



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Plan de combate electrónico: Diseño de una herramienta web
para la elaboración del plan de combate en buques de la Armada*

Grado en Ingeniería Mecánica

ALUMNO: D. Joaquín Garat Loureiro, AF

DIRECTORES: D. Norberto Fernández García
D. Rafael Asorey Cacheda

CURSO ACADÉMICO: 2015-2016

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Plan de combate electrónico: Diseño de una herramienta web
para la elaboración del plan de combate en buques de la Armada*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

Este TFG se plantea como el desarrollo de una aplicación *web* para mejorar la creación y gestión de los planes de combate en los buques de la Armada.

Un plan de combate es un documento en el que se presentan los puestos asignados a cada miembro de la dotación de un buque, en cualquier circunstancia en la que se encuentre el buque.

Los planes de combate están sujetos a un alto nivel de variación debido a la naturaleza cambiante de las dotaciones. Es por esta razón que se considera la necesidad de una aplicación que facilite la gestión de esta clase de documentos.

Esta aplicación se desarrolla utilizando un sistema LAMP, que coordina las herramientas necesarias (Linux, Apache, MySQL y PHP) para poder desarrollar aplicaciones *web* dinámicas.

Desarrollada la aplicación, se llevan a cabo periodos de validación a nivel interno para comprobar el adecuado funcionamiento de la aplicación, y a nivel externo, para evaluar las capacidades de la herramienta en un entorno realista.

Finalmente se concluye que esta aplicación se presenta como un enfoque válido, aunque no completo, al problema de la gestión de los planes de combate. Para completar la aplicación se establecen una serie de líneas futuras como la creación de relaciones entre puestos o el uso del lenguaje JavaScript para aumentar el carácter dinámico de la aplicación.

PALABRAS CLAVE

Aplicación *Web*, Plan de Combate, PHP, HTML, MySQL, Apache

AGRADECIMIENTOS

A mi familia por su apoyo incondicional. A mis compañeros, y amigos, de camareta, por aguantar con buena cara mis monólogos acerca de este TFG. A mis compañeros y amigos Luis Astorga y Pablo Martínez, por ayudarme a superar las dificultades de este proyecto y motivarme a seguir trabajando día a día. A mi comandante de brigada y a mis tutores por su inestimable ayuda. Y especialmente a mi hermano Rafa, por haber seguido muy de cerca el transcurso de mi carrera y por la inestimable ayuda que me ha prestado a lo largo del desarrollo de este TFG.

CONTENIDO

Contenido	1
Índice de Ilustraciones	4
Índice de Tablas.....	6
1 Introducción y objetivos	7
1.1 Motivación	7
1.2 Los manuales de organización	8
1.3 Propuesta de mejora	13
1.4 Organización de la memoria	13
2 Estado del arte	14
2.1 Situación en la Armada	14
2.2 Situación en el sector civil	15
2.2.1 Autotask.....	16
2.2.2 GestPeople	16
2.2.3 WorkMeter.....	17
2.2.4 E-Volution	18
2.2.5 SimpleHRM.....	18
2.2.6 eMagister RRHH	19
2.2.7 Comparativa.....	20
2.3 Conceptos informáticos.....	21
2.3.1 Lenguaje de marcado, HTML.....	21
2.3.2 Lenguaje de programación, PHP	22
2.3.3 Sistemas de Gestión de Bases de Datos, MySQL.....	23
2.3.4 Servidor web, Apache.....	24
2.3.5 Linux, Apache, MySQL, PHP (LAMP)	25
3 Desarrollo del TFG	26
3.1 Instalación de un entorno de programación web.....	26
3.1.1 Instalación de una máquina virtual Ubuntu	26
3.1.2 Instalación del sistema LAMP	27
3.1.3 Creación de un entorno de programación web	27
3.2 Creación y estructuración de la base de datos.....	29
3.2.1 Estructura de la base de datos	29
3.2.2 Conceptos SQL	29
3.3 Introducción a PHP	30

3.3.1 Patrón MVC	31
3.3.2 Las bases del lenguaje PHP	32
3.4 Introducción a HTML	34
3.4.1 Texto	34
3.4.2 Enlaces	35
3.4.3 Tablas.....	35
3.4.4 Formularios.....	36
3.5 CSS y la etiqueta <code><style></code>	36
3.6 Código de la Aplicación.....	37
3.6.1 Estructura de la Aplicación.....	37
3.6.2 Index.php	38
3.6.3 Vista1.php	38
3.6.4 Vista2.php	39
3.6.5 Vista3.php	40
3.6.6 Vista4.php	41
3.6.7 Vista5.php	42
3.6.8 Generarpdf.php	42
3.6.9 Control.php	43
3.6.10 Modelo.php	44
4 Validación	45
4.1 Validación Interna.....	45
4.2 Validación Externa.....	58
5 Conclusiones y líneas futuras	59
5.1 Conclusiones	59
5.2 Líneas Futuras	59
6 Bibliografía.....	61
Anexo I: Base de Datos	66
Anexo II: index.php.....	67
Anexo III: vista1.php.....	68
Anexo IV: vista2.php	73
Anexo V: vista3.php.....	75
Anexo VI: vista4.php	77
Anexo VII: vista5.php	79
Anexo VIII: generarpdf.php	82

Anexo IX: control.php 88

Anexo X: modelo.php 92

ÍNDICE DE ILUSTRACIONES

Ilustración 1-1 : MANORG de las Fragatas F-100.	8
Ilustración 1-2 : Sección de un Plan de Combate.	10
Ilustración 2-1 : Portada de la D-CP-03. Tomada de [3].	14
Ilustración 2-2 : Interfaz de la aplicación <i>web AutoTask</i> . Tomada de [42].	16
Ilustración 2-3 : Interfaz de la aplicación <i>web GestPeople</i> . Tomada de [43].	17
Ilustración 2-4 : Interfaz de la aplicación <i>web WorkMeter</i> . Tomada de [44].	18
Ilustración 2-5 : Interfaz de la aplicación <i>web E-Volution</i> . Tomada de [45].	18
Ilustración 2-6 : Interfaz de la aplicación <i>web SimpleHRM</i> . Tomada de [46].	19
Ilustración 2-7 : Interfaz de la aplicación <i>web eMagister RRHH</i> . Tomada de [47].	20
Ilustración 2-8 : Logotipo de HTML5. Tomado de [48].	22
Ilustración 2-9 : Logotipo de PHP. Tomado de [49].	23
Ilustración 2-10 : Logotipo de MySQL. Tomado de [50].	24
Ilustración 2-11 : Logotipo de Apache. Tomado de [51].	25
Ilustración 2-12 : Logotipo de LAMP. Tomado de [52].	25
Ilustración 3-1 : Logotipo del sistema operativo Ubuntu. Tomado de [53].	26
Ilustración 3-2 : Logotipo del programa VirtualBox. Tomado de [54].	27
Ilustración 3-3 : Respuesta del navegador con Apache funcionando.	28
Ilustración 3-4 : Respuesta del navegador con PHP correctamente instalado.	28
Ilustración 3-5 : Estructura de la base de datos.	29
Ilustración 3-6 : Patrón MVC. Tomado de [55].	31
Ilustración 3-7 : Texto	35
Ilustración 3-8 : Enlace.	35
Ilustración 3-9 : Tabla	35
Ilustración 3-10 : Formulario.	36
Ilustración 3-11 : Formulario con CSS.	37
Ilustración 3-12 : <code>index.php</code>	38
Ilustración 3-13 : <code>vista1.php</code>	39
Ilustración 3-14 : <code>vista2.php</code>	40
Ilustración 3-15 : <code>vista3.php</code>	41
Ilustración 3-16 : <code>vista4.php</code>	41
Ilustración 3-17 : <code>vista5.php</code>	42
Ilustración 3-18 : <code>generarpdf.php</code>	43
Ilustración 4-1 : Seleccionar grado de alistamiento.	45
Ilustración 4-2 : Seleccionar situación operativa.	46

Ilustración 4-3 : Seleccionar puesto.	46
Ilustración 4-4 : Asignar puesto.	46
Ilustración 4-5 : Puesto asignado.....	47
Ilustración 4-6 : Validación interna - 1ª Comprobación.....	47
Ilustración 4-7 : Introducir datos.	48
Ilustración 4-8 : Alerta por falta de datos.	48
Ilustración 4-9 : Validación interna - 2ª Comprobación.....	48
Ilustración 4-10 : Desembarcar persona.	49
Ilustración 4-11 : Validación interna - 3ª Comprobación.....	49
Ilustración 4-12 : Desembarcar dotación completa.	49
Ilustración 4-13 : Introducir nueva situación.....	50
Ilustración 4-14 : Validación interna - 4ª Comprobación.....	50
Ilustración 4-15 : Borrar situación existente.	50
Ilustración 4-16 : Validación interna - 5ª Comprobación.....	51
Ilustración 4-17 : Gestionar puestos.	51
Ilustración 4-18 : Añadir puesto.	52
Ilustración 4-19 : Formulario de nuevo puesto.....	52
Ilustración 4-20 : Puesto añadido.	52
Ilustración 4-21 : Validación interna - 6ª Comprobación.....	53
Ilustración 4-22 : Borrar puesto.....	53
Ilustración 4-23 : Puesto borrado.	53
Ilustración 4-24 : Validación interna - 7ª Comprobación.....	54
Ilustración 4-25 : Generar archivo PDF.	54
Ilustración 4-26 : Validación interna - 8ª Comprobación.....	55
Ilustración 4-27 : Seleccionar grado de alistamiento.	55
Ilustración 4-28 : Seleccionar situación operativa.	55
Ilustración 4-29 : Validación interna - 9ª Comprobación.....	56
Ilustración 4-30 : Seleccionar control.	56
Ilustración 4-31 : Seleccionar subcontrol.	56
Ilustración 4-32 Validación interna - 10ª Comprobación.....	57
Ilustración 4-33 : Seleccionar miembro de la dotación.	57
Ilustración 4-34 : Validación interna - 11ª Comprobación.....	57

ÍNDICE DE TABLAS

Tabla 1 : Relación "Grados de Alistamiento" - "Situación Operativa".....	11
Tabla 2 : Comparación entre las aplicaciones de gestión de RR.HH.....	21

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Motivación

Cualquier empresa, independientemente de a qué se dedique o de su tamaño, está integrada por personas. Por lo tanto, que la organización sea exitosa o por el contrario, un fracaso, depende completamente de las personas que en ella trabajen. Es por esta razón que el recurso humano constituye siempre el recurso más apreciado de cualquier compañía. El objetivo de toda empresa es cumplir con sus objetivos para con los clientes, ya sea vender productos, vender mano de obra, o proporcionar seguridad. La manera más eficaz de que una organización cumpla con sus objetivos es conseguir que todos los empleados apoyen y compartan los objetivos de la compañía. Para conseguir esta integración entre todos los miembros de la empresa utilizamos los sistemas de gestión de recursos humanos.

Cuando hablamos de los objetivos de una empresa, casi siempre estaremos hablando de una intención de producir más y mejor. Sea cual sea el producto, el objetivo común a toda empresa es el de tomar el producto que proporciona, mejorarlo y ampliar los destinatarios a los que va dirigido. La gestión de los recursos humanos debe contribuir al cumplimiento de esos objetivos buscando la mejora del rendimiento de los empleados. Las principales acciones que se toman para mejorar el rendimiento son las de ofrecer bienestar a los empleados y explicar claramente las tareas a desempeñar. Para conseguir esto, la gestión de los recursos humanos se plantea sus propios objetivos en concordancia con los objetivos de la empresa. Estos objetivos son clasificables en objetivos a largo plazo y a corto plazo, estos últimos a su vez subdivididos en objetivos explícitos e implícitos [1].

Los objetivos explícitos son aquellos encaminados a atraer a personas cualificadas para trabajar en la empresa y ayudarla a mejorar, retener a todos aquellos empleados considerados indispensables, motivar a los empleados a adquirir un alto nivel de compromiso e implicación con la compañía y ayudar a las personas a mejorar internamente. En resumidas cuentas son los objetivos encaminados hacia las personas como individualidades [1].

Los objetivos implícitos, por el contrario, van encaminados hacia las personas que forman la empresa como un conjunto. Estos objetivos buscan la mejora de la productividad, de la calidad de vida de los empleados en su trabajo y de asegurar que se cumple con una normativa de trabajo [1].

Los objetivos a largo plazo son aquellos que plantean el futuro de la propia empresa. Buscan establecer un camino que asegure la perdurabilidad de la empresa de cara al futuro [1].

La Armada Española, a pesar de que no busque un fin económico, es a todos los efectos una empresa. Y como cualquier otra empresa, está integrada por personas, en el caso de la Armada,

aproximadamente 31.000 efectivos. El alto número de personas que integran la Armada hace bastante irrefutable la realidad de que necesita un sistema de gestión de recursos humanos. Efectivamente la Armada cuenta con una rama de su organización enteramente dedicada al recurso personal bajo la dirección del Almirante de Personal (ALPER). Enfocando los sistemas de gestión de personal de los buques de la Armada, y más concretamente sus objetivos implícitos, nos encontramos con los “Manuales de Organización” (MANORGs). Cada buque de la Armada cuenta con un MANORG específico para el buque en cuestión y que hace la función de guiar al buque hacia el cumplimiento de los objetivos implícitos de la Armada, es decir, los de mejorar el rendimiento de la dotación del buque como conjunto [2].

La organización del personal toma algunos cambios como concepto al hablar dentro del ámbito de la Armada. Con la organización de los buques se busca proporcionar una ordenación metódica del personal y material con el que está dotado el buque para alcanzar su objetivo, que no es otro que la mayor eficacia en su empleo operativo. Esta organización debe estar íntegramente relacionada con la preparación de la dotación y el empleo seguro de los medios. Para lograr este objetivo, los buques deben ser organizados buscando alcanzar la eficiencia, la armonización, la disciplina, la jerarquía, la delegación, la capacidad de control, la permanencia y la economía de recursos humanos. Estos fundamentos no se tratan de manera aislada, sino que todos ellos, en conjunto, hacen posible que se alcance el objetivo fundamental del buque de guerra. Para alcanzar este objetivo, la organización debe tratar de conseguir, en las personas, un alto nivel de formación, instrucción y adiestramiento, disciplina, seguridad física, motivación y espíritu de equipo [3].

1.2 Los manuales de organización



Ilustración 1-1 : MANORG de las Fragatas F-100.

El camino a seguir para alcanzar estos objetivos se plasma en el manual de organización. El MANORG es una publicación, específica para cada buque, cuya finalidad es facilitar al mando la preparación, alistamiento de los medios humanos y materiales y el empleo operativo del buque para el cumplimiento de su misión. Se estructura en tres volúmenes:

1. El primer volumen trata la organización general del buque. En él se contemplan las relaciones orgánicas, deberes, atribuciones y responsabilidades del personal, en el funcionamiento general y en el administrativo.

2. El segundo es el conocido como “Plan de Combate”. En él se recogen y desarrollan de forma independiente todas las situaciones que se deben afrontar, fijando el personal que ha de cubrir cada puesto, cuál es su misión y cuál la cadena de responsabilidad.
3. El tercer y último volumen regula las actividades relativas a la vida a bordo, como guardias, servicios y cometidos diversos de la dotación.

En todo buque, el MANORG está compuesto por estructuras de mando jerarquizadas. Por lo tanto, estas estructuras siempre estarán encabezadas por el comandante y el 2º comandante apoyados por los oficiales adjuntos a ambos. El resto del personal está organizado según dos orientaciones bien diferenciadas, una para el empleo del buque (Organización Operativa) y otra para su preparación y administración (Organización Administrativa).

La primera se estructura en “Controles”, con cometidos y relaciones orgánicas de acuerdo con el alistamiento operativo ordenado. Prevé la disposición del personal para afrontar las diferentes situaciones, actividades o emergencias que se puedan presentar. Esta organización se ve representada en el plan de combate” [3].

