



Escuela  
Politécnica  
Superior

# Guía para el desarrollo de TFG/TFM



Grado en Ingeniería Informática

## Trabajo Fin de Grado

Autor:  
Álvaro Navarro López-Menchero

Tutor/es:  
Domingo Gallardo López

Mayo 2020



Universitat d'Alacant  
Universidad de Alicante



## **Guía para el Desarrollo de los Trabajos Fin de Grado y Fin de Master.**

Este documento ha sido confeccionado por José Vicente Berná Martínez con el fin de facilitar a los alumnos de grado y máster el desarrollo de sus trabajos finales. Contiene además de los estilos, una descripción de los posibles apartados que pueden ser incluidos en el trabajo. Los alumnos pueden utilizarlo como plantilla, sustituyendo la explicación de cada apartado por sus propios contenidos, conservando formatos y estilos. El alumno puede personalizar el documento.

### **Versión del documento**

2018.06.01

### **Licencia**

Se permite la reproducción, distribución y comunicación pública de la obra, incluso con fines comerciales siempre y cuando reconozca y cite la obra de la forma especificada por el autor o el licenciante.



# Resumen

Este proyecto ha sido bautizado como “ChessCloud”, en el propio nombre podemos ver la idea de que todo lo que hagamos esté en la nube. El objetivo no es lucrativo, por lo que no habrá servicios premium de distinta índole, sino que será totalmente gratuito con la finalidad de que los usuarios formen parte de la dualidad ayudo-soy ayudado.

La tecnología utilizada ha sido Laravel con MVC (Modelo-Vista-Controlador) para poder gestionar de manera eficiente todas las partes de la web.

Podríamos hablar del proyecto como un taburete basado en tres sólidas patas:

1. Poder ver partidas subidas por los usuarios para beneficio propio.
2. Poder subir las nuestras para ayudar al resto de usuarios.
3. Poder practicar con una inteligencia artificial para comprobar nuestro progreso.

El proyecto ha sido elaborado durante todo el curso 2019-20 bajo el tutelaje de Domingo Gallardo López, hemos utilizado técnicas de SCRUM y Kanban y él revisaba los commits que iba realizando para controlar que todo fuera por el camino adecuado, incluso llegando a dirigirse a mí si lo veía oportuno para informarme acerca de lo que consideraba importante, tras la situación de emergencia nos empezamos a reunir con la herramienta meet mientras que antes las reuniones eran presenciales.

El concepto aquí trabajado puede no impresionar en cuanto a tecnologías usadas o dominio de éstas, pero el concepto propuesto es completamente innovador y, de tener éxito, podría ser expandido para romper con el sistema actual de servicios premium.

Podemos observar cómo servicios como YouTube al principio eran gratuitos y sin publicidad, más tarde introdujeron publicidad y recientemente para eliminar dichos anuncios es necesario abonar una cuota de pago. Toda esta tela de araña aparentemente indesenredable puede ser destruida con plataformas del futuro como la que aquí propongo para el ajedrez, mediante la colaboración de los usuarios y de quienes elaboramos estas herramientas podríamos tumbar esos imperios para poder llegar a un punto en el que los usuarios podrían disfrutar sin necesidad de abonar pequeñas cuotas en distintos sitios.

En resumen, no espero haber reinventado la rueda, sé que no lo hice. Tampoco espero haber sido el mejor diseñador ni implementador, pero me gustaría transmitir con este trabajo la

posibilidad de cambiar las cosas y llegar a un nuevo tipo de red que todos puedan disfrutar más cómodamente gracias a la colaboración de los usuarios.

# Abstract

This project has been baptised “ChessCloud”, in its own name we can see the idea about all of our work will be in the cloud. The purpose is not to earn money then there aren't premium services or something similar. Instead, this platform will be completely free with the desire that the users could use this program to join in the idea about the duality of help-be-helped.

The technology that I chose for this project is Laravel with MVC (Model-Controller-View) to can admin each part of the web.

We could speak about the project as a stool with three solid legs:

1. To see games that some user has pushed.
2. To push our games to help other users.
3. To can train with the AI.

This project has been made in the curse 2019-20 with the help of Domingo Gallardo López, we have used SCRUM's and Kanban's techniques, and he inspected the commits I had done in order to review the quality of the project. until the point that when he considered we needed a conference then he wrote me. After this situation with the COVID-19 we have had the conference through the application Google Meet.

The concept presented here may not be impressive as of applied technologies or proficiency in using them, albeit the proposed concept is, in fact, an innovative concept and, in case of it being successful later on, it could be then expanded in order to break the rule of the current practice around the existence of premium services.

We can observe how services such as YouTube were, in the very beginning, costless, and ad-free. Later on, advertisements were introduced, and recently, in order to remove these advertisements, a monthly or yearly payment is required. All of these apparently indestructible systems can be wiped out with the help of futuristic platforms, such as the one proposed here, oriented to chess. Through the users' and the ones creating these tools' collaboration, we would be able to destroy these massive empires, as to reach the ability of the users to be able to make use of our platforms, without the necessity to pay small fees in different sites.

To wrap up, by no means I expect to have reinvented the wheel because I know I did not. Neither do I expect to be the best designer nor implementer, but I would like to share with this project

the possibility to change things over and reach a new kind of network which everyone can enjoy more comfortably, and thanks to the users' collaboration.

# Resum

Aquest projecte ha estat batejat com "ChessCloud", al propi nom podem veure la idea que tot el que fem és al núvol. L'objectiu no és lucratiu, de manera que no hi haurà serveis premium de diferents tipus, sinó que serà totalment gratuït per tal que els usuaris siguin part de la dualitat ajude-sóc ajudat.

La tecnologia emprada ha sigut Laravel amb MVC (Model-Vista-Controlador) per poder gestionar eficientment totes les parts del web.

Podríem parlar del projecte com un tamboret basat en tres cames sòlides:

1. poder veure partides pujades pels usuaris per al seu propi benefici.
2. poder pujar les nostres per ajudar a la resta d'usuaris.
3. poder practicar amb una intel·ligència artificial per comprovar el nostre progrés.

El projecte s'ha elaborat al llarg de tot el curs 2019-20 sota la tutela de Domingo Gallardo López, hem utilitzat tècniques d'SCRUM i Kanban i ell revisava els commits que anava realitzant per controlar que tot anava pel camí adequat, fins i tot arribant dirigir-se'm si ho veia oportú per informar-me sobre el que considerava important, després de la situació d'emergència vam començar a reunir-nos amb l'eina meet mentre que abans les reunions eren presencials.

El concepte aquí treballat pot no impressionar en termes de tecnologies usades o domini d'aquestes, però el concepte proposat és completament innovador i, si té èxit, podria expandir-se per trencar amb el sistema actual de serveis premium.

Podem veure com serveis com YouTube inicialment eren gratuïts i sense publicitat, més endavant van introduir publicitat i recentment per eliminar aquests anuncis és necessari abonar una quota de pagament. Tota aquesta teranyina aparentment indesenredable pot ser destruïda amb plataformes del futur com la que aquí proposem per als escacs, mitjançant la col·laboració dels usuaris i dels que fan aquestes eines podríem enderrocar aquests imperis per poder arribar a un punt on els usuaris puguin gaudir sense haver d'abonar petites quotes en diferents llocs.

En resum, no espere haver reinventat la roda, sé que no ho vaig fer. Tampoc espere haver sigut el millor dissenyador ni implementador, però m'agradaria transmetre amb aquest treball la possibilitat de canviar les coses i arribar a un nou tipus de xarxa que tothom pugui gaudir més còmodament gràcies a la col·laboració dels usuaris.



## Motivación, justificación y objetivo general

Hijo de ajedrecista comencé a disfrutar desde que prácticamente tengo consciencia de este llamémosle ¿deporte, juego, arte?

Toda mi vida he jugado, entrenado y competido en esta disciplina que tanto me apasiona y estoy seguro de que cuando tenga hijos e hijas ellos y ellas serán encantados por su esencia.

Hoy en día soy presidente del Club Escacs Raspeig desde el 2016, club referencia a nivel provincial y autonómico. Además, soy entrenador y oriento a distintos alumnos al ambiente competitivo.

Todo esto y mucho más es lo que me motivó para contactar con Domingo Gallardo López y proponerle este proyecto que a él le pareció apropiado, una herramienta para poder ver herramientas en la web y aprender, algo que aunque ahora parece muy común cuando todos éramos algo menos jóvenes no existía, cuando éramos algo más niños.

Si alguien me pregunta la justificación de este proyecto y no otro le diré que no necesito un motivo para hacer lo que pienso es apropiado, pues no arrepentirme de nada es una parte importante de lo que soy, de lo que quiero ser. Un brainstorming junto a mi tutor y su aprobación es todo lo que necesito para dar rienda suelta a mi creatividad.

En cuanto al objetivo, diría que es un arco que lanza dos flechas. En primer lugar, todos y todas nos metemos en ingeniería informática siendo unos/as novatos/as y vamos aprendiendo día a día, crédito a crédito, pero ¿cuándo finaliza ese aprendizaje? La respuesta es sencilla: Nunca. Sin embargo, como bien le dije a mi tutor no me hubiera gustado acabar esta fase de mi vida sin ver en el espejo a alguien capaz de hacer un proyecto completo independientemente de si fue éxito mayor o menor, ¿pues qué es el éxito sino el proceso de superación y ver a tu “yo del ayer” un poco más lejos? Pues es paso a paso como uno se forja y ser capaz de hacer proyectos es un aspecto fundamental en estos días para poder tener más oportunidades de salidas laborales, pues como diré más adelante, uno no debería soñar su vida sino vivir su sueño.

Por otro lado, ¿qué mejor forma que el ajedrez para hacer algo que me satisfaga? ¿Acaso estudiar un itinerario distinto o tener un puesto de trabajo de otro ámbito te impide poder programar?

Por último, la idea conceptual innovadora que se va a exponer aquí ha sido de mi agrado por lo que podría suponer a nivel social, Free Collaborative Server (FCS) podría suponer un cambio en como los usuarios viven los servicios actuales.

En conclusión, una vida entera de ajedrez y una fase de estudio que espero haber alcanzado los requisitos para poder dar el siguiente paso.

## Agradecimientos

Lo bonito de esta parte es pensar en que tardaría menos en decir “desagradecimientos”.

En mi vida hay mucha gente importante para mí, mis padres, mi hermano y como no decirlo, mi abuela qué deseo esté descansando como se merece.

Por supuesto, también agradecer a mi tutor su ayuda en el proyecto.

Y qué decir de ellos, de todos y cada uno de mis amigos. ¿Qué sería de mi sin mi apreciado Pablo alias “WireHack” o sin Axel y sus consejos, que podría haber hecho sin conocer a Carlos y Óscar o sin Javi, Alberto, Susana o César? ¿Y qué sin mi club y esos niños que cuando te preguntas si el trabajo merece la pena te dicen que te echaron de menos? ¿Y Alejandro? ¿Y Llopis? ¿Mi querido Sergio y Rubén el maestro de la poesía y la kizomba (casi tanto como yo 😊)? Sois tantos que me siento como si me entregaran un Óscar sin tiempo para todos, pero sabéis quienes sois y que siempre estaréis conmigo.

Y aunque ella ya no me vea también se lo dedico, a esa persona que transformó a Álvaro en este otro más fuerte y qué ahora se atreve a cosas que nunca imaginó, este Álvaro ambicioso que sueña con tantas cosas que le cuesta saber si está despierto, este Álvaro que aún desde lejos espera sea la más feliz del mundo pues las estrellas no piden permiso para brillar y como tales tu luz me llegará incluso si pasan años sin saber de ti.

## Citas

Aquí hay tantas ideas que quiero mencionar que me costará seleccionar unas pocas.

*Una vez que cuestionas tus propias creencias, estás acabado.*

*Naruto Uzumaki*

*Quiero hacer contigo  
lo que la primavera hacer con los cerezos.*

*Pablo Neruda*

*Recuerda que no obtener lo que uno quiere,  
a veces es un golpe de suerte maravilloso.*

*Su Santidad Dalai Lama*

*Lucha por tus sueños o los demás te impondrán los suyos.*

*Paulo Coelho*

*Be water my friend.*

*Bruce Lee*

*Cuando te gusta una flor,  
sólo la arrancas.  
Pero cuando amas a una flor,  
la cuidas y la riegas a diario.  
Quien entiende esto entiende la vida.*

*Buda Gautama*

*El ajedrez es principal y enfáticamente un juego de filósofos.*

*Paul Murphy*

*Llora cuanto quieras, llena mares si te place,  
pero recuerda que por muy dura que sea la tormenta,  
siempre sale el sol.*

*Acuérdate de levantarte cuando todo acabe,  
acuérdate de reír y disfrutar hasta en tu peor momento.*

*Porque amigo, sonreír con un corazón triste,  
eso es de haber entendido todo.*

*Álvaro Navarro López-Menchero*

# Índice de contenidos

Resumen.....	3
Abstract .....	5
Resum.....	7
Motivación, justificación y objetivo general .....	8
Agradecimientos .....	9
Citas.....	10
Índice de Ilustraciones .....	15
Índice de tablas .....	16
1. Introducción .....	17
2. Objetivos .....	18
2.1. Descripción conceptual .....	18
2.2. Descripción técnica .....	19
3. Estado del arte. ....	20
3.1. Descripción.....	20
3.2. ¿Quién domina ahora el mercado?.....	21
3.2.1. ChessBase DataBase.....	21
3.2.2. Chess.com .....	21
3.2.3. Chess24 .....	22
3.2.4. Lichess .....	23
3.3. Solución ChessCloud .....	23
4. Metodología .....	24
5. Tecnologías.....	26
5.1. Chess.js.....	27
5.1.1. Inicialización .....	27
5.1.2. Funciones .....	27
5.2. Chessboard.js .....	28

