# Documentación proyecto UFC

Joel Acoran Cruz Morales


## Proyecto UFC

Documento para el proyecto final de 2do curso del Ciclo formativo de grado superior en Diseño de Aplicaciones Web de Joel Acoran Cruz Morales

# 1. Descripción del proyecto

## a. Descripción

El proyecto consiste en una aplicación web interactiva centrada en la empresa UFC (Ultimate Fighting Championships), en esta permitirá a los usuarios explorar la información detallada de cada participante y además crear y gestionar eventos y peleas asociadas a ellos. Este enfoque nos va a brindar un control más eficiente sobre las próximas veladas y combates, incluyendo la fecha y ubicación de cada evento.

Para el uso, la eficacia y control de la información, nuestra aplicación se integrará con una API pública que accederá a datos almacenados en nuestra base de datos. Esto asegura que la información esté siempre sincronizada y actualizada en tiempo real con nuestra base de datos.

La modularidad de esta aplicación permitirá su expansión hacia diferentes áreas deportivas,ayudando así a ser una aplicación valiosa y versátil para la gestión y organización de eventos deportivos en general.

Es decir, que el mismo sistema podría aplicarse a diferentes federaciones o ligas ya sea de lucha o de otros eventos deportivos, proporcionando una base centralizada y eficiente para la administración y promoción de dichos eventos deportivos de las diferentes disciplinas que se requiera utilizar.

## b. Características principales

### i. Login

- Los usuarios contarán con un login en el que nos dará unas credenciales y nos proporcionará una seguridad y un acceso a todas las funciones.
- Sin estas credenciales no se podrá acceder a ninguna función y tampoco crear ningún evento o pelea.

### ii. Exploración de participantes

- Los usuarios podrán navegar y buscar a través de nuestra vista, los datos de todos los participantes.
- Cada luchador tendrá un perfil detallado que incluirá información como nombre, apellido, rol, técnica, altura, peso, país y a que categoría pertenece.
- Está implementado un sistema de filtrado en el que facilita la búsqueda de participantes por nombre o apellidos , categoría, técnica o visualizar los árbitros y jueces existentes.

iii. **Creación y gestión de veladas**

- Los usuarios logueados podrán crear eventos, editarlos e incluso eliminarlos.
- Para cada velada, los usuarios podrán asignar nombre, fecha y ubicación.
- Se permitirá la personalización de cada evento por si se tuviera que trasladar de ubicación o de fecha.

iv. **Creación y gestión de peleas**

- Los usuarios logueados podrán crear las peleas y asignarlas a eventos previamente creados.
- Para cada pelea, los usuarios deberán añadir nombre de la pelea, seleccionar categoría, participantes, jueces y árbitros y además seleccionar en el evento donde se celebrará.
- Se permitirá la personalización de cada pelea por si algún luchador sufre alguna baja o si se tuviera que cambiar de evento por imposibilidad de fecha por ambos participantes.

v. **Integración de la API con la base de datos**

- Se creará y se utilizará una API pública de nuestro proyecto para obtener los datos actualizados de los participantes y de todos los componentes de nuestro proyecto.
- La aplicación estará sincronizada con la API para mantener la información actualizada.

# 2. Desarrollo del proyecto

## a. Definición de requisitos

### i. Requisitos funcionales

1. Autenticación y autorización.

El sistema debe permitir a los usuarios registrarse y acceder con credenciales para acceder a las funciones de la aplicación. El sistema verificará la autenticidad de los usuarios antes de permitirles el acceso mediante un token.

2. Exploración de participantes .

   La aplicación debe permitir a los usuarios buscar y navegar por la información de cada participante, incluyendo nombre, rol, técnica entre otros.

3. Creación y gestión de eventos.

   La aplicación deberá permitir a los usuarios logueados poder crear, editar y eliminar eventos, incluyendo nombre, fecha de cada evento y su localización.

4. Creación y gestión de peleas.

   La aplicación deberá permitir a los usuarios logueados poder crear, editar y eliminar peleas y asignarlas a eventos previamente creados, incluyendo la categoría y cada participante, jueces y árbitros.

5. Integración con la API.

   La aplicación estará integrada con una API pública creada que accede a la base de datos para obtener los datos actualizados de los participantes y componentes del proyecto.

### ii. Requisitos no funcionales

1. Seguridad.

   La aplicación garantizará la seguridad de los datos y la autenticación de los usuarios .

2. Escalabilidad

   La aplicación debe ser escalable para adaptarse a un aumento en el número de usuarios y eventos.

3. Usabilidad.

   La aplicación debe ser fácil de usar y navegar para los usuarios.

4. Rendimiento.

   La aplicación tendrá un rendimiento óptimo para garantizar una experiencia de usuario fluida.

### iii. Requisitos de la base de datos

1. Diseño de la base de datos.

   La base de datos estará diseñada para inserción de datos mediante forma segura para que no se dupliquen los datos y además

deberá estar diseñada para almacenar información de todos los componentes. Además de ser escalable, protegida y segura.