La segunda se articula en “Servicios”, con dos cadenas de mando: una funcional a través de los “Destinos”, elemento orgánico que agrupa equipos o sistemas similares funcionalmente relacionados, y otra administrativa formada por las “Brigadas” y sus mandos naturales. La actividad a desarrollar marca la preeminencia de una sobre otra. Esta organización se ve reflejada en el volumen 1 del MANORG [3].

Dentro de la coexistencia de las dos organizaciones se puede presentar una casuística grande; por lo tanto, es preciso establecer la preeminencia de órdenes o instrucciones impartidas. Como norma general, primarán las órdenes canalizadas a través de la organización operativa. Como complemento a todo lo anterior también es necesario ordenar lo relativo a la vida a bordo, guardias, servicios, funciones y cometidos diversos de los individuos, etc., que constituye el “Régimen Interior”. Podemos encontrar todo lo referente al régimen interior en el volumen 3 del MANORG [3].

Una vez establecidas las partes en las que se subdivide el MANORG y después de analizar cada una con detenimiento se puede extraer que hay una parte que constituye una dificultad en el uso práctico del documento. El volumen 1 es simplemente un organigrama que establece la jerarquía de los miembros de la dotación bajando desde el comandante hasta el marinero más moderno, esta jerarquía es constante con la dotación y por lo tanto irrefutable, de este modo se puede decir que este volumen no representa ningún problema. El volumen 3 es un conjunto de normas de comportamiento a bordo, estas normas se basan en las normativas vigentes en la Armada y por lo tanto también adquieren un determinado grado de constancia que hace que su validez sea, una vez más, irrefutable. El volumen 2, también conocido como plan de combate, sí que presenta un problema.

Para poder entender correctamente la problemática causada por el plan de combate se ha considerado necesario realizar un estudio en profundidad para responder a la pregunta de qué es exactamente este documento.

Según la referencia [3]: “El plan de combate constituye la organización primordial a establecerse para lograr la máxima eficacia operativa del buque. Por eso, los puestos en “Zafarrancho de Combate” y el desdoblamiento en “Vigilancias” dimensionan normalmente la plantilla del buque. En su definición se detallan los órganos funcionales (controles) que la articulan, su dependencia orgánica y las relaciones funcionales.

Dicho de otro modo: Un plan de combate es un conjunto de tablas que relacionan los puestos ocupados por la dotación en cada momento. El objetivo buscado con este documento es que los miembros de la dotación efectúen sus cambios de puesto de manera rápida y eficaz. En la Ilustración 1-2 se puede observar un ejemplo de las tablas que constituyen un plan de combate.

[illegible]

Ilustración 1-2 : Sección de un Plan de Combate.

Para ordenar la capacidad operativa que deben adoptar los buques para reaccionar ante todo tipo de amenaza en un momento determinado, el mando cuenta con los “Grados de Alistamiento”. Los grados de alistamiento son unos indicadores que reflejan que cantidad de dotación debe encontrarse en sus puestos (Ejemplo: El grado de alistamiento primero exige que toda la dotación este activa). Aunque normalmente se ordenará un grado de alistamiento concreto, se podrán ordenar grados particulares cuando la situación así lo requiera (Ejemplo: La situación “3 Vigilancias Modificada” se corresponde con el grado de alistamiento tercero con una relajación en las capacidades que debe mantener el buque. Se podría ordenar si el buque prevé estar en situación de 3 vigilancias durante más tiempo que el habitual). Los detalles específicos de estos grados particulares se deben reflejar en el MANORG en función de su aplicación en cada tipo de buque. Los grados de alistamiento se reflejan en la distribución de la dotación en puestos del buque a través de las “Situaciones Operativas”. Existen además, las “Situaciones Especiales”, previstas para afrontar circunstancias particulares dentro de una situación operativa general. Algunas son comunes a todos los buques y otras específicas de cada tipo.

Los grados de alistamiento bajo los cuales puede navegar un buque son, según la referencia [3] los siguientes:

1. Grado de alistamiento PRIMERO: Proporciona el máximo alistamiento para el combate. Su propósito es el empleo de la máxima capacidad operativa del buque así como la mejor disposición para afrontar con éxito el control de daños. Todos los controles deben estar cubiertos y listos para reaccionar en cualquier momento.
2. Grado de alistamiento SEGUNDO: Permite alcanzar la mayor capacidad operativa del buque que pueda ser mantenida durante un periodo prolongado (2 o 3 semanas) con una disposición que permita una reacción ante el control de daños.
3. Grado de alistamiento TERCERO: Permite mantener durante un periodo indefinido de tiempo las armas preparadas para reaccionar ante una amenaza inesperada, y disponer del suficiente tiempo para alistar un grado superior. Los equipos y sistemas para el control de daños deben estar preparados para su uso.
4. Grado de alistamiento CUARTO: Permite mantener todos los sistemas operativos para conseguir una vigilancia adecuada. Se establecerá en ambientes no hostiles. El control de daños debe estar alistado para prevenir y reaccionar ante emergencias en la mar.
5. Grado de alistamiento QUINTO: Es el grado más bajo y solo se ordenará en determinados puertos o fondeaderos donde pueda ser necesaria la salida a la mar. La distribución de la dotación permitirá la salida del buque a la mar por razones de emergencia o seguridad con el personal que integra la guardia. La disponibilidad operativa de personal y equipos será fijada por el mando superior.

De forma genérica, se pueden establecer equivalencias entre cada grado de alistamiento y una situación operativa como se muestra a continuación en la Tabla 1:

GRADO	SITUACIÓN OPERATIVA
Primero	I Zafarrancho de Combate
Segundo	II Dos Vigilancias
Tercero	III Tres Vigilancias
Cuarto	IV Guardias de Mar
Quinto	V Guardias de Puerto

Tabla 1-1 : Relación "Grados de Alistamiento" - "Situación Operativa".

Todo buque debe prever en su organización la distribución de su personal para atender a los 5 grados de alistamiento.

Además de lo expuesto, en determinados tipos de buques se consideran situaciones orientadas a adoptar el máximo alistamiento en aquellas misiones para las que han sido específicamente diseñados. Estos casos particulares se sitúan al margen de la situación de zafarrancho de combate por lo que la capacidad del buque para el control de daños se ve disminuida en beneficio de la misión genérica de cada tipo de buque. La referencia [3] considera los siguientes:

1. Zafarrancho de vuelo (Buques con capacidad aérea).
2. Zafarrancho anfibio (Buques anfibios).
3. Zafarrancho de aprovisionamiento (Buques logísticos).
4. Zafarrancho de buceo (Buques de salvamento y rescate).

Por otra parte, para poder afrontar con garantías situaciones concretas, es necesario adaptar la situación operativa, estableciéndose las, anteriormente denominadas, situaciones especiales. Cada buque, según la referencia [3] debe tener organizado y previsto en su plan de combate la adaptación de, al menos, las siguientes situaciones:

1. Guardia de puerto reducida.
2. Preparación del buque para salir a la mar.
3. Babor y estribor de guardia.
4. Aprovisionamiento en la mar.
5. Saludo a la voz.
6. Operaciones de vuelo.
7. Remolque.
8. Defensa antiterrorista.
9. Auxilio exterior.
10. Defensa NBQ.
11. Emergencia en puerto.
12. Emergencia en la mar.
13. Emergencia aeronáutica.
14. Abandono de buque.
15. Hombre al agua.
16. Arriado de embarcaciones menores.
17. Fondeo.
18. Navegación con mal tiempo.
19. Fallo de gobierno.
20. Varada.
21. Colisión.
22. Recogida de naufragos.

Además de las situaciones especiales, en las que la organización operativa se adapta a una situación, actividad o emergencia determinada, deben estar previstas otras posibilidades en las que solo se implica a una parte específica de la dotación para llevar a cabo cometidos muy concretos. Entre ellas, la referencia [3] contempla las siguientes:

1. Navegación en aguas restringidas y/o con baja visibilidad.
2. Visita y registro.
3. Dotación de presa.
4. Trozo de desembarco.
5. Observación de fuego naval.

Independientemente del grado de alistamiento bajo el que se encuentre navegando el buque, y de la situación operativa, particular o especial que esté en vigor, los puestos de trabajo a ocupar por la dotación se encuadrarán por norma general en dos controles: “Control de Operaciones” y “Control de Plataforma”. Cada uno a su vez subdividido en “Subcontroles” [3].

1.3 Propuesta de mejora

Realizado el estudio necesario para entender el plan de combate, queda por responder la pregunta de cuál es la problemática asociada a él. La realidad es que el uso del plan de combate no resulta práctico por una simple razón y esta es que dicho documento se realiza teniendo en cuenta a todos los miembros de la dotación y suponiendo que todos estarán presentes para cada navegación, este caso, a día de hoy, no se da en prácticamente ningún buque de la Armada.

Como resultado de este problema, y previamente a cada navegación, los oficiales responsables de cada control deben realizar un plan de combate específico para cada navegación. Esto supone una innecesaria carga de trabajo y papeleo, sobre todo si se considera que el plan de combate, en buena parte de los buques de la Armada, se sigue realizando de la manera más tradicional posible, con papel y lápiz.

Para solucionar este problema, a lo largo de este “Trabajo de Fin de Grado” (TFG), se propone y desarrolla una aplicación web diseñada específicamente para la creación y actualización de planes de combate de una manera más rápida y sencilla, programada de manera que su uso sea posible por parte de cualquier persona sin necesidad de ningún tipo de formación más que la necesaria para saber lo que es un plan de combate y cómo se debe desarrollar.

1.4 Organización de la memoria

Tras esta introducción, se extienden los capítulos de “Estado del Arte”, “Desarrollo del TFG”, “Validación”, “Conclusiones y Líneas Futuras”, “Bibliografía” y “Anexos”.

En el “Estado del Arte” se describe la actualidad de los planes de combate en los buques de la Armada así como algunos programas usados en las grandes empresas civiles para llevar a cabo la gestión de recursos humanos. También, en el “Estado del Arte”, se presentan una serie de conceptos y herramientas informáticas necesarias para el desarrollo del presente trabajo.

En el capítulo “Desarrollo del TFG” se aborda todo el proceso de realización de la aplicación informática que se propone. En este capítulo se cubre desde la instalación del software necesario para la realización de dicha herramienta informática, hasta el propio código fuente.

En la “Validación” del presente TFG, se trata de exponer los periodos de prueba y evaluación necesarios para demostrar el buen funcionamiento de la aplicación informática que se propone.

En el capítulo de “Conclusiones y Líneas Futuras” se explican las lecciones aprendidas con la realización de este trabajo, la viabilidad de su implantación y las líneas futuras a tomar para la mejora del mismo.

Por último, en los distintos “Anexos”, se adjunta toda aquella información referente al proyecto que no se ha considerado necesario incluir en la memoria para su comprensión, pero que aporta información de mayor profundidad técnica.

2 ESTADO DEL ARTE

En este capítulo de la memoria se expone el proceso actual de desarrollo de planes de combate en la Armada. Adicionalmente, se muestran algunas herramientas utilizadas en el sector civil para la gestión de los departamentos de recursos humanos. Por último se cubrirán una serie de conceptos informáticos necesarios para la correcta comprensión del capítulo de desarrollo.

2.1 Situación en la Armada

Actualmente los planes de combate en la Armada se realizan tomando como referencia una publicación interna de la Armada Española, la “D-CP-03”. Esta publicación recoge los principios que deben orientar la redacción de los MANORGs en general y, por lo tanto, también de los planes de combate. Este documento también recoge los objetivos a alcanzar así como las normas para su desarrollo, la terminología empleada, etc., con el fin de conseguir la eficacia y uniformidad deseada en todos ellos.

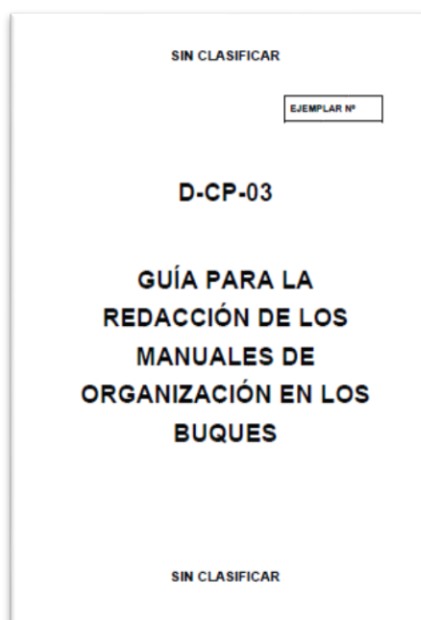


Ilustración 2-1 : Portada de la D-CP-03. Tomada de [3].

En la D-CP-03 se concretan los aspectos de carácter permanente, dejando que publicaciones complementarias recojan los que sean provisionales, sujetos a continua adaptación, o aquellos que para

un control próximo deben estar bajo la responsabilidad de los mandos orgánicos inmediatos al comandante del buque.

En este documento se huye, deliberadamente, de hacer concreciones demasiado específicas debido a la casuística que se presenta, y la conveniencia de dar margen suficiente al responsable de la redacción de un manual para acomodar la norma general a la particular del buque en cuestión.

En este manual se tratan las figuras orgánicas que la Armada ha sentido necesidad de crear en sus unidades con cometidos relacionados con la seguridad, contratación, bienestar y material, entre otros.

Teniendo en cuenta que el buque más frecuente y versátil en la Armada es la fragata, esta publicación se centra en este tipo de unidad para extrapolarse a otras unidades mayores y simplificarse para otras de menor entidad [3].

Es de entender por lo tanto, que la D-CP-03 no es más que una guía para la realización de los MANORGs y los planes de combate. Por lo tanto, por sí sola, esta documentación resulta incompleta para poder llevar a cabo la redacción de los mismos. Para concretar en las necesidades de cada unidad de la Armada, esta cuenta con un gran número de publicaciones en las que se establece el número de personas que debe componer cada control y cada subcontrol en cada tipo de buque de la Armada, así como las funciones que debe desempeñar cada uno.

Sin embargo, este conjunto de publicaciones no establecen realmente un método para la realización de los MANORG y los planes de combate. Únicamente establecen el contenido que estos deben de tener, pero la metodología en sí para su desarrollo no viene establecida en ningún documento. Es por esta razón que la tarea de redactar los planes de combate resulta ineficiente y de muy larga duración, realmente nadie ha establecido nunca una metodología de trabajo. Ante esta ausencia de procedimientos normalizados, los oficiales encargados de la redacción de estos documentos se dedican, simplemente, reunirse, contemplar los listados de personal y puestos, e ir asignando a cada miembro de la dotación un puesto en función de su idoneidad. Posteriormente, estas asignaciones son reflejadas en un conjunto de tablas que constituyen lo que llamamos plan de combate.

Este problema ha sido abordado de diversas maneras, sin demasiado éxito, a lo largo de los años. Se han usado principalmente los programas de “*Microsoft Excel*” y “*Microsoft Access*” para tratar de informatizar el procedimiento de gestión de los planes de combate y así disminuir el tiempo requerido para su actualización, pero estos intentos se han realizado siempre a nivel interno de cada buque, lo que suponía un problema cada vez que había cambios en la plantilla de oficiales del buque, hecho que, en la Armada, ocurre con relativa frecuencia. El problema de estos intentos es que, al ser el plan de combate un documento tan extenso y complejo, un simple documento de “*Microsoft Excel*” o “*Microsoft Access*” no es suficiente para poder reflejar de manera sencilla y clara la totalidad del plan de combate. Por otro lado, este tipo de software no trae consigo ningún tipo de automatización por lo que añadir o quitar a un miembro de la dotación sigue implicando ir punto por punto actualizando cada situación en la que se encontrase la persona en cuestión.