5.2.1.	Inicialización .....	28
5.2.2.	Funciones .....	29
5.2.3.	Conclusiones.....	30
6.	Estudio de viabilidad .....	30
6.1.	Análisis DAFO .....	30
6.1.1.	DAFO.....	31
6.1.2.	Estrategias .....	33
6.2.	Lean Canvan .....	35
6.3.	Análisis de riesgos .....	37
7.	Análisis y especificación .....	39
7.1.	Casos de uso .....	39
7.1.1.	Sesión .....	39
7.1.2.	Guardar y ver partidas.....	40
7.2.	Pirámide de requisitos .....	40
7.2.1.	Sesión .....	41
7.2.2.	Partidas.....	42
8.	Diseño e implementación .....	44
8.1.	Diseño de la persistencia.....	44
8.2.	Diseño arquitectura tecnológica Front/Back-end .....	45
8.2.1.	Modelo Vista Controlador .....	45
8.2.2.	Back-end.....	47
8.2.3.	Front-end.....	48
8.3.	Diseño Interfaces.....	49
8.3.1.	Interfaz de usuario no registrado.....	49
8.3.2.	Interfaz de usuario registrado .....	50
8.3.3.	Interfaz de usuario administrador.....	51
8.4.	Implementación .....	52
9.	Funcionalidades.....	53

9.1.	Vistas .....	53
9.1.1.	Vistas de usuario no registrado.....	54
9.1.2.	Vistas de usuario registrado.....	56
9.1.3.	Vistas de administrador .....	60
10.	Pruebas y validación.....	61
10.1.	Pruebas.....	61
10.2.	Preparación del entorno .....	61
10.3.	Validación .....	62
11.	Conclusiones y trabajo futuro .....	62
	Referencias.....	64

## Índice de Ilustraciones

ILUSTRACIÓN 1: CB DATABASE .....	21
ILUSTRACIÓN 2: CHESS.COM .....	22
ILUSTRACIÓN 3: CHESS24 .....	22
ILUSTRACIÓN 4: LICHESS .....	23
ILUSTRACIÓN 5: LOGO CHESSCLOUD .....	23
ILUSTRACIÓN 6: DIAGRAMA DE FLUJO DE CADA ITERACIÓN .....	25
ILUSTRACIÓN 7: LOGO TRELLO .....	25
ILUSTRACIÓN 8: LOGO GITHUB .....	26
ILUSTRACIÓN 9: LOGO GOOGLE MEET .....	26
ILUSTRACIÓN 10: LOGO MAIL “GENÉRICO” .....	26
ILUSTRACIÓN 11: ESQUEMA DE UN ANÁLISIS DAFO .....	31
ILUSTRACIÓN 12: DAFO CHESSCLOUD .....	31
ILUSTRACIÓN 13: ESTRATEGIAS CHESSCLOUD .....	33
ILUSTRACIÓN 14: CUADRO PARA EL ANÁLISIS LEAN CANVAN .....	35
ILUSTRACIÓN 15: MODELO LEAN CANVAN .....	36
ILUSTRACIÓN 16: CASOS DE USO SESIÓN .....	40
ILUSTRACIÓN 17: CASOS DE USO GUARDAR Y VER PARTIDAS .....	40
ILUSTRACIÓN 18: ESQUEMA ENTIDAD RELACIÓN .....	45
ILUSTRACIÓN 19: EJEMPLO MVS .....	46
ILUSTRACIÓN 20: MOCKUP LISTA DE PARTIDAS CHESSCLOUD .....	49
ILUSTRACIÓN 21: VER PARTIDA CHESSCLOUD .....	50
ILUSTRACIÓN 22: GUARDAR PARTIDA CHESSCLOUD .....	51
ILUSTRACIÓN 23: MOCKUP VISTA ADMINISTRADOR CHESSCLOUD .....	52
ILUSTRACIÓN 24: EJEMPLO WEB.PHP .....	52
ILUSTRACIÓN 25: HOME CHESSCLOUD .....	54
ILUSTRACIÓN 26: ENTRAR CHESSCLOUD .....	54
ILUSTRACIÓN 27: REGISTRARSE CHESSCLOUD .....	55
ILUSTRACIÓN 28: LISTA DE PARTIDAS CHESSCLOUD .....	56
ILUSTRACIÓN 29: PERFIL CHESSCLOUD .....	57
ILUSTRACIÓN 30: PRACTICAR CHESSCLOUD .....	57
ILUSTRACIÓN 31: VER PARTIDA CHESSCLOUD .....	58



ILUSTRACIÓN 32: GUARDAR PARTIDA CHESSCLOUD .....	59
ILUSTRACIÓN 33: ADMINISTRACIÓN DE USUARIOS CHESSCLOUD.....	60
ILUSTRACIÓN 34: ADMINISTRACIÓN DE PARTIDAS CHESSCLOUD .....	60
ILUSTRACIÓN 35: RESULTADO DE LOS TESTS .....	62

## Índice de tablas

TABLA 1: REQUISITO INICIAR SESIÓN .....	41
TABLA 2: REQUISITO CERRAR SESIÓN .....	42
TABLA 3: REQUISITO REGISTRARSE .....	42
TABLA 4 REQUISITO GUARDAR PARTIDAS.....	43
TABLA 5: REQUISITO VER PARTIDAS.....	43

# 1. Introducción

Este trabajo está enfocado en crear una plataforma web para que cualquier usuario pueda acceder a partidas subidas por el administrador y/o por otros usuarios. De esta forma conseguiríamos un sistema de compartición de partidas al más puro estilo de programación “código abierto” como bien conocemos, con una colaboración de cada contribuyente para el beneficio de todos. Además, podremos jugar contra una inteligencia artificial para poder practicar donde especificaremos el nivel deseado y por último podremos usar dicha inteligencia para analizar las partidas mientras las observamos, con la opción a “navegar” entre las distintas opciones que tuvo la partida dada una posición.

Un proyecto que aborda varios de los problemas que desea un jugador de ajedrez cuando busca este estilo de plataformas, en especial cuando están comenzando. Un proyecto fácilmente escalable al que añadir nuevas funcionalidades. Y, sobre todo, un proyecto que busca que todos los usuarios hagamos posible la base de datos de partidas más grande que nunca hubo, sin necesidad de que nadie busque un interés en publicar sino el espíritu de ayuda de todos.

Todas las funciones de dicha plataforma son completamente gratuitas, ya que, como comento el objetivo no es en lo más mínimo lucrativo sino poder llevar a todos los usuarios la posibilidad de aprender y sentirse uno más de la comunidad para ayudar y ser ayudado en su camino dentro de este juego tan bonito.

En el aspecto de las asignaturas diría que la más importante ha sido Diseños de Sistemas Software, pues es ahí donde conocí la tecnología Laravel que he utilizado con el MVC (modelo-vista-controlador). Sin embargo, la destreza que actualmente poseo ha sido gracias todas y cada una de las asignaturas, aunque he de reconocer que si hubiera cursado otro itinerario sus asignaturas habrían sido más aplicables a la problemática de mi trabajo.

Un ejemplo de esto hubiera sido la asignatura de Metodologías ágiles de desarrollo impartida por mi tutor de TFG Domingo Gallardo López en el itinerario de sistemas software.

## 2. Objetivos

El objetivo principal del proyecto será diseñar una herramienta que permita a los usuarios compartir partidas para aprender y ayudar unos con otros y a otros, un primer paso en el potencial que tiene el concepto aquí trabajado.

A continuación, definiré las partes de manera más concreta y específica que nos ayudarán a conseguir el objetivo principal de manera más sencilla.

### 2.1. Descripción conceptual

A nivel “teórico” podríamos hablar de los siguientes puntos:

- Estudio del estado del arte: En primer lugar, necesitamos recopilar toda la información de nuestras experiencias de usuarios y, además deberemos informarnos más aún sobre el resto de las plataformas y analizarlas, nuestro objetivo aquí será plantear una plataforma distinta a nivel conceptual que rompa con las que tenemos ahora.
- Paso a paso: Con la clara idea de romper con la idea actual de servicios premium, en este caso aplicados al ajedrez, debemos “tener los pies en el suelo” y no aspirar a eso ahora, sino a crear una plataforma que vaya enfocada en esa línea y que cumpla los requisitos suficientes para ser ahora aprobado y pueda ser fácilmente escalable y ampliable para el futuro.
- 300 horas: Es el tiempo estimado para el proyecto, en ese tiempo debemos conseguir un resultado “visible” de lo que queremos plasmar, las funcionalidades básicas y el “escaparate” como “marca de la casa”, en esta ocasión esa marca será nuestra política de compartir las partidas que sube un usuario con todos los usuarios de la web.
- Cuanto sea necesario: A corto plazo este proyecto tiene una estimación como la que hemos comentado antes, sin embargo, de escalarlo podríamos llegar mucho más lejos. Tras el diseño y la implementación de la plataforma podríamos desarrollar hitos nuevos con funcionalidades objetivo en cada uno de ellos, como por ejemplo añadir una Arena de juego online en un mes de tiempo, etc. Mediante esta idea del trabajo podríamos ir expandiendo el proyecto constantemente.

## 2.2. Descripción técnica

Aquí entraremos a analizar los objetivos de un punto de vista más técnico:

- Alojaremos todos los datos en una base de datos relacional como, por ejemplo, MySQL
- Tendremos una vista con todas las partidas para poder elegir cual queremos reproducir, con una vista previa no reproducible de las jugadas de ésta y los jugadores que la disputaron.
- Un visor donde se muestre un tablero y las jugadas de la partida para poder reproducirla.
- La capacidad de insertar jugadas o bien desde 0 o bien a partir de una partida donde ya haya jugadas.
- Podremos guardar nuestras propias partidas desde dicho reproductor.
- En el caso de administradores podremos guardarlas mediante un string sin necesidad de reproducir la partida.
- La aplicación de filtros para la búsqueda de partidas con según qué datos (torneo, jugador).
- Un módulo de análisis a partir de ciertos conocimientos usando algoritmos de IA para poder dar una valoración de la posición.
- La posibilidad de jugar contra dicho módulo.

### 3. Estado del arte.

En este apartado intentaremos recoger diferente información sobre las distintas soluciones del mercado con funciones de nuestra misma índole, para así adquirir un mayor conocimiento y poder realizar este trabajo más objetivamente. Enfocaremos todo desde un punto de vista conceptual de lo que el usuario percibe.

#### 3.1. Descripción

Como jugador, entrenador y directivo de ajedrez tengo una perspectiva más que amplia de lo que el usuario busca y sobre lo que el usuario encuentra.

Mis primeras experiencias fueron con aplicaciones de escritorio como “Jaque Mate”, “ICC”, “Playchess”, etc. Plataformas ahora extintas. Sobre esa época fue cuando surgió “Buho21” completamente innovador al entrar desde un navegador y no dejar sólo acceder a una “Arena” sino además ofrecer otros servicios.

¿Qué paso con “Buho21”?

La aplicación de escritorio ChessBase ha sido siempre y continúa siendo la herramienta más profesional para el entrenamiento, todos los jugadores y entrenadores de un nivel relativamente alto la utilizamos regularmente, pues como era de esperar el pez grande pensó en comerse al chico.

ChessTempo surgió algo después con la idea de entrenar en la web, haciendo ejercicios. ¿A nadie se le pasó por la cabeza fusionar todo eso? ¿Nadie pensó en una plataforma web para jugar y entrenar?

Poco a poco han salido chess.com, chess24 y lichess quienes se encargan de aportar ejercicios a los usuarios para practicar y proporcionar una Arena de juego online, además, proporcionan videos y retransmisiones para que el usuario se vea siempre motivado a permanecer en la web. Por otro lado, tenemos chessdb que se encarga de justo lo que no hacen ellos, creado por ChessBase como antes comentaba proporciona la misma idea de análisis de partidas, pero esta vez online.

Tras esta pequeña introducción hablaremos de las más importantes actualmente.

## 3.2. ¿Quién domina ahora el mercado?

### 3.2.1. ChessBase DataBase

Seguramente a la que más se acabará pareciendo nuestra aplicación, su objetivo es el entrenamiento (no hay una “arena” para jugar con otros jugadores).



Ilustración 1: CB DataBase

Fuente: Propia

En primer lugar, tenemos en el centro el tablero donde podremos mover las piezas, también podemos retroceder y hacer nuevas ramas del mismo árbol (denominadas “variantes”), abajo del tablero podemos ver distintas variantes, recorrerlas, hacer anotaciones, etc.

A la izquierda vemos partidas importantes que coinciden con la posición, por coincidir entendemos que esa posición se dio exactamente así en un momento de la partida independientemente de cómo continuó.

En la parte superior derecha tenemos las jugadas que más se han jugado en esa posición, con cantidad de partidas, porcentaje de éxito, etc.

Debajo de lo anterior tenemos un análisis a través de determinada IA (módulos de análisis) de la posición, valorando quien está mejor y aportando cuál sería la mejor opción para el bando que juega en este instante, puede haber más de una IA.

### 3.2.2. Chess.com

Es la plataforma ajedrecística más utilizada, dispone de gran variedad de posibilidades, pero su éxito erradica en la comodidad y sencillez de su modo de juego *online*.



Ilustración 2: Chess.com  
Fuente: Propia

Un menú intuitivo, un chat que se puede desactivar, facilidad y rapidez para jugar partidas al ritmo elegido, y un tablero cómodo son las claves principales de su éxito.

### 3.2.3. Chess24

La plataforma que triunfa por “la sala de máquinas”, su modo de entrenamiento y de juego no es nada que destacar, pero una gran cantidad de videos por maestros y diarios producen una gran llamada a los *fans*, muchos aficionados disfrutaban de ver videos comentados de las mejores partidas o simplemente explicativos.



Ilustración 3: Chess24  
Fuente: Propia

Aquí arriba al principal reclamo en España, el Gran Maestro Pepe Cuenca, quien su manera de comentar ha llamado a mucha gente llegando incluso a acudir como [invitado en el popular programa de A3Media “Zapeando”](#).

### 3.2.4. Lichess

Esta plataforma es un término medio entre las dos anteriores, sin tantos fondos detrás como chess24 ni tantos usuarios como chess.com es una plataforma muy sencilla y divertida para competir con otros jugadores.



Ilustración 4: Lichess  
Fuente: Propia

El problema de todas (actuales como anteriores) es el mismo, ¿Qué partidas ves? ¿Hay servicio “completo” y gratuito?

### 3.3. Solución ChessCloud



Ilustración 5: Logo ChessCloud

Fuente: Propia



Como comentaba antes, en chessdb puedes acceder a determinadas partidas, ¿Cuáles? Las que decide chessdb.

