*PROYECTO***:**

**ChessGame**

Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

**Manual de código**

Realizado por: ***Miguel Ortega Álvarez***

Contenido

[Manual de código 3](#_Toc166599342)

[API Backend (PHP) 3](#_Toc166599343)

[API Frontend (Js) 12](#_Toc166599344)

[Componentes (React) 13](#_Toc166599345)

[Arbiter, helper y reducer 19](#_Toc166599346)

# Manual de código

El manual de código es una guía fundamental para entender y trabajar con el código fuente de la aplicación. Proporciona una visión detallada de la estructura, organización y funcionamiento del código, lo que permite a los desarrolladores comprender rápidamente cómo interactúan los diferentes componentes y cómo realizar contribuciones efectivas al proyecto. En esta sección, se presentará una descripción general del manual de código, destacando su importancia y el alcance de la información proporcionada para facilitar el desarrollo, mantenimiento y colaboración en el proyecto.

# API Backend (PHP)

Esta API se encarga de redirigir las peticiones a la base de datos dependiendo del endpoint que contenga la solicitud que proviene del API del front.

Captura de pantalla de computadora

Descripción generada automáticamente

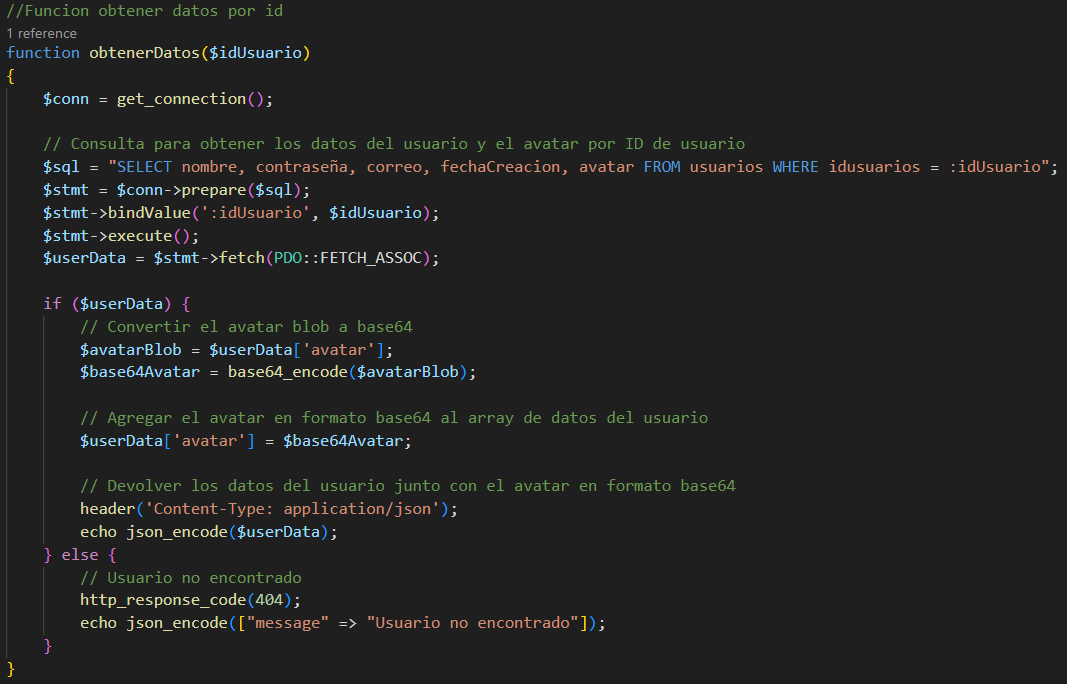
Esta es la función principal que asegura una conexión con la base de datos en local.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