2.2 Situación en el sector civil

Dejando a un lado el sistema de gestión de personal de la Armada y centrándonos en los sistemas empleados en empresas civiles, se puede observar que rara es la compañía que, hoy en día, no usa algún tipo de software informático para agilizar la tarea de la gestión de recursos humanos. Hay demasiados programas dedicados a esta tarea como para contemplarlos todos en este TFG, por lo tanto, tomaremos un reducido grupo de ellos para proporcionar una visión suficientemente amplia de lo que nos podemos encontrar en las empresas civiles.

Los programas seleccionados para estudio se han tomado de entre los más recomendados por referencias como [4] y [5]:

1. Autotask.

2. *GestPeople*.
3. *WorkMeter*.
4. *E-Volution*.
5. *SimpleHRM*.
6. *eMagister RRHH*.

2.2.1 Autotask

“Autotask” es una herramienta de gestión diseñada para distribuidores de informática. Permite gestionar todo un negocio de servicios de TI (Tecnología de la Información), gestión de servicios y facturación. La aplicación permite generar contratos basados en niveles de servicios, gestionar tickets, facturar automáticamente servicios y contratos, realizar un seguimiento comercial de clientes y oportunidades, asignar personas y recursos a proyectos, organizar los departamentos, etc. “Autotask” no es un software puramente dedicado a la gestión de recursos humanos, sino a la organización y gestión de la totalidad de una empresa. Para abarcar la totalidad de las funciones necesarias para la gestión de empresas, “Autotask” se compone de varios módulos. De entre todos los módulos que componen la aplicación, los relativos a la gestión de recursos humanos son los módulos de contratación, programación técnica y gestión de proyectos. El módulo de contratación gestiona los contratos de las plantillas de empleados, el de programación técnica asigna recursos, tanto materiales como humanos, a proyectos y el módulo de gestión de proyectos se encarga de gestionar la colaboración de un equipo y de asignar tareas [6] [7].



Ilustración 2-2 : Interfaz de la aplicación web AutoTask. Tomada de [42].

2.2.2 GestPeople

“GestPeople” es un programa de gestión de recursos humanos con una amplia funcionalidad estructurada por 8 módulos totalmente integrados entre ellos, que cubren la mayoría de aspectos relacionados con el personal, como el organigrama, los perfiles profesionales, competencias, objetivos y evaluación del desempeño. Gracias a esta integración es posible acceder a la información desde diferentes sitios. Este software permite consultar, agregar o editar datos de personal de manera rápida a través de unos formularios [8] [9] [40].

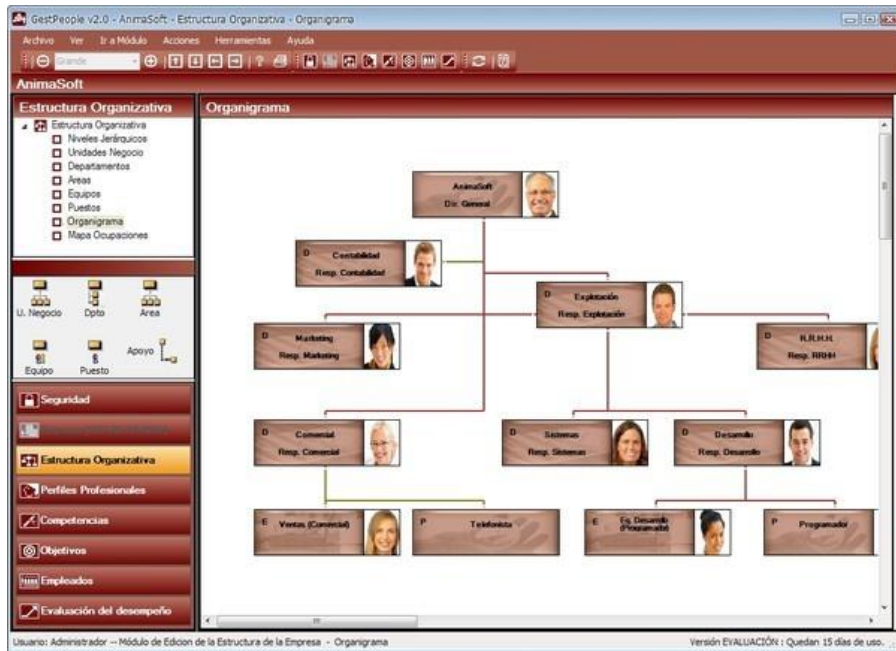


Ilustración 2-3 : Interfaz de la aplicación web *GestPeople*. Tomada de [43].

2.2.3 WorkMeter

“*WorkMeter*” es una aplicación informática que permite conocer, gestionar y medir la productividad de los empleados. Este programa recopila información del uso que los empleados le dan a las aplicaciones y páginas web para determinar si las personas están enfocadas en su trabajo, así como su nivel de esfuerzo. Esta información es visible para el propio empleado, generando así un ambiente de objetividad, transparencia y autocritica. Con este software se pueden definir unos mapas de productividad personalizados, identificando qué aplicaciones son productivas y cuáles no lo son, de manera que se obtienen los tiempos de actividad exactos de cada aplicación en uso. “*WorkMeter*” fomenta la autogestión, informando al empleado de los periodos en los que es productivo y de aquellos en los que no lo es. Genera un entorno de consciencia en el cual el empleado puede ver en qué pierde el tiempo [10] [11].



Ilustración 2-4 : Interfaz de la aplicación web *WorkMeter*. Tomada de [44].

2.2.4 E-Volution

“*E-Volution*” es una aplicación que simplifica y automatiza procesos y actividades de administración de personal que permite a los gestores de recursos humanos dedicarse al desarrollo de talento, planes de carrera, formación y objetivos centrados en los colaboradores. Esto permite además la mejora de los servicios a los colaboradores y la adopción de tecnologías de forma con los modelos de autoservicio, es decir, el uso de herramientas de autoevaluación por parte de los propios empleados. “*E-Volution*” cuenta con 37 módulos de trabajo para la gestión de los recursos humanos. Sus procesos de competencias están basados en las metodologías de las mayores consultoras en América Latina y el mundo como “*Deloitte*”, “*Price Waterhouse*”, “*Hay, Lominger*”, “*Martha Alles*”, etc., permitiendo adaptarse a las metodologías que el cliente configure [12] [13] [14].

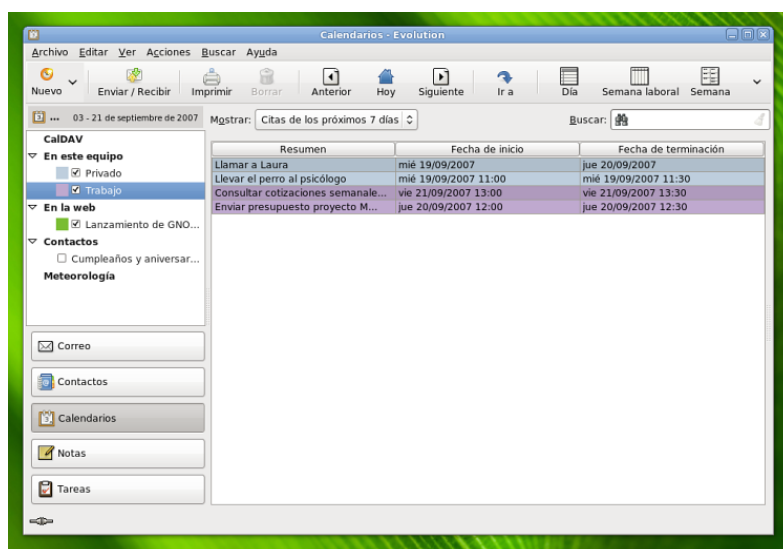


Ilustración 2-5 : Interfaz de la aplicación web *E-Volution*. Tomada de [45].

2.2.5 SimpleHRM

“*SimpleHRM*” es una solución “*Open Source*”, o de código abierto, que permite la gestión de recursos humanos en pequeñas y medianas empresas. La aplicación permite realizar un seguimiento de la información de los empleados, simplificar su gestión, capturar y analizar estadísticas sobre el

rendimiento de los empleados y eliminar o simplificar los trámites de gestión de recursos humanos. Por lo tanto, “SimpleHRM” permite ahorrar tiempo y dinero a través de la centralización de los datos de los empleados [15] [16].

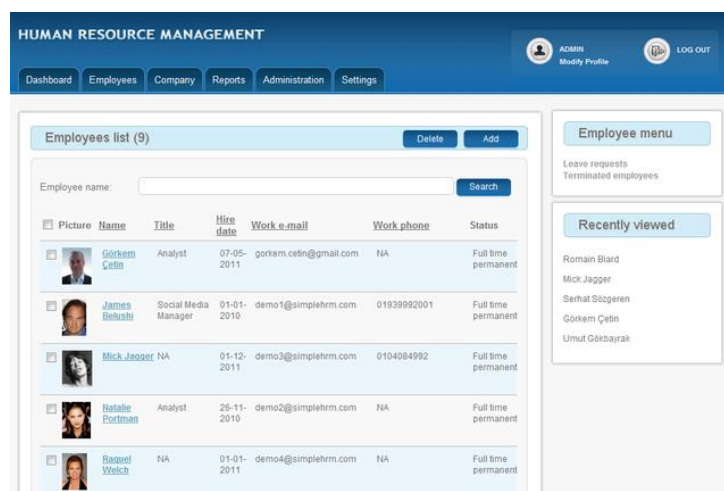


Ilustración 2-6 : Interfaz de la aplicación web SimpleHRM. Tomada de [46].

2.2.6 eMagister RRHH

“eMagister RRHH” se trata de un software que permite gestionar los recursos humanos de la empresa, especialmente en lo que se refiere a la información del personal, sus conocimientos, puestos y competencias. Esta aplicación permite gestionar organigramas con las unidades organizativas y puestos de su estructura, identificando a las personas que ocupan cada puesto y extrayendo los organigramas actualizados. En “eMagister RRHH” el usuario accede a un diccionario de conocimientos y competencias que permite asignar requerimientos a los puestos así como identificar los conocimientos y competencias del personal [17].

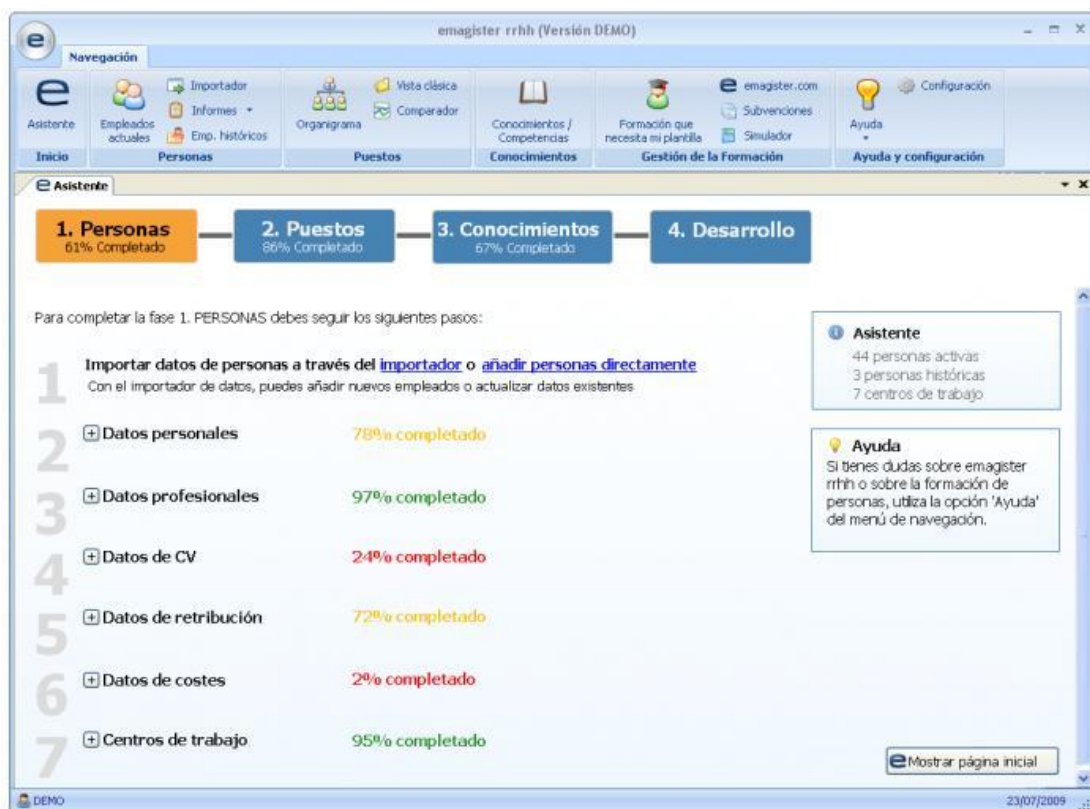


Ilustración 2-7 : Interfaz de la aplicación web eMagister RRHH. Tomada de [47].

2.2.7 Comparativa

Realizando un estudio preliminar de las herramientas de gestión anteriormente mencionadas, se llega a la conclusión de que cualquier empresa, al buscar un software de gestión de recursos humanos, busca que cumpla, al menos, con 5 puntos principales:

1. Gestión de nóminas.
2. Gestión de las prestaciones del trabajo.
3. Gestión de las prestaciones sociales.
4. Gestión de recursos humanos.
5. Indicadores de productividad.

Con respecto a la gestión de las prestaciones sociales, siendo uno de los requisitos principales que buscan las empresas en las herramientas de gestión de recursos humanos, cabe destacar que muchas empresas combinan una herramienta expresamente dedicada a este requisito, con otra dedicada a los demás.

En la Tabla 2 que se muestra a continuación se presenta una comparativa entre las herramientas de gestión anteriormente mencionadas, evaluando su capacidad para responder a los 5 puntos principales que se han establecido anteriormente.

	<i>Autotask</i>	<i>GestPeople</i>	<i>WorkMeter</i>	<i>E-Volution</i>	<i>SimpleHRM</i>	<i>eMagister RRHH</i>
Gestión de nóminas	✓	X	X	✓	✓	✓

Gestión de las prestaciones del trabajo	✓	✓	X	X	✓	X
Gestión de las prestaciones sociales	X	X	X	X	X	X
Gestión de recursos humanos	✓	✓	✓	✓	✓	✓
Indicadores de productividad	✓	✓	✓	✓	✓	✓

Tabla 2-1 : Comparación entre las aplicaciones de gestión de RR.HH.

Como se puede observar en la Tabla 2, en la mayoría de los casos hablamos de programas que buscan gran accesibilidad y automatización con el objetivo de poder llevar a cabo la tarea de gestión del personal desde cualquier parte y reducir el trabajo de organización por parte del usuario. Para ello, la mayoría de estas herramientas se basan en aplicaciones *web*, lo que resulta en un conjunto de bases de datos, permanentemente accesibles y actualizadas.

La metodología empleada en estos programas para la organización del recurso humano no está accesible al público, por lo tanto la metodología empleada para la gestión del personal en la aplicación *web* que se desarrolla a lo largo del presente TFG será la misma utilizada hasta ahora en la Armada. Este TFG no pretende desarrollar nuevos métodos de trabajo para los departamentos de recursos humanos. Más bien desarrollar una aplicación *web* que, sin alterar su método de trabajo, les proporcione una herramienta informática para agilizar la organización del personal.

2.3 Conceptos informáticos

Para la realización de este TFG hay una serie de conceptos previos relativos a la informática que deben entenderse. A continuación trataremos los temas del lenguaje de marcado, lenguaje de programación, bases de datos y servidores *web*.

