UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA
SUPERIOR
Grado en Ingeniería en Informática

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 e José Ignacio Santos Martín

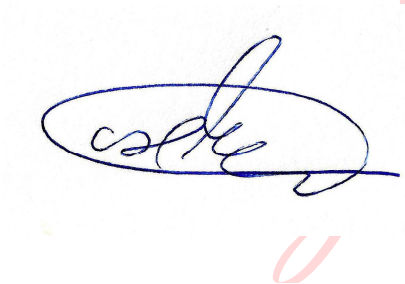
D. José Manuel Galán Ordax, profesor del departamento de Ingeniería de Organización, área de Organización de Empresas

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



Firmado digitalmente por GALAN ORDAX JOSE MANUEL - 71121824M
Fecha: 2020.07.20 12:29:50 +02'00'

## Resumen

Este proyecto creará una aplicación web que permita ayudar a los jugadores del juego *Magic: the Gathering* a tomar decisiones de cara al juego debido a su complejidad computacional. La aplicación resultante será un analizador de estadísticas que computará las partidas registradas para ofrecer a los usuarios un cuadro-resumen con los datos resultantes.

A la hora de participar en un torneo, cada jugador debe elegir una baraja y no podrá cambiarla durante el mismo. Estos torneos constan de varias rondas que enfrentan a los jugadores en uno contra uno, en un formato de liga.

La complejidad de elegir la baraja reside en la cantidad de cartas impresas utilizables, y el marco, ya generado, de barajas "ganadoras" que reduce considerablemente las opciones de victoria si no se escoge una de esas veinte, veinticinco o treinta barajas "ganadoras".

Esta herramienta pretende ayudar a los jugadores en dicha decisión computando la mayor cantidad posible de enfrentamientos entre distintas barajas para conocer cuáles son las que tienen unos mayores índices de victoria.



## Descriptores

Medidor Estadístico de Metajuego, Magic: the Gathering, Arquetipos, Aplicación web.

## Abstract

This project will create a web application that helps Magic: the Gathering game players make decisions regarding the game due to its computational complexity. The resulting application will be a statistics analyzer that will compute the registered games to offer users a summary table with the resulting data.

When participating in a tournament, each player must choose a deck and cannot change it during it. These tournaments consist of multiple rounds pitting players against each other, in a league format.

The complexity of choosing the deck lies in the number of usable printed cards, and the frame, already generated, of "winning" decks that considerably reduces the victory options if one of those twenty, twenty-five or thirty "winning" decks is not chosen.

This tool aims to help players in this decision by computing as many matches as possible between different decks to find out which ones have the highest victory rates.



## Keywords

Metagame Statistical Metre, Magic: the Gathering,

# Índice de contenido

# Índice de figuras

# 1- Introducción

## 1.1- Introducción general

¿Qué es *Magic: the Gathering* y cuál es el objetivo de este proyecto?

El presente proyecto tiene como idea principal desarrollar una aplicación web para jugadores del juego de cartas coleccionables *Magic: the Gathering*. A continuación, se explica muy por encima en qué consiste el juego. Para profundizar un poco más, un jugador muy conocido a nivel mundial, *Reid Duke*, redactó el artículo "¿Qué es Magic?" para la página oficial (¿Qué es Magic? 2015).

Este juego de cartas coleccionables nació en el año 1993, de la mano de un profesor de matemáticas estadounidense (*Richard Gardield*). Consiste en dos jugadores representando a dos hechiceros, que se enfrentan, cada uno con su propia baraja de cartas.

A lo largo de los años el juego ha ido evolucionando, se han ido imprimiendo nuevas cartas (cada año salen al mercado cuatro ediciones nuevas y cada edición consta de unas doscientas cartas) y su accesibilidad al público se ha incrementado, permitiendo la iniciación a nuevos jugadores de forma recurrente.

Las tiendas especializadas realizan torneos de forma regular, una o varias veces por semana, de distintos formatos[1], e incluso varios torneos el mismo día, como muestran las figuras 1 y 2:

[1] FORMATOS: Con la amplia variedad de cartas a elegir, Wizards of the Coast regula los formatos, determinando en cada uno de qué ediciones pueden utilizarse cartas y teniendo cada uno su propia lista de cartas prohibidas. Esto hace que cada formato tenga una dinámica (partidas más cortas, más lentas, formatos especiales para partidas de más de dos jugadores…)

Wizards of the Coast es la organización que regula el juego, diseña e imprime las cartas, vende los productos a las tiendas especializadas, regula los formatos de torneos y organiza los torneos de índole mundial.

Figura 1: Hoja de torneos de una semana de la tienda Avalon en Burgos

Figura 2: Calendario de torneos de un mes de la tienda Itaca de Madrid

El objetivo de este proyecto es desarrollar una aplicación web donde los jugadores puedan registrar sus resultados durante los torneos y tener acceso a sus estadísticas y, apoyándose en ellas, mejorar la toma de decisiones, debido a la complejidad del juego.

## 1.2- Motivación personal

¿Qué tiene de especial *Magic: the Gathering*? Nacido como un juego de matemáticas (sumas y restas), con su evolución ha ido adquiriendo nuevas reglas y mecánicas y tiene una amplia variedad de conceptos que en el mundo de la informática son muy comunes (tales como la pila, los triggers, las interrupciones, reglas de prioridad…)

A menudo se ha intentado comparar con otros juegos a nivel de dificultad computacional, ya que en muchas ocasiones se han lanzado productos digitales para permitir conocer el juego de forma más automatizada, amena y gratuita, pero siempre se ha encontrado con una Inteligencia Artificial sin apenas dificultad.

Se han hecho estudios sobre la complejidad computacional del juego, como este del año 2019: (Churchill, Biderman, y Herrick 2019), tratando de desglosar esa complejidad, y artículos hablando sobre "un juego tan completo que ni las máquinas saben cómo ganar": (Pérez 2020).

Entre los jugadores esto se traduce en un amplio abanico de toma de decisiones en cada jugada y la incertidumbre de saber con exactitud, en la mayoría de los casos, que va a suceder después, en base a la jugada con la que el oponente responda.

La parte competitiva del juego, el Magic profesional y los torneos de peso y con ganancia económica, hay que hacer mención a los Grand Prix, Pro Tour y mundial.

El Grand Prix (Grand Prix 2020) se celebra aproximadamente cada dos semanas, en ciudades a lo largo del mundo, y permiten inscripciones en función del aforo permitido en el lugar donde se celebren, como se puede ver en el artículo citado, algunos rondan una participación de 7500 jugadores, aunque lo habitual es que sean entre 1000 y 1500 (Grand Prix (<i>Magic 2020)) La organización de Wizards of the Coast (MAGIC 2003), encargada de estos eventos, dispone de un calendario a largo plazo con la programación de dichos eventos (https://magic.wizards.com/es/events/schedules#/list), aunque a fecha de escribir este documento (mayo, junio de 2020), no hay

eventos programados debido a la reducción de reuniones sociales a causa del Covid-19 (COVID-19 2020).

