

PROYECTO INTEGRADO
CFGS: Desarrollo de aplicaciones web

Hundir la flota



Alumna: Elisabet Martin Muriel

Repositorio GIT:

https://github.com/emmuriel/hundirflota/tree/Intelligent_boot

CAPÍTULO 1: INTRODUCCIÓN

Índice

1.- Capítulo 1: Introducción

1.1.- Introducción	7
1.2.- Propósito	7
1.3. Objetivos	8
1.4. Coste estimado del proyecto	
1.4.1. Coste del desarrollo	8
1.4.2. Coste de implantación	9

2.- Capítulo 2 ANÁLISIS DEL SISTEMA

2.1. Introducción	12
2.2. Análisis de requisitos	12
2.2.1. Requisitos funcionales	12
2.2.2. Requisitos no funcionales	20
2.3. Casos de uso y narrativas de casos de uso	22
FIGURA 1. DIAGRAMA UML-UC-GENERAL.....	23
FIGURA 2. DIAGRAMA UML-UC-01. IDENTIFICACION USUARIO.....	23
FIGURA 3. DIAGRAMA UML-UC-02. REGISTRO USUARIO	24
FIGURA 4. DIAGRAMA UML-UC- 03. CARGAR PARTIDA	24
FIGURA 5. DIAGRAMA UML- UC-04. JUGAR PARTIDA	25
FIGURA 6. DIAGRAMA UML- UC-08. VER RANKING JUGADORES	25

3.- Capítulo 3 DISEÑO DEL SISTEMA

3.1. Introducción	34
3.2. Diagramas de clases	34
FIGURA 7. DIAGRAMA UML- DIAGRAMA DE CLASES	34

3.3. Diseño de la base de datos	35
3.3.1. Diseño conceptual	35
FIGURA 8. DIAGRAMA . ENTIDAD RELACIÓN	35
3.3.2. Diseño lógico	36
FIGURA 9. DIAGRAMA. ESQUEMA RELACIONAL	36
3.4. Diseño de la interfaz	37
3.4.1. Diagrama de interfaces.	
FIGURA 11 . DIAGRAMA DE INTERFACES	37
3.4.2. Especificación de interfaces.	
FIGURA 12 . ESPECIFICACION DE ELEMENTOS INTERFAZ “index.php”	38
FIGURA 13 . ESPECIFICACION DE ELEMENTOS INTERFAZ “registro.php”	39
FIGURA 14 . ESPECIFICACION DE ELEMENTOS INTERFAZ “confirmacion.html” ...	40
FIGURA 15 . ESPECIFICACION DE ELEMENTOS INTERFAZ “partida.html”	41
FIGURA 16 . ESPECIFICACION DE ELEMENTOS INTERFAZ “ranking.html”	42

4.- Capítulo 4 IMPLEMENTACIÓN

4.1. Introducción	44
4.2. Arquitectura cliente-servidor	44
FIGURA 17 . ESQUEMA CLIENTE-SERVIDOR. PROTOCOLO HTTP	44
FIGURA 18 . ESQUEMA REQUEST-RESPONSE. PROTOCOLO HTTP	44
FIGURA 19 . EJEMPLO REQUEST-RESPONSE. PROTOCOLO HTTP	45
4.3. Lenguajes de programación	45
4.4. Herramientas de desarrollo	46
4.5. Codificación	47
FIGURA 20 . CODIGO INDEX.PHP	47
FIGURA 21 CODIGO PARTIDA.PHP	48

<i>FIGURA 22 Y 23 . CODIGO PARTIDA.PHP</i>	49
<i>FIGURA 24 . CODIGO PARTIDA.PHP</i>	50
<i>FIGURA 25 . CODIGO RESPONSEJSON</i>	50
<i>FIGURA 26 . CODIGO CLASE CEREBSERVIDOR</i>	51
<i>FIGURA 27. CODIGO CLASE CEREBSERVIDOR</i>	52

5.- Capítulo 5 PRUEBAS DE SOFTWARE

5.1. Introducción	54
5.2. Técnicas de prueba	55
5.2.1. Técnicas de caja blanca	56
5.2.2. Tecnicas de caja negra	57

6.- Capítulo 6 . CONCLUSIONES

6.1. Conclusiones	64
6.2. Propuestas futuras	64

7 - Capítulo 7. BIBLIOGRAFÍA Y REFERENCIAS

7.1. Referencias bibliográficas	66
---------------------------------------	----

ANEXO I. MANUAL DE INSTALACIÓN

ANEXO II. MANUAL DE USUARIO

1.1.- INTRODUCCIÓN DEL PROYECTO

La siguiente memoria describe los pasos seguidos para el análisis e implementación del proyecto integrado para el ciclo formativo superior “Desarrollo de Aplicaciones Web”.

La razón por la que se ha escogido este proyecto no es económica ni buscando satisfacer una necesidad de mercado, este proyecto nace con el firme propósito de afianzar un aprendizaje integrando las diferentes tecnologías que existen en un proyecto web, así pues, parafraseando a uno de mis mejores maestros, “ *Jugar es la manera más entretenida de aprender*”.

1.2.- PROPÓSITO

Este proyecto tiene como propósito desarrollar plataforma de juegos en web.

El juego que se implementará es una recreación del mítico juego BattleShip o Batalla Naval. Un antiguo juego de mesa, donde cada jugador tiene un tablero dividido en una cuadrícula, con una flota de barcos y otro tablero donde se va “disparando” sobre el tablero rival y descubriendo las posiciones en las que el enemigo ha colocado su flota de barcos. El juego gestiona por turnos según el acierto/error del jugador que dispara.

El objetivo del juego es bombardear la flota contraria con el mínimo número de intentos y antes de que la flota propia sea bombardeada.

En este proyecto solo se implementa una modalidad de juego:

Jugador vs Boot. El jugador solicita la partida y juega contra la máquina

1.3 - OBJETIVOS DEL PROYECTO

Lista de los objetivos principales:

- Ob1.** Analizar, definir y adaptar las reglas del juego a un proyecto realizable en tiempo y forma ajustado al protocolo cliente-servidor.
- Ob2.** Analizar los requisitos para crear la Base de datos.
- Ob3.** Modelar e implementar una Base de Datos Relacional que utilice la aplicación.
- Ob4.** Diseñar un sistema de autenticación de usuarios.
- Ob5.** Implementar uso de sesiones para mantener la persistencia de datos.
- Ob6.** Diseñar e implementar la interfaz de usuario
- Ob7.** Implementar peticiones http asíncronas con el uso de AJAX.
- Ob8.** Diseñar clases y métodos. POO
- Ob9.** Implementar un sistema monojugador contra la máquina.
- Ob10.** Implementar un algoritmo del lado del servidor que simule lógica de juego contra un usuario.
- Ob11.** Implementar una lógica de turnos.-

1.3.- COSTE DEL PROYECTO

Estimación del coste y esfuerzo del software y/ o hardware requerido para la realización este proyecto.

1.4.1.-COSTES DE DESARROLLO

Costes para la realización del sistema.

Costes Recursos Informáticos

Software:

Aplicaciones: Se ha tomado la determinación, por la naturaleza del proyecto (académico) que todo el software utilizado para el desarrollo de este proyecto será software libre con licencia gratuita.

SO: El sistema operativo que utiliza el hardware es Windows 10 Pro. Sistema operativo de 64 bits, procesador basado en x64

Hardware:

Laptop marca ASUS AF52, con las siguientes características:

Procesador: Intel(R) Core(TM) i3 CPU M 350 @ 2.27GHz 2.27 GHz

Adaptador Gráfico: [Intel Graphics Media Accelerator \(GMA\) HD Graphics](#)

RAM: 8,00 GB (7,79 GB usable)

Memoria secundaria: 1TB SSD

Pantalla: 15.60 pulgadas, 16:9, 1366 x 768 pixeles

Coste actual: 400€.

Periféricos:

Ratón optico: 12€.

Auriculares: 25€.

Coste estimado de hardware: 437€.

Costes de personal. Factores a tener en cuenta:

El numero de horas totales dedicadas a este proyecto deben ser aproximadamente 360 horas.

Teniendo en cuenta que se trata de un proyecto académico, el coste real del personal no es monetizable.

1.4.2 COSTES DE IMPLANTACIÓN

Costes para poner en funcionamiento el sistema. Resumen del coste total de ponerlo en funcionamiento de manera desglosada:

Licencias de software: Al usar sólo tecnologías de software libre, el coste es nulo.

Costes de desarrollo: Coste de hardware: 437€.

Costes de formación: Toda la formación recibida ha sido totalmente gratuita.

Despliegue en servidor: Coste del servicio: 5,95€/mes (71,4€ Anuales).

El alquiler servicios de hosting a terceros, es la alternativa más razonable para la aplicación en este momento si fuera a ser desplegada. Supone una inversion inicial

mucho menor hasta que la aplicación pueda ser monetizada y además nos libera de la tarea de mantenimiento, soporte y seguridad del servidor.

Necesitamos escoger un proveedor contrastado y fiable.

Proveedor Hostinger . Paquete Web Hostin Premium. Características:

100 sitios web
100 GB de SSD
~25 000 visitas al mes
Email
Ancho de banda ilimitado
Bases de datos ilimitadas

Seguridad

SSL ilimitado
Protección de nameservers Cloudflare

Bonos gratis

Creador de sitios web
Copias de seguridad **semanales**
Dominio **incluido** (9,99 €)

Opciones de WordPress

WordPress **administrado**
Aceleración **WordPress**

Servicio y soporte

Garantía de **30 DÍAS** de reembolso
Soporte **24/7**
Garantía de **99,90%** de uptime

Detalles técnicos

Acceso **GIT**
Acceso **SSH**

Más funciones

Gestión de **DNS**
Administrador de **acceso**
100 subdominios
Cuentas FTP **ilimitadas**

TOTAL Costes de implantación del sistema: 508€ el primer año.

CAPÍTULO 2: ANÁLISIS DEL SISTEMA

2.1.- INTRODUCCIÓN

A continuación se recogen los requisitos principales que debe cumplir el software en un catálogo numerado y las especificaciones de los casos de uso que describen las interacciones del sistema con los diferentes actores o roles de usuario, aportando así la definición del comportamiento de la aplicación y las acciones que se pueden ejercer sobre ella.

2.2.- ANÁLISIS DE REQUISITOS

2.2.1. REQUERIMIENTOS FUNCIONALES.

Este documento es el resultado de revisar y modificar el catálogo de requisitos previos, describiendo qué debe hacer el sistema.

Con el fin de facilitar la lectura y comprensión de este catálogo, los requisitos han sido organizados atendiendo a la siguiente organización:

1. *Requisitos para la gestión de usuarios.*
2. *Requisitos para el desarrollo de partidas.*
3. *Requisitos relativos a la seguridad de la aplicación*
4. *Requisitos legales*

1. Requisitos para la gestión de usuarios

CÓDIGO	R01
NOMBRE	Registro Usuario
PRIORIDAD	Alta
DESCRIPCIÓN	Opción que permite crear una nuevo usuario en la BBDD, con la información necesaria para dicha operación
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R02
NOMBRE	Autenticación Usuario
PRIORIDAD	Alta
DESCRIPCIÓN	Permite la identificación de un usuario a través de la comprobación y confirmación de credenciales (Nombre de

	usuario y contraseña)
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R03
NOMBRE	Actualizar última conexión
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema registrará la fecha y hora de la última conexión del usuario
TIPO	Funcional – Base de datos
USUARIO	Sistema

CÓDIGO	R04
NOMBRE	Ver score jugadores
PRIORIDAD	Alta
DESCRIPCIÓN	El usuario podrá ver un listado ordenado de mayor a menor, de los veinte jugadores con mayor numero de victorias conseguidas.
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R05
NOMBRE	Cambio de contraseña
PRIORIDAD	Media
DESCRIPCIÓN	Todos los usuarios una vez iniciada sesión deben poder cambiar su contraseña de acceso a la plataforma.
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

2. Requisitos para el desarrollo de partidas

CÓDIGO	R06
NOMBRE	Jugar Nueva Partida
PRIORIDAD	Alta
DESCRIPCIÓN	El usuario podrá iniciar una nueva partida contra la máquina
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R07
NOMBRE	Abandonar Partida
PRIORIDAD	Alta
DESCRIPCIÓN	El usuario podrá abandonar la partida en el momento que desee una vez haya empezado, sin necesidad de cerrar el navegador.
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R08
NOMBRE	Guardar Partida en juego
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema guardará automáticamente el estado actual de la partida con el fin de poder retomarla en la siguiente solicitud de partida.
TIPO	Funcional – Base de datos
USUARIO	Sistema

CÓDIGO	R09
NOMBRE	Retomar o Descartar partida
PRIORIDAD	Alta
DESCRIPCIÓN	El usuario que tenga una partida en juego guardada debe tener la posibilidad de retomarla o descartarla al solicitar

	una nueva partida.
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R10
NOMBRE	Proclamar ganador
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe monitorear la partida en juego hasta que se abandone o se establezca un ganador
TIPO	Funcional – Base de datos
USUARIO	Sistema

CÓDIGO	R11
NOMBRE	Eliminar Partida jugada
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema eliminará automáticamente la partida en juego una vez se haya establecido un ganador
TIPO	Funcional – Base de datos
USUARIO	Sistema

CÓDIGO	R12
NOMBRE	Registrar victoria
PRIORIDAD	Alta
DESCRIPCIÓN	El numero de victorias del usuario se incrementará en 1 cada vez que el usuario consiga ganar una partida
TIPO	Funcional – Base de datos
USUARIO	Usuario/Jugador

CÓDIGO	R13
NOMBRE	Gestión de turnos en partida
PRIORIDAD	Alta

DESCRIPCIÓN	El sistema debe implementar la lógica y comprobaciones necesarias para gestionar y asegurar el turno de juego entre jugadores en el momento de la partida.
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R14
NOMBRE	Asignación de tableros
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema proporcionará de manera aleatoria a los jugadores los tableros válidos para la partida
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R15
NOMBRE	Cambiar tablero
PRIORIDAD	Alta
DESCRIPCIÓN	El jugador debe tener la posibilidad de solicitar un cambio de tablero al sistema antes de comenzar la partida. Sin límite de cambios.
TIPO	Funcional
USUARIO	Usuario/Jugador

CÓDIGO	R16
NOMBRE	Gestión de turnos en partida
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar la lógica y comprobaciones necesarias para gestionar y asegurar el turno de juego entre jugadores en el momento de la partida. Actualizando el estado de la partida en la BBDD.
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R17
NOMBRE	Ejecutar disparo
PRIORIDAD	Alta
DESCRIPCIÓN	El usuario debe poder ejecutar disparos sobre las posiciones del tablero siempre que sea su turno. Actualizando el estado de la partida en la BBDD.
TIPO	Funcional
USUARIO	Usuario/Jugador

CÓDIGO	R18
NOMBRE	Jugador boot
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar un boot que de manera automática juegue una partida contra el usuario que la solicite. El boot debe seguir las reglas de juego, ser capaz de tomar decisiones de tiro en base a los resultados anteriores, simular tiempos de respuesta de una persona humana.
TIPO	Funcional
USUARIO	Sistema

3. Requisitos relativos a la seguridad de la aplicación

CÓDIGO	R19
NOMBRE	Validación de datos de registro de usuario
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe requerir y validar todos los campos del formulario de registro
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R20
NOMBRE	Validación de datos formulario login
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe requerir y validar todos los campos del formulario autenticación antes de consultar la BBDD.
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R21
NOMBRE	Securización frente a ataques SQLi
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe filtrar los datos de entrada para bloquear posibles ataques SQLi
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R22
NOMBRE	Securización frente a ataques XSS
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe filtrar los datos de entrada para bloquear posibles ataques XSS
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R23
NOMBRE	Contraseña segura
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar los mecanismos necesarios para procurar el uso de establecer contraseñas seguras por parte de los usuarios.
TIPO	Funcional

USUARIO	Sistema
CÓDIGO	R24
NOMBRE	Gestión de accesos
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe permitir el acceso a las partidas a usuarios correctamente autenticados.
TIPO	Funcional
USUARIO	Sistema

CÓDIGO	R25
NOMBRE	Establecer sesión
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe iniciar una sesión para cada usuario que sea autenticado en el sistema.
TIPO	Funcional
USUARIO	Sistema

4. Requisitos legales

CÓDIGO	R26
NOMBRE	Consentimiento cookies
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar un aviso legal, con un banner que verifique o rechace el uso de cookies de acuerdo a la normativa vigente, basándose en la “Guía para la implementación de cookies” ofrecida por la AEPD.
TIPO	Funcional
USUARIO	Sistema

2.2.2. REQUERIMIENTOS NO FUNCIONALES

Los requisitos no funcionales dice sobre 'qué debería ser un sistema'.

