UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño y desarrollo de funcionalidades específicas de Buzzer Beater, una red social para la práctica deportiva amateur.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Millán Roldán, David

Tutor/a: Albert Albiol, Manuela

Cotutor/a: Torres Bosch, María Victoria

CURSO ACADÉMICO: 2022/2023

Agradecimientos

En primer lugar, me gustaría agradecer a Vicente por formar parte, apoyar y creer en la idea de este proyecto desde el principio. Han sido unos duros meses de trabajo que, nuestro apoyo mutuo y el deseo de hacer realidad este proyecto por parte de los dos, me han ayudado a seguir trabajando en ello cada día.

En segundo lugar, mi más sincero agradecimiento a Victoria y Manuela, tutoras de este TFG, por apoyar la idea desde un principio, por la rápida respuesta siempre recibida por su parte, por su ayuda y sus consejos para poder realizar un gran trabajo.

Por último, dar las gracias a María por animarme y apoyarme durante todos estos meses. Por darme fuerzas y motivarme para completar este trabajo, por los consejos, por prestarse a probar los diseños, proponer mejoras y por creer en nuestra idea y en mí.

Resumen

Este Trabajo de Fin de Grado tiene como finalidad desarrollar las funcionalidades específicas de Buzzer Beater, una red social dedicada a la organización, publicación y participación de eventos deportivos a nivel aficionado. Los usuarios registrados en dicha red social podrán crear eventos deportivos, realizar búsquedas de eventos publicados, y apuntarse a un evento. Cada evento podrá personalizarse de acuerdo al número de participantes requeridos, el deporte en cuestión, el nivel requerido de los participantes, el día del evento, la localización y el coste por participante. Además, los usuarios podrán unirse a grupos para facilitar la creación de eventos con usuarios del mismo grupo, tanto privados como públicos. Todos estos parámetros podrán ser utilizados para realizar búsquedas. El desarrollo de dicha red social se abordará como una aplicación web *responsive* a la que se pueda acceder tanto desde ordenadores (escritorio y portátiles) como desde dispositivos móviles. Es por ello por lo que la tecnología a utilizar será WordPress como CMS mediante el desarrollo de un tema desde cero, en el *frontend* HTML y SASS (CSS) para maquetar todas las páginas y JQuery (JavaScript) para animaciones y otras funcionalidades básicas. En cuanto a la tecnología *backend*, se empleará PHP para la conexión con la base de datos relacional (SQL), y el paso de datos entre el servidor y el cliente.

Palabras clave: Desarrollo, *fullstack*, aplicación web, red social, deportes, WordPress, PHP, JavaScript, HTML5, CSS3, JSON, SQL, servidor, depuración

Abstract

The aim of this thesis is to develop the specific functionalities of Buzzer Beater, a social network dedicated to the organization, publication and participation of amateur sporting events. Users registered in this social network will be able to create sporting events, search for published events, and sign up for an event. Each event can be customized according to the number of participants required, the sport in particular, the required level of participants, the day of the event, the location and the cost per participant. In addition, users will be able to join groups to facilitate the creation of events with users of the same group, both private and public. All these parameters will be searchable. The development of this social network will be approached as a responsive web application that can be accessed from both computers (desktop and laptops) and mobile devices. That is why the technology to be used will be WordPress as CMS by developing a theme from scratch, in the frontend HTML and SASS (CSS) to layout all pages and JQuery (JavaScript) for animations and other basic functionality. As for the backend technology, PHP will be used for the connection to the relational database (SQL), and the passage of data between the server and the client.

Keywords: Development, fullstack, web application, social network, sport, WordPress, PHP, JavaScript, HTML5, CSS3, JSON, SQL, server, testing

Tabla de contenidos

Agradecimientos 1
Resumen 2
Abstract 3
Tabla de contenidos 4
1. Introducción 8
1.1 Motivación 8
1.2 Objetivos 10
1.3 Estructura 10
1.4 Colaboraciones 11
2. Metodología 12
2.1 Planificación e implementación de SCRUM 12
2.2 Herramientas de planificación 14
2.3 Herramientas de diseño 15
2.4 Herramientas de desarrollo 16
2.5 Herramientas de depuración 17
3. Estado del arte 18
3.1 Identificación de aplicaciones similares y sus funcionalidades 18
3.2 Análisis crítico del estado del arte 22
3.2.1 Listado y filtrado de eventos 22
3.2.2 Perfil de grupo 22
3.2.3 Creación de eventos 22
3.2.4 Conclusiones al análisis de la competencia 23
3.3 Propuesta 23
3.3.1 Listado y filtrado de eventos 23
3.3.2 Perfil de grupo 23
3.3.3 Creación de eventos 24
3.3.4 Perfiles profesionales 24
4. Análisis del problema 25
4.1 Requisitos funcionales de la aplicación 25
4.2 Propuestas alineadas con los requisitos funcionales 27
5. Diseño de la solución 29
5.1 Interfaces gráficas 29
5.1.1 Prototipado 29
5.1.2 Diseño final 31

5.2 Arquitectura del sistema 36
5.2.1 Tecnologías utilizadas 36
5.2.2 Estructura de WordPress 38
5.2.3 Estructura del tema 39
5.2.4 *Plugins* de WordPress 41
5.2.5 Librerías y APIs utilizadas 41
6. Desarrollo de la solución propuesta 43
6.1 Control de versiones 43
6.2 Instalación y preparación del entorno de desarrollo 44
6.3 Desarrollo de los elementos del *backend* 48
6.3.1 Estructura de datos 48
6.3.2 Llamadas AJAX 52
6.4 Consideraciones de seguridad 54
6.5 Base de datos común 55
7. Pruebas 56
8. Implantación 59
8.1 Despliegue 59
8.2 Resultado final 60
9. Conclusiones 64
9.1 Conclusiones 64
9.2 Relación del trabajo desarrollado con los estudios cursados 64
9.3 Trabajos futuros 65
Bibliografía 66
Anexos 68

Tabla de figuras

Figura 1: Variación de hábitos en la práctica deportiva tras la COVID-19. Fuente: (Ministerio de Cultura y Deporte, 2020) 8
Figura 2: Evolución de los usuarios de redes sociales en el mundo de 2015 a 2020. Fuente: (Una vida online, 2021) 9
Figura 3: Porcentaje de práctica deportiva por edades en el último año en España. Fuente: (Ministerio de Cultura y Deporte, 2020) 9
Figura 4: Edad de los usuarios de redes sociales en España. Fuente: (Una vida online, 2021) . 9
Figura 5: Disposición del tablero en Trello del TFG. Fuente: (Elaboración propia) 13
Figura 6: Diagrama de Gantt de los sprints y sus tareas. Fuente: (Elaboración propia) 13
Figura 7: Esquema de la metodología SCRUM. Fuente: (Sinnaps, 2022) 14
Figura 8: Página de eventos de Timpik. Fuente: Timpik 19
Figura 9: Página de eventos de IF7Sports. Fuente: IF7Sports 19
Figura 10: Perfil de grupo o club de Timpik. Fuente: Timpik 20
Figura 11: Pantalla de creación de eventos de Timpik – Nombre y deporte. Fuente: Timpik ... 20
Figura 12: Pantalla de creación de eventos de Timpik – Ubicación y fecha. Fuente: Timpik ... 20
Figura 13: Pantalla de creación de eventos de Timpik – Configuración. Fuente: Timpik 21
Figura 14: Pantalla de creación de eventos de Timpik – Configuración avanzada. Fuente: Timpik 21
Figura 15: Página de eventos y filtros. Fuente: (Elaboración propia) 29
Figura 16: Perfil de grupos. Fuente: (Elaboración propia) 30
Figura 17: Menú de creación de evento. Fuente: (Elaboración propia) 30
Figura 18: Menú de creación de evento – Listado de ubicaciones guardadas. Fuente: (Elaboración propia) 31
Figura 19: Menú de creación de evento – Selección de ubicación. Fuente: (Elaboración propia) 31
Figura 20: Menú de creación de evento – Guardado de ubicación. Fuente: (Elaboración propia) 31
Figura 21: Flujos de interacción de Buzzer Beater. Fuente: (Elaboración propia) 31
Figura 22: Átomo: Card de evento. Fuente: (Elaboración propia) 32
Figura 23: Átomo: campo de formulario con mensaje de error. Fuente: (Elaboración propia) 32
Figura 24: Átomo: mensajes de error, información y éxito. Fuente: (Elaboración propia) 33
Figura 25: Página de eventos y filtrado lateral. Fuente: (Elaboración propia) 33
Figura 26: Perfil de grupo. Fuente: (Elaboración propia) 34
Figura 27: Menú de creación de evento. Fuente: (Elaboración propia) 34
Figura 28: Menú de creación de evento – Listado de ubicaciones guardadas. Fuente: (Elaboración propia) 35
Figura 29: Menú de creación de evento – Selección de ubicación. Fuente: (Elaboración propia) 35
Figura 30: Menú de creación de evento – Guardado de ubicación. Fuente: (Elaboración propia) 35
Figura 31: Flujos de interacción de Buzzer Beater. Fuente: (Elaboración propia) 36
Figura 32: Arquitectura de la aplicación web. Fuente: (Elaboración propia) 37
Figura 33: Contenido del directorio wp-content. Fuente: (Elaboración propia) 38
Figura 34: Definición de plantilla implícita en un tema de WordPress. Fuente: (Elaboración propia) 40
Figura 35: Jerarquía de asignación de plantillas según contenido de WordPress. Fuente: (WordPress, s.f.) 40
Figura 36: Control de versiones por ramas en Bitbucket con git. Fuente: (Elaboración propia) 44

Figura 37: Interfaz de LocalWP. Fuente: (Elaboración propia) 45
Figura 38: Información del tema en el fichero style.css. Fuente: (Elaboración propia) 46
Figura 39: Estructura de la base de datos de WordPress. Fuente: (WordPress, s.f.) 49
Figura 40: Interfaz de campos de ACF. Fuente: (Elaboración propia) 51
Figura 41: Ubicación de grupo de campos de ACF - Post type evento. Fuente: (Elaboración propia) 51
Figura 42: Ubicación de grupo de campos de ACF - Rol de usuario organización. Fuente: (Elaboración propia) 52
Figura 43: Endpoint para llamadas AJAX save_location.php. Fuente: (Elaboración propia) .. 53
Figura 44: Conjunto de llamadas AJAX del proyecto. Fuente: (Elaboración propia) 54
Figura 45: Interfaz de Cypress con la ejecución de una prueba finalizada y su vista previa. Fuente: (Elaboración propia) 56
Figura 46: Fichero de prueba de Cypress para el filtrado de eventos por deporte. Fuente: (Elaboración propia) 57
Figura 47: Conjunto de pruebas creadas en Cypress para el proyecto. Fuente: (Elaboración propia) 57
Figura 48:Fichero con datos de entrada de prueba filter_event.json 58
Figura 49: Listado y filtrado de eventos. Fuente: (Elaboración propia) 60
Figura 50: Detalles de evento. Fuente: (Elaboración propia) 60
Figura 51: Menú de creación de evento 1/2. Fuente: (Elaboración propia) 61
Figura 52: Menú de creación de evento 2/2. Fuente: (Elaboración propia) 61
Figura 53: Menú de creación de evento - Selección de ubicación. Fuente: (Elaboración propia) 62
Figura 54: Menú de creación de evento - Selección de ubicación - Mapa. Fuente: (Elaboración propia) 62
Figura 55: Desplegable de marcador en mapa de selección de ubicación. Fuente: (Elaboración propia) 63
Figura 56: Menú de guardado de ubicación. Fuente: (Elaboración propia) 63
Figura 57: Perfil de grupo. Fuente: (Elaboración propia) 63

1. Introducción

En este primer apartado del Trabajo Fin de Grado (TFG), se presenta el proyecto a realizar, cuáles han sido las principales motivaciones y se especificarán los objetivos a alcanzar. Además, se explicará la estructura del trabajo escogida.

1.1 Motivación

Según la Encuesta de Hábitos Deportivos en España, realizada por el Ministerio de Cultura y Deporte, a lo largo del pasado año 2021, la práctica de todo tipo de deportes ha visto un notable auge en los últimos 7 años. De hecho, desde 2015 hasta 2020, se ha visto un más que considerable aumento de 6,1 puntos porcentuales.

De acuerdo al estudio realizado por el Ministerio de Cultura y Deporte, a pesar de la pandemia de la COVID-19, a raíz de la cual se ha apreciado una leve modificación en os hábitos de la práctica deportiva, la tendencia de la práctica deportiva continua al alza. Cerca del 62,4% de los ciudadanos afirman haber mantenido sus costumbres deportivas, mientras que un 17,1% ha disminuido su actividad y el restante 12,9%, asegura haber incrementado la misma.

De acuerdo con la Encuesta de Hábitos Deportivos en España, cerca del 60% de la población española, mayor de 15 años, ha practicado deporte en 2020 frente al anterior dato que se tiene de 53,5% del año 2015, lo cual indica una propensión positiva salvando los meses de pandemia que afectaron al sondeo investigado (Ministerio de Cultura y Deporte, 2021).

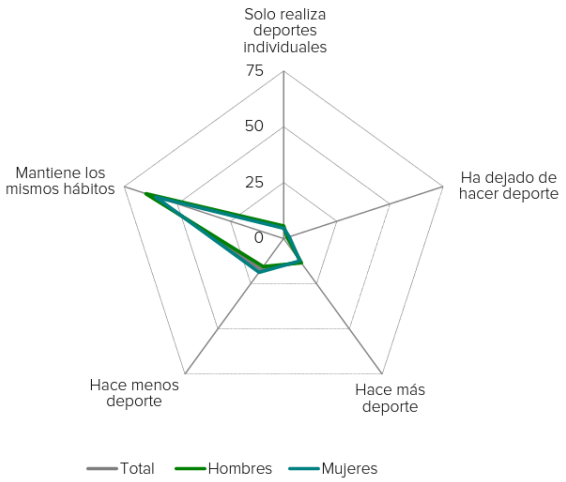


Figura 1: Variación de hábitos en la práctica deportiva tras la COVID-19. Fuente: (Ministerio de Cultura y Deporte, 2020)

Junto con la práctica del deporte, y de manera más notable, encontramos el creciente protagonismo de las redes sociales en el día a día. Estas han experimentado un crecimiento continuo, a nivel global, entre los años 2017 y 2021, alcanzando a ser utilizadas por el 58% de personas en el mundo (ver Figura 1). Actualmente, el 92,7% de la población española que utiliza Internet (cerca del 90%), tiene presencia en una o varias redes sociales (Una vida online, 2021).

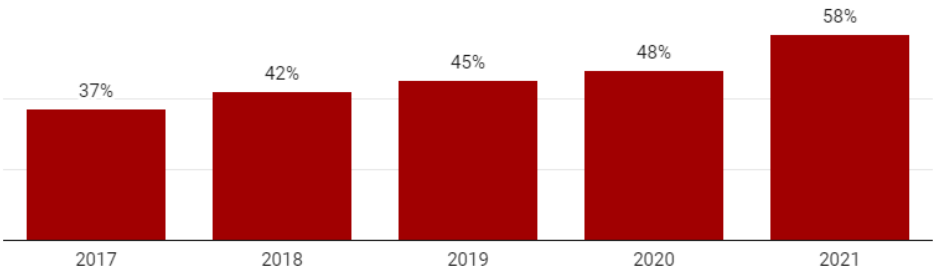


Figura 2: Evolución de los usuarios de redes sociales en el mundo de 2015 a 2020. Fuente: (Una vida online, 2021)

Es importante destacar que, en ambos ámbitos, en el uso de redes sociales y en el de la práctica deportiva, coinciden las franjas de edad que presentan mayor actividad. En cuanto a la práctica del deporte, se puede destacar un mayor hábito en edades entre los 15 y los 44 años que comprende niveles de práctica deportiva de entre el 70-80% (ver Figura 3). Este va seguido de un descenso de un 10% en la franja contigua, sucedida por una caída aún mayor para edades superiores a los 55 años (Ministerio de Cultura y Deporte, 2020).

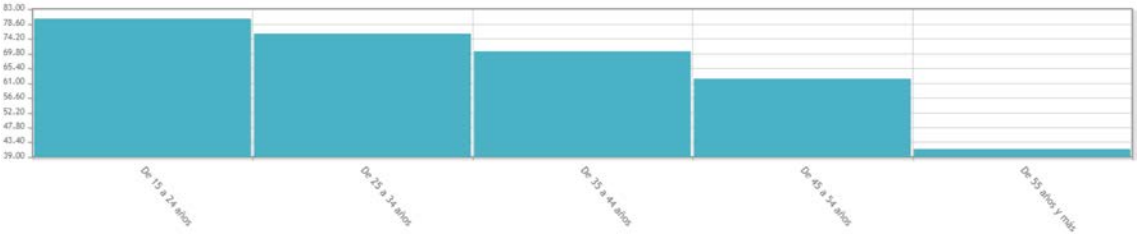


Figura 3: Porcentaje de práctica deportiva por edades en el último año en España. Fuente: (Ministerio de Cultura y Deporte, 2020)

De manera similar a la práctica deportiva, las redes sociales están más presentes en la vida de personas de mayor juventud. Las edades de entre 13 y 44 años albergan prácticamente la mitad de los usuarios en redes sociales en España con un 49,5% de la cuota (Una vida online, 2021).

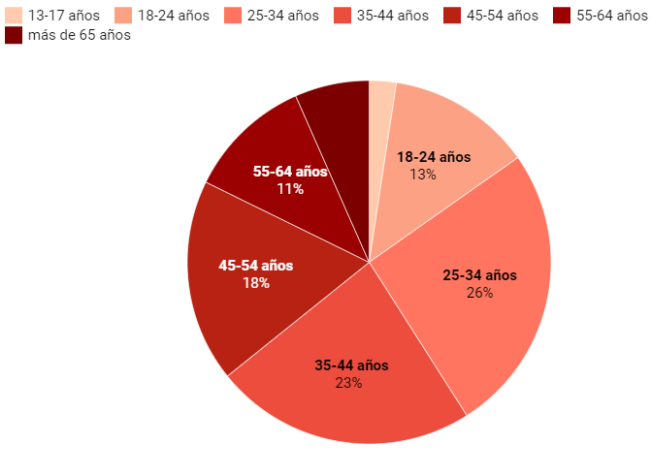


Figura 4: Edad de los usuarios de redes sociales en España. Fuente: (Una vida online, 2021)

Tomando en consideración los datos expuestos, se puede distinguir una oportunidad de relacionar ambos ámbitos ya que comparten un mismo público objetivo y desarrollar una solución elegante para la socialización por medio de la práctica deportiva. A lo largo de este TFG, se va a estudiar, planificar y desarrollar las bases de una red social que hemos llamado Buzzer Beater orientada a la práctica de deporte colectiva.

Por último, estas tendencias ponen aún más de manifiesto la necesidad de desarrollar esta aplicación ya que fomentaría varios de los Objetivos de Desarrollo Sostenible acordados por la Asamblea General de las Naciones Unidas en 2015. Buzzer Beater se alinea perfectamente con el objetivo de salud y bienestar (ODS 3) al promover la práctica deportiva y con el objetivo de alianza para lograr objetivos (ODS 17) por el fomento de la creación de grupos para la práctica colectiva de deporte.

1.2 Objetivos

Para poder elaborar una propuesta en detalle, se analizan las necesidades presentes y se determinan los siguientes objetivos:

1. Desarrollar y desplegar un portal de creación de eventos totalmente personalizables en una aplicación web accesible, ligera, rápida y segura.
2. Permitir la búsqueda y filtrado de eventos por diferentes criterios como el deporte en cuestión, la experiencia de los participantes, el sexo y el número de participantes.
3. Permitir la creación de grupos para facilitar la reunión de miembros y agilizar tanto la creación de eventos como la comunicación entre los mismos.

1.3 Estructura

En el presente documento se han estructurado los distintos los distintos apartados que se han desarrollado en este TFG. Estos son los siguientes:

1. **Introducción:** Para comenzar, se comentan las motivaciones de emprender este proyecto analizando algunos indicadores que respaldan la factibilidad de la idea. Se detallan los objetivos a alcanzar y, por último, unos detalles sobre cómo se ha desarrollado la colaboración con el compañero con el que se ha desarrollado la totalidad de la aplicación web y las necesidades o adaptaciones que ha acarreado.
2. **Metodología:** En este apartado, se detallará la metodología escogida para el desarrollo del proyecto y se argumentará con motivos relacionados con el tipo de proyecto y su magnitud.
3. **Estado del arte:** Se analizarán las actuales tecnologías disponibles para el desarrollo de una aplicación web, junto con las soluciones actualmente disponibles para los objetivos planteados.
4. **Análisis del problema:** Van a concretarse las necesidades específicas a satisfacer y cuáles serán los posibles perfiles interesados y cómo se plasmarán en el producto final.
5. **Diseño de la solución:** En esta etapa, se procederá a explicar cuál será la arquitectura utilizada para el proyecto, la estructura interna de la misma junto con una especificación de la estructura de datos que se utilizará. Se mostrará el proceso de prototipado y diseño preliminar de la solución propuesta.
6. **Desarrollo de la solución propuesta:** Se explicará en detalle algunas de las características técnicas más relevantes durante el desarrollo de esta. Además, se

relacionará con cómo ha debido de segmentarse y adatar el desarrollo al hecho de ser un proyecto colaborativo, pero con partes diferenciadas.

7. **Pruebas:** En este crucial apartado, se identificarán cada una de las posibles casuísticas posibles y se generarán diferentes pruebas para asegurar el correcto funcionamiento bajo cualquier situación.
8. **Implantación:** Tras el desarrollo del mismo, se darán detalles de los procesos finales de despliegue y otras consideraciones finales junto con la presentación del resultado final y las diferentes características implementadas.
9. **Conclusiones:** En esta sección, se hará una evaluación del proyecto, se relacionará con los estudios universitarios y se propondrán posibles mejoras o trabajos futuros sobre el proyecto.
10. **Bibliografía:** Finalmente se incluye la lista de referencias utilizadas.

1.4 Colaboraciones

El actual proyecto, es complementado con el TFG de Vicente Tormo Torres titulado "*Diseño y desarrollo de funcionalidades generales de Buzzer Beater, una red social para la práctica deportiva amateur*". Ambos TFGs se complementan para dar lugar a una aplicación de mayor funcionalidad. En particular, este TFG es complementado con las funcionalidades de una red social convencional que permite la interacción de los usuarios de la misma, siendo siempre, enfocado como tema principal, la práctica de deporte y personalizada para poder ofrecer una experiencia única.

Para llevar a cabo esta colaboración, hemos requerido una coordinación total y una división del trabajo muy marcada desde la concepción de la idea. Esto ha sido posible gracias a la utilización de herramientas colaborativas de prototipado y diseño como Uizard y Figma, de herramientas de gestión de las tareas y los entregables del proyecto como Trello y de herramientas para el control de versiones del código como Bitbucket que ha permitido un desarrollo paralelo, complementario y orgánico.

Vicente en su TFG aborda la creación de publicaciones de usuarios y su correspondiente cronología, los mensajes directos o "Buzzers" y los perfiles de usuario. En cambio, en este TFG se aborda la creación y gestión de eventos, el buscado y filtrado de los mismos, acorde a diversos parámetros y la creación de grupos de usuarios.

Ambos TFGs alimentan un mismo proyecto por lo que se ha tratado de aislar las diferentes partes al máximo, pero ha requerido de una buena sincronización y planificación para no obstaculizar nuestros trabajos por la existencia de dependencias circulares entre los mismos.

2. Metodología

En este apartado, se introduce la metodología utilizada para el desarrollo del proyecto.

En concreto, se ha optado por seguir el marco de SCRUM por ser una metodología ágil cuyo principio es la mejora continua del trabajo a realizar.

SCRUM propone una primera fase de consulta con los interesados o clientes, identificación de funcionalidades o núcleos que servirán para planificar los distintos *sprints*. Un *sprint* es un intervalo de tiempo para el cual se han establecido un cierto número de tareas a realizar. Esta primera fase viene seguida de la ejecución de los diferentes *sprints*. Cada sprint está estructurado en diferentes tareas. Estas son planificadas al principio del mismo y revisadas por medio de reuniones periódicas para poner en común las diferentes herramientas y cambios en los objetivos iniciales. Se prosigue con la ejecución de las diferentes tareas para alcanzar el incremento final del producto que ha supuesto el *sprint*. Este ciclo se repite con sus respectivas pruebas hasta que se alcanza una entrega de producto potencialmente entregable tras los incrementos (Drumond, 2021).