Sin entrar a detalles sobre los otros dos eventos mencionados, solo cabría mencionar que el Pro Tour (gira profesional) (<i>Magic 2020) se celebra cuatro veces al año y tiene un sistema clasificatorio que hace muy complicado llegar, y que el mundial (Galaxy 2017) se celebra solo una vez al año, además de que ambos reparten ganancias económicas que rondan los 50.000$ y los 15.000$, respectivamente.

# 2- Conceptos teóricos

Para explicar los conceptos teóricos de la aplicación es necesario tener un contexto sobre cómo funcionan tanto los torneos como los formatos, por tanto, se va a invertir el orden de los capítulos a fin de permitir comprender mejor el objetivo de este proyecto.

## 2.1- Formatos y precios

Anteriormente se ha mencionado la existencia de varios formatos, cada uno con sus propias cartas permitidas. La gran división se hace entre los formatos construidos (el jugador debe traer su propia baraja al torneo) y los limitados (el jugador recibe cierta cantidad de cartas como parte del torneo y debe jugar con ellas). Esta es la página oficial que explica los formatos y sus reglas: https://magic.wizards.com/es/content/formatos.

Este proyecto se basa únicamente en los formatos construidos, donde el jugador debe llevar su propia baraja.

Los formatos construidos tienen una peculiaridad, y es que las barajas "que ganan" están muy determinadas. Existe un conjunto de varias barajas y, por lo general, toda baraja fuera de este conjunto no cosecha ningún resultado positivo (salvo de forma muy esporádica, quizá cada dos años, con la impresión de una carta nueva, es posible construir una baraja en torno a esta nueva carta, la baraja funciona (gana) y entra en este conjunto de "ganadoras").

¿Cuántas barajas ganadoras hay? Hemos hablado de los formatos construidos, en plural, y los más comunes y jugados son: Estándar, Modern y Legacy.

Estándar solo permite utilizar las cartas impresas en los dos últimos años, y es complicado ver a la vez más de tres barajas ganadoras. El dato económico es que una baraja competitiva (ganadora) de este formato puede rondar los 300 o 400€, pero rara vez durará más de seis meses o un año.

Modern permite cartas impresas desde el año 2005, y su conjunto de ganadoras ronda entre las quince y veinte barajas. El precio de las barajas ganadoras de Modern puede oscilar entre los 600 y los 1200€, pero tienen una línea temporal mucho más sólida (una baraja en este formato puede durar diez años o más sin ningún problema).

Probablemente por este dato sea el formato favorito de la comunidad de jugadores y el más demandado.

Legacy, por último, permite cartas impresas en cualquier época, lo cual implica el uso de cartas más cerca de considerarse reliquias (que pueden rondar los 500€) lo cual hace que las barajas competitivas tengan un precio demasiado alto y no sea un formato muy demandado por jugadores que no poseen dichas cartas.

Para cerrar el tema de los formatos, es necesario indicar quién fija estos precios. Antiguamente las tiendas especializadas vendían cartas a granel y fijaban sus propios precios, pero actualmente existe la página *Cardmarket* (https://www.cardmarket.com/es/Magic) donde cualquier usuario puede comprar y vender cartas y en la fijación de precios también influyen jugadores que quieren vender cartas.

## 2.2- Rondas de torneo, main y side.

Durante un torneo de formato construido, cada jugador no podrá variar su baraja, y deberá jugar todo el torneo con la misma. Cada baraja debe tener un mínimo de sesenta cartas de base y quince de banquillo.

Los torneos tienen un formato de liga con emparejamientos semidirigidos, en la primera ronda todos los emparejamientos son aleatorios y, a partir de la segunda, el emparejamiento es aleatorio pero siempre contra un jugador con la misma cantidad de rondas ganadas en ese torneo.

Cada ronda se juega siempre al mejor de tres partidas y tiene un tiempo máximo de cincuenta minutos. En cada una de estas rondas, se decide qué jugador empieza en la primera partida de forma aleatoria, y después el perdedor de cada partida elige quién empezará la siguiente.

De estas dos (o tres) partidas por ronda, siempre la primera se realiza con las cartas de base de la baraja (llamaremos a esto main), y tras dicha partida, los jugadores pueden intercambiar cualquier cantidad de cartas de su base con su banquillo, siempre que sigan jugando con, al menos, sesenta cartas. Estas partidas las consideramos side.

## 2.3- Barajas, listas y metajuego

A efectos prácticos, se puede coger cualquier conjunto de cartas que cumpla las reglas de ediciones permitidas y el mínimo de sesenta y sería una baraja apta para un torneo, pero muy probablemente no sería una baraja ganadora.

Las barajas ganadoras de cada formato, que están bastante establecidas, parten de una idea: juntan cartas que comparten un criterio común y tienen buena sinergia entre ellas, o bien son las que mejor responden a llevar a cabo una estrategia determinada. Es posible que una misma baraja pueda tener distintas versiones (donde, de sus sesenta cartas, ocho o nueve sean diferentes) pero seguirán cumpliendo la misma estrategia y por tanto seguirá siendo la misma baraja.

Estas barajas evolucionan con la impresión de nuevas cartas: cuando aparece una carta que cumple un buen papel en una de estas barajas, y que es mejor que X carta que ya se estaba incluyendo, se empiezan a ver versiones con este cambio. Estos cambios son constantes, pero no modifican el conjunto de barajas ganadoras porque siguen respetándose las mismas estrategias.

Cada baraja se reconoce por un nombre de forma única, que puede hacer mención a la carta con la que gana, a la estrategia que sigue, o a una palabra clave de sus cartas.

El conjunto de todas las barajas que se juegan en un formato, ganadoras o no, se llama metajuego. El metajuego de un formato suele venir dado por lo que en programación denominamos diccionario o mapa: las claves son los nombres de las barajas y los valores son el porcentaje de representación en el formato.

Existen varias páginas (como https://www.mtggoldfish.com/ o https://mtgtop8.com/ ) que recogen qué barajas se han jugado más, de entre ellas, y sus calificaciones en torneos de nivel alto[2].

[2] Torneos de alto nivel: aquí me refiero a lo siguiente: los torneos de tiendas como Avalon Burgos o Itaca Madrid no requieren que los jugadores presenten un documento con la lista de cartas de su baraja, pero los torneos online sí requieren esto, y los Gran Prix, Pro Tour y mundial también.