Los clasificaremos en:

1. *Requisitos de rendimiento.*
2. *Requisitos de usabilidad.*
3. *Requisitos de mantenimiento.*

1.Requisitos de rendimiento

CÓDIGO	R27
NOMBRE	Limitar tiempo sesiones
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe acotar un periodo de tiempo máximo de vida de la sesión. El tiempo establecido para la sesión será de 30 minutos.
TIPO	No Funcional
USUARIO	Sistema

CÓDIGO	R28
NOMBRE	Tiempo de refresco de partida
PRIORIDAD	Alta
DESCRIPCIÓN	Tras una actualización del juego de la partida, el tiempo de refresco de los tableros no debe ser superior a 2 segundos.
TIPO	No Funcional
USUARIO	Sistema

CÓDIGO	R29
NOMBRE	Tiempo de carga
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe cargar la página de principal en 3 segundos como máximos después del login.
TIPO	No Funcional

USUARIO

Sistema

2.Requisitos de usabilidad

CÓDIGO	R30
NOMBRE	Diseño responsivo de la interfaz
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar un diseño responsive para adaptar la interfaz a dispositivos móviles y tablets.
TIPO	No Funcional
USUARIO	Sistema

CÓDIGO	R31
NOMBRE	Diseño minimalista
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe implementar un diseño minimalista que permita jugar una partida de manera rápida y eficaz.
TIPO	No Funcional
USUARIO	Sistema

3.Requisitos de mantenimiento

CÓDIGO	R32
NOMBRE	Encriptación actualizada
PRIORIDAD	Alta
DESCRIPCIÓN	Con el fin de facilitar la labor
TIPO	No Funcional
USUARIO	Sistema

CÓDIGO	R33
NOMBRE	Librerías actualizadas
PRIORIDAD	Alta
DESCRIPCIÓN	Con el fin de aumentar la vigencia del software, se implementará el código utilizando librerías con mantenimiento.
TIPO	No Funcional
USUARIO	Sistema

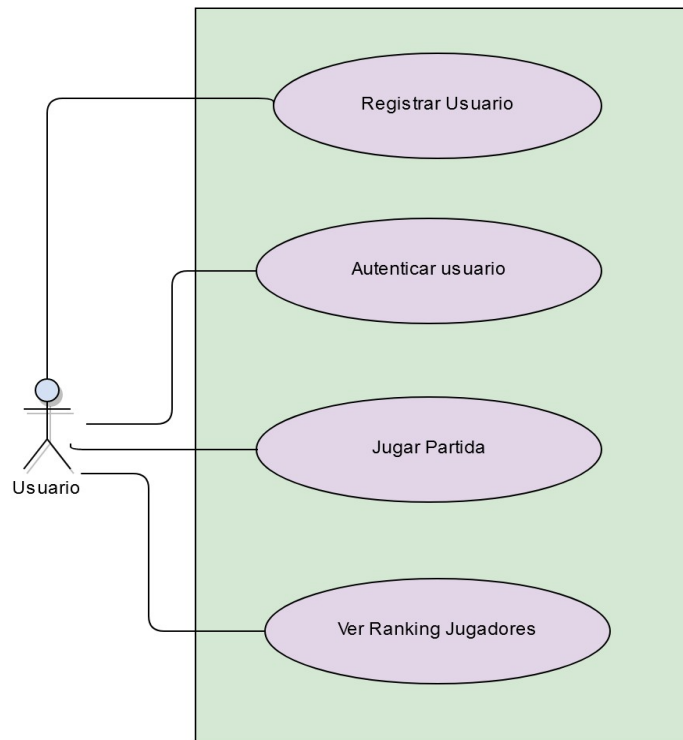
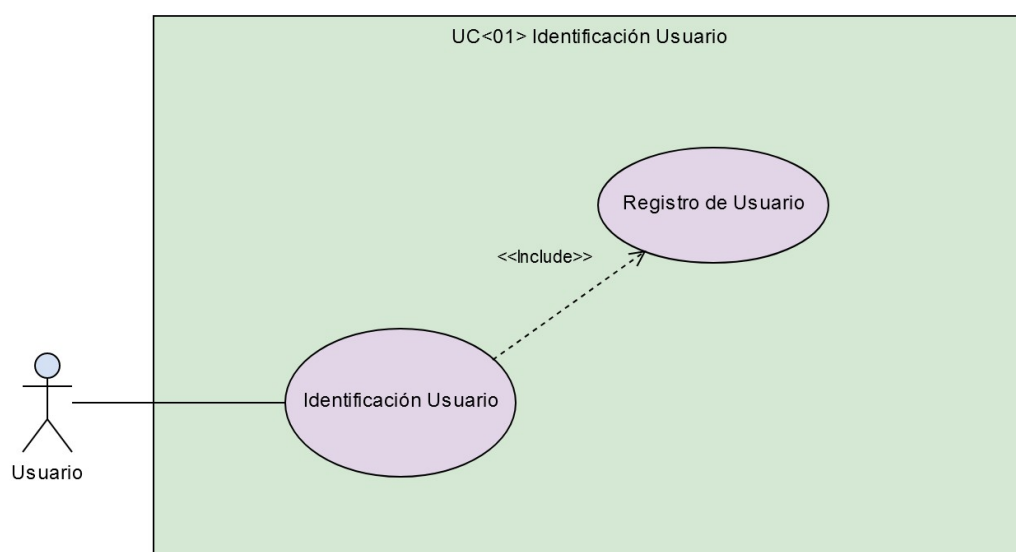
CÓDIGO	R34
NOMBRE	Repositorios de software
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema debe tener disponible un repositorio actualizado y ordenado con las diferentes realises disponibles.
TIPO	No Funcional
USUARIO	Sistema

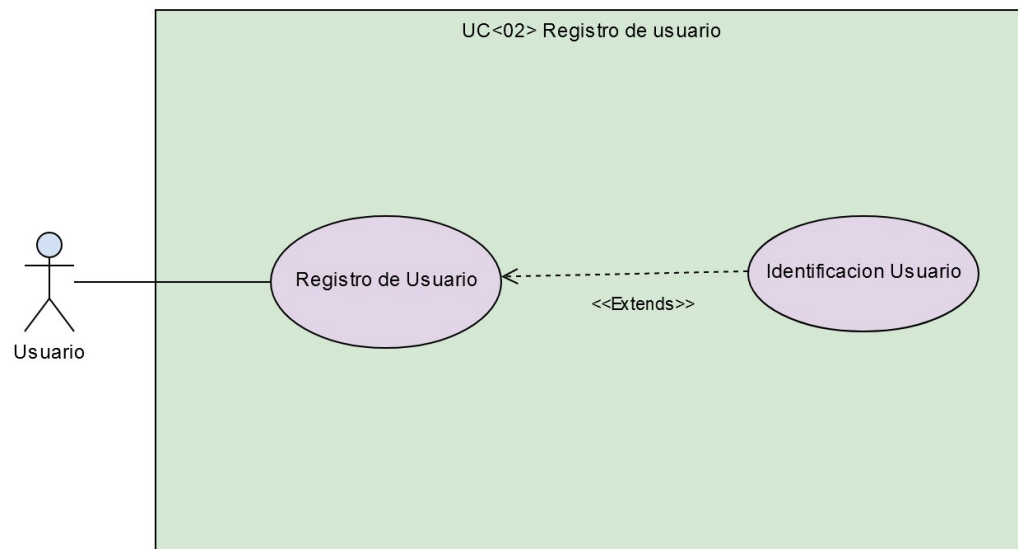
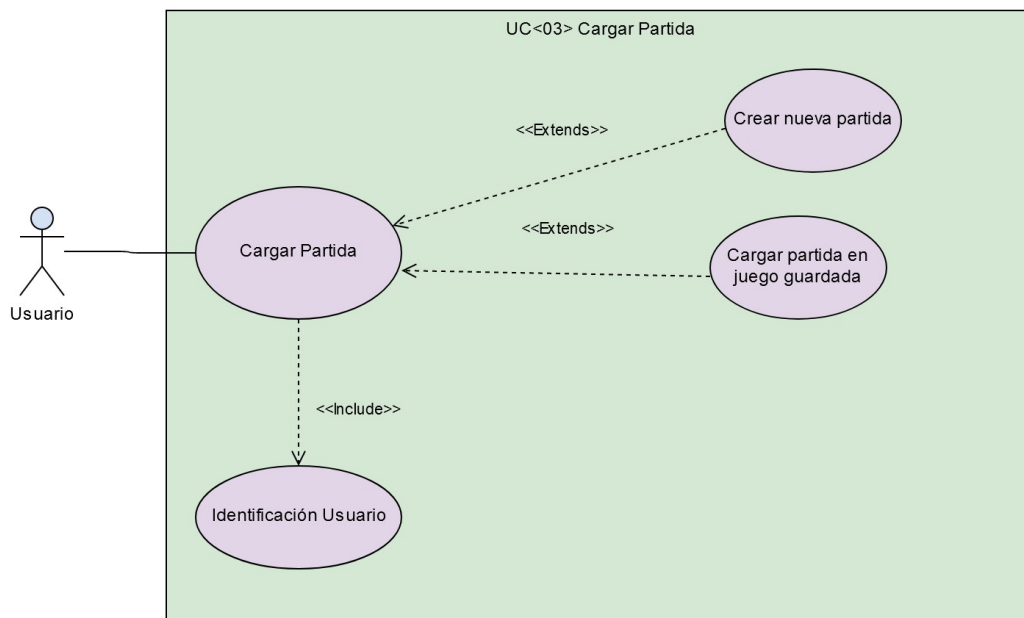
2.3.- CASOS DE USO: DIAGRAMAS Y NARRATIVAS DE CASOS DE USO

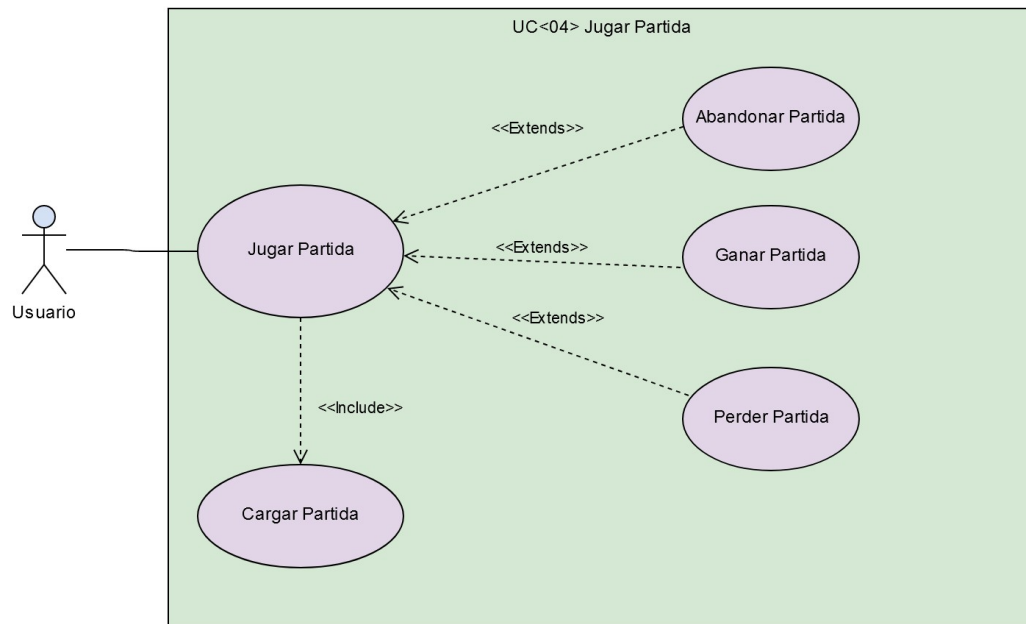
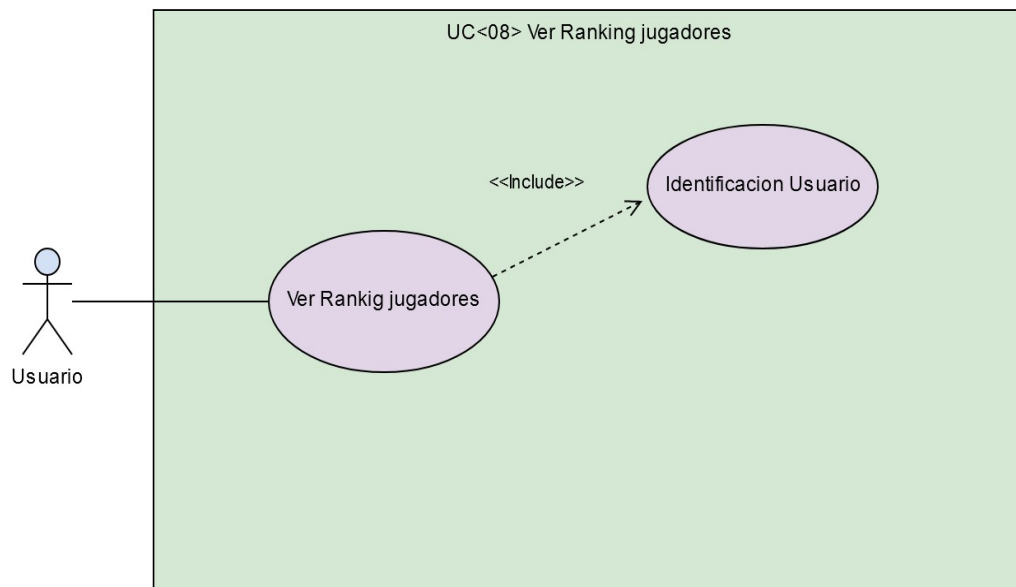
Los diagramas de casos de uso son grafos que describen los escenarios potenciales desde un punto del usuario según el rol que este desempeñe.

Identificación de actores:

Actor -01	<u>Usuario</u>
Descripción	Este actor representa al usuario que juega las partidas
Comentario	Este es el unico rol implementado en la aplicación. La administración del sistema se realiza con herramientas externas al sistema.

DIAGRAMAS DE CASO DE USO**CASO DE USO GENERAL***FIGURA 1. DIAGRAMA UML-UC-GENERAL***CASO DE USO 01 – IDENTIFICACIÓN USUARIO***FIGURA 2. DIAGRAMA UML-UC-01. IDENTIFICACION USUARIO*

CASO DE USO 02 – REGISTRO USUARIO*FIGURA 3. DIAGRAMA UML-UC-02. REGISTRO USUARIO***CASO DE USO 03 – CARGAR PARTIDA***FIGURA 4. DIAGRAMA UML-UC- 03. CARGAR PARTIDA*

CASO DE USO 04 – JUGAR PARTIDA*FIGURA 5. DIAGRAMA UML- UC-04. JUGAR PARTIDA***CASO DE USO 08 – VER RANKING JUGADORES***FIGURA 6. DIAGRAMA UML- UC-08. VER RANKING JUGADORES*

DESCRIPCIÓN DETALLADA DE LOS CASOS DE USO

UC-<01>	Identificación Usuario	
Descripción	Un usuario quiere acceder al sistema.	
Precondición		
Flujo de eventos	Paso	Acción
	P1	El usuario accede a la página web principal de la aplicación.
	P2	El sistema muestra la página principal con la ventana de la política de cookies, bloqueando la visualización del formulario de login.
	P3	El usuario acepta el uso de cookies pulsando el botón 'Aceptar'.
	P4	El sistema cierra la ventana de cookies dando acceso al usuario al formulario de login.
	P5	El usuario introduce su nombre de usuario y su contraseña y envía la información pulsando un botón de "Autenticar". El sistema comprueba que usuario y contraseña sean correctos.
	P6	El sistema inicia una sesión de usuario, que terminará cuando el usuario cierre la aplicación. Registra la fecha y hora de la conexión en la BBDD y carga la página juego.
Postcondición	Se inicia una sesión nueva sesión de usuario.	
Excepciones	Paso	Acción
	P4b	Si el usuario deniega el uso de cookies, se abrirá una nueva ventana con el siguiente mensaje "Lo sentimos, no es posible acceder. Este sitio web necesita el uso de cookies de sesión para su funcionamiento".
	P5b	Si el usuario clickea el hipervínculo para registrarse. El sistema lo redirigirá a la página de registro de usuario. Continúa con UC-<02> P1
	P6b	Si el sistema detecta algún campo está vacío, muestra un mensaje de error al lado de cada campo de texto bordeando

	P6c	<p>el campo con color rojo.</p> <p>Si el sistema no encuentra el nombre de usuario en la Base de Datos o la contraseña no es correcta, muestra el mensaje “Error: Credenciales Incorrectas” en el mismo cuadro de donde se inicia la sesión.</p>
--	------------	--

UC-<02>		Registro Usuario
Descripción	Un usuario quiere registrarse en el sistema	
Precondición	El usuario no está registrado en el sistema	
Flujo de eventos	Paso	Acción
	P1	El usuario accede a la página de registro
	P2	El sistema muestra el formulario de registro
	P3	El usuario introduce un nombre de usuario, email, contraseña, confirmación de contraseña y envía la información pulsando un botón de “Registrar”.
	P4	El sistema comprueba que los campos no están vacíos, que que las contraseñas son seguras y además coinciden.
	P5	El sistema comprueba que el nombre de usuario y email no están registrados en la BBDD.
	P6	El sistema registra al nuevo usuario en la Base de Datos
	P7	El sistema redirige al usuario a un página que contiene un mensaje que informa del éxito de la aplicación y un hipervínculo para volver a la página principal de login (<i>Sigue <UC 01> P4</i>)
Postcondición	El usuario está registrado en la BBDD	
Excepciones	Paso	Acción
	P4b	Si el alguno de los campos está vacío, o las contraseñas no coinciden, el sistema muestra un error con la información del error y bordeando de color rojo el campo del formulario que ha producido el error.
	P5b	Si el sistema detecta algún carácter prohibido en alguno de los campos, el sistema muestra un error con la información

		del error y bordeando de color rojo el campo del formulario que ha producido el error.
	P5c	Si el usuario o dirección de email ya están registrados en la BBDD, el sistema responderá con un mensaje de error bordeando en rojo los campos afectados. No registrará el nuevo usuario.

