)

[Figura E70: Tabla barajas\_usuarios 80](#_Toc45711325)

[Figura E71: Tabla cruces 80](#_Toc45711326)

[Figura E72: Tabla torneos 80](#_Toc45711327)

[Figura E73: Tabla roles 81](#_Toc45711328)

[Figura E74: Tabla roles\_usuarios 81](#_Toc45711329)

# APÉNDICE A: PLANIFICACIÓN DEL PROYECTO SOFTWARE

## A.1 Introducción

Este primer anexo recoge la planificación temporal del proyecto, dividido en etapas (sprints), explicando brevemente qué avances se produjeron en cada uno de ellos, qué dificultades, y qué cambios se dieron sobre la marcha, así como la viabilidad del producto final.

Para refrescar los conocimientos sobre la metodología Scrum y la división del trabajo en sprints y tareas, se consultó la página: <https://aulacm.com/mejores-hosting-web-alojamiento/> (Metodología SCRUM s. f.)

## A.2 Planificación temporal

**SPRINT INICIAL: 23 / octubre / 2019 – 22 / noviembre / 2019**

Este sprint inicial consiste en la elección de un nombre para la aplicación, definir algunas funcionalidades, investigar sobre trabajos similares y elegir las tecnologías.

Cabe destacar que, a la hora de buscar tecnologías y ser la primera vez que se va a desarrollar una aplicación web, se escoge JAVA por ser el lenguaje más conocido.

En el ámbito de la investigación, no existe ningún producto o servicio que haga exactamente lo mismo.



#### Figura A0: tareas del sprint inicial

**SEGUNDO SPRINT: 12 / noviembre / 2019 – 28 / noviembre / 2019**

“El siguiente paso es que tengas un login funcionando con una base de datos” son las palabras que inician el desarrollo del segundo sprint. Aunque estuviese decidido JAVA como lenguaje principal, se ve que hay una serie de elementos a interconectar (interfaz gráfica de usuario, control de los datos y base de datos), y que, además de los lenguajes, hacen falta un entorno de desarrollo, un servicio de base de datos y las correspondientes librerías que permitan que dichos elementos puedan interaccionar.

Como punto extra, al ser una aplicación web hace falta un elemento que tome el papel de servidor, al menos durante el desarrollo.

Se empieza a trabajar con Zenhub sobre Github, principalmente porque permite crear sprints y tareas sin estar asignadas a commits, y sobre todo en los primeros pasos no hay código que subir al repositorio.

Se desarrolla el prototipo de la aplicación con lo que serán los primeros bocetos de la interfaz gráfica.



#### Figura A0: tareas del segundo sprint

**TERCER SPRINT: 18 / noviembre / 2019 – 15 / enero / 2020**

Sprint con las primeras dificultades serias de desarrollo para interconectar los elementos previamente mencionados, que tiene una duración mucho más larga de la prevista inicialmente.

La tarea de este sprint es conseguir una aplicación que efectúe login contra la base de datos, desde una interfaz gráfica para el usuario.

También se realiza el primer diseño de la base de datos: qué tablas habrá y qué campos tendrán, aunque dicho diseño fue cambiando a medida que las funcionalidades añadidas lo requerían.

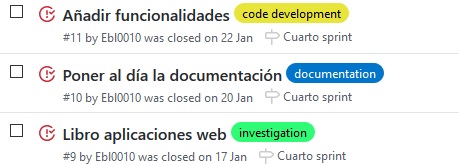


#### Figura A0: tareas del tercer sprint

**CUARTO SPRINT: 15 / enero / 2020 – 22 / enero / 2020**

Este cuarto sprint es bastante corto y básicamente conlleva un repaso de la teoría de las herramientas del repositorio (el uso de Zenhub, la definición de tareas, la asignación de labels a las mismas y los puntos de tiempo…) y documentar todo el proceso realizado en el sprint anterior, ya que hubo una reestructuración de elementos y, tras varias pruebas, se cambiaron algunas de las tecnologías inicialmente seleccionadas.

La tarea “libro aplicaciones web” hace referencia a terminar de leer el documento bibliográfico más importante para el desarrollo de este trabajo: (aplicaciones Web Java.pdf s. f.) que fue la clave para comprender la unión entre los distintos elementos: interfaz gráfica, control de datos y base de datos, pasando por el servidor.



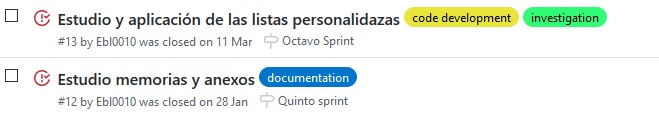
#### Figura A0: Tareas del cuarto sprint

**QUINTO SPRINT: 22 / enero / 2020 – 3 / febrero / 2020**

En este sprint se estudió la documentación de algunos repositorios de trabajos de fin de grado a fin de conocer la estructura de la memoria y anexos entregables, formato de estos, extensión y grado de detalle, para dar forma a la documentación propia en base a dichos documentos.

Se incrementa ligeramente la funcionalidad de la aplicación, permitiendo que los usuarios que acceden puedan empezar a ver sus datos (aunque aún no se pueden introducir, luego están siempre a 0, pero sí se modifican de forma manual en la base de datos para comprobar que la lectura es correcta).

Se inicializa la tarea “información sobre listas de valores personalizadas”, para controlar el flujo de datos de entrada (por ejemplo, si el usuario quiere seleccionar una baraja, mostrarle una lista con las barajas que existen, en vez de darle un campo de texto para la entrada manual), reduciendo considerablemente las probabilidades de errores en tiempo de ejecución.

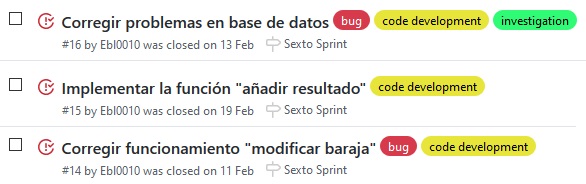


#### Figura A0: Tareas del quinto sprint

**SEXTO SPRINT: 3 / febrero / 2020 – 19 / febrero / 2020**

Este sprint añade la mayoría de la funcionalidad a la aplicación. Partiendo de una base que solicitaba una combinación nombre de usuario y clave de acceso, y permitía el ingreso de nuevos usuarios a la base de datos.

La tarea “implementar la función “añadir resultado” implica la programación de la función más importante de la aplicación, que conlleva un diseño sobre qué datos se van a guardar y cómo, una remodelación de la base de datos en función de dichas decisiones, y la programación de todo el proceso.



#### Figura A0: Tareas del sexto sprint

**SÉPTIMO SPRINT: 19 / febrero / 2020 – 2 / marzo / 2020**

En este sprint se toma una decisión de diseño que no estaba contemplada inicialmente: guardar un historial de los resultados en torneos. Entonces se modifica la función previa para que el usuario pueda introducir, no enfrentamientos de forma individual, sino historiales de torneos completos (que constan de N enfrentamientos individuales, siendo N el número de rondas jugadas en el torneo).

El sprint se da por finalizado con un commit en el repositorio con el código que tiene desarrollado un elevado porcentaje de lo reflejado en los requisitos funcionales inicialmente.



#### Figura A0: Tareas del séptimo sprint

**OCTAVO SPRINT: 2 / marzo / 2020 – 11 / marzo / 2020**

Este sprint viene tras una revisión con una tarea muy clara “las funcionalidades están bastante bien implementadas, hay que mejorar la interfaz del usuario cambiando los cuadros para introducción de texto y/o números por listas desplegables”.

Por tanto, se basa únicamente en la investigación sobre dichas listas y, una vez adquirido el conocimiento, la implementación en todas las secciones de código que lo requerían.

Sobre esta tarea cabe destacar que se inicializa a la altura del quinto sprint, pero se desanexa del sprint y se deja flotando, a la espera de ser retomada. Era una tarea pendiente, pero en ese momento la decisión es continuar desarrollando la funcionalidad, ya que, una vez establecidos los nombres de las variables donde se van a guardar los datos que se envían a la parte del controlador, la implementación de listas desplegables solo conllevaba cambios en los ficheros fuentes de HTML (o HTML) y no en el grueso del código.



#### Figura A0 Tareas del octavo sprint

**NOVENO SPRINT: 11 / marzo / 2020 – 6 / abril / 2020**

Este sprint solo tiene una tarea, y conllevó principalmente un estudio del diseño de la base de datos. Ante la duda del uso de identificadores enteros o continuar utilizando los nombres; concluyendo que resultaba mejor el uso de los nombres. Se realizan varios cambios en las mismas y se va modificando el código.

Se estudia la posibilidad de encriptar la URL de conexión a la base de datos, o al menos, la contraseña de acceso, y se deja como una tarea desanexada “estudio de encriptación de datos”, ya que también podía ser aplicable al login. Esta tarea se desechó cuando se vio, en el sprint final, que no se iba a desplegar en un servicio de hosting y únicamente se iba a proporcionar un despliegue local.

Además, se cambia el código para que todas las interacciones con la base de datos utilicen sentencias preparadas y protegerse de posibles ataques SQL Injection.

También se añade la última funcionalidad, consistente en mostrar el cuadro de los emparejamientos, y también se toma una decisión de diseño respecto a que la base de datos guarde emparejamientos “espejos” e información duplicada, por ser más fácil el acceso posterior de esta forma.



#### Figura A0: Tareas del noveno sprint

**DÉCIMO SPRINT: 6 / abril / 2020 – 21 / abril / 2020**

Sprint enfocado en la interfaz gráfica. Tiene dos tareas que pueden dividirse como el estudio teórico y toma de contacto con los elementos de css, y una segunda que consiste en la aplicación de dichos conocimientos desarrollando la interfaz gráfica de la aplicación.

Este sprint no presenta dificultad, pero sí inversión de tiempo, ya que tiene muchas correcciones de posiciones, cambios de colores, de fuentes, etc.



#### Figura A: Tareas del décimo sprint

**UNDÉCIMO SPRINT: 21 / abril / 2020 – 14 / mayo / 2020**

Podríamos decir que se trata de un sprint sin avance “principal”, o “de retoques”. En este sprint, con la interfaz de usuario ya prácticamente completa, hay varias tareas para añadir elementos muy puntuales:

Centrar el texto en las tablas: merece una mención especial, ya que, aunque se está trabajando con css y XHTML, las tablas que permiten mostrar datos dinámicos y se generan en tiempo de ejecución son elementos del framework (JavaServerFaces), y su etiqueta de HTML es: <h:datetable>. Están sujetas a reglas distintas del <table> “clásico”, y determinar el formato para una determinada fila o columna, o para todas las filas y columnas que no sean la primera hubo que hacer una investigación un tanto exhaustiva.

El resto de las tareas están bastante ligadas entre sí, ya que todas vienen a implementar una diferencia de roles de acceso. Antes de determinar qué tareas puede hacer un usuario con privilegios y uno sin ellos, hay que determinar cómo reconoce la aplicación el tipo de rol de las cuentas; de ahí que se modifique, antes de nada, la base de datos añadiendo tablas y campos para recoger esta información, que el login redireccione a una vista u otra en función de los permisos del usuario y, finalmente, un sistema interno que permita a los usuarios con rol más restrictivo solicitar ese tipo de cuenta con mayores privilegios.



#### Figura A Tareas del undécimo sprint

**DUODÉCIMO SPRINT: 14 / mayo / 2020 – 7 / junio / 2020**

Este sprint está orientado a la factorización del código. En este punto existe una clase (GestorBD) que tiene cerca de 1500 líneas de código. Contiene todos los métodos que trabajan contra la base de datos, y la idea es desglosar esta clase y crear varias clases “gestoras” de la base de datos, en función de los datos que vayan a tratar (una clase para los métodos que trabajan con los usuarios, tales como el registro de usuarios, el login o la gestión de roles), otra para toda la gestión de barajas, otra para la gestión de resultados…

A la par que se hacía este trabajo se iba depurando el tratamiento de excepciones, donde hasta ahora solo se mostraban las trazas en la consola. Se considera la opción de incluir un logger, por lo que se dejan comentadas las líneas que podían utilizarse en dicho caso, y además se plantea la opción de mostrarle al usuario mensajes por pantalla con una retroalimentación para que entienda qué ha podido pasar y qué puede hacer.

Esto deriva en la investigación del uso de JavaScript, y las funciones que permitan mostrar texto u ocultarlo, que no resultan ser compatibles con el framework utilizado ni con las clases ManagedBeans. La solución adoptada es crear una vista nueva, similar al login, con un cuadro de texto central, y manejar los mensajes de error a través de una clase y un método nuevo para mostrárselos al usuario en esta pantalla y después controlar a qué pantalla vuelve la aplicación.



#### Figura A Tareas del duodécimo sprint

**DECIMOTERCER SPRINT: 7 / junio / 2020 – 14 / junio / 2020**

Este sprint corto presenta una tarea que fue inicializada como la tarea número 24 del proyecto y se ha cerrado a la vez que la número 35. Es uno de los ejemplos de tareas “arrastradas” que en cierto momento se ha propuesto, se ha investigado y, sin llegar a una solución tras una inversión considerable del tiempo, se ha dejado aparcada para seguir avanzando con el proyecto para retomarla en un futuro.

Esta tarea es la del uso de un contexto JNDI para la conexión a la base de datos: no utilizar una URL directamente en el código, donde se puedan ver las credenciales de conexión a la base de datos y, por otro lado, mejorar la gestión de las conexiones utilizando un pool de conexiones, tal como se vio en la asignatura Aplicaciones de Bases de Datos.

Precisamente, la clase PoolDeConexiones.java tiene mucho código heredado de las prácticas de la asignatura, proporcionado por los profesores de la asignatura, mencionados como autores de dicha clase.

La otra inclusión al proyecto fue la de un logger para recoger las trazas de los errores.



#### Figura A: Tareas del decimotercer sprint

**ÚLTIMO SPRINT: 14 / junio / 2020 – 30 / junio / 2020**

Este sprint tenía la idea de ser el cierre final del proyecto, terminar de pasar a limpio la memoria, darle el formato apropiado, escribir los anexos y desplegar la aplicación.

Pero este punto fue la mayor dificultad encontrada durante el desarrollo del proyecto, y propició que la entrega se demorase hasta el 20 de julio.

Para realizar el despliegue de una aplicación web, la compilación genera un fichero .war, que debe colocarse en la ruta del servicio del sistema operativo que simula el servidor y realizar una serie de pasos. Además, la conexión a la base de datos requiere que el fichero donde se guardan las propiedades esté en una ruta a la que pueda llegar este servicio.