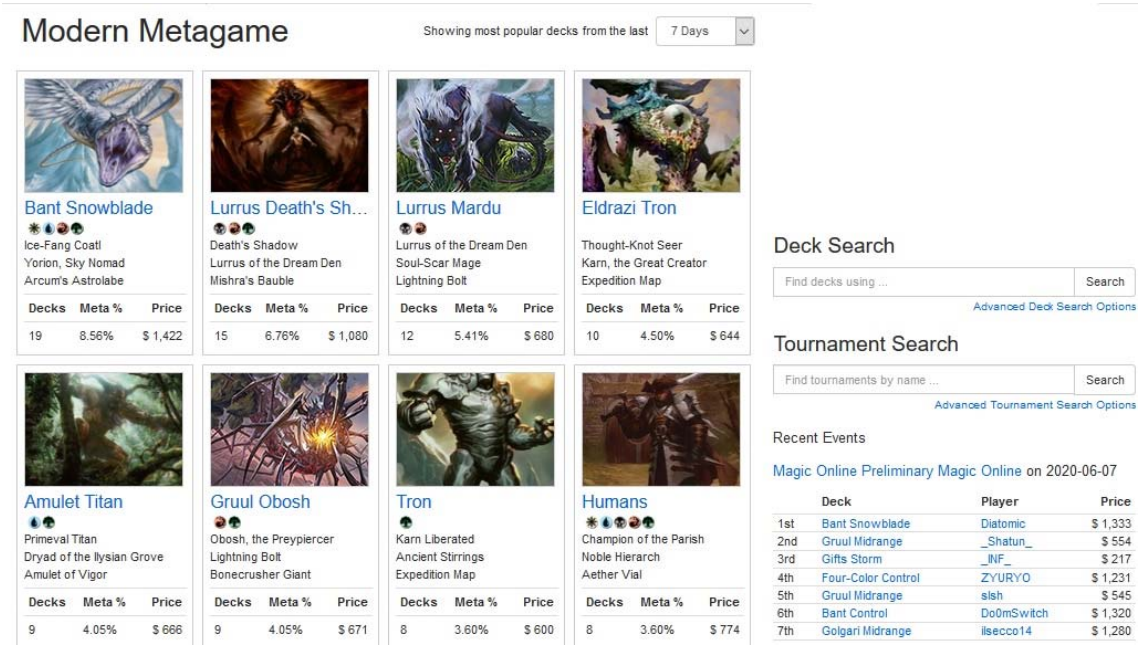


Figura 3: Entrada general de MtgGoldfish para el formato modern

La figura 3 muestra un pantallazo de www.MtgGoldfish.com. Aquí se ve el metajuego de un determinado formato (Modern), con las barajas más jugadas, donde se puede aplicar un filtro temporal (arriba a la derecha), se pueden buscar barajas por nombre concreto o torneos, y a la derecha más abajo, se pueden ver los torneos más recientes de los que se han introducido estadísticas.

En el cuadrante principal se ven las barajas por su nombre y algunos datos de interés para los jugadores.

Si entramos a una de las barajas (pinchando en su nombre, marcado en azul para indicarnos que es un vínculo) vemos algo así:

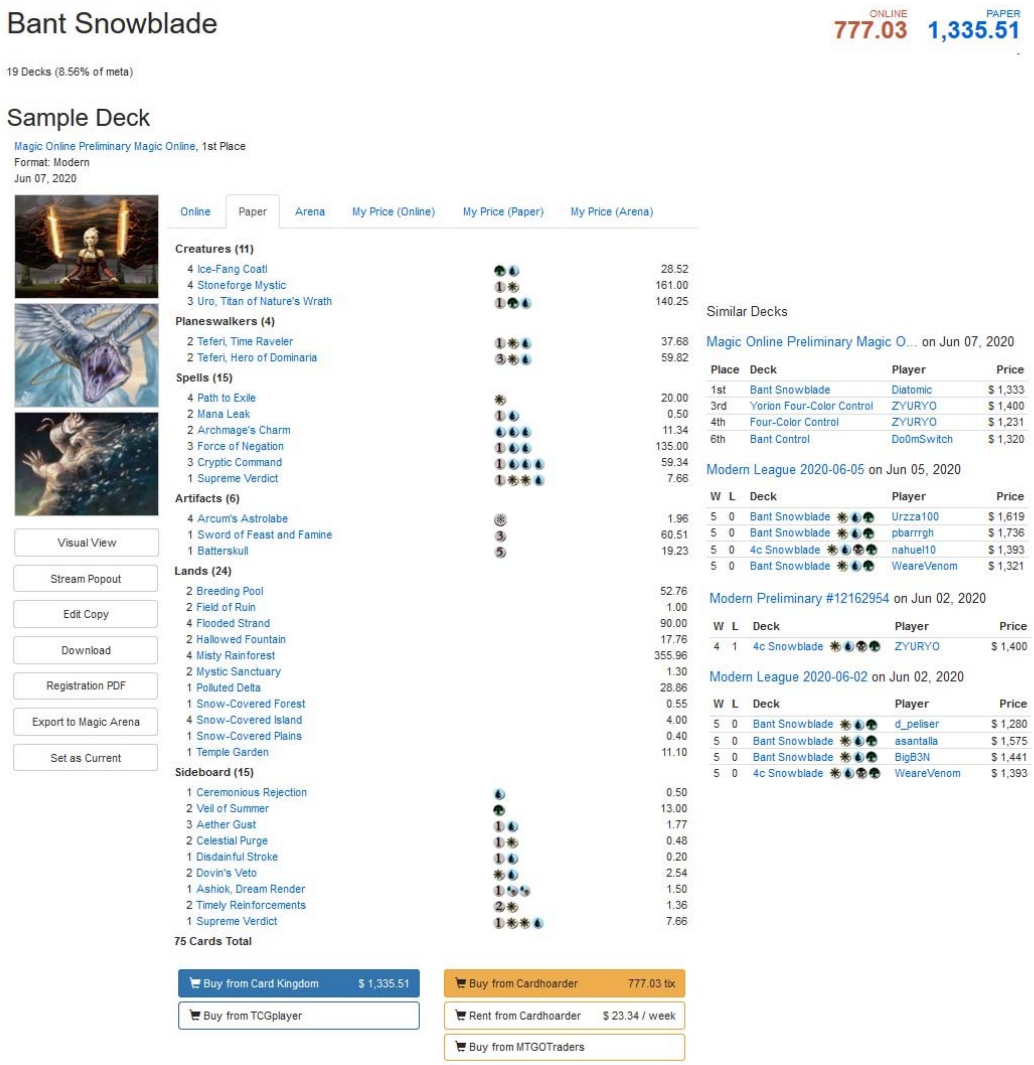



Figura 4: Desglose de una baraja en MtgGoldfish

Lo que vemos en el bloque principal de la página es la última lista (sesenta más quince) registrada en un torneo, con algunos datos relevantes para los jugadores (los símbolos de la segunda columna) y un precio aproximado de las cartas (tercera columna). A la derecha aparecen otras listas (sesenta más quince) de esta misma baraja. Si sobre esta misma página bajamos un poco más vemos lo siguiente:

## Creatures


4.0 in 94% of decks


3.6 in 94% of decks


2.5 in 50% of decks


4.0 in 35% of decks

## Planeswalkers


1.9 in 98% of decks


2.9 in 87% of decks


2.0 in 69% of decks

## Spells


4.0 in 100% of decks


3.3 in 100% of decks


2.9 in 100% of decks

Figura 5: Desglose de las cartas que llevan todas las listas de una baraja concreta

Vemos un desglose con el porcentaje de listas de esta misma baraja que juegan cada una de las cartas (como este documento está realizado entre mayo y junio del año 2020, debido a las restricciones de personal en reuniones sociales se ha reducido considerablemente el número de torneos y por eso aparecen datos un poco "extraños", como esos porcentajes decimales).