2. Integración con la API.

La base de datos será accesible a través de la API para obtener los datos , modificarlos e insertarlos.

**iv. Requisitos de interfaz de usuario**

1. Diseño de la interfaz de usuario.

La interfaz de usuario será fácil de usar y navegar para los usuarios, siendo intuitiva y proporcionando vista clara para cada sitio.

2. Accesibilidad.

La interfaz de usuario será accesible para todo tipos de usuarios.

**v. Requisitos de la API**

1. Diseño de la API.

La API será diseñada para proporcionar el acceso a la base de datos y obtener datos actualizados de los participantes y componentes del proyecto.

2. Autenticación y autorización.

La api estará implementada con autenticación y autorización para garantizar la seguridad de los datos.

## b. Análisis y diseño

El proceso de análisis y diseño de este proyecto fue un proceso crucial para el desarrollo del mismo, se llevó a cabo un análisis detallado de los requisitos funcionales del sistema, para identificar las necesidades para nuestro proyecto. A continuación, se presentará el diseño del sistema, que incluye el esquema de entidad relación y su diagrama, el diseño de la base de datos con sus tablas interconectadas y el modelo UML.

### i. Esquema entidad relación

En este proyecto se ha realizado un esquema que muestra las entidades, atributos y relaciones que existen entre ellas. En este proyecto, se ha diseñado una estructura como parte principal del diseño para posterior entender la estructura de la base de datos de manera eficiente. A continuación veremos las relaciones de cada una y por último su diagrama.

- Participante - Categoría (Un participante pertenece a una sola categoría y una categoría puede tener varios participantes).


*Figura de Diagrama de entidad y relación entre Participante y Categoría*

- Participante - Pelea (Un participante puede participar en múltiples peleas y una pelea puede involucrar a múltiples participantes).


*Figura de Diagrama de entidad y relación entre Participante y Pelea*

- Localización - Velada (Una localización puede tener múltiples veladas, pero cada velada solo puede tener una localización).


*Figura de Diagrama de entidad y relación entre Localización y Velada*

- Velada - Pelea (Una velada puede tener múltiples peleas, pero cada pelea solo puede pertenecer a una velada).


*Figura de Diagrama de entidad y relación entre Velada y Pelea*

- País - Participante (Un País puede tener múltiples participantes, pero cada participante pertenece a un país).

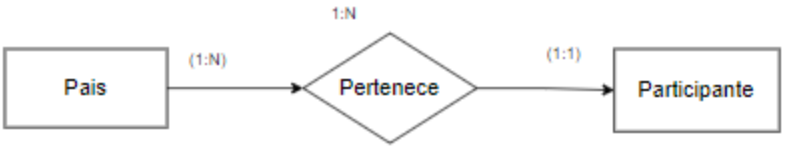


*Figura de Diagrama de entidad y relación entre País y Participante*

- Rol - Participante (Un Rol puede tener muchos participantes, pero cada participante tiene solo un rol).


*Figura de Diagrama de entidad y relación entre Rol y Participante*

- Técnica - Participante (Una Técnica es utilizada por múltiples participantes, pero un participante usa una técnica).


*Figura de Diagrama de entidad y relación entre Técnica y Participante*

- Usuario - Validación ( Un usuario obtiene una validación, la validación es obtenida por el usuario).

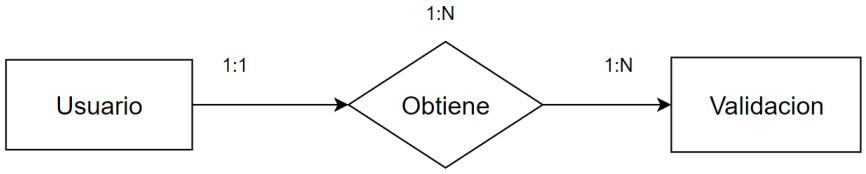


*Figura de Diagrama de entidad y relación entre Usuario y Validación*

- País - Localización (Un país pertenece a una localización y una localización pertenece a un país).

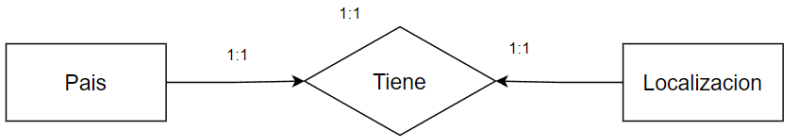


*Figura de Diagrama de entidad y relación entre País y Localización*

En estos fragmentos del esquema de entidad-relación del proyecto UFC presentan las entidades y relaciones entre ellas que componen el sistema. Las entidades que se han creado después de una investigación han sido Participante, Categoría, Pelea, Localización, Velada, País, Rol, Técnica y

Usuario. Las relaciones entre estas entidades se establecen a través de las relaciones de uno a uno, uno a muchos y muchos a muchos.