UC-<03>		Cargar partida
Descripción	Un usuario quiere crear una partida	
Precondición	El usuario debe estar autenticado en sistema para tener acceso al servicio	
Flujo de eventos	Paso	Acción
	P1	El usuario accede a la página de partida.
	P2	El sistema muestra la interfaz de partida y el boton "Jugar"
	P3	El usuario pulsa el botón "Jugar".
	P4	El sistema comprueba si el usuario tiene una partida en curso guardada en la BBDD.
	P5	El sistema genera una nueva partida, asigna los tableros y envía la información al usuario. Carga en pantalla dos tableros, uno para el usuario y otro para el servidor. En el centro el botón Empezar y Cambiar tablero
	P6	El usuario está conforme con el tablero asignado y pulsa el boton "Empezar"
	P7	El sistema habilita el sonido, deshabilita el botón "Empezar" y el botón "Cambiar tablero" y muestra en su lugar el botón Abandonar. El tablero con el borde rojo indica el tablero sobre el que se ejecutará el disparo cambiando según el turno.
Postcondición	La partida en juego está cargada y registrada en la BBDD	
Excepciones	Paso	Acción
	P5b	Si el usuario ya tiene una partida en juego registrada en la BBDD, el sistema abrirá una ventana de dialogo para

		preguntar si quiere cargar la anterior partida o empezar una nueva. El usuario elige "Empezar partida nueva", sigue paso P5
	P5c	Si el usuario ya tiene una partida en juego registrada en la BBDD, el sistema abrirá una ventana de dialogo para preguntar si quiere cargar la anterior partida o empezar una nueva. El usuario elige "Cargar partida anterior", el sistema carga la partida registrada, dando el turno a quien correspondía.
	P6b	Si el usuario no está conforme con el tablero asignado y pulsa el botón "Cambiar tablero", el sistema asignará aleatoriamente otro tablero de la Base de Datos. Vuelve a P6

UC-<04>		Jugar partida
Descripción	Un usuario quiere jugar una partida ya cargada	
Precondición	El usuario debe estar autenticado en sistema para tener acceso al servicio. La partida debe estar ya creada, registrada y empezada	
Flujo de eventos	Paso	Acción
	P1	El usuario en su turno pasa por encima con el puntero sobre el tablero de la derecha, al que lanzará una bomba. El puntero coloreará de rojo la posición seleccionada.
	P2	El usuario hace doble click sobre la posición y lanza el disparo.
	P3	El sistema recoge la información, y procesa si el disparo ha sido erróneo o acertado.
	P4	El sistema actualiza la información de la partida en la BBDD y refresca la interfaz con el resultado. Gestiona el turno en base a si el disparo ha sido certero o no y selecciona con el borde rojo el tablero sobre el que se ejecutará el próximo disparo.
	P5	El sistema determina fallo. El turno es del boot. El sistema

		realiza un disparo sobre el tablero. El sistema actualiza la información de la partida en la BBDD y refresca la interfaz con el resultado.
	P6	El sistema determina fallo. El turno es del usuario, el sistema actualiza la información de la partida en la BBDD y refresca la interfaz con el resultado. Vuelve a P1
Postcondición		La partida en juego será actualizada
Excepciones	Paso	Acción
	P2b	Si el usuario no tiene el turno y lanza un disparo sobre una de las casillas a disparar, el sistema no recoge esa interacción.
	P5b	El sistema ha determinado acierto. Turno sigue siendo del jugador. Vuelve a P1
	P6b	El sistema ha determinado acierto. El turno sigue siendo del boot. Vuelve a P5

UC-<05>	Abandonar partida	
Descripción	En la partida en juego el usuario abandona una partida	
Precondición	El usuario debe estar autenticado en sistema para tener acceso al servicio. La partida debe estar ya creada, registrada y empezada.	
Flujo de eventos	Paso	Acción
	P1	El usuario pulsa el botón "Abandonar"
	P2	El sistema abre un cuadro de diálogo para confirmar la operación.
	P3	El usuario confirma la operación.
	P4	El sistema elimina la partida en juego de la BBDD y muestra de nuevo la página de partida sin cargar. Sigue UC<03> P1
Postcondición	La partida en juego será actualizada	
Excepciones	Paso	Acción
	P3b	Si el usuario no confirma la operación. Sigue la partida

--	--	--

UC-<06>	Ganar partida	
Descripción	El usuario descubre todos los barcos de la flota enemiga en el tablero antes que su rival el boot, por tanto es el ganador.	
Precondición	El usuario debe haber acertado todas las casillas en las que se encuentra un barco en el tablero enemigo	
Flujo de eventos	Paso P1 P2 P3	Acción El usuario clicka en la última casilla válida El sistema recoge el disparo y comprueba que no hay más posiciones posibles para explotar. Declara ganador al usuario, actualiza en la BBDD el numero de victorias del usuario y borra la partida ya terminada. Emite un mensaje de éxito al jugador en un cuadro de diálogo. El jugador acepta el cuadro de diálogo. Sigue UC<03>Cargar partida.
Postcondición	La partida habrá sido eliminada y el usuario habrá incrementado en 1 el numero de partidas ganadas.	

UC-<07>	Perder partida	
Descripción	El boot descubre todos los barcos de la flota enemiga en el tablero antes que su rival el usuario, por tanto es el ganador.	
Precondición	El boot debe haber acertado todas las casillas en las que se encuentra un barco en el tablero enemigo	
Flujo de eventos	Paso P1 P2	Acción El boot dispara en la última casilla válida en el tablero. El sistema recoge el disparo y comprueba que no hay más posiciones posibles para explotar. Declara ganador al boot,

	P3	Emite un mensaje humillante de derrota al jugador en un cuadro de diálogo. Borra la partida ya terminada. El jugador acepta el cuadro de diálogo. <i>Sigue UC<03>Cargar partida.</i>
Postcondición	La partida habrá sido eliminada	

UC-<08>	Ver Ranking Jugadores	
Descripción	El usuario solicita al sistema ver el ranking de los 20 jugadores con mayor numero de victorias de la aplicación	
Precondición	El usuario debe estar autenticado en sistema para tener acceso al servicio. .	
Flujo de eventos	Paso	Acción
	P1	El usuario pulsa la opcion de ver ranking
	P2	El sistema hace una consulta ordenada a la BBDD y muestra una lista ordenada en orden decreciente, de los 20 usuarios con mayor numero de victorias registradas.
Postcondición	El listado con el ranking de jugadores será mostrado	
Excepciones	Paso	Acción

CAPÍTULO 3: **DISEÑO DEL SISTEMA**

3.1.- INTRODUCCIÓN.

En el presente capítulo se tratará ver cómo llevar a cabo el diseño teniendo como punto de vista el dominio de la solución a dicho problema. Por tanto con este apartado se busca diseñar con destreza una solución que satisfaga los requisitos.

Se desarrollará una solución lógica, donde se llevarán a cabo diferentes fases.

3.2.- DIAGRAMA DE CLASES.

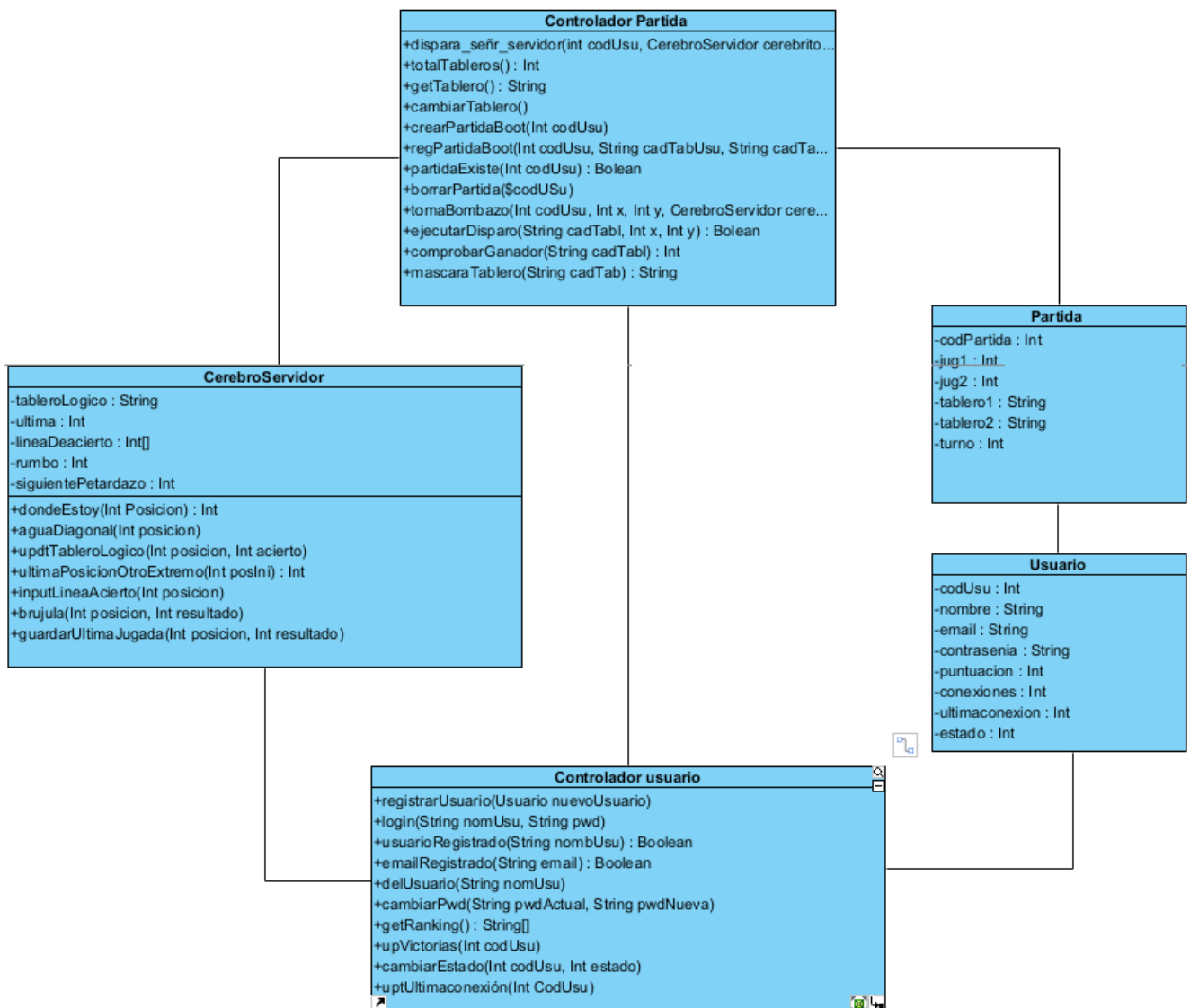


FIGURA 7. DIAGRAMA UML- DIAGRAMA DE CLASES

3.3.- DISEÑO DE LA BASE DE DATOS.

La base de datos, será la estructura de soporte de datos del sistema. Se ha elegido una base de datos relacional. A raíz de la especificación de requisitos elaboramos el siguiente modelo de datos.

3.3.1.- DISEÑO CONCEPTUAL.

El modelo entidad-relación muestra una representación simple de las estructuras y los tipos de relaciones que existen entre ellas

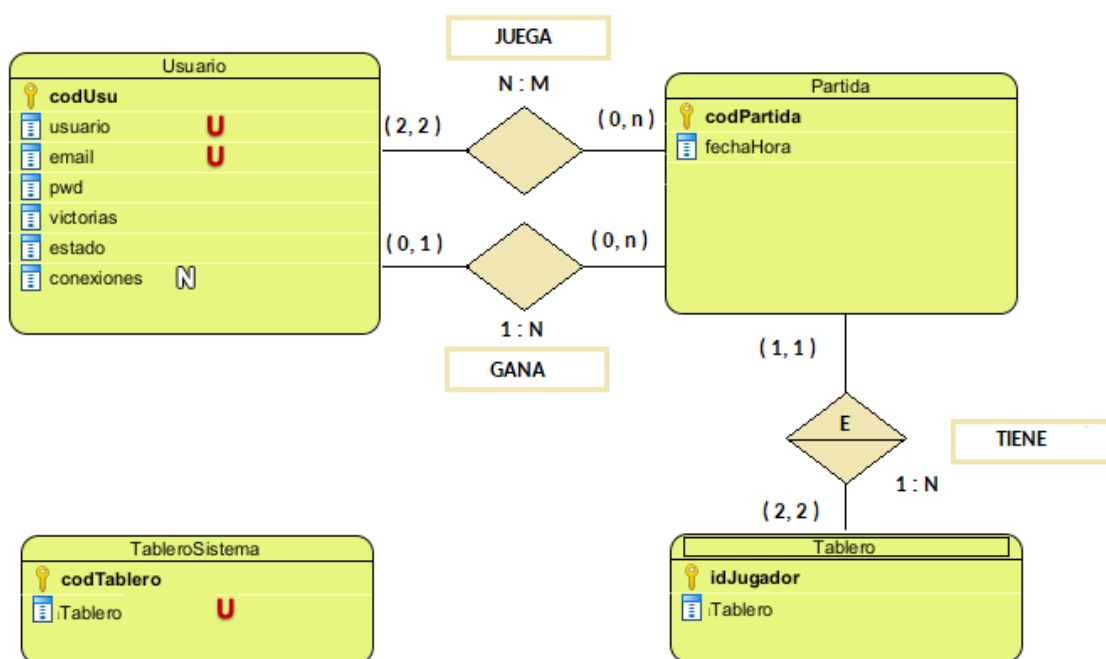


FIGURA 8 MODELO ENTIDAD /RELACIÓN

* *Consideraciones:* La entidad *TableroSistema* guarda información de constantes que sirven para dar el valor inicial a los tableros de la partida, por su propio carácter inmutable no debe ser modificado, por tanto no es esta entidad la que se relaciona con la partida.

3.3.2.- DISEÑO LÓGICO.

Esquema Relacional

Usuario (PK· codUsu, U· usuario , U· email , pwd, victorias, estado, N·conexiones)

TableroSistema (PK· codTablero, U·tablero)

(*) **Partida** (PK· codPartida, FK· jug1 → Usuario, FK· jug2 → Usuario, FK· ganador → Usuario)

TableroPartida (PFK· codPartida →Partida , PK· idJugador, tablero)

(*) Por razones de rendimiento a la hora de hacer las consultas, se ha decidido agregar los codigos de usuario como clave ajena en la tabla Partida, en lugar de crear una nueva tabla.

Procedimiento regPartida(jug1, jug2, tablero1, tablero2) . Ejecuta una transacción, para registrar correctamente la clave primaria compuesta de la **tabla tablero partida** que además es clave ajena que apunta a **tabla partida**.

Procedimiento updtPartida(jug1, jug2, tablero1, tablero2) Ejecuta una transacción, para actualizar en una sola operación las **tablas partida** y **tableropartida**.