El despliegue en local se consiguió efectuar exitosamente el domingo 28 de junio, y tras varios intentos para alojar la aplicación en un servicio de hosting gratuito, sin éxito, se optó por montar una máquina virtual y permitir un despliegue en local de la aplicación. Esto conllevó varias horas también, ya que había que configurar todos los servicios necesarios en la máquina virtual (gestor de la base de datos y simulador de servidor), además de los componentes que requieren ser instalados para que ambos funcionen (Java Developtment Kit, Java RunTime Enviroment, librerías de Visual Studio C++…) y configurar las rutas y parámetros necesarios para su funcionamiento. Finalmente, crear un script que automatizase el proceso de arranque de servicios y lanzamiento de la aplicación.

Esto se consiguió la mañana del 2 de julio, y aquí se amplió el sprint.

**SPRINT FINAL (2º PARTE) 30 / junio / 2020 – 20 / julio / 2020**

Con una máquina virtual donde se puede desplegar la aplicación localmente, este sprint vuelve a presentar las tareas de redacción de la memoria y los anexos, aunque son elementos que se han ido construyendo de forma gradual a medida que se avanzaba con el proyecto, falta revisarlos, añadir los últimos cambios, incluir imágenes, índices, portadas y subirlos a la plataforma.

También se grabarán los videos solicitados, se estudiarán las posibles licencias a elegir y finalmente se subirán a una cuenta de OneDrive todos los elementos necesarios para ser referenciados con links en el fichero readme del repositorio.

## A.3 Viabilidad del producto

Económicamente hablando no es un producto costoso, ya que existen servicios de hosting gratuitos y podría realizarse un primer despliegue online sin coste alguno. Debido a que el nivel de recursos que requiere es bastante reducido, también puede considerarse la opción del uso local por los usuarios, que no tendría ningún coste al utilizar herramientas de código libre.

Para un lanzamiento en un servicio de hosting más profesional, que permita albergar una mayor cantidad de información en su base de datos, podemos realizar un estudio económico en base al estudio de la página: (Roca 2015), resumido en la figura A14:



#### Figura A: Tabla de análisis de servicios de hosting

Vemos que el ejemplo de 1&1 es el servicio que presenta la mejor relación servicios / precio (también en la página (Roca 2015).)



#### Figura A: Detalles del servicio 1&1

Suponiendo el uso de este servicio, tendríamos que realizar un pago de, redondeando, 12€ al año. Ahora habría que hacer una valoración de cuánto dinero merece la pena pagar por la aplicación, teniendo en cuenta el servicio que ofrece, y el número de usuarios potenciales.

Teniendo en cuenta los precios de los torneos y de las cartas que los jugadores sí pagan, y valorando la información que la aplicación otorga a sus usuarios, el proceso que automatiza y, en su contra, que es algo que los usuarios pueden replicar de forma gratuita con hojas de cálculo, el precio inicial podría rondar los 5€.

Con estos datos parece difícil que la aplicación no resultase rentable, e incluso, permitiría reducir el coste inicial si al principio no alcanza demasiada popularidad. La estrategia de venta sería ofrecer un producto sencillo pero útil, y llegar a muchos usuarios.

# APÉNDICE B: ESPECIFICACIÓN DE REQUISITOS FUNCIONALES Y CASOS DE USO

## B.1 Introducción

Partiendo de los objetivos definidos en la memoria, se han determinado una serie de requisitos funcionales y no funcionales, así como casos de uso para la aplicación. Este anexo recoge el desglose de todos ellos, con las descripciones y proceso de cada uno de ellos lo más detalladamente posible.

Se han revisado los artículos (Requerimientos Funcionales y No Funcionales, ejemplos y tips | by Requeridos Blog | Medium s. f.) y (Guía para la redacción de casos de uso | Marco de Desarrollo de la Junta de Andalucía s. f.) para identificar y diferenciar los requisitos funcionales y no funcionales del proyecto, y determinar los casos de uso así como la redacción de los mismos, respectivamente.

## B.2 Catálogo de requisitos

**B.2.1 Requisitos funcionales**

0- Funcionalidad principal:

Proveer a los usuarios de un sistema que les ayude a identificar con qué baraja obtienen mejores resultados y qué baraja obtiene mejores resultados a nivel global.

Se derivan por tanto los requisitos de entrada de datos (1), procesamiento de datos (2) y presentación de datos (3), además del almacenamiento de estos (4) y la capacidad de reconocer distintos usuarios (5).

RF1- Entrada de datos:

Se deben poder diferenciar los distintos datos que cada usuario en función de su rol puede introducir o no.

RF1.1A- Introducir, modificar y borrar bajaras: Los usuarios con privilegios podrán introducir nuevas barajas a la base de datos, modificar los datos de las que ya existen o borrarlas.

RF1.1B- Asignar barajas: El usuario tendrá una lista de barajas asignadas, donde podrá añadir de forma manual todas las que quiera, de entre las que hay en la base de datos.

RF1.2- Registrar un torneo jugado: El usuario podrá introducir los datos de un torneo que acaba de jugar. Estos datos serán qué baraja ha utilizado el usuario, cuántas rondas ha jugado, y contra qué otra baraja, y qué resultado ha obtenido, en cada ronda. Obteniendo en cada una los datos de un cruce (baraja\_A, baraja\_B, victorias de main de la baraja\_A, victorias de main de la baraja\_B, victorias de side de la baraja\_A y victorias de side de la baraja\_B).

RF2- Procesamiento de datos:

Disponer de mecanismos internos (backend) que permitan procesar los datos que el usuario va introduciendo, de modo que se vayan actualizando sus ratios de victorias y sus estadísticas con cada baraja utilizada.

Cada vez que un usuario introduce datos de un torneo hay tres procesos respecto a esos datos:

RF2.1: Procesar los cruces. Esto significa que habrá una tabla que guarde, para cada par de barajas, las victorias de una frente a otra, tanto de main como de side, y los porcentajes de main, side y el global. Por cada ronda del torneo introducido se habrá generado un cruce, por tanto, se deberán actualizar los datos en todas las filas de la tabla cruces que corresponda.

RF2.2: Procesar los datos de cada baraja. Esto significa que habrá una tabla donde se almacenen las barajas y los datos de las victorias y derrotas totales de cada una, de main y side, y sus porcentajes. En cada cruce habrá que actualizar los datos de las dos barajas que se han enfrentado.

RF2.3: Procesar los datos del torneo, para el usuario. Tras procesar todas las rondas, hay unos resultados de victorias y derrotas, de main y side, totales en todo el torneo, del usuario con la baraja que ha utilizado. Se deben actualizar los datos que relacionan a ese usuario con esa baraja, y también los datos globales del usuario (donde se cuentan sus victorias y derrotas con todas las barajas). Además, para el usuario se contabilizarán también las rondas de torneo y el resultado de este (rondas ganadas – rondas perdidas – rondas empatadas).

RF3- Presentación de los datos:

Disponer de una interfaz gráfica que le muestre al usuario los datos que puedan ser útiles para él y de una forma sencilla y con fácil navegación entre ellos.

Estos datos serán sus ratios de victorias en tantos por cien, así como sus datos en bruto respecto a victorias y derrotas con cada baraja. Podrá también acceder a un historial de sus torneos para ver los resultados que obtiene con cada baraja.

RF3.1- Presentación al usuario de sus datos globales (rondas ganadas, rondas perdidas, rondas empatadas, partidas ganadas, partidas perdidas, porcentaje de partidas ganadas y porcentaje de rondas ganadas), así como un desglose de estos con cada una de las barajas que ha utilizado.

RF3.2- Presentación al usuario de su historial de torneos: el usuario podrá acceder a ver los resultados que ha obtenido en los torneos (en formato rondas ganadas -rondas perdidas – rondas empatadas), con cada baraja, y el número de veces que ese resultado se repitió.

RF3.3- Presentación al usuario de los datos globales de todas las barajas, y un desglose de estas. A nivel de usuario esta es la funcionalidad más importante, ya que implica conocer las ratios de victoria de cada baraja, en total, y de cada baraja contra cada una de las otras. Requiere que exista una tabla donde se almacenen los resultados de victorias y derrotas de cada baraja frente a cada otra [cruces] y las ratios porcentuales para cumplir este requisito.

RF4- Almacenamiento de los datos:

Los datos de victorias de un usuario con una baraja, y derrotas, y todos los datos de victorias y derrotas de cada baraja contra cada otra, para sacar las ratios de victorias entre cada par de barajas deben ser datos persistentes. Este requisito funcional explica la necesidad de utilizar un sistema gestor de base de datos.

RF5- Capacidad de reconocer distintos usuarios:

La aplicación requiere un sistema de login para que cada usuario pueda introducir, y más tarde consultar, sus propios datos: tanto sus resultados a nivel global como sus resultados utilizando cada una de las barajas.

RF5.1- Capacidad de la aplicación de tener distintos niveles de permisos y distintas acciones asociadas a cada permiso.

RF5.2- Capacidad de la propia aplicación para proveer de un servicio de peticiones de cambios de roles entre usuarios.

**B.2.2 Requisitos no funcionales**

RNF1- Seguridad y cohesión: Todas las transacciones contra la base de datos se realizan mediante sentencias preparadas, evitando así la posibilidad de ataques de SQL-Injection. Por otro lado, existen numerosos menús de selección, evitando que los usuarios puedan introducir datos incorrectos, evitando en gran medida un control de flujo de entrada y un tratamiento de excepciones más robusto.

RNF2- Aprendizaje: Sin la necesidad de un manual de usuario como soporte, prácticamente cualquier jugador ligeramente experimentado de Magic: the Gathering puede comprender los nombres utilizados en los menús y lo que se debe introducir. Para usuarios con menos experiencia, el manual supone un soporte del vocabulario. Los propios menús ayudan indicando al usuario qué debe introducir en cada caso, mediante menús despegables que evitan los errores, y mediante mensajes de feedback.

## B.3 Casos de uso

A continuación, se describen cada uno de los casos de uso, enumerando sus precondiciones, postcondiciones, excepciones, requisitos funcionales que cumple y secuencia detallada paso a paso, así como algún comentario para tener en cuenta durante su implementación.

Captura de pantalla de un celular

Descripción generada automáticamente

#### Figura B: Caso de uso 01

Captura de pantalla de un celular

Descripción generada automáticamente

#### Figura B: Caso de uso 02

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

#### Figura B: Caso de uso 03

Consideración asociada a este caso de uso: cómo almacenar los torneos.

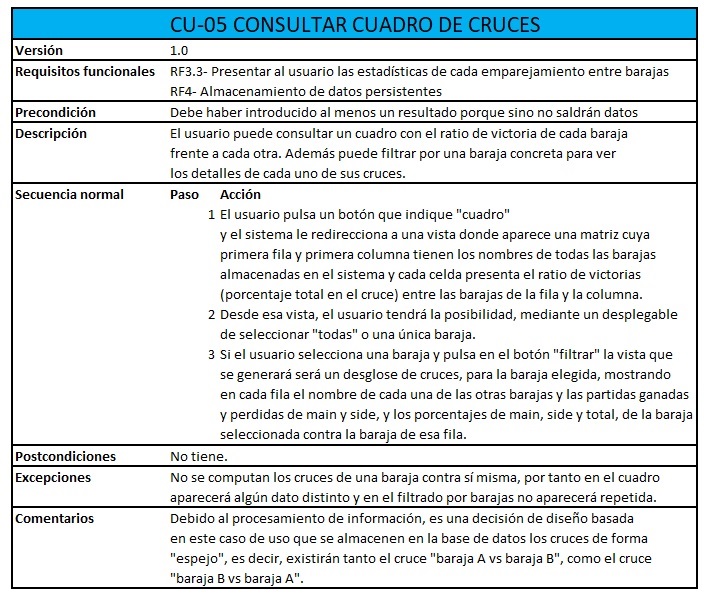
Entre jugadores de Magic: the Gathering, la referencia a los resultados de los torneos consta de dos cifras (rondas ganadas y rondas perdidas) y, eventualmente si hubiese alguna ronda empatada se diría en tercer lugar. Ejemplo: (“Tres, uno” hace referencia a tres rondas ganadas y una perdida. “Dos, uno, uno” hace referencia a dos rondas ganadas, una perdida y una empatada.)

Como se lleva un conteo de las variables locales de rondas ganadas, empatadas y perdidas, después de introducir estos datos en la tabla del usuario con la sesión activa, se puede introducir un registro en la tabla torneos con el usuario, la baraja que ha utilizado y un String [G-P-E], siendo G = rondas ganadas, P = rondas perdidas y E = rondas empatadas, y con otro campo entero que contabilice cuántas veces se ha dado ese resultado.

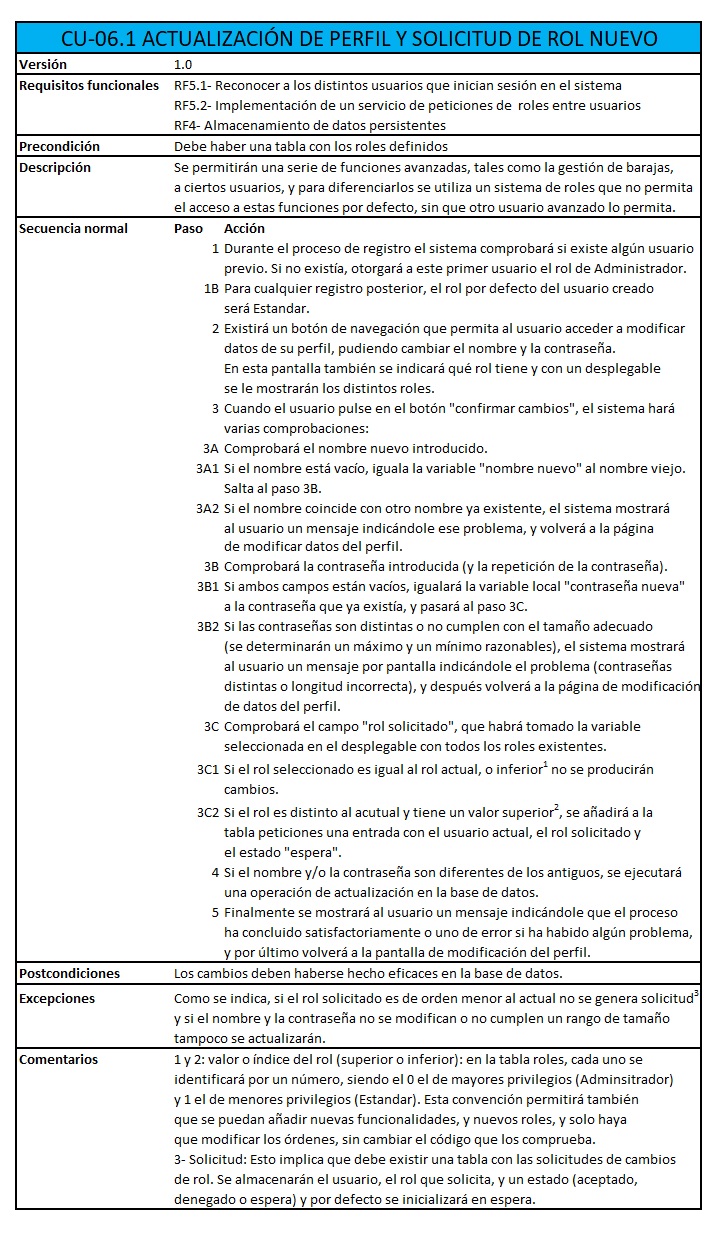
Captura de pantalla de un celular

Descripción generada automáticamente

#### Figura B: Caso de uso 04



#### Figura B: Caso de uso 05



#### Figura B: Caso de uso 06.1



#### Figura B: Caso de uso 06.2

# APÉDICE C: DOCUMENTACIÓN TÉCNICA PARA EL USUARIO (MANUAL DE USUARIO)

## C.1 Explicación de contenidos

El anexo C es el apartado donde se explican las funcionalidades de la aplicación, ayudan al usuario a entender los distintos menús y pantallas, así como la navegación entre los mismos. En este apartado no se especifican los componentes ni como realizar la instalación porque ya aparece en el repositorio de forma detallada, en el fichero “readme”.