2.3.1 Lenguaje de marcado, HTML

Un lenguaje de marcado es una forma de codificar un texto a través de etiquetas o marcas que aportan información sobre la estructura del texto o la manera de presentarlo. El lenguaje de marcado más conocido es el HTML (*Hyper Text Markup Language*). El lenguaje HTML está especialmente diseñado para la elaboración de páginas *web*. Toda página *web* toma como referencia para su elaboración el lenguaje HTML, este define una estructura básica y un código para la definición del contenido de la página *web*. HTML está considerado como el lenguaje *web* más importante, habiendo participado de manera crucial en el desarrollo y expansión de la “*World Wide Web*”. Este lenguaje se ha impuesto como estándar para la visualización de páginas *web* y es el que todos los navegadores actuales han adoptado [18] [19]. Para la aplicación que se desarrolla a lo largo del presente TFG se utiliza la versión HTML5. Esta versión del lenguaje se desarrolló con el objetivo de mejorar el lenguaje para atribuirle la capacidad de soportar las nuevas versiones de archivos multimedia, manteniendo a su vez una facilidad de lectura para las personas y una consistente comprensión por parte de los ordenadores. HTML5 se finalizó y publicó el 28 de Octubre del 2014, tomándose en ese mismo momento como lenguaje de referencia por el W3C (*World Wide Web Consortium*) [41].



Ilustración 2-8 : Logotipo de HTML5. Tomado de [48].

2.3.2 Lenguaje de programación, PHP

Un lenguaje de programación es un lenguaje, diseñado para ser comprendido por un ordenador, que sea capaz de realizar procesos a través de dicho ordenador. Estos lenguajes están formados por conjuntos de símbolos y reglas sintácticas y semánticas que definen la estructura del lenguaje y el significado de sus elementos y expresiones. [20]

En la actualidad existen una gran cantidad de lenguajes de programación. Según [21] los más importantes según su uso son:

1. Java
2. JavaScript
3. Python
4. C
5. PHP
6. C++
7. Ruby
8. C#
9. Perl
10. Objective-C

Para la realización de la aplicación *web* de la que trata este TFG se ha optado por el lenguaje de programación PHP. Este lenguaje es en la actualidad uno de los más utilizados por los desarrolladores para crear aplicaciones *web* de todo tipo. Desde simples páginas *web* personales pasando por gestores de contenidos hasta grandes aplicaciones corporativas dan buen ejemplo de la potencia, versatilidad, flexibilidad de uso y adaptación al cambio de un lenguaje de programación asentado plenamente en el desarrollo de software. Desde su creación, PHP fue ganando popularidad hasta convertirse en uno de los lenguajes más utilizados para la creación de páginas *web* dinámicas, entre otras, las principales razones, según la referencia [22], por las que PHP se hizo tan popular son:

1. Fácil de usar: En comparación con otros lenguajes de programación utilizados para el mismo propósito, como JAVA o ASP, PHP es un lenguaje muy fácil de aprender. Por esta razón, no es necesario realizar un estudio en demasiada profundidad para ser capaz de realizar sencillas aplicaciones.
2. Embebido en HTML: Las páginas escritas en PHP son, en muchos casos, páginas escritas en HTML que, además de las etiquetas normales, contienen el programa que queremos ejecutar.
3. Multiplataforma: Se ejecuta en casi cualquiera de las plataformas, sistemas operativos y servidores existentes. Es compatible con los tres servidores líderes en el mercado: Apache, “Microsoft Internet Information Server” y NGINX.

4. Licencia “Open Source”: La licencia de código abierto implica que el código PHP es libre de ser descargado e inspeccionado. Si acompañamos Apache, el servidor más popular, a la instalación de PHP y añadimos alguna base de datos código libre como MySQL, tendremos un sistema de desarrollo completo y gratuito.
5. Multitud de extensiones: PHP se desarrolla para dar la mayor versatilidad y flexibilidad a sus usuarios. Para proporcionar estas características, PHP cuenta con muchas extensiones del lenguaje que permiten utilizar nuevas bases de datos, protocolos, librerías, etc.
6. Velocidad: El motor que incorpora PHP acelera los procesos de ejecución del código.

La mayoría de los proyectos en PHP necesitarán almacenar la información de forma persistente de alguna manera, ya que es un elemento clave en la creación de una *web* dinámica. En la actualidad una aplicación *web* no se limita a mostrar páginas estáticas. Utilizando PHP se pueden crear aplicaciones que almacenen información dada por el usuario, consiguiendo de esta manera la que la aplicación sea dinámica y el servicio al usuario, personalizado.



Ilustración 2-9 : Logotipo de PHP. Tomado de [49].

2.3.3 Sistemas de Gestión de Bases de Datos, MySQL

Esa información se puede almacenar de varias formas dependiendo de la utilidad que le quiera dar. Se pueden utilizar archivos de texto para guardar información relativa a la configuración de la página, que son datos de pequeño tamaño y varían poco. Si manejamos aplicaciones que hacen distinción entre usuarios que guardan información privada en la misma, se almacenaría temporalmente dicha información utilizando sesiones, que son pequeños componentes de información que existen durante el tiempo que el usuario este accediendo a la *web*, en la memoria del servidor.

Cuando se trata de información de mayor tamaño, la mejor manera de almacenarla es utilizando algún tipo de base de datos. Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Las bases de datos más utilizadas son las de tipo relacional, que agrupan la información en conjuntos lógicos relacionados (bases de datos, tablas y campos) y definen un lenguaje de consultas muy potente conocido como “*Structured Query Language*” (SQL) que permite obtener la información de una manera rápida y eficiente [22].

Existen programas denominados “Sistemas Gestores de Bases de Datos” (SGBD) que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada [23]. Entre los principales se encuentran:

1. MySQL
2. Oracle
3. Microsoft SQL Server
4. Microsoft Access
5. Visual FoxPro
6. PostgreSQL

7. Apache Derby
8. Base (Open Office)

De entre todos ellos, y teniendo como objetivo la realización de la aplicación *web* que se propone en el presente TFG, se ha considerado MySQL como el más idóneo, debido principalmente que es una herramienta de código libre, y por lo tanto gratuita, y a que no requiere un alto nivel de conocimientos del lenguaje SQL. Entre el resto de alternativas encontramos unos SGDB que aun teniendo una gran capacidad de procesamiento, son de pago (Oracle) o por el contrario tienen una limitada capacidad de procesamiento (“*Microsoft Access*”) o consumen muchos recursos (PostgreSQL). Para muchos otros, la razón de descarte es que exigen un alto grado de conocimiento del lenguaje SQL (Apache Derby).

MySQL es un SGDB relacional, multihilo y multiusuario con más de seis millones de instalaciones. Está desarrollado como software libre en un esquema de licenciamiento dual bajo licencia GPL (“*General Public License*”). MySQL existe como una aplicación de software instalada y funcionando en una o múltiples máquinas en una red, que sirve información a través de esa red a aplicaciones de software instaladas en otras máquinas que solicitan datos o los envían para ser almacenados. Su conectividad, velocidad y seguridad hacen de MySQL un SGDB altamente apropiado para acceder a bases de datos en Internet [24] [25].



Ilustración 2-10 : Logotipo de MySQL. Tomado de [50].

2.3.4 Servidor web, Apache

Con los conceptos anteriormente tratados, ya se puede programar una aplicación informática, pero para transformar esa aplicación informática en una aplicación *web* necesitamos un servidor *web*.

Un servidor *web* es un programa encargado de publicar aplicaciones en la *web* permitiendo conexiones, tanto unidireccionales como bidireccionales y asíncronas como síncronas, entre el servidor y el cliente, generando una que se presenta al cliente en un navegador *web*. Para la transmisión de todos estos datos hace falta el uso de protocolos, generalmente HTTP. En resumidas cuentas, un servidor *web* es cualquier tipo de software que suministra servicios de acceso a información, a través de la *web*, a otros equipos que así lo han solicitado [26] [27].

Existen una gran variedad de servidores *web* entre los cuales, según la referencia [22], cabe destacar:

1. Apache
2. Microsoft IIS
3. NGINX

De entre todos estos servidores, Apache es el más utilizado, líder con el mayor número de instalaciones a nivel mundial muy por delante de cualquiera de las otras opciones. Apache es un software “*Open Source*” con licencia GPL, por lo tanto, completamente gratuito. Una de sus principales ventajas es que es multiplataforma, por lo que puede trabajar con diferentes sistemas operativos sin ver perjudicado su rendimiento. Entre sus principales características, cabe citar que tiene un soporte de seguridad SSL/TLS, puede acceder a datos a través de SGDB y da soporte a diferentes lenguajes de programación. Su uso principal es el de enviar páginas *web* a través de HTTP/S, ya sean

estáticas o dinámicas. Este servidor se integra con otras aplicaciones creando el conocido paquete XAMP (X, Apache, MySQL y PHP) [28] [29].

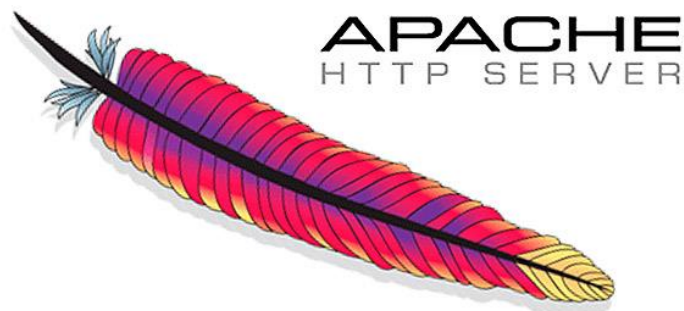


Ilustración 2-11 : Logotipo de Apache. Tomado de [51]

2.3.5 Linux, Apache, MySQL, PHP (LAMP)

En el acrónimo XAMP la “X” es sustituible por la inicial de cualquier sistema operativo. En el caso que se presenta en este TFG, utilizaremos el sistema operativo de Linux Ubuntu, puesto que es el sistema operativo en el cual se tiene mayor experiencia a la hora de programas. En este caso, XAMP pasara a llamarse LAMP (Linux, Apache, MySQL y PHP).



Ilustración 2-12 : Logotipo de LAMP. Tomado de [52].

3 DESARROLLO DEL TFG

En este capítulo, se expondrán todos los pasos, etapas, procesos y acciones que han conformado este TFG, desde la instalación de todo el software necesario hasta el desarrollo del código fuente de la aplicación. Es importante destacar que la aplicación que a continuación se desarrolla podría haberse creado utilizando muchos otros lenguajes, sistemas operativos, bases de datos, etc. El entorno de programación aquí descrito es solo un ejemplo de las posibilidades, aunque, según las fuentes mencionadas en el capítulo anterior, estas son, posiblemente, las más idóneas para la tarea que se pretende.

3.1 Instalación de un entorno de programación web

Como se vio en el “Estado del Arte”, el sistema operativo en el que se tiene mayor experiencia de programación *web* es Linux Ubuntu, por lo tanto, es el que se utilizará para la creación de esta aplicación informática. Ya sea disponiendo de un ordenador con Ubuntu instalado o a través de una máquina virtual, lo importante es disponer de un entorno Linux.

En el caso que se plantea, se dispone de un ordenador con sistema operativo Windows, por lo tanto se procederá a instalar una máquina virtual Ubuntu.



Ilustración 3-1 : Logotipo del sistema operativo Ubuntu. Tomado de [53]

3.1.1 Instalación de una máquina virtual Ubuntu

Para la instalación de la máquina virtual se ha optado por el uso del software de “Oracle” conocido como “Virtual Box” debido a la familiaridad obtenida con este programa a través de los años. Virtual Box puede ser obtenido gratuitamente a través de la página oficial de Oracle [32]. Dentro de la página se seleccionará la opción de descarga que se adapte a nuestro sistema operativo, en el caso del presente TFG: “VirtualBox 5.0.14 for Windows hosts x86/amd64”.

Una vez se disponga de un software de máquina virtual, el siguiente paso es asociarle un sistema operativo Ubuntu.



Ilustración 3-2 : Logotipo del programa VirtualBox. Tomado de [54].

3.1.2 Instalación del sistema LAMP

Una vez se disponga de una máquina Ubuntu, virtual o no, se debe establecer un sistema LAMP. Para la instalación de todos los elementos necesarios a lo largo del presente TFG, siempre se recurrirá al método de instalación a través del terminal de Ubuntu. Para la instalación del sistema LAMP bastará con introducir los siguientes comandos en el orden indicado [33].

- `sudo apt-get update`
- `sudo apt-get install apache2`
- `sudo apt-get install mysql-server`
- `sudo apt-get install php5 libapache2-mod-php5`
- `sudo /etc/init.d/apache2 restart`

3.1.3 Creación de un entorno de programación web

Una vez finalizado el proceso anterior, ya se contará con el conjunto de Apache, MySQL y PHP trabajando conjuntamente. El siguiente paso será la creación de una base de datos que posteriormente se estructurará para que cumpla con las necesidades de la aplicación. Para la creación de dicha base de datos procederemos, en primer lugar, a acceder a MySQL, para ello se introducirá el siguiente comando en el terminal de Ubuntu:

- `mysql -u root -p`

Una vez introducido este comando, se pedirá la contraseña de usuario para poder acceder a MySQL. Habiendo accedido al gestor de bases de datos se procederá a la creación de una base de datos propia, para ello introduciremos el comando:

- `create database PLANCO;`

Dado que operar en la base de datos “root” no se considera lo más adecuado, tras la creación de la base de datos, se procederá a la creación de un usuario de MySQL. A este usuario se le otorgarán permisos sobre la base de datos anteriormente creada y con él se realizará todo el trabajo referente a la creación de una base de datos funcional y estructurada. Para ello se introducirá la siguiente pareja de comandos en MySQL, consiguiendo la creación de un nuevo usuario con el primer comando y asignando permisos sobre la base de datos con el segundo:

- `create user "tfg"@"localhost" identified by "contraseña";`

```
- grant all on PLANCO.* to "tfg"@localhost;
```

Para finalizar la creación del entorno, se deberá comprobar que Apache y PHP funcionan correctamente. Para el primero bastará con abrir en el navegador la dirección “localhost”. En el caso de que Apache esté funcionando correctamente, el navegador nos deberá devolver la página que se muestra en la Ilustración 3-3 a continuación.

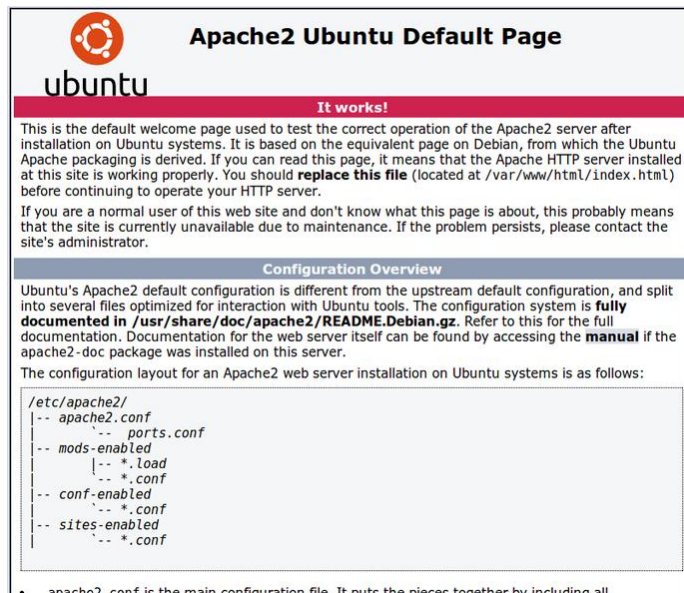


Ilustración 3-3 : Respuesta del navegador con Apache funcionando.

Para comprobar que PHP se ha instalado correctamente y funciona, se creará una página PHP con nombre “*test.php*” con el siguiente contenido:

```
- <?php
    phpinfo();
?>
```

Una vez creado este archivo PHP se le llamará desde el navegador introduciendo “localhost/test.php”. Al hacer esto, y en el caso de que PHP se haya instalado correctamente, el navegador nos devolverá la página que se muestra en la Ilustración 3-4 a continuación.