Diseño Relacional

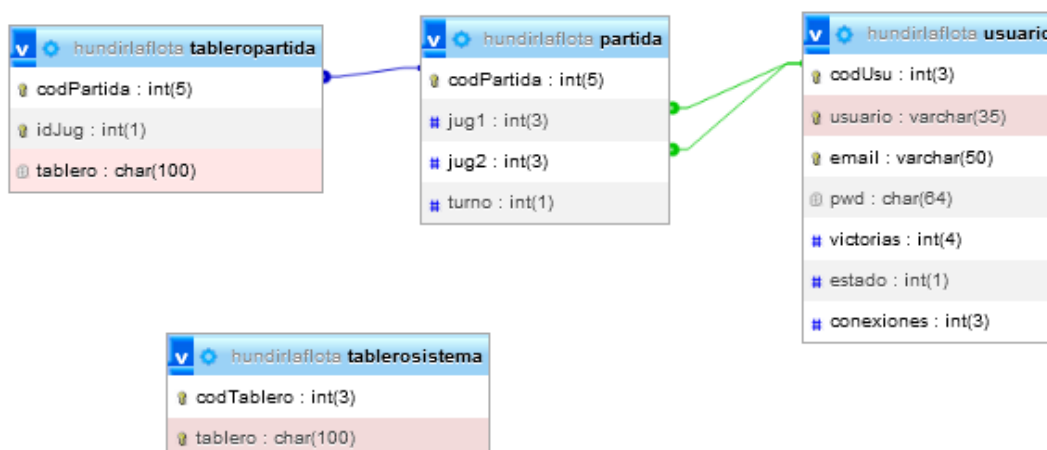


FIGURA 9. MODELO RELACIONAL

3.4.- DISEÑO DE LA INTERFAZ.

Las interfaces deben tener un diseño muy sencillo e intuitivo que facilite al usuario usar la aplicación para su labor principal (jugar partidas).

Si bien las interfaces buscan ser minimalistas, el estilo de la aplicación es una mezcla entre lo moderno y el “old school” con el objetivo de que el juego tenga una estética con trazos de “juego Arcade” (Los videojuegos populares en la época de los 90 que fué donde el juego que estamos implementando alcanzó su popularidad)

3.4.1. Diagrama de interfaces

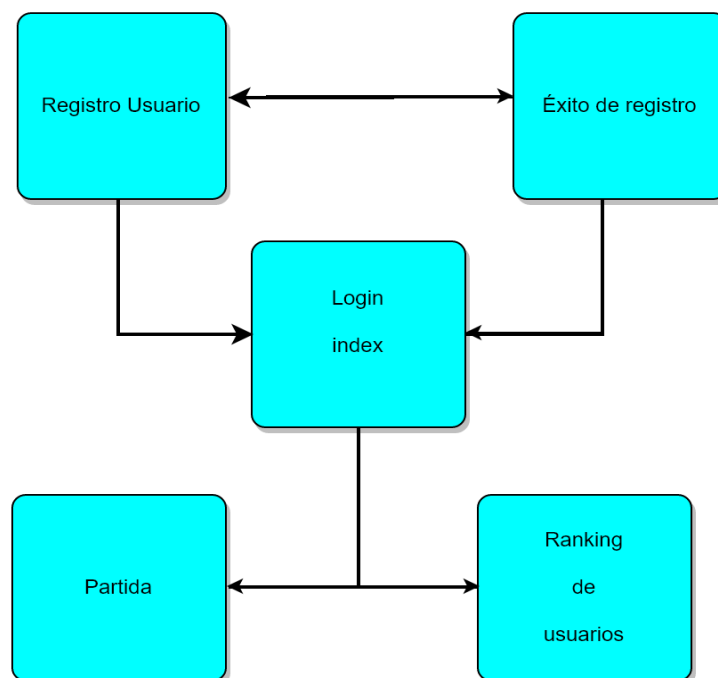


Figura 11. Diagrama de interfaces.

3.4.2. Especificación de la interfaz.

index.php (login).



Figura 12 . Especificacion de elementos interfaz “index.php”

Elementos principales de la interfaz

1. **Slide “Entra y Juega”**: Contiene información sobre la reglas del juego para los jugadores e imágenes.

2. **Fomulario Login**: Formulario para enviar los datos de autenticación al servidor.

Se compone de 2 campos de entrada:

3. **Campo de texto**: Campo tipo texto para insertar el nombre de usuario.

4. **Campo de texto** : Campo tipo password para insertar la contraseña de usuario.

5. **Submit**. Botón de envío de formulariollamado ENTRAR

6. **Hypervínculo** hacia formulario de registro.

registro.php (Registro de Usuario).

The image shows a web form titled 'Nuevo Usuario' (New User) on a dark background. The form is enclosed in a red border. It contains the following elements, each with a red number annotation:

- 1**: The application name 'Hundir Flota' at the top.
- 2**: The form title 'Nuevo Usuario'.
- 3**: The label 'Correo electronico:' above an email input field.
- 4**: The label 'Nombre para el usuario:' above a text input field.
- 5**: The label 'Tu contraseña:' above a password input field.
- 6**: The label 'Repite la contraseña:' above a second password input field.
- 7**: A blue 'Registrarse' (Register) button.
- 8**: A 'Más info' (More info) section containing the email '• Escribenos a »hundirlaflotawm@outlook.es'.

Below the password field, there is a note: 'La contraseña debe contener al menos un caracter numérico y una letra mayuscula-minuscula.'

Figura 13. Especificación de elementos interfaz “Registro.php”

Elementos principales de la interfaz

1. Nombre de la aplicación.

2. Fomulario de alta de Usuario: Envía la información para el registro de un nuevo usuario.

3. Campo de texto: Campo de tipo email, para introducir email.

4. Campo de texto: Campo de tipo texto, para introducir nombre de usuario.

5. Campo de texto: Campo de tipo password, para introducir contraseña válida.

6. Campo de texto: Campo de tipo password, para verificar contraseña.

7. Submit formulario: Botón de envío del formulario llamado “Registrarse”.

8. Banner de info: Banner con la dirección de email para ponerse en contacto con los administradores de la aplicación.

confirmacion.html (Confirmación de registro).

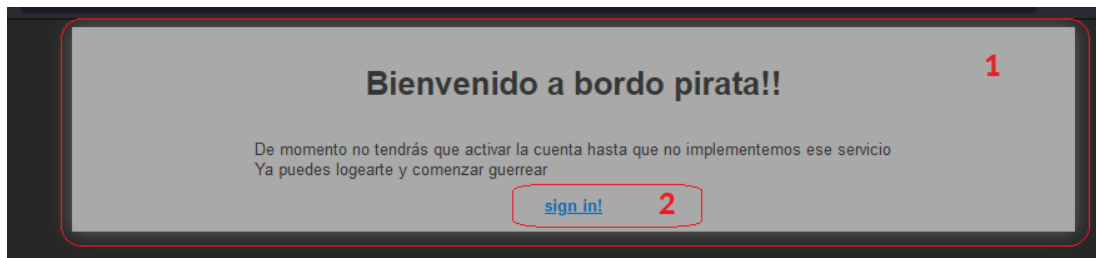


Figura 14. Especificación de elementos interfaz “confirmacion.html”

Elementos principales de la interfaz

1. Cuadro de texto: Contiene un mensaje informativo confirmando el correcto registro en la base de datos.

2. Hipervínculo: Redirige a la página principal para autenticarse.

partida.html(Partida)

Partida sin cargar:



Figura 15 . Especificación de elementos interfaz “partida.html”

Elementos principales de la interfaz

1. Nombre de la aplicación.

2. Campo de texto: Refleja un mensaje de bienvenida para el usuario y su puntuación.

3. Formulario de partida: Formulario sobre el que se juega la partida..

4. Tabla Jugador: Tabla en la que se mostrará el tablero del jugador y sobre la que el boot efectuará sus jugadas.

5. Tabla de Boot: Tabla en la que se mostrará el tablero del boot y sobre la que el jugador efectuará sus jugadas..

6. Botón Jugar: Botón que carga la partida

7. Navbar: Menú de navegación con la opción par aver el ranking.

Partida en juego:

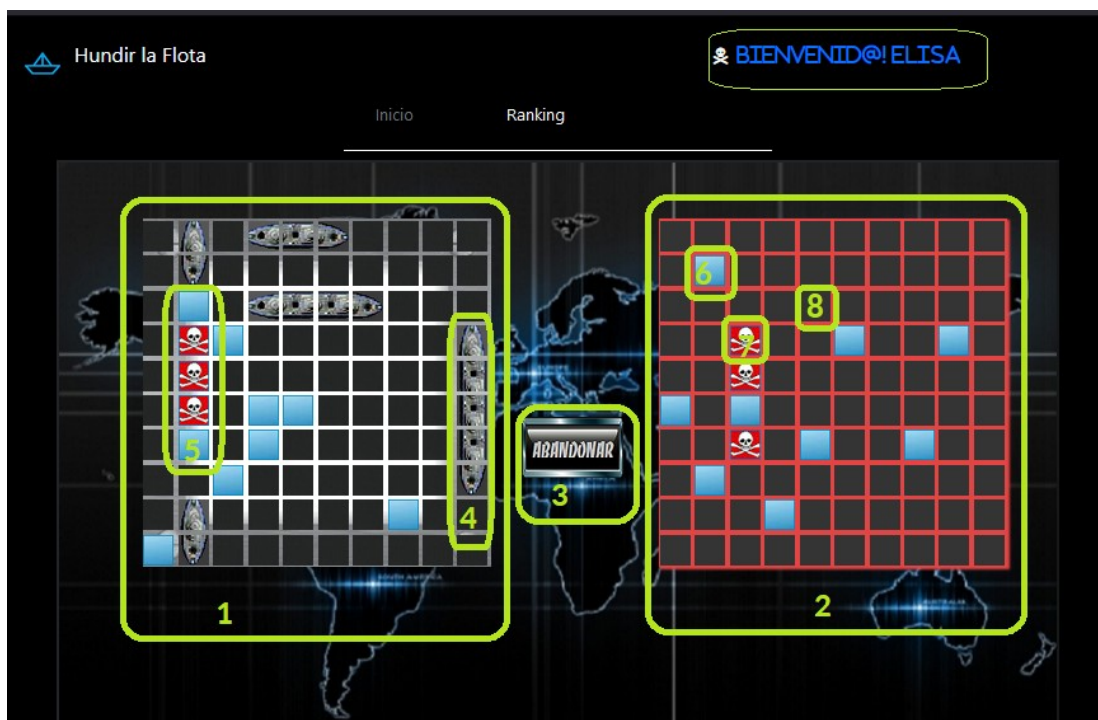


Figura 16 . Especificación de elementos interfaz “partida.html”. Partida en juego

Elementos principales de la interfaz

1. Tabla Jugador: Tabla en la que se mostrará el tablero del jugador y sobre la que el boot efectuará sus jugadas.

2. Tabla de Boot: Tabla en la que se mostrará el tablero del boot y sobre la que el jugador efectuará sus jugadas. Nótese que el tablero está bordeado de rojo, po tanto es sobre el que se ejecutará el próximo tiro, o lo que es lo mismo, el turno es del jugador.

3. Botón Abandonar: Botón que termian con la partida y la elimina.

4. Barco en el tablero: Cómo se muestra un barco en el tablero.

5. **Barco hundido:** Cómo se muestra un barco hundido en el tablero.
6. **Casilla explotada:** Imagen agua. Error al ejecutar el disparo.
7. **Casilla explotada:** Imagen calavera roja. Acierto al ejecutar el disparo.
8. **Casilla no explotada:** Botón sobre el que se puede ejecutar un disparo.

CAPÍTULO 4: IMPLEMENTACIÓN

4.1.- INTRODUCCIÓN.

En este momento ya se encuentra definido el problema y la solución, por lo que lo siguiente será transformar el modelo obtenido en las actividades anteriores en código fuente.

4.2.- ARQUITECTURA CLIENTE/SERVIDOR

La arquitectura cliente-servidor podemos definirla de forma sencilla como un modelo de intercambio de mensajes entre dos participantes (emisor y receptor).

El proceso de enviar un mensaje representa una petición, cuando el receptor del mensaje procesa el mensaje, elabora un mensaje de respuesta a la petición que envía al emisor de la petición.

Existen varios protocolos de comunicación (que estandarizan cómo deben estar contruidos esos mensajes), pero en particular, el protocolo usado la comunicación web es el protocolo HTTP. Actualmente el modelo más usado para este protocolo es el modelo de 3 capas, donde se separa la capa de aplicación de la capa de datos.

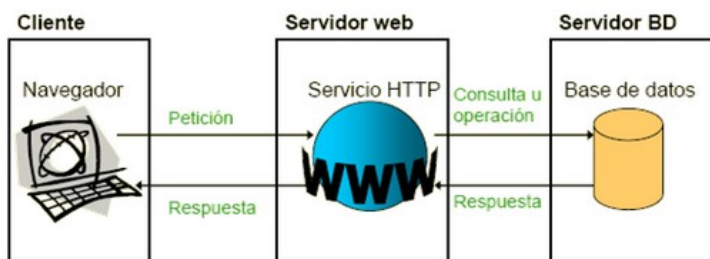


Figura 17. Esquema Cliente -Servidor . Protocolo HTTP

En el protocolo http solo encontramos dos tipos de mensajes, Request y Response (Petición y respuesta) Y la estructura y diferentes configuraciones de estos mensajes es lo que determina toda la comunicación entre cliente y servidor.

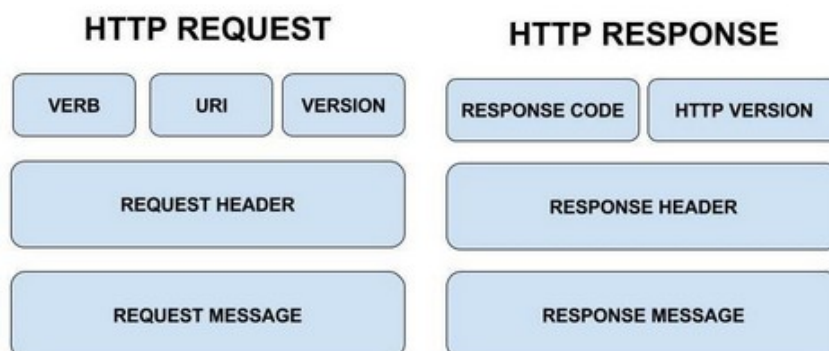


Figura 18. Esquema Request-Response Protocolo HTTP

Ejemplo de mensaje http:

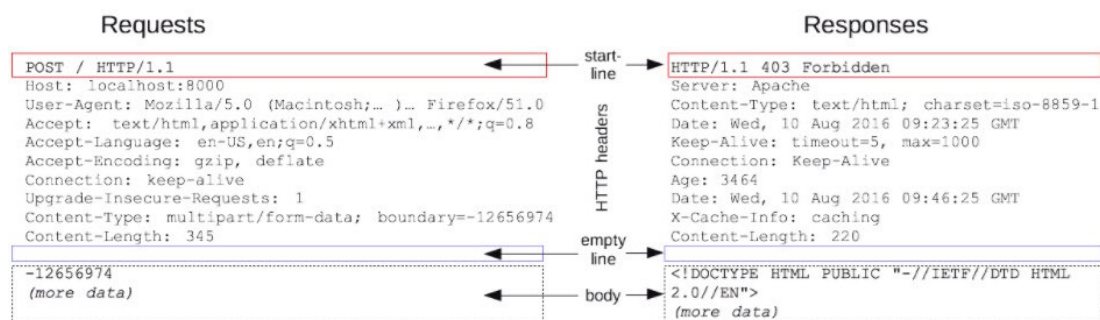


Figura 16 . Especificación de elementos interfaz “partida.html”. Partida en juego

4.3.- LENGUAJES DE PROGRAMACIÓN.

Este proyecto ha sido desarrollado con diferentes lenguajes de programación en que cada uno cumple el propósito para el que ha sido creado.

Del lado del servidor el lenguaje elegido es **PHP**, es un lenguaje interpretado, multihilo muy versátil y rápido, que permite paradigmas de programación estructurada y programación orientada a objetos

Javascript. Lenguaje interpretado que se ejecuta en cliente para realizar peticiones de manera asíncronas al servidor y realizar las funciones necesarias para procesar las respuestas modificando el DOM y los estilos.

Otros lenguajes:

HTML5. Lenguaje de hipertexto . Lenguaje de etiquetas necesario para poder representar la información pedida al servidor en el navegador (cliente).

CSS3. Hojas de estilo en cascada. Para poder dar formato vistoso y agradable a esa información recogida por el cliente

SQL. Sequel Query Language. Lenguaje de consulta estructurado para hacer peticiones a bases de datos SQL.

JSON. Aunque no es un lenguaje en sí, pues se trata de un subconjunto de la notación literal de objetos de JavaScript, debido a su amplia adopción como alternativa a XML, se considera un formato independiente. Es un formato de texto sencillo para el intercambio de datos que suele formar el body de peticiones y respuestas.