## C.2 Manual: Funcionalidades de la aplicación

### 1- REGISTRO Y ACCESO AL SISTEMA

En la figura C23 se puede observar la página de acceso al sistema. Si el usuario no dispone de una cuenta, deberá crear una, haciendo clic en el botón “registrarse”.



#### Figura C: Pantalla de login

La figura C24 muestra la página de creación de cuenta. Aquí el usuario debe introducir un nombre de perfil, de entre seis y veinte caracteres, una contraseña de entre seis y veinte caracteres, y una dirección de correo electrónico[[1]](#footnote-1).



#### Figura C: Pantalla de registro

Durante este proceso el sistema puede detectar algún error, entonces se lo notificará al usuario y volverá a la página de creación de una nueva cuenta.

Estos posibles errores son:

-El nombre de usuario es muy corto o largo.

-La contraseña es muy larga o corta.

-Las contraseñas no son iguales.

-Ya existe una cuenta con ese nombre de usuario.

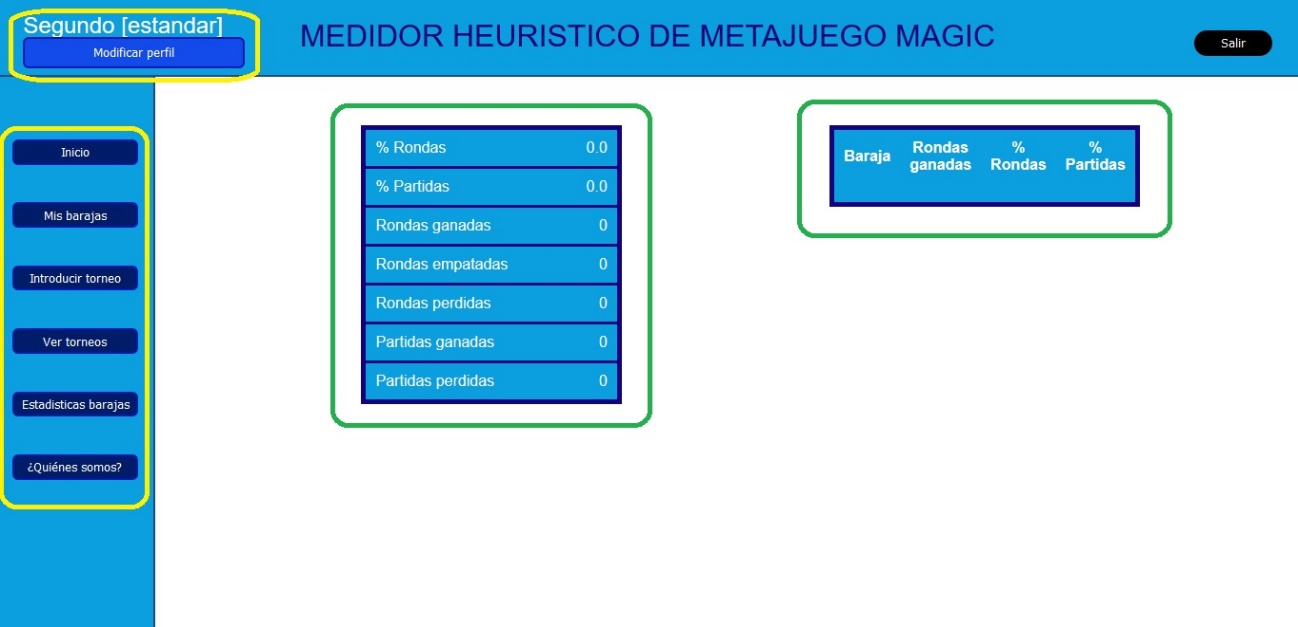
Por otro lado, si el usuario ya disponía de una cuenta, habrá de introducir su nombre y contraseña en la pantalla de login. Una vez que pulsa el botón entrar pueden producirse varios errores:

-No existe ninguna cuenta con ese nombre de usuario.

-La contraseña de esa cuenta de usuario no es correcta.

Si el registro, o acceso, es correcto, por cualquiera de ambos caminos, el sistema carga la pantalla principal de usuario.

La figura C25 muestra esta pantalla “home” del usuario con tipo de cuenta “estándar”.



#### Figura C: Pantalla home user

En el bloque bordeado de amarillo en la esquina superior izquierda se ve la información del perfil: nombre de cuenta, entre corchetes tipo de cuenta, y debajo el botón para modificar los datos del perfil.

Por debajo, también a la izquierda, está el menú de navegación principal, que permite acceder a las distintas funciones de la aplicación, que se desarrollarán más adelante.

En la zona central hay dos cuadros bordeados con verte; el que está más a la izquierda es el que contiene los datos estadísticos del usuario, y el que está más a la derecha es el que contiene los datos resumidos de las barajas que el usuario ha utilizado.

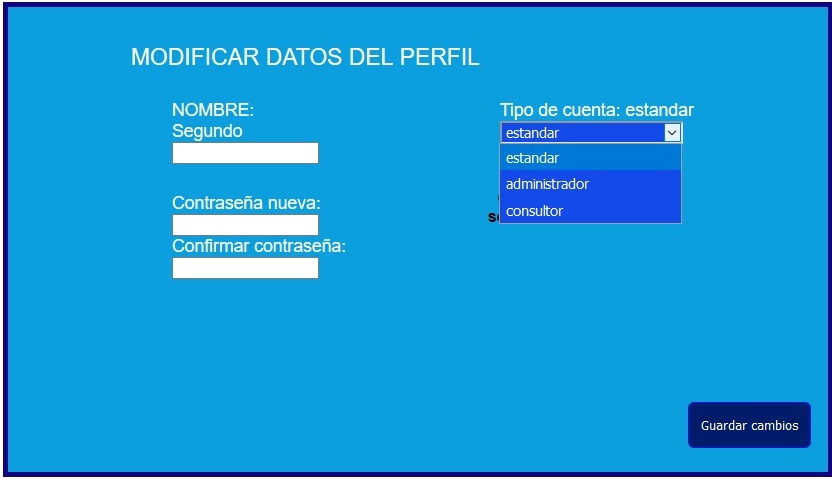
La figura C26 muestra la misma vista para una cuenta de tipo “administrador”, con algunas funciones añadidas (los tres botones amarillos nuevos) y con varios datos calculados.



#### Figura C: Pantalla home Superuser (cuenta administrador)

### 2- MODIFICACIÓN DE PERFIL Y PERMISOS

Utilizando el botón “modificar perfil”, se accede a la pantalla que se puede ver en la figura C27.



#### Figura C: Pantalla modificar datos del perfil

Esta página tiene varios recuadros muy descriptivos, que permiten al usuario introducir un nuevo nombre de cuenta, una nueva contraseña y solicitar un cambio de tipo de cuenta.

Cuando el usuario haga clic en el botón “guardar cambios” el sistema procesa todas las posibles entradas:

-Nombre: si el usuario ha introducido un nombre nuevo, el sistema comprobará que dicho nombre cumple la longitud estipulada (entre seis y veinte caracteres) y que no exista ya una cuenta con ese nombre. Si deja este campo en blanco no se producirá ningún cambio en el nombre.

-Contraseña nueva y confirmar contraseña: Si el usuario desea cambiar su clave de acceso debe introducir una nueva, en los dos espacios. Deben ser ambas iguales y cumplir la longitud estipulada (entre seis y veinte caracteres). Si el usuario deja este campo en blanco no se producirá ningún cambio.

-Tipo de cuenta: hay un desplegable con los posibles tipos de cuenta[[2]](#footnote-2). El usuario puede seleccionar el tipo de cuenta que desea, y el sistema comprobará si la cuenta que ha solicitado es menos restrictiva que la actual para procesar la solicitud.

NOTA: El usuario puede cambiar únicamente un parámetro, dos de ellos o los tres al mismo tiempo.

### 3- ASGINACIÓN DE BARAJAS

La figura C28 corresponde con la pantalla que el usuario verá si hace clic en el botón “mis barajas” del menú de navegación. En esta pantalla el usuario puede ver sus estadísticas utilizando las distintas barajas detalladas (victorias de main y side con cada baraja), y podrá añadir a sus barajas cualquier otra baraja[[3]](#footnote-3).

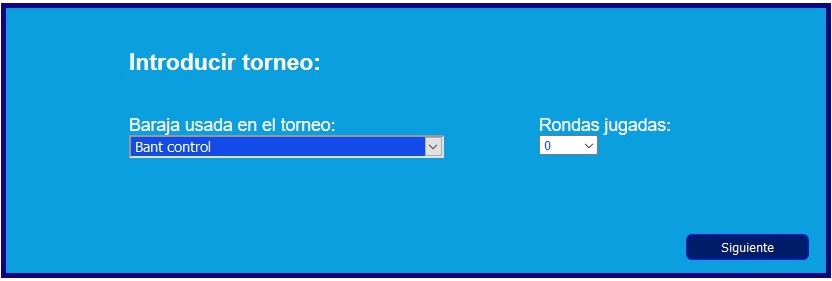


#### Figura C: Pantalla mis barajas

Si el usuario intenta agregar una baraja que ya tiene asignada el sistema le informará de que el proceso ha ido mal. (Para asignar una baraja el usuario debe seleccionar el nombre de esta en el panel desplegable y después hacer clic en “agregar”.)

### 4- INTRODUCIR DATOS DE UN TORNEO

El mecanismo principal de introducción de datos al sistema para su cómputo es el registro de torneos. Si el usuario hace clic en el botón “Introducir torneo” del menú de navegación, verá la pantalla de la figura C29.



#### Figura C: Pantalla introducir torneo paso 1

Aquí el usuario deberá seleccionar cuál de sus barajas ha utilizado en el torneo que quiere registrar, y de cuántas rondas ha constado este torneo[[4]](#footnote-4). Cuando pulse en aceptar, verá una pantalla como la de la figura C30.

En esta pantalla la información está separada en una rejilla: la columna izquierda representa al jugador que está introduciendo los datos y la derecha a su oponente; la primera fila representa las barajas que han utilizado cada uno, la segunda las victorias de main que ha obtenido cada uno y la tercera las victorias de side que ha obtenido cada uno.

Mediante desplegables el usuario seleccionará qué baraja utilizó su oponente en la primera ronda del torneo y los resultados en dicha ronda (por ejemplo, si ganó la primera partida, de main, y perdió las dos de side, su pantalla debería reflejar lo mismo que la figura C30).



#### Figura C: Pantalla introducir torneo paso 2 (introducir una ronda del torneo)

Cuando haga clic en el botón “aceptar”, el sistema procesará la ronda y le mostrará la misma pantalla, indicando que se trata de la segunda ronda (encima del nombre de la baraja que está utilizando el usuario aparece ronda X, siendo X la variable).

El problema que puede surgir es que el usuario se enfrente a una baraja que no está contemplada en la base de datos. Aquí puede optar por dos opciones: cancelar el proceso de registro, pulsando en cualquier botón del menú de navegación, aunque se recomienda la opción “Introducir torneo” para volver directamente al punto donde debe elegir qué baraja ha utilizado y el número de rondas, y elegir una ronda menos. Otra forma de solventar este problema es seleccionar la baraja oponente como LA MISMA que está utilizando, así la ronda se computará para su resultado a nivel del torneo, pero no el cruce contra la baraja.

Una vez que haya introducido todas las rondas, el sistema le mostrará una pantalla como la figura C31, donde podrá ver el resumen que ha introducido para validarlo o volver atrás si hay algún error.



#### Figura C: Pantalla de confirmación del torneo

Una vez que pulse en “guardar torneo” el resultado se computará, y el sistema devolverá al usuario a la pantalla de inicio “home”.

### 5- HISTORIAL DE TORNEOS

Además de las estadísticas de cada baraja, para el usuario se guardan sus resultados de torneos, como bien es sabido por los jugadores de Magic: The Gathering, el resultado de un torneo se menciona como una tupla de tres números, siendo el primero el número de rondas ganadas, el segundo las rondas perdidas y por último las rondas empatadas, pudiendo obviarse este cuando es un cero.

La figura C32 muestra la pantalla donde el usuario puede ver su historial de torneos, ordenados de mejor a peor resultado. Se especifica con qué baraja consiguió ese resultado y el número de veces que se ha repetido. El usuario puede además filtrar por una sola baraja utilizando el menú desplegable que hay en la parte superior, eligiendo solo una baraja para ver sus resultados conseguidos únicamente con dicha baraja (figura C33).



#### Figura C: Pantalla mis torneos

****

#### Figura C: Pantalla mis torneos (con un filtro aplicado)

### 6- ESTADÍSTICAS DE BARAJAS

Esta es la función estrella de la aplicación, que permite ver a todos los usuarios los ratios de victorias de todas las barajas, computados tras cada ronda de cada torneo introducida.

Para acceder a ella hay que hacer clic en el botón “estadísticas barajas” del menú de navegación, y el usuario verá una pantalla bastante similar a la figura C34.



#### Figura C: Pantalla estadísticas globales de las barajas

Desde aquí se pueden ver todas las barajas que hay en la base de datos, ordenadas por Tier, y sus ratios de victoria de main, side y el total.

Además, para ver los datos de una baraja concreta contra el resto del Metajuego[[5]](#footnote-5), el usuario puede utilizar el desplegable que hay en la parte superior para elegir la baraja cuyo desglose quiere analizar. Una vez elegida, cuando haga clic en “detalles”, verá una pantalla similar a la figura C35.



#### Figura C: Pantalla estadísticas globales de una baraja contra el resto

### 7- GESTIONAR BARAJAS (Administrador)

La función de gestionar barajas hace referencia a la capacidad para agregar nuevas barajas al sistema y modificar las existentes o borrarlas. Si el usuario, con una cuenta de Administrador, hace clic en el botón “Gestionar barajas” del menú de Administrador (botones amarillos y solo accesible desde la vista del home de las cuentas Administrador), su pantalla será similar a lo mostrado en la figura C36.