2.1 Planificación e implementación de SCRUM

SCRUM nos permite una monitorización constante de nuestro trabajo para continuar en perfecta armonía y salvar las dependencias que han ido surgiendo a lo largo del desarrollo del proyecto.

Antes del comienzo de la fase de desarrollo, hubo que tomar decisiones, crear maquetas y fijar diseños. En un principio, tras una lluvia de ideas inicial y la concreción de la idea a ejecutar, se comenzó con una fase de prototipado de las diferentes disposiciones de las páginas. Tras esta fase, se prosiguió con el diseño modular de los diferentes componentes clave como las *cards* de eventos, los mensajes de error y notificación, los diferentes filtros, menús y otras piezas clave. A continuación, se procedió a juntar todos estos elementos, también conocidos como átomos, en diversas propuestas de disposición.

Una vez se tuvo el diseño final, dio comienzo la fase de desarrollo que vendría a ser la más duradera y compleja. Al haberse creado una pila de tareas muy bien acotadas, se pudieron ordenar y agrupar en *sprints* siguiendo una lógica a nivel de desarrollo. Esta metodología, se vio reflejada en la herramienta Trello por medio de la creación de tarjetas y listas (ver Figura 5). A nivel de lista, distinguimos 7. Estas son “Sprints”, “Por hacer”, “Haciendo”, “Hechas”, ”Bloqueadas”, “Incrementos” y “Cerradas” las cuales podemos clasificar según el significado de sus tarjetas; tanto “Sprints” como “Incrementos”, representan listas que contienen *sprints*. Estos son representados por medio de tarjetas con cabeceras de colores y contienen un listado con las tareas que pertenecen a dicho *sprint*. Por otra parte, el resto de listas contienen tarjetas que representan tareas. Las listas “Por hacer”, “Haciendo”, “Hechas” y ”Bloqueadas”, representan el estado de las tareas dentro del *sprint* que se está desarrollando en el momento. La lista de “Cerradas”, desempeña la función de historial de tareas de todos los *sprints* ya finalizados. El código de color de las tareas del *sprint* actual es denotar a quién pertenece cada tarea. El color azul es para las tareas de mi compañero y el naranja para las mías. Las tarjetas de *sprints* no sigue ningún código de color para su encabezado, simplemente son para resaltarlas más frente a las tareas que las componen.

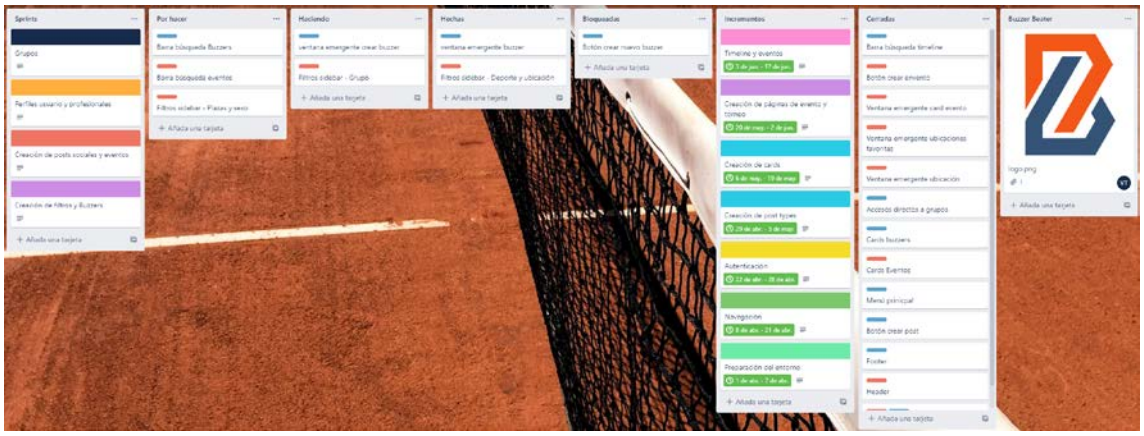


Figura 5: Disposición del tablero en Trello del TFG. Fuente: (Elaboración propia)

Además, Trello nos permite establecer plazos para cada tarjeta. Esta característica junto con un *Power-Up* o extensión llamado "Placker", hemos podido trasladar esta representación, de listas, a un diagrama de Gantt con dependencias, para poder tener una mejor y más general visión, del progreso del proyecto (ver Figura 6).

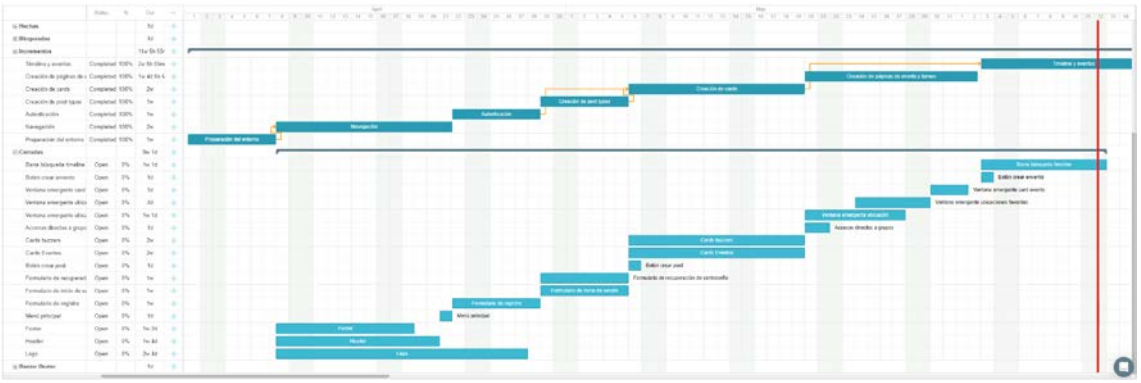


Figura 6: Diagrama de Gantt de los sprints y sus tareas. Fuente: (Elaboración propia)

Al inicio de cada *sprint*, tuvimos una reunión telemática por medio de Discord. En estas nos planificábamos y exponíamos los objetivos a lograr al final del *sprint* y comentábamos diferentes mejoras u optimizaciones que íbamos descubriendo para optimizar, tanto el código, como el tiempo empleado en las tareas. Estas podían ir desde buenas prácticas para la estructuración del proyecto, hasta descubrimiento de librerías o funciones útiles. Además, si el *sprint* se alargaba temporalmente, más de una semana, dependiendo de la carga de trabajo, hacíamos alguna reunión para desbloquear tareas o comunicarnos el progreso. Uno de los mayores retos y, por ende, uno de los temas más recurrentes en las reuniones, fue el tratamiento y guardado de datos. Para poder alcanzar el nivel de personalización necesario para el correcto funcionamiento de la aplicación e interconexión de todos los elementos, tuvimos que establecer unas plantillas, por lo general, en formato JSON, para que no hubiera discordancias en los datos y luego hubiera conexión entre los dos trabajos. Este problema se debe principalmente al trabajo individual en la copia que tenemos cada uno del incremento más reciente y a la decisión que se tomó de mantener la estructura de base de datos de WordPress intacta para evitar posibles problemas futuros.

Cada uno ha trabajado en una copia local en su máquina y ha desarrollado su parte correspondiente, con su base de datos, también local. Esta discrepancia entre bases de datos, se

pudo solucionar al establecer unas plantillas para los formatos de los diferentes tipos de datos. Sin embargo, el problema de tener visiones distintas, no nos permitía plantear, durante el desarrollo, los suficientes casos de prueba como para poder detectar el mayor número de errores. Hecho que se ponía de manifiesto, al terminar un *sprint* y actualizar el incremento local de cada uno con el progreso del otro. Esos fueron los motivos que, en última instancia, nos llevaron a unificar la base de datos y así poder depurar mejor la aplicación.

Figura 7: Esquema de la metodología SCRUM. Fuente: (Sinnaps, 2022)

Una vez finalizado el desarrollo del proyecto, se planteó la generación de test por medio de la herramienta Cypress, que automatiza este proceso. Permite la creación de sucesiones de acciones que, después, son lanzadas de formar automatizada y variando los valores de entrada para tratar de forzar el error de la aplicación.

Por último, se decidió llevar a cabo el despliegue del proyecto conjunto finalizado en un servidor real al cual apunta al subdominio http://buzzerbeater.millandavid.com, ambos contratados expresamente para la realización de este proyecto y para futuras pruebas en un entorno real de proyectos propios. Este proceso se llevó a cabo por medio de la creación y población inicial de una base de datos y la posterior instalación de los *plugins* utilizados en WordPress, junto con el tema desarrollado. Para ello, se dispuso del *plugin* Duplicator, el cual nos permitió no tener que partir de una instalación limpia de WordPress, sino exportar toda la configuración que ya habíamos realizado en nuestras copias locales.

2.2 Herramientas de planificación

Es imprescindible una buena planificación en un proyecto de estas dimensiones y, más aún, teniendo en cuenta el aspecto colaborativo. Para ello comenzamos con el análisis de los requisitos y objetivos planteados y su posterior división lógica en diferentes *sprints*. Para ello, nos hemos servido de diversos programas y herramientas:

- **Trello**: Es una herramienta de gestión de proyectos basada en el sistema japonés *kanban*. Este sistema se centra en la creación de tarjetas o carteles que identifican los productos, cantidades y tiempos necesarios de los diferentes componentes necesarios para la fabricación de un producto. Trello aplica este sistema en forma de un tablero dividido en listas, las cuales contienen tarjetas. Estas tarjetas pueden contener todo tipo de información desde estimaciones de tiempo, plazos, listas, enlaces, títulos, imágenes y

muchas otras características. Además, estas pueden ser expandidas por medio de *Power-Ups* (Trello, s.f.).
Nos hemos servido de esta herramienta para poder desarrollar el proyecto siguiendo una metodología SCRUM. En concreto, hemos identificado en diferentes listas, los diferentes estados de las tareas del *sprint* que se estaba desarrollando en cada momento. El resto de *sprints* planificados y los incrementos finalizados, se almacenan en sus propias listas. Además, ha sido de gran utilidad poder implementar por medio de un *Power-Up*, la generación, a partir de los plazos de las tareas, de un diagrama de Gantt.

- **Google Workspace:** El gigante tecnológico Google, proporciona herramientas con infinidad de posibilidades e interconectadas entre sí, ofreciéndonos un entorno muy cómodo para desarrollar este proyecto. Particularmente, hemos utilizado el editor de documentos en línea Google Docs para la planificación más detallada de los diferentes *sprints*, el editor de hojas de cálculo en línea Google Sheets para la realización de lluvias de ideas y la estructuración de las mismas y el almacenamiento en la nube de Google Drive para almacenar todo tipo de documentos, hojas de cálculo y recursos como imágenes u otros ficheros de interés.

- **Discord:** Como herramienta de comunicación en línea, esta aplicación ofrece llamadas de video y voz de gran calidad. Discord nos ha permitido llevar a cabo todas nuestras reuniones virtuales, mostrar nuestro progreso y resolver dudas gracias a funcionalidades como la de compartir pantalla.

2.3 Herramientas de diseño

El paso previo al desarrollo es la creación de prototipos y su posterior concreción en diseños alineados con la experiencia de usuario que se quiere alcanzar. Estos han sido los siguiente:

- **Uizard**: Se trata de la herramienta de prototipado y *wireframing* colaborativa utilizada para diseñar todo tipo de interfaces con facilidad y rapidez. Además, dispone de herramientas para crear demostraciones interactivas para demostraciones y test A/B. Gracias a la utilización de Uizard, hemos podido crear un prototipo con *wireframes* en poco tiempo sin entrar en detalles y poder coordinarnos entre nosotros para decidir sobre la navegación de la aplicación web. Su carácter colaborativo permite crear los prototipos en tiempo real con una interfaz que permite ver, en todo momento, al resto de tu equipo y los cambios que estos realizan.

- **Figma:** Consiste en una potente herramienta de prototipado y diseño de interfaces. Es una aplicación web centrada en la colaboración, permitiendo la edición simultánea, en tiempo real, por parte de todos los miembros de un equipo. Además, destaca su "*desgin system*" el cual han concebido como un sistema que permite la creación de átomos y su almacenamiento en librerías. Estos átomos son personalizables y están sincronizados todos bajo una sola especificación. Con este método, Figma proporciona una coherencia y una rápida conexión de todos los elementos que, permite cambiar toda la apariencia de los diseños en pocos minutos (Figma, s.f.).
Una vez hemos dispuesto de los prototipos, hemos utilizado Figma para la creación final de los diseños con total precisión. Hemos podido crear, una librería con los colores corporativos pensados para Buzzer Beater y un conjunto de átomos que, nos han permitido la rápida actualización de las diferentes vistas de la aplicación web, para agilizar el proceso de diseño creativo.

2.4 Herramientas de desarrollo

A lo largo del desarrollo de la aplicación web, se han utilizado diversas herramientas que han permitido el buen desarrollo del proyecto en paralelo con Vicente. Estas son las siguientes:

- **Visual Studio Code:** Se trata del editor de código gratuito, de código abierto u *open source* y multiplataforma de Microsoft. Supone una gran ventaja sobre otros editores de código por la gran variedad de extensiones que ofrece y las compatibilidades entre estas que, en conjunto, permiten personalizar la experiencia de desarrollo al máximo para la mayor comodidad del desarrollador. Además, al tratarse de una aplicación web, se ha decidido utilizar este editor por encima de un entorno integrado de desarrollo o *IDE* en inglés por su carácter liviano o *lightweight*. Su perfecta integración con git, el resaltado de código o las herramientas que ofrece para depurar código, lo hacen una herramienta aún más idónea para la tarea en cuestión.

- **XAMPP:** Es un paquete de diferentes softwares necesarios para poder lanzar una página o aplicación web en una máquina de forma local. Está compuesto, como sus iniciales indican, por un servidor web Apache pre configurado, un gestor de base de datos basado en MariaDB o MySQL, e intérpretes para PHP y Perl. La X indica que es la versión multiplataforma que funciona en máquinas con sistemas operativos Windows, MacOS y Linux. Con esta herramienta se comenzó el desarrollo, pero se decidió cambiar a otra más conveniente llamada LocalWP.

- **LocalWP:** Se decidió hacer la transición a LocalWP por su especialización en aplicaciones webs basadas en WordPress. Al igual que XAMPP, es una distribución de Apache con características pre configuradas para el manejo de bases de datos y soporte para PHP. Además, una de las características más atractivas frente a su antecesor, es la posibilidad de enviar y capturar mensajes por medio del protocolo SMTP que, de utilizar otro software como XAMPP, habría que pasar por una configuración previa más extensa. En adición a esto, LocalWP también admite el cambio de versión de PHP y WordPress de manera nativa y sencilla.

- **Bitbucket:** Para mantener un control de versiones por medio del sistema de control de versiones o *VCS* en inglés, git. Al pertenecer a la familia de aplicaciones de Atlassian, mantiene una relación estrecha, tanto en interfaz como en características, con Trello. Gracias a esta, hemos podido mantener el código almacenado en un repositorio privado y accesible desde cualquier dispositivo y diferenciado por ramas y versiones de estas para poder deshacer cambios erróneos o compara cambios. Además, las opciones que permite de visualización de ramas, han facilitado mucho la comprensión de la evolución del proyecto.

- **FileZilla:** Para poder acceder a los ficheros, del subdominio que se tiene contratado en un servidor compartido del proveedor de servicios Hostinger, la mejor solución siempre la ofrecen las aplicaciones que permiten el acceso por medio del protocolo de transferencia de ficheros FTP a los mismos. Gracia a FileZilla, se ha podido crear varias conexiones a diferentes subdominios que se prepararon para emular el desarrollo de un proyecto real. Para la descarga y subido de ficheros al servidor, Filezilla proporciona una interfaz muy amigable y similar al explorador de archivos de Windows con una cola de transferencias para poder llevar un control de las acciones realizadas y el tiempo estimado en completarlas.

- **Chrome DevTools:** Se trata de un conjunto de herramientas para desarrolladores incluidas de manera nativa en el buscador Google Chrome. Estas nos han permitido analizar los problemas en las solicitudes al servidor con el envío de formularios, depurar errores en scripts o incluso modificar de manera rápida la estructura y los estilos de las diferentes páginas de nuestra aplicación web para después trasladarlas de forma permanente al repositorio.

2.5 Herramientas de depuración

Con la finalización de la fase de desarrollo, es imprescindible comprobar cualquier posible fallo en los diferentes casos de uso, en la usabilidad y cualquier problema que pudiera surgir referente al rendimiento y velocidad de la aplicación web. Para ello se han utilizado las siguientes herramientas:

- **Chrome DevTools:** La misma herramienta de Google que se ha utilizado como apoyo para el desarrollo, permite la visualización de estadísticas muy relevantes sobre la velocidad de carga, el peso de los recursos que se utilizan y su porcentaje de uso. Otro punto de interés son las mejores posibles a realizar a raíz del análisis en materia de optimización en motores de búsqueda o *SEO* en inglés, que realiza.

- **Cypress:** Como herramienta principal para la realización de test, se ha elegido este *framework* basado en NodeJS para la escritura y ejecución de test y su posterior análisis. Cypress permite la personalización de test por código junto con la creación de los mismos en un entorno visual. Ha permitido ejecutarlos y poder navegar a través del tiempo para identificar los componentes concretos que fallaban en la aplicación.

3. Estado del arte

A pesar de la amplia variedad de redes sociales que coexisten a día de hoy, aquellas enfocadas en un sector concreto o de mayor especificidad, siguen siendo una minoría. En este apartado, se van a analizar las diferentes características de lo que son los principales competidores de Buzzer Beater y se expondrá la propuesta sugerida.

3.1 Identificación de aplicaciones similares y sus funcionalidades

La ventaja del enfoque minoritario de este proyecto, reside en la escasa y bien definida competencia donde actualmente destacamos solo cuatro aplicaciones (Timpik, IF7Sports, PadelTrack y Fubles). De estas cuatro, solo las dos primeras mencionadas mantienen un desarrollo continuo y tienen características lo suficientemente completas como para ser puestas en consideración (Rubio, 2019).

A continuación, se propone un análisis de las tres características principales o funciones ofrecidas por estas dos aplicaciones seleccionadas, Timpik e IF7Sports, las cuales incluyen:

- Listado y filtrado de eventos
- Perfil de grupo
- Creación de eventos

3.1.1 Listado y filtrado de eventos

Dada la naturaleza de la aplicación, es muy conveniente disponer de un panel en el que se muestren eventos deportivos relevantes con información concisa y útil. Se puede ver como ambas aplicaciones apuestan por unas tarjetas o *cards* de eventos muy simplistas, albergando solo información acerca horario del evento, el precio del mismo si lo requiere, información sobre el aforo del evento y un título junto con la ubicación para poder identificarlo. Además, las dos aplicaciones, paginan los eventos por día y permiten filtrarlos por ubicación.

Un par de detalles discordes entre una y otra, es la iconografía personalizada que ofrece Timpik sobre el deporte en cuestión y su nivel. Esta diferencia se debe principalmente al alcance de estas ya que, Timpik contempla una gran variedad de deportes, mientras que IF7Sports, se centra principalmente en las variantes del fútbol. Otra diferencia también debida a ello es el filtrado extra de eventos que ofrece Timpik, permitiendo elegir entre todos los deportes, los deportes que tengamos guardados, los de una organización concreta o los creados por uno mismo. Por último, dispone de un mapa en el lateral izquierdo que muestra las ubicaciones de los eventos listados para poder ubicarlos de forma más visual. En cuanto a características que tiene propias IF7Sports, se aprecia una vista de pestañas que agrupa los eventos según se hayan o no inscrito y de su proximidad temporal (ver Figura 8 y Figura 9).

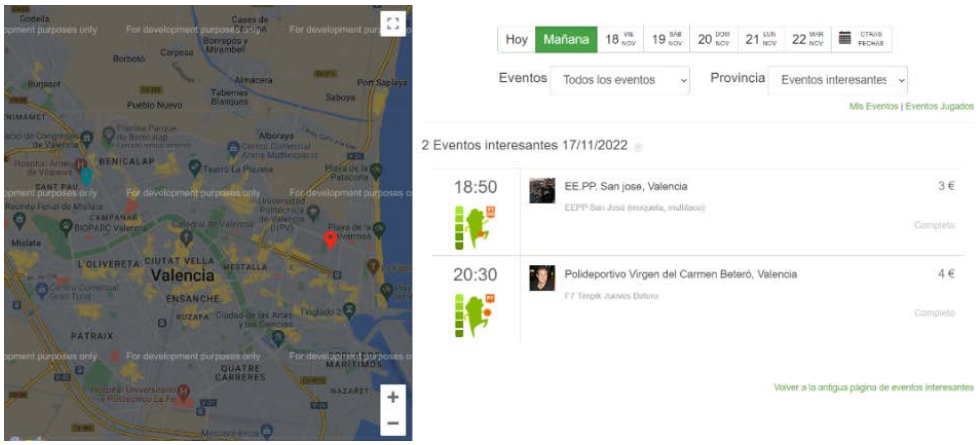


Figura 8: Página de eventos de Timpik. Fuente: Timpik

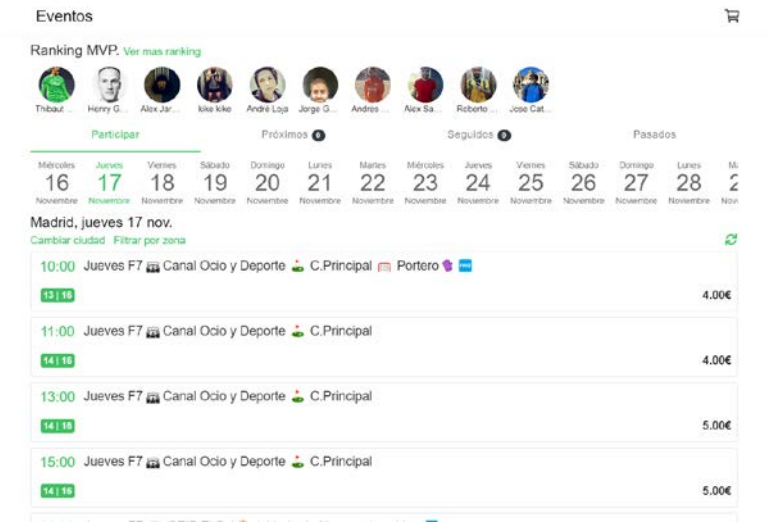


Figura 9: Página de eventos de IF7Sports. Fuente: IF7Sports

3.1.2 Perfil de grupo

Otra característica propia de las aplicaciones deportivas son los grupos o clubes. Estos permiten la agrupación de usuarios afines o con intereses similares para la mejor organización de eventos. Un claro ejemplo de esta característica, se puede apreciar en Timpik con los llamados “clubs”. Están compuestos, principalmente, de un muro de mensajes y un listado de próximos eventos en el panel central. Además, cuentan con un menú de navegación lateral para acceder a paneles que albergan solo una de estas características recién mencionadas, un listado de miembros y una página de administración del grupo. Como información general, disponen de una cabecera personalizable con un icono y nombre de grupo, sucedido por un listado de deportes habituales (ver Figura 10). En el caso de IF7Sports, no existe la posibilidad de crear o unirse a grupos o clubes.

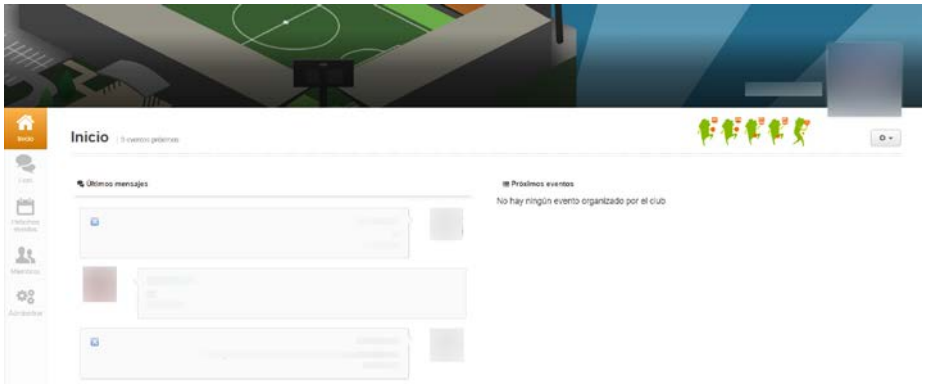


Figura 10: Perfil de grupo o club de Timpik. Fuente: Timpik

3.1.3 Creación de eventos

Como eje central del proyecto, es de gran interés analizar las diferentes propuestas en las interfaces de creación de eventos que proponen las diferentes aplicaciones ya existentes para evitar crear un proceso complejo o tedioso y que conlleve la perdida de usuarios.