PHP Version 5.5.9-1ubuntu4.13	
	
System	Linux ns1 3.19.1-x86_64-linode53 #1 SMP Tue Mar 10 15:30:28 EDT 2015 x86_64
Build Date	Sep 29 2015 15:27:05
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/fpm
Loaded Configuration File	/etc/php5/fpm/php.ini
Scan this dir for additional .ini files	/etc/php5/fpm/conf.d
Additional .ini files parsed	/etc/php5/fpm/conf.d/05-opcache.ini, /etc/php5/fpm/conf.d/10-pdo.ini, /etc/php5/fpm/conf.d/20-apcu.ini, /etc/php5/fpm/conf.d/20-curl.ini, /etc/php5/fpm/conf.d/20-gd.ini, /etc/php5/fpm/conf.d/20-imagick.ini, /etc/php5/fpm/conf.d/20-json.ini, /etc/php5/fpm/conf.d/20-mysql.ini, /etc/php5/fpm/conf.d/20-mysqli.ini, /etc/php5/fpm/conf.d/20-pdo_mysql.ini, /etc/php5/fpm/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212

Ilustración 3-4 : Respuesta del navegador con PHP correctamente instalado.

3.2 Creación y estructuración de la base de datos

3.2.1 Estructura de la base de datos

Como se comentó en el capítulo 2, como herramienta de gestión de datos, en nuestra aplicación se usará MySQL, un SGBD relacional. Los sistemas de bases de datos relacionales estructuran la información en tablas. Una de las primeras etapas en el desarrollo de la aplicación consiste en diseñar la estructura de esas tablas y como se relacionan entre sí.

Previamente a comenzar a crear las tablas que compondrán la base de datos debemos tener en cuenta los datos con los que trabajará la aplicación y las relaciones que querrá establecer entre los distintos tipos de datos, ultimando la respuesta a estas preguntas en una estructura de tablas, que posteriormente crearemos, y que compondrán la base de datos que alimentará la aplicación.

Para el caso de este TFG, y teniendo en cuenta la explicación dada acerca de los planes de combate en el capítulo 1, la estructura de la base de datos será tal y como se muestra en la Ilustración 3-5 a continuación.

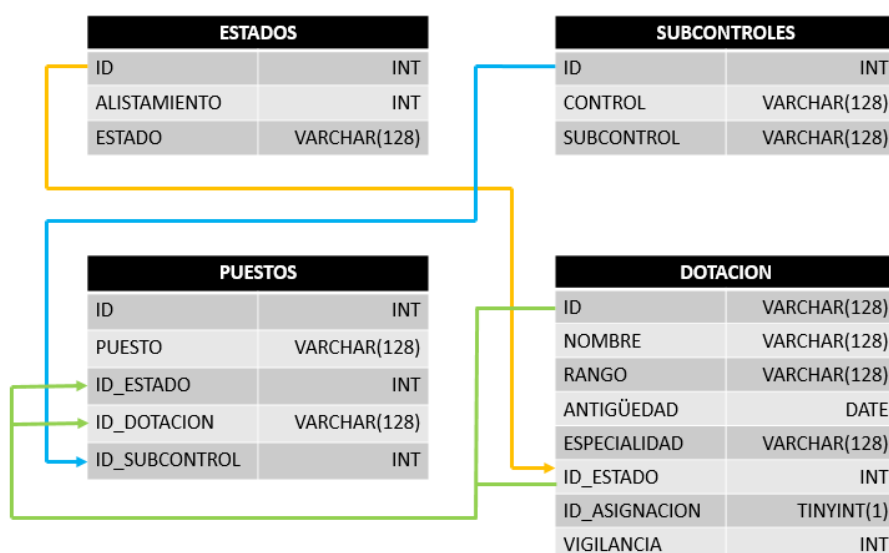


Ilustración 3-5 : Estructura de la base de datos.

Como se puede observar, la aplicación dispone de 4 tablas relacionadas entre sí, en las cuales establecemos los distintos estados (combinación de grado de alistamiento con situación operativa) bajo los que navegará el buque, los puestos a cubrir en cada uno de ellos, los subcontroles de los que dependen estos últimos y las personas que los cubrirán.

3.2.2 Conceptos SQL

Para poder entender correctamente la estructura planteada hay una serie de conceptos que se deben definir referentes al lenguaje SQL: Tipos de datos, claves primarias, claves foráneas y motor de almacenamiento.

- **Tipos de datos:** MySQL ofrece varios tipos de datos para la definición de las columnas de las tablas. Estos tipos de datos se encuentran clasificados en grupos en función de su utilidad, unos sirven para introducir cadenas de caracteres, otros para introducir números, otros introducen valores lógicos (booleanos), otros ofrecen la posibilidad de introducir fechas y horas y existe otro grupo para datos binarios. En la estructura mostrada en la Ilustración 3-5 usamos todos los anteriormente mencionados a excepción de los datos binarios [34].

- Las columnas tipo INT definen un dato numérico de tipo entero positivo o negativo en 32 bits.
- El tipo de dato VARCHAR (n) define una cadena de longitud variable, de “n” caracteres como máximo (n entre 0 y 65.535).
- El tipo DATE permite almacenar fechas comprendidas entre el 01/01/1000 y el 31/12/9999, siguiendo el formato predeterminado YYYY-MM-DD.
- Los datos definidos con TINYINT devuelven un número entero positivo o negativo de 8 bits. Este tipo de dato está considerado como datos de tipo numérico, sin embargo, en el caso planteado TINYINT (1) solo podrá devolver un número entero de 1bit (0 o 1), por lo tanto este dato es considerado como un booleano en el cual el valor “0” es considerado como falso y “1” como verdadero.
- Claves primarias: Una clave primaria (o restricción de clave primaria) garantiza que no habrá nunca dos filas en la tabla con el mismo valor en la columna o columnas que componen la clave. Por otro lado, todas las columnas de la clave primaria son obligatorias. Una clave primaria puede estar constituida por una o varias columnas. Al insertar un nuevo dato en una tabla, si el valor que pretendamos introducir en la columna que contenga la clave primaria ya se encontraba previamente en la tabla, se producirá un error. De la misma manera, también se producirá un error si al introducir un nuevo dato, no especificamos un valor para la columna con la clave primaria. Por otra parte, las claves primarias se utilizan para identificar registros en la base de datos y sirven como referencia del registro para las claves foráneas. En la estructura definida en la Ilustración 3-5, todas las tablas tienen definida la columna “ID” como clave primaria a excepción de la tabla “DOTACION” que tiene asociada una clave primaria compartida entre las columnas “ID” y “ID_ESTADO” [34].
- Claves foráneas: Una restricción de clave foránea, o clave externa, permite garantizar que cada valor almacenado en una tabla “secundaria” existe realmente en una clave primaria previamente definida en otra tabla “primaria”. Estas restricciones también permiten controlar lo que pasa cuando una de las filas de la tabla primaria se elimina o la clave primaria de una fila se modifica. En la Ilustración 3-5, se representan a través de flechas, las filas de las tablas que tienen una clave foránea y las filas con clave primaria de otras tablas a las que hacen referencia [34].
- Motor de almacenamiento: El motor determina la manera en la que los datos son almacenados en el disco duro y que operaciones se pueden realizar sobre dichos datos. Para esta aplicación se utilizara el motor “InnoDB”. El motor InnoDB permite gestionar las claves externas así como transacciones y bloqueos a nivel de fila. El motor de almacenamiento que se utilice se especifica en el propio código SQL a la hora de crear las tablas [34].

Una vez queda definida la estructura de la base de datos, comenzará la creación de las tablas que la conforman. El conjunto de comandos que crean la base de datos de la aplicación *web* definida en este TFG se han reflejado en el Anexo I de la presente memoria.

3.3 Introducción a PHP

Previamente al comienzo de la programación del código que compondrá la propia aplicación, hay una serie de conceptos que es necesario entender.

3.3.1 Patrón MVC

La mayoría de las páginas *web* que se desarrollan en la actualidad con PHP y MySQL utilizan el patrón de diseño denominado como MVC (Modelo – Vista – Controlador). Un sistema MVC intenta separar en varias capas una aplicación de manera que los cambios en una de ellas no impliquen cambios en el resto de capas. Es un patrón de arquitectura de las aplicaciones *web* que separa la lógica de negocio de la interfaz de usuario, consiguiendo de esta manera facilitar la evolución por separado de ambos aspectos e incrementando su reutilización y flexibilidad [34].

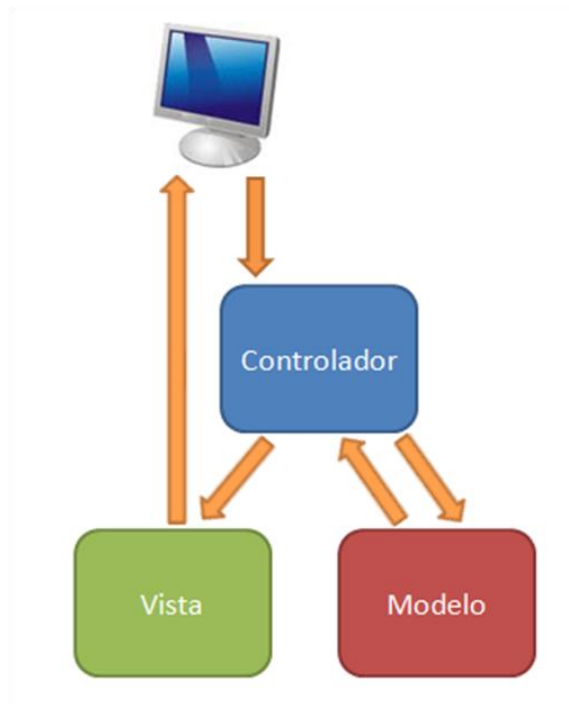


Ilustración 3-6 : Patrón MVC. Tomado de [55].

El funcionamiento que se muestra en la Ilustración 3-6 es el siguiente:

1. El usuario solicita la URL al Controlador.
2. El Controlador evalúa la petición y solicita al modelo los datos.
3. El modelo conecta con la base de datos y extrae la información.
4. El Controlador obtiene los datos en bruto y los estructura.
5. Se compone la Vista con los datos estructurados en el Controlador.
6. El usuario obtiene el resultado de su solicitud a través de la Vista.

Toda aplicación MVC debe tener un punto de entrada. El archivo “index.php” será el encargado de cumplir con este objetivo. Este archivo es el primero que le proporciona al usuario una interfaz a través de la cual comunicarse con el controlador.

El controlador es la parte que controla el flujo de navegación de la aplicación. Es el encargado de ver qué acciones hay que ejecutar, transmitirlos al modelo y devolver los resultados a la vista correspondiente. Las aplicaciones basadas en MVC suelen tener un controlador principal y un controlador por cada módulo existente. En el caso de la aplicación que tratamos en este TFG usaremos solo un controlador principal debido a que el código no es lo suficientemente complejo como para precisar de más controladores en los que dividirlo. En el controlador se evalúan parámetros enviados en formularios para saber qué es lo que pide el usuario. Una vez el controlador ha juzgado dichos valores, llama a la función del modelo encargada de realizar la tarea o, en el caso de que la tarea no requiera conectar con la base de datos, redirige al usuario a otra vista [34].

El modelo es el encargado de obtener los datos de MySQL y ejecutar la lógica de negocio. Es una capa entre la base de datos y el controlador. Todo lo que tiene que ver con la gestión de datos, debe pasar por aquí. En el modelo se definen todas las funciones que, activadas por el controlador, conectan con la base de datos para introducir, actualizar, eliminar o recuperar la información que el usuario solicite a través de las variables. Una vez realizada su función, el modelo informa al controlador del éxito o error de las tareas que le fueron encomendadas [34].

La vista es la parte gráfica de la aplicación *web*. Los usuarios interactúan directamente con la vista leyendo la información, haciendo clic en los botones y rellenando los formularios. A través de las distintas vistas que componen la aplicación, el usuario expresa, a través de enlaces, botones, formularios, etc., la información que quiere introducir, eliminar, actualizar o seleccionar de la base de datos. Las peticiones del usuario serán enviadas al controlador para que este juzgue la acción a tomar y se lo comunique al modelo [34].

3.3.2 Las bases del lenguaje PHP

A medida que se recorra el código, se verá en todos los archivos PHP el uso común de una serie de elementos principales: Las variables, los arrays, las estructuras de control, las funciones, el elemento `$_REQUEST` y las sesiones (el elemento `$_SESSION`).

Una variable es una zona de memoria identificada por un nombre que contiene un valor legible o modificable en la aplicación. En PHP, las variables están identificadas por el prefijo “\$” seguido de un nombre. El nombre de las variables tiene en cuenta las mayúsculas y minúsculas, por lo tanto `$Nombre` y `$nombre` serán vistas por PHP como dos variables distintas. Las variables se definirán automáticamente en su primera utilización. El alcance de una variable será la página en la que esté definida, por lo tanto, una variable se podrá definir en una primera sección del código PHP y reutilizarse en otra parte del mismo código. Cuando una página se termina, se eliminarán todas las variables que en ella se hayan iniciado [34]. A continuación se puede observar un ejemplo de definición de una variable.

```
- $variable = 10;
```

Un array es un variable especial con capacidad para almacenar más de un valor a la vez. En PHP, un array es un mapa ordenado, es decir, un tipo de datos que asocia valores con claves. Los valores asociados a un array pueden ser otros arrays, creando de esta manera arrays multidimensionales [34]. En el ejemplo a continuación se puede apreciar el método para definir arrays.

```
- Array simple: array( clave1=valor1, clave2=valor2, clave3=valor3);  
- Array multidimensional: $variable = array( array(1,2,3), array(4,5,6),  
array(7,8,9));
```

Las estructuras de control permiten controlar el flujo de la aplicación tomando decisiones, realizando acciones repetitivas, etc., dependiendo de las condiciones que se establezcan en el código. Las estructuras de control se dividen en dos grupos según si sirven para tomar decisiones en función del valor de un parámetro (Sentencias condicionales) o si sirven para repetir una acción mientras se mantenga el valor de un parámetro (Sentencias repetitivas o bucles). Las sentencias condicionales son el núcleo para la toma de decisiones en el código PHP. Estas sentencias evalúan el valor (verdadero o falso) de una expresión y ejecutan una parte del código para cada valor. En la aplicación desarrollada en este TFG se utilizan dos tipos de sentencias condicionales: Sentencia “*if*” y sentencia “*if ... else*”. La sentencia “*if*” comprobará si se cumple una condición determinada, en caso positivo se ejecutará un código determinado, y en caso negativo se saltará ese código. La sentencia “*if ... else*” es muy parecida a la sentencia “*if*” con la diferencia de que si no se cumple la condición establecida en el “*if*”, establecerá un camino alternativo dentro del código definido a través de la sentencia “*Else*”. Por otro lado, las sentencias de bucles son la principal manera de indicarle al intérprete de PHP que ejecute una acción, de manera repetitiva, mientras se cumpla una condición determinada. En esta aplicación se

usan tres tipos de sentencias de bucles: Bucles “*for*”, bucles “*foreach*” y bucles “*while*”. El bucle “*while*” establecerá una condición que, mientras se cumpla, ejecutará una y otra vez el mismo código. En cuanto deje de cumplirse se saltará el código. El bucle “*for*” funciona igual que el “*while*” con la diferencia de que en el “*for*” se establecerá previamente cuántas veces se va a repetir el bucle, es decir, es necesario conocer de antemano el número de veces que es necesario repetir el código. El bucle “*foreach*” sirve para recorrer de manera repetitiva los valores de un array hasta que se quede sin valores. A continuación se pueden observar los métodos de uso de las estructuras de control mencionadas [34] [35] [36]:

- *if: if (condición) {código php representando una acción;}*
- *if ... else: if (condición) {código php representando una acción;} else { código php representando una acción alternativa;}*
- *while: while (condición) {código php representando una acción en bucle;}*
- *for: for (\$contador=0; \$contador<condición; \$contador++) {código php representando una acción en bucle;}*
- *foreach: foreach (\$array as \$valor) {código php representando una acción en bucle;}*