#### Figura C: Pantalla gestión de barajas

### 7.1- AÑADIR BARAJAS

En esta misma pantalla, a la derecha hay un cuadro que indica “agregar nueva baraja”. Aquí el usuario puede introducir el nombre de una nueva baraja que quiera añadir al sistema, y su tier, y pulsar en “agregar”. El sistema introducirá la nueva baraja a la base de datos, si el tier es mayor que cero, y el nombre no coincide con el de otra baraja ya existente.

### 7.2- MODIFICAR Y BORRAR BARAJAS

Encima de la lista con todas las barajas hay un desplegable que contiene los nombres de todas las barajas. Si el usuario selecciona una y después hace clic en el botón “modificar”, a la derecha, y en el mismo panel, el sistema le reconducirá a una pantalla similar a la que se ve en la figura C37.



#### Figura C: Pantalla modificar y eliminar baraja

Desde aquí el usuario puede introducir un nuevo nombre y modificar el tier. Cuando pulse el botón “confirmar cambios”, el sistema realizará unas comprobaciones:

-Si el nombre está en blanco, solo intentará cambiar el tier.

-Si el tier se deja en cero, solo intentará cambiar el nombre.

-Si el nombre coincide con el de una baraja ya existente, el sistema le informará

de ello al usuario.

-Si el nombre se deja en blanco y el tier a cero, no ocurrirá nada.

En esta misma pantalla hay un botón que indica “eliminar”. Si el usuario lo pulsa se borrarán todos los datos de esa baraja, incluidos los registros de esa baraja en las estadísticas de las barajas, todos los emparejamientos que tuviesen involucrada a esa baraja y las estadísticas de todos los usuarios con esa baraja. Esto no borra los datos globales del usuario:

🡪 Si el usuario A ha jugado un torneo con la baraja B, y posteriormente la baraja B es borrada del sistema, en sus páginas de inicio y de desglose de barajas, el usuario no verá más la baraja B, pero todas las rondas y partidas que hubiese ganado con ella sí permanecen en el cuadro izquierdo del inicio, donde se muestran las partidas y rondas totales ganadas por el usuario.

### 8- CONCEDER PERMISOS

Pulsando sobre el botón “Gestionar usuarios” el sistema redirige al usuario a la pantalla para gestionar las peticiones de permisos. Aquí se muestra una lista con todos los usuarios que solicitan una petición de cuenta con mayores privilegios que la actual.

En la figura C38 se muestra una única petición: se ve el nombre de la cuenta y el tipo de cuenta que solicita, y en un desplegable, el usuario tiene tres opciones: conceder, denegar y esperar.



#### Figura C: Pantalla gestionar solicitudes de permisos

Inicialmente aquí aparecerán todas las peticiones en estado “denegar” o “esperar”. Si el usuario elige “conceder”, la petición desaparecerá de la lista y, la próxima vez que el usuario solicitante acceda al sistema, accederá a la página “home” de super usuario, viendo los cambios pertinentes (verá que en el tipo de cuenta le indica “Administrador”, y además verá el menú que permite gestionar barajas y usuarios y añadir un resultado).

Si se elige la opción denegar, la petición continuará en la lista, para poder volver a cambiar el estado, pero el solicitante verá que su petición está en estado “denegada” si accede a modificar su perfil, y no podrá realizar una nueva solicitud.

### 9- INTRODUCIR RESULTADOS

Opción únicamente para las cuentas con permisos de administrador que permiten introducir un resultado individual de un cruce (como una ronda individual de torneo) seleccionando tanto las dos barajas enfrentadas como las victorias y derrotas de cada una.

La idea de esta funcionalidad es poder añadir resultados que han sido observados a jugadores que no están dados de alta en la aplicación y así poder aumentar la información que se procesa. (La figura C39 muestra esta pantalla con todos sus campos y la figura C40 muestra desplegado uno de los dos campos de selección de baraja)



#### Figura C: Pantalla introducir resultado individual

****

#### Figura C: Pantalla introducir resultado individual con el menú de selección de baraja desplegado

# APÉNDICE D: DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN (MANUAL DEL DESARROLLADOR)

## D.1- Introducción: ¿qué hay en este anexo?

Este manual de programador de la aplicación Medidor Estadístico de Metajuego es una guía para instalar los distintos componentes que permiten hacer funcionar la aplicación, así como el desarrollo del código de esta.

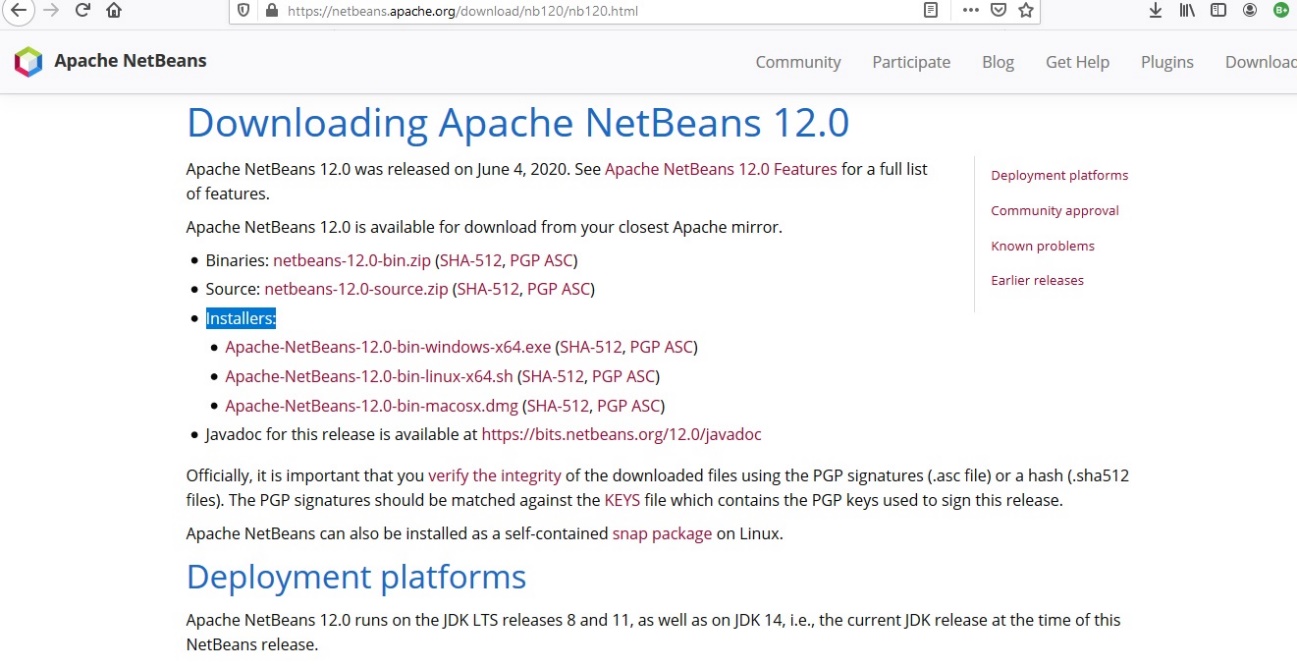
También presenta una explicación de la base de datos: qué tablas hay, qué campos tienen y cómo funcionan, y, finalmente, una explicación de las clases que se encuentran en el código fuente, cómo están relacionadas entre sí y qué estructura forman.

Se recomienda potencialmente el uso de este manual junto con la documentación de los comentarios del código para su mayor comprensión. (<https://github.com/Ebl0010/Medidor-estadistico-metajuego-Magic-The-Gathering/tree/master/dist/javadoc>)

## D.2- Elementos, instalación y configuración

La aplicación está desarrollada en el entorno de desarrollo Apache NetBeans, y aunque es posible desarrollarla también en Eclipse, este manual solo contempla la primera opción ya que ha sido la utilizada.

Para descargar el IDE hay que ir a su página oficial (<https://netbeans.apache.org/>) y al apartado descargas. Es preferible descargar una de las opciones “instalador”. (Figura D41)

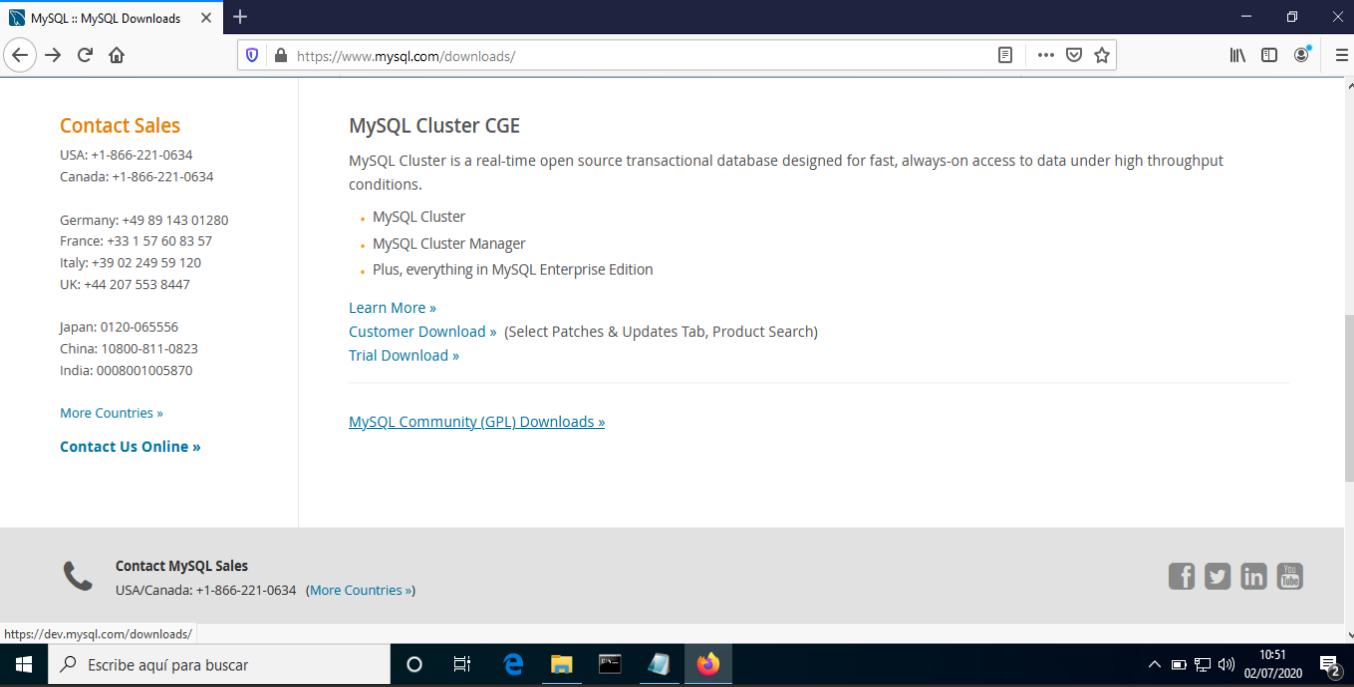


#### Figura D: Descarga de Apache NetBeans

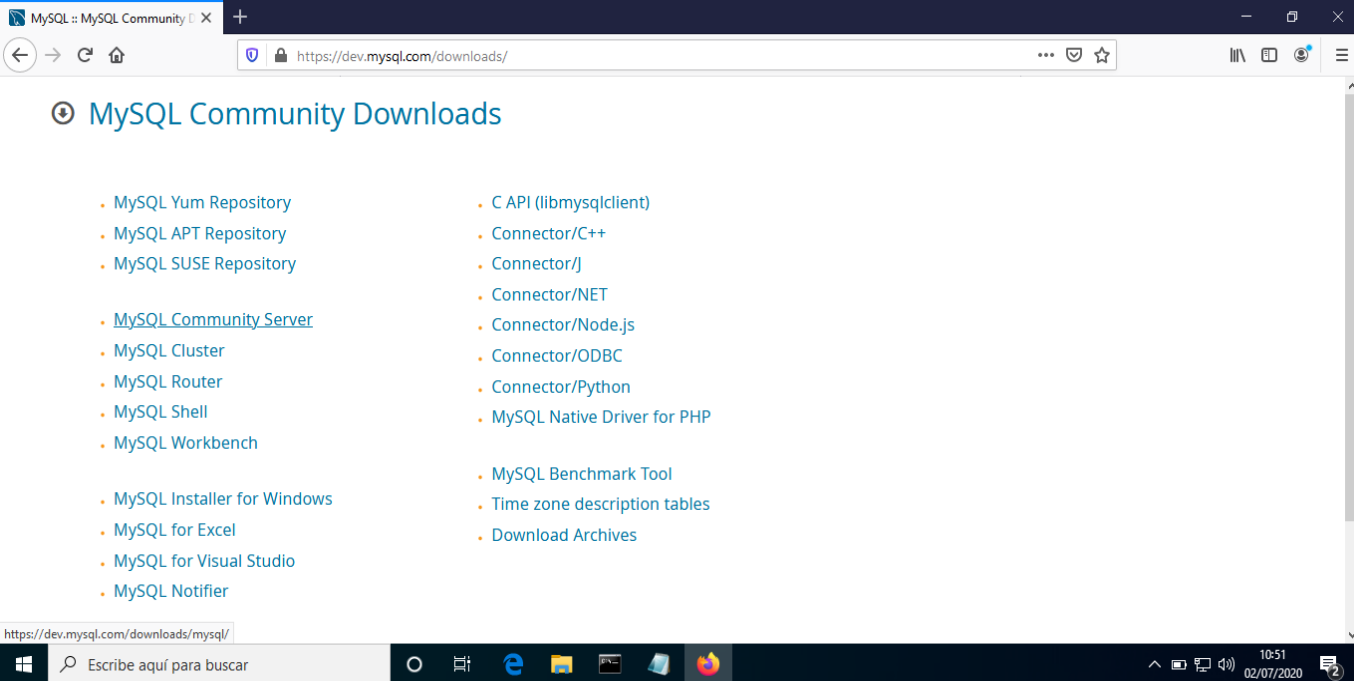
Una vez descargado, hacer doble clic y dejar que el instalador haga su trabajo.

NOTA: es necesario disponer del entorno JAVA: JRE y JDK, si el instalador no los encuentra informará al usuario de que son necesarios y no podrá continuar con el proceso.