Timpik ofrece una interfaz dividida en diferentes pasos o etapas. En la primera se escoge el nombre del evento y el tipo de deporte a realizar (ver Figura 11).

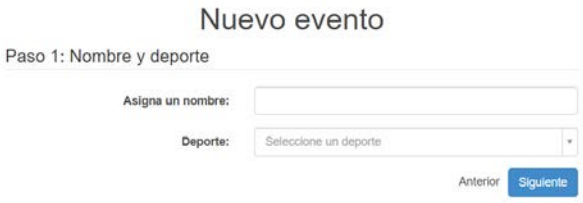


Figura 11: Pantalla de creación de eventos de Timpik – Nombre y deporte. Fuente: Timpik

A continuación, se pide introducir otros detalles referentes a la ubicación y la fecha (ver Figura 12).

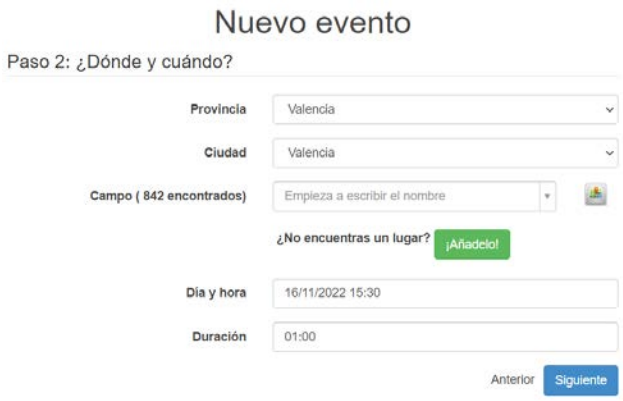


Figura 12: Pantalla de creación de eventos de Timpik – Ubicación y fecha. Fuente: Timpik

El último paso consiste en la concreción del número máximo de jugadores, el coste y el club que lo organiza si fuera necesario (ver Figura 13).

Figura 13: Pantalla de creación de eventos de Timpik – Configuración. Fuente: Timpik

Por último, si se decide tener un mayor nivel de control sobre el mismo, en las opciones avanzadas, se disponen de diversos ajustes como la visibilidad, el sexo o el nivel entre otros (ver Figura 14).

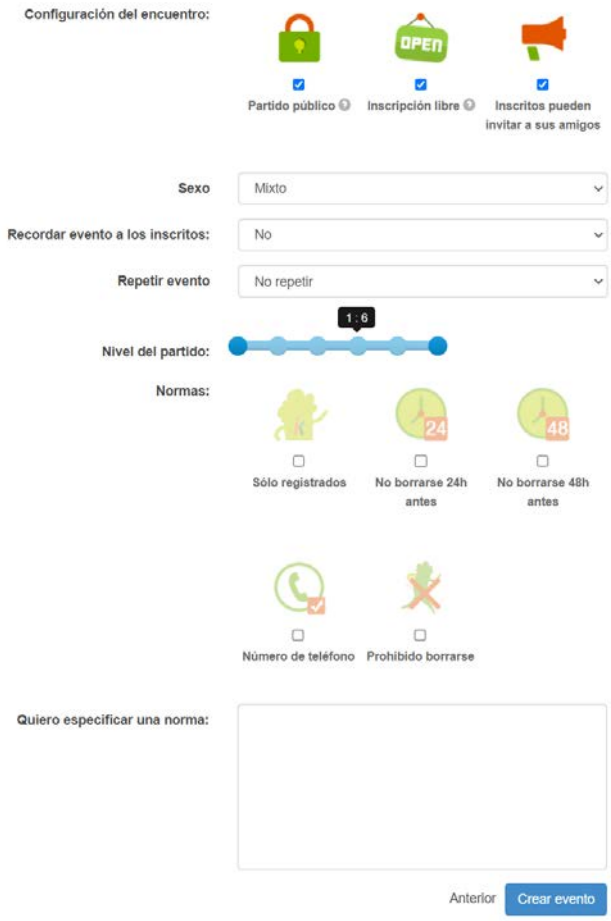


Figura 14: Pantalla de creación de eventos de Timpik – Configuración avanzada. Fuente: Timpik

3.2 Análisis crítico del estado del arte

Tras la identificación de las principales características específicas de las redes sociales deportivas más extendidas, se procede a una crítica con el propósito de alcanzar una mejor propuesta.

3.2.1 Listado y filtrado de eventos

La idea base de estructurar los eventos en forma de tarjetas con información relevante pero escueta, es una solución elegante y directa, fácilmente comprensible por cualquier usuario y que proporciona suficiente claridad en un intervalo muy corto de tiempo. La paginación sin embargo no se considera la mejor práctica ya que obliga a cambiar de pestaña para poder ver todos los eventos disponibles, además, el filtrado de eventos es pobre ya que un evento contiene muchos parámetros relevantes para un usuario (ver Figura 8 y Figura 9). Ni Timpik ni IF7Sports, han demostrado este nivel de precisión en la búsqueda de un evento.

3.2.2 Perfil de grupo

Timpik tiene una buena concepción del concepto de un grupo, pero no acaba de explotarla al máximo. En el panel principal se muestran mensajes recientes y eventos próximos, lo cual es muy adecuado al mostrar la información más relevante en un principio. Sin embargo, el menú de navegación lateral, además de romper con la experiencia global del menú horizontal principal, muestra 2 secciones cuyo único propósito es hacer que estos 2 muros de información, ocupen un mayor espacio en la pantalla (ver Figura 10).

3.2.3 Creación de eventos

En cuanto a la creación de eventos, este debe ser un proceso que resulte liviano, rápido y claro, pero a la vez, que permita una gran personalización. Timpik consigue un proceso rápido y claro pero presenta una escasez de claridad y personalización en algunos niveles. La claridad no la alcanza por la experiencia de usuario que supone no ver todos los campos a rellenar en un instante inicial. Los formularios por pasos son una buena opción para agrupar tipos de información o seccionar cuestionarios largos. Un formulario en el que se es consciente de la cantidad de pasos restantes, convendría en el caso de que se requiriesen muchos campos. En el caso contrario, si los requerimientos son reducidos, conviene un formulario sin pasos que muestre, de manera escueta, todos los requerimientos a nivel de información de usuario (Joanne, 2016). En este caso, el extremo aislamiento inicial de 2 campos (ver Figura 11) seguido de un siguiente paso con bastantes más datos (ver Figura 12 y Figura 13), pueden llevar a confusión por la falta de información sobre los pasos restantes o el progreso del formulario. Por último, destacar la falta de personalización en campos como la selección de ubicación del evento. Este campo, además de contener muchas opciones duplicadas, dispone de una opción para añadir una localización nueva que, al seleccionarla, se nos redirige a una página sin maquetar con opciones de selección de ubicación obsoletas.

Por otra parte, IF7Sports, no cuenta con la opción de creación de eventos. Estos eventos solo pueden ser creados por administradores oficiales.

3.2.4 Conclusiones al análisis de la competencia

Como conclusión, se puede afirmar que, a pesar de que, el principal competidor, Timpik, cuenta con un amplio abanico de características, estas no han sido resueltas de la mejor forma posible. Por otra parte, la segunda aplicación con mayor renombre, IF7Sports, carece de características esenciales como la creación de grupos y de eventos. De hecho, las críticas más notables a la plataforma en las reseñas de Google, provienen de estas, más que notables, carencias.

3.3 Propuesta

Tras analizar las diferentes propuestas ya existentes en el mercado, se presenta un plan que abarca las diferentes áreas cubiertas en los anteriores subapartados y afrontar las debilidades encontradas.

3.3.1 Listado y filtrado de eventos

Para resaltar el eje central de la aplicación web, la página de eventos será la página principal de la misma. Esta seguirá la buena práctica de las tarjetas o *cards* para mostrar con facilidad los diferentes eventos disponibles en función del filtrado aplicado, las cuales, aparecerán en una disposición vertical, ordenadas por relevancia, de arriba a abajo. Estas, han de contener, al menos, un título, un icono representativo del deporte en cuestión, un indicativo del número de plazas ocupadas y disponibles, la ubicación del evento, el organizador del mismo, si están limitadas a un grupo y su visibilidad.

Otra característica de vital importancia es el filtrado de eventos. Se ha optado por un filtrado que evite la paginación por días próximos para poder crear un desplegable con un calendario en miniatura para una mayor flexibilidad de fechas. Por otra parte, se da opción al filtrado por uno o varios deportes simultáneamente con un selector de tipo *dropdown* combinado con tarjetas o *cards* de deporte. Otros parámetros interesantes para el filtrado serían la cantidad de jugadores, el sexo de los participantes, la ubicación, si pertenece a un grupo y si se está buscando un evento competitivo o no. Por último, se considera la opción de añadir la opción de hacer todos los filtros requeridos de forma conjunta para búsquedas muy concretas, o requeridos de forma individual, para un mayor abanico de resultados. La primera opción, implicaría la des habilitación de los selectores múltiples, como el de deporte, ya que no podría plantearse un evento que englobe más de un deporte simultáneamente, o el de grupos. Todas estas opciones, se van a presentar en una interfaz lateral vertical para ser diferenciadas de la navegación general de la aplicación.

3.3.2 Perfil de grupo

A la hora de plantear un perfil de grupo, no se cree necesaria la creación de una navegación interna. Los grupos están enfocados en reunir usuarios, eventos y comunicarlos entre sí. Por estos motivos, se propone una visualización de los próximos eventos en un bloque, paginados de dos en dos, dentro de un contenedor de diapositivas o carrusel. Estos, además, dispondrán de unos botones para poder abrir una ventana nueva o desplegable, una vista con un mayor número de eventos.

Análogamente, con los mensajes más recientes internos, se plantea una visualización, previa, limitada a una alto concreto que permita un desplazamiento vertical, con opción a la apertura de este chat grupal en una ventana nueva o desplegable.

Se mantendría la idea de una sección superior con información sobre el grupo como la cantidad de participantes y un acceso rápido a los perfiles de estos, los deportes destacados del grupo, una imagen del grupo y el nombre del mismo.

Los administradores del grupo, además, tendrían posibilidades adicionales de edición de datos del grupo, de promoción o expulsión de usuarios, la opción de abrir o cerrar el acceso al grupo y la de aprobación de eventos, mensajes y usuarios, si así lo desean. Estos también tendrían acceso a estadísticas propias del grupo como porcentajes de participación, nivel medio de los eventos y otras métricas relacionadas con la actividad dentro de este.

3.3.3 Creación de eventos

Como principal objetivo, se desea simplificar al máximo el proceso de creación de eventos y mantener un alto nivel de personalización. En primer lugar, se pretende unificar en una sola ventana o vista emergente, toda la información necesaria para la creación de un evento. Los campos cubrirán el nombre del evento, la fecha y hora de realización, el deporte en cuestión, la ubicación, el sexo de los participantes y su privacidad. El campo de selección de deporte, desbloqueará de forma condicional, si fuera necesario, un selector de variante que cubrirá desde diferentes superficies hasta diferentes modalidades del mismo. También modificará el selector de rango de número de jugadores que se adaptará a las posibilidades de aforo del deporte en cuestión. Dependiendo de si el nivel de privacidad elegido es de grupo, se habilitará un selector de grupo con los grupos a los que pertenezca el usuario actual. Por último, la selección de la ubicación consistirá en un botón que abrirá un menú con un listado de las ubicaciones favoritas guardadas por el usuario. A este listado se le podrán agregar ubicaciones desde una ventana que se desplegará con un mapa, poblado con marcadores que identificarán las diferentes ubicaciones calificadas como oficiales y donde se podrán crear otras personalizadas para ser utilizadas o guardadas como favoritas. En caso de no tener ninguna ubicación guardada, se abriría directamente el mapa para seleccionar o crear una.

3.3.4 Perfiles profesionales

Se ofrece la posibilidad a las diferentes empresas, polideportivos, clubes y similares, de que puedan registrarse en la plataforma como usuarios normales y, tras ser verificados, acceder a un perfil especializado para empresas donde se podrá añadir las diferentes ubicaciones físicas que estas posean, crear grupos oficiales o informativos, crear eventos o competiciones con premios y estructurar los diferentes eventos dentro de los mismos.

4. Análisis del problema

A lo largo de este apartado, se van a identificar los diferentes requisitos que tendrá la red social basándonos en los objetivos y la propuesta expuestos en anteriores apartados de este TFG para que resulte una experiencia atractiva al público objetivo de la misma.

4.1 Requisitos funcionales de la aplicación

Como es lógico, una red social debe albergar usuarios y las operaciones que estos requieren, como puede ser su registro, su inicio de sesión, su edición de perfil o cambios de contraseña. Estas funcionalidades, son tratadas en el TFG complementario de Vicente y permiten dar paso a diversas necesidades que tiene este.

A continuación, se muestra, en diferentes tablas, los diferentes requisitos funcionales planteados:

- Requisitos generales de la aplicación:

ID	RF1
Nombre	Notificación de errores y eventos
Resumen	La aplicación deberá mostrar todo tipo de notificaciones al usuario para confirmar el éxito o fallo de las diferentes operaciones realizables en la aplicación.

- Requisitos relacionados con los Eventos:

ID	RF2
Nombre	Visualización rápida de los eventos
Resumen	Los usuarios han de poder obtener un resumen de la información más relevante sobre un evento en un formato visual adecuado.

ID	RF3
Nombre	Vista previa detallada de los eventos
Resumen	Los usuarios han de poder acceder a una vista específica de cada evento que proporcione toda la información referente a este y les permita interactuar con este.

ID	RF4
Nombre	Creación de eventos
Resumen	Los usuarios han de poder crear eventos y personalizar todos los parámetros del mismo.

ID	RF5
Nombre	Guardado de ubicaciones
Resumen	El usuario ha de poder guardar ubicaciones oficiales, de forma sencilla, para poder acceder a ellas más rápidamente.

ID	RF6
Nombre	Creación de ubicaciones
Resumen	Los usuarios han de poder crear ubicaciones personalizadas y guardarlas para poder acceder a ellas más rápidamente.

ID	RF7
Nombre	Filtrado de eventos
Resumen	El usuario ha de poder filtrar el listado de eventos, en base a los parámetros más adecuados, acordes a sus preferencias.

ID	RF8
Nombre	Búsqueda de eventos
Resumen	El usuario ha de poder buscar eventos por medio de su nombre, su identificador o por su creador.

- Requisitos relacionados con los grupos:

ID	RF9
Nombre	Visualización de información de grupos
Resumen	El usuario ha de poder visualizar los datos más relevantes de un grupo como su nombre, su imagen de grupo o su fecha de creación.

ID	RF10
Nombre	Visualización de los deportes practicados en los grupos
Resumen	El usuario ha de poder visualizar qué deportes son practicados, mayoritariamente, en un grupo.

ID	RF11
Nombre	Visualización de los miembros de los grupos
Resumen	El usuario ha de poder visualizar los usuarios miembros de un grupo.

ID	RF12
Nombre	Visualización de los eventos privados para los grupos
Resumen	El usuario ha de poder visualizar los eventos privados de un grupo.

ID	RF13
Nombre	Visualización de los *posts* privados para los grupos
Resumen	El usuario ha de poder visualizar los *posts* privados de un grupo.

- Requisitos relacionados con los perfiles profesionales:

ID	RF14
Nombre	Visualización de información del profesional
Resumen	El usuario ha de poder visualizar los datos más relevantes de un perfil profesional como su nombre, su imagen o un breve resumen.

ID	RF15
Nombre	Visualización de los deportes ofertados
Resumen	El usuario ha de poder visualizar los diferentes deportes que se pueden practicar por medio del perfil profesional.

ID	RF16
Nombre	Visualización de las ubicaciones disponibles
Resumen	El usuario ha de poder visualizar las diferentes ubicaciones de las que dispone el perfil profesional y del listado de deportes que se pueden practicar en esta en caso de diferir en las diferentes ubicaciones.

ID	RF17
Nombre	Creación de competiciones
Resumen	El perfil profesional, ha de poder crear competiciones compuestas de eventos.

ID	RF18
Nombre	Visualización de competiciones
Resumen	El usuario ha de poder visualizar las competiciones publicadas por los perfiles profesionales.

4.2 Propuestas alineadas con los requisitos funcionales

En primer lugar, en cada operación realizada por el usuario y que suponga una modificación de la base de datos, deberá mostrar, dependiendo del resultado de esta, un mensaje de error o de éxito. Estos se mostrarán como menús flotantes llamativos, en la parte inferior de cada pantalla que lo requiera (RF1).

El elemento básico necesario será la *card* de evento. Estas son representaciones de eventos en formato de tarjeta con la información más relevante de este (RF2). Será necesario la implementación de una página o vista previa de evento, con el objetivo de mostrar aquellos detalles que no se visualizan en las *cards* e interactuar con esta. Se propone un menú flotante que ocupe la mayor parte de la pantalla para esto (RF3). Estos eventos requerirán de una interfaz, para su creación, en la que se puedan ajustar todos sus parámetros. Se propone un menú flotante que no ocupe la totalidad de la pantalla pero que permita una buena visualización de los ajustes (RF4).

Además, será necesario la implementación de un sistema de ubicaciones que distinga entre ubicaciones registradas como oficiales, creadas por empresas o perfiles profesionales, y otras que serán personalizadas y propias de cada usuario. Ambos tipos, deberán de ser personalizables en términos del nombre con el que se guarden para ser identificadas más eficazmente por los usuarios. De esta manera, un evento podrá ser creado en una ubicación oficial cuyo creador tiene

guardada con un nombre personalizado y, el resto de usuarios, la identificarán con el nombre personalizado que estos le hayan dado o, en su defecto, por el nombre original de la misma. Las ubicaciones se podrán visualizar y crear desde un mapa integrado por medio de marcadores con controles para su guardado, personalización y borrado (RF5 y RF6).

También va a ser necesario poder acceder a un listado de eventos que se adapte a los intereses de cada uno, con un filtrado previo que mostraría los eventos más próximos primero. Además, estos deberán mostrarse solo si son accesibles para el usuario actual, es decir, solo aquellos que sean públicos, visibles para grupo a los que el usuario pertenece, o privados creados por amigos del mismo. A partir de esta vista previa de eventos, se podrá realizar un filtrado con un mayor nivel de detalle con las diferentes opciones ya propuestas (RF7). Otra característica sobre el listado de eventos, será la posibilidad de realizar una búsqueda personalizada por medio de una barra de búsqueda tradicional. Permitirá buscar coincidencias en nombre de eventos, en nombres de usuarios o de grupos y por medio de identificadores de evento (RF8).

Los grupos deberán poder administrarse por un usuario que tenga el rol de administrador del grupo. Este rol le pertenecerá a su creador y podrá ser transferible. Este usuario podrá cambiar la información básica del grupo como la descripción, la imagen, los deportes habituales y la visibilidad del grupo para poder hacerlo privado y que los miembros deban ser aceptados, o público (RF9 y RF10). Se mostrará el número de miembros pertenecientes al grupo y un listado con las imágenes de perfil y sus respectivos nombres para poder acceder a sus perfiles (RF11). Además, podrá desvincular eventos del grupo sin llegar a borrarlos y borrar o aprobar mensajes publicados en el muro del mismo, si así se configura. Estos se mostrarán como un panel de diapositivas o *swiper*, de dos en dos, al igual que las *cards* de evento (RF12 y RF13).

Los perfiles profesionales, además de editar su información básica como el resto de usuarios (RF14), podrán establecer manualmente los deportes que estos ofrecen (RF15). Podrán también crear ubicaciones oficiales ofertadas por estos con un listado, asignado a cada una de estas, de los deportes que en ella se pueden practicar en un mapa en miniatura (RF16). También tendrán la posibilidad de crear competiciones con eventos programados dentro de estas y asignar premios (RF17). Estos eventos se podrán ver listados en el perfil profesional del organizador o ser encontrados desde la página de eventos por medio de una búsqueda o un filtrado de competiciones (RF18).

5. Diseño de la solución

A lo largo de este apartado, se mostrará el proceso que se ha llevado a cabo para diseñar tanto el *frontend* y el *backend* de Buzzer Beater. Se cubrirá desde el prototipado inicial hasta el diseño final por la parte visual. En cuanto a la parte lógica, se analizará la arquitectura escogida para el desarrollo, las tecnologías que esta involucra y otros complementos específicos, tanto de la arquitectura, como APIs y librerías que serán necesarias.

5.1 Interfaces gráficas

Para llevar a cabo el diseño de las interfaces gráficas, se ha seguido un proceso de dos fases. En primer lugar, se ha desarrollado unos prototipos que definieran, para cada interfaz gráfica de la aplicación, el tipo de información a mostrar y la funcionalidad ofrecida (ver apartado 5.1.1). En segundo lugar, a partir de estos prototipos se han desarrollado las interfaces finales donde hemos optado por un diseño minimalista basado en las directrices del estilo de diseño Material Design propuesto por Google (ver apartado 5.1.2).

5.1.1 Prototipado

Como principal herramienta de prototipado, se ha escogido Uizard por su simplicidad, carácter gratuito y amplia librería de recursos pre establecidos. Además, este programa permite la creación de flujos interactivos para poder probar estos bocetos de forma interactiva.

El principal objetivo ha sido mantener una estructura que no resulte sobrecargada para el usuario y que tenga un carácter propio, pero al mismo tiempo, familiar con las redes sociales y aplicaciones deportivas que ya existen.

Los siguientes prototipos, conforman los bocetos originales, con leves modificaciones propuestas, principalmente, por las tutoras de este TFG y por tomas de decisión conjuntas con mi Vicente.