ChessCloud pretende ser el barco que adelante por la izquierda a las plataformas como [chess.com](https://chess.com), [chess24](https://chess24.com) y [lichess](https://lichess.org) a la vez que por la derecha adelanta a [chessdb](https://chessdb.com).

¿Cómo?

La respuesta es sencilla, en lugar de buscar un beneficio económico como ellas planteamos un sistema de “código abierto” aplicado al ajedrez, ¿por qué ver sólo las partidas que alguien quiere? ¿por qué no colaborar con los usuarios? ¿por qué lo que yo subo no lo ve él y viceversa?

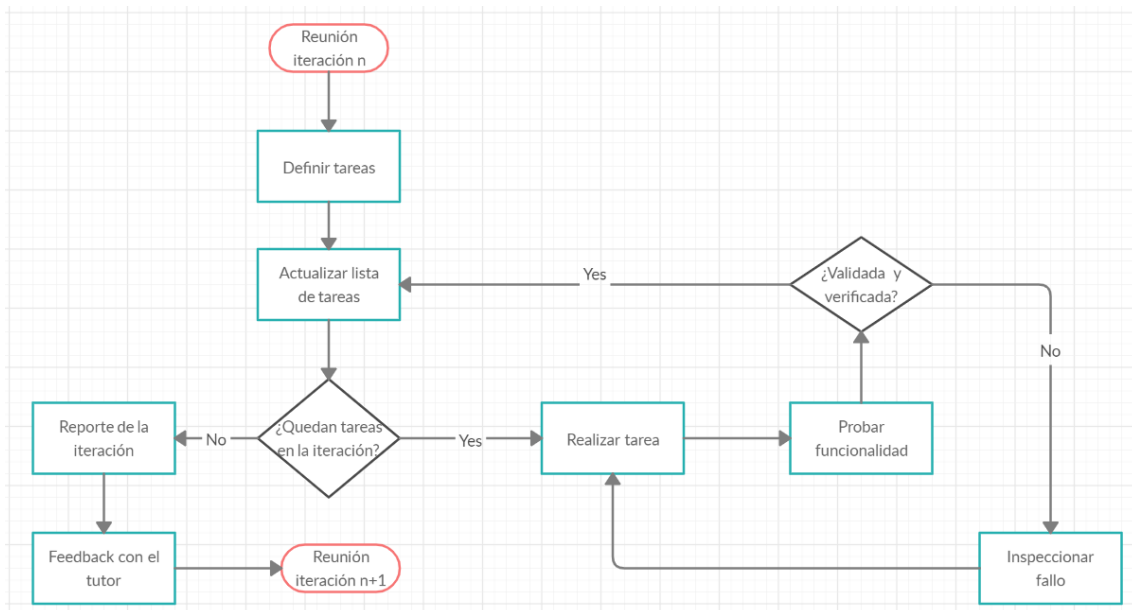
Planteamos el escenario de colaborar con la dualidad ayudo-soy ayudado, una vez esto sea posible todo podría crecer, podríamos subir vídeos gratuitos, podríamos implementar una Arena, podríamos hacer todo.

Frente a su muro de los servicios premium y su lucro tenemos nuestra catapulta de la colaboración de usuarios y de la contribución del desarrollo por parte de nosotros, los ingenieros informáticos.

Este modelo podría aplicarse a todo, este podría ser el principio de un concepto nuevo de plataformas que luche contra la política de pago de pequeñas cuotas que ahora domina los servicios de internet.

## 4. Metodología

En primer lugar, hemos trabajado con una metodología iterativa con algunas aplicaciones de SCRUM a modo de sprints, dividíamos los objetivos en tareas y éstas a su vez en subtareas que pudiéramos ir resolviendo y revisando. Los pasos se podrían definir con el siguiente diagrama de flujo:



*Ilustración 6: Diagrama de flujo de cada iteración*  
 Fuente: propia hecha en <https://app.creately.com/>

Como podemos observar las iteraciones seguían un esquema claro. Tras las reuniones Actualizaba mi lista de tareas y me ponía a trabajar en ellos con la obligación de comprobar su funcionamiento antes de pasar a la siguiente.

En cuanto a herramientas utilizadas en el desarrollo podría hablar de VSCode, HeidiSQL, Laragon, etc.

En cuanto a las herramientas para el SCRUM la elección no fue difícil, tanto mi tutor como yo estuvimos de acuerdo desde el primer momento, utilizando tarjetas visuales al estilo Kanban.

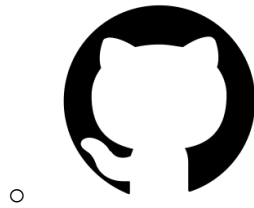
Las herramientas que utilizamos fueron:

- Trello: Aquí poníamos a modo iterativo las tareas de esa iteración. Las tarjetas las dividíamos principalmente en tres grupos:
  - Backlog: Sin comenzar.
  - Sprint: En curso
  - Done: Tarea finalizada.



*Ilustración 7: Logo Trello*

- GitHub: Como sistema de control de versiones (SCV) para ir subiendo los cambios y así poder revisarlos.



*Ilustración 8: Logo GitHub*

- Meet: Para las reuniones online.



*Ilustración 9: Logo Google Meet*

- Mail: Para tener feedback.



*Ilustración 10: Logo mail "genérico"*

## 5. Tecnologías

Aquí hablaremos de las tecnologías empleadas, en concreto de las dos librerías de Javascript, ambas han sido llamadas desde nuestras vistas para las distintas funciones, ahora pasamos a exponer cada una de ellas.

Antes de pasar a ello, es menester comentar la forma de la que ambas "se combinan", siendo Chess.js la parte interna y Chessboard.js la parte gráfica.

1. Chess.js realiza la función correspondiente.
2. Chessboard.js hace la acción ejecutada en el paso de uno de forma gráfica.

## 5.1. Chess.js

A chess.js podríamos referirnos como el “motor”, dicho de otra forma, será quién haga todas las funciones de la partida tales cómo mover las piezas, obtener una posición etc.

Según su github oficial que podremos ver en las [referencias](#) podemos definirlo de la siguiente manera:

“chess.js is a Javascript chess library that is used for chess move generation/validation, piece placement/movement, and check/checkmate/stalemate detection - basically everything but the AI.

chess.js has been extensively tested in node.js and most modern browsers.”

Dicho esto, ¿cómo hemos funciones hemos usado de esta librería?

### 5.1.1. Inicialización

Debemos declarar la variable en el código javascript

```
var chess = new Chess()
```

### 5.1.2. Funciones

Pondremos las funciones principales:

*Chess.move(“jugada”)*

Realizará la jugada especificada.

*Chess.load(“game\_fen”)*

Cargará el momento de la partida que se le pase por parámetro.

*Chess.turn()*

Nos dirá el turno de juego, “w” si es blancas y “b” si es negras.

*Chess.fen()*

Nos devolverá el fen de la partida, especificando no sólo la posición sino quién juega entre otros aspectos.

*Chess.pgn()*

Nos devolverá un pgn, donde ese pgn son los movimientos realizados en esa partida en formato de [notación algebraica](#) inglesa.

*Chess.game\_over()*

Nos devolverá un valor booleano indicando si la partida ha acabado o no.

*Chess.start()*

Nos configura la partida para que vuelva al “inicio”.

*Chess.undo()*

Deshace el último movimiento realizado.

*Chess.moves()*

Posibles movimientos en la posición actual.

*Chess.ascii()*

Esta es la única función aquí puesta no utilizada en el código, pero sí ha sido utilizada para depurar en varias ocasiones pues nos dice en una matriz la posición de cada pieza en el tablero, haciendo referencia a las piezas en su nomenclatura inglesa y usando letras mayúsculas para las piezas blancas y minúsculas para las negras y así diferenciarlas.

## 5.2. Chessboard.js

Aquí tendremos la parte gráfica, no es cuestión de mover la “partida” sino de qué se mueva gráficamente, es decir, que el usuario vea el movimiento de las piezas.

### 5.2.1. Inicialización

Debemos declarar la variable en el código javascript, pondremos un ejemplo básico.

```
Var board = Chessboard('myBoard', {  
    draggable: true,  
    moveSpeed: 'slow',  
    onChange: makeBestMove,  
    onDrop: onDrop,  
    position: 'start'  
})
```

Ahora analizaremos las partes principales:

- **draggable:** ¿Podremos mover las piezas o no? El valor nos dirá la respuesta. Puede parecer “poco útil” no dejar mover las piezas poniendo este valor a “false”, pero si sólo quisiéramos visualizar sin opción a análisis sería lo apropiado.
- **moveSpeed:** El efecto visual que observaremos al ver el movimiento de cada pieza.
- **onChange:** Es un “evento”, cuando la posición cambie se invocará a la función que se especifique aquí.
- **onDrop:** Es un “evento”, cuando la “soltemos” una pieza (se realiza una jugada) se invocará a la función que se especifique aquí.
- **position:** la posición de la que partimos.

Es interesante comentar una parte de la función “onDrop”:

```
/**/

var move = chess.move({

    from: source,

    to: target,

    /**/

});

/**/
```

En esta parte especificamos que el movimiento tendrá origen y destino, ¿para qué sirve esto? Para conseguir que las jugadas sean legales y no se produzcan movimientos incorrectos.

### 5.2.2. Funciones

Pondremos las funciones principales:

*Board.position(“fen”)*

Cambiará la posición del tablero a la del fen que le pasamos como parámetro.

*Board.move(“move”)*

Realizamos un movimiento.

*Board.fen()*

Nos devuelve el fen del tablero, importante tener claro que NO es el mismo fen del de partida, pues este no incluirá aspectos como quien juega, sino que definirá únicamente la posición de las piezas.

### 5.2.3. Conclusiones

Tras un duro trabajo de sincronización entre librerías es plausible (y recomendable) utilizarlas para ir realizando todas las funciones necesarias, hay funciones aquí no expuestas pues no se han usado en el proyecto, pero se ha usado una gran variedad de ellas para cumplir con los requisitos.

## 6. Estudio de viabilidad

En esta sección se puede hacer un estudio de viabilidad. Antes de arrancar con el resto del proyecto se puede analizar un poco si el proyecto en sí mismo, sus pretensiones u objetivos son viables, son pertinentes, son necesarios. También es muy adecuado indicar los riesgos y los planes de contingencia.

Para hacer este estudio de viabilidad se pueden utilizar herramientas como las siguientes, aunque el alumno puede seleccionar la que más le guste o crea conveniente, incluso valorando opciones.

El tamaño total de esta sección dependerá del contenido.

### 6.1. Análisis DAFO

El DAFO (de las iniciales de Debilidades, Amenazas, Fortalezas y Oportunidades) es una metodología de estudio de la situación de un proyecto, analizando sus características internas (Debilidades y Fortalezas) y su situación externa (Amenazas y Oportunidades) en una matriz cuadrada como muestra la Ilustración 11.



Ilustración 11: Esquema de un análisis DAFO.  
(Fuente <http://egesoftware.blogspot.com.es>)

Ahora expondré la matriz cuadrada de ChessCloud, para su realización he utilizado la [herramienta](#) proporcionada por el gobierno de España.

#### 6.1.1. DAFO

Debilidades	Amenazas
<ul style="list-style-type: none"> <li>Falta de financiación inicial.</li> <li>Falta de experiencia.</li> </ul> <p>Añadir debilidad +</p>	<ul style="list-style-type: none"> <li>Al manejar dos esquemas de base de datos es menos consistente.</li> <li>Usuarios con poco conocimiento digital.</li> <li>Cambio en las preferencias de los usuarios.</li> <li>Dependeremos en gran medida de la colaboración de los usuarios</li> </ul> <p>Añadir amenaza +</p>
Fortalezas	Oportunidades
<ul style="list-style-type: none"> <li>Buen liderazgo.</li> <li>Conocimientos sobre herramientas y tecnologías necesarias.</li> <li>Conocimientos en cuanto a promoción, redes sociales, etc.</li> </ul> <p>Añadir fortaleza +</p>	<ul style="list-style-type: none"> <li>Cada usuario podrá ver como está dentro del conjunto y motivarse para ayudar y ser ayudado.</li> <li>Fácilmente escalable.</li> <li>Se pueden añadir funcionalidades fácilmente.</li> <li>Innovador.</li> <li>Ningún usuario será lo bastante influyente sobre el resto.</li> <li>Explotación del auge de la nube,</li> </ul> <p>Añadir oportunidad +</p>

Ilustración 12: DAFO ChessCloud  
Fuente: Propia realizada en <https://dafo.ipyme.org/Home>



Cómo podemos observar en la figura “A1” tendremos cinco aspectos internos (dos debilidades y tres fortalezas) y nueve aspectos externos (cuatro amenazas y cinco oportunidades), procedo a explicar cada uno:

- Situaciones internas:
  - Debilidades:
    - Deb1: Entendemos por falta de financiación inicial a la situación de entrar al proyecto sin un beneficio económico directo, le asignamos una importancia baja debido a qué no es el objetivo del proyecto el lucrarse por lo que no hay motivo para no sumergirnos en esta aventura.
    - Deb2: Falta de experiencia es claramente una referencia a qué es el primer proyecto abordado de este aspecto y, sobre todo, de este tamaño. Le asignamos importancia baja porque la experiencia la iremos consiguiendo conforme avancemos en el proyecto.
  - Fortalezas:
    - For1: Buen liderazgo es sin lugar a duda un aspecto crucial para cumplir los objetivos, sin una buena gestión un proyecto está condenado al más absoluto desastre. Le asignamos importancia alta por todo ello.
    - For2: Tener conocimiento sobre que metodologías, tecnologías... Es poco menos que fundamental para ahorrar tiempo y para conseguir no sólo cumplir plazos, sino cumplirlos bien. Le asignamos importancia alta por todo ello.
    - For2: Los conocimientos de promoción serán también de relativa relevancia, sin embargo, le asignaremos una importancia media debido a que no es tan fundamental en el desarrollo.
- Situaciones externas:
  - Amenazas:
    - Am1: El manejo de una base de datos con dos “subesquemas” distintos dentro de un mismo esquema complica su manejo, importancia media.
    - Am2: Los usuarios con poco conocimiento en la red pueden ser más complicados para que comprendan nuestra plataforma, importancia media.
    - Am3: Deberemos estar siempre pendientes de lo que el usuario busca y lo que no, lo que le satisface y lo que no, etc. Importancia media.