4.4.- HERRAMIENTAS DE DESARROLLO.

Siguiendo la línea del proyecto, se han usado herramientas de software libre gratuitas.

Visual Studio Code: Es un IDE gratuito ampliamente configurable a través de pluggins.

Bootstrap: Framework para maquetación, con estilos estandarizados.

SGBD: MySQL. Sistema gestor de base de datos relacionales, respaldado por Oracle y basado en el lenguaje de consultas SQL.

Apache: Servidor Web Local:

Git y Github: Para crear repositorios en la nube y mantener el control de versiones.

Otras Herramientas:

Documentos: Google Documents.

Creación de diagramas UML : <https://gitmind.com> , VisualParadigm Enterprise, phpmyadmin (Diseñador)

4.4.- CODIFICACIÓN.

Todo el código desarrollado se encuentra disponible en el pendrive / moodle que se adjunta con esta memoria (a determinar por el departamento de informática).

3. Códigos fuentes relevantes

Manejo del array asociativo \$_SESSION

Index.php

```
index.php > ...
1  <?php
2  session_name("HundirFlota");
3  session_start();
4  <!DOCTYPE html>
5  <html lang="es">
```

Figura 17 Código index.php

```

39 <?php
40 /*****
41 require_once("modelo/clases.php");
42 require_once("modelo/controlador_usuario.php");
43 require_once("modelo/moduloConexion.php");
44
45
46 if (isset($_POST['entrar'])) { #Login
47
48     if (empty($_POST['txt_usuario']) || empty($_POST['txt_password'])) { //Validación de datos de entrada
49         echo "<div id='error'><span class='error'>Error ** Todos los campos son obligatorios</span></div>";
50     } else {
51         extract($_POST, EXTR_PREFIX_ALL, 'S');
52         $ctrlUsu = new ControlUsuario();
53         //Sanitización de las variables pasada por POST
54         $nombreOk= htmlspecialchars($_S_txt_usuario); #elimina caracteres especiales HTML --Evita ataques XSS
55         $nombreOk=htmlentities($nombreOk, ENT_QUOTES); #elimina comillas dobles y simples --evita SQLInjection
56         $passOk= htmlspecialchars($_S_txt_password);
57         $passOk=htmlentities($passOk, ENT_QUOTES);
58         $obUsu = $ctrlUsu->login($nombreOk, $passOk); //devuelve un usuario
59         if ($obUsu->getCodusu() <= 0) { //el usuario / contraseña incorrectos
60
61             echo "<div id='error'><span class='error'>Error ** El usuario o la contraseña son incorrectos</span></div>";
62         } else {
63             #AKI DEBERIAMOS MACHACAR LA SESION ANTIGUA DE LA BBD CON LA NUEVA SESION PARA EVITAR DOBLES LOGEOS
64             #Esto se implementará más adelante, modificacion de la BBDD
65
66             #Cargar usuario en Session serializando y redireccionar a partida
67             $_SESSION['usuario'] = serialize($obUsu);
68             header("Location: https://localhost/HF/Partida.html");
69         }
70     }
71 }
72
73 if (isset($_POST['registrar'])) { #Registro
74     header("Location: https://localhost/HF/Registro.php");
75 }

```

Figura 18. Código index.php

La función `serialize()` para guardar una vez hecho el login el valor de un objeto Usuario inicializado con los valores devueltos por MySQL a la hora de hacer la autenticación.

Este objeto guardado en el array asociativo `$_SESSION` permite guardar datos como el código de usuario y obtenerlo en cualquiera de los archivos php que componen la aplicación.

Funciones de seguridad de aplicación:

`htmlspecialchars()` – Elimina los caracteres especiales como `<` y `>`

`htmlentities(string, ENT_QUOTES)` - No permite la entrada de comillas dobles ni simples

Partida.php Uso de variables de sesión para autorizar el acceso

```
7 require_once("modelo/clases.php");
8 require_once("modelo/controlador_partida.php");
9 require_once("modelo/controlador_usuario.php");
10 require_once("modelo/moduloConexion.php");
11 require_once("control/funciones.php");
12
13 //Controlar que el usuario esté logeado
14 if ($_SESSION['usuario']) {
15     $objUsu = unserialize($_SESSION['usuario']); # Deserializacion del objeto Usuario.
16     $json= json_decode(file_get_contents('php://input'),true);
17
18     if (isset($json['peticion'])) { #PROCESAR JSON
19         $ctrlPartida = new controlPartida();
20         $peticion=$json['peticion'];
21
22         switch ($peticion) {
```

Figura 19. Código partida.php. Uso de variables de sesión para autorizar el acceso.

Sólo si el usuario se ha logeado correctamente existirá \$_SESSION['usuario'] y el código de partida.php se ejecutará.

De lo contrario la cookie de session se machaca y el usuario solo podrá ver el código html de la aplicación, es decir Partida.html

```
} else {
    session_destroy();
    setcookie('HundirFlota','',time()-100);
    header("Location: https://Error.php");
}
```

Figura 20. Código partida.php. Uso de variables de sesión para autorizar el acceso

Partida.php Lógica de las peticiones de juego

Partida.php no envía HTML ante las peticiones, solo envía archivos JSON y recibe las peticiones a través de archivos JSON

```
<?php
session_name("HundirFlota");
session_start();
header('Content-Type: application/json');

/*****
require_once("modelo/clases.php");
require_once("modelo/controlador_partida.php");
require_once("modelo/controlador_usuario.php");
require_once("modelo/moduloConexion.php");
require_once("control/funciones.php");

//Controlar que el usuario esté logeado
if ($_SESSION['usuario']) {
    $obUsu = unserialize($_SESSION['usuario']); # Deserializacion del objeto Usuario.
    $json= json_decode(file_get_contents('php://input'),true);

    if (isset($json['peticion'])) { #PROCESAR JSON
        $ctrlPartida = new controlPartida();
        $peticion=$json['peticion'];
```

```
switch ($peticion) {
case 1: #CARGAR DATOS USUARIO
    $datosUsu = utf8_encode('BIENVENID@ ' . $obUsu->getNombre() . '!! Total Partidas Ganadas: ' . $obUsu->getPuntuacion());
    $response = array ("datosUsu"=>$datosUsu);
    echo json_encode($response,JSON_FORCE_OBJECT,512); // al array hay que forzarlo a ser objeto
    break;

case "2": #CREAR NUEVA PARTIDA

    //Comprueba que ya no haya una partida jugandose

    $existePartida = $ctrlPartida->partidaExiste($obUsu->getCodUsu());
    if (!$existePartida) {
        //Empieza partida
        $ctrlPartida->crearPartidaBoot($obUsu->getCodUsu());
        $partida = $ctrlPartida->obtenerPartida($obUsu->getCodUsu());
        #Enmascarar tablero boot
        $tablero2=$ctrlPartida->mascaraTablero($partida->getTablero2());
        $partida->setTablero2($tablero2);

        responseJson($obUsu, $partida, "0");
    }else{
        #Enmascarar tablero boot
        $partida = $ctrlPartida->obtenerPartida($obUsu->getCodUsu());
        $tablero2=$ctrlPartida->mascaraTablero($partida->getTablero2());
        $partida->setTablero2($tablero2);

        responseJson($obUsu, $partida, "0");
    }
    break;

case "3": #PROCESAR DISPARO USUARIO

    $x=$json['x']; //Recoge las coodenadas mandadas x json
    $y=$json['y'];
    procesarComprobacionDisparo($obUsu,$x,$y);
    break;
```

Figura 21 y 22. Código partida.php. Manejo de las peticiones http asíncronas

```

case "4": #PROCESAR RECARGA... dispara señor servidor...

$ganador;
$car_gan;
$ctrlPartida->dispara_señr_servidor($obUsu->getCodUsu());

$partidaActualizada = $ctrlPartida->obtenerPartida($obUsu->getCodUsu());

#Comprobar ganador
$ganador = $ctrlPartida->comprobarGanador($partidaActualizada->getTablero1());

if ($ganador == true) {
    $car_gan = "2"; //Gana señor servidor
} else {
    $car_gan = "0"; //No hay ganador
}

#Enmascarar tablero boot
$tablero2=$ctrlPartida->mascaraTablero($partidaActualizada->getTablero2());
$partidaActualizada->setTablero2($tablero2);
responseJson($obUsu, $partidaActualizada, $car_gan); #Response server

break;

case "5": #ABANDONAR PARTIDA
$ctrlPartida->borraPartida($obUsu->getCodUsu());
$ok=1;
$oka= array ("ok"=>$ok);
echo json_encode($oka, JSON_FORCE_OBJECT,3); #Response server
break;

case "6": #ABANDONAR PARTIDA Y CERRAR SESION
$ctrlPartida->borraPartida($obUsu->getCodUsu());
$ctrlUsu = new ControlUsuario();
$erro = $ctrlUsu->cambiarEstado($obUsu->getCodUsu(), 0);
logout();

```

Figura 23. Código partida.php. Manejo de las peticiones http asíncronas

El resultado de estas peticiones es un JSON con la información necesaria para que el cliente procese la respuesta, generalmente son actualizaciones de la partida.

La mayoría de las respuestas JSON son generadas con la función `responseJson()` del archivo `funciones.php`

```

22  /*-----
23  ' Nombre: responseJson
24  ' Proceso: Concatena los datos de usuario y de la partida para mandar una respuesta http
25  ' Entradas: un objeto Usuario, Un objeto partida, y un entero que indica quien es el ganador.
26  ' Salidas: no tiene, envia la cadena por http al cliente en background
27  '-----*/
28  function responseJson($usu, $partida, $ganador)
29  {
30      $respuesta= array("usuario"=>$usu->getCodUsu(),
31                      "partida"=>$partida->getTablero1() . "|" . $partida->getTablero2(). "|",
32                      "turno"=>$partida->getTurno(),
33                      "ganador"=>$ganador);
34      echo json_encode($respuesta,JSON_FORCE_OBJECT,512);
35  }

```

Figura 24. Código Response Json

clases.php Clase CerebroServidor

La clase que se encarga de manejar la lógica de juego del boot para la partida. El siguiente código muestra el método CerebroServidor::dondeEstoy() que mapea la posición de disparo y devuelve la posición según el mapa lógico. El resultado de este método es uno de los factores con los que se determina la posición el siguiente tiro.

```

/ ****
Mapa conceptual:
1222222223
8999999994
8900000094
8900000094
8900000094
8900000094
8900000094
8900000094
8999999994
7666666665

Mapa Conceptual resumen:
0- Zona centro. Al menos una d
1- Esquina Superior Izquierda
2- Borde Superior
3- Esquina Superior Derecha
4- Borde Lateral Derecho
5- Esquina Inferior Derecha
6- Borde Inferior
7- Esquina Inferior Izquierda
8- Borde Lateral Izquierdo
9- Posición pre-Borde

```

```

private function dondeEstoy($posicion){
    $posicionConceptual=0; //Por defecto estamos en zona 0
    if ($posicion==0){
        $posicionConceptual=1;
    }else if ($posicion>0&&$posicion<9){
        $posicionConceptual=2;
    }else if ($posicion==9){
        $posicionConceptual=3;
    }else if($posicion==90){
        $posicionConceptual=7;
    }else if ($posicion>90&&$posicion<99){
        $posicionConceptual=6;
    }else if($posicion==99){
        $posicionConceptual=5;
    }else if(intval($posicion/10)==0){
        $posicionConceptual=8;
    }else if($posicion%10==9){
        $posicionConceptual=4;
    }
    return $posicionConceptual;
}

```

CAPÍTULO 5: PRUEBAS DE SOFTWARE

5.1.-INTRODUCCIÓN.

La pruebas de software son la investigación empírica para comprobar el correcto comportamiento de las especificaciones implementadas en un software. Existen varios tipos de pruebas: pruebas unitarias, pruebas de carga(que miden el rendimiento), pruebas de integración y pruebas funcionales.

5.2.- TÉCNICAS DE PRUEBA.

Para este proyecto se han definido dos tipos de prueba: Pruebas de caja blanca y pruebas de caja negra. Se definen a continuación.

5.2.1.- PRUEBAS DE CAJA BLANCA O ENFOQUE ESTRUCTURAL.

No se han definido como tal, el código ha ido probándose poco a poco y se han usado scripts de php propios como conductores para probar las entradas y salidas de los métodos.

5.2.2.- PRUEBAS DE CAJA NEGRA O ENFOQUE FUNCIONAL.

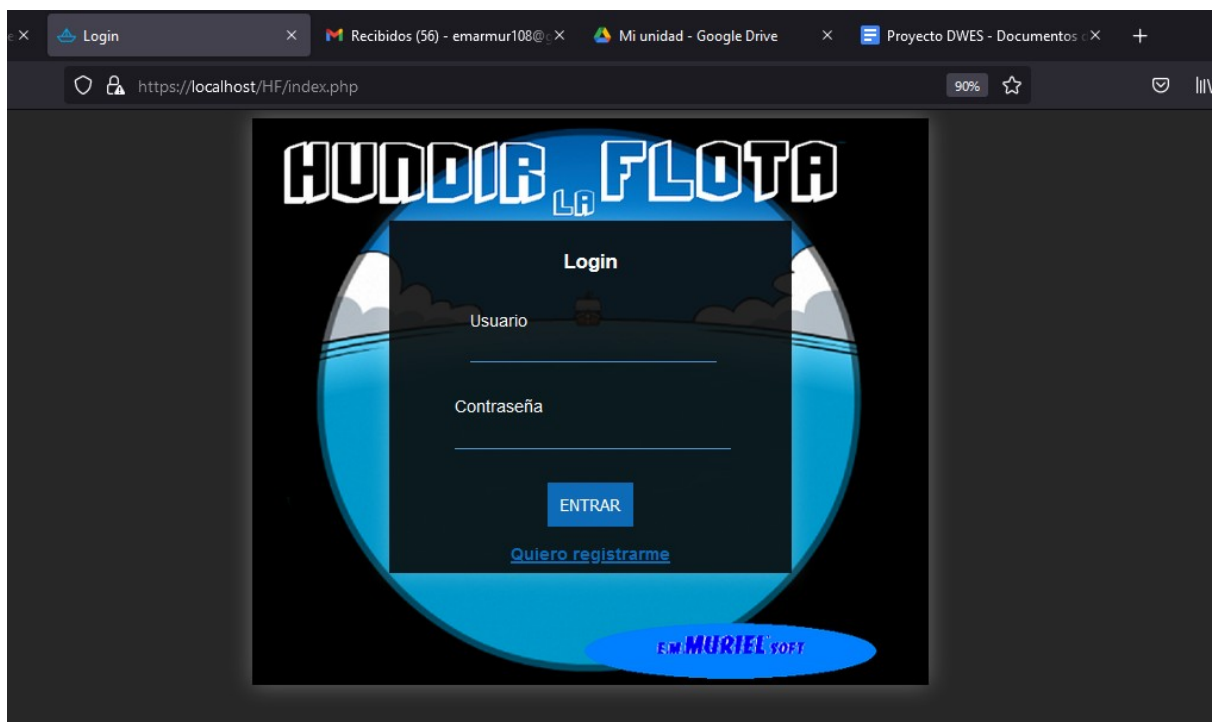
El principal objetivo de esta prueba es probar la funcionalidad / comportamiento de la aplicación. Los casos de prueba deben redactarse según la Especificación de requisitos. El testeador desconoce el funcionamiento de la aplicación, es una prueba que se basa en entradas y salidas. Los casos de prueba tendrán más detalles sobre las condiciones de entrada, los pasos de prueba, los resultados esperados y los datos de prueba.