Este esquema es fundamental para entender la estructura de la base de datos y cómo se relacionan entre sí las diferentes entidades. A continuación, se presentará el diagrama de entidad-relación completo que muestra las entidades y relaciones entre ellas.

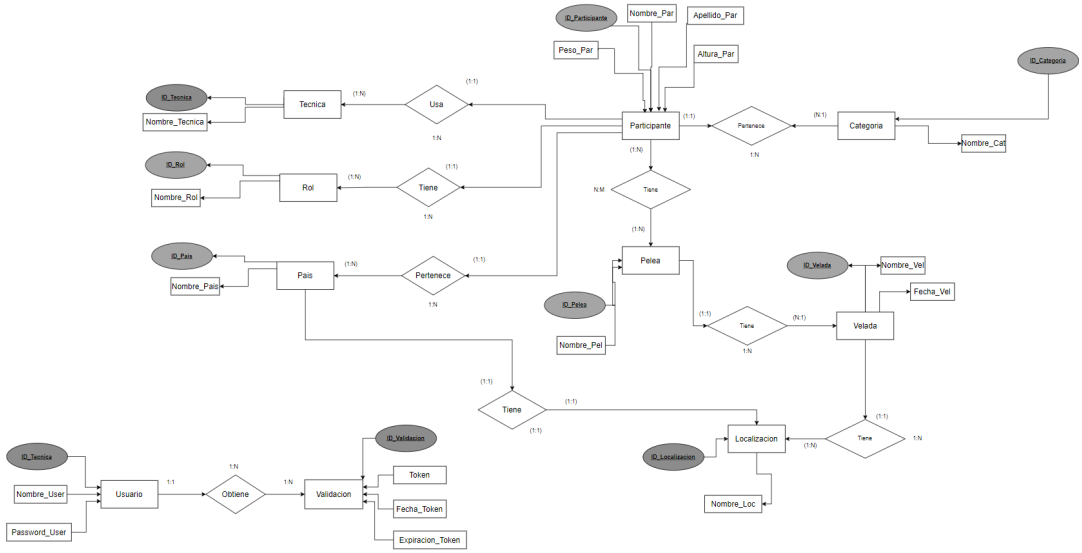


*Figura de Diagrama de entidad y relación*

Este diagrama muestra la estructura y cómo se relacionarán entre sí, lo que permitirá una mejor comprensión de la lectura de la base de datos que veremos en el siguiente punto.

## ii. introducción para diagrama de la base de datos

La base de datos es el componente principal después de realizar el modelo de entidad-relación. En este proyecto, se diseñó la base de datos que almacena la información estática como dinámica de algunos componentes, a continuación se mostrará el diagrama de nuestra base de datos y la explicación de esta misma.

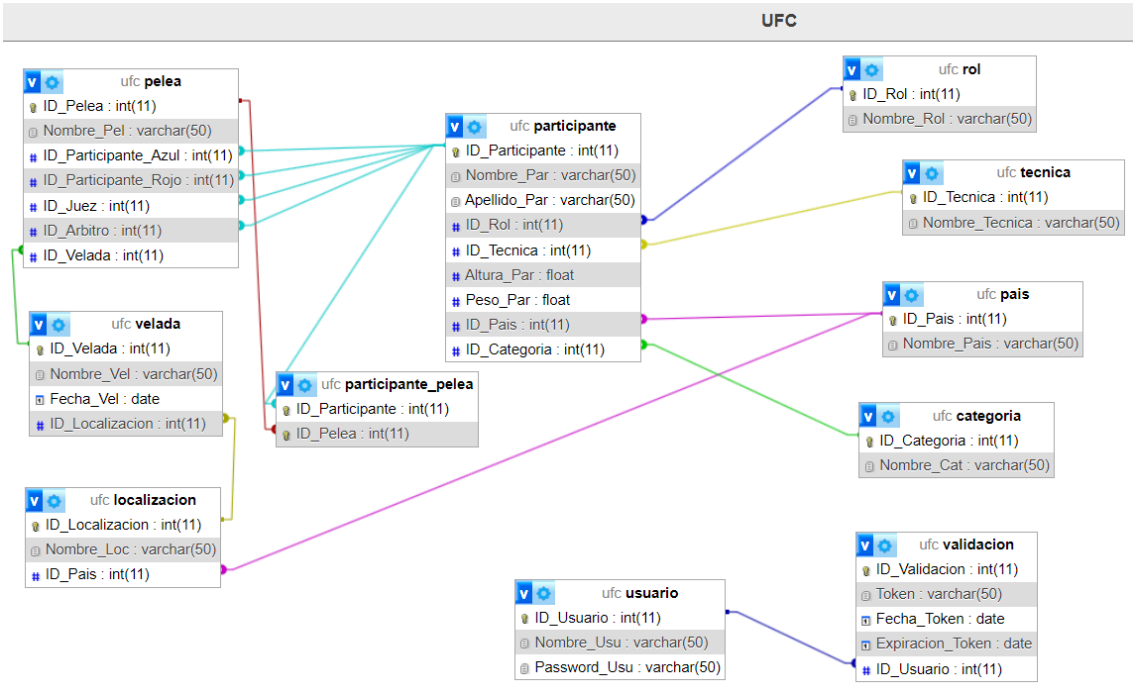


*Figura de Diagrama de base de datos*

Como podemos observar, nuestra base de datos consta de 11 tablas, las esenciales para que nuestra aplicación funcione correctamente, las tablas son:

1. **Pelea:** Consta de su id como primary key, nombre de pelea, id del participante azul y rojo, id del juez y el árbitro y como foreign key el id de la Velada .
2. **Velada**: Consta de su id como primary key, nombre de la velada, fecha de ella y como foreign key el id de Localización.
3. **Localización**: Consta de su id como primary key, nombre de localización y como foreign key el id del País.
4. **Participante Pelea**: Esta es una tabla intermedia en la que lleva el id de participante e id de Pelea.
5. **Participante**: Consta de su id como primary key, nombre de participante, apellido, peso , altura y como foreign keys lleva el id de Rol, Técnica, País y Categoría.
6. **Rol**: Consta de su id como primary key, y el nombre del rol.
7. **Técnica**: Consta de su id como primary key, y el nombre de la técnica .
8. **País**: Consta de su id como primary key, y el nombre del país.

9. **Categoría**: Consta de su id como primary key, y el nombre de la categoría.
10. **Usuario**: Consta de su id como primary key, el nombre del usuario y su contraseña.
11. **Validación**: Consta de su id como primary key, token , fecha de creación y de expiración y como foreign key el id del usuario.

De todas estas tablas, sólo se hacen el crud completo en la tabla Pelea y Velada. En otra solo se hace una inserción de datos automáticamente cuando el usuario inicia sesión crea un token para poder realizar la validación y tenga acceso a los datos.

## iii. introducción para Modelo uml

En este proyecto se ha diseñado un modelo UML en el que se representa la estructura y el comportamiento de las clases y el uso en sí de ellas. A continuación, se presentará el modelo UML y la explicación del mismo:

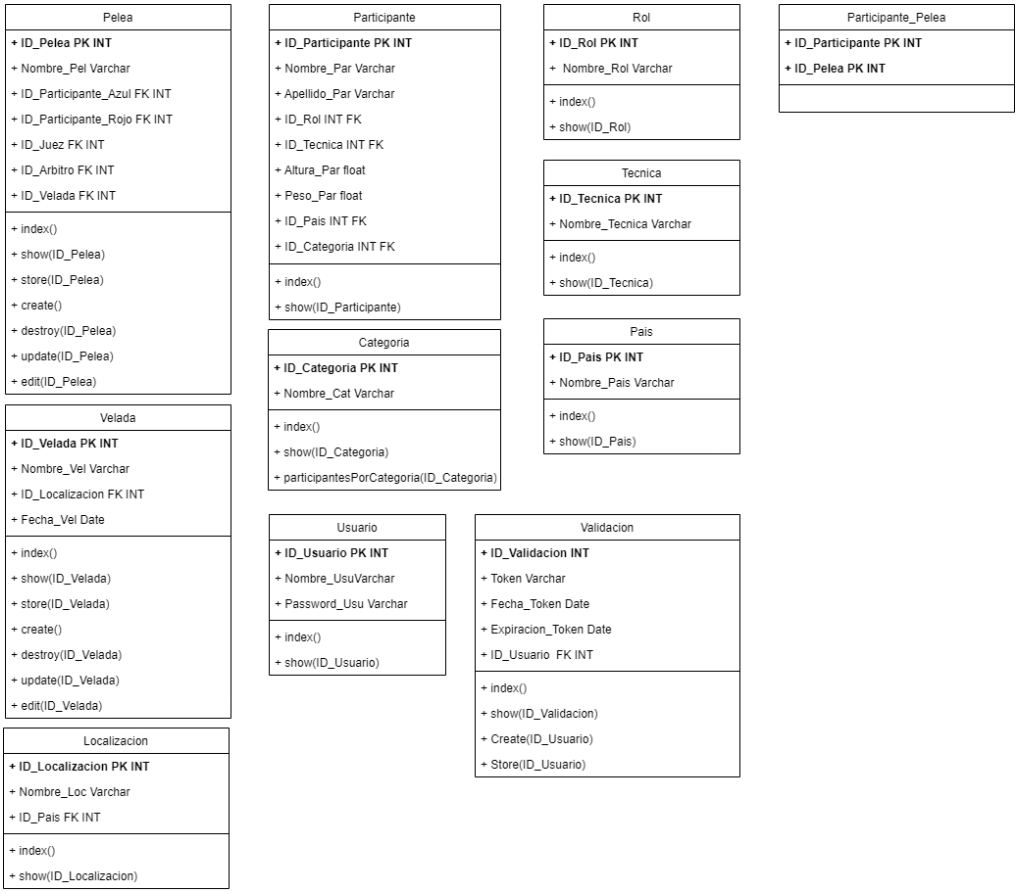


*Figura de modelo de UML*

Como podemos observar en cada clase que hemos definido, tienen diversidad de funciones según la utilización que deseemos en nuestro proyecto, vamos a explicar cada una de ellas:

1. **Pelea**: Esta clase consta de un crud completo, para ver con las funciones de index y show, crear con las funciones store y create , editar con las funciones update y edit y eliminar con la función destroy.
2. **Velada**: Esta clase consta de un crud completo, para ver con las funciones de index y show, crear con las funciones store y create , editar con las funciones update y edit y eliminar con la función destroy.
3. **Localización**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
4. **Participante**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
5. **Categoría**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno y además una función de

filtrado llamado participantesPorCategoria que consigue ver los participantes que pertenecen a categorías determinadas.

6. **Usuario**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
7. **Rol**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
8. **Técnica**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
9. **País**: Esta clase consta de una funcionalidad solo de vistas, por ello solo utilizaremos el index para conseguir todos los campos el show para recoger solo uno.
10. **Validación**: En esta clase se realizará un crud, pero no completo, solo se utilizará la función index para la vista de todo, el show para ver un único token, create y el store que va junto al id de usuario cuando inicia sesión .

Con todas estas clases y funciones lograremos que nuestro proyecto funcione de forma óptima y de la forma que queremos principalmente.

# c. Implementación

## i. Transformado de diseño en código

Para la creación de nuestro proyecto una vez tenemos el UML en el que vamos a trabajar y el diagrama de entidad-relación, crearemos nuestro proyecto con el método script safe. En este caso al entregar la query solo se hará un ejemplo de la inserción a la base de datos de una sola tabla contando con la creación del proyecto, creación de la tabla y su inserción de datos.

### 1. Creación de la base de datos

Para la creación la realizaremos con el método safe, primero comprobando si existe la base de datos o si no, la creamos.

```sql
-- Crear la base de datos UFC si no existe
CREATE DATABASE IF NOT EXISTS UFC;

-- Seleccionar la base de datos UFC
USE UFC;
```

*Figura de creación de base de datos*

## 2. Creación de tabla

Para la creación de la tabla, la realizaremos con el método safe haciendo así su fácil uso para evitar duplicidad de datos. Comprobaremos antes de crear la tabla si ya existe o sino crearla y si ya existe, añadirle nuevos campos si los hubiera. En este caso añadiremos el ID del participante como primary key, nombre, apellido , rol, técnica, altura, peso , id del país y también id de categoría .

```sql
-- Crear tabla Participante
CREATE TABLE IF NOT EXISTS Participante (
    ID_Participante INT AUTO_INCREMENT PRIMARY KEY,
    Nombre_Par VARCHAR(50),
    Apellido_Par VARCHAR(50),
    ID_Rol INT,
    ID_Tecnica INT,
    Altura_Par FLOAT,
    Peso_Par FLOAT,
    ID_Pais INT,
    ID_Categoria INT
);
```

*Figura de creación de la creación de la tabla Participantes*

### 3. Añadir claves foráneas

Además luego le añadiremos las claves foráneas para evitar problemas con los valores. En este caso añadiremos las claves foráneas que están en nuestra tabla al hacerlo de esta forma nos encargaremos de que no haya redundancia de datos y evitar errores en un futuro.

```sql
ALTER TABLE Participante
    ADD FOREIGN KEY (ID_Categoria) REFERENCES Categoria(ID_Categoria) ON UPDATE CASCADE ON DELETE CASCADE,
    ADD FOREIGN KEY (ID_Rol) REFERENCES Rol(ID_Rol) ON UPDATE CASCADE ON DELETE CASCADE,
    ADD FOREIGN KEY (ID_Tecnica) REFERENCES Tecnica(ID_Tecnica) ON UPDATE CASCADE ON DELETE CASCADE,
    ADD FOREIGN KEY (ID_Pais) REFERENCES Pais(ID_Pais) ON UPDATE CASCADE ON DELETE CASCADE;
```

*Figura de añadir claves foráneas a la tabla*

### 4. Inserción de datos

Para la inserción de datos la realizaremos con el método safe para a la hora de replicar la query nunca nos duplique los datos estáticos ya predefinidos. Los datos de las claves foráneas deberán estar previamente creadas para que no nos de problemas.

```sql
INSERT IGNORE INTO Participante (ID_Participante, Nombre_Par, Apellido_Par, ID_Rol, ID_Tecnica, Altura_Par, Peso_Par, ID_Pais, ID_Categoria) VALUES
-- Peso Mosca
(1, 'Brandon', 'Moreno', 1, 7, 1.64, 54, 12, 1),
(2, 'Amir', 'Albazi', 1, 3, 1.65, 57, 21, 1),
(3, 'Brandon', 'Royval', 1, 5, 1.75, 56, 1, 1),
(4, 'Alexandre', 'Pantoja', 1, 1, 1.65, 57, 2, 1),
```

*Figura de inserción de datos*

5. **Implementación en el framework**

La implementación de nuestro proyecto se hizo en el framework de Laravel, creando una estructura de carpetas para su posterior utilización y puesta en marcha que se comprobará con las herramientas entregadas. La estructura de la carpeta consta de lo siguiente:

**A. App / models**

a. Encontraremos los modelos de todos nuestras clases.