La base de datos es My SQL, habrá que descargar primero el servicio (MySQL Server) y después, de forma opcional, aunque recomendable, la interfaz gráfica para interaccionar con la base de datos, comprobar los cambios en tiempo real y probar la ejecución de código SQL (My SQL Workbench). Ambas herramientas están disponibles en su página oficial (<https://www.mysql.com/downloads/>) Para encontrar la descarga, dentro de My SQL Downloads, hay que ir a la sección “comunity (GPL) downloads”, como se muestra en las figuras D42 y D43

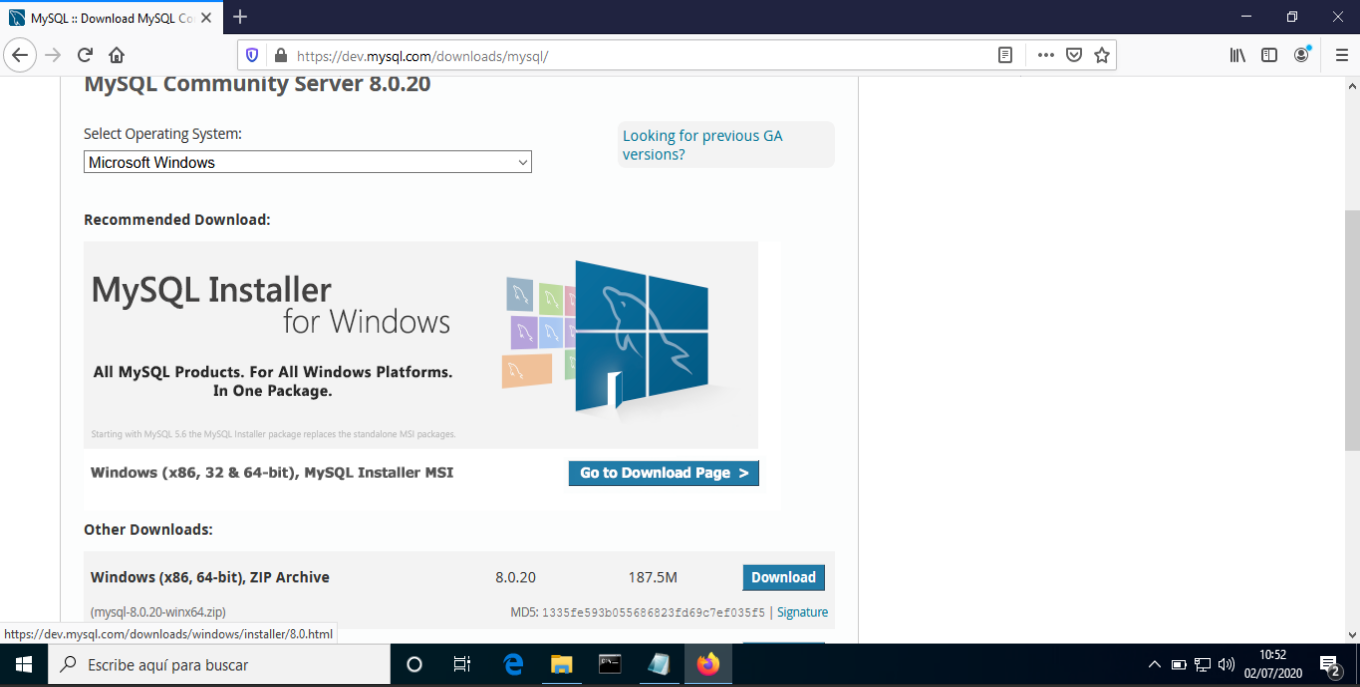


#### Figura D: Descarga de MySQL Server (paso 1)

****

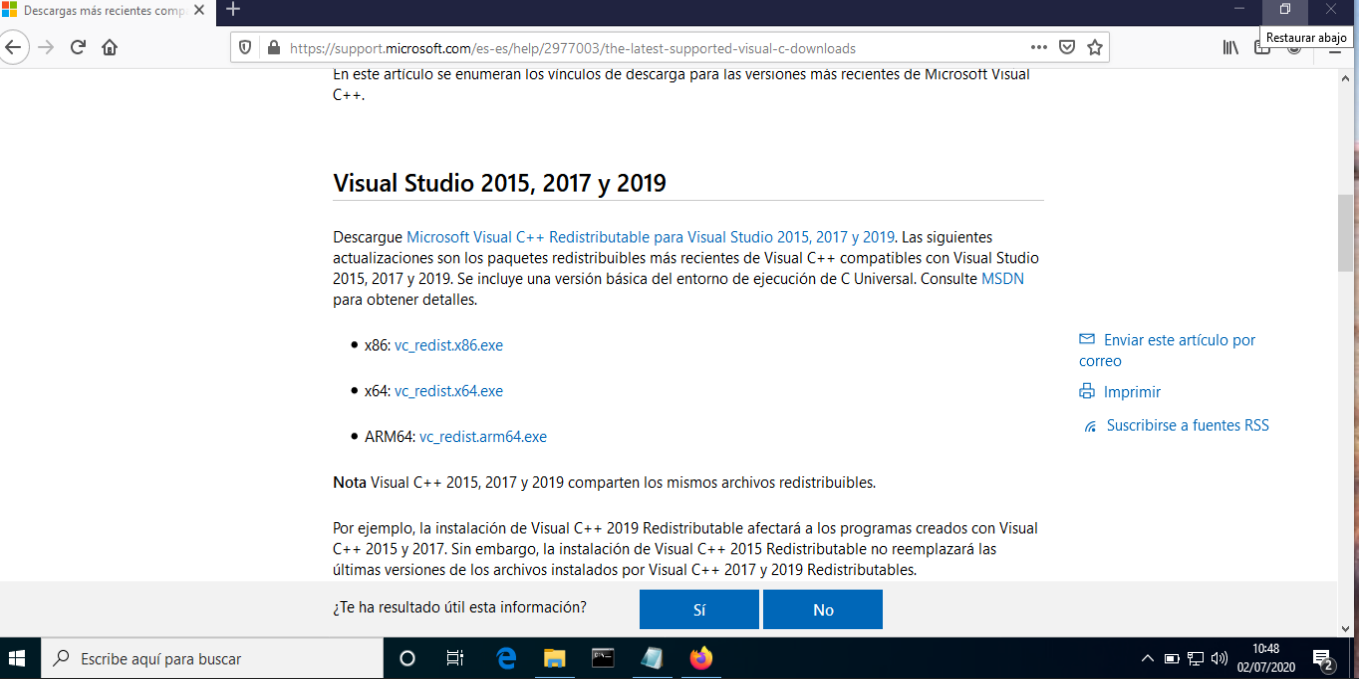
#### Figura D: Descargas de MySQL Community

Además, para Workbench es recomendable el paquete con el instalador, como se muestra en la figura D44.



#### Figura D: Descarga de My SQL Installer

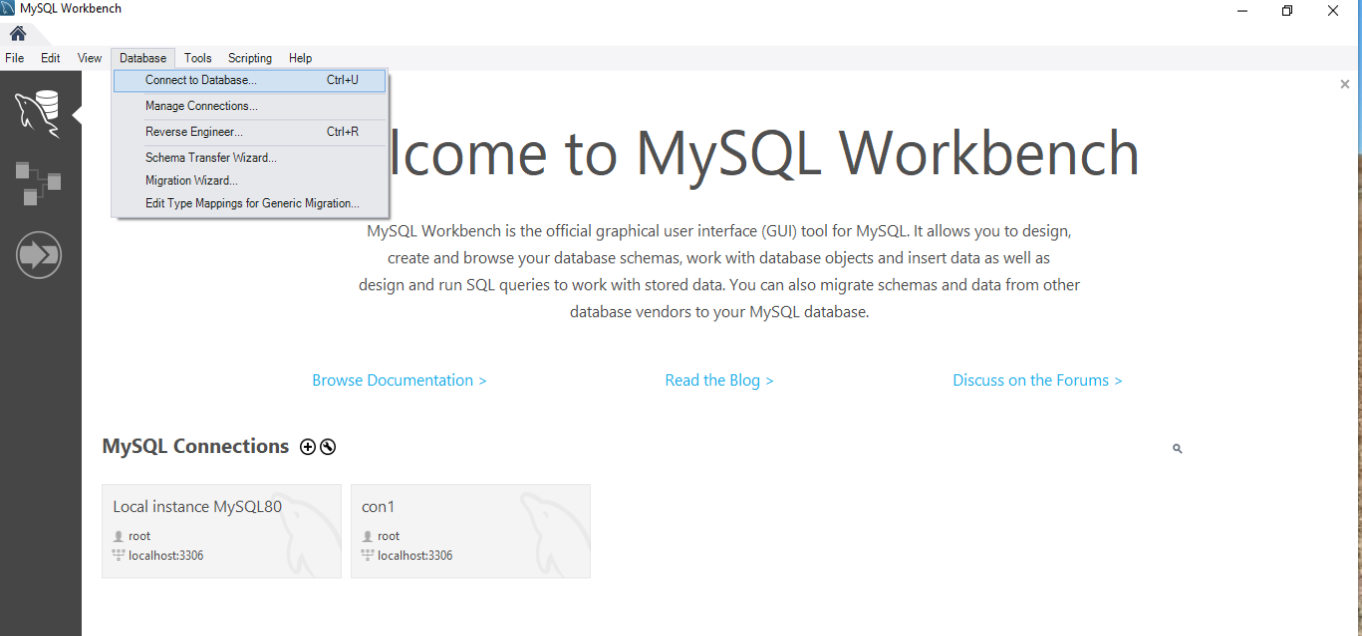
Esta última solicita librerías Visual C++ Redistributable actualizadas, si no se dispone de ellas pueden buscarse en el servicio oficial de Microsoft y descargarse, solo las librerías (<https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads>), sin necesitar ningún otro programa o entorno. (Figura D45.)



#### Figura D: Descarga de librerías Visual Studio C++ Redistributable

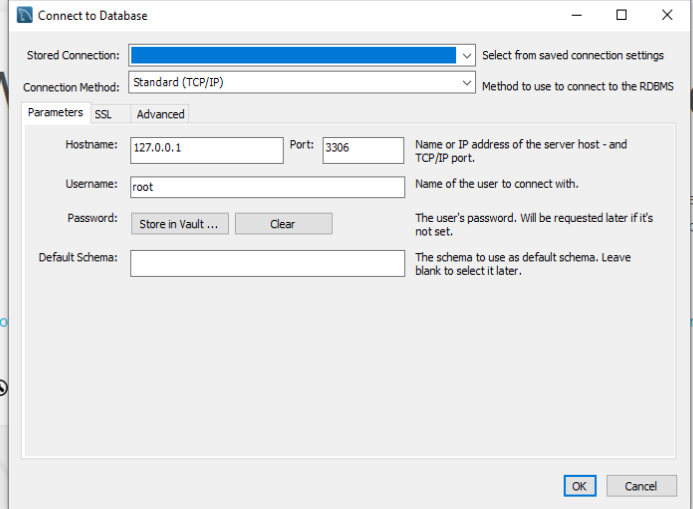
Durante la instalación, MySQL Server solicitará un nombre de usuario y una contraseña para establecer la conexión a la base de datos. Cambiar la contraseña después es un proceso tedioso, por lo que se recomienda ENCARECIDAMENTE apuntar estas credenciales.

Una vez instalado WorkBench, para crear una conexión a la base de datos, hay que seleccionar “database” en el menú superior, y “conect to database”. Se abre un cuadro para establecer la configuración (figura D46).



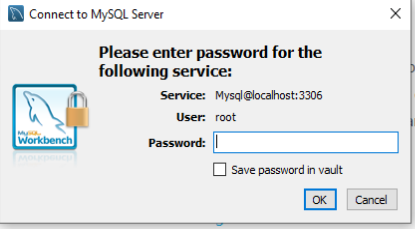
#### Figura D: Interfaz principal de MySQL Workbench

En los parámetros que aparecen por defecto solo se debe cambiar hostname, que tiene la dirección IP donde se va a conectar (tenemos que poner localhost), y después el nombre de usuario de la conexión (se puede utilizar cualquiera), como se ve en la figura D47.



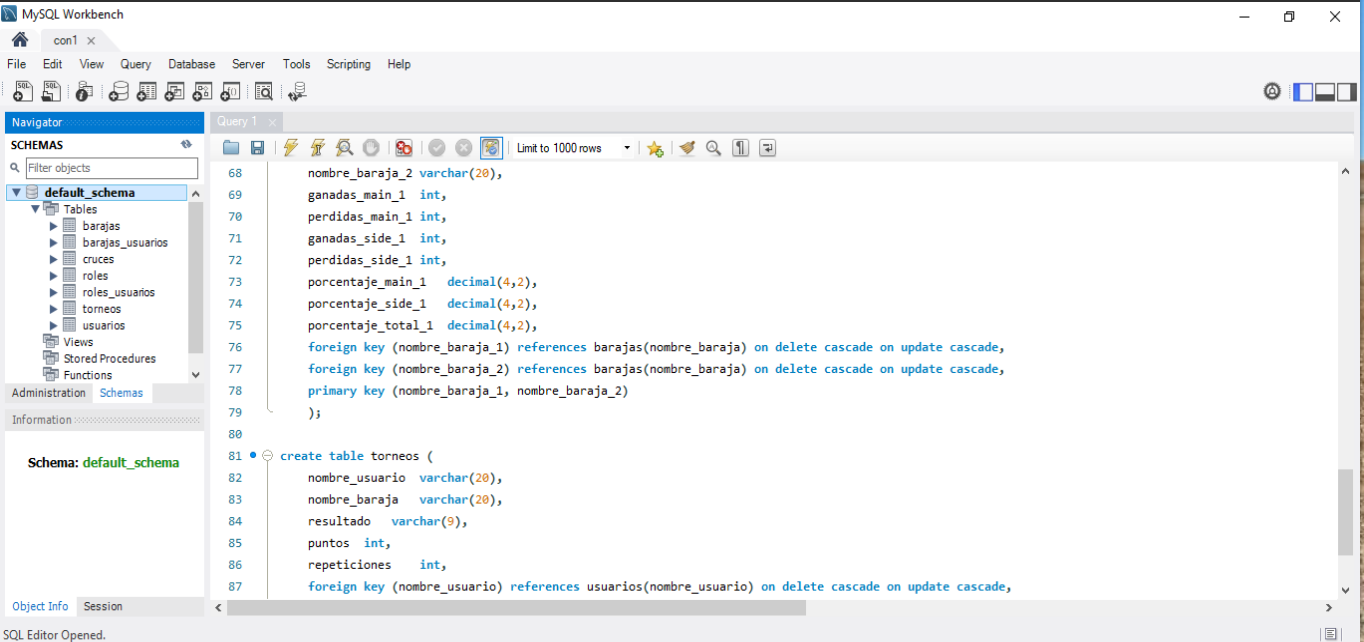
#### Figura D: Perfil de conexión a MySQL Server desde MySQL Workbench

Al aceptar e intentar utilizar esta conexión nos solicitará la clave establecida durante la instalación, como se ve en la figura D48.