Una función es un conjunto de instrucciones identificadas por un nombre, cuya ejecución devuelve un valor y cuya invocación puede utilizarse como elemento de una expresión. Son partes del código que pueden ser llamadas para realizar una tarea específica. Se les pueden pasar, a la hora de llamarlas, una serie de parámetros en forma de variables para que utilicen sus valores para realizar las operaciones correspondientes. Las funciones PHP se definen con la palabra clave “*function*” seguida de un nombre y un par de paréntesis. El resultado de una función puede ser de cualquier tipo. La instrucción “*return*” detiene la ejecución de la función y devuelve el resultado al código que la invocó. Si no existe ninguna instrucción “*return*” dentro de la función, esta devolverá el valor “*NULL*” [34]. La función que se muestra a continuación es un ejemplo sacado de la propia aplicación definida en este TFG.

- ```
function desembarcar_todos() {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $desembarcar_todos = array();
 $conn = mysqli_connect ($servername, $username, $password,
 $dbname);
 if (!$conn) {
 die ("Conexión fallida:".mysqli_connect_error ());
 }
 mysqli_set_charset ($conn, "utf8");
 $sql = "DELETE FROM dotacion";
 $result = mysqli_query ($conn, $sql);
 mysqli_close ($conn);
}
```

El protocolo HTTP es un protocolo “sin estado”: nada permite identificar que es el mismo usuario quien estaba anteriormente en la página A y que ahora accede a la página B. En cuanto a PHP, se vio que el alcance de una variable es la página en la que está definida y esta existe solo durante la ejecución de la misma. Una aplicación *web* dinámica que va más allá de la simple visualización de una serie de páginas, necesita, desde el punto de vista de la lógica aplicativa, ser capaz de identificar a un

usuario y conservar datos relativos a dicho usuario de una página a otra. Una manera de hacer que los datos de una página sean accesibles en otra, es a través de las sesiones HTTP. Las sesiones se borran al abandonar la aplicación o tras un periodo largo de inactividad (30min. por defecto) [34] [37]. Las sesiones PHP se inician llamando a la función “`session_start()`”, debiendo definir previamente que sesión se está iniciando a través de la función “`session_name(“nombre”)`”. Esta función comprueba si hay alguna sesión ya creada con el nombre indicado, y si no la hay, crea una nueva. El código que se muestra a continuación es el utilizado en la aplicación desarrollada en este TFG para los inicios de sesión.

```
- session_name("TFG");
 session_start();
```

Para guardar datos en sesión se usa el array asociativo “`$_SESSION[]`”. Todos los datos que se guarden en este array serán accesibles desde cualquier página de la aplicación que sea accedida por el usuario al que está asociada la sesión, durante el tiempo en el que la misma siga activa. A continuación se muestra un ejemplo del método para guardar datos en sesión.

```
- $_SESSION["array"] = $variable;
```

Otro método para pasar datos de una página a otra es a través del elemento “`$_REQUEST`”. Hay dos maneras principales de usar este elemento: asignándole un valor a una variable en la dirección URL o bien a través de formularios. Una vez asignado un valor se puede llamar a este valor en otra página utilizando el array “`$_REQUEST`”. El código que se muestran a continuación se corresponde con el método para traspasar información entre páginas a través de formularios.

```
- Página A:
 <form method=post>
 <input type=text name='nombre'>
 <input type=submit>
 </form>

- Página B:
 $nombre = $_REQUEST["nombre"];
```

### 3.4 Introducción a HTML

Además del código PHP, cada página necesitará usar código HTML para presentar información al usuario a través de las vistas. HTML utiliza un sistema de etiquetas para crear las páginas. Existen múltiples tipos de etiquetas que definen el tipo de campo que se pretende presentar (texto, enlaces, tablas, formularios, etc.). A continuación se explicarán las principales etiquetas usadas en la aplicación.

#### 3.4.1 Texto

En HTML se puede representar texto simplemente escribiendo el texto en sí o bien se puede especificar qué tipo de texto es (encabezados, párrafos, líneas, etc.). Los encabezados se representan con las etiquetas “`<h#>texto</h#>`” en las cuales el campo “`#`” se sustituye por un número del 1 al 6 para indicar el tamaño de fuente. Los párrafos se indican con las etiquetas “`<p>texto</p>`” entre las cuales se introduce el texto que se quiere que forme el párrafo [38]. El código HTML que se muestra a continuación genera el resultado mostrado en la Ilustración 3-7.

```
- <h1> Esto es un encabezado1</h1>
 <p>Esto es un párrafo</p><p>Esto es otro párrafo</p>
```



Ilustración 3-7 : Texto

### 3.4.2 Enlaces

Los enlaces se indican utilizando la etiqueta “<a href=’>texto</a>” en la cual se introducirá el enlace dentro de las comillas precedidas por “href=” convirtiendo el texto en un enlace que llevará al usuario a la URL indicada cuando accione el enlace [38]. En la Ilustración 3-8 se ve el resultado del código HTML que se muestra a continuación.

– <a href=’ http://www.google.es/’> Enlace a Google </a>

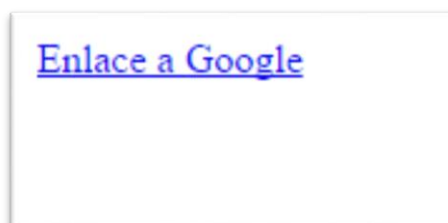


Ilustración 3-8 : Enlace

### 3.4.3 Tablas

Para crear tablas en HTML se utilizará la etiqueta “<table></table>” dentro de la cual se añadirá la etiqueta “<tr></tr>” para indicar que se añade un fila en la tabla y dentro de esta última etiqueta se añadirá “<td></td>” para indicar que se añade una columna dentro de la fila [38]. El código que se muestra a continuación define la tabla de la Ilustración 3-9.

– <table>  
    <tr><td>Columna 1</td><td>Columna 2</td></tr>  
    <tr><td>Fila 1</td><td>Fila 1</td></tr>  
    <tr><td>Fila 2</td><td>Fila 2</td></tr>  
    <tr><td>Fila 3</td><td>Fila 3</td></tr>  
</table>

| Columna 1 | Columna 2 |
|-----------|-----------|
| Fila 1    | Fila 1    |
| Fila 2    | Fila 2    |
| Fila 3    | Fila 3    |

Ilustración 3-9 : Tabla

### 3.4.4 Formularios

Los formularios se indican con la etiqueta “<form></form>” dentro de la cual se introducirán todos los campos del formulario con la etiqueta “<input type=‘>”. Dentro de las comillas simples precedidas por “type=” se indicará el tipo de campo que se está definiendo, pudiendo ser: *button* (botón), *checkbox* (casilla de marcado), *date* (fecha), *text* (texto), *hidden* (campo oculto) o *submit* (botón de envío del contenido del formulario) entre otros [38]. En la Ilustración 3-10 se ve el resultado del código HTML que se muestra a continuación.

```

- <div>
 <form method='post' action='control.php'>
 Nombre completo :

 <input type='text' name='nombre' required>

 TIM :

 <input type='text' name='id' required>

 Rango :

 <input type='text' name='rango' required>

 Antigüedad :

 <input type='date' name='antigüedad' placeholder='AAAA-MM-DD'>

 Especialidad :

 <input type='text' name='especialidad' required>

 <input type='hidden' name='accion' value='nueva_persona'>
 <input type='submit' name='enviar' value='Introducir Datos'>
 </form>
</div>

```

Ilustración 3-10 : Formulario

### 3.5 CSS y la etiqueta <style>

La última etapa de la creación de una aplicación *web* es la estilización de la interfaz de usuario. Como se ha visto en apartados anteriores, el simple uso del código HTML y PHP ya crea una herramienta funcional con una interfaz muy simple, pero esta interfaz no resulta vistosa para el usuario. Es por esta razón que HTML pone a disposición de los programadores la etiqueta “<style>”

que, a través de comandos “CSS” da la capacidad de personalizar la aplicación a conveniencia, consiguiendo una interfaz de usuario más amigable y fácil de usar. Las hojas de estilo en cascada o CSS (del inglés “*Cascading Style Sheet*”) son una serie de comando descritos dentro de la etiqueta “<style>”, introducida en la cabecera de un archivo HTML, o bien en un archivo CSS aparte referenciado a través de un enlace incluido dentro de la etiqueta “<head>”. Estos comandos permiten añadir atributos a cualquier tipo de etiquetas para personalizar la fuente del texto, su tamaño y color, el color de fondo de la página y el estilo de las tablas y formularios, entre otros. En la Ilustración 3-11 vemos un ejemplo del resultado de añadir información de estilo a un formulario [39].

```
- <style>
 body {background-color: #0011aa;}
 h1 {color:white; text-align:center}
 h2 {color:white; text-align:center}
 h3 {color:white; text-align:center}
 a: link {color:blue;}
 a: visited {color:blue;}
 a: hover {color:green;}
 a: active {color:green;}
 div {background-color: #f0ffff; padding:20px; width:30%; border-
 radius:5px; margin: auto;}
</style>
```

The image shows a web form with a light blue background and a dark blue border. The form contains several input fields and a button. The labels are: 'Nombre completo:', 'TIM:', 'Rango:', 'Antigüedad:', and 'Especialidad:'. The 'Antigüedad:' field has a placeholder text 'AAAA-MM-DD'. At the bottom of the form is a button labeled 'Introducir Datos'.

Ilustración 3-11 : Formulario con CSS

### 3.6 Código de la Aplicación

Ya definidas todas las partes que compondrían una aplicación *web* funcional, bastará con juntar todo lo anteriormente dicho para poder crear cualquier tipo de aplicación *web* sencilla. Para el caso de la aplicación que en este TFG se plantea, se adjuntan en los anexos del II al X todos los archivos PHP que componen la misma, incluyendo en ellos los elementos HTML y CSS que forman el interfaz de usuario. A continuación se detalla la estructura de la aplicación creada así como los problemas más destacables que tuvieron que ser abordados y cómo se resolvieron.

#### 3.6.1 Estructura de la Aplicación

La aplicación *web* para la gestión de planes de combate o “Plan de Combate Web” (PCW), se compone de 9 archivos PHP formando un patrón MVC. La aplicación comienza desde el archivo “index.php” desde donde se ofrece una serie de opciones que dirigirán al usuario a los archivos “vista1.php”, “vista2.php”, “vista3.php” o “vista4.php”. Desde el archivo “vista1.php,” el usuario podrá ser redirigido al archivo “vista5.php”. A su vez, desde el archivo “vista4.php” el usuario será redirigido hacia el archivo “generarpdf.php”. Por último existen otros dos archivos con nombre “control.php” y “modelo.php”, que se encargan de la lógica de negocio y de las peticiones hacia la

base de datos, que no ofrecen ningún interfaz al usuario pero que se encuentran fuertemente ligados a todos los demás archivos PHP mencionados.

### 3.6.2 *Index.php*

El archivo “index.php” es un simple menú con el nombre de la aplicación que le ofrece al usuario la posibilidad de elegir entre las principales acciones que puede llevar a cabo con la aplicación: Ver plan de combate, embarcar nuevo personal, desembarcar personal y generar archivos PDF. Cada una de estas opciones redirigen al usuario, respectivamente, a los archivos “vista1.php”, “vista2.php”, “vista3.php” y “vista4.php”.

En cuanto al código HTML, el archivo “index.php” es bastante sencillo. Dispone de un elemento “<table></table>” integrado por enlaces a cada uno de los archivos anteriormente mencionados y por un elemento de encabezado “<h1></h1>”.

En sí, este archivo no presenta ningún tipo de problemática al ser un archivo bastante sencillo. Ha de tenerse en cuenta que la sencillez es precisamente una de las principales características que se pretenden conseguir en esta aplicación, y más aún en la primera página que se le presenta al usuario. En la Ilustración 3-12 se puede ver la interfaz obtenida a través de este archivo. El código que conforma este archivo puede verse en el Anexo II de la presente memoria.

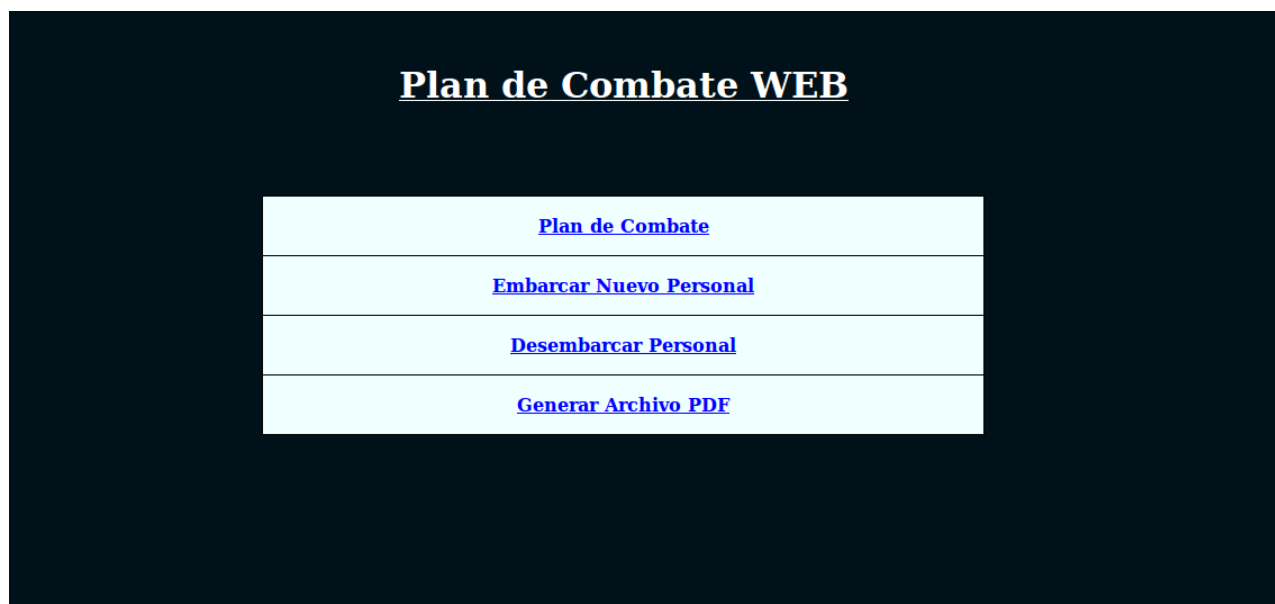


Ilustración 3-12 : index.php

### 3.6.3 *Vista1.php*

El archivo “vista1.php” es el archivo principal de gestión del plan de combate. En este archivo, se le pide al usuario que seleccione uno de los 3 grados de alistamiento bajo los cuales navega un buque de la armada. Una vez seleccionado, y tras haber sido redireccionado a los archivos “control.php” y “modelo.php”, se le mostrarán al usuario las situaciones operativas contenidas dentro del grado de alistamiento seleccionado, así como la posibilidad de añadir una nueva situación o borrar una de las ya existentes. El usuario podrá seleccionar una de las situaciones operativas para ver los puestos que existen dentro de la misma, el subcontrol del que depende cada puesto y si tienen, o no, una persona asignada, dando la posibilidad de vaciar el puesto en el caso de que ya esté asignado o de asignárselo a alguien en caso de que este vacante. Si el puesto está asignado, el usuario puede acceder a los datos de la persona que lo ocupa accionando el identificador de la persona, que actúa como enlace. Si el puesto no está asignado, cuando el usuario seleccione la opción de asignar puesto, se le mostrará una tabla con todos los miembros de la dotación que estén disponibles para ocupar el puesto. La regla para considerar si los miembros de la dotación están o no disponibles, es que no tengan ya asignado un

puesto en la situación operativa a la que pertenezca el puesto que pretendemos asignar. Además, si nos encontramos en el caso específico de una situación operativa bajo el grado de alistamiento “3 Vigilancia”, antes de mostrarle los puestos, se le presenta al usuario la opción de escoger la vigilancia cuyos puestos quiere ver. Para este caso, la lista de miembros de la dotación disponibles adquiere una segunda restricción, siendo esta que los disponibles tienen que pertenecer a la vigilancia que el usuario haya seleccionado. Por otra parte, el usuario podrá accionar un enlace, indicado por el nombre “Gestionar Puestos”, para ser redirigido al archivo “vista5.php”. Por último, durante todo el proceso descrito, existen dos enlaces que permiten al usuario vaciar todos los puestos asignados y volver a la página inicial “index.php”.