- Am4: Nuestra plataforma dependerá de lo que el usuario decida, si decide implicarse será un éxito, pero si por el contrario no es así no tendremos plataforma. Importancia crucial.
- Oportunidades:
  - Op1: Que los usuarios se impliquen y se vean en el núcleo y la comunidad será muy importante.
  - Op2: Plataforma fácilmente de escalar, importancia media.
  - Op3: Plataforma a la que fácilmente le añadiremos nuevas funcionalidades, importancia media.
  - Op4: Pioneros en esta idea de web sobre ajedrez con colaboración entre usuarios, importancia alta.
  - Op5: Ningún usuario será más importante que el resto, importancia crucial para el usuario.
  - Op6: Explotaremos el auge de la demanda de la nube dejando un poco “de lado” la idea de las plataformas de escritorio, importancia media.

### 6.1.2. Estrategias

Ahora hablaremos de las estrategias ordenadas por prioridad y la matriz de factores. El orden está relacionado con los factores DAFO asociados a cada estrategia.

En primer lugar, a modo de resumen observamos la figura A2.

 <b>Estrategias Supervivencia</b> 	 <b>Estrategias Adaptativas</b> 
Formación    <div>Añadir estrategia +</div>	Aprovechar oportunidades de un mercado aún no explotado   Satisfacción del usuario   <div>Añadir estrategia +</div>
 <b>Estrategias Defensivas</b> 	 <b>Estrategias Ofensivas</b> 
Feedback con los trabajadores   Investigación   <div>Añadir estrategia +</div>	Política de derechos ofensiva.   Política de búsqueda de demanda ofensiva.   Política de promociones ofensiva   <div>Añadir estrategia +</div>

*Ilustración 13: Estrategias ChessCloud*

*Fuente: propia realizada en <https://dafo.ipyme.org/Home>*

Primero hablaremos sobre cada estrategia:

- Las estrategias de supervivencia se encargan de las debilidades (aspecto interno) y de las amenazas (aspecto externo).
- Las estrategias adaptativas afrontan las debilidades (aspecto interno) y las oportunidades (aspecto externo).
- Las estrategias defensivas se ocupan de las fortalezas (aspecto interno) y amenazas (aspecto externo).
- Las estrategias ofensivas abordan las fortalezas (aspecto interno) y oportunidades (aspecto externo).

Como podemos ver en ningún caso una estrategia “mezcla” aspectos internos con internos ni externos con externos, sino que a cada uno de ellos lo relaciona con cada uno del otro, obteniendo cuatro tipos de estrategias como resultado de  $n^m$  donde n es el número de aspectos internos y m el de externos (o viceversa ya que son el mismo número).

Ahora hablaremos de cada estrategia que hemos desarrollado:

- Es1. Estrategia ofensiva. Política de derechos ofensiva.
  - El usuario sabrá en todo momento que no habrá ningún otro usuario por encima de ellos. afectará a las oportunidades Op1 y Op5, para que el usuario se percate de los múltiples beneficios.
- Es2. Estrategia defensiva. Feedback con los trabajadores.
  - El equipo debe verse implicado e importante en el desarrollo, con un equipo feliz tendremos un resultado feliz. Afectará a la fortaleza For1.
- Es3. Estrategia Adaptativa. Aprovechar oportunidades de un mercado aún no explotado.
  - Inexistencia de una plataforma similar en la red, siendo ésta la primera al más puro estilo "código abierto". Aborda la Oportunidad Op4.
- Es4. Estrategia Adaptativa. Satisfacción del usuario.
  - Ellos son "el jefe", si ellos son felices nosotros somos felices. Trabaja la oportunidad Op1.
- Es5. Estrategia Defensiva. Investigación.
  - Deberemos estar en constante contacto con cómo reaccionan o no los usuarios de la plataforma, mediante herramientas como Google Analytics, encuestas, etc. La relacionamos con las amenazas Am1, Am2 y Am4.
- Es6. Estrategia Ofensiva. Política de promociones ofensiva.
  - Deberemos "atacar" las nuevas tecnologías para hacerlas llegar al usuario de la mejor forma posible. Afectará a la oportunidad Op6.

- Es7. Estrategia Ofensiva. Política de búsqueda de demanda ofensiva.
  - Deberemos percatarnos de qué y en qué medida solicitan los usuarios para abordar esas partes en específico. Afectará a las oportunidades Op2 y Op3.
- Es8. Estrategia Supervivencia. Formación.
  - Invertir tiempo no sólo en producir y ganar experiencia sino también en formación sobre nuevas tecnologías teniendo siempre en cuenta la curva del aprendizaje y las exigencias inmediatas. Tendrá relación con la amenaza Am1 y con la debilidad Deb2.

Aquí han sido expuestas las estrategias para gestionar los aspectos internos y externos considerados.

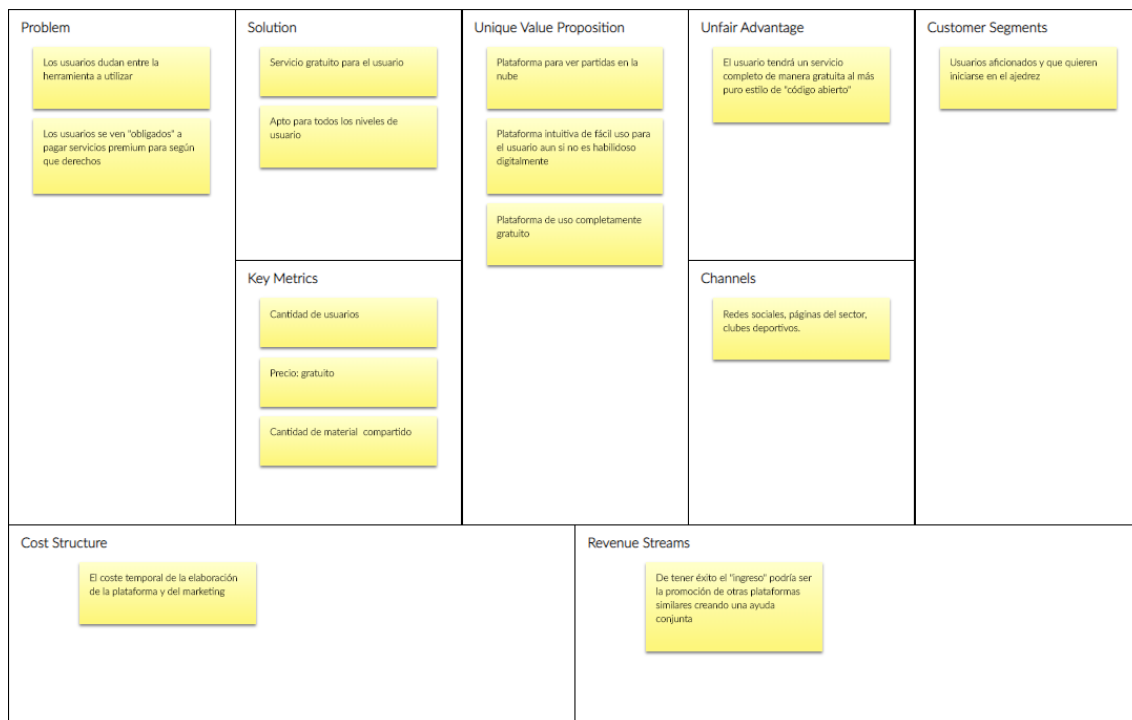
## 6.2. Lean Canvan

Lean Canvan es otra de las herramientas que pueden ser idóneas para analizar un producto que está siendo creado y que tiene carácter innovador, ya sea porque es una solución nueva a un problema ya existente o bien porque es una mejora sobre otras soluciones que ya existen. La idea de Lean Canvan es analizar el proyecto desde diferentes perspectivas de interés como muestra la **¡Error! No se encuentra el origen de la referencia.**, no solo desde el desarrollo o los costes.

Problema  (alternativas)	Solución	Proposición de valor única	Ventaja especial	Segmentos de cliente  (usuarios visionarios)
	Métricas clave		Canales	
Estructura de costes			Flujos de ingresos	

*Ilustración 14: Cuadro para el análisis Lean Canvan  
Fuente: Propia*

Una vez mencionado esto pasaremos a exponer la figura A3 con el lean canvan e chesscloud



*Ilustración 15: Modelo lean canvas*  
*Fuente: Propia realizada en <https://canvanizer.com/>*

Como podemos observar tenemos varias partes:

- **Problema:** Los problemas de los usuarios que deberemos tratar.
  - La necesidad de elección de los usuarios en una plataforma.
  - El coste de éstas para según qué servicios.
- **Solución:** Como lo vamos a solventar.
  - Un servicio gratuito.
  - Un servicio valido para todos los usuarios.
- **Métrica clave:** nuestras características más importantes.
  - Cantidad de usuarios.
  - Precio.
  - Cantidad de material compartido.
- **Valor único:** Qué nos hace ser como somos.
  - Servicio gratuito.
  - Plataforma intuitiva y fácil de usar incluso para un usuario no hábil digitalmente.
  - Plataforma para ver partidas en la nube.
- **Ventaja:** Qué nos diferencia del resto.
  - El usuario tendrá un servicio completo y gratuito al más puro estilo "código abierto".

- Canales: Por donde nos promocionaremos.
  - Redes sociales.
  - Clubes deportivos.
  - Páginas especializadas de ajedrez.
- Clientes: A quién dirigimos la plataforma.
  - A quién sea aficionado o desee iniciarse en el ajedrez.
- Costes: Costes de la plataforma.
  - El coste temporal de crear la plataforma.
- Flujo de ingresos:
  - El “ingreso” sería poder llegar a mayor cantidad de gente y promocionar, por ejemplo, otra plataforma de la misma filosofía para una colaboración mutua y continua.

### 6.3. Análisis de riesgos

He de decir que en el aspecto de carga de trabajo quien me aconsejó fue mi tutor Domingo Gallardo, él fue mi profesor en Lenguajes y Paradigmas de la Programación y siempre he dicho y diré que al verle explicar se puede ver a alguien que disfruta de hacerlo, sin llegar a entrar en juicios personales de si es buen o mal profesor no querría haber contactado con un tutor que pienso no se hubiera implicado e incluso hubiera disfrutado al ver los progresos.

Le comenté varias ideas indicándole que esta era mi favorita y, por suerte, él coincidió.

Esta era una aventura muy complicada para mí debido a que no he cursado el itinerario más especializado en el desarrollo, sino que opté por las tecnologías de la información. Sin arrepentirme en absoluto de mi decisión he de reconocer que cursando el otro itinerario todo hubiera sido más sencillo, menos complicado.

Inspirado por el querer hacer algo que me hiciera sentir no solamente bien por lo conseguido, sino también un “informático completo” que ha conseguido desarrollar un proyecto por su

cuenta me adentré en este bosque con muchos obstáculos como tener que estudiar de nuevo Laravel o introducirme en JS. Todo ello con la dificultad de como plantear todo para poder realizarlo de la mejor forma posible.

No podría asegurar haber cumplido las 300 horas ni por encima ni por abajo, tampoco le doy una importancia al hecho en si fueron 250 o 350, lo que me ha llevado aquí es haber iterativamente ido resolviendo distintas tareas empezando con un sencillo iniciar sesión y llegando a la totalidad del proyecto. BootStrap fue otro desafío pues no se trata de hacerlo, sino de hacerlo bien y que el usuario así lo vea.

Tras todo este análisis de los problemas de afrontar disciplinas menos trabajadas me adentré a afrontar el problema, pues a vivir se aprende viviendo y a desarrollar, desarrollando

Para acabar, como “especializado” en otro campo fue duro y un proceso de aprendizaje constante enfrentarme a este proyecto, pero si tuviera que elegir de nuevo el proyecto no tendría dudas en afrontar ChessCloud.

## 7. Análisis y especificación

El problema que debemos afrontar es la situación de los usuarios (aficionados al ajedrez) para encontrar la mejor página y, sobre todo, para abordar el problema de las suscripciones premium para determinados servicios, sin ningún tipo de colaboración con el resto de usuarios.

En esta parte he realizado un análisis mediante ingeniería inversa de los requisitos, ahora pondré mis conclusiones al respecto.

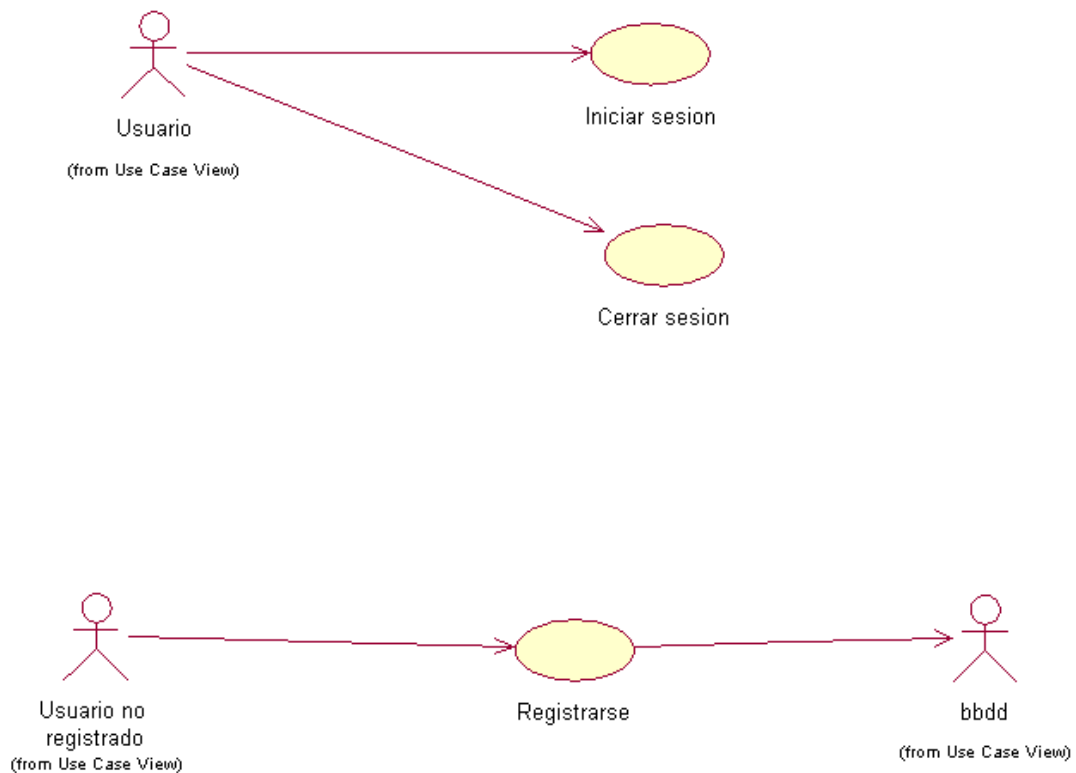
### 7.1. Casos de uso

En esta ocasión hemos usado la herramienta Rational Rose.