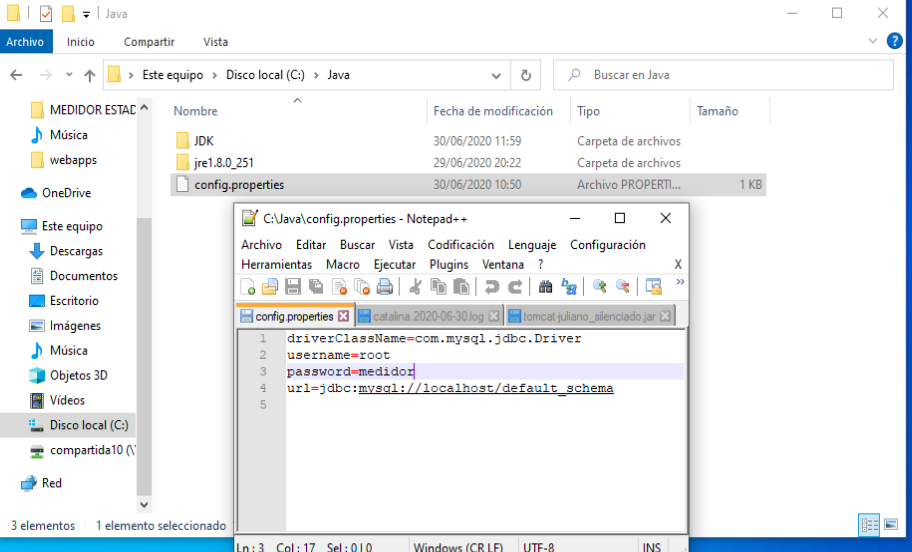
#### Figura D: Solicitud de contraseña de MySQL

Para que la aplicación pueda funcionar, se debe ejecutar el contenido del script “iniciar tablas” que está en el repositorio, en la carpeta “componentes” (figura D49).



#### Figura D: Página de trabajo de MySQL Workbench

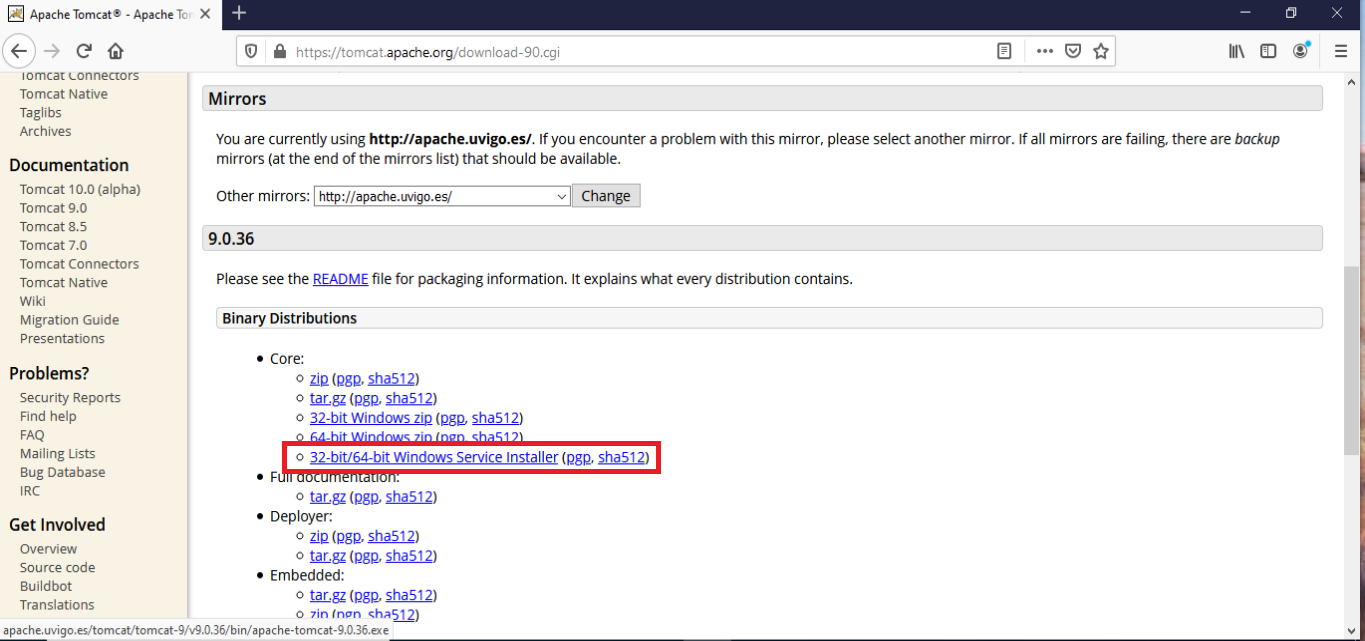
El nombre de usuario y contraseña utilizados para esta conexión deben establecerse también en las líneas 2 y 3 del fichero “config.properties”, aportado también en la carpeta componentes, como se ve en la figura D50.



#### Figura D: Ubicación y contenido del fichero config.properties

El último elemento es Apache Tomcat, el simulador de servidor. Lo descargamos de la web oficial (<https://tomcat.apache.org/download-80.cgi>) y después seleccionamos la última versión posible (el desarrollo de la aplicación se ha hecho desde la versión 8.5).

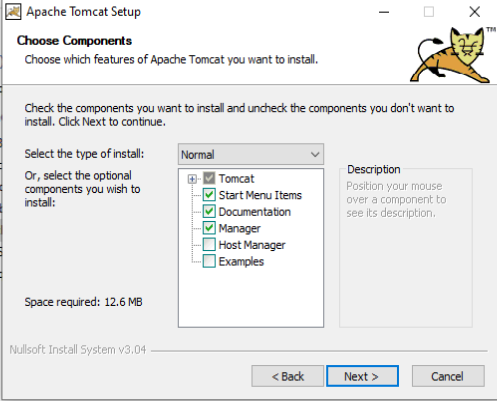
Como con MySQL, se descarga el instalable, como se muestra en la figura D51.



#### Figura D: Descarga de Apache Tomcat

Durante el proceso se deben seleccionar algunas cofiguraciones:

En el primer paso dejamos todo como está (figura D52).



#### Figura D: Instalación de Apache Tomcat paso 1

En el siguiente paso aparece la configuración de los puertos. El puerto para el servicio es el 8080, pero por defecto, el puerto para detener el servicio está configurado en -1. Se debe cambiar esto ya que el -1 impide cerrar los servicios de forma automática y solo se puede hacer matando el proceso. Cambiando este -1 por 8085 se habilita el cierre correcto del servicio (figura D53).

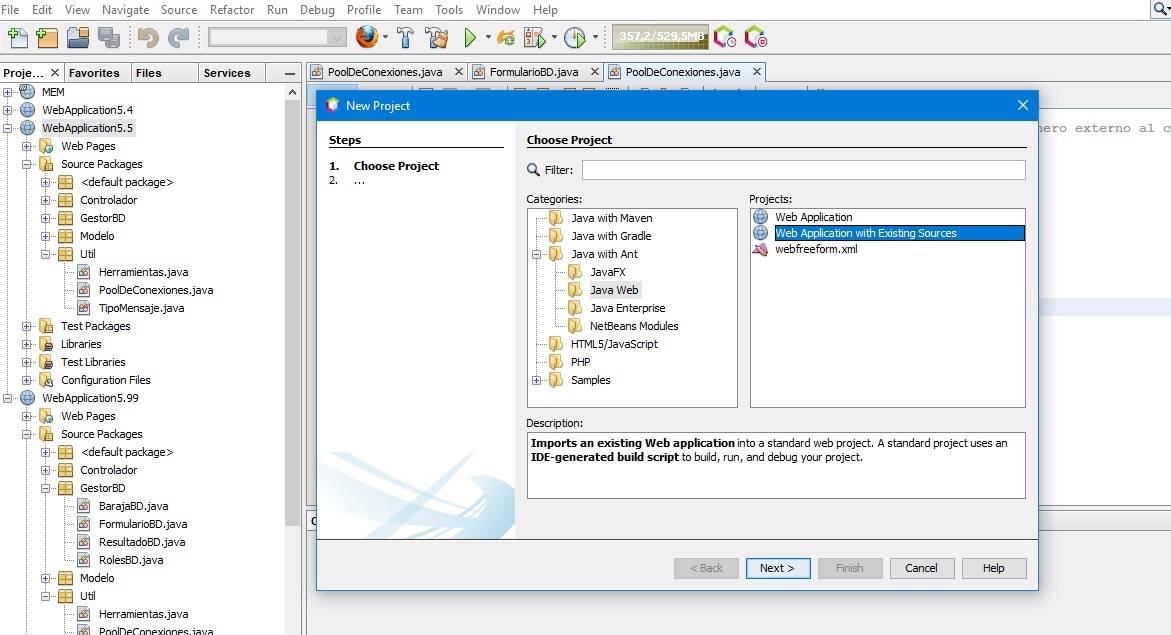


#### Figura D: Instalación de Apache Tomcat: configuración de puertos

Además, en esta misma pantalla solicita unas credenciales de conexión; este par usuario/clave se va a solicitar más tarde cuando queramos ejecutar Apache Tomcat desde el entorno de desarrollo NetBeans, por tanto, también es algo que hay que apuntar. La recomendación es utilizar el mismo par usuario/clave que para la conexión con MySQL Server.

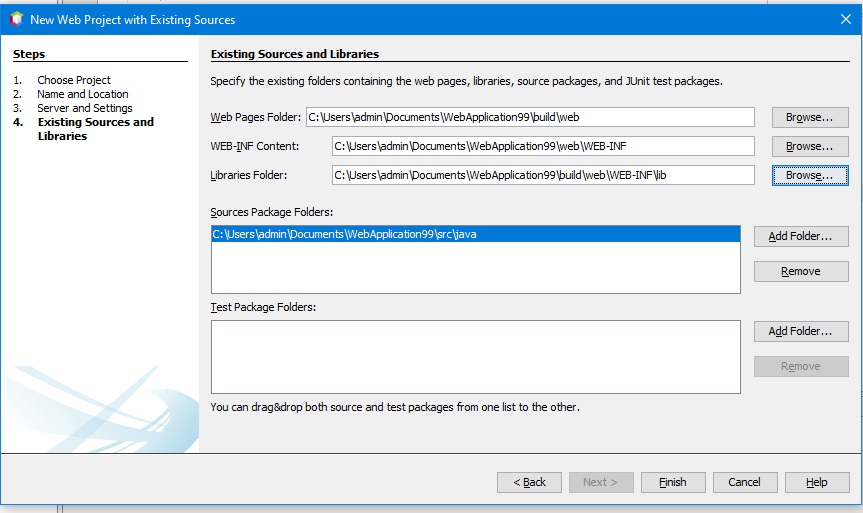
## D.3 Cómo importar el proyecto y continuar el desarrollo

Una vez que todos los componentes están instalados y configurados, hay que importar el proyecto a NetBeans. Se debe descomprimir el proyecto descargado en una carpeta, y luego abrir NetBeans e ir a: nuevo proyecto, y elegir la misma ruta que aparece en la figura D54:

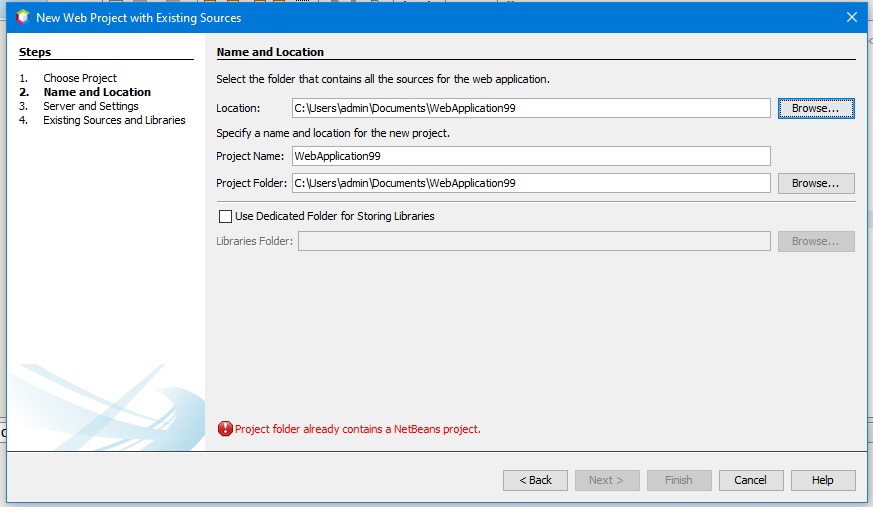


#### Figura D: Importación de proyecto en NetBeans paso 1

En el siguiente paso se selecciona en “location” la carpeta con el proyecto descomprimido y, aunque NetBeans rellena automáticamente el nombre del proyecto y el nombre de la carpeta en el workspace, ambos pueden cambiarse a gusto de consumidor (si el nombre del proyecto o de la carpeta donde se va a copiar pisan uno ya existente lo indica con un mensaje en rojo y no nos deja continuar…), como se ve en las figuras D55 y D56:

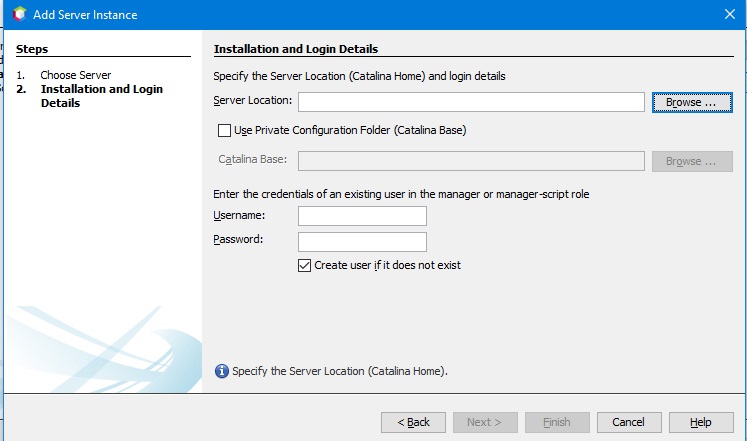


#### Figura D: Importación de proyecto en NetBeans paso 2 correcto

****

#### Figura D: Importación de proyecto en NetBeans paso 2 incorrecto

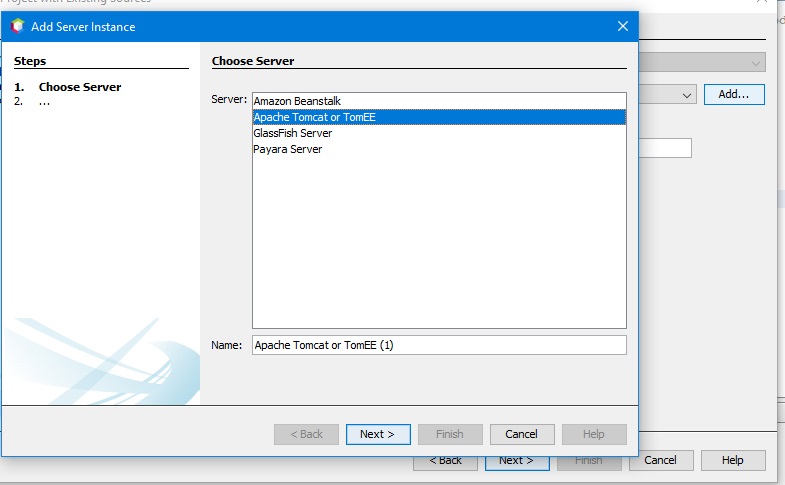
El siguiente paso solicita el servidor, si Apache Tomcat ha sido instalado correctamente podremos buscarlo (no aparecerá como en la imagen porque aún no se ha seleccionado), como se ve en la figura D57