El código HTML asociado a este archivo incluye, en mayor número, los mismos elementos que el archivo “index.php” a excepción del elemento “<form></form>”, que se encuentran en este archivo pero no en el anterior. Sin embargo, a diferencia de “index.php”, el código PHP de este archivo es mucho más complejo. En este archivo se han tenido que incluir estructuras de control, variables, arrays, peticiones de datos de sesión y eliminación de datos de sesión.

El principal problema que surgió en la creación de este archivo fue el de conseguir que la aplicación distinguiese el caso en el que el usuario seleccionaba el grado de alistamiento “3 Vigilancia” de cualquiera de los otros. Se solucionó añadiendo una columna llamada “alistamiento” en la tabla “estados” de la base de datos de manera que todas las situaciones operativas dentro de cada grado de alistamiento comparten el mismo valor en la columna del mismo nombre. De esta manera, añadiendo una variable con el valor de la columna “alistamiento” de la base de datos, la aplicación al leer dicha variable, reconocía en qué grado de alistamiento se encontraba. En la Ilustración 3-13 se muestra el interfaz principal proporcionado por el archivo “vista1.php”. El código que conforma este archivo puede verse en el Anexo III de la presente memoria

Volver a Inicio

## Seleccionar Grado de Alistamiento

Vaciar todos los puestos

Zafarrancho de Combate	Babor y Estribor de Guardia	3 Vigilancia
------------------------	-----------------------------	--------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:  Añadir

Borrar situacion con nombre:  Borrar

Zafarrancho de Combate	Operaciones de Vuelo	Hombre al Agua	Escuadras de Socorro	FPL-1
------------------------	----------------------	----------------	----------------------	-------

Ilustración 3-13 : vista1.php

### 3.6.4 Vista2.php

Al acceder, desde “vista1.php”, al archivo “vista2.php”, el usuario se encontrará con una tabla relacionando todos los puestos que estaba viendo en el archivo “vista1.php” antes de accionar el hipervínculo “Gestionar Puestos”. En este archivo el usuario podrá crear y eliminar puestos, dentro de la situación operativa en la que estuviese cuando accionó el hipervínculo. Se le mostrará al usuario una tabla con cada puesto dentro de la situación operativa, el subcontrol al que pertenece y un hipervínculo con el nombre “Borrar Puesto” para borrar dicho puesto. Por otra parte, encima de la tabla se encuentra otro hipervínculo con el nombre “Añadir Puesto”, que al accionarse le mostrará al usuario un formulario en el cual introducirá el nombre del puesto que desea introducir. A su vez, el usuario verá

en todo momento otros dos hipervínculos en la parte superior de la página que le permitirán volver al menú principal, o bien, volver al archivo “vista1.php”.

Los códigos HTML y PHP de esta página, aun siendo más simples que los de la página anterior, siguen manteniendo un grado de complejidad considerable. Con respecto al HTML, esta página utiliza elementos de tablas, formularios, enlaces y encabezados. Por su parte, el código PHP requiere la definición de variables, arrays, estructuras de control y la llamada a datos almacenados en sesión.

A continuación, en la Ilustración 3-14, se puede observar el resultado obtenido por el usuario al llamar al archivo “vista2.php”. El código completo de este archivo se puede consultar en el Anexo IV de la presente memoria.

Añadir situación con nombre:

Añadir

Borrar situación con nombre:

Borrar

Zafarrancho de Combate	Operaciones de Vuelo	Hombre al Agua	Escuadras de Socorro	FPL-1
Puesto	Subcontrol	Asignado a		
Supervisor Propulsion	11	61644966957Y	Vaciar Puesto	
Supervisor Electricidad	11		Asignar Puesto	
Operador DCC	11		Asignar Puesto	
Suboficial SI	11		Asignar Puesto	
Operador CAPES	11		Asignar Puesto	
Jefe de Escena PROA	11		Asignar Puesto	
Telefonista PROA	11		Asignar Puesto	
Electricista PROA	11		Asignar Puesto	
Controlador EERR PROA	11		Asignar Puesto	
Investigador PROA	11		Asignar Puesto	
Ayudante Investigador PROA	11		Asignar Puesto	
Mensajero PROA	11		Asignar Puesto	
Jefe Grupo CI PROA	11		Asignar Puesto	
Manguera PROA	11		Asignar Puesto	
Manquera PROA	11		Asignar Puesto	

Ilustración 3-14 : vista2.php

### 3.6.5 Vista3.php

En el archivo “vista3.php” el usuario tiene la opción de, a través de un formulario, añadir una nueva persona a la dotación. El formulario que se muestra en este archivo se compone de 6 campos: nombre, TIM, rango, antigüedad, especialidad y vigilancia. En todos los campos, a excepción de los campos antigüedad y vigilancia, el usuario tiene libertad para introducir cualquier cadena de caracteres. Las restricciones planteadas en este formulario son las siguientes: Los campos nombre, TIM y rango deben estar rellenos para que el usuario pueda accionar el botón “Introducir Datos”, el campo antigüedad debe rellenarse con una fecha siguiendo el formato AAAA-MM-DD y el campo vigilancia debe rellenarse con un número entero. Una vez el usuario cumplimente el formulario correctamente y accione el botón “Introducir Datos”, los datos se enviarán a la base de datos y el usuario tendrá la opción, a través de dos enlaces, de volver al menú principal o recargar el formulario y continuar introduciendo personas. El enlace que permite al usuario volver al menú principal está presente en todo momento.

El código HTML de este archivo se compone por un elemento enlace y un elemento formulario, por lo tanto, se puede decir que es bastante sencillo. El código PHP tampoco es de gran complejidad, los únicos elementos PHP utilizados en este archivo son variables y estructuras de control.

La problemática asociada a este archivo vino a la hora de establecer las restricciones de los campos del formulario antigüedad y vigilancia. Para el campo antigüedad, se aplicó la restricción utilizando un atributo HTML “*input type='date'*” y añadiendo un texto predefinido en el fondo del campo a través del atributo “*placeholder='AAAA-MM-DD'*”. Para el campo vigilancia, se añadió el atributo “*input type='number'*” de manera que el usuario no introduce un texto en este campo, sino que escoge un número de un contador. En la Ilustración 3-15 podemos ver la página que se le mostraría al usuario al acceder al archivo “vista3.php”. En el Anexo V de la presente memoria, puede verse el código de este archivo.

[Volver a Inicio](#)

### Nueva Persona

Nombre completo:

TIM:

Rango:

Antigüedad:

Especialidad:

Vigilancia:

Ilustración 3-15 : vista3.php

### 3.6.6 Vista4.php

Al acceder al archivo “vista4.php”, el usuario tendrá 3 posibilidades: desembarcar a un miembro de la dotación, desembarcarlos a todos o volver al menú principal. La vista 3 le muestra al usuario una tabla con todos los miembros de la dotación acompañados de un enlace con el nombre “Desembarcar”. Si el usuario acciona ese enlace, la persona asociada será eliminada de la base de datos y todos los puestos que ocupase quedarán automáticamente vacantes. Lo mismo ocurre, ampliado a toda la dotación, si el usuario acciona el enlace “Desembarcar Todos”.

Tanto el código HTML como el PHP de este archivo son bastante simples. En cuanto a HTML, este archivo utiliza elementos de tabla, hipervínculo y encabezado. El código PHP de este archivo utiliza variables, arrays, estructuras de control y llamadas a datos de sesión.

El principal problema asociado a este archivo fue el de conseguir que el listado de puestos se actualizase automáticamente a medida que se efectuaban cambios sobre la dotación. Este problema se abordó y solucionó utilizando el dato del nombre de la persona para obtener su identificador de la tabla de dotación para posteriormente entrar en la tabla de puestos y borrar los identificadores de dotación de todos los puestos en los que coincidiese con el de la persona desembarcada. En la Ilustración 3-16 se puede ver la interfaz principal obtenida al ejecutar este archivo. El código completo de “vista4.php” se puede consultar en el Anexo VI de la presente memoria.

[Volver a Inicio](#)

### Seleccionar Persona

Dotacion	<a href="#">Desembarcar Todos</a>
BGD Persona4	<a href="#">Desembarcar</a>
MRO Persona2	<a href="#">Desembarcar</a>
CB Persona1	<a href="#">Desembarcar</a>
MRO Persona3	<a href="#">Desembarcar</a>

Ilustración 3-16 : vista4.php

### 3.6.7 Vista5.php

Al accionar la opción “Generar Archivo PDF” del archivo “index.php”, accederemos a “vista5.php”. Este archivo no es más que un archivo de redireccionamiento. La interfaz del archivo le muestra al usuario 3 opciones: Listado de Dotación, Situación Operativa, Control o Subcontrol y Ficha Personal. Cada una de estas opciones es un hipervínculo que redirige al usuario a la página “generarpdf.php”. Los hipervínculos “Situación Operativa”, “Control o Subcontrol” y “Ficha Personal”, previamente a dirigir al usuario al siguiente archivo, le pedirán que seleccionen, de una tabla, de qué situación quiere generar el PDF, de que control quiere imprimir los puestos o de que miembro de la dotación quiere generar una “Ficha de Personal”. Por otro lado, durante todo el funcionamiento de este archivo, existirá un enlace que le ofrece al usuario volver a la página de inicio.

Los códigos HTML y el PHP de esta aplicación son bastante sencillos. Los elementos HTML incluidos se limitan a tablas, hipervínculos y encabezados. En cuanto a los elementos PHP, este archivo define variables, arrays y realiza llamadas a datos de sesión.

Dada la sencillez de esta página, y teniendo en cuenta la experiencia de programación obtenida hasta el momento de programar este archivo, el código asociado no produjo ningún problema. En la Ilustración 3-17, que se muestra a continuación, podemos ver el interfaz que se le mostraría al usuario al acceder al archivo. El código que conforma este archivo puede consultarse en el Anexo VII de la presente memoria.

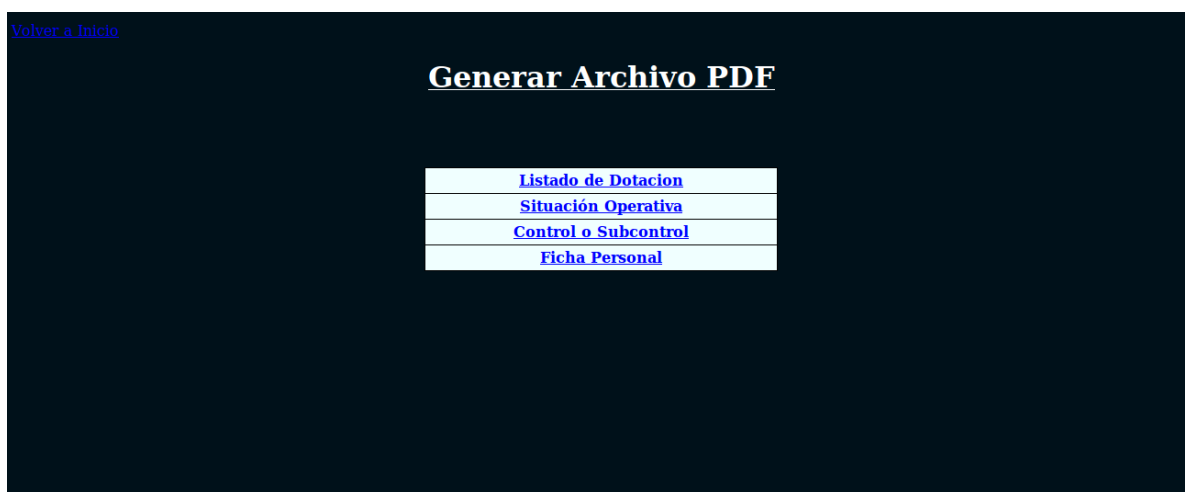


Ilustración 3-17 : vista5.php

### 3.6.8 Generarpdf.php

El archivo “generarpdf.php” comienza con una estructura de control del tipo “if” que comprueba cuál de las opciones planteadas en la página anterior ha escogido el usuario. Si el usuario seleccionó la opción “Listado de Dotación”, en esta página obtendrá un archivo PDF con una tabla en la que se relacionan todos los miembros de la dotación por nombre y rango. Si el usuario seleccionó la opción “Situación”, el archivo “generarpdf.php” devolverá un archivo PDF en el que se muestren todos los puestos de la situación seleccionada, los subcontroles de los que dependen y las personas que los ocupan. Y por último, si el usuario selecciona la opción “Ficha Personal”, en esta página obtendrá un archivo PDF en el que se muestren todos los estados en el que la persona seleccionada tiene un puesto que ocupar, y cual es dicho puesto.

Para el desarrollo de esta parte del código hizo falta el uso de las librerías FPDF (*Free PDF*). Las librerías FPDF permiten generar documentos PDF directamente desde PHP sin necesidad de utilizar las librerías PDFlib. Por sus siglas, se puede ver que la principal ventaja de las librerías FPDF sobre las PDFlib es que las primeras son gratuitas y por lo tanto utilizables para cualquier propósito y modificables con el fin de adaptarlas para satisfacer las necesidades de la aplicación. Por otra parte, las

librerías FPDF tienen menor velocidad a la hora de generar documentos que las librerías PDFlib, pero dado que los documentos a generar por esta aplicación no son de demasiada complejidad, el usuario no debería notar este decremento en velocidad [61].

El código HTML de este archivo es completamente inexistente ya que, al ser un archivo encargado de generar archivos PDF, la totalidad del archivo son comandos PHP que se encargan de construir dicho archivo PDF con los datos indicados por el usuario. El código PHP encargado de esta tarea puede ser considerado complejo puesto que a diferencia del resto de archivos vistos hasta ahora, en este archivo se ejecutan funciones de llamada a la base de datos para obtener información.

El principal problema que surgió a la hora de desarrollar este archivo fue el hecho de que el código utilizado para realizar las llamadas a la base de datos, no es del tipo “mysqli procedimental”. El código “mysqli procedimental”, el cual, como veremos posteriormente, se utiliza de para el desarrollo del archivo “modelo.php”, es el código clásico para relacionar archivos PHP con la base de datos MySQL. Para el archivo “generarpdf.php”, en ausencia de ejemplos sobre cómo generar archivos PDF utilizando el lenguaje “mysqli procedimental” hubo que utilizar el lenguaje alternativo “mysqli orientado a objetos”. Este cambio causó la necesidad de aprender a manejar dicha variante del lenguaje SQL, pero, una vez aprendido, el desarrollo del archivo dejó de causar problema alguno. En la Ilustración 3-18 se puede observar el resultado obtenido cuando el usuario acceda al archivo “generarpdf.php” habiendo solicitado un listado de la dotación. En el Anexo VIII se puede ver el código completo del archivo.

Dotacion	
Nombre	Rango
Persona4	BGD
Persona2	MRO
Persona1	CB
Persona3	MRO

Ilustración 3-18 : generarpdf.php

### 3.6.9 Control.php

El archivo “control.php” es el cerebro de esta aplicación web. Es un archivo constituido exclusivamente por código PHP, que se encarga de tramitar todas las acciones definidas en cualquiera de los demás archivos. Se estructura, principalmente, como una gran estructura de control de tipo “If”, a través de la cual se busca el valor de la variable superglobal “accion”. Según el valor que adquiera esta variable, “control.php” redirigirá al usuario al archivo “modelo.php” para ejecutar algún tipo de consulta con la base de datos o a algún otro archivo de la aplicación para devolverle una interfaz determinada.