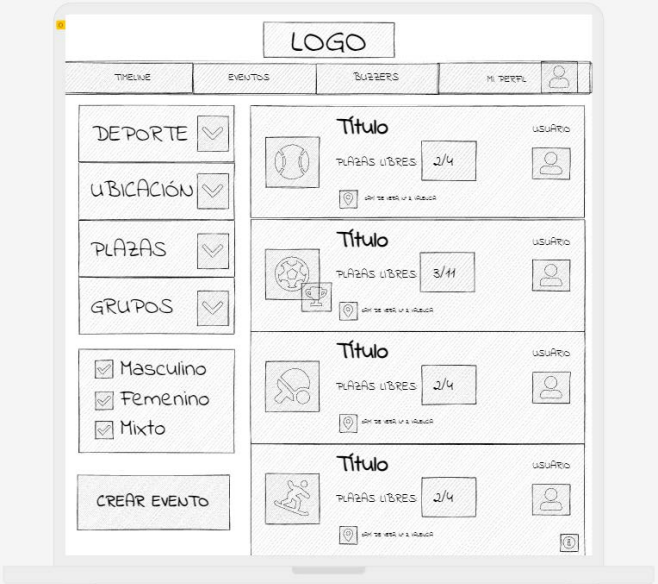


Figura 15: Página de eventos y filtros. Fuente: (Elaboración propia)

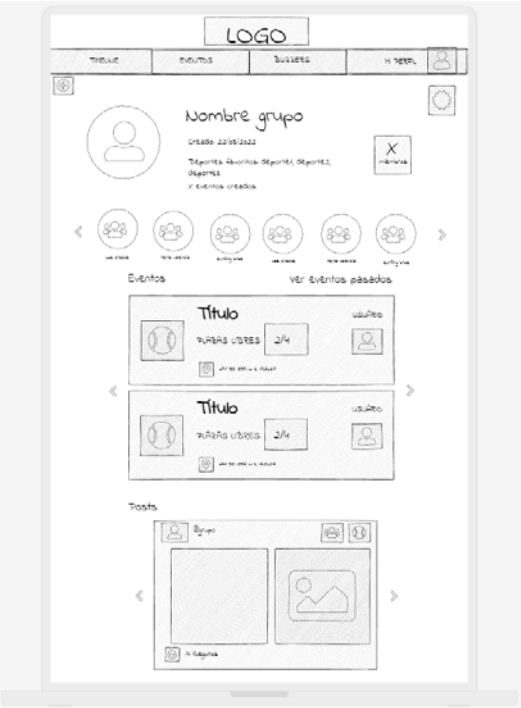


Figura 16: Perfil de grupos. Fuente: (Elaboración propia)

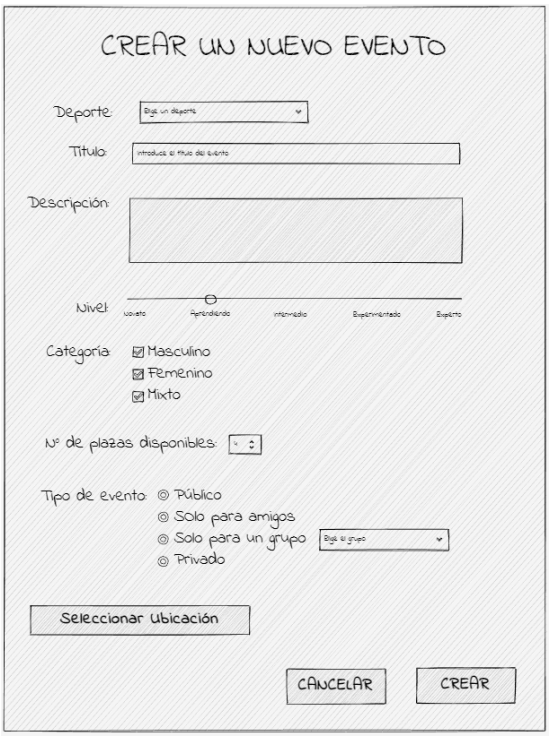


Figura 17: Menú de creación de evento. Fuente: (Elaboración propia)

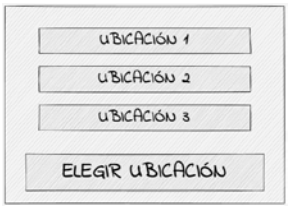


Figura 18: Menú de creación de evento – Listado de ubicaciones guardadas. Fuente: (Elaboración propia)

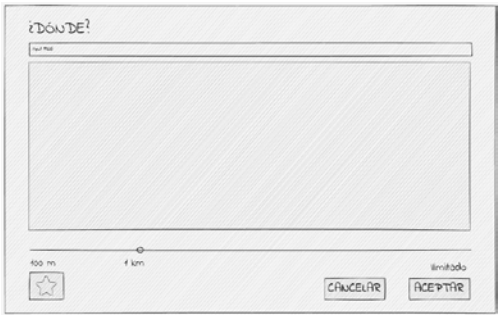


Figura 19: Menú de creación de evento – Selección de ubicación. Fuente: (Elaboración propia)

Figura 20: Menú de creación de evento – Guardado de ubicación. Fuente: (Elaboración propia)

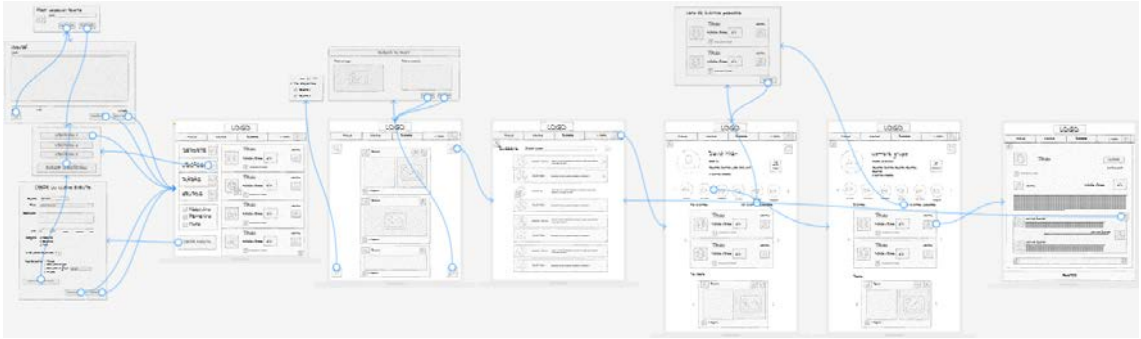


Figura 21: Flujos de interacción de Buzzer Beater. Fuente: (Elaboración propia)

5.1.2 Diseño final

Los diseños finales parten de la unión del prototipado anteriormente propuesto, con los principios de diseño de Material Design, adaptados a las necesidades de Buzzer Beater. En esta fase se pretende crear una paleta de colores, unos estándares de espaciado y tamaños y otras definiciones para ser utilizadas, de forma uniforme, a lo largo de toda la aplicación.

El estilo de Material Design, se centra en la simplificación y división en componentes modulares de todos los elementos visibles, lo cual se consigue mediante el resaltado de la interactividad de los elementos a lo largo de todos sus posibles estados, el establecimiento de paletas de colores homogéneas y que permitan una cómoda accesibilidad sin presentar faltas o excesos de contraste y el encapsulamiento de la información relevante (Google, s.f.).

La herramienta utilizada en este caso, ha sido Figma. Esta es una de las herramientas, en términos de diseño web, más potentes y utilizadas en la industria, siendo esta, para el 77% de los diseñadores, su herramienta primaria (Dexter, 2021).

Figma nos ha permitido colaborar, en vivo, durante el proceso de diseño. Es una aplicación basada en la web que puede ser utilizada fácilmente, no solo desde ordenadores sino también desde teléfonos móviles. Además, posee al igual que Uizard, la posibilidad de crear flujos de interacción para poder mostrar el producto final y realizar test A/B con muestras representativas de la población o público objetivo. Por otra parte, se ha utilizado el *plugin* oficial de Material Design Icons para la introducción de iconos oficiales de Google de uso público.

Los diseños que se muestran a continuación, han sido creados para Buzzer Beater y modificados en función de las recomendaciones recabadas de parte de algunos usuarios potenciales tras la realización de test A/B con la opción interactiva de Figma con amigos y familiares.

En primer lugar, siguiendo las directrices de Material Design, es importante crear elementos o átomos que sean reutilizables para mantener una coherencia a lo largo de toda la aplicación. Los átomos más característicos son las *cards* de evento (ver Figura 22), el campo de texto por defecto para cualquier formulario (ver Figura 23) o los mensajes de error, informativos o de éxito (ver

Figura 24).

La *card* de evento se ha dividido en tres regiones. La primera región, muestra la privacidad del evento, el deporte en particular que se practicará y si se trata de una competición o no. Esta región, trata de ser muy visual y solo se muestran iconos. La región central está compuesta de todos los datos relevantes y decisivos de un evento. La última región, muestra información sobre el usuario que ha creado el evento (ver Figura 22).

Figura 22: Átomo: Card de evento. Fuente: (Elaboración propia)

Para el átomo de los *inputs* de formulario, se han condensado todas las posibles casuísticas que se pueden dar para poder ver en conjunto, como se vería en cada situación. Desde un icono para mostrar el texto oculto de un campo de contraseña, hasta un mensaje de error genérico (ver Figura 23).

Figura 23: Átomo: campo de formulario con mensaje de error. Fuente: (Elaboración propia)

Para los errores, se han creado unas barras horizontales que se mostrarán por encima de todos los elementos de la página en la que se encuentre. Estas ocupan el ancho completo de la pantalla y tienen unos códigos de color representativos de la naturaleza de la información que están proporcionando (ver Figura 24).

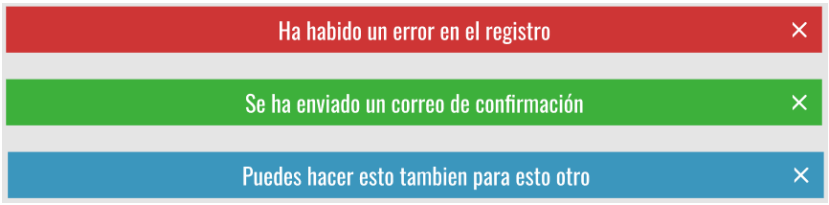


Figura 24: Átomo: mensajes de error, información y éxito. Fuente: (Elaboración propia)

La página de eventos es una combinación de diversos átomos y de otros elementos propios de esta página en concreto. Se ha encapsulado, en un panel lateral o *sidebar*, las diferentes opciones de filtrado de eventos. Cada una de estas opciones, proviene de átomos reutilizables que se actualiza, simultáneamente, con cualquier modificación del original. El botón de creación de evento es un elemento que debe resaltar y no ser encapsulado con otros elementos para darle una mayor importancia (ver Figura 25).

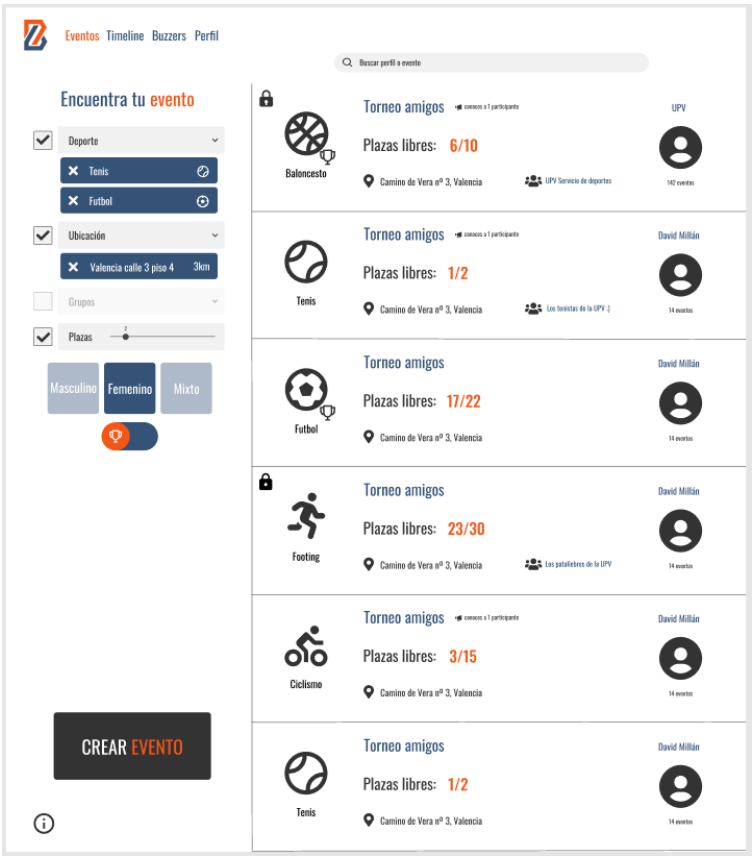


Figura 25: Página de eventos y filtrado lateral. Fuente: (Elaboración propia)

El perfil de grupo se caracteriza por agrupar información sobre sus miembros, los eventos que estos han creado para el grupo y las publicaciones de sus miembros. La información sobre la composición del grupo, los deportes favoritos o la fecha de creación, se muestran en la parte superior de este junto con la imagen identificativa del grupo y su nombre. Se ha diseñado con una estética y estructura igual a la del perfil individual que desarrolla Vicente en su TFG por la similitud de la información que muestra y para que sigan una homogeneidad conjunta (ver Figura 26).

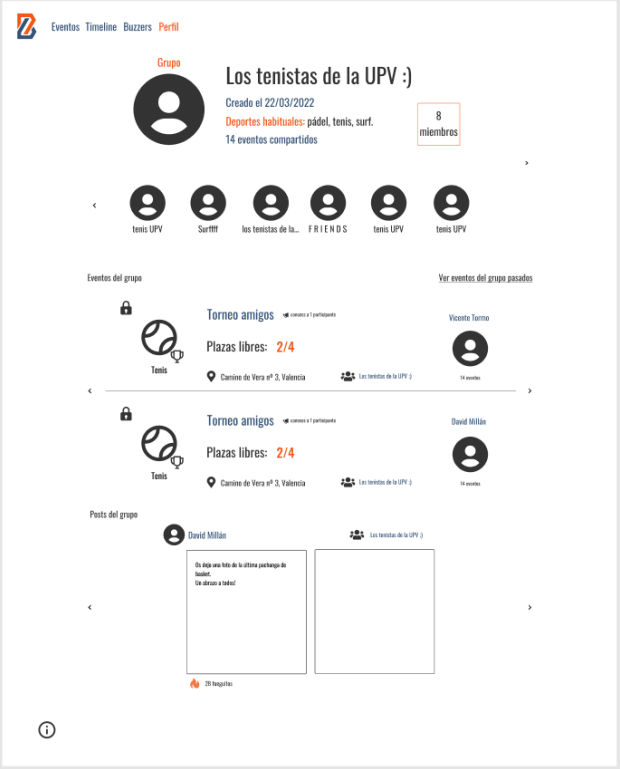


Figura 26: Perfil de grupo. Fuente: (Elaboración propia)

El proceso de creación de un evento se ha condensado en un formulario escueto pero con mucho poder de personalización (ver Figura 27).

Figura 27: Menú de creación de evento. Fuente: (Elaboración propia)

En el formulario de creación de evento, ha de completarse la información referente a la ubicación del mismo. Este menú desplegable (ver Figura 28), muestra un listado de ubicaciones guardadas como favoritas para poder ser seleccionadas rápidamente.

Figura 28: Menú de creación de evento – Listado de ubicaciones guardadas. Fuente: (Elaboración propia)

Para el guardado de nuevas ubicaciones, ya sean oficiales o personalizadas, se puede acceder a un mapa con diversos tipos de marcadores (ver Figura 29). Los marcadores en color negro, representan ubicaciones oficiales creadas por perfiles profesionales. Los marcadores naranjas, denotan ubicaciones personalizadas guardadas por el usuario. Al seleccionar cualquier de los marcadores, se despliega un menú con opciones de guardado, edición o eliminación en caso de tener dicha ubicación guardada.

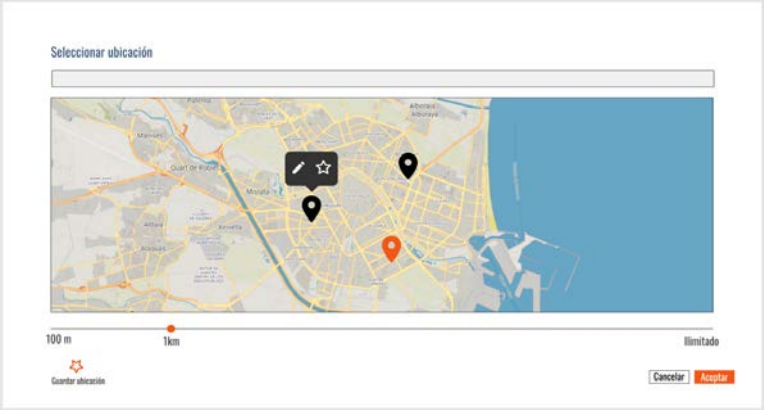


Figura 29: Menú de creación de evento – Selección de ubicación. Fuente: (Elaboración propia)

La opción de guara dado de una ubicación, permite establecer un nombre personalizado a esta para una mejor identificación (ver Figura 30).

Figura 30: Menú de creación de evento – Guardado de ubicación. Fuente: (Elaboración propia)

Gracias a las posibilidades que ofrece Figma, se ha podido diseñar el flujo de interacciones en la aplicación por medio de conexiones entre los diferentes elementos (ver Figura 31).

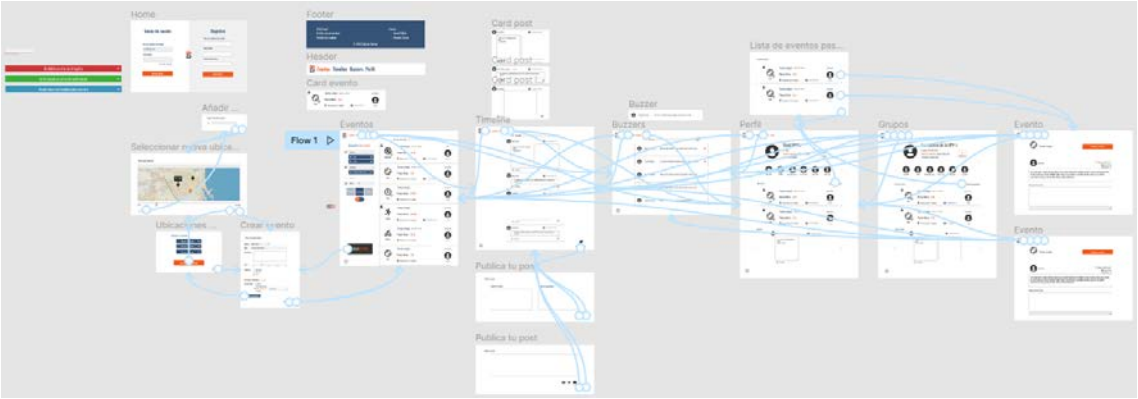


Figura 31: Flujos de interacción de Buzzer Beater. Fuente: (Elaboración propia)

Siguiendo los principios de Material Design, se ha diseñado en Figma, animaciones y otras interacciones como resaltado de elementos o cambios de color en botones para ofrecer una respuesta visual a las interacciones del usuario o para avisar de la interactividad de los diferentes elementos.

5.2 Arquitectura del sistema

A lo largo de este subapartado, se van a exponer los diferentes componentes que se han utilizado para la realización de este proyecto. Se comenzará con la base de la arquitectura utilizada y las tecnologías que esta involucra y se hablará de ciertas características adicionales que han hecho falta, propias de la arquitectura escogida, y del papel que estas desempeñan.

5.2.1 Tecnologías utilizadas

La arquitectura va a seguir un modelo cliente-servidor. Esta arquitectura se compondrá de un servidor basado en Linux con Nginx como software servidor en el cual estará alojado el núcleo de WordPress ejecutando el código PHP en el mismo (ver Figura 32). Nginx es una alternativa al servidor web Apache que se ha posicionado como número 1 en la industria, sobrepasando a Apache, a mediados de 2020, tras haber sido siempre, el servidor web más utilizado de todos los tiempos de manera indiscutible (W3Techs, 2022). Nginx, ofrece un mayor rendimiento ya que puede manejar múltiples conexiones, con diferentes hilos, de forma simultánea, mientras que Apache, destina un hilo a cada conexión (Marketers Group, 2022). Es por ello que el ahorro de memoria, se ve reflejado en un mayor rendimiento y un mayor aprovechamiento de los recursos.

La instancia de WordPress funcionará gracias a una base de datos a la cual se accede a través del sistema gestor de base de datos MySQL por medio de instrucciones SQL. Además, WordPress utiliza temas y *plugins* para poder personalizar las funcionalidades de la aplicación web y su aspecto.

Por otra parte, la conexión con APIs REST externas se realiza desde el servidor por medio de peticiones HTTP. Las APIs REST son las interfaces de programación de las aplicaciones basadas en transferencia de estados representacionales (Representational State Transfer Aplication

Programming Interface (REST API) en inglés) (Amazon Web Service, s.f.). Estas APIs son las más comunes y flexibles en Internet a día de hoy. Se basan en la transferencia, por parte del usuario, de unos datos de entrada por medio de una petición HTTP que ha de resolver la API y para la cual ha de devolver una respuesta en base a la operación.

Por lo que al cliente respecta, el usuario será el que, accediendo desde su navegador de elección, generará peticiones HTTP que serán dirigidas, gracias a su dominio, al servidor correspondiente. El servidor se encargará del procesamiento de las mismas accediendo, si la petición lo requiere, a la base de datos o las APIs conectadas. Por último, este devolverá al cliente los recursos requeridos por su petición.

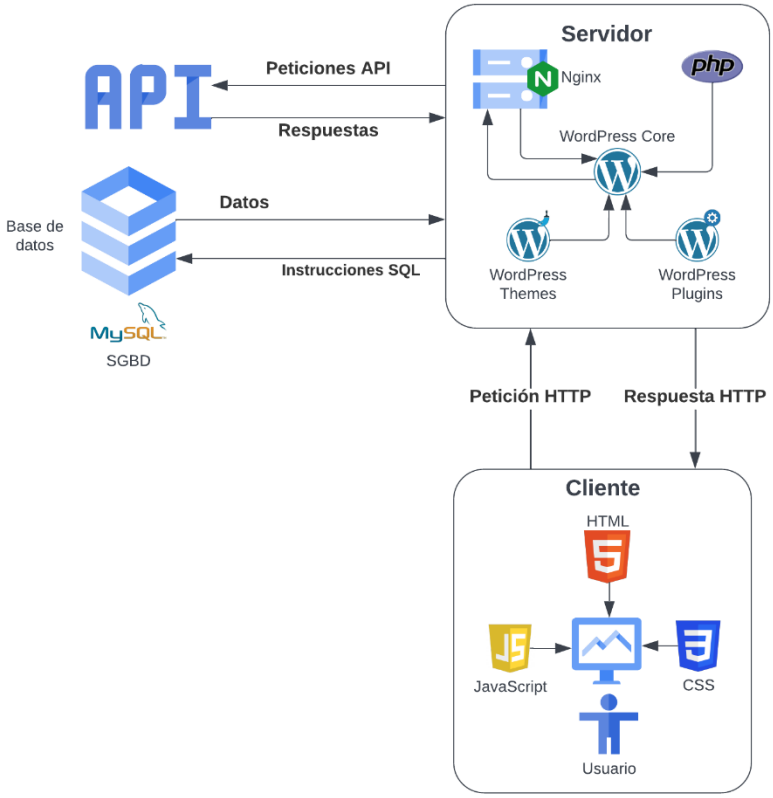


Figura 32: Arquitectura de la aplicación web. Fuente: (Elaboración propia)

Como no podría ser de otra manera, la aplicación o página ha de estar escrita, por lo menos, por un lenguaje de marcado para crear la estructura de la misma, por un lenguaje de estilo o presentación de la estructura de la página y, de manera más opcional pero utilizada prácticamente en todas las páginas web, un lenguaje de s*cripting* que permita manipular el Modelo de Objetos de Documento (Document Object Model (DOM) en inglés). El DOM es una interfaz o API que abstrae la estructura de un documento HTML a un árbol de nodos que representan cada elemento del documento.

Los lenguajes mencionados son los componentes más básicos. No obstante, si se quiere dar una mayor funcionalidad a una página web y poder calificarla de aplicación, entran en juego otros factores. Estos vienen a ser, una base de datos con su lenguaje para realizar consultas y un lenguaje para el *backend*. Este último es el encargado de todo el procesamiento interno de datos y el acceso a la base de datos. Su ejecución, a diferencia de los anteriormente mencionados, se lleva a cabo en el lado del servidor y, su resultado, es transformado a lenguajes *frontend* para la correcta interpretación de los navegadores.

Una vez identificados los diferentes componentes básicos, se puede concretar que los seleccionados serán, HTML5 como lenguaje de marcado, CSS3 como lenguaje de hoja de estilos, JavaScript ECMA6 como lenguaje de *scripting*, PHP 7.4.3 como lenguaje *backend* y SQL como lenguaje de consulta de bases de datos relacionales. Como sistema gestor de bases de datos, se ha utilizado MySQL 8.0.16.

Para la coordinación de toda esta tecnología y la administración del contenido de la aplicación, se utilizará el sistema gestor de contenido (Content Management System (CMS) en inglés) WordPress 6.1.1. Se trata del CMS más popular en Internet ya que, cerca del 43,3% de los sitios web, lo utilizan. Esto supone una cuota de mercado del 65,1% en cuanto a los CMS. Es el más seguro y el más personalizable en comparación con sus competidores (Osman, 2021).

Ya que WordPress funciona con lo que se conoce como temas, el objetivo va a consistir en crear un tema para WordPress totalmente personalizado que se adapte a los requisitos de Buzzer Beater. Estos temas están compuestos por plantillas que son asignadas a cada una de las páginas del sitio, las cuales se construyen a partir de otros recursos como hojas de estilo, *scripts*, recursos multimedia, librerías, *plugins* y APIs.