## 2.4- Enfrentamientos buenos y malos y ciclo de vida del metajuego:

En todos los formatos, entre las barajas más ganadoras, se cumple una ley de piedra-papel-tijera bastante rigurosa. Esto significa que, todas las barajas tienen ciertos emparejamientos buenos (es muy probable que los ganen), ciertos emparejamientos malos (es muy difícil que los ganen) y ciertos emparejamientos muy abiertos (dependen de quién juegue mejor, el factor suerte, etcétera).

Aunque no hay ningún estudio sobre esto (y es lo que pretende hacer este proyecto), todos los jugadores conocen, al menos por encima, esta tabla de "quién gana a quién".

Normalmente los jugadores, antes de los torneos, utilizan páginas como las mencionadas anteriormente, para ver qué barajas son más representativas del metajuego (cuales tienen un mayor porcentaje y por tanto, cuales es más probable que se encuentren en dicho torneo) y elegir barajas que, en ese momento no son tan populares y la gente no se espera, y que además tengan un buen emparejamiento contra las más jugadas. Esto provoca ciclos constantes en el metajuego, aunque siempre cuente con las mismas quince o veinte ganadoras, entre ellas nunca hay una invencible.

## 2.5- Glosario de términos

Tras toda esta información, esta subsección pretende resumir los conceptos que van a ser utilizados a lo largo del proyecto de forma habitual, siendo nomenclatura técnica entre los jugadores

experimentados, sirviendo como glosario o guía de referencia para ubicar en cualquier momento dichos términos:

-Jugador: cada uno de los participantes en un torneo de *Magic the Gathering*.

-Oponente: al estar analizando las cosas desde el punto de vista de un jugador, siempre se estará enfrentando a otro jugador, y este será su oponente.

-Baraja: conjunto de cartas, comúnmente sesenta de base y quince de banquillo, que responden a una estrategia concreta. Aunque de un torneo a otro, un jugador varíe tres o cuatro cartas, si la estrategia es la misma, la baraja sigue siendo la misma.

-Ronda: conjunto de partidas, al mejor de tres, entre dos jugadores, donde.

-Main: partidas que se juegan con las sesenta cartas de base.

-Side: partidas que se juegan tras alterar la baraja intercambiando cartas entre la base y el banquillo.

-Metajuego: conjunto de todas las barajas que se juegan en un formato.

# 3- Objetivos del proyecto

Los objetivos de este proyecto pueden dividirse en dos bloques: por un lado, los objetivos técnicos respecto a la usabilidad de la aplicación, y por otro, los objetivos académicos respecto a los conocimientos que se deben aplicar y los que se deberán obtener para poder aplicar también, en base a ciertas necesidades que vayan surgiendo durante su desarrollo.

## 3.1- Objetivos de usabilidad

Los objetivos de usabilidad del proyecto son aquellos que van a dar forma a la aplicación que se va a desarrollar. En base a todos los conceptos explicados en el apartado anterior, el objetivo es que los jugadores que utilicen la aplicación puedan ver los ratios de victorias de todas las barajas, para determinar cuáles cosechan más victorias por torneos o partidas jugadas, independientemente de su popularidad en cada momento.

Partiendo de este punto, los objetivos también incluyen conocer con exactitud los ratios de victoria de cada baraja contra cada otra (dentro de un único formato) y ayudar a cada usuario a analizar con qué baraja obtiene mejores resultados.

## 3.2- Objetivos académicos

Para llevar a cabo los objetivos de usabilidad, se desarrollará una aplicación web, proceso mediante el cual se alcanzarán otros objetivos como familiarizarse con ciertas herramientas o pulir el conocimiento que ya se tiene de ellas.

-Realizar el diseño de una base de datos, determinando qué tablas debe contener, así como los campos de las mismas y las relaciones entre ellas (Bases de datos, Aplicaciones de bases de datos y Análisis y diseño de sistemas).

-Realizar el diseño de los objetos y cómo se interrelacionan los mismos con la base de datos (Metodología de la programación y Análisis y diseño de sistemas).

-Determinar qué datos se le deben presentar al usuario, y cómo (Interacción hombre máquina).

-Adquirir conocimiento sobre cómo se debe desarrollar una aplicación web con la estructura cliente-servidor.

-Adquirir conocimiento sobre el desarrollo de interfaces gráficas y el tráfico de datos entre el servidor y la vista.

-Utilizar un repositorio de forma constante durante el desarrollo del proyecto.

# 4- Técnicas y herramientas utilizadas

En este apartado se comentan las herramientas y técnicas utilizadas, así como se justifica sus elecciones frente a otras posibilidades.

## 4.1- Herramientas

Como lenguaje principal para el desarrollo del backend se ha elegido JAVA (Qué es Java 2019), debido a la familiaridad con él por utilizarlo en varias asignaturas, así como por preferencia del alumno frente a otros (como Python (Qué es Python 2003), con una sintaxis distinta; o PHP (¿Qué es PHP? 2019), desconocido).

Como servidor interno se ha elegido Apache Tomcat (¿Qué es Apache? 2018). Sobre este aspecto no había experiencia y en todas las páginas, documentos y libros consultados sobre el desarrollo de páginas web con JAVA se recomendaba el uso de dicha herramienta, especialmente en el libro utilizado en las primeras fases del desarrollo (aplicaciones Web Java.pdf s. f.).

Como base de datos se ha utilizado MySQL (¿Qué Es MySQL? 2019). La otra gran opción era OracleXE (Oracle Database Express Edition | Oracle España 2020), ya que también se había utilizado en otra asignatura y hay ya una base de conocimientos respecto al uso de la misma. Pero para seguir los tutoriales del libro (aplicaciones Web Java.pdf 2017) se instaló esta herramienta y resultó tener un rendimiento en la máquina local mejor que OracleXE, principalmente por el uso de recursos: OracleXE se arranca con el sistema operativo, es necesario configurarlo para activar y desactivar los servicios manualmente, y, cuando están activados, el uso de otras aplicaciones (el navegador web, un procesador de textos y el entorno de desarrollo) se vuelve sumamente pesado y lento. En cambio, MySQL trabaja de una forma mucho más ligera y no es necesario desligar el inicio de procesos del arranque del sistema operativo.

Como interfaz gráfica de conexión con MySQL se ha utilizado MySQL Workbench (Workbench MySQL 2017), para probar las sentencias SQL antes de añadirlas en el código y para comprobar en tiempo real los cambios que generaba la ejecución de la aplicación en las tablas.

Como entorno de desarrollo se ha utilizado NetBeans (Welcome to Apache NetBeans 2020), de Apache, por una situación similar, se tomó como prueba para seguir los tutoriales del libro anteriormente citado y

su rendimiento resultó mejor que el de la primera gran opción: (¿Qué es eclipse? 2010).

Como framework de JAVA para el desarrollo web se eligió JavaServerFaces (JavaServer Faces Technology 2020). No hay una razón objetiva para haberlo elegido, tras consultar varias listas era la primera opción en casi todas ellas, y en el libro (aplicaciones Web Java.pdf 2017) también se utiliza.