**B. App / Http / controllers**

a. Encontraremos los controllers.

**C. public / JavaScript**

a. Encontraremos todas las funciones para nuestro proyecto.

**D. public / img**

a. Encontraremos una carpeta con todas las imágenes que se han utilizado.

**E. public**

a. Encontraremos un archivo estilos.css que es el que se ha implementado en el proyecto.

**F. resources / views**

a. Se encontrarán todas las vistas y repartidas en subcarpetas para diferenciarlas bien.

**G. routes**

a. Aquí se encontrarán las rutas web en web.php y las rutas para nuestra api en api.php.

**H. .env**

a. En este archivo tan solo configuraremos la conexión al puerto y a la base de datos.

## d. Pruebas

### i. Pruebas de funcionalidad de la aplicación mediante postman

Para este apartado, he probado la funcionalidad mediante postman para así tener la conexión con la base de datos mediante la api usando ajax. En este caso se estuvo utilizando la api haciendo llamadas gets, post, put y del, para posteriormente utilizarlos después en nuestra aplicación y hacerlo de forma más segura.

Estas pruebas se hicieron mediante las restricciones que creamos en nuestros controladores para conseguir así una seguridad y una validación para cada funcionalidad óptima, siendo así nuestra aplicación más segura y funcional.

Por último, generando bien las llamadas, las peticiones y dando respuestas con códigos de estado 201 de todas las clases a las que

queremos realizar el crud, ya se empezó a realizar las vistas para el proyecto.

**ii. Pruebas de funcionalidad de la aplicación mediante las vistas**

Para este apartado hemos realizado diversas formas de funcionalidad, tanto para la inserción de datos, eliminación y modificación de los mismos. Usando así diversas formas como validaciones de nombres con caracteres especiales, nombres con menos caracteres de los permitidos, utilizando además la no repetición de valores ya creados previamente, haciendo que un participante no pueda pelear contra el mismo y además con todos los campos requeridos.

Solo permitiendo el acceso a la web mediante el login del usuario sin él no se le permite hacer nada, ni acceder. Eliminado el token, modificando o poniendo un token con la fecha vencida, este nos devolverá un error especificando el motivo , devolviéndole al login.

Si el usuario desactiva el JavaScript la aplicación le redirigirá a otra página en el que se le especificará que el JavaScript debe estar siempre activo para el uso de nuestra web.

## e. Despliegue

### i. **Configuración de xampp**:

Primero deberíamos de tener configurado el xampp como se hizo en el anexo **Guía de Usuario Joel A. Cruz Morales**, una vez configurado e instalado, arrancamos el servicio en modo administrador para tenerlo más seguro.

### ii. **Configuración de apache**:

Ya arrancado el servicio, deberemos arrancar el panel de control de xampp y arrancar los módulos de Apache y Mysql previamente configurados. Accederemos a phpmyadmin clicando en Admin en el módulo de MySQL y haremos la inserción de la query de la base de datos que está explicada en el anexo explicado en el punto anterior.

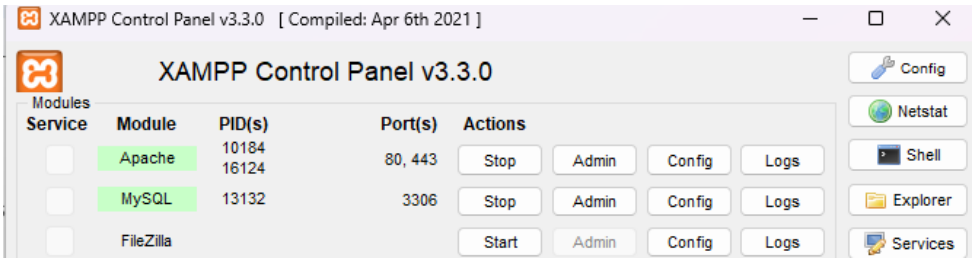


*Figura de puesta en marcha de los servicios*

iii. **Configuración de laravel**:

En nuestra aplicación de laravel implementaremos nuestra carpeta del proyecto al Visual Studio Code, una vez implementado, modificamos el archivo .env con nuestro puerto, usuario y contraseña de la base de datos. Una vez tengamos esto lo que haremos es abrir la terminal de Visual, ejecutar el comando php artisan serve y acceder a la ip que nos proporciona