#### Figura D: Seleccionar servidor paso 1

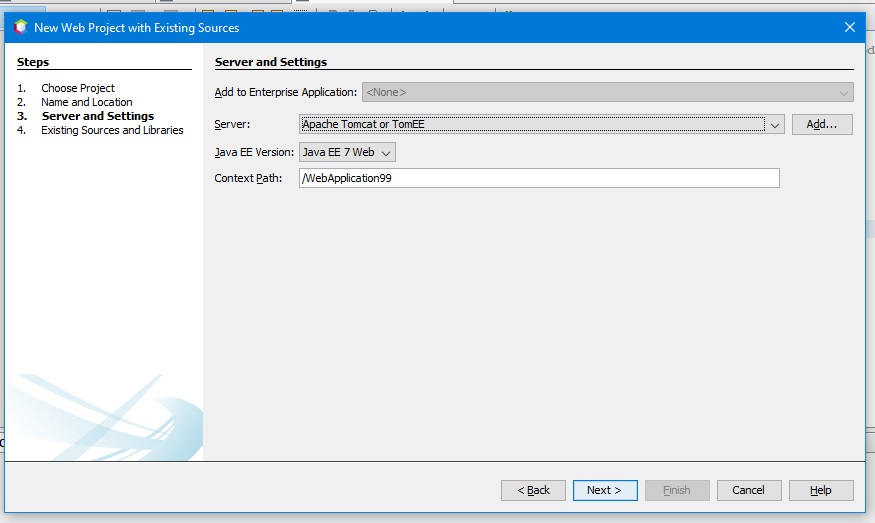
Damos a “add” y se abre un cuadro de diálogo, elegimos Apache Tomcat y damos a siguiente.

Elegimos la ruta y un perfil de conexión (estas credenciales nos las va a solicitar cada vez que queramos ejecutar o debugear la aplicación desde NetBeans. Consejo: utilizar las mismas que para la conexión a la base de datos), como se ve en la figura D58.



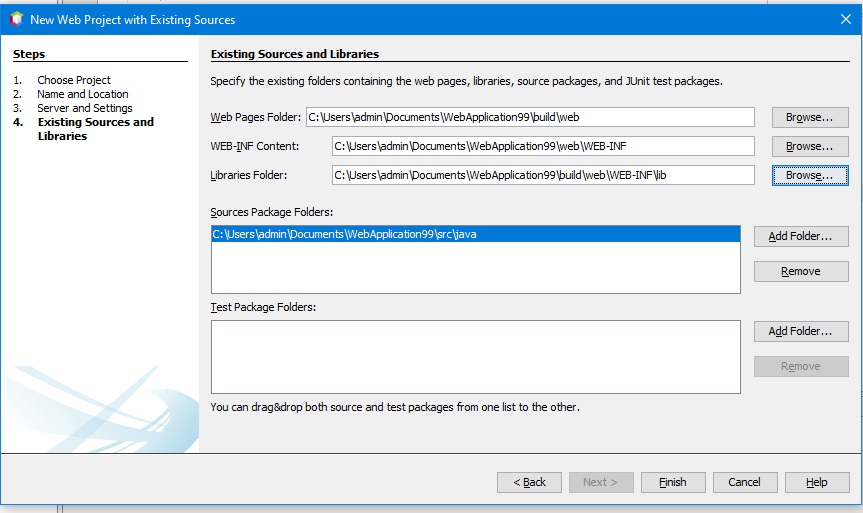
#### Figura D: Seleccionar servidor paso 2

Una vez que está configurado vuelve a mostrarnos la información del servidor y podemos continuar, como se ve en la figura D59.



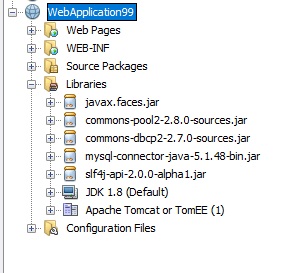
#### Figura D: Seleccionar servidor paso 3

El siguiente paso es indicar dónde están los elementos del proyecto: los ficheros fuente de código, las librerías y los ficheros web, como se ve en la figura D60.



#### Figura D: Importación de proyecto en NetBeans último paso

Por últimos damos en continuar y se termina de importar el proyecto. En la pestaña de la izquierda debería quedar algo similar a lo que aparece en la figura D61.



#### Figura D: Esquema del proyecto importado

Para poder probar la aplicación solo falta un paso más: el perfil de conexión a la base de datos viene dado desde un fichero externo (config.properties). En SourcePackages/Util/PoolDeConexiones.java, el método private PoolDeConexiones(), carga las propiedades desde un fichero. El método new File (“ruta”), indica la ruta donde se ubica este fichero y es donde va a buscar el fichero de configuración. Se puede modificar sin ningún problema, siempre que esta ruta y la ubicación del fichero config.properties coincidan.

Con todo esto hecho, en el menú superior hay que hacer clic en el botón con un martillo (built) para compilar la aplicación. Esto generará el fichero .war (por defecto en la carpeta /dist del proyecto).

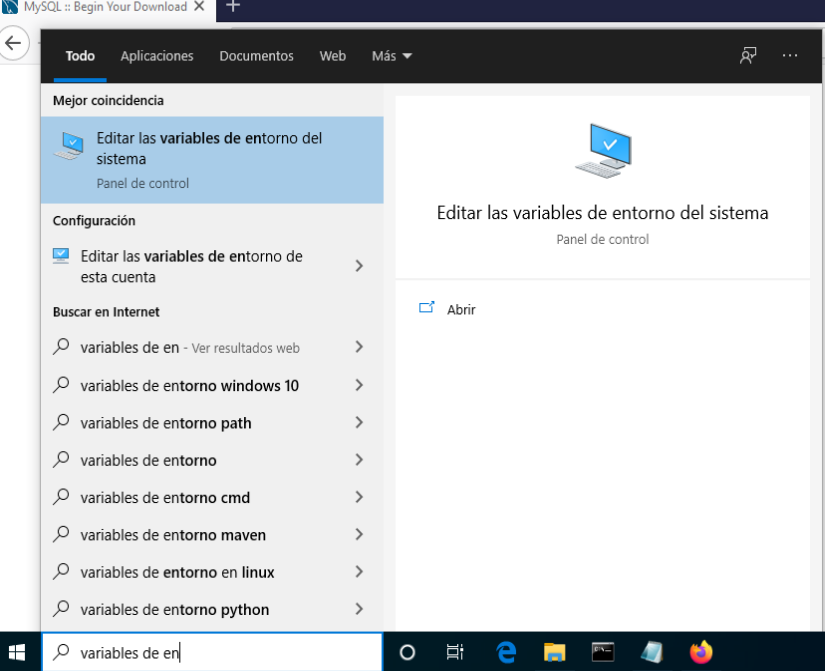
Ahora se puede dar al botón “play” con un triángulo en verde y la aplicación se ejecutará en el navegador seleccionado por Net Beans (en el botón a la izquierda del martillo, con una esfera del mundo, puede cambiarse el navegador).

## D.4 Ejecución fuera del IDE (desde el escritorio)

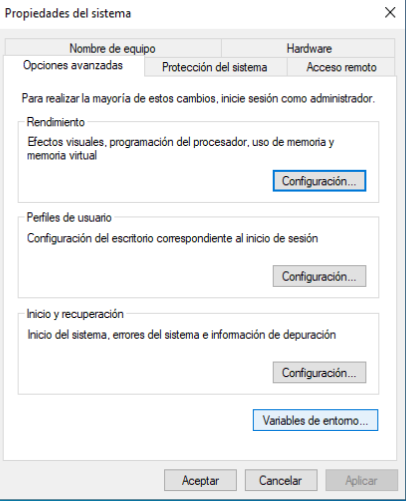
Para poder abrir la aplicación fuera del entorno de desarrollo hay que seguir los siguientes pasos:

Primero hay que definir las variables de entorno: JAVA\_HOME y CATALINA\_HOME.

Hay que abrir inicio, escribir “variables de entorno” y abrir la aplicación “editar las variables de entorno del sistema”. En la ventana que se abre, hacer clic en el botón “variables de entorno y se abrirá otra ventana (figuras D62 y D63).

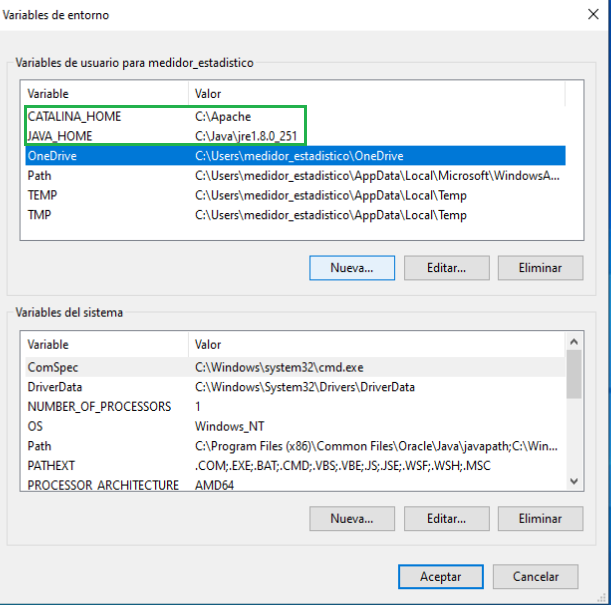


#### Figura D: Abrir las variables de entorno paso 1

****

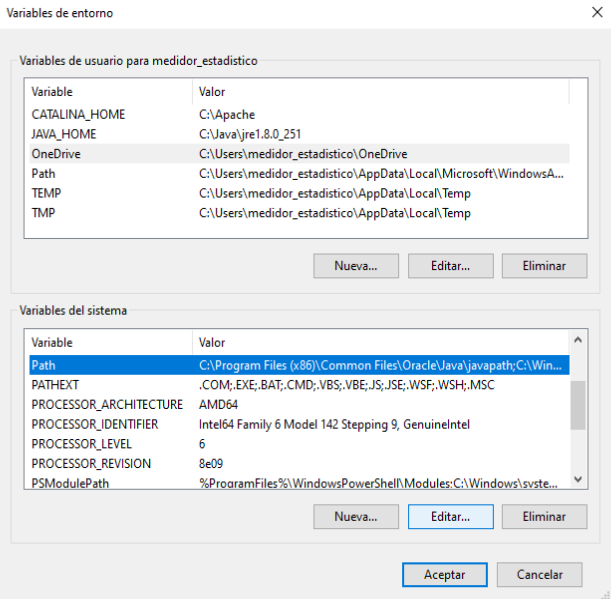
#### Figura D: Abrir las variables de entorno paso 2

Esta ventana tiene dos secciones divididas verticalmente; en la superior hay que pulsar “nueva”, y se abre un cuadro nuevo con dos opciones: nombre de la variable y ruta. Donde indica nombre de la variable hay que escribir CATALINA\_HOME, y en la ruta poner la ruta donde se ha instalado Apache Tomcat. Si no existe una variable JAVA\_HOME hay que hacer lo mismo, con la ruta del JRE (figura D64).



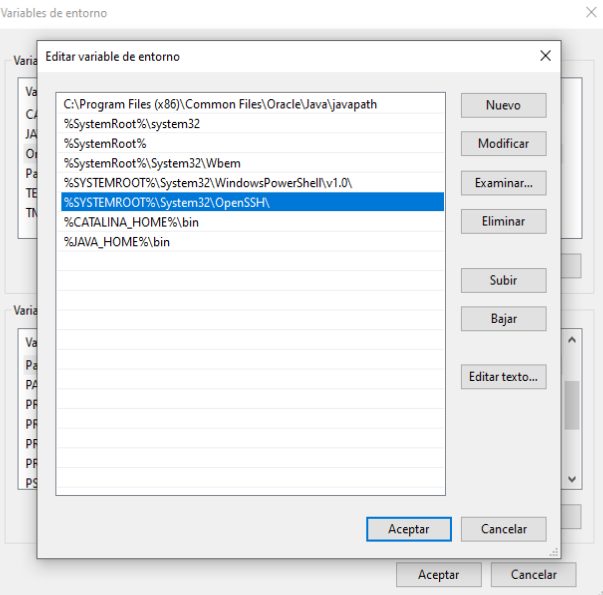
#### Figura D: Modificar las variables de entorno “Home”

Con las dos variables creadas, en la zona inferior, hay que seleccionar la línea que indica “path” y pulsar el botón “editar” (figura D65).



#### Figura D: Editar el path

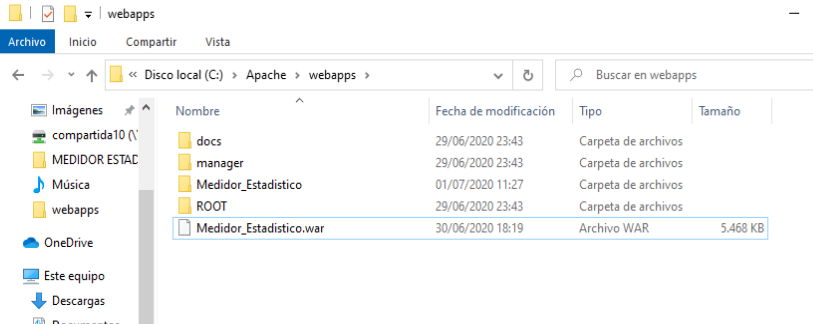
Sobre el cuadro nuevo que se abre hay que pulsar el botón “nuevo”, y escribir: %CATALINA\_HOME%\bin y aceptar. Si no está añadida JAVA\_HOME hay que hacer lo mismo (escribiendo el nombre de la variable entre paréntesis y añadiendo al final \bin como en la figura D66).



#### Figura D: Editar el path paso 2

Una vez que las variables se han definido, abrir una consola de comandos COMO ADMINISTRADOR y escribir “catalina start”. Si funciona correctamente se abrirá una nueva ventana de consola de comandos con título “Tomcat”, donde se verá la salida del log durante el inicio del servicio.

Después habrá que copiar el fichero aplicación.war a la ruta apache\webapps, y veremos cómo se crea una carpeta con el mismo nombre (figura D67).