Prueba 1:

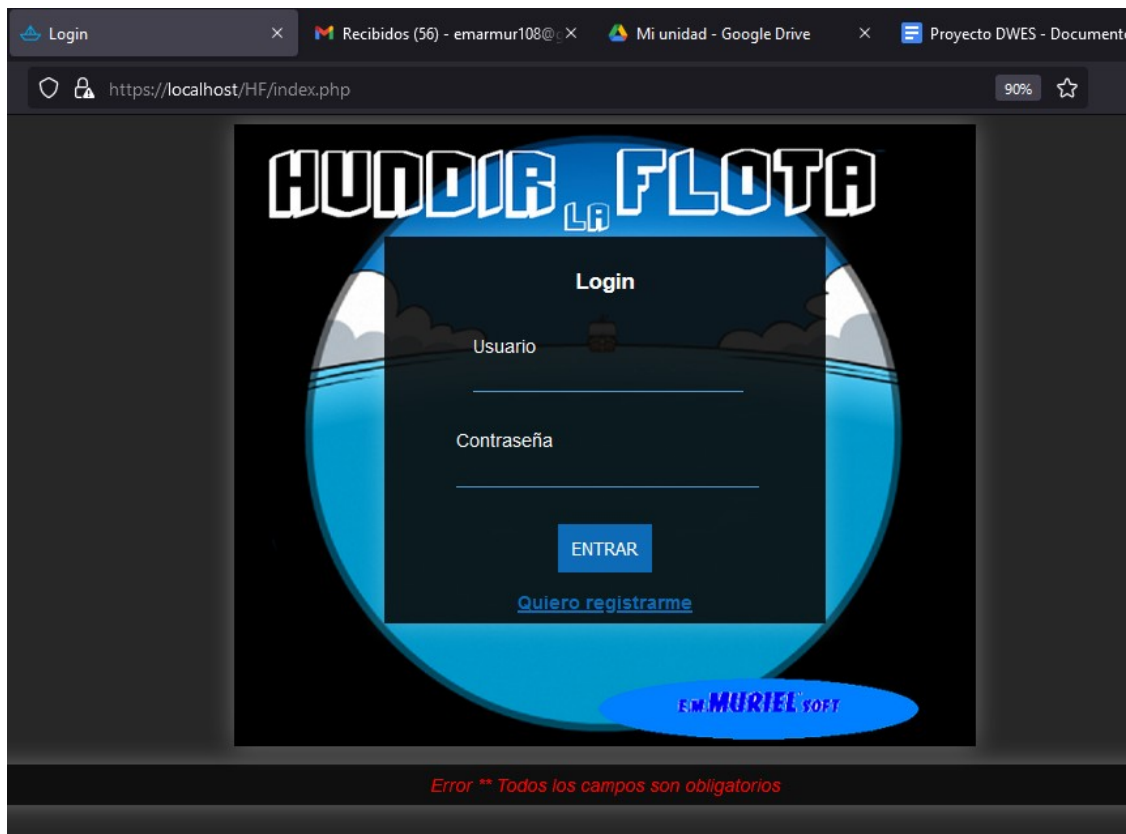
Interfaz: index.php

Entrada:

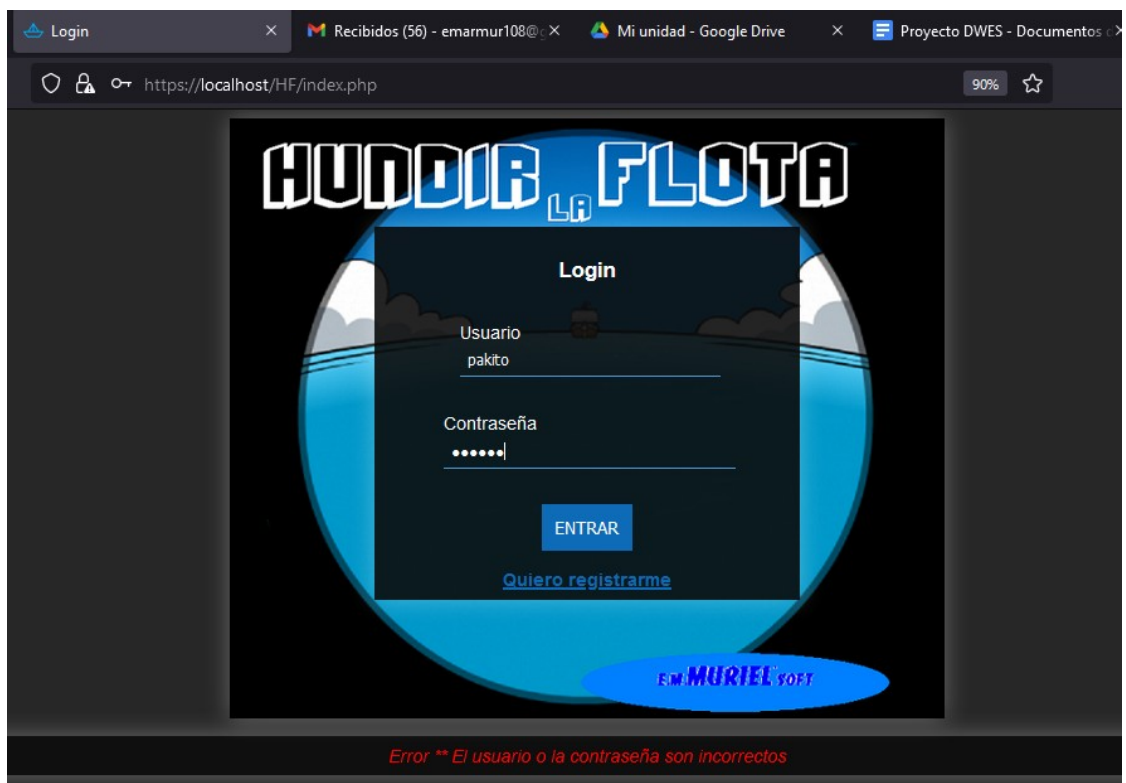
Login . Index.php



Error de Login. Campos vacíos.



Error de login: Usuario o contraseña no válido:



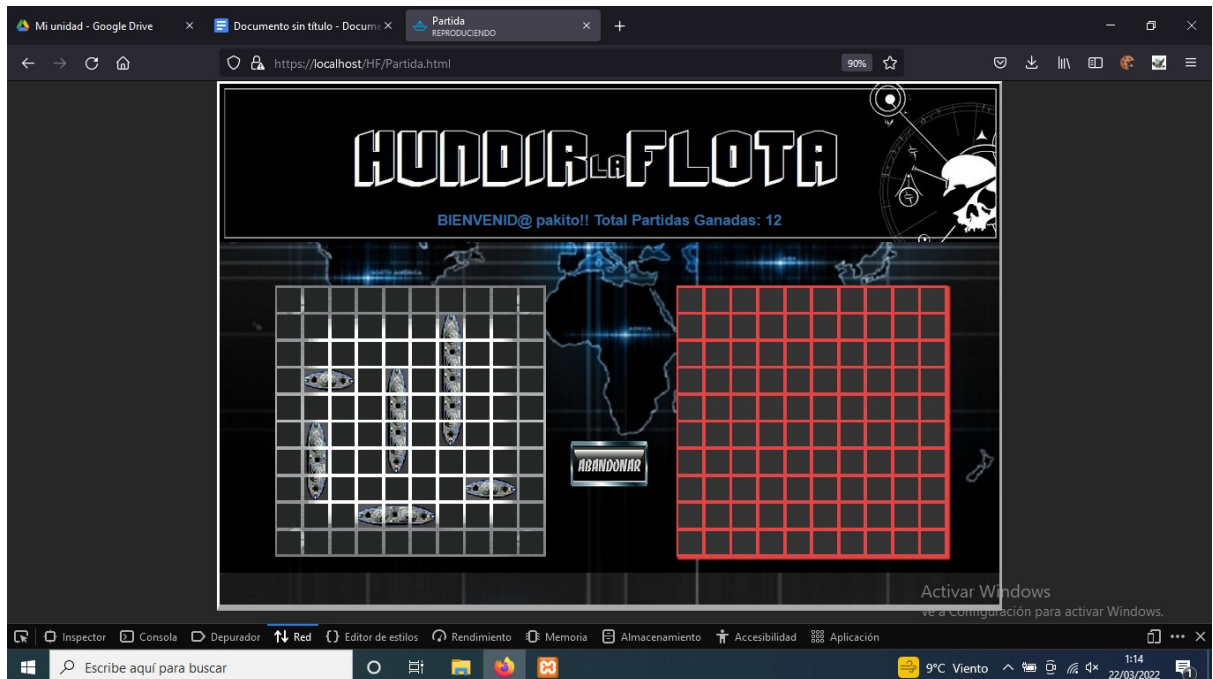
Login correcto. Carga datos de usuario . Partida.html



Ejecutado sobre	Núm.	Descripción	Resultado
index.html	T01	Fomulario de autenticación. Error: Campos vacíos	T01
index.html	T02	Fomulario de autenticación. Error: Credenciales	T02
index.html	T03	Fomulario de autenticación. Login correcto	T03

Pruebas sobre la partida: Cargar partida

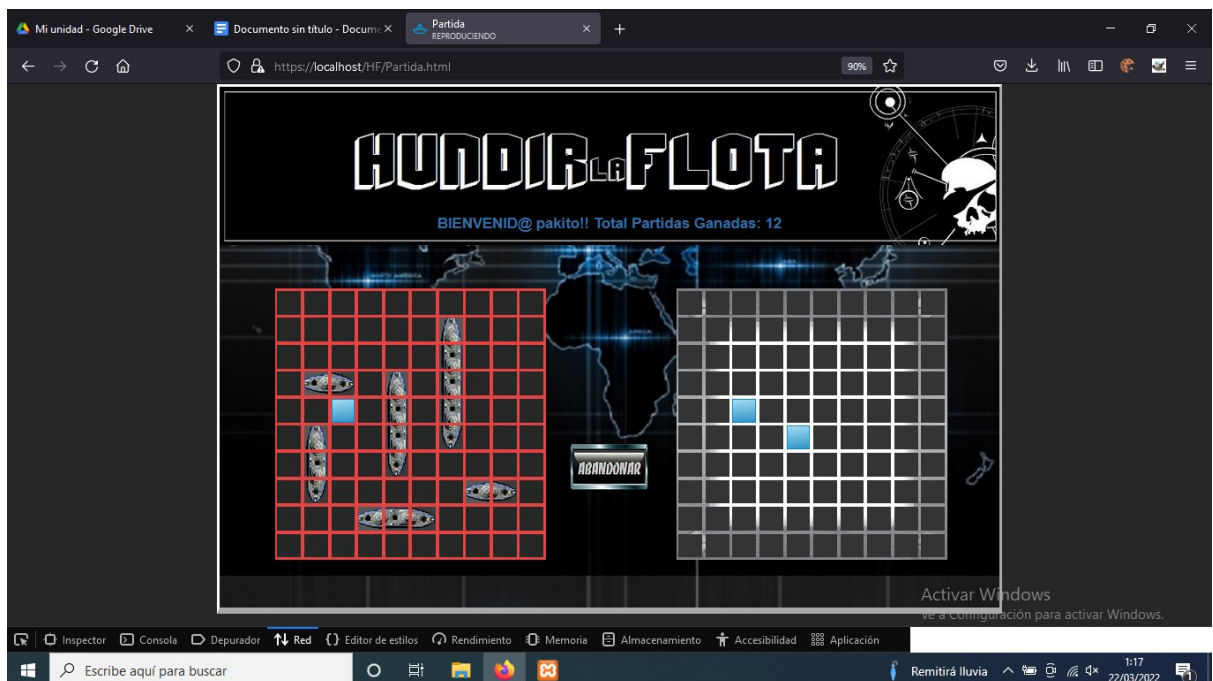
Carga correcta de una nueva partida con el botón “Jugar”

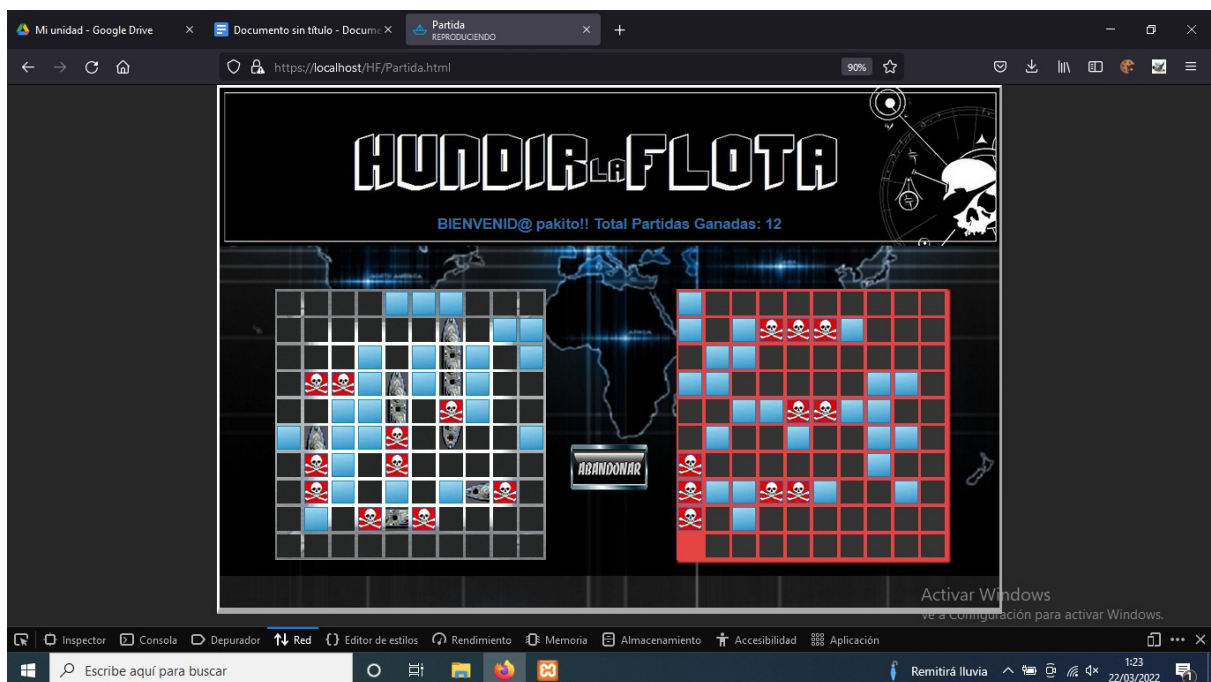
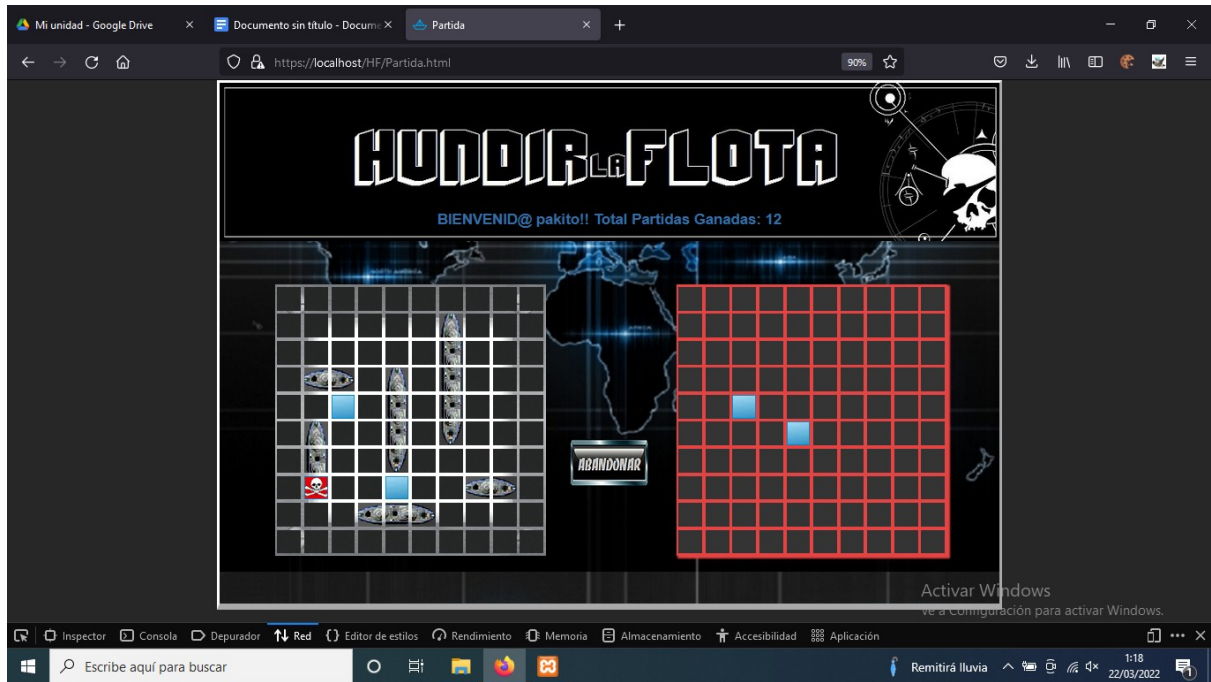


Siempre empieza el Usuario. Es decir el tablero en rojo.