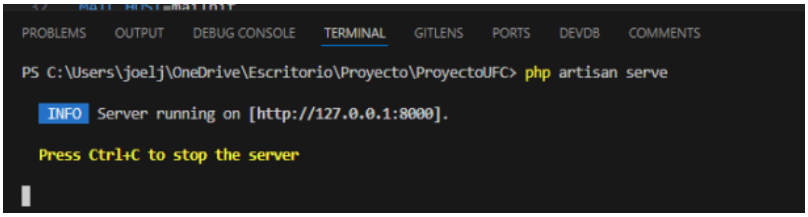


*Figura de puesta en marcha del servidor*

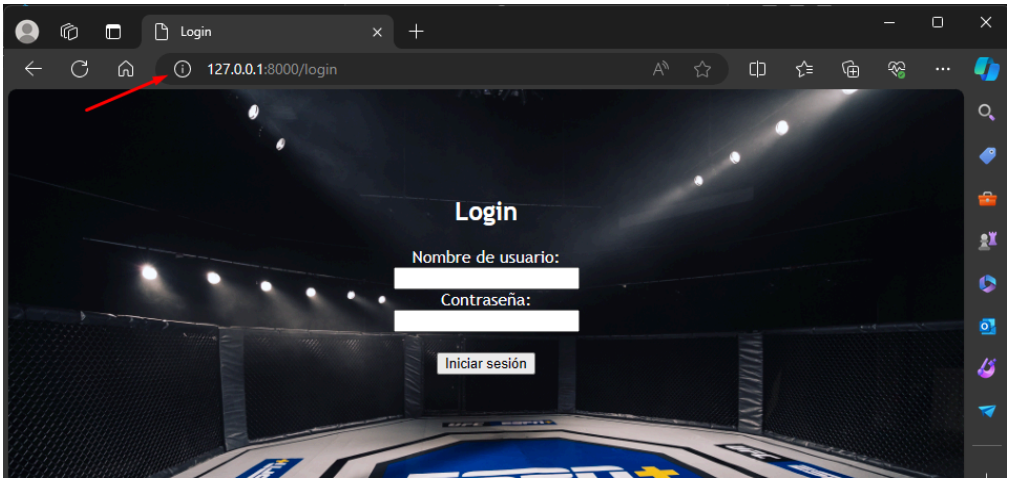


*Figura del servidor en marcha*

# 3. Tecnologías empleadas

Para este proyecto he utilizado el Framework Laravel para el desarrollo de esta aplicación web en php, haciendo uso del modelo, vista y controlador. A continuación se detallan las tecnologías empleadas para el proyecto.

a. **HTML, CSS, y JavaScript(JS)**:

   i. **HTML**: Es el lenguaje que utilicé para el contenido visual de la aplicación web, gestionando el contenido, como el formulario de inicio de sesión, la visualización de los datos y la inserción de ellos.

   ii. **CSS**: Es el que elegí para que se encargue del diseño y la presentación visual de mi aplicación web, personalizando el aspecto a mi gusto, el diseño de botones, tablas y vistas. En mi caso no elegí Bootstrap ya que preferí hacer totalmente solo el diseño con CSS.

   iii. **JavaScript**: El empleado para realizar la interactividad con la aplicación web, creando funciones para que respondan a las acciones del usuario, como la búsqueda de participantes,

cogestión de peleas y veladas, también es útil para realizar las validaciones y la manipulación de la página .Principalmente haciendo uso de peticiones a la API mediante AJAX

b. **PHP**: Es el lenguaje de programación del lado del servidor, utilizado para crear la lógica del proyecto y de la api, también utilizado para la autenticación y gestión de todos los eventos de nuestro proyecto, y además la interacción directa con la base de datos.

c. **Laravel**: Este Framework se utilizó ya que su funcionamiento está enfocado en php, usando solo el modelo de modelo, vista y controlador por su robustez y modularidad que ofrece.

En resumen, se eligieron estas tecnologías para el proyecto debido a la comodidad que presenta para el desarrollo web y lo cómodo que me resulta. Lo destacable de este Framework es la capacidad para manejar la lógica del proyecto, la interactividad del usuario y su compatibilidad con los requisitos de este proyecto.

# 4. Integración y coste empresarial

## a. Aportación y beneficios a la empresa

La implementación de esta idea puede proporcionar una serie de beneficios significativos para una empresa. En primer lugar, puede mejorar la eficiencia para centralizar la gestión y organización de este tipo de eventos, lo que a su vez significa una reducción significativa en administración y mejora de la productividad del personal. Además, puede generar nuevas fuentes de ingresos a partir de la comercialización de eventos y venta de entradas como nueva sección de la web, también de nuevas oportunidades de patrocinio y asociaciones con diferentes organizaciones deportivas.