-Como repositorio para alojar el proyecto en la nube se ha utilizado GitHub (¿Qué Es GitHub? 2019), con la extensión Zenhub (ZenHub - Agile Project Management for GitHub 2020) para definir las tareas y sprints no ligados a commits.

-VirtualBox (FM 2020) como herramienta para crear una máquina virtual donde probar la aplicación una vez exportada y comprobar su funcionamiento en otros entornos.

## 4.2- Técnicas

Hay dos técnicas utilizadas que se han elegido entre varias opciones:

JDBC (Java Database Connectivity 2020): frente a JPA (Java Persistence API - Wikipedia, la enciclopedia libre 2020), principalmente por la sintaxis de las operaciones con la base de datos: en JDBC se utilizan las mismas cadenas de texto que en SQL, y se podían probar directamente todas ellas con la herramienta MySQL Workbench.

ManagedBeans (Managed Beans in JavaServer Faces Technology - The Java EE 6 Tutorial s. f.) como clases de JAVA para la interrelación entre los datos manejados por la aplicación (backend) y los datos presentados al usuario e introducidos por él (frontend). La otra opción era usar clases heredadas de Servlet (Tyson 2018) y la razón para esta elección es que las clases Servlet se basan en respuestas y tienen una vida útil del paso de mensaje, mientras que las clases ManagedBean generan objetos cuya vida útil puede controlarse (puede ser de respuesta o de sesión), y en la mayoría de los casos, vamos a rellenar los atributos de estos ManagedBeans con datos que queremos que permanezcan durante toda la sesión.

Ejemplo práctico: un usuario accede a su perfil, ve sus datos y registra los datos de un torneo que ha jugado con una baraja. Al introducir estos datos a la base de datos hay varios caminos: guardar los “viejos” en variables, con selects, aplicar los resultados que ha

introducido y actualizar la base de datos, (con los datos viejos ya cargados en variables al habérselos mostrado solo habría que actualizarlos e introducirlos). Con un servlet, los datos mostrados previamente ya no estarían disponibles, por tanto, deberíamos obtenerlos previamente de nuevo, guardarlos en variables, actualizarlos con los nuevos y guardarlos en la base de datos. Un tercer camino sería el de hacer una única transacción que hiciera un update, pasándole como argumentos los datos nuevos, pero esto implicaría una conexión más larga, ya que habría que, igualmente, obtener los datos previos para actualizarlos con los nuevos). Nótese que al hablar de actualizar nos referimos a algo así: datos viejos: (1, 2, 5, 3) datos nuevos: (1, 0, 2, 0) datos finales actualizados: (1+1, 2+0, 5+2, 3+0).

# 5- Aspectos relevantes

Dentro de este apartado cabría destacar todas las decisiones que afectan al diseño del backend, almacenando datos que quizá parezcan redundantes, y, por otro lado, aquellos puntos que supusieron una dificultad durante el desarrollo por enfrentarse a determinadas situaciones por primera vez.

## 5.1- Decisiones sobre el diseño

Hay dos decisiones importantes que afectan al diseño almacenando datos que podrían ser prescindibles.

Almacenamiento de porcentajes: en la base de datos se almacenan muchos datos decimales que corresponden a porcentajes que son datos derivados de los números brutos (por ejemplo, calcular el porcentaje de partidas ganadas es un dato que deriva de las partidas ganadas y perdidas).

Respecto a los usuarios, la operación login busca la pareja nombre de usuario y contraseña en la base de datos y devuelve todos los datos del mismo. Podría aquí calcular los datos derivados del mismo en vez de almacenarlos en la base de datos, pero esto permite que se pueda generar una clasificación de jugadores en base a sus porcentajes (aunque esta funcionalidad no está aún implementada, es una idea de escalabilidad para un futuro).

Respecto a las barajas: Hay una vista que carga un cuadro donde se ve el porcentaje de victorias de cada baraja frente al resto. Si estos datos no se almacenan y se calculan al cargar esta vista, habría una carga de trabajo bastante intensa para el servidor.

Ejemplo:

| | Bant snowblade | Death's shadow | Lurrus mardu | Eldrazi tron |
|---|---|---|---|---|
| Bant snowblade | | 65% | 62% | 42% |
| Death's shadow | 35% | | 50% | 56% |
| Lurrus mardu | 38% | 50% | | 69% |
| Eldrazi tron | 58% | 44% | 31% | |

Figura 6: Ejemplo de una tabla de emparejamientos realizada manualmente

Para cargar una tabla como esta estamos haciendo varias selects en un bucle. Cada operación de consulta solo necesita un dato, por ejemplo, para el 65% de la segunda fila, la operación será "select porcentaje_total from cruces where baraja_1 = 'Bant snowblade" and baraja_2 = 'Death's shadow';"

Si en vez de esto, calculásemos ese dato dinámicamente, ese bloque de código sería así:

Rs = st.executeQuery("select * from cruces where baraja_1 = 'Bant snowblade" and baraja_2 = 'Death's shadow';"

Rs.next();
Float dato_mostrar = (main1 + side1)*100 / (main1 + side1 + main2 + side2);

En el primer caso la select solo trae un valor y después dato_mostrar es igual a lo que ha devuelto esta select, no necesita este cálculo, por tanto, el tiempo de respuesta al usuario es menor.

Observando el cuadro, también, y en base a la lógica, sabemos que es una matriz casi transpuesta, ya que si el ratio de victoria de baraja A contra B es 45%, el ratio de B contra A será 100-45 = 55%, y que por tanto la mitad de los datos de esta tabla son dependientes.

¿Merece la pena almacenar solo la mitad de los cruces y calcular la otra mitad con esta operación?

A todos los niveles sí, porque igualmente en la select solo devolvería un dato y después la operación sería más sencilla (una resta) y solo se haría en la mitad de los datos a devolver. Pero no está implementado así.

Esta decisión está tomada en base a las tablas concretas para cada baraja. Partiendo de la vista de ese cuadro podemos acceder a una vista de los emparejamientos para cada una de las barajas:

Si almacenásemos solo la mitad de los cruces (la idea original era ordenar las barajas alfabéticamente para introducir solo A contra B), en todos estos casos habría que obtener primero todos los cruces de la baraja seleccionada, y después, para cada baraja de la tabla barajas que no esté contenida entre las que ya tenemos, obtener los cruces (baraja no contenida vs baraja seleccionada) y calcular los datos de forma inversa. Esto sí lleva una carga de trabajo bastante mayor que hacer una única select para cada fila de esta tabla, por tanto, en la tabla cruces aparecerán tanto los datos de A contra B como los de B contra A.

El desarrollo de la interfaz gráfica merece un punto a destacar. Tras realizar todo el desarrollo funcional inicialmente, se han encontrado ciertas pegas en cuanto a funcionalidades que el framework elegido no puede realizar, tales como la ejecución de código JavaScript (Free JavaScript training, resources and examples for the community s. f.).