5.2.2 Estructura de WordPress

Una instalación "limpia" de WordPress consiste en diversos elementos. En primer lugar, encontramos en el más alto nivel de su estructura tres directorios llamados "wp-admin", "wp-content" y "wp-includes". Junto a estos podemos destacar algún fichero de configuración como "wp-config.php" que contiene los datos referentes a la conexión con la base de datos o "index.php" que sirve como punto de entrada de nuestra aplicación. Este hace una llamada a todos los ficheros necesarios para el funcionamiento de WordPress. Además, cabe destacar la importancia de ".htaccess" ya que este es un fichero básico y común a todos los sitios web, basados en Apache o Ngix, que permite personalizar el tratamiento de las peticiones que el servidor recibe, pudiendo así limitar el acceso a ciertas partes de la web, realizar redirecciones de todo tipo, desindexar páginas o incluso configurar otras funciones de PHP. Volviendo a los directorios, "wp-includes", contiene toda la API y las librerías que utiliza WordPress para su correcto funcionamiento. La carpeta "wp-admin", contiene todos los ficheros referentes al panel de control que convierten a WordPress en un CMS. Contiene plantillas que conforman el panel de control, también conocido como *backoffice* y todas las operaciones que se realizan desde el mismo. Por último, dentro de "wp-content", es donde se encuentran los temas, los *plugins*, los recursos y otros contenidos editables para la personalización de la aplicación como los idiomas.

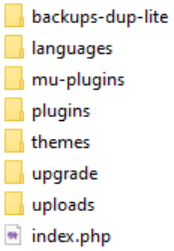


Figura 33: Contenido del directorio wp-content. Fuente: (Elaboración propia)

Como se puede apreciar, existe un fichero también denominado "index.php" a este nivel (ver Figura 33). Este será el fichero que se muestra como plantilla principal en caso de que no hubiera

ningún tema instalado ni activado. Inicialmente, es un fichero vacío. Resulta interesante comentar la utilidad de varios de estos directorios en mayor detalle:

- Directorio **languages**: Este almacena los ficheros que se encargan de hacer la web multilingüe. Está compuesto de ficheros con extensiones .po y .mo para cada idioma, agrupados por temas y *plugins* además de los originales, que traducen el panel de control. Estos albergan un conjunto de claves y valores junto con un comentario de las ubicaciones en las que se han encontrado el id.
- Directorio **plugins**: Se trata de un directorio que contiene todos los ficheros de configuración de los *plugins* que se tienen instalados.
- Directorio **themes**: Este será el principal foco de atención del desarrollo de esta aplicación. Contiene, en directorios separados, los diferentes temas instalados con todas sus plantillas, configuraciones y otros recursos propios de cada tema. Dentro de este directorio, se encuentra el tema desarrollado en un directorio bajo el nombre de "buzzerbeatertheme".
- Directorio **upgrade**: Se trata de un directorio en el cual WordPress guarda información crítica para su funcionamiento para poder restaurarse en caso de fallo durante una actualización del sistema o de un *plugin*.
- Directorio **uploads**: Dividido en sub-directorios nombrados por año y mes, almacena todos los ficheros multimedia subidos a través del panel de control.

Los directorios no mencionados no forman parte de los directorios originales de una instalación limpia de WordPress pero serán mencionados más adelante.

5.2.3 Estructura del tema

WordPress organiza su apariencia y funcionamiento en unidades llamadas temas. Estos albergan todas las plantillas y el funcionamiento de la página web que se quiere alcanzar. Los temas pueden ser activados o desactivados sin afectar al contenido, que persiste almacenado en la base de datos, con tal de cambiar la funcionalidad y la apariencia de la página a desarrollar. Existen una amplia gama de temas, tanto gratuitos como de pago, que sirven distintos propósitos u ofrecen apariencias distintas. Para el desarrollo de este proyecto, se va a crear un tema propio desde cero para disponer de un total control del mismo y prescindir de características innecesarias de otros.

Una vez se conoce mejor la estructura original de WordPress, es importante organizar la estructura del tema a desarrollar y seguir una serie de buenas prácticas que facilitarán el mantenimiento del mismo.

En primer lugar, hemos de conocer como WordPress muestra las diferentes páginas de nuestra aplicación. Estas dependen siempre de las llamadas plantillas. Estas plantillas son ficheros PHP en los que se describe, en primera instancia, cómo el servidor debe procesar los datos de entrada y después, como se va a visualizar este contenido. Las plantillas pueden conformar páginas enteras o ser partes más pequeñas reutilizables en otras plantillas. Estas últimas se conocen como bloques. Conociendo estos datos, podemos concluir que WordPress nos permite visualizar el contenido de las páginas de diferentes formas:

- Editor de bloques: WordPress permite la creación de páginas web con su editor nativo llamado "Gutenberg" a través de la disposición de bloques nativos y de otros personalizables.
- Plantillas asignadas: Existe la posibilidad de crear un fichero PHP, dentro del directorio del tema, que contenga el siguiente fragmento de código:

```php
<?php /* Template Name: Nombre */ ?>
```

Figura 34: Definición de plantilla implícita en un tema de WordPress. Fuente: (Elaboración propia)

Este fragmento es un comentario de PHP que WordPress reconoce e interpreta como una declaración de plantilla. Esta en concreto, se almacenará bajo el nombre de "Nombre" y podrá ser asignada a cualquier página desde el panel de configuración.

- Plantillas basadas en contenido: Estas plantillas no requieren de ninguna definición implícita en ellas, son asignadas siguiendo una jerarquía definida que, dependiendo del tipo de contenido de la página, son seleccionadas según su nombre de fichero (ver Figura 35). Este sistema busca primero la coincidencia de mayor especificidad y, en caso de no encontrarla, va recorriendo el árbol en busca de otras con menor detalle hasta acabar llegando a la más básica que es "index.php". Es por ello que todo tema ha de tener obligatoriamente un fichero "index.php" como comodín para todos sus tipos de páginas.

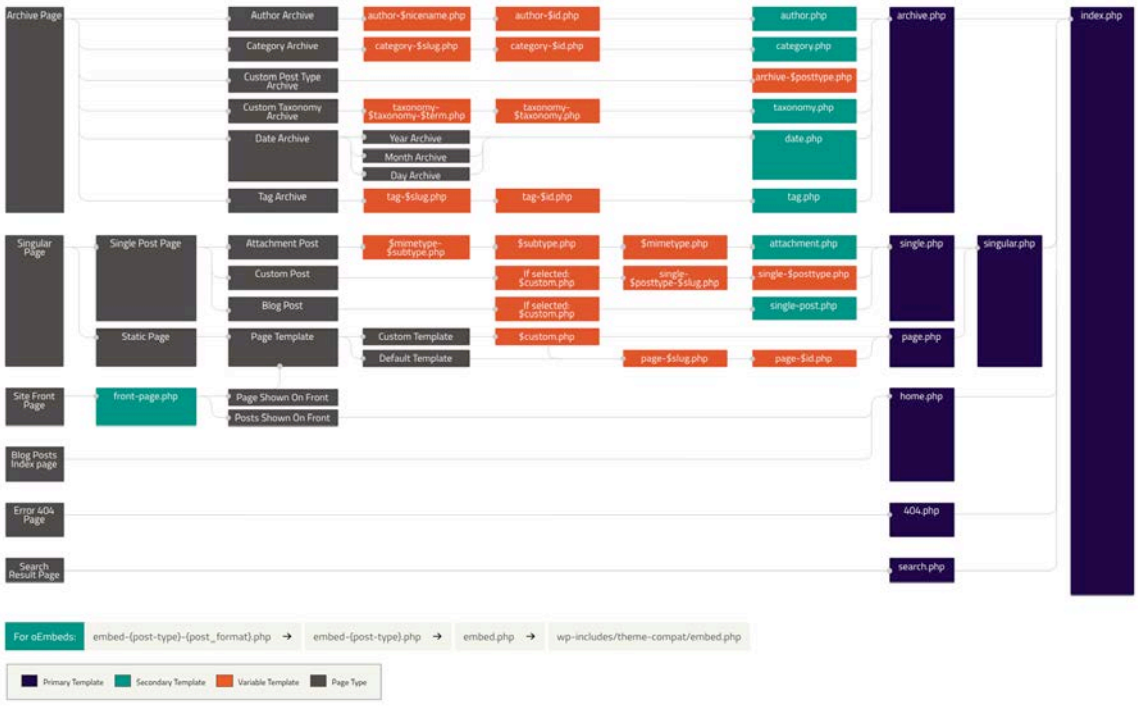


Figura 35: Jerarquía de asignación de plantillas según contenido de WordPress. Fuente: (WordPress, s.f.)

Conociendo esta estructura, se decidió separar las plantillas del tema en un directorio bajo el nombre de "templates". El resto de archivos como el referente al encabezado como "header.php", el pie de página como "footer.php" u otros ficheros de interés como la página de inicio de sesión como "login-page.php", se encuentran en el directorio principal del tema.

Otro aspecto relevante es la organización de los ficheros SASS. Para seguir un orden, establecimos, mi compañero y yo, la separación en directorios de los diferentes ficheros. Cada directorio tendría un archivo llamado "bundle.scss" en el cual se importarían todos los ficheros de su mismo directorio. A la vez, este sería importado por un "bundle.scss" situado en un nivel superior en la jerarquía de directorios hasta llegar al principal que luego se compila en código CSS.

5.2.4 *Plugins* de WordPress

Para la confección de este tema, se ha querido hacer una elaboración completamente personalizada para un mejor mantenimiento y para evitar incompatibilidades y caídas de rendimiento por programas de terceros o *plugins*. Sin embargo, se han considerado tres *plugins* que han permitido un desarrollo más eficiente y no crean dependencias críticas insalvables en caso de su desinstalación. Estos son los siguientes:

- **Duplicator**: No es un *plugin* vital para el funcionamiento de la aplicación y se acabará quitando, pero su utilidad es digna de mención. Este *plugin* gratuito, es el encargado de crear una copia perfecta de las configuraciones tanto de WordPress como de los *plugins*, de los temas instalados, de los recursos utilizados y de todo aquello que hace que funcione en perfectas condiciones una página web incluyendo su base de datos. Esta herramienta nos ha permitido, en fases iniciales del proyecto, crear una copia perfecta de la instalación y configuración inicial, para poder compartirla entre mi compañero y yo y asegurarnos de que utilizamos exactamente el mismo punto de partida. Además, es la herramienta que nos ha permitido, al finalizar el proyecto, hacer un despliegue de forma rápida y sencilla, incluyendo todo el trabajo ya plasmado en la base de datos local de la que se dispone. Este *plugin* permite la creación de estos paquetes de instalación, de manera totalmente personalizada para poder excluir ciertos directorios o limitar el tamaño de la exportación de la base de datos y otras características. La única limitación de este plan gratuito es el tamaño del archivo de instalación, pero no nos ha supuesto un problema debido a los tamaños de ficheros que manejamos.

- **LocoTranslate**: Este *plugin* proporciona una interfaz de edición de todas las cadenas traducibles de la aplicación con una enorme facilidad. Como ya se ha comentado antes, WordPress ya contempla la traducción del sitio con archivos de traducción. El problema que estos ficheros presentan es el formato en el que se editan y la detección de todas las cadenas traducibles del proyecto. Esta herramienta, también gratuita, se encarga de escanear todos los ficheros de la aplicación (no solo del tema) y generar un listado dividido en dominios para una traducción más cómoda y eficiente. Este dispone de opciones de pago como la traducción automática haciendo uso de inteligencia artificial pero no es una opción que se haya planteado si quiera.

- **Advanced Custom Fields (ACF)**: Como su nombre indica, este *plugin* permite la inserción de campos avanzados con una personalización muy potente. El objetivo principal de este reside en la abstracción de la capa de datos de la capa lógica de la aplicación. Además, ofrece una interfaz muy cómoda para la edición de páginas y la posibilidad de creación de bloques personalizables como los que se comentaban antes nativos del editor Gutenberg. Por desgracia, esta y muchos de los campos personalizados de mayor nivel, forman parte de la versión de pago que ha tenido que ser pagada.

5.2.5 Librerías y APIs utilizadas

A la hora de desarrollar una aplicación de estas dimensiones, aparece la necesidad de crear componentes complejos o que requieren de datos externos. Estos suelen presentar un desafío y una inversión de tiempo más que notable. Es por ello que existen librerías que ya implementan estas funcionalidades y APIs que nos ofrecen un acceso a datos que pueden ser de utilidad para nuestros proyectos.

En cuanto a librerías, se ha optado por utilizar las librerías de JavaScript gratuitas, jQuery y SwiperJS.

JQuery es la librería de JavaScript más utilizada eclipsando el mercado con un 94,8% de uso en páginas web con librerías de *scripting*. Esto supone que un 77,3% de todas las páginas web del mundo, utilizando jQuery (W3Techs, 2022). Esta proporciona un conjunto de funciones que simplifican el uso de diversas características nativas de JavaScript para una programación más legible, rápida y cómoda.

La otra librería que se ha utilizado ha sido SwiperJS, una librería para la creación de paneles de diapositivas deslizantes, muy cómoda de utilizar, que ofrece una gran variedad de configuraciones distintas y que se integra perfectamente con cualquier sistema operativo, buscador y resolución.

Por otra parte, para implementar el sistema de ubicaciones, se ha requerido de una API de mapas que permitiera visualizar un mapa, añadir marcadores personalizados y extraer información acerca de coordenadas cartográficas. Se ha barajado la posibilidad de utilizar la API de Google Maps por su facilidad y mi familiaridad con esta, pero, al final, se ha optado por la de OpenLayers. Las ventajas que ofrece esta API frente a la de Google son su gran variedad de capas de personalización visual y su carácter gratuito y de código abierto, frente al sistema de créditos por uso que Google ofrece. El único inconveniente es que supone un mayor reto al tratarse de una API más compleja y con mayor nivel de personalización.

6. Desarrollo de la solución propuesta

A lo largo de este apartado, se van a mostrar, los pasos seguidos durante el desarrollo del tema de WordPress que conforma el núcleo de la aplicación. Se cubrirá desde la preparación inicial, algunos de los aspectos más relevantes del código del proyecto como su estructura de datos o el tratamiento de la información desde el *backend*, una explicación de cómo hemos utilizado el control de versiones en nuestro favor y un breve resumen de cómo y por qué, acabamos adaptando nuestro desarrollo, a una base de datos unificada.

6.1 Control de versiones

Un aspecto muy relevante a la hora de desarrollar software, es el control de versiones. Este punto cobra aún más relevancia cuando se trata de un proyecto colaborativo para, además de controlar y depurar los diferentes incrementos que se van produciendo, poder trabajar de forma paralela con otros programadores.

Para poder tener un seguimiento total del proyecto, se considera como el primer punto a tener en cuenta antes de comenzar a escribir código o preparar el propio proyecto.

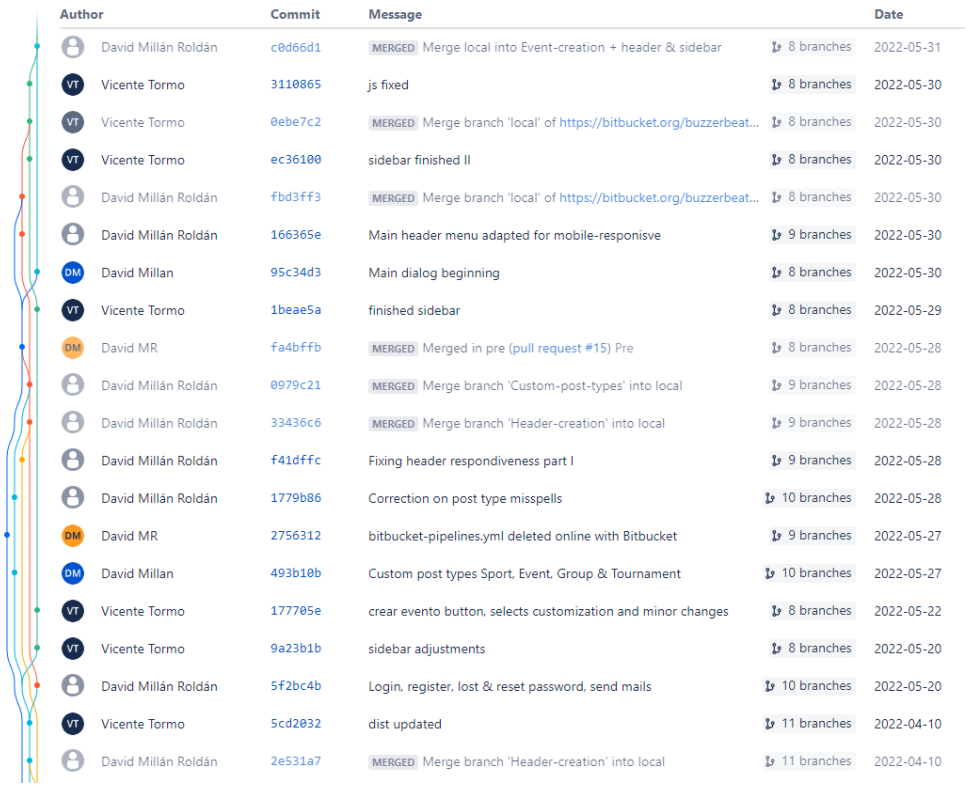
Se comienza inicializando un repositorio local en el directorio en el que se va a desarrollar el tema, es decir, en el directorio “themes/buzzerbeatertheme”. Para empezar, se abre una consola y se navega hasta el directorio del tema. Una vez situados, se ejecuta el comando “**git init**” para inicializar el repositorio y “**git remote add origin [USUARIO]:[APP_TOKEN]@[URL]**” para conectarlo con un repositorio remoto. Como se puede observar, para poder conectarlo al repositorio, hará falta proporcionar un nombre de usuario y una contraseña en el formato anteriormente mostrado. En el caso de Bitbucket, esta contraseña se llama *App token*.

Se ha decidido emular una aplicación web real por medio de la creación de tres ramas básicas:

- Master: Es la rama que alberga el estado actual de la web que se encuentra en producción, es decir, que está publicada.
- Pre: Es la rama dedicada a la realización de pruebas en el servidor remoto. Esta está conectada con una carpeta del servidor a la cual apunta un subdominio protegido por contraseña. Todas las actualizaciones le llegan desde la rama “Local” y representan incrementos.
- Local: Es la rama que recibe más actualizaciones. Es la rama que se utiliza en el entorno local para poder probar y hacer cambios en el código. Es importante porque, para cada *sprint*, mi compañero y yo nos creamos ramas separadas. Estas 2 ramas por *sprint*, han de confluir en esta rama Local, ser probadas en nuestras máquinas y, si todo funciona correctamente, actualizar la rama “Pre” con el incremento completo.

Dentro de cada rama, cada uno sigue una política propia de confirmación de cambios o *commit*. En mi caso, cada tarea completada y probada del *sprint* actual, se ha ido confirmando en la rama (de manera local).

Para cada *sprint*, se ha creado una rama distinta la cual se iba actualizando con los diferentes *commits* y se iba actualizando con ramas paralelas con las que tuviera algún tipo de dependencia (ver Figura 36). Finalmente, todas las ramas acaban por confluir en la rama “local”.



Author	Commit	Message		Date
David Millán Roldán	c0d66d1	MERGED Merge local into Event-creation + header & sidebar	8 branches	2022-05-31
Vicente Tormo	3110865	js fixed	8 branches	2022-05-30
Vicente Tormo	0ebe7c2	MERGED Merge branch 'local' of https://bitbucket.org/buzzerbeat...	8 branches	2022-05-30
Vicente Tormo	ec36100	sidebar finished II	8 branches	2022-05-30
David Millán Roldán	fbd3ff3	MERGED Merge branch 'local' of https://bitbucket.org/buzzerbeat...	8 branches	2022-05-30
David Millán Roldán	166365e	Main header menu adapted for mobile-responisve	9 branches	2022-05-30
David Millan	95c34d3	Main dialog beginning	8 branches	2022-05-30
Vicente Tormo	1beae5a	finished sidebar	8 branches	2022-05-29
David MR	fa4bffb	MERGED Merged in pre (pull request #15) Pre	8 branches	2022-05-28
David Millán Roldán	0979c21	MERGED Merge branch 'Custom-post-types' into local	9 branches	2022-05-28
David Millán Roldán	33436c6	MERGED Merge branch 'Header-creation' into local	9 branches	2022-05-28
David Millán Roldán	f41dffc	Fixing header respondiveness part I	9 branches	2022-05-28
David Millán Roldán	1779b86	Correction on post type misspells	10 branches	2022-05-28
David MR	2756312	bitbucket-pipelines.yml deleted online with Bitbucket	9 branches	2022-05-27
David Millan	493b10b	Custom post types Sport, Event, Group & Tournament	10 branches	2022-05-27
Vicente Tormo	177705e	crear evento button, selects customization and minor changes	8 branches	2022-05-22
Vicente Tormo	9a23b1b	sidebar adjustments	8 branches	2022-05-20
David Millán Roldán	5f2bc4b	Login, register, lost & reset password, send mails	10 branches	2022-05-20
Vicente Tormo	5cd2032	dist updated	11 branches	2022-04-10
David Millán Roldán	2e531a7	MERGED Merge branch 'Header-creation' into local	11 branches	2022-04-10

Figura 36: Control de versiones por ramas en Bitbucket con git. Fuente: (Elaboración propia)

6.2 Instalación y preparación del entorno de desarrollo

En primer lugar, para poder visualizar una página o aplicación web, hace falta poner en marcha un servidor que se encargue de la gestión de los ficheros de la misma, reciba peticiones, las procese y devuelva una respuesta al cliente que realizó la solicitud.

Existen diversas herramientas para poder levantar un servidor en una máquina no dedicada. Por familiaridad, se optó en un principio por XAMPP. Este potente programa ofrece una distribución de Apache como *software* servidor web bajo el protocolo HTTP de código abierto, MariaDB como sistema gestor de base de datos basado en MySQL de código abierto e intérpretes para Perl y PHP como lenguajes *backend*. XAMPP permite una muy rápida instalación y configuración del servidor local proporcionando un directorio llamado “htdocs” dentro de su carpeta de instalación donde se deben situar todos ls ficheros que queramos que formen parte de nuestro servidor web. Este ofrece un punto de acceso por medio de la dirección local “localhost” o su equivalente en IPv4 “127.0.0.1”. El principal inconveniente que presenta es el hecho de que, para actualizar la versión de PHP, hace falta instalarse otra versión distinta de XAMPP. Esto genera una necesidad de mover todos los ficheros del anterior directorio de instalación al Nuevo cada vez que esta situación se da. Además, para poder emular un servidor que implemente el protocolo de correo SMTP, hay que realizar unos cambios en la configuración de la aplicación algo complejos y no permite la captura de los correos salientes.

Al final, se cambió de XAMPP a LocalWP como herramienta de gestión del servidor web local durante el desarrollo del proyecto. LocalWP, es un entorno de desarrollo especializado en WordPress. Permite la sincronización con páginas web que utilizan este CMS para realizar cambios y subirlos en vivo. Además de proporcionar todas las herramientas necesarias para levantar un servidor web, ofrece un sistema de captura de correos salientes y la posibilidad de

crear un entorno público, protegido por contraseña bajo un subdominio de la compañía que lo desarrolla, para poder compartir los avances con desarrolladores o clientes sin la necesidad de que estos tengan una copia del proyecto o estén juntos de forma presencial. Como ya se ha comentado antes, un punto bastante importante es la instalación de actualizaciones de PHP y cualquier otro componente que lo requiera. LocalWP ofrece un entorno personalizado para cada Proyecto, permitiendo controlar y administrar, en todo momento, el software utilizado junto con sus respectivas versiones.

Para comenzar, se instaló LocalWP y se creó un Nuevo Proyecto bajo el nombre de BuzzerBeater. Este se ha configurado como un servidor local basado en una distribución de Nginx con un intérprete de PHP 7.4.3 y MySQL 8.0.16 (ver Figura 37). La instalación limpia de WordPress está actualizada a la última versión actual, la 6.1.1.