Esta idea puede mejorar la imagen de la marca y la perspectiva del empresario al demostrar la gestión de eventos de una forma rápida y cómoda. Esto puede aumentar la capacidad de creación de eventos, ya que al ser de una manera tan intuitiva y rápida supondría bajo coste y tiempo de realización.

Además con la modularidad de esta aplicación permite por ejemplo a una federación de lucha permitir tipos de eventos, ya sea una

competición de Judo, Boxeo o diversas artes marciales, con tan solo la inserción de nuevos participantes a sus respectivas categorías, ya tendría la aplicación lista para funcionar.

## b. Costes hardware y software

i. **Hardware**:

Para la implementación de este proyecto, es importante considerar los costes para que el hardware y el software tengan un rendimiento óptimo. En la parte de hardware, podríamos optar por un pc de gama media ya que no necesitaremos muchos requisitos, solo lo utilizamos para alojar la aplicación y el mantenimiento de ellos. En este caso elegí un pc de sobremesa con un procesador Intel Core i5, con 16 GB de RAM y un disco de 500 GB de SSD y 1 T de HDD para el desarrollo, con un coste aproximado de 800€

En cuanto a servidores, podríamos optar por servidores de gama media-alta para poder alojar la aplicación y la base de datos en el entorno de producción. Para ello necesitamos dos servidores para poder alojar bastante información y con estos iríamos muy bien.

Elegí para los servidores, un servidor DELL con procesadores Intel Xeon E-5 , 64GB de RAM. Elegí estos ya que son los más óptimos

para el almacenamiento de muchos datos (son los usados por Cauce). El coste de estos están alrededor de 640€, teniendo administración remoto y compatible para hacer RAID.

ii. **Software**:

Para el software se necesitarán licencias para el sistema operativo de los servidores, como Windows Servers que puede tener un coste alrededor de 1000€ por servidor. Además se requerirían licencias para herramientas de desarrollo, pero en nuestro caso nos será gratuito al usar Visual Studio Code, aunque pueden oscilar entre los 100€ y 300€ por licencia.

## c. Elección de precios, hosting, licencias, etc

i. **Hosting**:

Para este apartado, opté por el servicio de hosting en l anduve como AWS, ya que tiene características similares a nuestro servidor mencionado en el anterior punto, esto tendrá un coste de 100€ al mes. Esto incluiría almacenamiento por parte de servidor y en la nube como parte principal, ancho de banda y capacidad de procesamiento.

ii. **Licencias**:

Para el proyecto, se necesitarán licencias para cualquier herramienta o software para el desarrollo. En este caso las herramientas empleadas son gratuitas, pero dependiendo de la versión y las funcionalidades requeridas podrían tener un coste adicional.

En resumen, los costes totales del hardware, software, hosting y licencias para el proyecto, pueden variar dependiendo de las especificaciones exactas , las necesidades del proyecto y la variación de los componentes en el mercado. Sin embargo, la estimación inicial del proyecto podría ser alrededor de 2000€ a 3000€ para el hardware y software inicial, y alrededor de 100€ al mes para el hosting en la nube y las licencias adicionales.

# 5. Bibliografía

**Referencias:**

Amazon Web Services (AWS). Recuperado de https://aws.amazon.com/es/ (Accedido el 22 de abril del 2024).

PC Componentes. Dell Windows Server 2022 Standard Edition ROK 16 Núcleos 1 Licencia para Servidores Dell. Recuperado de https://www.pccomponentes.com/dell-windows-server-2022-standard-edition-rok-16-nucleos-1-licencia-para-servidores-dell (Accedido el 24 de abril del 2024).

ETB Tech. Servidor Dell PowerEdge. Recuperado de https://www.etb-tech.com/dell-poweredge-r730xd-1x12-3-5-sas-2-x-e5-2670-v3-2-3ghz-twelve-core-64gb-perc-h730-idrac8-enterprise-svr-r730xd-018.html?currency=EUR&source=google&medium=cpc&campaign=EU_Shopping_Spain&gad_source=4 (Accedido el 24 de abril de 2024).

PC Componentes. Para la obtención de precios para el pc que queríamos. Recuperado de https://www.pccomponentes.com/ (Accedido el 24 de abril de 2024).

Documentación de clase. Material proporcionado durante el curso de Desarrollo de Aplicaciones Web en las materias Desarrollo web en entorno servidor (DSW) y Desarrollo web en entorno cliente (DEW). (Accedido desde el inicio del proyecto).

# 6. Anexos

- DOC Guia Desarrollador Velada Joel A. Cruz Morales
- DTS Clase Velada Joel A. Cruz Morales
- Guía de estilos JavaScript Joel A. Cruz Morales
- Guía de estilos PHP Joel A. Cruz Morales
- Guía de Usuario Joel A. Cruz Morales
- Guías de estilo HTML y CSS Joel A. Cruz Morales
- Enlace al proyecto GitHub: https://github.com/Joelcrmo/ProyectoUFC