#### Figura D: Ruta de Apache Tomcat para colocar el fichero .war

Finalmente hay que abrir una ventana de cualquier navegador web y en la barra de direcciones escribir: localhost:8080/[nombre de la aplicación] ejemplo:

Si el fichero compilado se llama Medidor.war, la url debe ser: “localhost:8080/Medidor” y la aplicación se ejecutará si todo es correcto.

Para cerrar los servicios correctamente, una vez que se cierra la pestaña del navegador sobre la que se ha ejecutado, se debe volver a la consola de comandos abierta previamente y escribir “catalina stop”.

## D.5 Estructura del código

Dada la documentación sobre el código con comentarios, exportada con Javadoc, aquí solo se comentará la distribución de paquetes y las tareas de cada clase java, así como detalles para tener en cuenta para la funcionalidad del proyecto.

La estructura del proyecto es la siguiente: todas las páginas web .XHTML cuelgan del mismo directorio, así como la carpeta css, donde están los ficheros .css. Los ficheros css utilizan nombres que dan pistas sobre su uso, o contenido, para que sea más fácil saber cuáles hay que abrir o modificar, sin tener que subir hasta la cabecera del HTML para ver el enlace al fichero css.

En lo que respecta a la estructura de las clases y los paquetes, están divididos de la siguiente forma:

El paquete modelo tiene las clases que corresponden a los objetos de la base de datos, y además alguna extra que se utiliza para obtener un conjunto de datos, actualizarlos y volver a guardados, como es el caso de la clase resultadoUsuarioBaraja, que tiene como atributos los distintos datos de un usuario con una baraja (victorias de main, derrotas de main…) que leerán de la base de datos la fila del usuario U con la baraja B, tendrá los métodos que permitirán introducir los datos a actualizar (las nuevas victorias y derrotas cosechadas por el usuario durante un torneo), actualizará los datos de la instancia, sumando los que había cargado de la base de datos más los nuevos, y recalculando después los porcentajes, y permitiendo luego pasarle ese objeto al método correspondiente que vuelva a meterlos en la base de datos. Funcionan como encapsuladores de datos.

El paquete útil tiene la enumeración de los tipos de mensajes que se pueden devolver, el pool de conexiones y una clase herramientas con el método que lanza los mensajes y el método “tratarNombre”.

El paquete GestorBD contiene las clases que trabajan contra la base de datos. Estas clases están nombradas acorde a los datos que manejan con la base de datos, por ejemplo, formularioBD maneja los datos referentes al login, registro de nuevos usuarios y carga de roles.

Por último, el paquete controlador tiene las clases ManagedBean, que son la clave de la funcionalidad.

La clase ManagedBean es la piedra angular del framework JavaServerFaces para interconectar la vista (páginas HTML) con el controlador (código Java).

Las clases ManagedBean tienen una serie de atributos, que deben tener su métodos get y set, para poder ser accedidas desde las vistas .html. La conexión se establece de la siguiente forma: en HTML, todos los campos de entrada de texto (bien input o inputSecret), tendrán una mención al managed bean: {#nombre\_managed\_bean.atributo} para guardar el valor necesario. Por otro lado, los campos de impresión de texto en el fichero html, podrán leer el contenido de los atributos del managed bean con la misma nomenclatura. OJO: si no existe método get y se intenta leer un atributo dará un error al cargar la página, lo mismo que si intentamos grabar un valor sobre un atributo que no tiene método set.

La nomenclatura de los métodos get y set está establecida por el framework, y es la siguiente: getAtributo, setAtributo. Ejemplo: si un atributo es nombre\_usuario, los get y set deben ser: getNombre\_usuario() y setNombre\_usuario(String param). (El nombre del parámetro del método set no está sujeto a ninguna restricción de nombre).

## D.6 Extra: procesos que requieren autenticación

Para la ejecución desde Net Beans es necesario tener el controlador de MySQL en el proyecto, y el servidor Apache Tomcat. Antes de desplegar la aplicación, Net Beans arranca el servicio de Tomcat y después lanza la aplicación. Al iniciarlo solicita unas credenciales, estas son las que se introdujeron durante la instalación de Apache Tomcat.

Una vez que se ha abierto la aplicación, tanto el login como el sistema de registro de nuevos usuarios utilizan la conexión a la base de datos, por tanto, si dicha configuración es incorrecta, la aplicación no pasará de este punto. Esta conexión se obtiene mediante la clase “pool de conexiones”, en el método del constructor privado, utilizando las cadenas del fichero de texto “config.properties”, que se obtiene de una ruta, modificable en el código. Dicho fichero se aporta, pero contiene los valores que permiten la ejecución con los elementos de la máquina virtual proporcionada en el repositorio, sin modificar parámetros. Como se explica anteriormente en este mismo manual, y para realizar pruebas, se aconseja probar el nombre de usuario y la contraseña que aparecen en el fichero con la aplicación My SQL Workbench, en una nueva conexión a la base de datos.

Para el despliegue local, una vez montado el fichero .war con Net Beans (con el icono del martillo), habrá que colocar el fichero [nombre].war en la ruta donde se haya instalado Apache Tomcat, en la carpeta “webapps”. Después se ejecuta la aplicación Símbolo del sistema como administrador y se arranca el servicio Tomcat con “catalina start”. Una vez que el servicio se haya inicializado, se podrá abrir una ventana del navegador y escribir: localhost:8080/nombre (como se indica también en este manual, el puerto es variable).

# APÉNDICE E: ESPECIFICACIÓN DE DISEÑO

## E.1 Introducción

Este capítulo expone cómo se estructuran los datos en la base de datos, apoyada en diagramas entidad relación realizados en la página <https://cloud.smartdraw.com/>.

## E.2 Glosario de términos

Para comprender la estructura de la base de datos y la información que se trata es necesario definir los conceptos que se utilizan, ya que serán las palabras clave las que se utilicen como nombres de los campos en las tablas.

-Usuarios: usuario es todo actor que pueda acceder al sistema, identificado por su nombre de usuario.

-Barajas: una baraja se identifica únicamente por su nombre y representa todo el conjunto de configuraciones distintas (60 de base y 15 de banquillo) para esa baraja concreta, sin hacer distinciones.

-Rondas: una ronda de torneo es un enfrentamiento entre dos jugadores (cada uno utilizando una baraja). A efectos del tratamiento de datos, una ronda es un enfrentamiento entre dos barajas, que cuenta con partidas de main y side.

-Partida: una partida siempre involucra a dos barajas y su resultado solo puede acabar en victoria para una de las dos.

-Main: todos los lugares donde aparece el término “main” se refiere a “partidas de main”, que significa partidas jugadas con las sesenta cartas de base.

-Side: todos los lugares donde aparece el término “side” se refiere a “partidas de side”, que significa partidas jugadas después de hacer cambios entre las sesenta cartas de base y las quince de banquillo.

-Resultado de las rondas: no se utiliza en la base de datos pero es necesario definirlo. Una ronda tiene una duración predeterminada por las reglas del torneo (comúnmente 50 minutos). La ronda termina en empate si, al cumplirse este tiempo, el número de partidas ganadas por cada jugador es igual.

-Tier: es un atributo numérico de la baraja que representa su popularidad: cuanto más se juegue una baraja, mayor tier tendrá. Se puede reconocer porque, al ver las páginas donde se lista el metajuego ([www.goldfish.com](http://www.goldfish.com) o [www.mtgtop8.com](http://www.mtgtop8.com)), aparecen ordenadas.

Los resultados de los torneos se almacenan como una cadena con el formato: rondas ganadas, rondas perdidas, rondas empatadas, de modo que “3-1” significa tres victorias y una derrota, “3-0-1” significa tres victorias, cero derrotas y un empate, y “2-1-1” dos victorias, una derrota y un empate. Internamente, durante el cómputo de rondas del torneo, se utilizará el cálculo de 3 puntos por ronda ganada y 1 punto por ronda empatada, para valorar el resultado en base a un entero y poder devolverlos ordenados de mejor a peor puntuación. Además, se guardará qué jugador ha realizado ese torneo y con qué baraja, y por último el número de veces que se ha dado ese caso (si el mismo jugador con la misma baraja realiza dos o más veces el mismo resultado es más fácil e intuitivo mostrar un “x2” o “x3” que la misma cadena “G-P-E” dos o tres veces seguidas).

Los cruces son el grueso de la aplicación; mostrar el ratio de victorias de cada baraja frente a cada una de las demás. Cada ronda de torneo que se procesa involucra a dos barajas, como se explicó anteriormente. La idea de esta tabla es recoger los resultados de cada par de barajas, de forma que se pueda filtrar por barajas e ir viendo todos los cruces para cada una.

Esta tabla tiene un diseño espejo: para el cruce A-B existe también el cruce B-A, y está decidido así porque, para mostrar al usuario todos los cruces de una baraja, es más eficiente tener resultados duplicados en la base de datos y devolverlos en una única operación de consulta, que tener que ir comprobando si la baraja X está en el campo “baraja\_1” y, en caso de no estar, hacer otra consulta de todas las entradas con la baraja X en el campo “baraja\_2” y luego invertir los datos para mostrárselos al usuario.

Los roles se almacenan con un entero que funciona como clave primaria y la descripción, que será la que el usuario vea cuando quiera actuar sobre peticiones, y se deja abierto a una línea futura el añadir más roles, siendo siempre el menor número, incluido el 0, el que mayores privilegios tiene.

Y finalmente la tabla que relaciones los roles con los usuarios, añadiendo el campo “estado” que puede ser: 0 (concedido), 1 (solicitado y en espera) y 2 (denegado), de forma que un usuario solo puede tener un estado concedido y una única petición para un estado que NO tiene concedido Y que tiene mayores privilegios que el actual.

## E.3 Computación de los datos

Por un lado, se almacenan los datos de cada jugador (rondas y partidas que gana, pierde y empata, además de los ratios porcentuales). Por otro lado, para cada baraja, se almacenan las partidas que gana y pierde, de main y side, para obtener los porcentajes de victorias de cada baraja, de main, side y el total.

Finalmente, se almacena la información que relaciona a cada usuario utilizando cada baraja (no es obligatorio que cada jugador utilice todas las barajas, y de hecho, lo más frecuente es que cada jugador no tenga más de cuatro o cinco barajas, y existan un total de veinte o veinticinco barajas). Esta información implica las partidas y rondas que gana un jugador utilizando una baraja concreta.

Si un jugador, en una ronda, gana la primera partida con la baraja principal y después pierde las dos con banquillo habrá ganado una partida de main y perdido dos de side. Además, habrá utilizado una baraja X y su oponente otra baraja Y. Y puede ser igual a X, pero, si no lo fuera, los datos que se procesan son los siguientes:

-La baraja X ha ganado una partida de main y ha perdido dos de side, para sus estadísticas como baraja aislada.

-La baraja X ha ganado una partida de main y ha perdido dos de side contra la baraja Y, que por su parte ha perdido una de main y ha ganado dos de side contra la baraja X, datos que serán procesados para el cruce entre estas dos barajas.

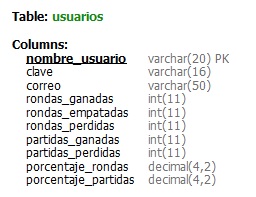
-El jugador que registra el torneo ha ganado una partida de main y ha perdido dos de side, por un lado, con la baraja X, por otro lado, para sus datos generales, y además habrá perdido una ronda del torneo en curso.

Como dato relevante se mostrarán al usuario ratios porcentuales de toda estadística valorable, como rondas de torneo ganadas, partidas ganadas, datos del usuario con cada una de las barajas que utilice y datos de las propias barajas.

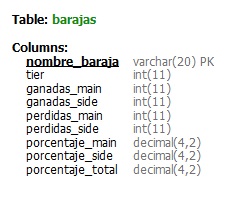
Todas las barajas se reconocen por un nombre concreto, por tanto, se almacenará y referenciará a las mismas por dicho nombre. Además, tienen un dato, tier, que hace referencia a su nivel: las barajas con tier 1 son las que más se juegan en torneos y además las que mayor número de victorias cosechan.

## E.5 Tablas y campos

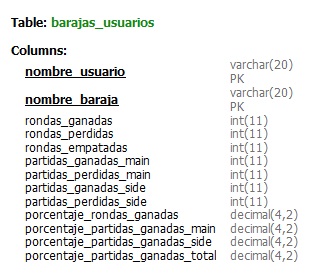
A continuación, se muestran los nombres de las tablas con el nombre y tipo de sus campos:



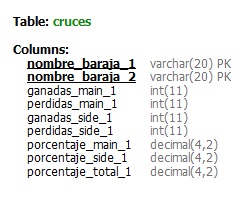
#### Figura E: Tabla usuarios



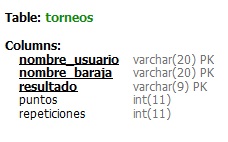
#### Figura E: Tabla barajas



#### Figura E: Tabla barajas\_usuarios



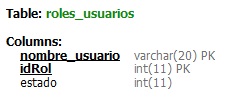
#### Figura E: Tabla cruces



#### Figura E: Tabla torneos



#### Figura E: Tabla roles



#### Figura E: Tabla roles\_usuarios

III- Bibliografía

«aplicaciones Web Java.pdf».

«Guía para la redacción de casos de uso | Marco de Desarrollo de la Junta de Andalucía». http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416 (10 de junio de 2020).

«Metodología SCRUM: ¿qué es y cómo aplicarlo en tu trabajo? | Sinnaps». *Gestor de proyectos online*. https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum (15 de octubre de 2019).

«Requerimientos Funcionales y No Funcionales, ejemplos y tips | by Requeridos Blog | Medium». https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a (11 de junio de 2020).

Roca, David Campos. 2015. «Los 5 mejores Hosting Web: el gratuito, el barato y el mejor». https://aulacm.com/mejores-hosting-web-alojamiento/ (20 de junio de 2020).

1. En la versión Beta no se verifica que el correo introducido sea correcto ya que es una versión que solo funciona a nivel local. [↑](#footnote-ref-1)
2. En la versión BETA solamente hay tres tipos de cuenta posibles, de los cuales solo dos están en uso: “Administrador” y “Estándar”, pero está programado para admitir nuevos haciendo los cambios únicamente en la base de datos. [↑](#footnote-ref-2)
3. Las barajas que el usuario puede añadir a su perfil deben haber sido añadidas previamente a la base de datos. Si no hay ninguna se debe consultar la sección “Gestionar barajas como administrador”. [↑](#footnote-ref-3)
4. El usuario solo podrá seleccionar una baraja de entre las que posee, por lo que si no se ha asignado ninguna baraja no podrá registrar ningún torneo. Ver sección 3: Asignación de barajas. [↑](#footnote-ref-4)
5. Metajuego es el conjunto de las barajas que son suficientemente competitivas para considerarse punteras dentro de un formato. [↑](#footnote-ref-5)