A modo de resumen nos centraremos en los conceptos de la sesión y el de compartir partidas para así reflejar su funcionamiento.

En primer lugar, por usuario entendemos cualquier usuario registrado en la web.

#### 7.1.1. Sesión

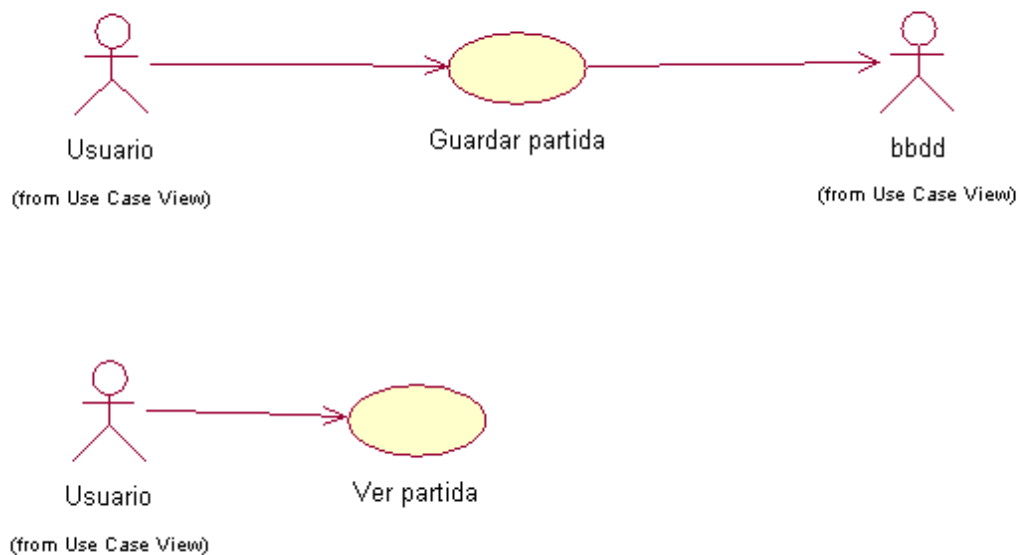




*Ilustración 16: Casos de uso sesión*  
*Fuente: Propia*

Como podemos ver en la figura A5 el usuario puede iniciar sesión o cerrarla según su estado actual. por otro lado, si no estamos registrado debemos interactuar con la base de datos para hacerlo. Tal como está implementada nuestra herramienta al registrarnos iniciaremos sesión de manera automática.

#### 7.1.2. Guardar y ver partidas



*Ilustración 17: Casos de uso guardar y ver partidas*  
*Fuente: Propia*

Como podemos observar en la imagen A6 el usuario guardarla partida y ésta se registra en la bd para que luego un usuario pueda verla.

## 7.2. Pirámide de requisitos

La herramienta que hemos utilizado ha sido en Rational RequisitePro.

Con pirámide entendemos a la de todo el proyecto, os expondré al igual que antes los principales.

### 7.2.1. Sesión

Aquí tendremos tres partes, iniciar, cerrar y registrar. Ahora pondremos una tabla a modo explicativo de cada una de estas funciones.

RF-001	Iniciar sesión
Objetivos asociados	Aspectos de la plataforma web
Requisitos asociados	Aspectos relacionados con la sesión del usuario
Descripción	El sistema deberá comportarse tal como se describe en su caso de uso
Precondición	Estar registrado
Secuencia normal	<ol style="list-style-type: none"><li>1. El usuario se dirige al formulario</li><li>2. El usuario rellena los campos</li><li>3. El usuario manda sus datos a la base de datos</li></ol>
Postcondición	Ninguna
Excepciones	Si las credenciales son incorrectas no funcionará
Rendimiento	Alta
Frecuencia esperada	Alta
Estabilidad	Alta

*Tabla 1: requisito iniciar sesión  
Fuente: Propia*

RF-002	Cerrar sesión
Objetivos asociados	Aspectos de la plataforma web
Requisitos asociados	Aspectos relacionados con la sesión del usuario
Descripción	El sistema deberá comportarse tal como se describe en su caso de uso
Precondición	Estar con la sesión iniciada
Secuencia normal	<ol style="list-style-type: none"><li>1. El usuario presiona en cerrar sesión</li><li>2. La sesión se cancela</li></ol>
Postcondición	Ninguna
Excepciones	Ninguna

Rendimiento	Alta
Frecuencia esperada	Alta
Estabilidad	Alta

*Tabla 2: requisito cerrar sesión  
Fuente: Propia*

RF-003	Registro de usuario
Objetivos asociados	Aspectos de la plataforma web
Requisitos asociados	Aspectos relacionados con la sesión del usuario
Descripción	El sistema deberá comportarse tal como se describe en su caso de uso
Precondición	Estar en la plataforma sin estar con una sesión iniciada
Secuencia normal	<ol style="list-style-type: none"> <li>1. El usuario se dirige al formulario</li> <li>2. El usuario rellena los campos</li> <li>3. El usuario manda sus datos a la base de datos</li> </ol>
Postcondición	Ninguna
Excepciones	<ol style="list-style-type: none"> <li>1. Campos erróneos</li> <li>2. El usuario ya existe</li> </ol>
Rendimiento	Alta
Frecuencia esperada	Alta
Estabilidad	Alta

*Tabla 3: requisito registrarse  
Fuente: Propia*

### 7.2.2. Partidas

Aquí tendremos dos partes, guardar y ver partidas. Ahora pondremos una tabla a modo explicativo de cada una de estas funciones.

RF-004	Guardar partidas
Objetivos asociados	Aspectos de la plataforma web

Requisitos asociados	Aspectos relacionados con las partidas
Descripción	El sistema deberá comportarse tal como se describe en su caso de uso
Precondición	Estar en la plataforma sin estar con una sesión iniciada
Secuencia normal	<ol style="list-style-type: none"> <li>1. El usuario acude al formulario</li> <li>2. El usuario introduce los campos</li> <li>3. El usuario introduce los movimientos, si es administrador tendrá la opción adicional de cargar un PGN y sino sólo podrá reproducirla.</li> <li>4. La partida se guarda en la base de datos</li> </ol>
Postcondición	Ninguna
Excepciones	<ol style="list-style-type: none"> <li>1. Campos erróneos</li> <li>2. Los movimientos no se corresponden a los de una partida “jugable”</li> </ol>
Rendimiento	Alta
Frecuencia esperada	Alta
Estabilidad	Alta

*Tabla 4 requisito guardar partidas*

*Fuente: Propia*

RF-004	Ver partidas
Objetivos asociados	Aspectos de la plataforma web
Requisitos asociados	Aspectos relacionados con las partidas
Descripción	El sistema deberá comportarse tal como se describe en su caso de uso
Precondición	Estar en la plataforma sin estar con una sesión iniciada
Secuencia normal	<ol style="list-style-type: none"> <li>1. El usuario acude a la lista de partidas</li> <li>2. El usuario selecciona la partida que desea ver</li> </ol>
Postcondición	Ninguna
Excepciones	Ninguna
Rendimiento	Alta
Frecuencia esperada	Alta
Estabilidad	Alta

*Tabla 5: requisito ver partidas*

*Fuente: Propia*

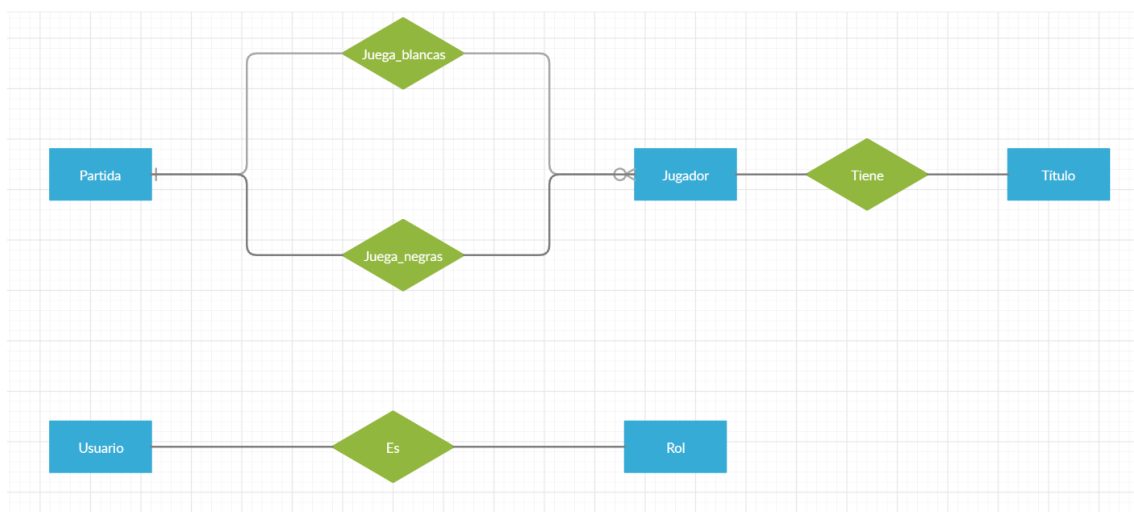
## 8. Diseño e implementación

El capítulo de diseño es con diferencia el más importante del TFG. Podría decirse que es el corazón de dicho trabajo. En este punto se ha de diseñar la solución de forma que dé respuesta a todos y cada una de los requerimientos funcionales y no funcionales anteriormente establecidos. Por tanto en esta sección se hará constante referencia a los identificadores de los requerimientos que se están abordando. Hay que tener en cuenta que un requerimiento puede necesitar de diferentes elementos en un software para ser resuelto. Por ejemplo, si uno de los requerimientos es el control de acceso posiblemente requerirá de dotar de identificación mediante usuario/contraseña al sistema, y por tanto necesitaremos un sistema de gestión de usuarios y permisos de usuario compuesto por una interfaz gráfica, posiblemente algún tipo de API, web servirse u otro acceso a datos, algún sistema de persistencia (base de datos, motor de base de datos, etc.), control de errores, cifrado de las comunicaciones a nivel de infraestructura y cifrado de datos a nivel de BD, etc.

El diseño de soluciones es muy complejo y en él intervienen aspectos a distintos niveles o vistas de un proyecto. Una buena práctica es agrupar el diseño en apartados en función de la materia o el aspecto que se está definiendo, concentrando así todo lo relativo a ese contexto en un único apartado. Una aplicación será la suma en realidad de todos esos diseños.

### 8.1. Diseño de la persistencia

En nuestro caso hemos utilizado una base de datos local MySQL.



Cómo podemos ver en la Figura A7 nuestro esquema consta de dos subesquemas:

1. Esquema de partidas: Aquí tendremos las entidades de Partida, Jugador y Título atendiendo a las siguientes indicaciones:
  - Una partida tendrá un y sólo un jugador de blancas.
  - Una partida tendrá un y sólo un jugador de negras.
  - Un jugador podrá disputar un número  $n$  de partidas tanto con blancas como con negras, pudiendo ser  $n$  igual a 0.
  - Un jugador tendrá o no un título, máximo uno.
  - Un título lo podrán tener 0 o un número  $n$  de jugadores.
2. Esquema de usuarios: Aquí la lógica es más sencilla:
  - Un usuario tendrá un y sólo un rol.
  - Un rol tendrá o no usuarios asociados con un máximo de  $n$  usuarios por cada rol.

Cada entidad se representa en una tabla con un distintos atributos en el que usaremos como identificador un valor numérico incremental no repetible y no nulo para cada una de estas tablas.

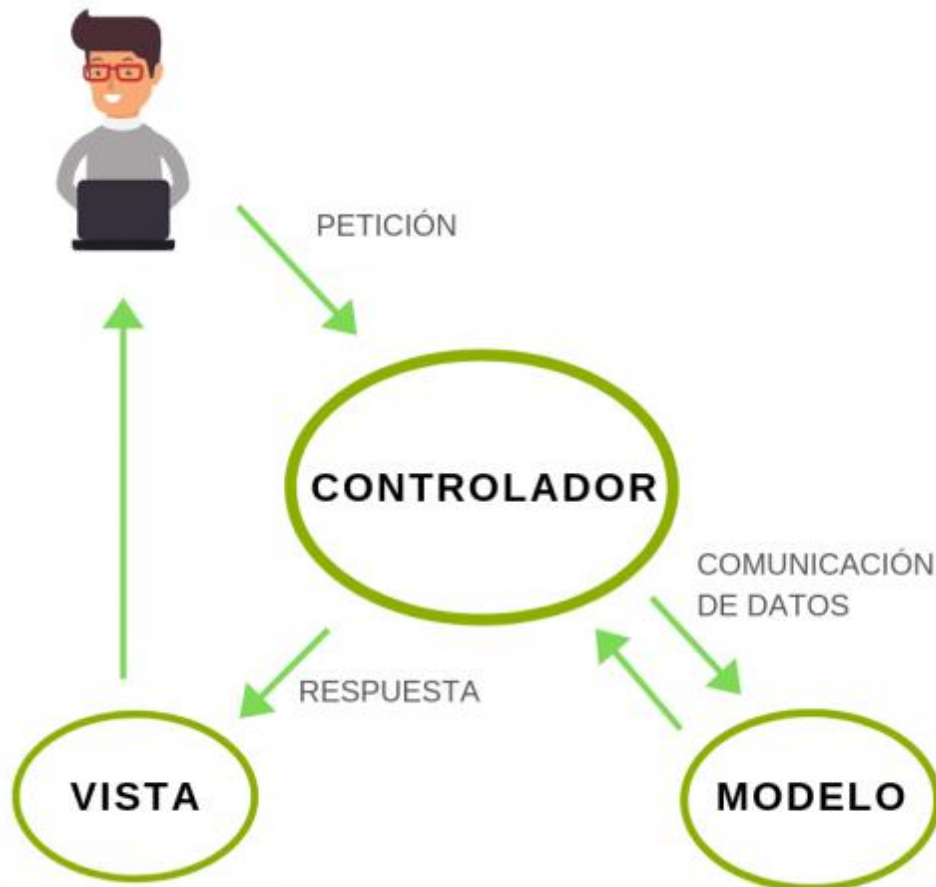
## 8.2. Diseño arquitectura tecnológica Front/Back-end

### 8.2.1. Modelo Vista Controlador

He utilizado el Framework Laravel utilizando la versión de php 7, el modelo que se ha implementado es el Modelo-Vista-Controlador.