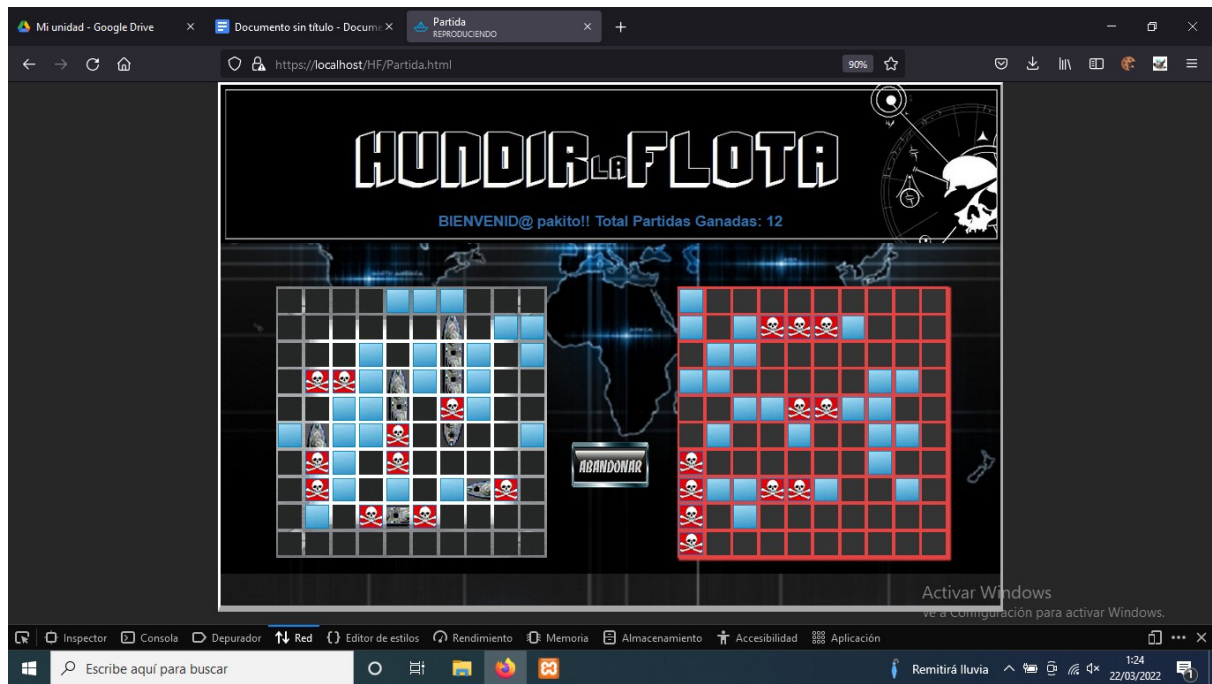
Pruebas sobre la partida: Probar disparos y manejo del turno:

Tras un disparo fallido el turno pasa al servidor. Podemos ver el color rojo en nuestro tablero.

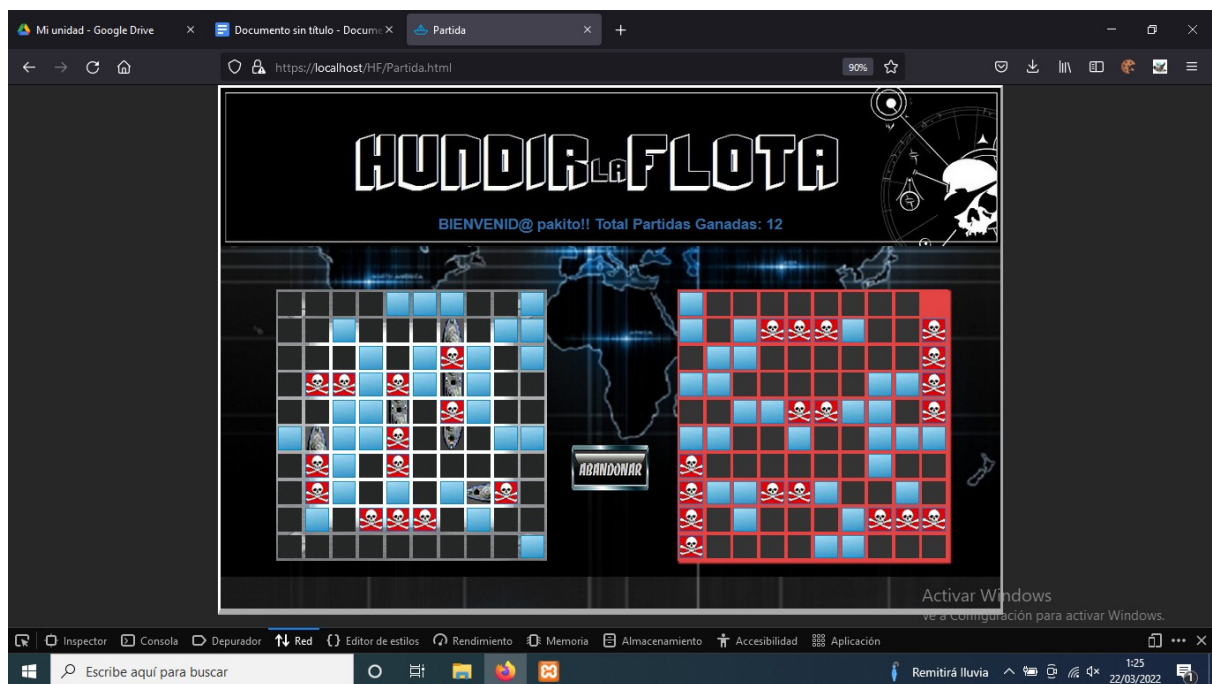


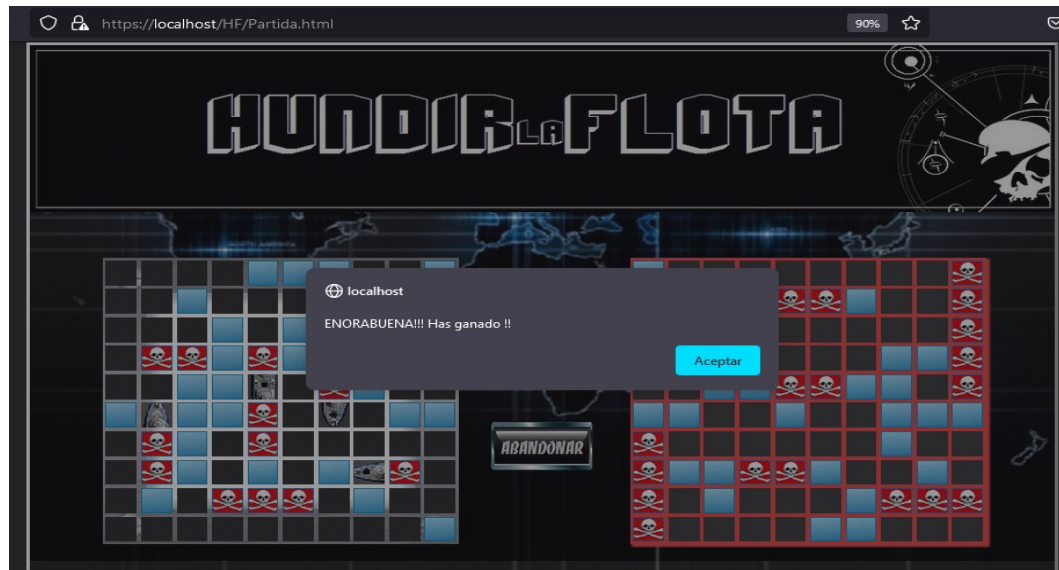


Podemos observar como despues de acertar el disparo seguimos teniendo el turno



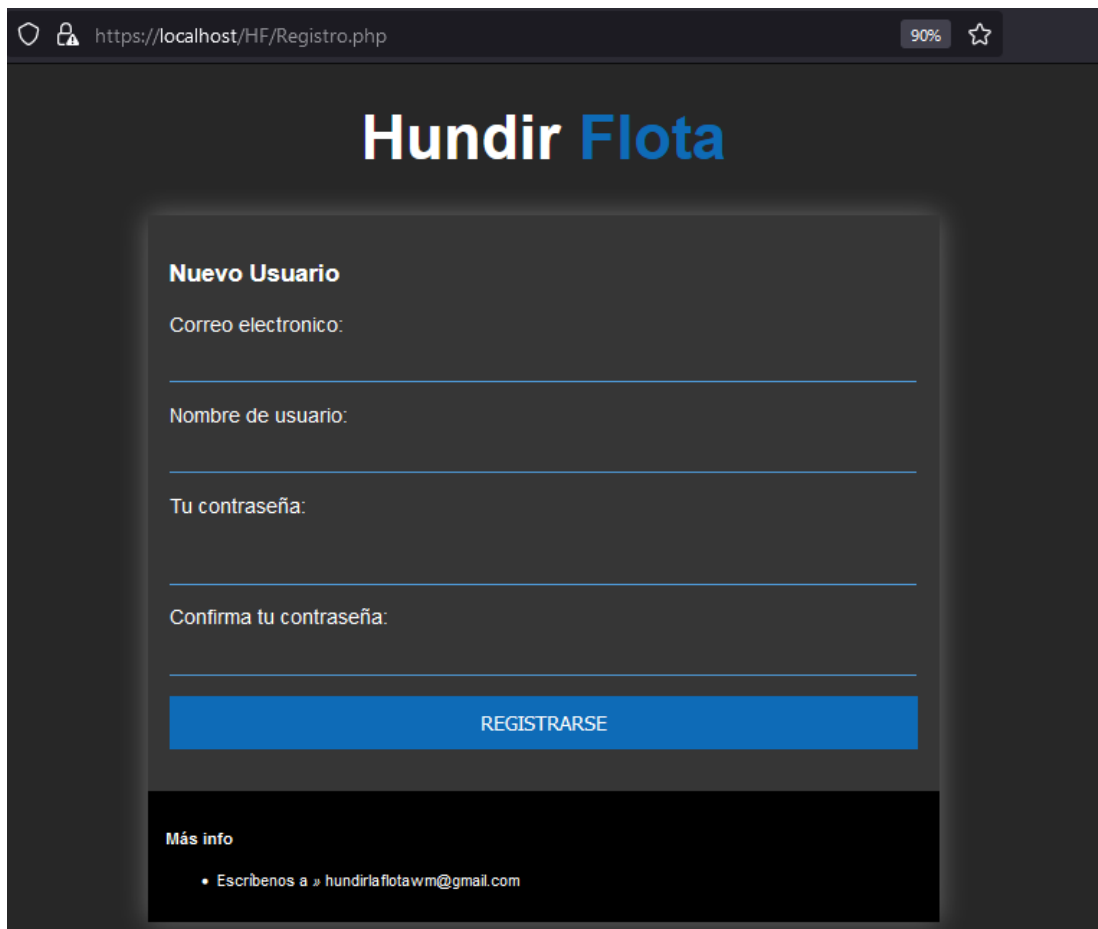
Pruebas sobre la partida: Ganador usuario





Ejecutado sobre	Num. Descripción de la prueba	Resultado
Partida.html	T04. Carga de la página con todos los elementos	Test 04
Partida.html	T05. Carga de partida nueva.	Test 05
Partida.html	T06. Carga de partida guardada.	Test 06
Partida.html	T07. Disparo fallido jugador. Actualiza página	Test 07
Partida.html	T08. Disparo fallido jugador. Cambia turno	Test 08
Partida.html	T09. Disparo fallido boot. Actualiza página	Test 09
Partida.html	T10. Disparo fallido boot. Cambia turno	Test 10
Partida.html	T11. Jugador gana. Actualiza página. Mensaje victoria	Test 11
Partida.html	T12. Jugador gana. Suma victoria a su marcador	Test 12
Partida.html	T13. Jugador pierde. Actualiza página. Mensaje victoria	Test 13
Partida.html	T14. Abandonar Partida. Elimina partida y Actualiza página	Test 14
Partida.html	T15. Jugador gana. Actualiza página	Test 15
Partida.html	Total Test: 11 Total positivos: 11	

Formulario de Registro. Registro.php



The screenshot shows a web browser window with the URL `https://localhost/HF/Registro.php`. The page has a dark background with the title "Hundir Flota" in large, bold, blue and white letters. Below the title is a registration form titled "Nuevo Usuario". The form contains four input fields: "Correo electronico:", "Nombre de usuario:", "Tu contraseña:", and "Confirma tu contraseña:". Below these fields is a blue button labeled "REGISTRARSE". At the bottom of the form, there is a section titled "Más info" with a bullet point: "• Escribenos a » hundirlaflotawm@gmail.com".

T16 Errores de formulario. Campos vacios



The screenshot shows the same registration form as above, but with an error message at the bottom: "**Error: Todos los campos son obligatorios". The error message is in red text. The form fields are empty, and the "REGISTRARSE" button is still visible.

Correo/ usuario ya registrado.

Ejecutado sobre	Num.	Descripción de test	Resultado
Registro.php	T16.	Redireccion correcta	Test 16
Registro.php	T17.	Valida campos vacios. Muestra error	Test 17
Registro.php	T18.	Valida campo email. Muestra error	Test 18
Registro.php	T19.	Valida formato contraseña. Muestra error	Test 19
Registro.php	T20.	Valida contraseña dos veces. Muestra error	Test 20
Registro.php	T21.	Valida nombre usuario único. Muestra error	Test 21
Registro.php	T22.	Valida email único. Muestra error	Test 22
Registro.php	T23.	Registra usuario.	Test 23
Registro.php	T24.	Redirige a pantalla de éxito tras registro.	Test 24
Registro.php	Total Test: 9		Total positivos:9

CAPÍTULO 6: CONCLUSIONES.

6.1.- CONCLUSIONES

Es evidente que la falta de experiencia a la hora de diseñar el software ha jugado un papel en contra a lo largo del desarrollo del proyecto y ha mermado el catálogo de requisitos previos. La toma de decisiones como descartar la incorporación del modo 2 jugadores, para evitar el fenómeno de “long pool” por las llamadas asincrónicas y por la dificultad de incorporar el uso websockets en PHP y su despliegue en servidor, ha sido condicionada por la elección del lenguaje de servidor, habiendo lenguajes actuales con mejores librerías para el uso de estas tecnologías “en tiempo real”. Desde luego es una experiencia que ha ayudado a comprender mejor la importancia crítica del estudio de la viabilidad del sistema y su alcance y la investigación previa para justificar la elección de las tecnologías de desarrollo evitando el posible fracaso del proyecto.

6.2.- PROPUESTAS FUTURAS (OPCIONAL).

Dada las conclusiones hechas, se proponen a futuro algunas soluciones a los problemas encontrados así como la implementación de requisitos que no han sido satisfechos:

- Migración del backend con todas sus funcionalidades a lenguaje Nodejs, que aporta mejores librerías de desarrollo para los propósitos de la aplicación, así como otras mejoras.
- Implementación de partidas en tiempo real entre dos jugadores con el uso de tecnologías de web sockets.
- Implementar un pequeño chat en vivo en la partida de dos jugadores para permitir la comunicación entre ellos.
- Mejoras en la inteligencia del bot, como crear una tabla para almacenar de cada casilla el número de veces que se tira y el número de veces que se acierta, pudiendo calcular así un ratio de acierto que pueda ser usado para la toma de decisiones a la hora de calcular la próxima jugada.

CAPÍTULO 7: **BIBLIOGRAFÍA Y REFERENCIAS**

7.1.-REFERENCIAS BIBLIOGRÁFICAS.

Librerías de php.

<https://www.php.net>

Imitar la sobrecarga de constructores en php.

<https://desarrolloweb.com/articulos/sobrecarga-constructores-php.html>

Multitud de consultas (errores, Json, etc)

<https://stackoverflow.com>

<https://es.stackoverflow.com>

<https://www.anerbarrena.com/php-array-tipos-ejemplos-3876/>

https://coderwall.com/p/p2kumg/json_encode-vs-serialize-with-php-arrays

https://www.w3schools.com/php/func_string_str_split.asp

<https://stackoverflow.com/questions/804045/preferred-method-to-store-php-arrays-json-encode-vs-serialize>

<https://tursos.com/como-definir-el-tiempo-de-vida-de-una-variable-de-sesion-en-php/>

<https://mixkit.co/free-sound-effects/>

Guía sobre el uso de cookies: <https://www.aepd.es/es/documento/guia-cookies.pdf>

ANEXO 1: MANUAL DE INSTALACIÓN

Configuración de un entorno local para ejecutar la aplicación:

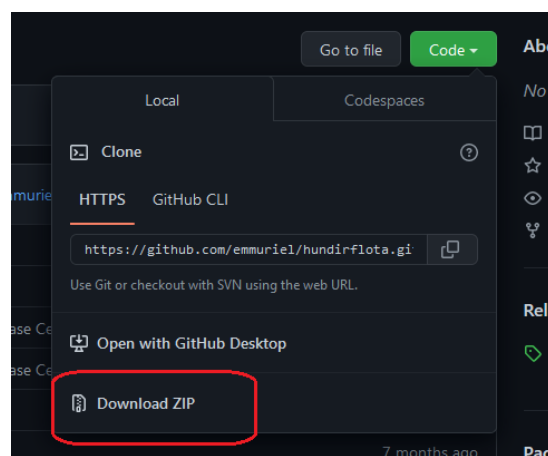
SO Windows 10 , 11 :

1. Descargue el paquete XAMPP para Windows. Puede descargarlo aquí en el sitio oficial <https://www.apachefriends.org/es/index.html>

2. Instale el paquete con el instalador de Windows. Esto instalará un servidor web apache con el módulo de PHP y un SGBD.