En el caso de los mensajes de retroalimentación al usuario, la idea original era mostrar ventanas emergentes, o un texto en la misma pantalla, por ejemplo, cuando el login fuese incorrecto debido a una mala contraseña. Esto requeriría de una función en código JavaScript que permitiese, o bien lanzar mensajes alert, o bien cambiar la visibilidad de un panel con texto.

Tras un estudio de las posibilidades, frente a la opción de Bootstrap, hay otro framework de JAVA especialmente diseñado para suplir las carencias de JSF: JavaPrimeFaces (PrimeFaces Showcase s. f.). Este framework es muy sencillo de implementar (solo requiere añadir el JAR correspondiente a las librerías del proyecto), y mejora los elementos de JSF (JavaServerFaces).

Por contra, conlleva una modificación del código ya que muchos elementos de JavaServerFaces no son compatibles, o bien, los elementos de JavaPrimeFaces tienen etiquetas html diferentes, propiedades diferentes, y requiere realizar cambios considerables sobre el código para implementar una interfaz gráfica basada en dicho framework.

## 5.2- Nuevas situaciones

La primera situación novedosa se da a la hora de implementar la estructura cliente-servidor, ya que, en el asignatura Aplicaciones de bases de datos se ha trabajado conectando aplicaciones JAVA con una base de datos, pero en este caso la conexión se realizaba desde el servidor, en este caso Apache Tomcat.

La línea de código para conectarse es la misma (bien por URL mediante driver, o bien mediante un fichero de contexto (JNDI)) pero hacía falta una biblioteca JAR que conectase la base de datos con este servidor.

Otro aspecto relevante, por su complicación, ha resultado ser el despliegue de la aplicación. Al tratarse de una aplicación web se ha intentado desplegar en varios servicios de hosting gratuitos, sin tener éxito en el proceso, porque en el momento de meter una base de datos dejaban de ser gratuitos, como Heroku (Cloud Application Platform | Heroku s. f.) o RedHat (Red Hat - Desarrollamos tecnologías de open source para su empresa s. f.).

Merece una mención especial el servicio *Firebase* (Firebase s. f.) ,de *Google*, que añade ciertas modificaciones al fichero de la aplicación web compilada (fichero con extensión .war), para poder subirlo a su almacenamiento on-line.

Tras seguir la documentación oficial (Documentación | Firebase s. f.) y ampliarla con algún videotutorial (03. Programación avanzada. Creación de proyecto en NetBeans (Java y Firebase) - YouTube s. f.) resulta estar modificando parámetros en un fichero que mi aplicación web no tiene, basado en otra decisión de diseño: no es un proyecto Maven (jgarzas 2014).

El proyecto Maven se puede generar, en NetBeans como nuevo proyecto, y en Eclipse desde un proyecto de aplicación web JAVA. La única diferencia entre ambos proyectos es la existencia de un fichero "pom.xml" llamado fichero de dependencias, ya que, a la hora de buscar las librerías de JAVA (ficheros .jar) a importar en el proyecto, se puede buscar en su lugar la dependencia Maven de esa misma librería (ver ilustración) y añadirla (copiando el texto) al fichero pom. Después, al compilar el proyecto, el entorno de desarrollo descarga los ficheros .jar necesarios automáticamente, en base a las dependencias que aparezcan en dicho fichero.


Figura 7: Ejemplo de dependencia Maven y enlace de descarga de fichero .jar

Después de varias pruebas sin éxito con estos sistemas de hosting, se cambia el rumbo de despliegue a una máquina virtual, limitándolo a una ejecución en local, como una versión beta. Se crea la máquina virtual alojada en el repositorio (), y se empiezan a instalar los servicios necesarios.

Para que la aplicación, una vez compilada y construido el fichero .war funcione, es necesario disponer de: Apache Tomcat como servidor, MySQL como sistema gestor de la base de datos, JAVA, que ya es requisito de Tomcat y, finalmente, las librerías de C++. (Todas estas instalaciones vienen explicadas en el anexo D, manual del programador, y resulta redundante explicarlo aquí.)

Una vez se han instalado todos los servicios y las rutas son correctas, falta añadir un detalle: la ruta del fichero con los parámetros para la conexión a la base de datos, que se pudo hacer funcionar, finalmente, con una ruta absoluta, y se colocó en un directorio de fácil acceso (C:\JAVA), aunque es variable siempre que se corrija la ruta correctamente en el código.

# 6- Trabajos relacionados

La motivación inicial de este trabajo era "determinar cuáles son los emparejamientos buenos y malos para cada baraja", aunque más adelante también se menciona que "todo jugador ligeramente experimentado conoce los tres o cuatro mejores y peores emparejamientos de casi todas las barajas".

Existen MUCHOS documentos de análisis de barajas (artículos, videos, blogs…), justificando por qué son mejores unas combinaciones de cartas que otras, muchas guías sobre cómo realizar los cambios entre cartas de la base y del banquillo en función de la baraja que juega el oponente, y todas estas guías hacen menciones a los emparejamientos buenos, malos y abiertos, como podemos ver en la figura, extraída de un artículo del grupo ShowAndTellMTG (Ojeda 2019):

| Deck | Metagame Share | Pairing | % WR |
|---|---|---|---|
| Dredge | 4,72% | Very unfavourable | 30% |
| Hardened Scales | 3,05% | Very unfavourable | 30% |
| Bant Spirits | 2,77% | Very unfavourable | 30% |
| KCI | 5,13% | Quite unfavourable | 35% |
| Mardu Pyromancer | 1,11% | Quite unfavourable | 35% |
| UB Faeries | 0,69% | Quite unfavourable | 35% |
| Hollow One | 2,77% | Unfavourable | 40% |
| Living End | 1,39% | Unfavourable | 40% |
| Whir Prison | 1,25% | Unfavourable | 40% |
| Blue Moon | 1,11% | Unfavourable | 40% |
| Mono Red Phoenix | 1,11% | Unfavourable | 40% |
| Goryo's Reanimator | 0,69% | Unfavourable | 40% |
| Jeskai Control | 1,53% | Slightly unfavourable | 45% |
| UW Control | 2,64% | Even | 50% |
| BG Rock | 2,50% | Even | 50% |
| Jund | 2,50% | Even | 50% |
| Mono G Tron | 3,74% | Slightly favourable | 55% |
| Titanshift | 1,25% | Slightly favourable | 55% |
| Affinity | 0,69% | Slightly favourable | 55% |
| UR Phoenix | 7,91% | Favourable | 60% |
| Humans | 4,02% | Favourable | 60% |
| Amulet Titan | 3,33% | Favourable | 60% |
| GW Bogles | 1,25% | Favourable | 60% |
| UB Mill | 1,11% | Favourable | 60% |
| Bridgevine | 0,83% | Favourable | 60% |
| Eldrazi and Taxes | 0,69% | Favourable | 60% |
| BG Elves | 0,42% | Favourable | 60% |
| Grixis Death's Shadow | 5,27% | Quite favourable | 65% |
| UR Storm | 2,77% | Quite favourable | 65% |
| Counters Company | 0,69% | Quite favourable | 65% |
| Bushwhacker Goblins | 0,55% | Quite favourable | 65% |
| Burn | 4,16% | Very favourable | 70% |
| Traverse Shadow | 2,08% | Very favourable | 70% |
| Infect | 1,25% | Very favourable | 70% |
| Ad Nauseam | 1,11% | Very favourable | 70% |

Figura 8: Ejemplo tabla de emparejamientos

En este ejemplo vemos una tabla sacada de un artículo que analiza una baraja concreta y los emparejamientos de la misma. El problema de esta tabla es que es estática, y está actualiza en el momento de su publicación (enero de 2019). (Se ve cómo identifica las barajas únicamente por su nombre, y todo jugador experimentado sabe identificarlas.)