Por Modelo-vista-controlador entendemos al patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Gracias a esta separación podremos modificar cada uno de los componentes sin afectar al resto.

Ahora, pasaremos a analizar cómo funciona el modelo.



*Ilustración 19: Ejemplo MVS*  
*Fuente: <https://nicobobb.com/mvc/>*

Podemos observar 3 patrones en el modelo.

- **Modelo:** Es quién se comunica a más bajo nivel con el controlador, se encarga tanto de hacer consultas a base de datos como de controlar los permisos. Teniendo una representación no sólo de la base de datos, sino también de la lógica de negocio y los mecanismos de persistencia.
- **Vista:** Muestra las interfaces al usuario (con los datos que correspondan) a modo de respuestas a sus peticiones.
- **Controlador:** Es quien se encarga de, dada una petición de un usuario gestionarla mediante el modelo y la vista. Podría decirse que sigue el siguiente *modus operandi*:
  1. Recibe la petición del usuario y la procesa.
  2. Se comunica con el modelo quien realiza sus tareas de base de datos.
  3. Recibe la información (datos) necesarios.

4. Lo comunica a la vista para devolver la interfaz adecuada con la información (datos) adecuada.
5. El usuario recibe su respuesta.

En cuanto a las ventajas del MVC. ¿Cuáles son?

- Organización modular. Podremos dividir la lógica del programa haciendo la aplicación más escalable y menos dependiente, lo cual facilita realizar modificaciones.
- Puedes hacer abstracción de datos para facilitar la realización de consultas a la base de datos.
- Podremos tener un código muy organizado y legible.

### 8.2.2. Back-end

En el aspecto del Back-end tendríamos el Modelo y su comunicación con la base de datos, la lógica de negocio y la persistencia.

Pondremos un ejemplo de como accedemos a la base de datos:

```
Route::get('/partidas', 'GameController@verPartidas');
```

Aquí tenemos un método get especificado en el fichero “web.php” donde al tener la url “/partidas” y la petición http de tipo “get” llamaremos al controlador “GameController” y a su método “verPartidas”.

```
public function verPartidas() {  
    //muestro las partidas de la bd  
  
    $games = DB::table('games')->paginate(10);  
  
    return view('verPartidas')->with('games', $games);  
}
```

En este caso no necesitamos al objeto “request” pues será una sencilla consulta del tipo SELECT \* FROM TABLA;

Como podemos ver primero nos declaramos la variable y le asignamos el resultado de la consulta a la tabla “games”, lo paginaremos para poder verlo en mejores condiciones.

Ahora devolveremos la vista “verPartidas” con los datos recogidos en esa consulta para poder utilizarlos.



```

@if(!empty($games) && count($games) > 0)

    @foreach ($games as $g)

        <div class="row">

            <div class="card" style="width: 100%;">

                <div class="card-header" style="background-
color: #007bff; border-color: #007bff; color: white;">Partida
{{{ $g->id }}}</div>

                <div class="card-body">

                    <h4 class="card-title"> {{{ $g-
>surname_white }}} , {{{ $g->name_white }}} vs {{{ $g-
>surname_black }}} , {{{ $g->name_black }}} in {{{ $g->tournament }}}</h4>

                    <p class="card-text">{{{ $g->movements }}}</p>

                    <a href="/partida/{{{ $g->id }}}" class="btn btn-
primary">Ver Partida</a>

                </div>

            </div>

        </div>

    @endforeach

@else

    <p>No hay partidas del jugador</p>

@endif

```

Finalmente podemos ver como lo gestiona la vista, comprueba que no sea vacía la lista de partidas recogidas y si efectivamente no lo es comienza a recorrerla y a realizar sus acciones correspondientes. Si la lista fuera nula avisamos al usuario.

### 8.2.3. Front-end

En este aspecto no hemos utilizado ninguna tecnología concreta, hemos utilizado nuestros conocimientos sobre el lenguaje JavaScript para acceder al DOM y así cubrir las necesidades de

manera conjunta a todas las facilidades de Laravel en su apartado de vistas del modelo MVC como son los formularios, HTML, Etc.

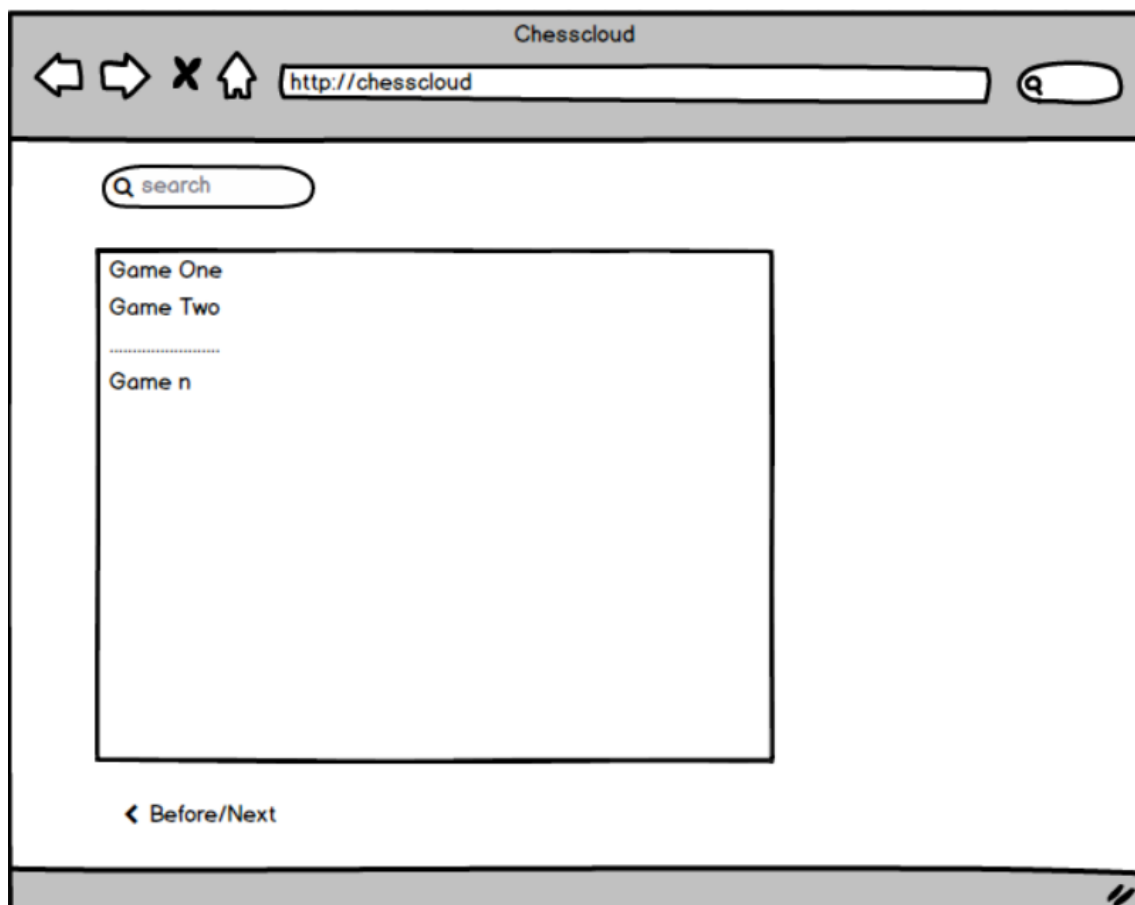
Las librerías gráficas fueron incorporadas a las vistas de Laravel donde eran necesarias, para cargar las funciones de gestión de las partidas tales cómo jugar, analizar, etc.

### 8.3. Diseño Interfaces

En nuestro caso la interfaz la dividiremos en tres partes: La que verán los usuarios no registrados, la que verán los usuarios registrados y la que verán los administradores. Aquí expondremos el mockup principal de cada parte.

#### 8.3.1. Interfaz de usuario no registrado

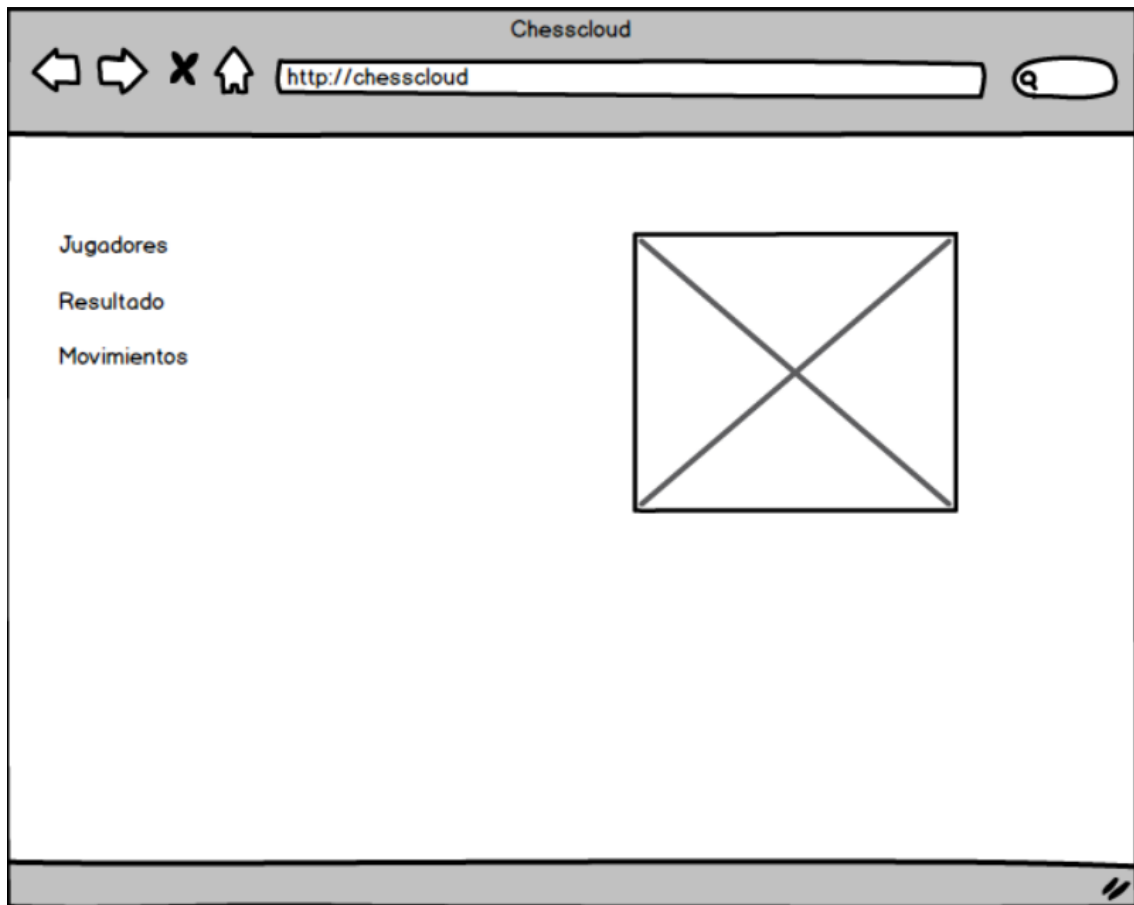
Como podremos ver en la figura A9 el usuario no registrado podrá ver la lista de partidas sin llegar a acceder a ellas.