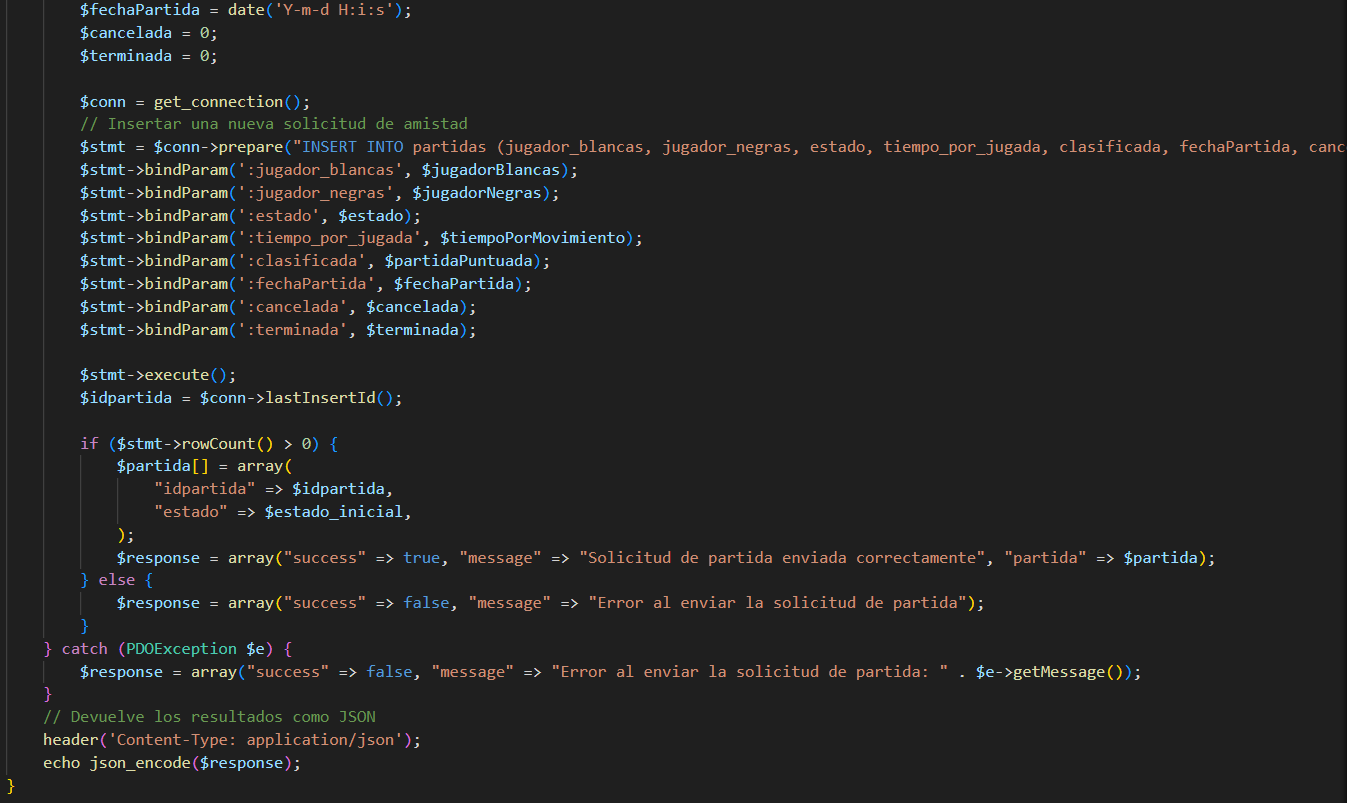


Texto

Descripción generada automáticamente

Estas son algunas de las funciones mas sencillas de la aplicación. A continuación, mostrare las funciones relacionadas con la solicitud de partida, recuperación de la partida, abandonar partida, etc.

Texto

Descripción generada automáticamente

Esta función es la encargada de crear una partida con el estado inicial.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Esta función es la encargada de obtener una partida al pulsar sobre la partida en el listado de partidas.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Esta función se ejecuta al confirmar un movimiento, almacena las fechas del movimiento y el estado de la partida.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Esta función se encarga de marcar una partida como cancelada o terminada y de administrar los puntos en función de los movimientos de la partida y si es clasificada o no. Esta función solo se ejecuta cuando el usuario pulsa en abandonar partida.

Al igual que existe esta función, también existe la función de abandonar partida por tiempo, que se ejecuta cuando uno de los dos usuarios se queda sin tiempo para mover.

Por último, la función finalizar partida, que se ejecuta cuando uno de los usuarios hace jaque mate dentro de una partida.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

# API Frontend (Js)

Esta API se encarga de almacenar la lógica de las funciones generadas de todos los componentes. Esta realizada en JavaScript y se comunica con el API del backend mediante peticiones axios.

En el caso de trabajar en local con la aplicación, la ruta de las solicitudes quedaría así:

Texto

Descripción generada automáticamente

Pero esto se debe cambiar al subirlo a un dominio, indicando en la constante BASE\_URL el dominio en cuestión y modificando la ruta de las carpetas en caso necesario.