Con el cambio constante del metajuego, en unos meses esto varía radicalmente, y tomar esta tabla como ejemplo año y medio después de su publicación, solo sirve como orientación, ya que es posible que algunas barajas hayan desaparecido, que haya otras nuevas que aquí no están contempladas y que las que hay hayan sufrido cambios, haciéndolas más, o menos, resistentes a la baraja para la que se analizan.

El presente proyecto pretende mejorar este sistema y realizar este tipo de tablas de forma dinámica, almacenando y procesando una cantidad masiva de enfrentamientos, y permitiendo que los porcentajes de la tabla vayan variando con el paso del tiempo.

Ya se han mencionado otras herramientas que recogen estadísticas sobre las barajas, enfocado a cuánto se juega una determinada baraja y qué configuraciones distintas de esa misma baraja se juegan (https://www.mtggoldfish.com/metagame/modern#paper).

# 7- Conclusiones y líneas de trabajo futuras

## 7.1- Conclusiones

A nivel personal y de trabajo, en la parte positiva, se puede hablar de haber desarrollado una aplicación empezando de cero; partiendo de una definición de casos de uso y requisitos funcionales y no funcionales, se han modelado los objetos, se ha diseñado un sistema entidad-relación para desarrollar un esquema para la base de datos y se ha programado todo el código; de la parte lógica, la cohesión con la base de datos, y la interfaz gráfica de usuario (con su correspondiente cohesión de datos para la interacción).

Además, se han empleado tecnologías no utilizadas durante el grado (tales como la inclusión de un sistema servidor para la parte del controlador de la aplicación) y otras similares a las aprendidas (como MySQL).

El trabajo responde a la búsqueda de una herramienta que ayude una actividad de ocio, habiendo mostrado una capacidad de aplicar los conocimientos adquiridos a un ámbito conocido.

Como contrapuntos, inicialmente se escogió JAVA como lenguaje para trabajar debido al amplio conocimiento del mismo y, aun cerca de haber terminado el trabajo, usando otro lenguaje (como Pyhton o PHP), con un conocimiento parcial, o nulo, de los mismos hubiese supuesto una dificultad añadida.

Durante el avance del proyecto y la búsqueda de contenido se han encontrado muchas respuestas a preguntas en otros lenguajes de programación, viendo que el uso de PHP para realizar aplicaciones web es mucho más común que JAVA, y queda la duda de si este lenguaje hubiese permitido realizar otras funciones.

Enfocándolo más a la parte visual, la interfaz de usuario queda bastante limitada al usar JavaServerFaces. Aunque se ha visto que es posible la integración con Bootstrap y, como se menciona en el capítulo “decisiones de diseño”, se dedicó un tiempo al estudio de JavaPrimeFaces, pero no se llegó a implementar por el requerimiento temporal para modificar todo el código. Quizá si se hubiese elegido antes este framework y se hubiese empezado a diseñar la interfaz gráfica con él tendría un aspecto más amigable.

Se ha diseñado una aplicación web y se ha desarrollado una versión beta, que solo puede ejecutarse en modo local al no haber sido

desplegada en ningún servicio de hosting, y también aquí se ha visto un lastre por una decisión de diseño: hay mucha más documentación para aplicaciones web que utilizan PHP o Python como lenguajes principales del backend que con JAVA, y además, la documentación para aplicaciones JAVA implica en la mayoría de los casos proyectos Maven.

Por tanto, a nivel personal, queda sin responder la pregunta: ¿qué hubiera pasado si se llega a elegir PHP como lenguaje principal? ¿El tiempo invertido en aprender dicho lenguaje se hubiese visto compensado a la hora de encontrar documentación más fácilmente?

La conclusión final es que el uso de JAVA ha facilitado el desarrollo de la parte funcional, ya que la aplicación cumple con la gran mayoría de requisitos funcionales descritos inicialmente, pero en los apartados de despliegue y de interfaz gráfica no se han podido mejorar o implementar tanto como hubiese sido preciso.

## 7.2- Líneas de trabajo futuras

Las líneas de trabajo futuras pueden dividirse en dos tendencias: las que buscan mejorar el aspecto gráfico y visual de la aplicación y las que buscan mejorar la funcionalidad de la aplicación.

Respecto a la mejora del aspecto gráfico y visual: prima el estudio e implementación del framework JavaPrimeFaces.

Respecto a una implementación completa: habría que estudiar más servicios de hosting para un despliegue on-line.

Respecto a las mejoras de funcionalidad: habría que implementar la gestión de correo electrónicos, añadir algún sistema de rankings de usuarios, para que cada jugador vea su puesto en una clasificación general, y/o poder ver una comparativa de sus ratios de victorias en comparación con la media de todos los jugadores inscritos.

Al haber hablado de distintos formatos, también podría ampliarse la aplicación para guardar información de uno o dos formatos más. Esto requeriría cambios en la base de datos, quizá la ampliación de la tabla bajaras añadiendo un nuevo campo, y una modificación de casi todas las cláusulas “where” en las consultas que extraen barajas de la base de datos.

# 8- Bibliografía


«1.6. Cómo incluir CSS en un documento XHTML (Introducción a CSS)». https://uniwebsidad.com/libros/css/capitulo-1/como-incluir-css-en-un-documento-xhtml (10 de abril de 2020).

«03. Programación avanzada. Creación de proyecto en NetBeans (Java y Firebase) - YouTube». https://www.youtube.com/watch?v=hV6mBtD-Knw (25 de junio de 2020).

«aplicaciones Web Java.pdf».

«BasicDataSource: Pool de conexiones». http://www.chuidiang.org/java/mysql/BasicDataSource-Pool-Conexiones.php (3 de junio de 2020).

Caules, Cecilio Álvarez. 2013. «Maven y Dependencias (I)». *Arquitectura Java*. https://www.arquitecturajava.com/maven-y-dependencias/ (9 de febrero de 2020).

Churchill, Alex, Stella Biderman, y Austin Herrick. 2019. «Magic: The Gathering Is Turing Complete». *arXiv:1904.09828 [cs]*. http://arxiv.org/abs/1904.09828 (16 de julio de 2020).

«Cloud Application Platform | Heroku». https://www.heroku.com/ (17 de julio de 2020).

«¿Cómo leer el archivo de la ruta relativa en el proyecto de Java? java.io.File no puede encontrar la ruta especificada». https://stackoverrun.com/es/q/889212 (17 de julio de 2020).

«COVID-19». 2020. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=COVID-19&oldid=127767454 (17 de julio de 2020).