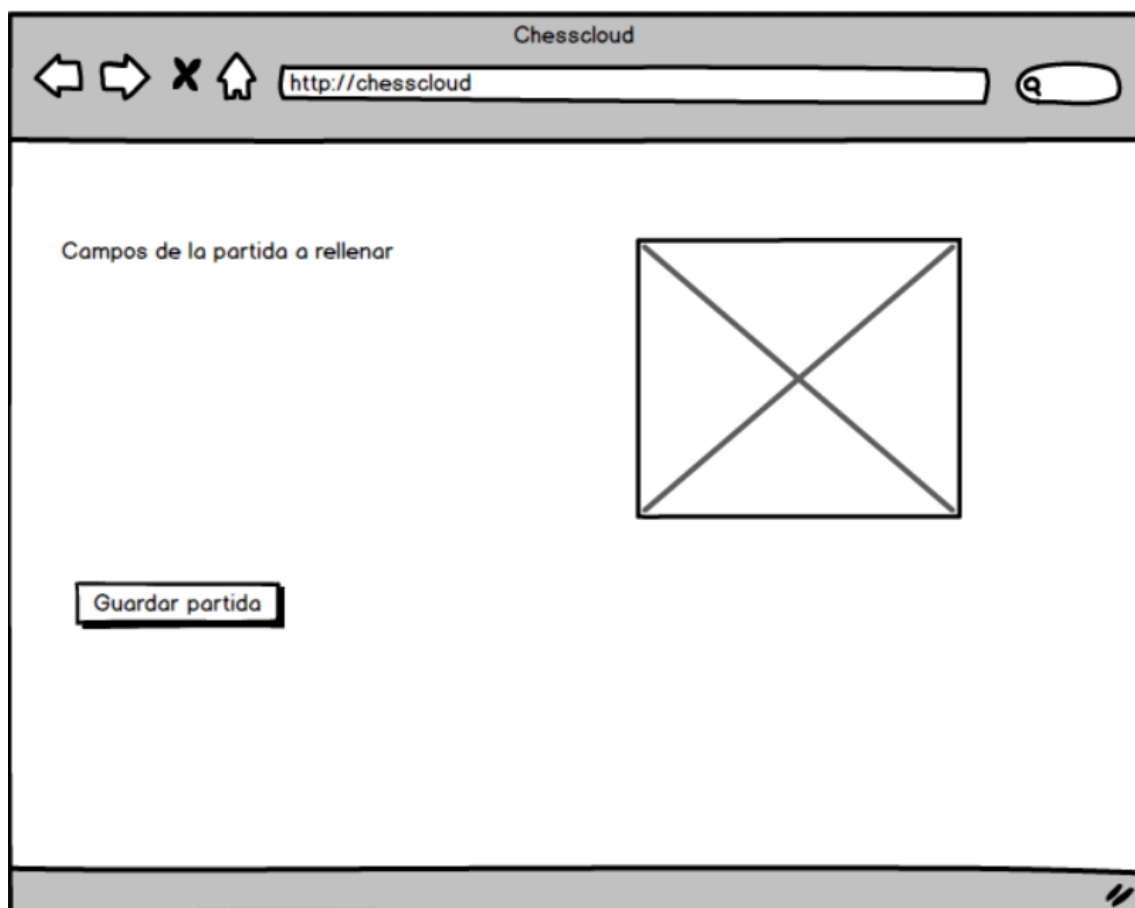
*Ilustración 20: Mockup lista de partidas ChessCloud  
Fuente: Propia*

### 8.3.2. Interfaz de usuario registrado

Como podremos ver en las figuras A10 y A11 el usuario registrado podrá acceder a visualizar cada partida y guardar sus propias partidas, respectivamente.



*Ilustración 21: Ver partida ChessCloud  
Fuente: Propia*



*Ilustración 22: Guardar partida ChessCloud  
Fuente: Propia*

### 8.3.3. Interfaz de usuario administrador

Como podremos ver en la figura A12 el usuario administrador podrá ver las vistas de administrador.

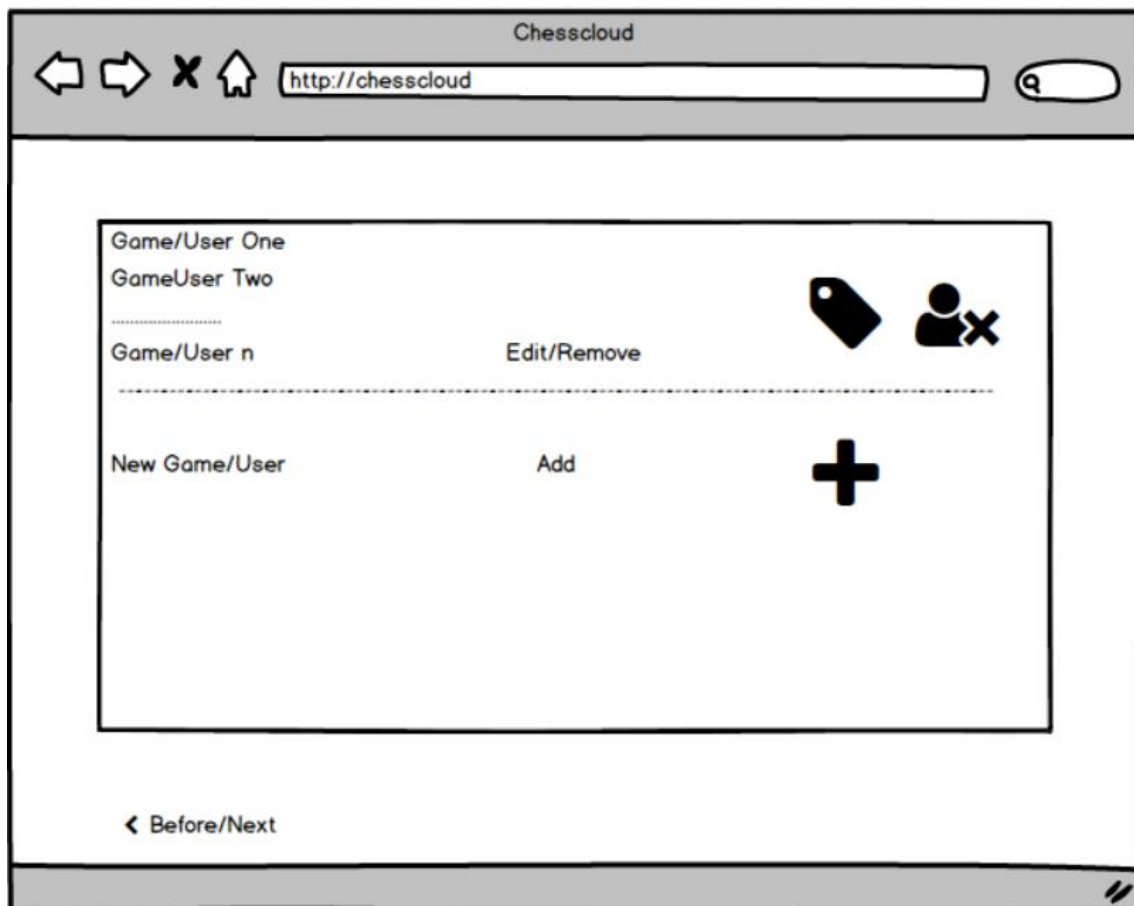


Ilustración 23: Mockup vista administrador Chesscloud  
Fuente: Propia

## 8.4. Implementación

¿Cómo hemos implementado?

Todos tenemos en mente que para crear una vista necesitamos que esa vista corresponda a una URL que pondremos en nuestro navegador, ¿qué pasos debemos realizar?

En primer lugar, el archivo “principal” en todo esto será web.php dónde no sólo tendremos la gestión de errores y el middleware (especificados aquí y en kernel.php esa sintaxis la asignamos a determinado fichero) sino que también tendremos la gestión de los controladores en función a la url que tengamos. Pongamos un ejemplo:

```
Route::get('/admin/usuarios', 'UserController@adminUser')->middleware('admin');
Route::post('/admin/usuarios', 'UserController@execAdmin')->middleware('admin');
```

Ilustración 24: Ejemplo web.php  
Fuente: Propia

Observamos en la figura A13 que tendremos en primer lugar el tipo de método, get si recogemos datos y post y los enviamos, habrá casos que solamente necesitemos un get, pero en este ejemplo como el administrador podrá ver y subir nuevas partidas desde tal vista también tendrá la opción de post. Nos fijamos en qué tendremos dos parámetros correspondientes a la URL que detectemos y al controlador siguiendo el modelo 'controlador'@'metodo\_del\_controlador', por lo que cuando obtengamos esas URL con los métodos get o post llamará a una función u otra según la especificación del web.php. Por último, tendremos opcionalmente el middleware aplicado.

Una vez llamamos al método del controlador no acaba ahí, éste devolverá la vista oportuna con o sin datos según la necesidad, en el método del controlador es dónde se realizaría el intercambio de datos con el modelo del que hablamos cuando explicamos el [MVC](#), a menudo usamos métodos recogidos en el formulario de la vista anterior para consultar nuevos datos como el usuario al que debemos aplicar la modificación, etc. En cualquier caso, la vista que devolverá con los datos considerados (podría ser una vista "a pelo" sin datos adicionales) será la que le llegué al usuario como respuesta a su petición.

Este ha pretendido ser un resumen de la forma de trabajar y como eran los "saltos" que dábamos entre web.php, controladores y vistas, ahora pasaremos a exponer como eran las que veían los usuarios.

## 9. Funcionalidades

Ahora nos meteremos más en el territorio de qué hemos hecho y qué no en la plataforma.

### 9.1. Vistas

Dicho lo anterior, ahora hablaremos de las principales vistas de nuestra web.

### 9.1.1. Vistas de usuario no registrado



*Ilustración 25: Home ChessCloud  
Fuente: Propia*

En la barra de navegación veremos las opciones:

- Entrar: Para poder iniciar sesión.
- Registrarse: Para poder darnos de alta.
- Partidas: Para poder consultar la lista de partidas.

En esta página Home veremos también un carrusel de fotografías sobre las que podemos también seleccionar Entrar y Registrarse como en la barra de navegación.

*Entrar*

The image shows the login page of the ChessCloud website. The page has a blue header with the word 'Entrar' in white. Below the header, there is a user icon and three input fields labeled 'Email', 'Nombre', and 'Contraseña'. At the bottom left, there is a blue button labeled 'Entrar'.

*Ilustración 26: Entrar ChessCloud*

*Fuente: Propia*

En esta parte veremos un formulario que debemos rellenar, los campos son email, nombre y contraseña. Al rellenarlos y pulsar en el botón de entrar se mandará la petición al controlador que verificará si los datos han sido correctos o no.

*Registrarse*

The image shows a web form titled "Registrarse" (Register) in a blue header. Below the header, there is a user icon and four input fields labeled "Email", "Nombre" (Name), "Contraseña" (Password), and "Confirmar contraseña" (Confirm password). Below the "Confirmar contraseña" field, there is a checkbox with a checkmark and the text "¿Acepta las condiciones de uso ...?". At the bottom left, there is a blue button labeled "Registrarse".

**Registrarse**

Email

Nombre

Contraseña

Confirmar contraseña

☒ ¿Acepta las condiciones de uso ...?

Registrarse

*Ilustración 27: Registrarse ChessCloud*  
*Fuente: Propia*

En esta vista veremos otro formulario que debemos rellenar, los campos son email, nombre y contraseña la cual deberá ser confirmada, además debemos marcar la casilla de condiciones de uso. Al rellenarlos y pulsar en el botón de registro se mandará la petición al controlador que verificará si los datos han sido correctos o no.



## Partidas

The screenshot displays the 'Partidas' (Games) section of the ChessCloud application. At the top, there is a search bar titled 'Buscador' with four input fields: 'nombre o apellidos' (name or surnames), 'ranking', 'torneo' (tournament), and 'ambos' (both). A blue 'Buscar' (Search) button is located below these fields. Below the search bar, the section is titled 'Lista de partidas' (List of games). Under this title, there is a pagination bar with a series of numbered links from 1 to 9, with the first link (1) being highlighted. Below the pagination bar, a specific game is displayed under the heading 'Partida 1'. The game title is 'Epifanio,Aitor vs Tevar,Flanagan in Magistral Barcelona'. Below the title, the game moves are listed in a single line: '1.e4 e6 2.b3 d5 3.Bb2 dxe4 4.Nc3 Cd7 5.Nxe4 Ngf6 6.Nc3 Be7 7.g3 0-0 8.Bg2 c6 9.Nge2 Nd5 10.a4 Nxc3 11.Nxc3 Nf6 12.0-0 Nd5 13.Ba3 Bxa3 14.Rxa3 Nxc3 15.dxc3 Qe7 16.Ra1 Rd8 17.Qe2 Bd7 18.Rfd1 Be8 19.Rd4 Qf6 20.Rad1 Rxd4 21.Rxd4 Rd8 22.Qd2 Rxd4 23.Qxd4'. At the bottom of the game details, there is a blue button labeled 'Ver Partida' (View Game).

*Ilustración 28: Lista de partidas ChessCloud  
Fuente: Propia*

En esta ocasión veremos una lista de todas las partidas de la base de datos con paginación para poder navegar cómodamente entre ellas ordenadas por su id, veremos una vista previa de los movimientos de esa partida y podremos presionar en ver la partida. Sin embargo, al no estar con la sesión no podremos verla y se nos redirigirá a la página de Entrar antes explicada.

En adición a esto tenemos un buscador de partidas, en él podremos seleccionar nombre o apellidos del jugador, ranking de éste y especificar si la búsqueda será para el jugador de blancas, el de negras o ambos. También podremos seleccionar el torneo donde se disputo la partida como campo de búsqueda.

### 9.1.2. Vistas de usuario registrado

¿Qué tendremos nuevo?

- Ahora no habrá Registrar ni Entrar sino Cerrar sesión para poder finalizar nuestra sesión.
- Podremos ver nuestro Perfil y modificar datos.
- Podremos ver las partidas.
- Podremos guardar partidas.
- Podremos practicar.

## Perfil

### Editar Perfil

Nombre:

Email:

Contraseña:

Confirmar contraseña:

*Ilustración 29: Perfil ChessCloud  
Fuente: Propia*

En esta vista podemos modificar nuestros datos. Al mandar la petición el controlador comprobará que los datos sean correctos.

## Practicar

### Juega y práctica vs nuestra IA!

Search depth:

Positions evaluated:

Time:

Positions/s:

Tablero

8								
7								
6								
5								
4								
3								
2								
1								
	a	b	c	d	e	f	g	h

*Ilustración 30: Practicar ChessCloud*

Fuente: Propia

Como observamos podremos seleccionar la dificultad ala que enfrentarnos en un baremo entre uno y cinco. Además, nos irá marcando los movimientos y el tiempo de la IA en jugar, nosotros jugaremos con blancas y ella responderá con negras.

[Ver partida](#)

The screenshot displays the 'Ver partida' (View game) interface on ChessCloud. On the left, a sidebar titled 'Partida 1' (Game 1) shows the game details: 'Epifanio,Aitor 1450 vs Tevar,Flanagan in Magistral Barcelona', the result 'Resultado: 1 - 0', and the color 'juegan blancas' (white to move). Below this, the full game notation is listed: '1. e4 e6 2. b3 d5 3. Bb2 dxe4 4. Nc3 Nd7 5. Nxe4 Ngf6 6. Nc3 Be7 7. g3 0-0 8. Bg2 c6 9. Nge2 Nd5 10. a4 Nxc3 11. Nxc3 Nf6 12. 0-0 Nd5 13. Ba3 Bxa3 14. Rxa3 Nxc3 15. dxc3 Qe7 16. Ra1 Rd8 17. Qe2 Bd7 18. Rfd1 Be8 19. Rd4 Qf6 20. Rad1 Rxd4 21. Rxd4 Rd8 22. Qd2 Rxd4 23. Qxd4'. A toggle switch for 'Desactivado' (Deactivated) is also present. The main area on the right features a chessboard titled 'Tablero' (Board) showing the position after move 23. The board is labeled with ranks 1-8 and files a-h. Below the board are navigation buttons: 'Anterior' (Previous), 'Siguiente' (Next), and 'Volver a la jugada 1 donde juegan blancas' (Return to move 1 where white plays).