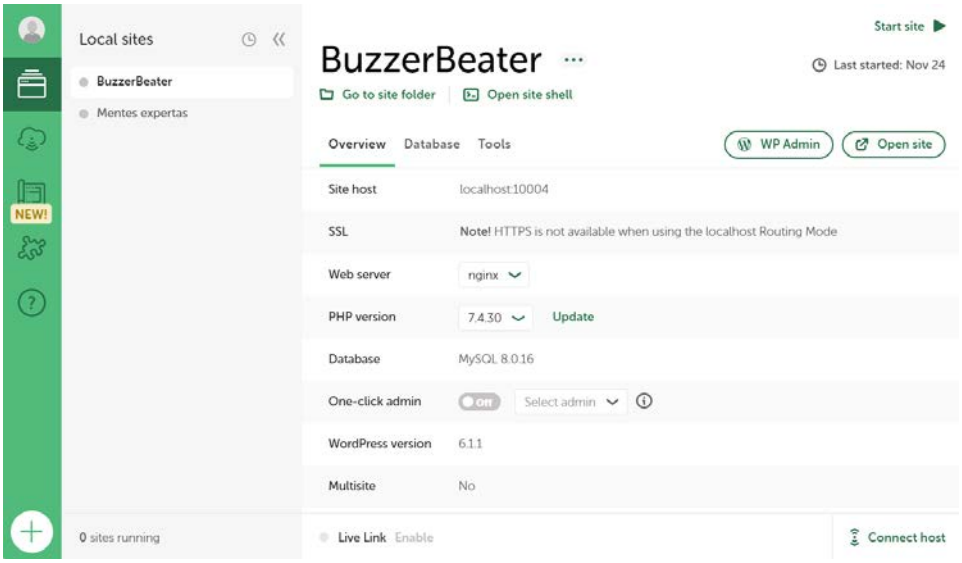


Figura 37: Interfaz de LocalWP. Fuente: (Elaboración propia)

Una vez se tiene el entorno de desarrollo preparado, comienza la preparación del entorno de desarrollo. En primer lugar, se inició un repositorio git local que se sincronizó con Bitbucket. En este repositorio, se preparó el tema partiendo de los ficheros básicos y mínimos necesarios para ser identificado un tema en WordPress. Estos son los ficheros “functions.php”, “index.php” y “style.css”. Estos ficheros deben llamarse exactamente como se acaba de indicar para que el tema sea identificado. Sus propósitos son los siguientes:

- Fichero “functions.php”: Se encarga del registro de funciones, estilos, *scripts* y cualquier otro recurso que se pueda utilizar. Además, es utilizado para tomar ventaja de los *hooks* que ofrece WordPress. Estos *hooks* son puntos de entrada de código, dentro del código fuente de la base de WordPress, por medio de los cuales, se pueden realizar acciones en puntos clave muy concretos como durante el registro de un usuario, o para modificar comportamientos internos por defecto del CMS.
- Fichero “index.php”: Esta es la, anteriormente mencionada, plantilla base comodín para cualquier página que no tenga asignada una, dentro del tema, a través del árbol de jerarquías por defecto de WordPress.
- Fichero “style.css”: Consiste del fichero principal de estilos del tema. Este, además de código CSS, puede contener información de utilidad sobre el tema a modo de comentario de código. Esta información incluye el nombre del tema, el autor del mismo, una descripción, la versión de este, la licencia y su URI y un texto de dominio interno.

```
/*
Theme Name: Buzzer Beater Theme
Theme URI: https://millandavid.com
Author: David Millán & Vicente Tormo
Author URI: https://millandavid.com
Description: Tema desarrollado para la red social deportiva Buzzer Beater
Tags: social network, timeline, events, acf, custom
Version: 1.0
Requires PHP: 7.4.2
Text Domain: buzzerbeater
*/
```

Figura 38: Información del tema en el fichero style.css. Fuente: (Elaboración propia)

Además, para un mejor mantenimiento y separación del código, se optó por estructurar el tema de la siguiente forma:

- **Plantillas**: Se almacenan todas bajo el directorio "templates". Dentro de este podemos encontrar plantillas de página, un directorio para las plantillas de los correos bajo el directorio "emails" y un directorio con las diferentes partes reutilizables como pueden ser las *cards* de evento o los menús desplegables, bajo el directorio "parts".

- **Traducciones**: Para poder hacer el tema transferible sin tener dependencias fuera de su propio directorio, se ha creado un directorio "languages" en el cual se generan, en archivos de extensión .pot, las traducciones propias del tema.

- **Procesamiento *backend***: Para tratar el flujo de información, tanto registros, como creación de eventos o la interacción entre usuarios y grupos, se ha situado, bajo un directorio llamado "ajax_calls", todos los ficheros que tratan estas peticiones al servidor y la base de datos de manera asíncrona. Más adelante se hablará de las llamadas AJAX en detalle.

- **Archivos multimedia**: Los archivos multimedia han de ser optimizados y comprimidos sin perder nivel de detalle. Para ello, los ficheros originales, se sitúan bajo el directorio "src/media" y sus análogos optimizados en "dist/media".

- **Estilos y *scripts***: Por último, es muy importante destacar la importancia de la organización de un proyecto a nivel de directorios y ficheros. Podría seguirse una estructura plana para su desarrollo, pero la agrupación de unidades correlacionadas, además de ofrecer modularidad, a la hora de depurar aumenta la legibilidad y la edición del código. Por ello, para la creación de estilos, se ha optado por utilizar el lenguaje de hoja de estilos SASS. Este ofrece la posibilidad de crear hojas de estilos con jerarquías complejas y reglas o *mixins* que CSS no permite. Ayuda a crear un código más limpio y escalable con menos líneas de código. Sin embargo, los navegadores no soportan SASS por lo que hace falta pre procesarlo y generar un fichero CSS homólogo. Para ello, los ficheros SASS irán localizados bajo el directorio "src/scss" y el fichero generado, en el directorio "dist/css". Los scripts se estructuran de una manera similar, se han separado en módulos bajo el directorio "src/js" y el fichero generado se sitúa bajo "dist/js".

Los distintos procesos a los que han de ser sometidos los diferentes tipos de ficheros que se encuentran bajo el directorio "src", se orquestan desde NodeJS y su sistema gestor de paquetes npm. Gracias a npm, se ha podido poner a punto un conjunto de herramientas para automatizar

procesos durante el desarrollo de la aplicación que han ayudado a que este sea un proceso más ágil y menos costoso. Con npm, se instaló la herramienta de automatización de procesos Gulp. Esta permite la definición de funciones ejecutables que, en conjunción con otros paquetes de NodeJS, nos han permitido pre procesar los ficheros SASS, unificarlos al igual que los *scripts* y comprimirlos, ubicando los ficheros resultantes en sus respectivos directorios del directorio padre "dist".

El proceso de implementación de estas herramientas es el siguiente:

1. Inicialización de un proyecto con npm usando el comando "**npm init**" desde la consola en la ubicación del tema.

2. Instalación global de gulp con el comando "**npm install --global gulp-cli**" e instalación en el directorio específico del proyecto con el comando "**npm install --save-dev gulp**".

3. Los siguientes pasos consisten en la instalación de diversos paquetes de NodeJS con npm. Para no mostrar todos los comandos ejecutados, se muestra el esquema del comando que se repitió para instalar cada uno de estos: "**npm install –save-dev PAQUETE[@VERSION]**". La versión es opcional, pero por temas de compatibilidades, hubo que ajustar la versión de diversos paquetes.

4. El último paso consistió en la declaración de estas tareas automatizadas de Gulp. Estas se definen en un fichero llamado "**gulpfile.js**". Sin embargo, como se está desarrollando el proyecto con la versión de JavaScript ECMA6, hace falta la traducción de esta especificación a la versión anterior denominada ECMA5 por medio del paquete Babel para evitar incompatibilidades con Gulp y otros paquetes. Esto deriva en el renombramiento a este archivo a "gulpfile.babel.js".

Una de las grandes ventajas de npm es el registro que mantiene de los paquetes y sus versiones. Esto lo hace gracias al fichero "package.json". Este está ubicado en la raíz del proyecto y gracias a este, una vez se tiene el proyecto debidamente configurado con los paquetes de NodeJS necesarios, basta con copiar este archivo a otro proyecto y lanzar, por línea de comandos, la instrucción "**npm install**". Esta se encargará de leer todas las dependencias, del fichero anteriormente mencionado, e instalarlas para el correcto funcionamiento del proyecto.

Algunos de los paquetes, compatibles con Gulp, más destacables de npm utilizados son los siguientes:

- **Gulp-sass**: Ha permitido la compilación de código SASS a CSS.
- **Gulp-clean-css**: Ha permitido la compresión de código CSS.
- **Gulp-if**: Ha permitido la ejecución condicional de tareas de Gulp dependiendo de variables de entorno. Esto ha permitido, con una misma función, realizar ciertos procesos propios de un entorno en desarrollo, como la creación de mapas de SASS, del cual se hablará a continuación, y otros procesos distintos, propios de un entorno en producción, como la compresión de código CSS y JavaScript.
- **Gulp-imagemin**: Ha permitido la compresión de imágenes y vídeos de cualquier formato compatible.
- **Gulp-sourcemaps**: Ha permitido la creación de mapas de SASS. Estos crean una equivalencia entre el código compilado en CSS con los ficheros originales de SASS para una rápida identificación de ficheros y corrección de errores en los estilos de la aplicación.

Todos estos paquetes se han utilizado en diferentes funciones, ejecutables individualmente, que se han terminado unificando en un único proceso que se ejecuta bajo el comando "**gulp**" ya que

este proceso, se exportó como la función por defecto la cual es definida en el fichero "package.json" junto con las dependencias de paquetes de NodeJS del proyecto. Este se encarga de vaciar el directorio "dist", en paralelo, trata los archivos multimedia, los *scripts* y los estilos. Finalmente, copia los resultados generados al directorio "dist" y se queda a la escucha de cualquier modificación dentro de este directorio "src" para volver a ejecutar todo este proceso de forma automática al guardar cualquier fichero. Los procesos que ocurren en paralelo, durante el tratamiento de archivos multimedia, *scripts* y estilos consisten la compilación de estilos, y su posterior compresión junto con los *scripts* y los archivos multimedia.

6.3 Desarrollo de los elementos del *backend*

Del desarrollo de este proyecto, es destacable la planificación a nivel de código dentro de cada fichero para poder mantener un proyecto limpio y sin incongruencias entre la parte de mi compañero y la mía. Se va a explicar cómo se estructuraron los datos con la ayuda de las herramientas que proporciona WordPress, del tratamiento de los datos en el lado del servidor, de las consideraciones que se llevaron a cabo en todo momento sobre seguridad, de la administración de versiones y el control de ramas y, por último, de la decisión de unificar y compartir una única base de datos.

6.3.1 Estructura de datos

WordPress proporciona una estructura de datos basada en los *posts*. Estos objetos, representados como instancias de la tabla *wp_posts*, pueden modificar su propiedad denominada *post type* para representar un tipo de objeto distinto. Los *post type* más comunes son el *post* del blog o la página. Cada uno de estos tipos, tienen sus propiedades únicas almacenadas en una tabla a parte denominada *wp_postmeta*. En esta, se pueden especificar, para un *post* con un identificado por su *ID* como *post_id*, diversas propiedades gracias a los pares de clave *meta_key* y valor *meta_value*.

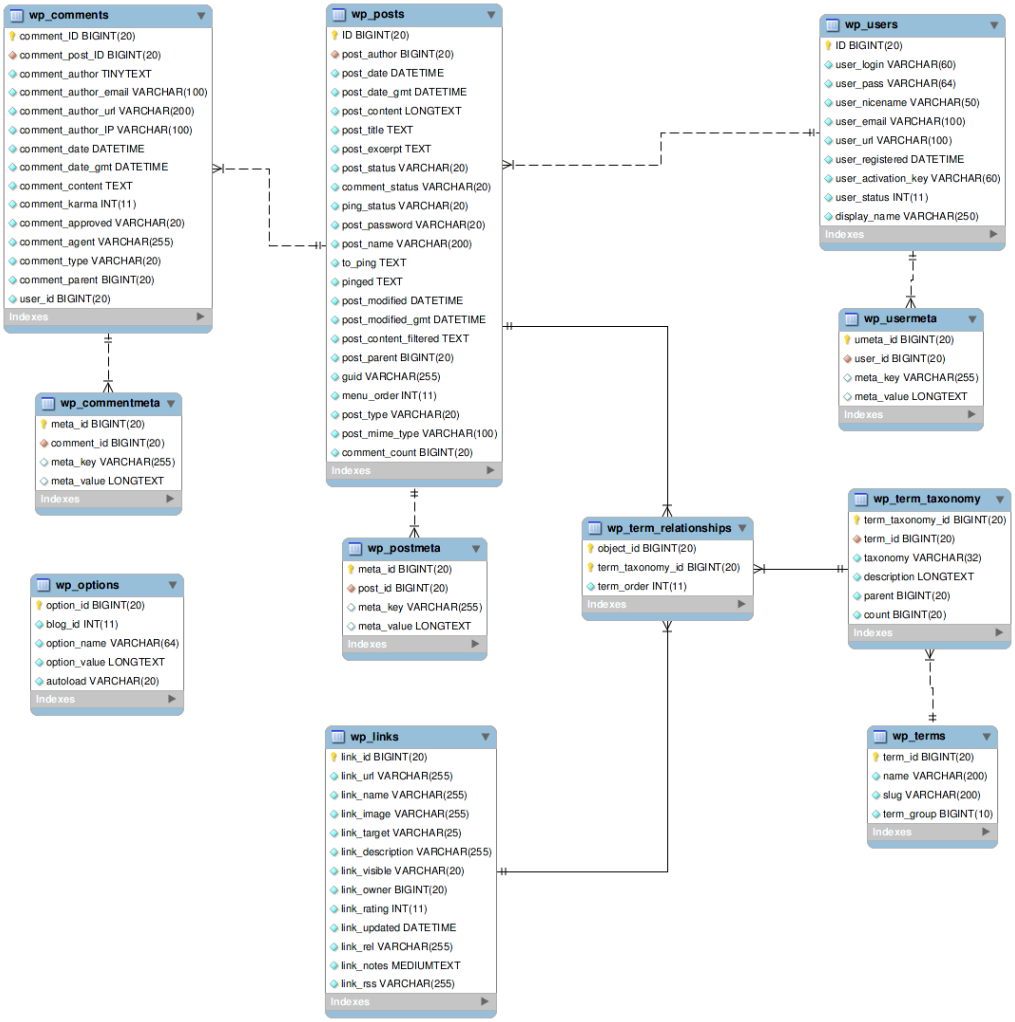


Figura 39: Estructura de la base de datos de WordPress. Fuente: (WordPress, s.f.)

Teniendo en cuenta la estructura de la base de datos que proporciona WordPress, se opta por la utilización del *plugin* Advanced Custom Fields o ACF, que nos permite la creación y administración de una mayor variedad de tipos de datos para un *post type* dado. Este nos proporcionará herramientas de creación, recuperación, modificación y eliminación de estos datos sin la necesidad de preocuparse por la estructura interna de la base de datos del CMS. La estructura de la base de datos de WordPress, no se ve modificada con la creación de campos de ACF ya que estos datos son inyectados en la base de datos como instancias de "wp_postmeta" asignándole pares de claves y valores, a los *posts* del *post type* al que pertenecen.

A pesar de poder crear nuevos campos para los tipos de datos, es conveniente la posibilidad de crear otros tipos de datos para una mejor administración y manipulación de la aplicación. Para ello, WordPress si permite la creación de *post types* personalizados de forma sencilla. Esto nos servirá para definir esta estructura de datos necesaria.

Los *post types* creados para poder diferenciar el tipo de contenido, junto con sus campos personalizados han sido los siguientes:

- Deporte (sport):
  - Variantes (*Array* de campos de texto)
  - Jugadores (*Array* de números enteros)
  - Icono (Imagen)
- Evento (event):

    - Descripción (Cadena de texto)
    - Deporte (ID de evento)
    - Fecha (Selector de fecha y hora)
    - Nivel (Selector de cadena de texto)
    - Variante (Selector de cadena de texto)
    - Categoría (Selector de cadena de texto)
    - Competición (Booleano)
    - Ubicación (Grupo de campos):
        - ID (ID de ubicación)
        - Nombre (Cadena de texto)
        - Coordenadas (Grupo de campos):
            - Latitud (Número de coma flotante)
            - Longitud (Número de coma flotante)
    - Jugadores (*Array* de IDs de usuario)
    - Máximo número de jugadores (Número entero)
    - Visibilidad (Grupo de campos):
        - Privacidad: (Selector de cadena de texto)
        - Grupos: (*Array* de IDs de grupo)
    - Creador (ID de usuario)
- Grupo (group):
    - Líder (ID de usuario)
    - Descripción (Campo de texto)
    - Miembros (*Array* de IDs de usuario)
    - Imagen de grupo (Imagen)
- Torneo (tournament)
    - Descripción (Campo de texto)
    - Deporte (ID de deporte)
    - Fecha (Selector de fecha y hora)
    - Ubicación (Grupo de campos):
        - ID (ID de ubicación)
        - Nombre (Cadena de texto)
        - Coordenadas (Grupo de campos):
            - Latitud (Número de coma flotante)
            - Longitud (Número de coma flotante)
    - Jugadores (*Array* de IDs de usuario)
    - Categoría (Selector de campo de texto)
    - Eventos (*Array* de IDs de evento)
    - Estructura (Mapa):
        - Clave (Campo de texto de la fase)
        - Valor (ID del evento)
    - Organizador (ID de usuario)
- Ubicación (location)
    - Deportes (*Array* de IDs de deporte)
    - Latitud (Número de coma flotante)
    - Longitud (Número de coma flotante)
- Organizaciones:
    - Ubicaciones (*Array* de IDs ubicación)

Como es lógico, cada uno de estos *post types*, tienen por defecto un título o nombre y un identificador o ID que nos permite establecer relaciones entre ellos. Para el resto de campos, el *plugin* ACF, nos proporciona una interfaz muy cómoda (ver Figura 40) para la creación de campos

agrupados bajo un nombre y establecer su ubicación. Esta ubicación representa una acotación que delimita a qué *post types* van asignados qué campos para evitar que cada campo aparezca bajo cualquier tipo de dato.

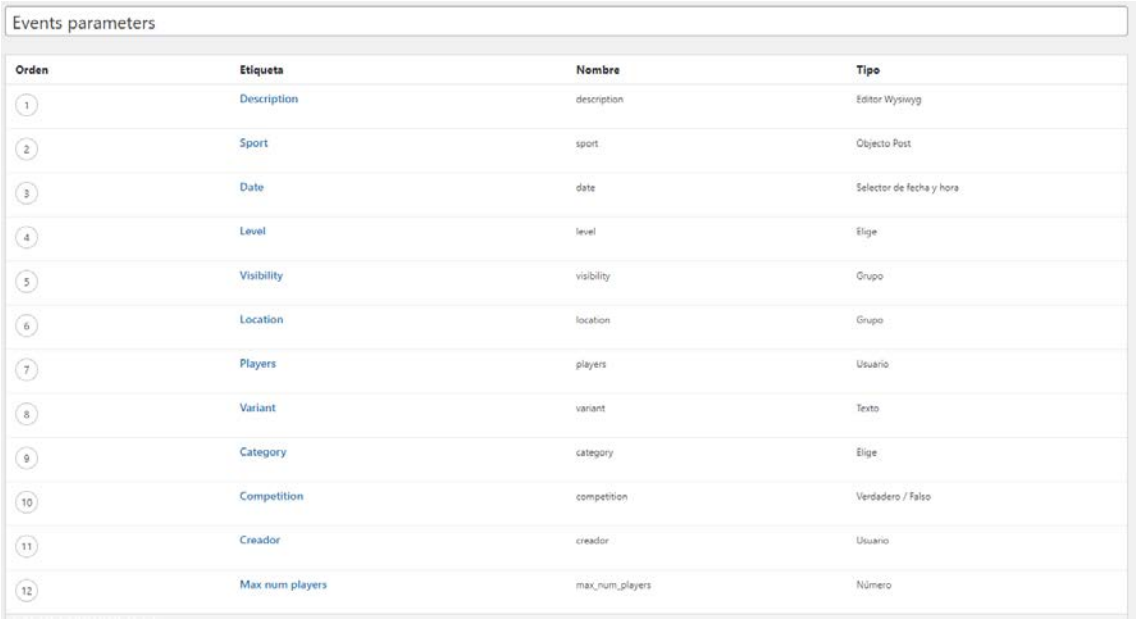


Figura 40: Interfaz de campos de ACF. Fuente: (Elaboración propia)

Se destaca de esta estructura la decisión de dar la opción, en los campos de ubicación, de poder seleccionar una ubicación por su ID o especificar una personalizada por medio de su nombre, su latitud y su longitud. Este sistema permite la creación de ubicaciones personalizadas, no oficiales, para que los usuarios no dependan solo de empresas o polideportivos. Por otra parte, los usuarios también tienen campos personalizados. El más relevante a destacar es el de ubicaciones guardadas. Este es un campo en formato JSON que permite el guardado de ubicaciones, tanto oficiales, por medio de su ID, como personalizadas, por medio de su nombre, latitud y longitud, e incluso un híbrido que consiste en una ubicación oficial y un nombre asignado por el usuario. Esto permite poder ver las ubicaciones oficiales, de los distintos eventos, con un nombre personalizado que solo puede ver este.

Otro concepto a tener en cuenta, es que las organizaciones no son un *post type* sino un rol de usuario. Estas son tratadas, como ya se explicó, como un usuario verificado. ACF nos permite establecer la ubicación de los campos personalizados en base a una gran variedad de opciones. En este caso, las más utilizadas han sido basadas en el *post type* (ver Figura 41) para los eventos, las ubicaciones, los torneos, los grupos y los deportes y en el rol de usuario (ver Figura 42) para el caso de los perfiles profesionales.

Figura 41: Ubicación de grupo de campos de ACF - Post type evento. Fuente: (Elaboración propia)

Figura 42: Ubicación de grupo de campos de ACF - Rol de usuario organización. Fuente: (Elaboración propia)

6.3.2 Llamadas AJAX

Un aspecto primordial de una aplicación web es el tratamiento de datos. Es necesario poder realizar operaciones CRUD sobre estos de forma segura. Las operaciones CRUD son las correspondientes a la creación, lectura, actualización y eliminación de datos, por sus siglas en inglés. El usuario será el principal sujeto que lleve a cabo estas operaciones, pero ha de ser en un entorno seguro para que no se dé la posibilidad de un ataque, una descohesión o una corrupción de los datos. Es por ello que la mejor manera de comunicar al usuario con el servidor es por medio de formularios que envían los datos necesarios a la API de la aplicación web. En concreto, se envía, dependiendo del tipo de operación y el tipo de datos, a una dirección concreta o *endpoint*.

Para una mejor experiencia de usuario, se ha decidido realizar estas operaciones de la manera más transparente posible, para el usuario. Es decir, los formularios habituales, envían pares de claves y datos y producen una recarga en la página para poder dar una respuesta al usuario. Un método de evitar estas molestas recargas para cualquier situación, es mediante el uso de llamadas AJAX. Por sus siglas en inglés, AJAX significa JavaScript y XML asíncrono (Asynchronous JavaScript And XML). Esta es una técnica que, por medio de una llamada AJAX, permite enviar desde un *script*, unos pares de claves y datos como con un formulario y recibir, en el mismo *script*, una respuesta del servidor en formato XML. Sin embargo, a día de hoy es más común la utilización del formato JSON. Con esta respuesta, se puede manipular el DOM para notificar del completado de la operación.

Conociendo todos estos datos, se ha creado una librería con un conjunto de operaciones relevantes para el buen funcionamiento de la aplicación. Estas van a ser llamadas desde el *frontend* para su procesado en el *backend* y su posterior devolución al *frontend*, una respuesta en formato JSON. Todas ellas se encuentran en el directorio “ajax_calls” y todas ellas siguen un mismo procedimiento. Este procedimiento sigue siempre tres fases muy bien diferenciadas:

1- Inicialización de datos: En una primera fase, se obtienen todas las variables que han sido enviadas por medio del protocolo HTTP con la operación POST y se crean un par de variables:
   - Response: Se trata de un mapa que contendrá siempre una clave “success” que se inicializa al valor lógico TRUE. Este será el objeto en formato JSON que devolverá el servidor al final de la operación.
   - Errors: Este es un *array* que contendrá los diferentes errores representados como cadenas muy concretas. En un principio, se inicializa vacío.

2- Lógica del servidor: En la segunda fase, se hacen todas las operaciones necesarias para completar la tarea en cuestión. Pueden hacerse desde comparaciones lógicas hasta modificaciones en la base de datos y cualquier otro posible tratamiento de los datos de entrada necesario. Mientras se realiza este proceso, si surge algún tipo de error o no se cumple alguna de las comprobaciones hechas, se refleja en el *array* de errores como una nueva entrada de texto con un código representativo del error concreto en formato kebab-case. Este formato consiste en la representación de cadenas de texto en minúsculas y separando las palabras con guiones.