«Documentación | Firebase». https://firebase.google.com/docs?hl=es-419 (17 de julio de 2020).

«Firebase». https://firebase.google.com/ (17 de julio de 2020).

FM, Yúbal. 2020. «VirtualBox: qué es y cómo usarlo». *Xataka*. https://www.xataka.com/basics/virtualbox-que-como-usarlo-para-crear-maquina-virtual-windows-u-otro-sistema-operativo (19 de julio de 2020).

«Free JavaScript Training, Resources and Examples for the Community». https://www.javascript.com/ (17 de julio de 2020).

Galaxy, Geek's Guide to the. 2017. «Why 'Magic: The Gathering' Beats Poker or Chess Any Day». *Wired*. https://www.wired.com/2017/04/geeks-guide-magic-gathering/ (17 de julio de 2020).

«Grand Prix». 2020. *MTG Wiki*. https://mtg.gamepedia.com/Grand_Prix (17 de julio de 2020).

«Grand Prix (*Magic: The Gathering*)». 2020. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Grand_Prix_(Magic:_The_Gathering)&oldid=953057842 (17 de julio de 2020).

«How to load Properties file from a file system? - Java Properties Class Examples». https://www.java2novice.com/java-collections-and-util/properties/file-system/ (12 de junio de 2020).

«java — Cómo desplegar un archivo war en Tomcat 7». https://www.it-swarm.dev/es/java/como-desplegar-un-archivo-war-en-tomcat-7/971790324/ (24 de junio de 2020).

«Java Database Connectivity». 2020. *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Java_Database_Connectivity&oldid=125831027 (17 de julio de 2020).

«Java Persistence API - Wikipedia, la enciclopedia libre». https://es.wikipedia.org/wiki/Java_Persistence_API (17 de julio de 2020).

«Java PreparedStatement - Javatpoint». *www.javatpoint.com*. https://www.javatpoint.com/PreparedStatement-interface (27 de febrero de 2020).

«JavaServer Faces Technology». https://www.oracle.com/java/technologies/javaserverfaces.html (17 de julio de 2020).

jgarzas. 2014. «Simple y rápido. Entiende qué es Maven en menos de 10 min.» *Javier Garzas*. https://www.javiergarzas.com/2014/06/maven-en-10-min.html (17 de julio de 2020).

Jimenez, Jhancarlos Marte. «DESPLIEGUE DE UNA APLICACIÓN SPRING BOOT SOBRE TOMCAT7 Y MYSQL5 EN LA PLATAFORMA CLOUD OPENSHIFT». : 22.

*JSF - Managed Beans*. 2014. https://www.youtube.com/watch?v=WvhhfPJ7r0w (20 de abril de 2020).

«JSF - Managed Beans - YouTube». https://www.youtube.com/watch?v=WvhhfPJ7r0w (21 de abril de 2020).

«JSF Tutorial #12 - Java Server Faces Tutorial (JSF 2.2) - JSF Forms and Managed Beans - YouTube». https://www.youtube.com/watch?v=0MR0ONn5J3c (5 de marzo de 2020).

«MAGIC: THE GATHERING». 2003. *MAGIC: THE GATHERING*. https://magic.wizards.com/es (17 de julio de 2020).

«Magic: The Gathering Pro Tour». 2020. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Magic:_The_Gathering_Pro_Tour&oldid=943081409 (17 de julio de 2020).

«Managed Beans in JavaServer Faces Technology - The Java EE 6 Tutorial». https://docs.oracle.com/javaee/6/tutorial/doc/bnaqm.html (17 de julio de 2020).

Ojeda, Iván. «The Rock Primer: descarte, Tarmo, Liliana: la Santa Trinidad (parte 2)». https://www.showandtellmtg.com/articulos/modern/980-bg-banquilleos-y-liga (17 de julio de 2020).

«Oracle Database Express Edition | Oracle España». 2020. https://www.oracle.com/es/database/technologies/appdev/xe.html (19 de julio de 2020).

Pérez, Enrique. 2020. «"Magic: The Gathering" es el juego más complejo del mundo, tanto que ni siquiera las máquinas saben cómo ganar». *Xataka*. https://www.xataka.com/literatura-comics-y-juegos/magic-the-gathering-juego-complejo-mundo-que-siquiera-maquinas-saben-como-ganar-1 (16 de julio de 2020).

«PrimeFaces Showcase». https://primefaces.org/showcase/ (17 de julio de 2020).

«¿Qué es Apache? Descripción completa del servidor web Apache». 2018. *Tutoriales Hostinger*. https://www.hostinger.mx/tutoriales/que-es-apache/ (19 de julio de 2020).

«¿Qué es eclipse? - PicandoJava.com». 2010. https://sites.google.com/site/picandojavacom/mundo-java/ide-eclipse/%C2%BFqueeseclipseinstalacion (19 de julio de 2020).

«¿Qué Es GitHub?» 2019. *Tutoriales Hostinger*. https://www.hostinger.es/tutoriales/que-es-github/ (19 de julio de 2020).

«Qué es Java». 2019. https://desarrolloweb.com/articulos/497.php (19 de julio de 2020).

«¿Qué es Magic?» 2015. *MAGIC: THE GATHERING*. https://magic.wizards.com/es/articles/archive/level-one/%C2%BFqu%C3%A9-es-magic-2015-07-21 (17 de julio de 2020).

«¿Qué Es MySQL?» 2019. *Tutoriales Hostinger*. https://www.hostinger.es/tutoriales/que-es-mysql/ (19 de julio de 2020).

«¿Qué es PHP? » Su Definición y Significado [2020]». 2019. *Concepto de - Definición de*. //conceptodefinicion.de/php/ (19 de julio de 2020).

«Qué es Python». 2003. https://desarrolloweb.com/articulos/1325.php (19 de julio de 2020).

«Red Hat - Desarrollamos tecnologías de open source para su empresa». https://www.redhat.com/es (17 de julio de 2020).

«Referencia CSS». *Documentación web de MDN*. https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS (10 de abril de 2020).

«SQL FOREIGN KEY Constraint». https://www.w3schools.com/sql/sql_foreignkey.asp (20 de marzo de 2020).

«SQL UPDATE Statement». https://www.w3schools.com/SQL/sql_update.asp (25 de marzo de 2020).

«SQL WHERE Clause». https://www.w3schools.com/SQl/sql_where.asp (25 de marzo de 2020).

Tyson, Matthew. 2018. «What Are Java Servlets? Request Handling for Java Web Applications». *InfoWorld*. https://www.infoworld.com/article/3313114/what-is-a-java-servlet-request-handling-for-java-web-applications.html (17 de julio de 2020).

«Welcome to Apache NetBeans». 2020. https://netbeans.apache.org/ (19 de julio de 2020).

«Workbench MySQL: si no lo conoces estás perdiendo un gran aliado.» 2017. *Pandora FMS - The Monitoring Blog*. https://pandorafms.com/blog/es/workbench-mysql/ (19 de julio de 2020).

«ZenHub - Agile Project Management for GitHub». https://www.zenhub.com/ (17 de julio de 2020).