Ilustración 31: Ver partida ChessCloud

Fuente: Propia

Esta es la parte más completa de la plataforma, pasamos a explicar cada aspecto:

- Datos de la partida: Id de la partida, jugadores que la disputaron (con un enlace a la ficha personal de éstos donde veremos sus datos y sus partidas) y resultado de la partida.
- Turno de quién debe mover (blancas o negras).
- Movimientos: Aspecto más importante, aquí tendremos todos los movimientos de la partida que visualizamos pudiendo navegar a la posición que deseemos presionando en el movimiento correspondiente.
- Inteligencia artificial: Opcionalmente podremos activarla o desactivarla, asumiendo que su uso ralentizará la fluidez de la web.
- Botones anterior y siguiente: Para poder ir recorriendo las jugadas del inicio al fin de la partida cómodamente.

- Tablero: Además de navegar podremos “inspeccionar” que hubiera pasado de que hubiera jugado otra jugada alguno de sus jugadores, por lo que podremos dada una posición “jugar” en ella con ambos bandos.
- Botón volver: si después de “jugar” deseamos volver al punto de donde partimos el botón nos llevará ahí indicando que jugada es y el color de quién debía mover.

#### Guardar partida

### Registrar partida

nombre jugador blancas

apellido jugador blancas

País jugador blancas

Título blancas

None ▾

nombre jugador negras

apellido jugador negras

País jugador negras

Título negras

None ▾

Elo jugador blancas

Elo jugador negras

torneo

1 - 0 ▾

Movimientos:

Registrar partida

Tablero

Deshacer última jugada

Reiniciar

*Ilustración 32: Guardar Partida ChessCloud*  
Fuente: Propia

Esta vista es más sencilla que la anterior, al tener como objetivo similar a Registrar anterior se decidió reducir algo el tamaño del tablero y enfatizar el título de Registrar partida para dar esa sensación de que lo realizado será guardado en la base de datos.

Aspectos principales de la vista:

- Formulario de campos: Todos serán necesarios.
- Tablero: En él reproduciremos la partida que deseemos guardar, haremos los movimientos hasta tener la partida deseada y sin ningún tipo de dificultad para el usuario los movimientos se guardaran correctamente en la base de datos.
- Botón deshacer: Si la última jugada fue un error podremos revertirla, tantas veces como gustemos.
- Botón Reiniciar: por si deseamos que la partida vuelva a comenzar.

Al presionar en Registrar partida se invocará al controlador que comprobará todos los datos.

### 9.1.3. Vistas de administrador

Aquí tendremos dos vistas nuevas, como dijimos en el [diseño de la persistencia](#) tendremos dos subesquemas de base de datos, tendremos una vista de administrador para cada uno.

#### Administración de usuarios

ID	Email	Nombre	Password	Rol
1	alvaronavarrolopez@hotmail.com	serverlayer	\$2y\$10\$1X58NLnXNaArM81mPRnJODKT8BJXi	Usuario
2	DavidBelmar777@raspeig.es	David Bolson	\$2y\$10\$eMeG8md6XujHCw.fLsKd1ebFNRK0qC	Editor
8	JavierGalindoEpifanio6@raspeig.es	Javier Galindo Epifanio	\$2y\$10\$aF7TE9foBjQZtr5.9kfy/OO8ZFnx6h6xYl	Administrador
11	AlvaroTejedorGalindo9@raspeig.es	Alvaro Tejedor Galindo	\$2y\$10\$jIy.ec.GTRkVe0q9vI0ueso2Ow55Ipemj	Administrador
12	FranciscoPerezPerez10@raspeig.es	Francisco Perez Perez	\$2y\$10\$XnGEMLjYKGdHPV50rmteGYwmYtft	Administrador
13	EugenioTejedorJimenez11@raspeig.es	Eugenio Tejedor Jimenez	\$2y\$10\$/HG6dFc0cNtgtz4n09eZzu8Ji5h9oStaC	Administrador
14	NereaPerezCarmona12@raspeig.es	Nerea Perez Carmona	\$2y\$10\$TcNkRg9VWwIsyuNxyvTOp3IHgxK7t	Administrador
15	DavidPerezEpifanio13@raspeig.es	David Perez Epifanio	\$2y\$10\$f9pDWB6ZqYoi/BpL5.sKuk6GaPIHqtO	Administrador
16	JavierLopezGarcia14@raspeig.es	Javier Lopez Garcia	\$2y\$10\$.G0CLQ5yulogCgDdH6mG3.5YZ7donB	Administrador
Nombre		Apellido	Contraseña	Usuario

Ilustración 33: Administración de usuarios ChessCloud  
Fuente: Propia

Como observamos tendremos todos los datos de los usuarios para poder trabajar con ellos (editar, crear y eliminar), aunque la parte de edición debería ser bajo la petición de éstos de algún cambio que no resuelven.

Podemos ver cada dato salvo la contraseña respetando siempre la privacidad de los usuarios y podremos navegar entre los distintos usuarios de la base de datos mediante paginación.

#### Administración de partidas

9	Flanagan Tevar	0 Grecia	Albert Belmar	0 China	Magistral Londres	0 - -	1.e4 e6 2.b3 d5 3.Bb2 dxe
10	Carmen Rodes	0 Cuba	Angel Jimenez	1360 Rusia	Liga Barcelona	1/2	1.e4 e6 2.b3 d5 3.Bb2 dxe
11	Nerea Rodríguez	1611 China	Albert Galindo	0 India	Magistral Londres	1 - (-	1.c4 e5 2.Nc3 Nc6 3.Nf3 c
nombre blancas apellido blancas		ranking blancas pais blancas	nombre negras apellido negras	ranking blancas pais negras	torneo	1 - (-	movimientos Ejemplo: 1.e4 e5 2.Nf3

Ilustración 34: Administración de partidas ChessCloud

*Fuente: Propia*

Aquí veremos de forma similar a la anterior todos los datos de las partidas de la base de datos también con paginación, pudiendo también añadir, borrar y editar datos de las partidas. No podremos editar los movimientos de cada partida pues cambiaría la partida en sí.

## 10. Pruebas y validación

Diferenciaremos en tres partes para hacer todo más sencillo.

### 10.1. Pruebas

Hemos creado pruebas automáticas en las que al ejecutar el archivo se ejecutan todas y cada una de las pruebas de éste. No hemos probado exhaustivamente ya que no era una parte fundamental de este TFG, sin embargo, sí he considerado importante probar mínimamente que algunas situaciones funcionasen como debiera ser.

### 10.2. Preparación del entorno

Primero de todo deberemos crear las pruebas, para ello ejecutamos el comando:

```
php artisan make:test UserTest
```

Los tests irán al directorio “/tests/Feature”.

Tras ello crearemos en el archivo “UserTest” las funciones que deseemos a modo de test, donde “UserTest” es el nombre del archivo que se creará y de la clase de éste.

Para preparar el entorno debemos realizar la instalación de phpunit con el siguiente comando:

```
composer global require "phpunit/phpunit=7.2.*"
```

Donde 7.2 es mi versión de php. Si se desconoce la versión se puede hallar de la siguiente forma:

- `php -v`
- `php -version`

Tras ello podremos ejecutar los test.

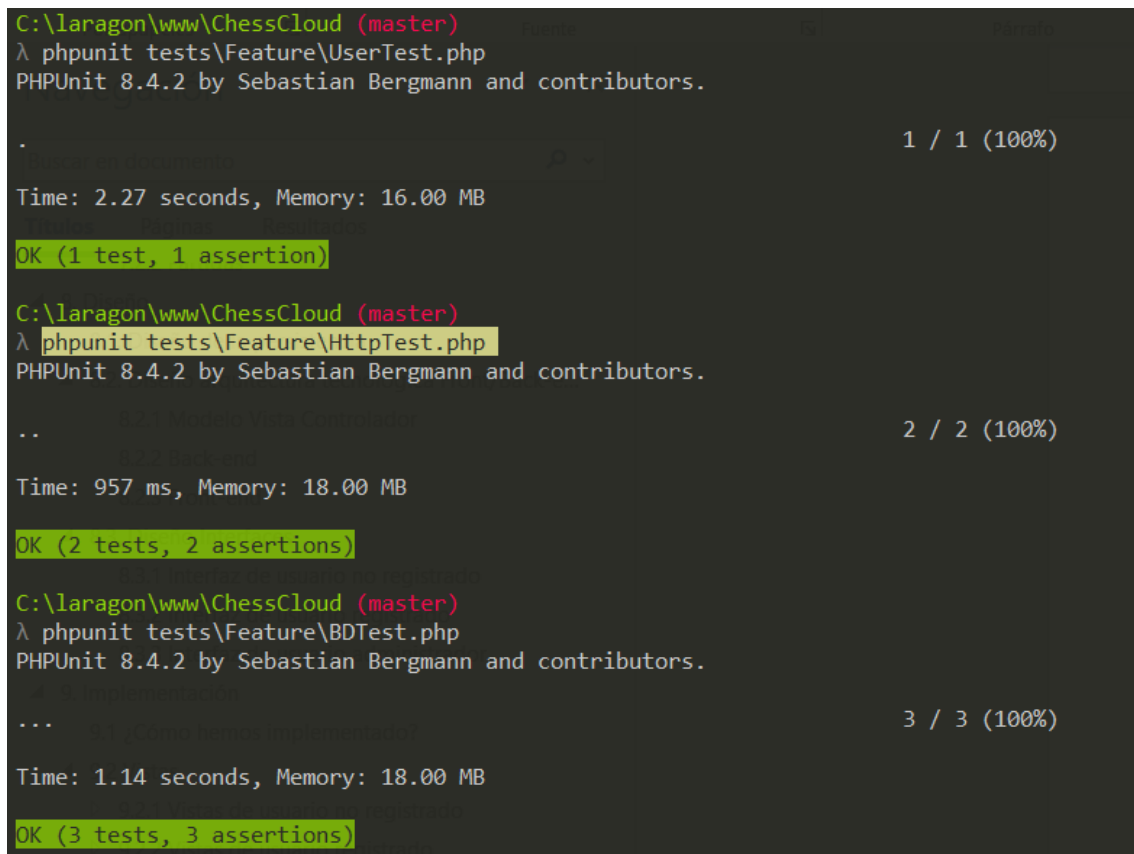
### 10.3. Validación

Ahora ejecutamos el comando:

```
phpunit tests\Feature\NameTest.php
```

Y podremos observar si los test se ejecutan correctamente con los resultados esperados.

Mostramos la figura A24 como demostración.



```
C:\laragon\www\ChessCloud (master)
λ phpunit tests\Feature\UserTest.php
PHPUnit 8.4.2 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 2.27 seconds, Memory: 16.00 MB

OK (1 test, 1 assertion)

C:\laragon\www\ChessCloud (master)
λ phpunit tests\Feature\HttpTest.php
PHPUnit 8.4.2 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 957 ms, Memory: 18.00 MB

OK (2 tests, 2 assertions)

C:\laragon\www\ChessCloud (master)
λ phpunit tests\Feature\BDTest.php
PHPUnit 8.4.2 by Sebastian Bergmann and contributors.

...                                                                3 / 3 (100%)

Time: 1.14 seconds, Memory: 18.00 MB

OK (3 tests, 3 assertions)
```

*Ilustración 35: Resultado de los tests  
Fuente: Propia*

## 11. Conclusiones y trabajo futuro

A modo de resumen de todo este trabajo diré que me ha hecho sentir orgullosos del tiempo dedicado a éste pues desde el minuto uno quise hacer algo que me hiciera sentir como un “informático completo”.

Haber trabajado con Laravel no fue tan complicado, sin embargo, haber tenido esta “dura pelea” con “chess.js” y su alter ego gráfico “chessboard.js” ha sido realmente costoso, tanto temporal como moralmente, pero es gracias a todo ese trabajo que ahora este proyecto está en esta posición y podría ser fácilmente escalable y ampliable con nuevas funcionalidades.

Agradecer de nuevo a mi tutor sus consejos y ayuda durante el proyecto, sin ellos habría sido más complicado todo.

Mis conclusiones respecto a este proyecto son, por el lado de la informática que mediante un servidor que gestione las vistas como en este caso es Laravel y su MVC combinado con JavaScript y llamadas a distintas API se pueden lograr cosas aparentemente muy difíciles. Por el lado conceptual, decir que sí es posible.

Es posible que mediante la colaboración de usuarios se pueda eliminar el modelo actual de servicios premium que ahora rige la política de servicios online, todo el proyecto he hablado de esta idea y de la dualidad ayudar-ser ayudado de los usuarios, me gustaría llevar ese concepto más lejos y que aquí ChessCloud sea la punta del iceberg que puede significar que lo que aquí se hace en ajedrez se hiciera en todo, me gustaría hablar de un concepto de FCS (Free Collaborative Service) que extienda la idea del código abierto a los servicios de las plataformas web.

Y sobre el trabajo futuro, sería ampliar y mejorar esta plataforma mientras se lucha por promocionar este concepto donde los usuarios no sólo son clientes, sino que son jefes.



## Referencias

En primer lugar, dejaré aquí el GitHub del proyecto:

<https://github.com/AlvaroGitHub96/ChessCloud>

Ahora dejaré alguna de las fuentes que me han servido más para recopilar información importante.

1. <https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/>
2. <https://chessboardjs.com/index.html>
3. <https://github.com/jhlywa/chess.js>
4. <https://laravel.com/docs/7.x>
5. <https://stackoverflow.com/>
6. <https://www.w3schools.com/>
7. <https://documentacion-laravel.com/testing.html>
8. <https://forum.laragon.org/>
9. Materiales teóricos y prácticos de la asignatura Diseño de Sistemas Software (tercer curso carácter obligatorio).
10. Materiales teóricos y prácticos de la asignatura Desarrollo de Aplicaciones en Internet (cuarto curso carácter optativo – itinerario Tecnologías de la Información).