3. Descargue el .zip del repositorio de la aplicación:

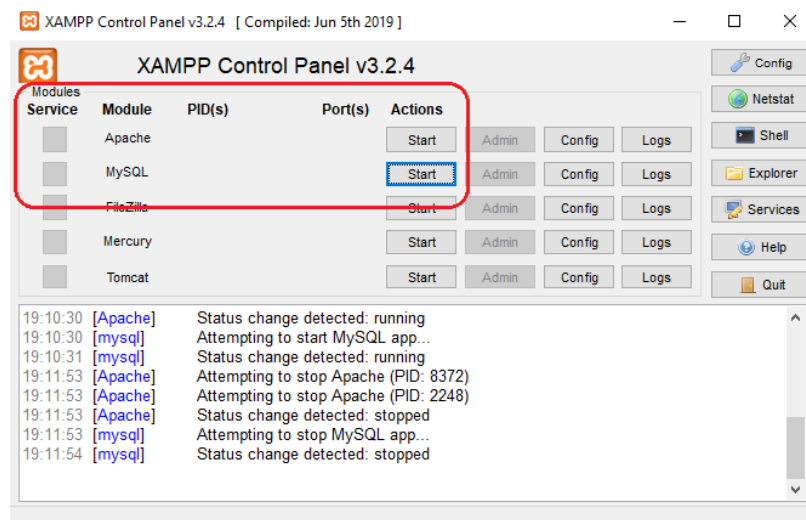
https://github.com/emmuriel/hundirflota/tree/Intelligent_boot



4. Extraiga los ficheros del .zip

5. Copie la carpeta de código fuente “hundirflota” en el directorio htdocs de XAMPP. (La configuración del DocumentRoot puede ser cambiado en *Apache httpd.conf*)

6. Abra el panel de control de XAMPP. Arranque el servidor web apache y el SGBD MySQL haciendo click en el boton start.

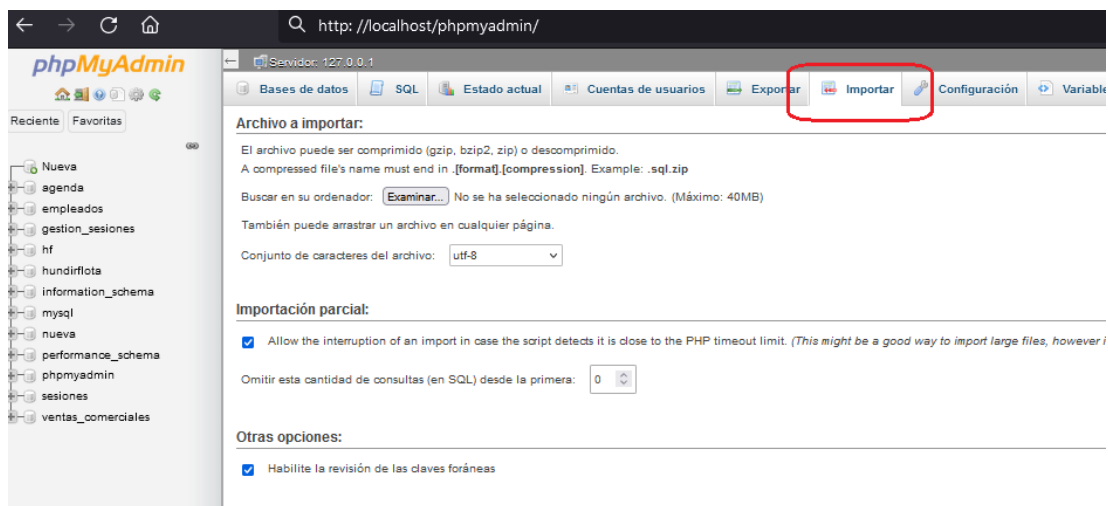


Procure que los puertos 80 y 445 no estén siendo usados por otra aplicación. Si lo necesitara puede cambiar la configuración de los puertos en el archivo `apache.config`.

7. Crear la Base de datos local.

Abra el panel de control Mysql *phpmyadmin*. Puede hacerlo desde el panel de control de XAMPP o agregando la siguiente ruta en la barra de direcciones de su navegador :

http: //localhost/phpmyadmin/



En la opción importar importe el archivo `.sql` que encontrará dentro de la carpeta BBDD del `.zip` que ha descargado de github.

Este script contiene la creación y carga inicial para pruebas de la base de datos con tablas y procedimientos almacenados.

Verifique que la base de datos ha sido creada correctamente.

Es necesaria la creación de un usuario en el SGBD con los permisos necesarios para ejecutar las operaciones, por favor, créelo en modo gráfico y asignele los siguientes permisos para la base de datos “hundirlaflota”.

Puede usar un usuario ya creado modificando el valor en el script de php “`hundirlaflota/models/moduloConexion.php`” con el usuario que tenga permisos en su base de datos. La configuración de privilegios recomendada es la siguiente:

Bases de datos SQL Estado actual Cuentas de usuarios Exportar Importar Configuración

Global Base de datos Change password Información de la cuenta

Editar los privilegios: Cuenta de usuario 'hfadmin'@'%'

Privilegios específicos para la base de datos

Base de datos	Privilegios	Conceder	Privilegios específicos para la tabla	Acción
hundirlaflota	SELECT,INSERT,UPDATE,DELETE	No	No	Editar privilegios Revocar

8. Conectarse a la aplicación web en local.

Si ha seguido todos los pasos anteriores, escriba en la barra de su navegador

<http://localhost/hundirflota/index.php>

Esto le llevará a la página principal del sitio

Si quiere entrar sin crear un nuevo usuario puede utilizar las siguientes credenciales de usuarios de prueba:

elisa/elisa

pakito/pakito

ANEXO 2: **MANUAL DE USUARIO**

Indice

1. Introducción:.....	
2. Reglas del Juego.....	
3. Registro de Usuario	
4. Acceso a la plataforma: Autenticación	
5. Menú de navegación	
6. Jugar partida	
6.1. Cargar partida	
6.2. Crear nueva partida	
6.3. Acciones sobre el tablero de juego	
7. Datos de Usuarios	
7.1. Cambio de contraseña.	
8. Ranking de jugadores	

MANUAL DE USUARIO.

1. Introducción.

El manual detallado a continuación es una guía de usuario para facilitar el uso de la aplicación web HundirLaFlota.

HundirLaFlota es una plataforma web en la que un usuario puede registrarse para jugar partidas del mítico juego “Batalla Naval” contra la máquina.

Requerimientos mínimos:

Para poder conectarse a la aplicación es imprescindible contar con un dispositivo con navegador web instalado, preferiblemente Mozilla FireFox o Google Chrome. Así como conexión a internet.

2. Normas de juego***Los tableros:***

El juego contiene 2 tableros de 10x10 posiciones. Un tablero pertenece al jugador y el otro a la máquina o boot (adversario).

Cada uno de estos tableros contiene una flota de barcos que serán el objetivo a bombardear.

La flota:

La flota se compone de:

- 1.4.- un barco de 5 casillas
- 1.5.- un barco de 4 casilla
- 1.6.- dos barcos de 3 casillas cada uno.
- 1.7.- dos barcos de 2 casillas cada uno.

Las posiciones de los barcos son suministradas aleatoriamente por el servidor para ambos tableros.

Los barcos solo pueden posicionarse en horizontal o vertical sobre el tablero, nunca en diagonal.

Las casillas limítrofes a un barco nunca podrán ser ocupadas por otro barco.

Gestión del turno de tiro:

En la aplicación el turno siempre empieza el jugador.

El juego consiste en disparar sobre el tablero del rival y descubriendo la ubicación de los barcos antes de que el rival descubra la ubicación de la flota propia.

Si el disparo alcanza un objetivo, el barco se dará por “Tocado” en esa posición, el jugador que acierta repite turno, en cambio si falla, el turno pasa al jugador contrario.

Las casillas sólo pueden ser disparadas una vez.

La partida termina cuando uno de los jugadores se rinda o cuando uno de los jugadores gane, es decir si consigue descubrir toda la flota enemiga antes de que su propia flota sea descubierta por el enemigo.

3. Registro de Usuario

El acceso a la plataforma está permitido solo para usuarios registrados y autenticados. Si es la primera vez que accede debe darse de alta como usuario registrado a través del formulario de registro.

Esta operación requerirá ciertos datos personales como una dirección de correo válida.

Para acceder al formulario de registro debe hacer click en el enlace de la página principal “Regístrate aquí” que se encuentra debajo del formulario de autenticación.

Esta acción le redirigirá al formulario de registro.

En el formulario de registro debe introducir:

Correo electrónico que no esté registrado.

Nombre de usuario (Será su identificador dentro de la plataforma)

Contraseña (Junto a su nombre de usuario será su clave para acceder a la plataforma. Debe contener mínimo 8 caracteres entre los cuales debe hacer caracteres alfabéticos, caracteres numéricos y caracteres especiales permitidos)

Repetir Contraseña : la misma contraseña que en el campo anterior

Una vez rellenado el formulario, pulse el botón “Registrarse” para darse de alta.

Si la operación ha sido exitosa será redirigido a un mensaje de éxito, de lo contrario la operación no se habrá registrado y la aplicación informará del motivo del error.

4. Acceso a la plataforma: Autenticación

Desde la página principal de la aplicación, tras prestar su consentimiento al uso de cookies por parte de la plataforma, podrá acceder al formulario de autenticación o login.

En este formulario debe intrucir su nombre de usuario registrado en la plataforma y contraseña, seguidamente haga click sobr eel botón “Entrar”

Si ocurre algún error durante el proceso de autenticación , la plafaforma mostrará un mensaje con el motivo del error. De lo contrario se cargará la página principal.

5. Menu de navegación.

6. Jugar partida

La pantalla de partida se muestra por defecto al entrar en el menú principal. En este momento la partida aún no se ha cargado por lo que los tableros de juego aparecerán vacíos.

6.1. Cargar partida

Para cargar la partida haga click el boton “Jugar” que se encuentra entre los dos tableros vacíos.

6.2. Crear nueva partida

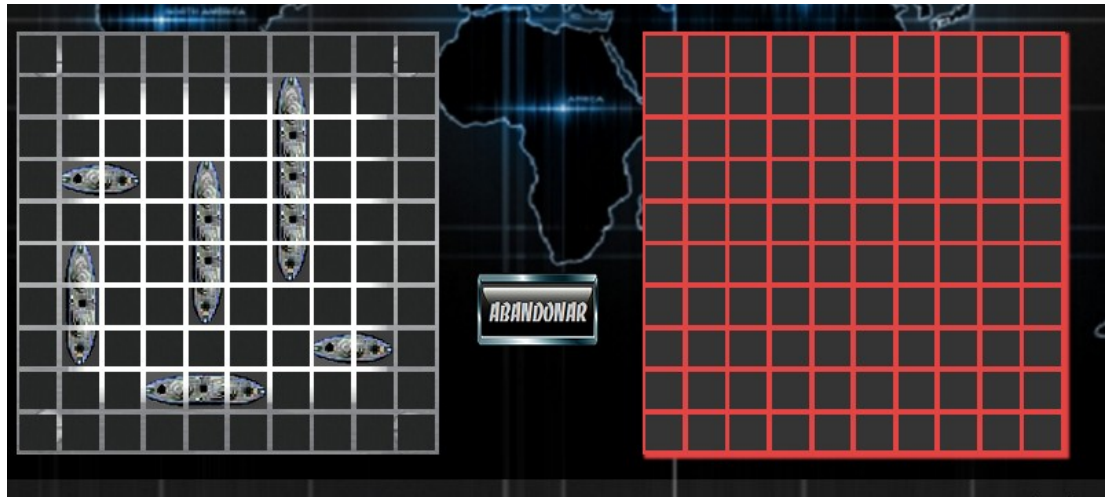
Si usted tiene una partida anterior inconclusa se cargará automáticamente. Si desea seguir jugando esta partida puede eliminarla pulsando el botón “abandonar” que se encuentra entre los dos tableros. Este botón elimina inmediatamente la partida en curso.

Para crear la nueva partida pulse de nuevo el botón “Jugar” y la nueva partida será cargada.

6.3. Acciones sobre el tablero de juego

Usted puede realizar acciones sobre el tablero de juego de su rival siempre y cuando tenga el turno de juego. Ninguna acción fuera de su turno de juego será realizada sobre el tablero.

El turno de juego será indicado visualmente. El tablero cuyos bordes se encuentra coloreado de rojo será el tablero sobre el que se van a ejecutar acciones.



Ejemplo 1. Partida nueva recién cargada. El turno es del usuario.

El tablero izquierdo pertenece al jugador, será el tablero sobre el cual el boot ejecutará sus jugadas. El tablero de la derecha pertenece al boot y es sobre el cual el Jugador ejecutará sus jugadas.

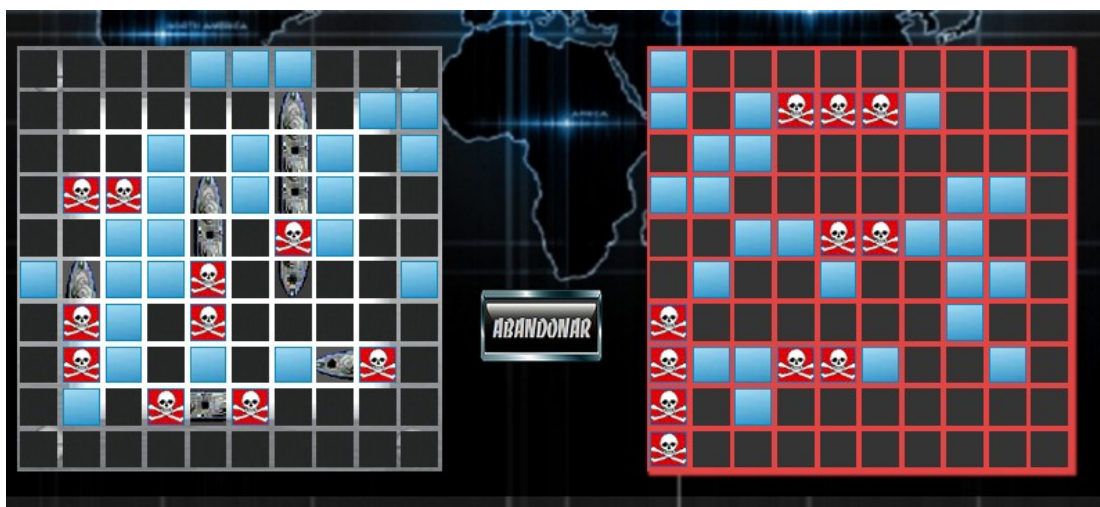
Seleccionar una casilla para ejecutar un disparo:

Las casillas aun por explotar son de color gris oscuro. Si pasamos el ratón sobre ellas veremos como el color de la casilla cambia a rojo.

Al hacer click sobre la casilla ejecutamos el disparo sobre ella.

Resultados posibles del disparo:

El disparo solo puede tener dos resultados: Agua (Casilla azul si se ha fallado) o Tocado, casilla roja con dibujo de calavera.



Ejemplo 2. La imagen muestra la partida en curso, con los resultados de los disparos efectuados.

El jugador que acierta al ejecutar el disparo mantiene su turno y tira otra vez, el jugador que falla al ejecutar su disparo pierde el turno de juego y tiene que esperar a que el rival falle para recuperarlo.

7. Datos de Usuarios

Para ver los datos guardados sobre su usuario dirijase hacia el icono de usuario en la esquina superior derecha. Haga click sobre el icono de usuario.

La aplicación abrirá un menú desplegable con las opciones 1. Mi cuenta , 2. Log off
Haga click sobre la opcion 1. Mi cuenta.

Esto le redirige hacia la pantalla micuenta donde podrá ver los datos almacenados de su cuenta.

7.1. Cambio de contraseña.

Dentro del menú de usuario diríjase a la opcion “Cambiar contraseña”.

El sistema le solicitará tres campos de texto:

Campo 1: La actual contraseña

Campo 2: La nueva contraseña (recuerde que debe cumplir todos los requisitos de seguridad)

Campo 3: Repetir nueva contraseña.

8. Ranking de jugadores

En el menú de navegación de la página principal puede encontrar la opción “Ranking jugadores”. Haga click sobre la opción para ver el ranking de jugadores. Puede volver atrás navegando a través de la misma barra de navegación