3- Comprobación y devolución: Una vez se ha ejecutado toda la lógica, los últimos pasos son siempre los mismos. Se comprueba si hay errores o no. En caso de que los haya, en el mapa de “response”, la clave de “success” se cambia al valor lógico de FALSE y se incluye una nueva clave llamada “errors” con el *array* “errors” como valor. En caso de que no hubiera ninguno, si se trata de una actualización, una creación o un borrado de datos, el mapa “response” no es alterado. En caso de que fuera una lectura de datos, se añade una clave que varía dependiendo de la operación en concreto. Esta albergará los datos leídos. Como paso final de esta tercera fase, se transforma el mapa “response” a formato JSON y se devuelve.

Se puede poner como ejemplo, la operación de guardar una ubicación existente. Esta se hace por medio del *endpoint* “save_location.php”:

En la siguiente figura (ver Figura 43), se puede apreciar cada una de las 3 fases anteriormente explicadas en cajas de diferentes colores. En la segunda fase (color azul), se ha resaltado en cajas más pequeñas, la asignación de estas claves de error.

```php
<?php

require_once('../../../../wp-load.php');

$location_id = $_POST['location_id'];

$response = ['success' => true];
$errors = array();

if ('publish' == get_post_status(intval($location_id))) {
    $userCustomLocations = get_field('saved_custom_locations', 'user_' . get_current_user_id());

    $i = 0;

    $response['test'] = array(
        'id' => array(),
        'location_id' => intval($location_id)
    );
    if ($userCustomLocations) {
        while ($i < sizeof($userCustomLocations) && intval($userCustomLocations[$i]['location']['id']) != intval($location_id)) {
            $response['test']['id'][] = array(
                'the_id' => intval($userCustomLocations[$i]['location']['id']),
                'comparison' => intval($userCustomLocations[$i]['location']['id']) == intval($location_id)
            );
            $i++;
        }
    } else {
        $userCustomLocations = array();
    }

    if ($i == sizeof($userCustomLocations)) {
        $userCustomLocations[sizeof($userCustomLocations)] = array(
            'location' => array(
                'id' => $location_id,
                'name' => null,
                'coordinates' => array(
                    'lat' => '',
                    'long' => ''
                )
            )
        );
        update_field('saved_custom_locations', $userCustomLocations, 'user_' . get_current_user_id());
    } else {
        $errors[] = 'location-already-saved';
    }
} else {
    $errors[] = 'location-not-exists';
}

if (sizeof($errors) != 0) {
    $response['success'] = false;
    $response['errors'] = $errors;
}

echo json_encode($response);
```

Figura 43: Endpoint para llamadas AJAX save_location.php. Fuente: (Elaboración propia)

La totalidad de las llamadas AJAX creadas se pueden ver a continuación con nombres descriptivos de sus operaciones:

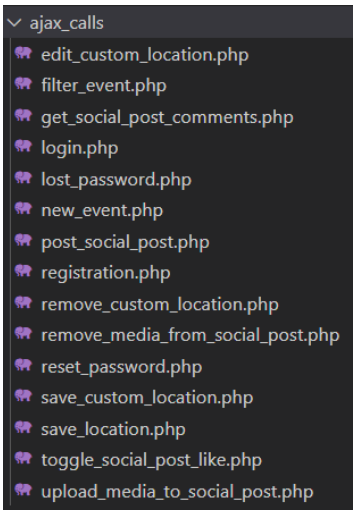


Figura 44: Conjunto de llamadas AJAX del proyecto. Fuente: (Elaboración propia)

Para algunas de las llamadas asíncronas, como es el caso de "filter_event.php", fue necesario la confección y ejecución de consultas SQL debido a la limitada personalización que permiten las consultas nativas de WordPress. Estas se definen como un conjunto de claves y valores anidados que se denominan WP Queries. En su lugar, para alcanzar los objetivos planteados, el filtrado de eventos tuvo que hacerse por medio de consultas complejas a las tablas nativas de la estructura de base de datos de WordPress. Se confeccionaron como cadenas de texto a las cuales se les iba anidando otras cadenas condicionales, dependiendo de ciertos parámetros de entrada basados en los filtros especificados por los usuarios.

6.4 Consideraciones de seguridad

Como se está utilizando un CMS que ya cuenta con diversas medidas de seguridad, los aspectos más importantes a tener en cuenta van a ser relacionados con los ataques XSS o *Cross-site Scripting*. Estos ataques se dan cuando los datos que se envían a un servidor por medio de un formulario, no son comprobados antes de ser devueltos al usuario. Esto puede llevar a la inyección de código por medio de etiquetas de *script* en HTML, con fragmentos de código malicioso que pueden realizar modificaciones en la página web, instalar programas, robar datos, generar redirecciones a otros sitios maliciosos y muchas otras variantes (Lázaro, 2018). La forma de combatir estos, es realizando una operación de escapado. El escapado consiste en un procesado de los datos introducidos por el usuario para transformar, cualquier posible fragmento de código malicioso, en simple texto plano que no podrá ser interpretado por el navegador como etiquetas HTML. Esto se consigue por medio de la función **htmlspecialchars()** en PHP.

Otras medidas de seguridad tomadas, han sido a la hora de cambiar o resetear la contraseña por parte de un usuario. Este, tras solicitarla, recibe un enlace, por correo electrónico, con un parámetro que contiene una clave aleatoria generada y asociada a su cuenta. Esta cuenta solo podrá realizar el cambio de contraseña si la clave asociada al usuario, coincide con la proporcionada en el enlace.

Por otra parte, a nivel de las diferentes operaciones que se realizan desde el servidor, antes de envío de cualquier dato a un *endpoint* de la API creada, se comprueban por medio de *scripts*, estos datos para que concuerden en formato y no contengan ningún elemento malicioso. Estos datos

vuelven a ser comprobados por parte del servidor como medida adicional de seguridad. Se comprueba el formato y se escapan para evitar ningún tipo de filtración de seguridad.

Como ya se ha comentado, WordPress ya implementa medidas de seguridad como la encriptación de contraseñas por medio del algoritmo de cifrado md5. Este cifrado se puede conseguir por medio de la función **md5()** de PHP.

6.5 Base de datos común

En un principio, se decidió seguir un enfoque conservador con bases de datos locales, totalmente independientes. Sin embargo, al igual que nacieron dependencias entre el código de ambos TFGs, lo mismo acabó ocurriendo en términos de la base de datos.

El principal problema se dio cuando comenzamos a poblar con datos de prueba nuestras bases de datos locales. Cada uno supuso un formateado de los datos, por parte del otro que al final no acabó funcionando. Esto llevó a largas jornadas de investigación de la base de datos para encontrar las posibles diferencias. Una vez localizadas, optamos por utilizar una base de datos común y resolver el problema de formato de los datos.

El proceso partió de la creación de una base de datos nueva desde el panel de administración de la empresa de *hosting* contratada llamada Hostinger. Una vez se pudo acceder al panel de control de la base de datos por medio de la herramienta PHPMyAdmin, se creó una exportación de mi base de datos local tras haber limpiado todos los registros de prueba que había creado. Se generó un archivo en formato .sql de exportación, y desde PHPMyAdmin, se ejecutó el fichero, importando así toda la configuración previa de WordPress, de los *plugins*, de los *post types* personalizados y de los usuarios. El último paso fue crear usuarios con accesos a la base de datos y cambiar los parámetros pertinentes en el fichero “wp-config.php” para que apuntaran a la nueva base de datos.

Este cambio ha supuesto una mayor coherencia entre ambos trabajos. Por otra parte, ha supuesto una ralentización en la carga de la aplicación debido al acceso a una base de datos que no se encuentra ubicada en el mismo servidor en el que se encuentra la aplicación. Esto ha incrementado la espera de la carga de la página en 2 segundos aproximadamente durante cada carga. Sin embargo, no es un problema que nos preocupara ya que, una vez finalizado el desarrollo, el despliegue de la aplicación al mismo servidor en el que se tiene alojada la base de datos, soluciona este problema de retardo en las consultas a base de datos.

7. Pruebas

Toda aplicación debe ser sometida a un examen de sus funciones. Este ha de ser un proceso minucioso y sistemático que sea capaz de comprobar el correcto funcionamiento de las diferentes mecánicas de la aplicación. Estas comprobaciones han de probar todo tipo de entradas, desde correctas, hasta erróneas o incluso valores inesperados.

Este puede convertirse en un proceso muy extenso y puede ser difícil cubrir las diferentes casuísticas que pueden llegar a darse, teniendo en cuenta que, en este caso, el usuario introduce datos constantemente y estos han de ser acotados.

La herramienta que se ha decidido utilizar es Cypress. Esta dispone a las aplicaciones de un entorno de pruebas en el que crear test y probar su ejecución, paso por paso, en busca de cualquier posible fallo. Cypress se instala como un paquete de NodeJS por medio de npm ya que está basado en JavaScipt. Este se ejecuta con el comando "**npm run cypress open**". Este comando, abre una interfaz (ver Figura 45) desde la cual se pueden ejecutar un conjunto de pruebas genéricas o ejecutar otras creadas a medida para los diferentes requisitos de nuestra aplicación.

A continuación, se puede ver el test realizado para la comprobación del correcto filtrado de eventos:

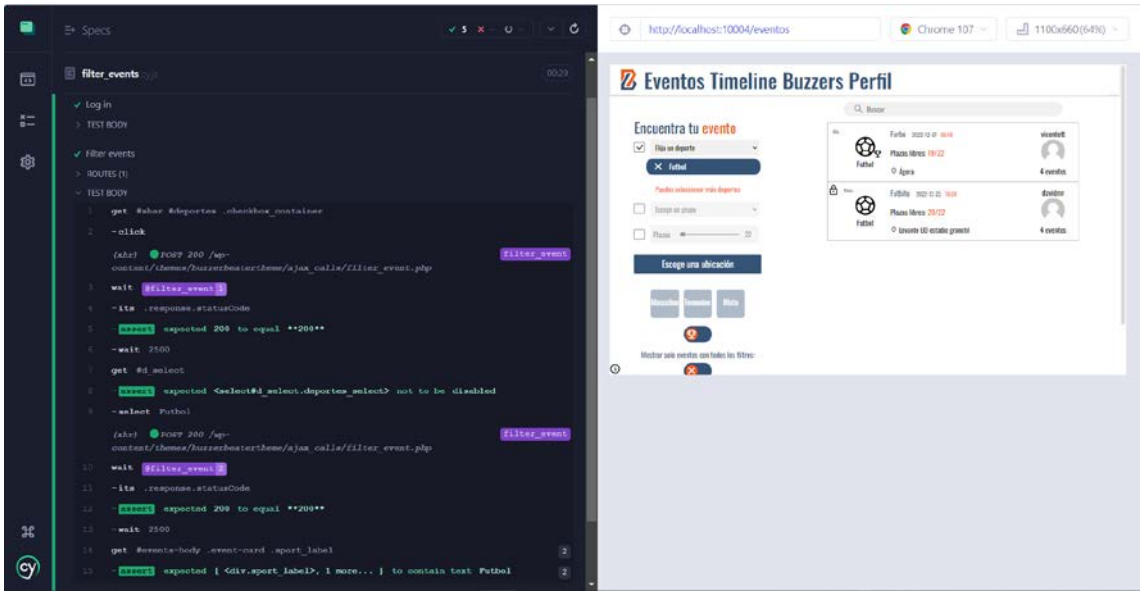


Figura 45: Interfaz de Cypress con la ejecución de una prueba finalizada y su vista previa. Fuente: (Elaboración propia)

En la anterior figura, se puede apreciar cómo se han ejecutado dos unidades de test que han sido creadas. Estas son "Log in" que se puede ver, por la marca de verificación verde, que se ha completado con éxito, y la unidad "Filter events". Esta última se ha desplegado para poder ver más en detalle el propósito de esta prueba. En primer lugar, se selecciona, por medio de una combinación de selectores de IDs y de clases, similar a los de JQuery, el filtro de deportes. A continuación, se espera a la primera llamada AJAX que actualiza el contenido del contenedor de eventos principal. Se puede apreciar que esta es una petición HTTP con el método POST y ha devuelto un código de éxito 200. El siguiente paso consiste en la comprobación de que el selector de deportes, ha dejado de estar deshabilitado y su inmediata selección de la opción de "Futbol".

Se vuelve a esperar a la respuesta de la misma llamada AJAX pero, esta vez, con el parámetro de filtrado de deporte con el valor de “Futbol”. Una vez ha finalizado la actualización de los campos, para lo cual se dejan 2,5 segundos de espera por los posibles retardos que suponen la base de datos remota, se pasa a comprobar que el filtrado ha tenido éxito y solo se muestran eventos cuyo deporte asignado es el futbol. El código de esta unidad en concreto, se puede apreciar a continuación por la legibilidad que presenta Cypress:

```
const fs = require('fs')
const cy = require('cypress')
let sports = []

describe('Log in and filter events by sport', () => {
    fs.readFile('./tests/filter_event.json', 'utf-8', (err, jsonData) => {
        if(err) {
            throw new Error('Unable to read test data file')
        }
        sports = JSON.parse(jsonData).sports
    })
    it('Log in', ()=>{···
    })

    it('Filter events', () => {
        cy.intercept('POST', 'wp-content/themes/buzzerbeatertheme/ajax_calls/filter_event.php').as('filter_event')
        sports.forEach(sport => {
            cy.get('#deportes .checkbox_container').click()
            cy.wait('@filter_event').is('response.statusCode').should('eq', 200).wait(2500)
            cy.get('#d_select').should('not.be.disabled').select(sport)

            cy.wait('@filter_event').its('response.statusCode').should('eq', 200).wait(2500)
            cy.get('#events-body .event-card .sport_label').should('include.text', sport)
            cy.get('.sport_filter .remove').click()
            cy.wait('@filter_event').is('response.statusCode').should('eq', 200).wait(2500)
        });
    })
})
```

Figura 46: Fichero de prueba de Cypress para el filtrado de eventos por deporte. Fuente: (Elaboración propia)

Estos test son interactivos y permiten la inspección del estado del DOM en cualquiera de los distintos puntos del test (ver Figura 45).

Estos test se han realizado sobre diversas de las tareas más relevantes, probando los diferentes mensajes de error que se han creado para todos los posibles valores de entrada. Las tareas que han sido probadas han sido las que se ven a continuación:

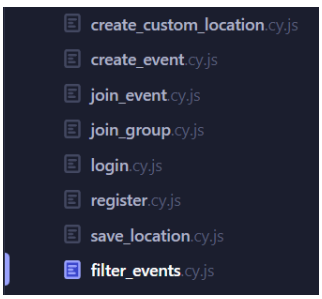


Figura 47: Conjunto de pruebas creadas en Cypress para el proyecto. Fuente: (Elaboración propia)

El diseño de los test se han realizado en base a los flujos de interacción planteados en la fase de diseño (ver Figura 21). Se ha simulado, por medio de la API que proporciona Cypress, la

interacción necesaria, por parte de un usuario, en las diferentes operaciones de la aplicación. Estos test se han diseñado con un listado de parámetros de prueba sobre los que se ha iterado (ver Figura 46). El fichero que contiene los datos de entrada se importa directamente en el archivo de test en formato JSON (ver Figura 48) para ser convertido a un objeto de tipo *Array* de JavaScript. En la siguiente figura se encuentran datos correctos de entrada:

```
{
    "sports" : [
        "Futbol",
        "Tenis",
        "Baloncesto",
        "Pin pon",
        "Padel",
        "Natación",
        "Voleibol"
    ]
}
```

Figura 48:Fichero con datos de entrada de prueba filter_event.json

8. Implantación

Este capítulo denota la finalización del desarrollo de Buzzer Beater y se centra en los pasos finales para su correcto despliegue en un entorno público.

8.1 Despliegue

Para el correcto despliegue de la aplicación web desarrollada, se ha optado, como ya se adelantó, por la utilización del *plugin* Duplicator. Este facilita las tareas de traspaso de base de datos y *plugins* ya que crea una copia idéntica, modificando solamente rutas y credenciales.

Para comenzar, se ha creado un subdominio en el servidor compartido de Hostinger bajo el nombre de http://buzzerbeater.millandavid.com. Este subdominio no se ha dotado de certificación SSL expedido por una autoridad certificadora por el sobrecoste que suponía.

A continuación, se ha creado una base de datos desde el panel de control de Hostinger.

El siguiente paso consiste en crear un directorio que aloje la aplicación web, en el servidor. A continuación, se procede al enlazado del subdominio con el directorio que acaba de crearse.

Una vez disponemos de un entorno preparado, hay que crear una exportación de nuestra aplicación desde el panel de Duplicator.

Esta exportación incluye un archivo comprimido con todos los ficheros y directorios de la web, una exportación de la base de datos y un archivo de instalación en PHP.

Tras completar todo el proceso de exportación, se accede con la herramienta FileZilla, por medio del protocolo de transferencia de archivos, FTP, al servidor de Hostinger contratado. Navegamos a la ubicación en la que se alojará nuestra aplicación web y subimos los ficheros generados por Duplicator. Una vez importados, el proceso de instalación se realiza accediendo a la dirección http://buzzerbeater.millandavid.com/installer.php. El despliegue de la web se hace de forma guiada. El único punto crítico, es la conexión con la nueva base de datos. La dirección, junto con las credenciales y el puerto de esta, han de ser introducidos durante el proceso de despliegue.

El último paso, como indica el instalador, es iniciar sesión en el panel de administración de WordPress para que se eliminen, de forma automática, todos los archivos residuales resultantes del despliegue.

Tras comprobar que todos los datos de los diferentes usuarios, *post types* y campos personalizados están intactos y que las páginas se ven correctamente, podemos dar por finalizado el despliegue y puede desinstalarse el *plugin* de Duplicator.

Un aspecto muy importante a tener en cuenta durante todo este despliegue, fue el mantener el subdominio protegido bajo contraseña. Si no se hubiera tenido este paso en cuenta, durante el despliegue, cualquiera podría haber accedido al fichero de instalación, realizar el proceso de despliegue y podría haberse llegado a conectar a una base de datos maligna.

8.2 Resultado final

A continuación, se muestran diversas imágenes del resultado final de la aplicación web tras el despliegue y poblado con datos de ejemplo:

Listado y filtrado de eventos

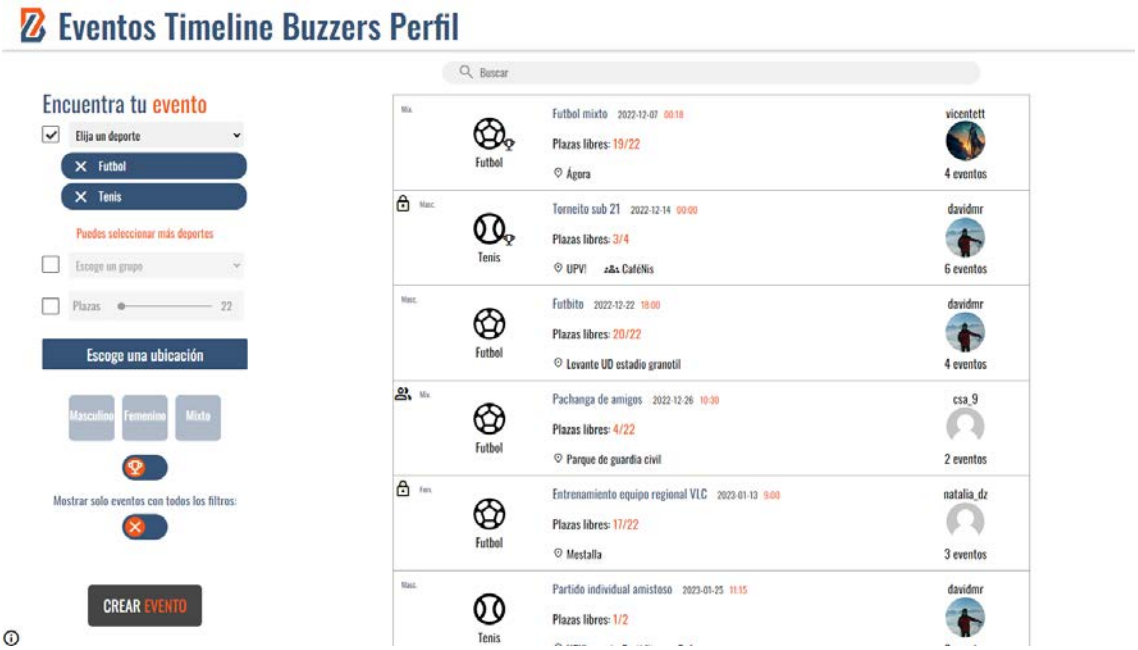


Figura 49: Listado y filtrado de eventos. Fuente: (Elaboración propia)

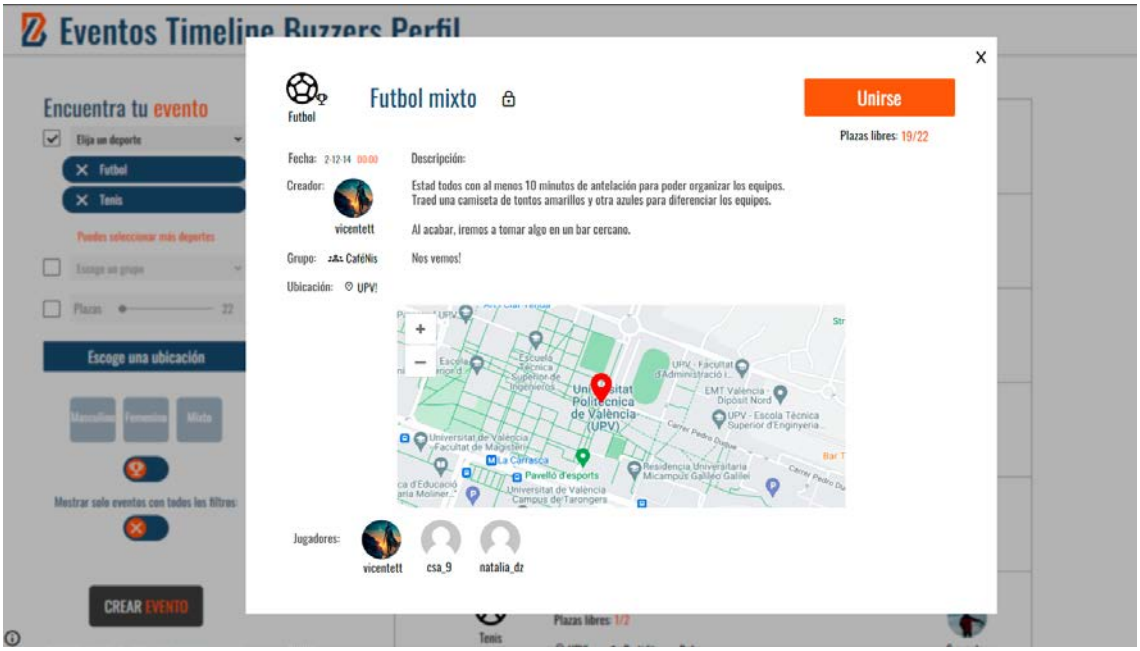


Figura 50: Detalles de evento. Fuente: (Elaboración propia)

Creación de eventos

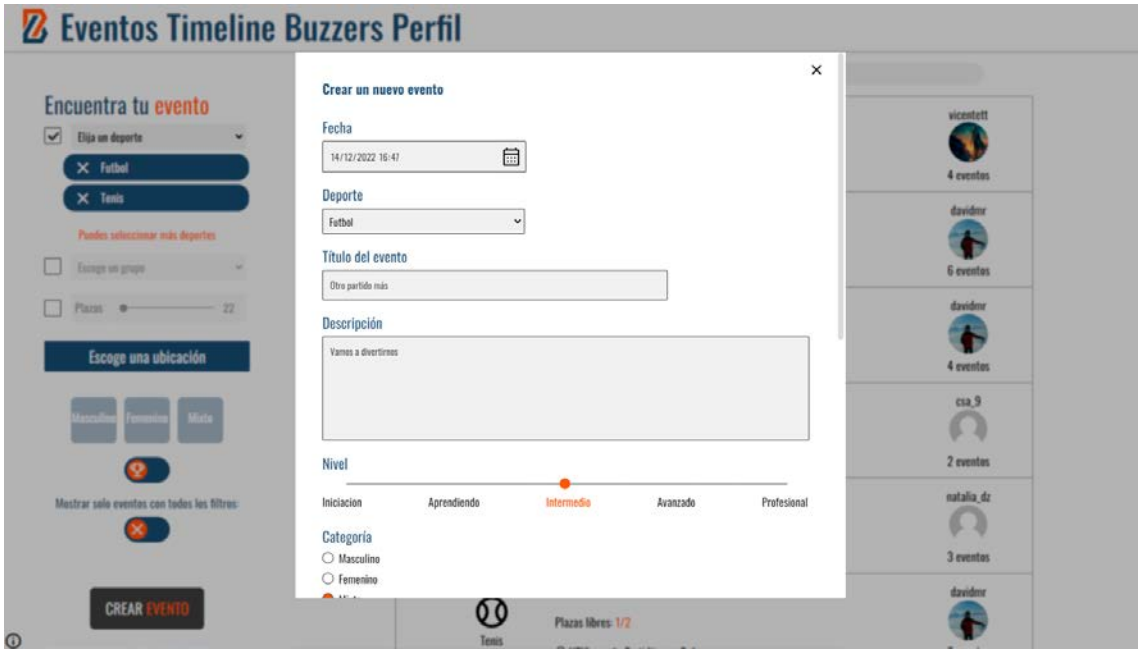


Figura 51: Menú de creación de evento 1/2. Fuente: (Elaboración propia)