Algunas de las comprobaciones como el tamaño del avatar, los requisitos de la contraseña, el correo etc, se realizan en este apartado para no sobrecargar el servidor.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Pero en general es una forma de administrar las funciones en un mismo fichero, sin necesidad de sobrecargar los componentes con la lógica de las funciones.

# Componentes (React)

En este apartado mostraré algunos de los componentes de la aplicación, como puede ser el componente Home o el componente Board.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza media

Este componente es el encargado de mostrar la información de las partidas que el usuario tiene en curso. Permitiendo acceder al componente Board con la partida en cuestion mediante un click.

El componente board es uno de los mas complejos de la aplicación. En el se encuentra el estado de la partida y las piezas del tablero.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Aquí se encontrarian las llamadas al API Services y las funciones para mostrar y ocultar el modal de abadonar partida.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza bajaTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Finalmente, esto sería lo que devuelve el componente Board, un tablero y varios componentes hijos que se renderizan en funcion de lo que ocurra en el componente padre. Por ejemplo, MoveList es el listado de movimientos de la partida y PromotionBox o GameEnds, son popups que se renderizan al promocionar un peon en la ultima casilla o al finalizar una partida.

# Arbiter, helper y reducer

Por otro lado, se encuentran diferentes carpetas relacionadas con la logica del ajedrez.

En la carpeta Arbiter encontraremos tres ficheros, en move.js encontramos los movimientos “especiales” que contiene el ajedrez, como es la captura al paso de un peon o el enroque largo y corto.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

En el fichero getMoves.js se encuentran las funciones para calcular los movimientos según la pieza y la posicion.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

En este ejemplo, tenemos la funcion que determina el movimiento de las torres y del rey.

Por ultimo, se encuentra el fichero arbiter, que es el encargado de gestionar que los movimientos sean validos y de contener la logica del ajedrez.

Texto

Descripción generada automáticamente

En este ejemplo, se calcula si el rey se encuentra en jaque.

Texto

Descripción generada automáticamente

Y en este otro, partiendo de si el rey se encuentra en jaque, se calcula si es jaque mate.

Una vez visto el fichero arbiter y todo lo que contiene, paso a explicar el fichero helper, que es el que contiene las funciones relacionadas con la notación de los movimientos y la posicion de las piezas.

Texto

Descripción generada automáticamente

Esta es la funcion para anotar el movimiento de las piezas. Esta notacion esta basada en la notacion algebraica, donde se calcula la fila y la columna, la pieza que se mueve, si es una captura o un enroque, etc.

Para mas información sobre las notaciones en ajedrez, puedes consultar el siguiente enlace: <https://es.wikipedia.org/wiki/Notaci%C3%B3n_(ajedrez)>

En este fichero, tambien se encuentra la funcion para crear una posicion inicial.

Texto

Descripción generada automáticamente

Por ultimo, el fichero reducer que contiene las ordenes que se envian según lo que ocurra en la partida.

En este fichero se encuentran las acciones que se pueden enviar a la partida.

Texto

Descripción generada automáticamente

En el ejemplo, se puede crear una nueva partida o volver a un movimiento anterior.

Texto

Descripción generada automáticamente

También es el encargado de ejecutar los movimientos en el tablero.

Para llamar a estas acciones se hace uso de la funcion dispatch(orden()), un ejemplo claro de esto, lo podemos obtener en el componente pieces, al realizar un movimiento.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

En este ejemplo, se ejecutaria todo lo relacionado al hacer un movimiento. Primero, se calcularia si un peon esta coronando en la ultima casilla, después se calcularia si es un enroque, actualizando el estado del enroque en caso de que se diese el enroque, ya que en el ajedrez solo se puede hacer un enroque por partida.

Después pasariamos al arbiter el movimiento que queremos hacer partiendo de la posicion actual y anotamos el movimiento en MoveList con la funcion getNewMoveNotation(), una vez hecho esto, damos la orden de ejecutar el movimiento haciendo uso del dispatch y calculamos posteriormente si la posicion después del movimiento se encuentra en estado de “material insuficiente”, “ahogado” o “jaque mate”.

Por ultimo, si la partida se encuentra en alguno de estos estados y no se encuentra en /Practice, se ejecutara finalizarPartida y se anotará en la base de datos el reparto de puntos ademas de darse la partida como finalizada.