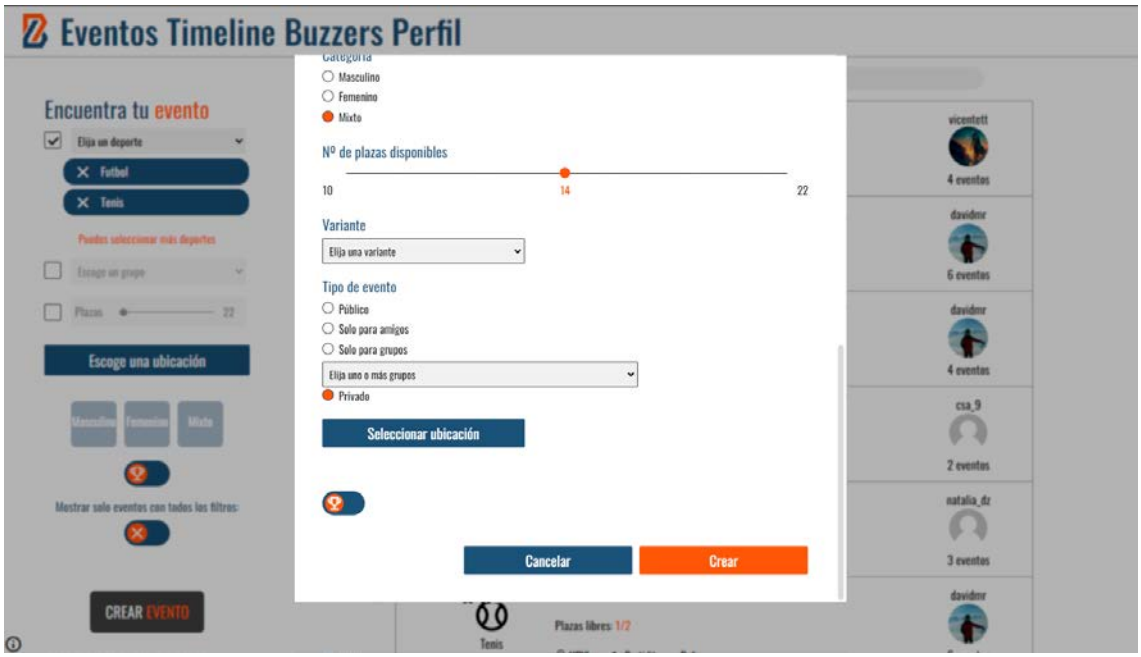


Figura 52: Menú de creación de evento 2/2. Fuente: (Elaboración propia)

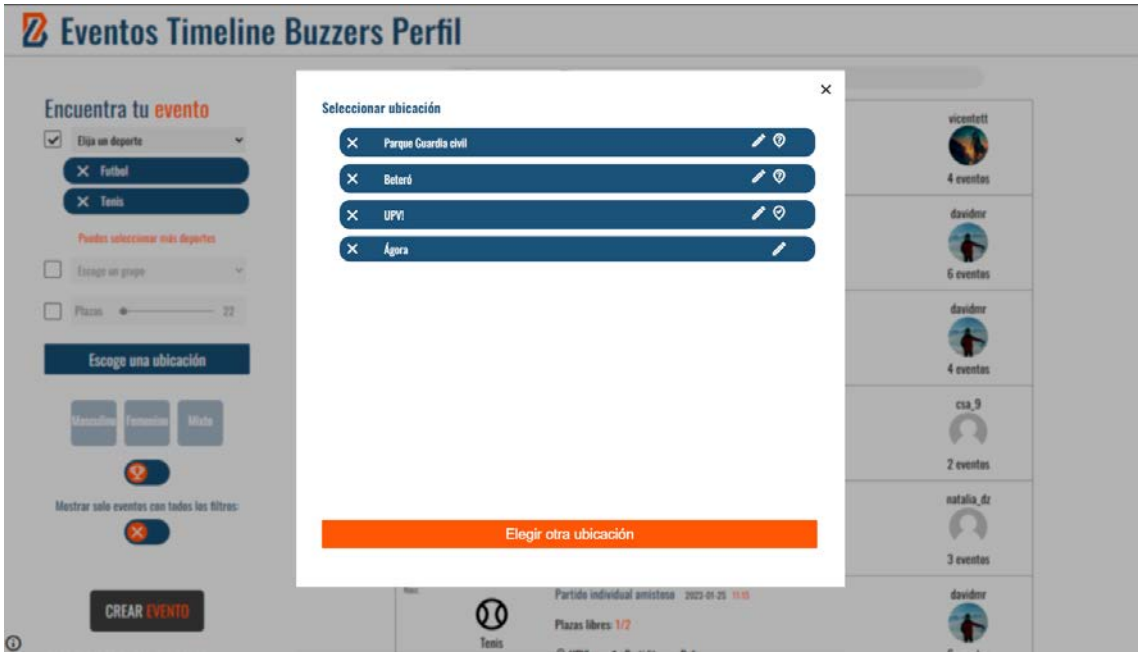


Figura 53: Menú de creación de evento - Selección de ubicación. Fuente: (Elaboración propia)

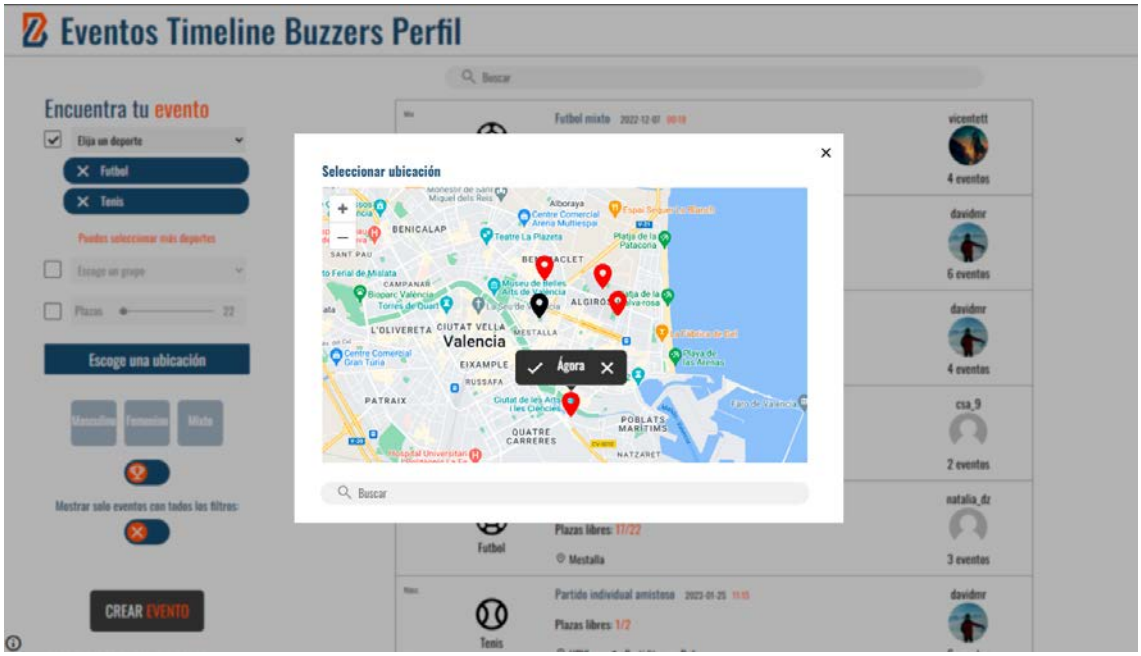


Figura 54: Menú de creación de evento - Selección de ubicación - Mapa. Fuente: (Elaboración propia)

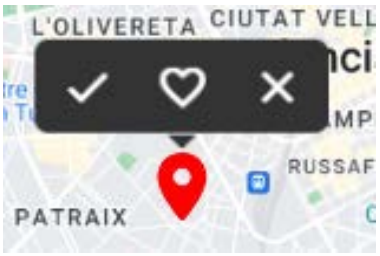


Figura 55: Desplegable de marcador en mapa de selección de ubicación. Fuente: (Elaboración propia)

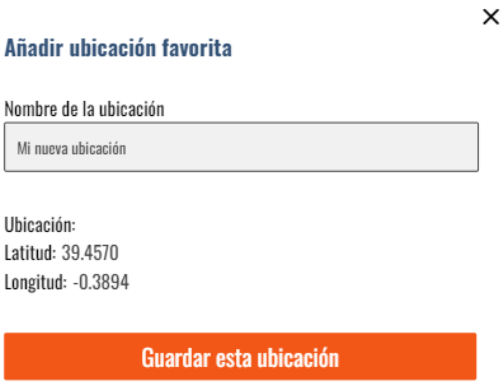


Figura 56: Menú de guardado de ubicación. Fuente: (Elaboración propia)

Perfil de grupo

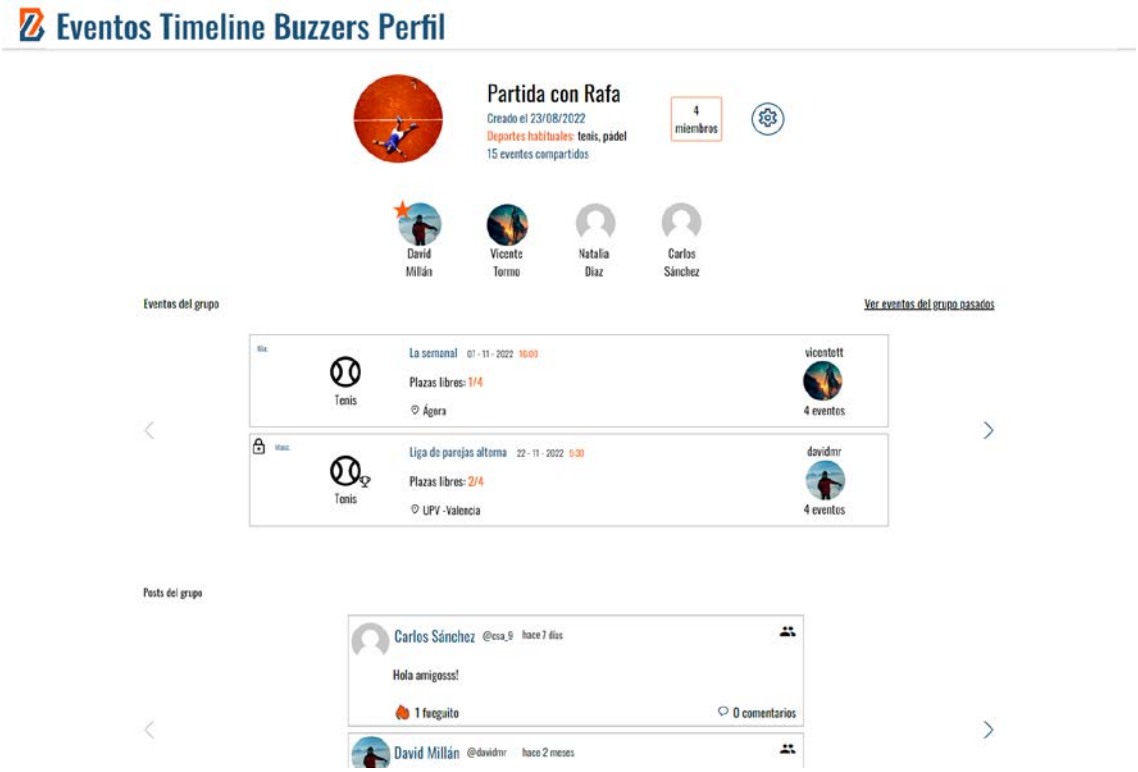


Figura 57: Perfil de grupo. Fuente: (Elaboración propia)

9. Conclusiones

En este apartado de cierre, se van a hacer una serie de reflexiones sobre este Trabajo de Fin de Grado, se va a relacionar, la materia visto en las clases, con las aplicaciones que se le han dado a lo largo de este y, por último, se hará una breve exposición de algunas de las ideas de mejora, modificaciones y acciones futuras que se quieren llevar a cabo con este proyecto.

9.1 Conclusiones

La realización de este proyecto, ha supuesto un reto a muchos niveles. A pesar de ello, me ha dado una gran satisfacción ver el progreso del mismo, su evolución, la planificación que se ha llevado y lo mucho que he aprendido por el camino.

En primer lugar, he aprendido a valorar la importancia que tiene una buena planificación inicial. Ha sido crucial para el correcto funcionamiento en paralelo, de mi compañero y mío, el establecimiento de tareas y *sprints*, la organización de reuniones semanales y la estructuración del repositorio.

Por otra parte, me ha supuesto, a nivel técnico, un gran aprendizaje y perfeccionamiento de mis conocimientos en PHP y JavaScript, sobre todo. Además, la utilización de una única API y una sola librería específica, me ha permitido tener un mayor control del proyecto al no depender de terceros. Este aspecto, también ha permitido que llegara a la resolución de problemas complejos, como pueden haber sido las consultas condicionales en SQL o las llamadas a APIs externas y el tratamiento de los datos devueltos. Ha sido muy satisfactorio también, poder resolver los problemas que se han generado a raíz de la base de datos, con los conocimientos que ya poseía de la carrera en materia de consultas en SQL.

9.2 Relación del trabajo desarrollado con los estudios cursados

Durante el transcurso del grado en Ingeniería Informática, muchas de las asignaturas que he tenido el placer de cursar, me han aportado los conocimientos necesarios para reunir todas las habilidades requeridas, para el desarrollo del actual proyecto. Las principales materias que han servido para este propósito han sido:

- **Bases de Datos** y **Sistemas de Información y Gestión de Bases de Datos**: En ambas asignaturas, se exploran el lenguaje de consultas de bases de datos SQL. Este ha sido vital para la creación de consultas complejas con sub-consultas anidadas y para la exploración de la base de datos común con el fin de resolver los problemas que se presentaron. Además, han ayudado a entender los esquemas de bases de datos para poder aprovechar al máximo la estructura de WordPress y utilizarla en nuestra ventaja.

- **Interfaces Persona Computador**: Gracias a esta asignatura, pudimos comprender la importancia que tienen las interfaces en la experiencia de usuario. Aprendimos a crear grupos de elementos relacionados y que toda la aplicación, que en su momento se desarrolló, mantuviera una coherencia. También fue de vital importancia la coordinación en equipo para el proyecto de esta y, aunque no era objeto de estudio de esta asignatura, llevamos a cabo un control de versiones mi compañero y yo para permitir el trabajo concurrente de ambos.

- **Ingeniería del Software**: A lo largo de esta asignatura, se propuso la creación, en equipo, de un programa que servía diversos propósitos. Gracias a ello, pudimos comenzar a dividir un proyecto en tareas y *sprints*. Para todo ello, se vio por primera vez el concepto de control de versiones y se crearon test para los diferentes componentes de la aplicación. Además, ayudó a estructurar un proyecto de *software* en diferentes capas independientes, para separar la capa lógica, la capa de datos y la interfaz.

- **Tecnologías de Sistemas de Información en la Red**: Una de las herramientas de mayor utilidad que se ha utilizado en este proyecto, han sido las posibilidades que ofrece NodeJS y todos los paquetes de utilidades disponibles hoy en día. En esta asignatura, se aprendieron las bases de NodeJS y conceptos más avanzados en JavaScript que han sido aplicados durante este TFG como la utilización de *callbacks*.

- **Gestión de Proyectos**: Esta asignatura nos presentó las herramientas y los conocimientos necesarios, para llevar a cabo un proyecto multidisciplinar en equipo y poder coordinar todo lo referente a este. La planificación de los recursos, la gestión del tiempo o la carga de trabajo, han sido algunos de los elementos clave que nos han acompañado a lo largo de todo el proyecto. Las herramientas que se nos presentaron y que más hemos utilizado han sido Trello y los diagramas de Gantt, que hemos podido implementar, directamente, desde Trello.

- **Gestión y Configuración de la Arquitectura de los Sistemas de Información**: Esta asignatura ha ayudado a comprender la importancia de la seguridad para evitar posibles ataques, tanto en aplicaciones web, como en máquinas propias. Además, en las prácticas, se configuró un servidor web basado en Apache desde cero. Se pudo configurar también un servicio de mensajería bajo el protocolo SMTP, la conexión con una base de datos y la creación de certificados SSL para la web que iría alojada en ese servidor. Estos conocimientos, contribuyeron en la configuración de diversos parámetros de la herramienta utilizada XAMPP y a la hora de ajustar diversos parámetros en el panel de administración del servicio de hosting compartido en Hostinger.

9.3 Trabajos futuros

En ningún momento, ni mi compañero ni yo hemos contemplado la posibilidad de abandonar este proyecto. Este TFG surge a raíz de una necesidad que nosotros mismos hemos experimentado y en la propuesta de un producto que nosotros mismos utilizaremos cuando esté totalmente desarrollado. Es por esto que, ya hemos acotado ambos TFGs para no extenderlos en exceso, pero, ya tenemos diferentes propuestas para continuar desarrollando funcionalidades para Buzzer Beater.

Los primeros pasos más inmediatos consistirán en las mejoras a nivel de optimización de búsquedas. A este campo suele ser referido como SEO, de sus siglas en inglés para optimización de motores de búsqueda. Consiste en una serie de medidas para optimizar los documentos HTML tanto a nivel de estructura como de contenido con el propósito de que esta sea mejor posicionada en los motores de búsqueda. Estas acciones pueden consistir en la inserción de metadatos descriptivos, en la optimización de etiquetas HTML, en reducción de los tiempos de carga, en la reducción del tamaño de los diversos recursos utilizados y muchas otras medidas.

Otros pasos a seguir más enfocados en el desarrollo de la misma, sería la creación de una API pública para la interacción de las diferentes empresas, polideportivos y otros interesados.

Bibliografía

Amazon Web Service. (s.f.). *AWS.* Recuperado el 24 de Noviembre de 2022, de ¿Qué es una API?: https://aws.amazon.com/es/what-is/api/

Dexter, S. (6 de Diciembre de 2021). *UX Collective.* Recuperado el 20 de Noviembre de 2022, de https://uxdesign.cc/figma-continues-to-skyrocket-63-reported-it-was-their-primary-ui-design-tool-in-2021-bb9390a8b96b

Drumond, C. (2021). *Atlassian.* Recuperado el 19 de Noviembre de 2022, de Scrum Aprende a utilizar scrum con lo mejor de él: https://www.atlassian.com/es/agile/scrum

Figma. (s.f.). *Figma.* Recuperado el 11 de Noviembre de 2022, de https://www.figma.com/design-systems/

Google. (s.f.). *Material Design.* Recuperado el 20 de Noviembre de 2022, de https://m3.material.io/foundations/

Joanne. (16 de Mayo de 2016). *Form Keep.* Recuperado el 17 de Noviembre de 2022, de Should You Use Single-Step or Multi-Step Forms?: https://blog.formkeep.com/should-you-use-single-step-or-multi-step-forms/

Lázaro, D. (2018). *Seguridad web en PHP.* Recuperado el 20 de Noviembre de 2022, de https://diego.com.es/seguridad-web-en-php

Marketers Group. (26 de Agosto de 2022). *Marketers Group.* Recuperado el 23 de Noviembre de 2022, de Diferencias entre Apache y Nginx: https://marketersgroup.es/diferencias-entre-apache-y-nginx/

Ministerio de Cultura y Deporte. (2020). *DeporteData.* Recuperado el 7 de Noviembre de 2022, de DeporteData. Base de datos: https://estadisticas.mecd.gob.es/DeporteJaxiPx/Datos.htm?path=/d12/f12/a2020/C03//l0/&file=H20F0301.px

Ministerio de Cultura y Deporte. (9 de Junio de 2021). *Consejo Superior de Deportes.* Recuperado el 7 de Noviembre de 2022, de Ministerio de Cultura y Deporte: https://www.csd.gob.es/es/la-practica-deportiva-aumenta-un-61-en-los-ultimos-cinco-anos

Osman, M. (29 de Abril de 2021). *Kinsta.* Recuperado el 20 de Noviembre de 2022, de Locos e Interesantes Datos y Hechos de WordPress (2022): https://kinsta.com/es/blog/wordpress-estadisticas/

Rubio, J. C. (27 de Febrero de 2019). *TreceBIts redes sociales y tecnología.* Recuperado el 20 de Noviembre de 2022, de Las cinco mejores apps para quedar con otros a hacer deporte: https://www.trecebits.com/2019/02/27/las-cinco-mejores-apps-para-quedar-con-personas-para-hacer-deporte/

Sinnaps. (2022). *Sinnaps.* Recuperado el 11 de Noviembre de 2022, de Metodología SCRUM: ¿qué es y cómo aplicarlo en tu trabajo?: https://www.sinnaps.com/blog-gestion-proyectos/metodologia-scrum

Trello. (s.f.). *Trello.* Recuperado el 10 de Noviembre de 2022, de https://trello.com/

Una vida online. (2021). *Estadísticas de uso de redes sociales en 2022 [España y mundo]*. Recuperado el 7 de Noviembre de 2022, de Una vida online: https://unavidaonline.com/estadisticas-redes-sociales/

W3Techs. (21 de Noviembre de 2022). *W3Techs.* Recuperado el 20 de Noviembre de 2022, de Historical quarterly trends in the usage statistics of web servers: https://w3techs.com/technologies/history_overview/web_server/ms/q

W3Techs. (2022 de Noviembre de 2022). *W3Techs.* Recuperado el 20 de Noviembre de 2022, de https://w3techs.com/technologies/comparison/js-angularjs,js-jquery

WordPress. (s.f.). *Codex.* Recuperado el 25 de Noviembre de 2022, de Database Description: https://codex.wordpress.org/Database_Description

WordPress. (s.f.). *Developer Resources.* Recuperado el 20 de Noviembre de 2022, de https://developer.wordpress.org/themes/basics/template-hierarchy/

Anexos

ANEXO A - ODS

OBJETIVOS DE DESARROLLO SOSTENIBLE

En el año 2015, la Asamblea General de las Naciones Unidas se congregó y llego al acuerdo de establecer 17 objetivos, con fecha límite 2030 para cumplirlos. Estos forman parte de la nueva agenda sostenible y tienen como propósito, erradicar la pobreza, fomentar el respeto y la inclusión, salvaguardar el planeta y lograr una sociedad más sostenible y concienciada con el planeta y las personas.

El grado de relación que presenta este TFG con los diferentes Objetivos de Desarrollo Sostenible (ODS) es el siguiente para cada uno:

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.		X		
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**	X			

El TFG que se presenta, está alineado con los siguientes Objetivos de Desarrollo Sostenible por los siguientes motivos:

- **Salud y bienestar:** La aplicación web Buzzer Beater, trata de facilitar y promover la práctica deportiva tanto en grupo como individual de forma colectiva. Esta motivación contribuye a la mejora de los hábitos de vida de los ciudadanos.

- **Reducción de desigualdades:** El deporte es un factor que promueve la socialización de personas sin importar el colectivo, la procedencia o la orientación sexual. Buzzer Beater va a suponer un medio de conexión de numerosas personas con diferencias para la práctica del deporte, con una gran facilidad.

- **Alianzas para lograr objetivos**: Con Buzzer Beater, la participación en eventos deportivos y la pertenencia a grupos, va a suponer un impulso para la colaboración y la reunión de miembros en ámbitos muy favorables para el crecimiento humano que supone la práctica de deporte colectivo.