Cada vez que el usuario ejecuta una acción que requiera el uso de la base de datos, en primer lugar se le redirige a “control.php”. Una vez en este archivo, “control.php” buscará el valor de la variable superglobal “accion” y evaluando el valor de la misma, sabrá que es exactamente lo que pretende hacer el usuario. Una vez establecida la acción a ejecutar, se llamara a todas las variables definidas por

el usuario que sean necesarias para ejecutar la consulta que corresponda con la base de datos. Cuando “control.php” disponga de todas las variables necesarias, llamara a la función concreta de “modelo.php” encargada de realizar la tarea que pide el usuario y le entregara todos los datos necesarios para realizar dicha tarea. Cuando la consulta en cuestión se haya llevado a cabo, “control.php” recuperará los resultados de la consulta y se los devolverá al archivo encargado de mostrárselos al usuario.

El código PHP que forma el archivo “control.php” es considerablemente complejo, ya que incluye prácticamente todos los tipos de elementos definibles en PHP. Este archivo utiliza variables, arrays, estructuras de control, funciones y llamadas a datos de sesión. Sin embargo, la mayor parte del código de este archivo converge en una metodología bastante repetitiva, por lo tanto se podría decir que una vez introducidas un par de “acciones”, el resto comienzan a desarrollarse sin mayor complicación. El código completo de este archivo puede consultarse en el Anexo IX de la presente memoria.

### *3.6.10 Modelo.php*

El archivo “modelo.php” es el archivo encargado de relacionarse con la base de datos. Al igual que el archivo “control.php”, este archivo no contiene código HTML, sin embargo no contiene únicamente código PHP, ya que también tiene partes del archivo en las que incorpora peticiones a la base de datos en lenguaje SQL. Este archivo está constituido por una serie de funciones que, al ser llamadas por el archivo “control.php”, ejecutan una serie de acciones de consulta sobre la base de datos (Introducir, extraer, eliminar o actualizar datos). El código PHP que constituye este archivo es el más complejo a encontrar en esta aplicación puesto que, aunque no incluye todos los elementos posibles a utilizar en PHP, cada función es distinta de la anterior, por lo que se ha de comprender el lenguaje PHP con ligera soltura para ser capaz de crear una función que haga un tarea específica de la manera correcta.

El archivo “modelo.php” está fuertemente ligado a “control.php”. Como ya se explicó anteriormente, “control.php” evalúa la petición del usuario en los archivos de interfaz y le transfiere a “modelo.php” los datos necesarios para realizar consultas con la base de datos. Por lo tanto, cada vez que la aplicación pase por el archivo “control.php”, el siguiente archivo a ejecutar será siempre “modelo.php”. Cuando “modelo.php” es llamado por “control.php”, ejecuta la función específica que se ha llamado con las variables definidas por el usuario y por “control.php”. El código que compone el archivo “modelo.php” se encuentra disponible en el Anexo X de la presente memoria.

## 4 VALIDACIÓN

En este capítulo se expondrán los distintos procesos de validación a los que se ha sometido la aplicación, distribuyéndolos en validaciones de tipo interna y validaciones de tipo externa. En las validaciones internas se evaluará la capacidad de la aplicación para llevar a cabo las funciones que de ella se esperan mediante el propio uso de la aplicación. Por su parte, las validaciones externas se llevaran a cabo sometiendo la aplicación a la evaluación por parte de oficiales de la Armada con experiencia en la realización de planes de combate.

### 4.1 Validación Interna

En este apartado, se evaluará la funcionalidad de la aplicación para asegurarse de que esta se ha implementado de manera correcta.

La principal capacidad que se espera de la aplicación es la asignación de un puesto, dentro de la organización operativa de un buque, a un miembro de la dotación. El cumplimiento de esta capacidad se comprobó a través del propio uso de la aplicación mediante el archivo “vista1.php”. Dentro de dicho archivo, seleccionando primero un grado de alistamiento (Ilustración 4-1) y después una situación operativa (Ilustración 4-2), conseguiremos obtener un listado de los puestos (Ilustración 4-3). Una vez tengamos los puestos, al seleccionar el enlace “Asignar Puesto” obtendremos otro listado con las personas disponibles para ocupar ese puestos (Ilustración 4-4). Si seleccionamos el enlace “Asignar”, que se encuentra junto a cada miembro de la dotación, el miembro al que le correspondiese el enlace quedará asignado al puesto que seleccionamos anteriormente (Ilustración 4-5). Si ahora volvemos a seleccionar la misma situación operativa que seleccionamos al principio, volveremos a ver la tabla con los puestos de dicha situación con la diferencia de que el puesto anteriormente vacío ahora tendrá asignada a la persona que hemos seleccionado (Ilustración 4-6).

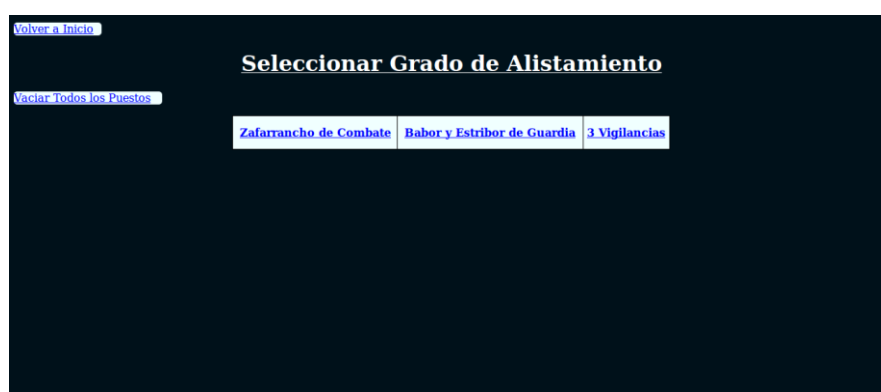


Ilustración 4-1 : Seleccionar grado de alistamiento.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

Zafarrancho de Combate	Babor y Estribor de Guardia	3 Vigilancias
------------------------	-----------------------------	---------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:

Borrar situacion con nombre:

Zafarrancho de Combate	Operaciones de Vuelo	Hombre al Agua	Escuadras de Socorro	FPL-1
------------------------	----------------------	----------------	----------------------	-------

Ilustración 4-2 : Seleccionar situación operativa.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

Zafarrancho de Combate	Babor y Estribor de Guardia	3 Vigilancias
------------------------	-----------------------------	---------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:

Borrar situacion con nombre:

Zafarrancho de Combate	Operaciones de Vuelo	Hombre al Agua	Escuadras de Socorro	FPL-1
------------------------	----------------------	----------------	----------------------	-------

[Gestionar Puestos](#)

Puesto	Subcontrol	Asignado a	
Supervisor Propulsion	ENPRO		<a href="#">Asignar Puesto</a>
Supervisor Electricidad	ENPRO		<a href="#">Asignar Puesto</a>
Operador DCC	ENPRO		<a href="#">Asignar Puesto</a>
Suboficial SI	ENPRO		<a href="#">Asignar Puesto</a>
Operador CAPES	ENPRO		<a href="#">Asignar Puesto</a>

Ilustración 4-3 : Seleccionar puesto.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

Zafarrancho de Combate	Babor y Estribor de Guardia	3 Vigilancias
------------------------	-----------------------------	---------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:

Borrar situacion con nombre:

Zafarrancho de Combate	Operaciones de Vuelo	Hombre al Agua	Escuadras de Socorro	FPL-1
------------------------	----------------------	----------------	----------------------	-------

Disponibles	
MRO Marinero 1	<a href="#">Asignar</a>
MRO Marinero 2	<a href="#">Asignar</a>
MRO Marinero 3	<a href="#">Asignar</a>
CB Cabo 1	<a href="#">Asignar</a>

Ilustración 4-4 : Asignar puesto.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Babor y Estribor de Guardia</a>	<a href="#">3 Vigilancias</a>
----------------------------------------	---------------------------------------------	-------------------------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:

Borrar situacion con nombre:

<a href="#">Zafarrancho de Combate</a>	<a href="#">Operaciones de Vuelo</a>	<a href="#">Hombre al Agua</a>	<a href="#">Escuadras de Socorro</a>	<a href="#">FPL-1</a>
----------------------------------------	--------------------------------------	--------------------------------	--------------------------------------	-----------------------

**Puesto Asignado**

Ilustración 4-5 : Puesto asignado.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Babor y Estribor de Guardia</a>	<a href="#">3 Vigilancias</a>
----------------------------------------	---------------------------------------------	-------------------------------

### Seleccionar Situacion Operativa

Añadir situacion con nombre:

Borrar situacion con nombre:

<a href="#">Zafarrancho de Combate</a>	<a href="#">Operaciones de Vuelo</a>	<a href="#">Hombre al Agua</a>	<a href="#">Escuadras de Socorro</a>	<a href="#">FPL-1</a>
----------------------------------------	--------------------------------------	--------------------------------	--------------------------------------	-----------------------

[Gestionar Puestos](#)

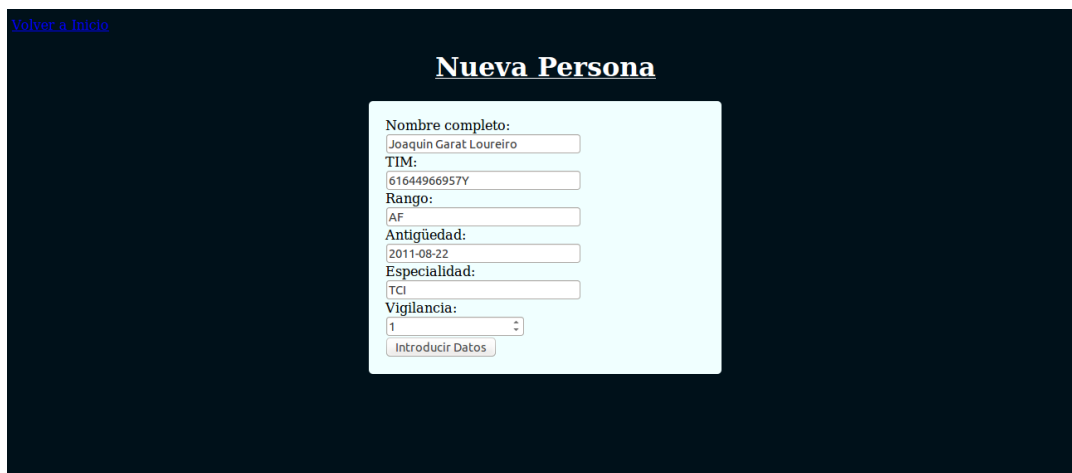
Puesto	Subcontrol	Asignado a	
Supervisor Propulsion	ENPRO	Marinero 1	<a href="#">Vaciar Puesto</a>
Supervisor Electricidad	ENPRO		<a href="#">Asignar Puesto</a>
Operador DCC	ENPRO		<a href="#">Asignar Puesto</a>
Suboficial SI	ENPRO		<a href="#">Asignar Puesto</a>
Operador CADES	ENPRO		<a href="#">Asignar Puesto</a>

Ilustración 4-6 : Validación interna - 1ª Comprobación.

Tras la comprobación del cumplimiento de los requisitos mínimos, se planteó la posibilidad de mejorar la aplicación, añadiéndole una serie de capacidades ampliatorias: Introducir y eliminar miembros de la dotación, añadir y eliminar situaciones operativas dentro de los distintos grados de alistamiento, añadir y eliminar puestos dentro de las situaciones operativas y generar archivos PDF para listar a los miembros de la dotación, listar los puestos, y las personas asignadas a los mismos, de una situación operativa concreta y listar los puestos asignados a una persona determinada en todas las situaciones operativas.

El cumplimiento de estas capacidades ampliatorias se comprobó mediante el uso de los archivos “vista3.php” (introducción de nuevos miembros a la dotación), “vista4.php” (eliminación de miembros de la dotación), “vista1.php” (añadir y eliminar situaciones operativas), “vista2.php” (añadir y eliminar puestos) y generarpdf.php (generar archivos PDF con listados de la dotación, relación de puestos y personas asignadas a los mismos en situaciones operativas, personas que ocupan los puestos asociados a un determinado subcontrol y listado de puestos ocupados por una persona en una situación operativa concreta).

En el archivo “vista3.php”, si introducimos los datos en el formulario y accionamos el botón “Introducir Datos” (Ilustración 4-7), podremos comprobar cómo se añade una nueva persona a nuestra dotación (Ilustración 4-9). Si por el contrario, accionamos el botón sin haber rellenado alguno de los tres primeros campos (Nombre, TIM y rango), podremos observar como la propia aplicación nos avisa de que el campo está vacío y no nos permitirá introducir datos mientras alguno de estos campos siga vacío (Ilustración 4-8).



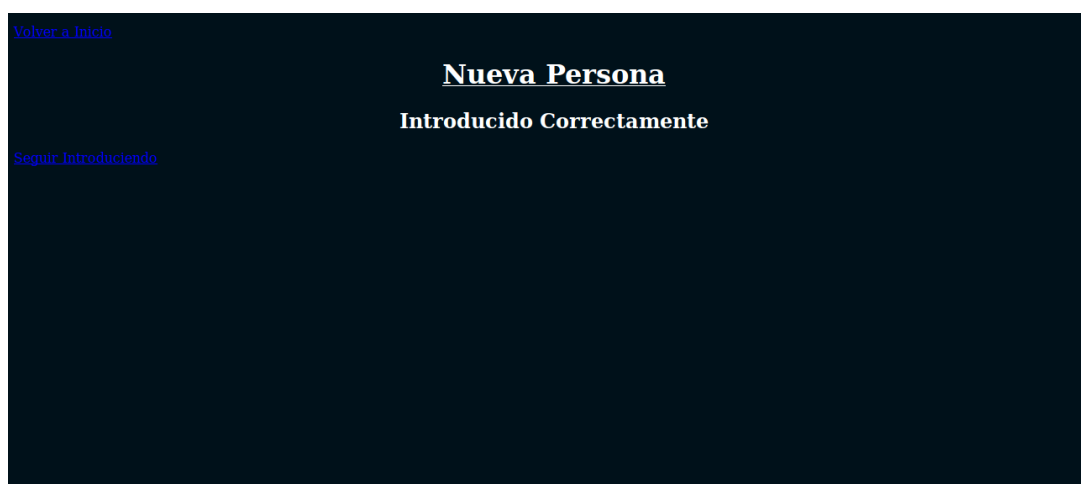
The screenshot shows a web application interface with a dark blue background. In the top left corner, there is a link labeled "Volver a Inicio". Centered at the top is the title "Nueva Persona" in a bold, white font. Below the title is a light blue rectangular form containing several input fields. The fields are labeled as follows: "Nombre completo:" with the value "Joaquín Garat Loureiro"; "TIM:" with the value "61644966957Y"; "Rango:" with the value "AF"; "Antigüedad:" with the value "2011-08-22"; "Especialidad:" with the value "TCI"; and "Vigilancia:" with a dropdown menu showing "1". At the bottom of the form is a button labeled "Introducir Datos".

Ilustración 4-7 : Introducir datos.



This screenshot shows the same "Nueva Persona" form, but with a validation error. The "Nombre completo:" field is highlighted with a red border. A grey tooltip box is positioned over the field, containing the text "Please fill out this field." Below the tooltip, the "Rango:" field is visible with the value "AF". The other fields and the "Introducir Datos" button remain the same as in the previous illustration.

Ilustración 4-8 : Alerta por falta de datos.



The screenshot shows the "Nueva Persona" form area, but the form itself is no longer visible. Instead, the text "Introducido Correctamente" is displayed in a bold, white font, indicating a successful submission. In the top left corner, the link "Volver a Inicio" is present, and in the bottom left corner, a new link labeled "Seguir Introduciendo" has appeared.

Ilustración 4-9 : Validación interna - 2ª Comprobación.

Para desembarcar miembros de la dotación, al llamar al archivo “vista4.php”, la aplicación nos mostrará una tabla con todos los miembros de la dotación (Ilustración 4-10). Si pulsamos el enlace “Desembarcar” situado junto a cada persona, desembarcaremos a esa persona específicamente (Ilustración 4-11). De manera similar, si pulsamos el enlace “Desembarcar Todos” situado en la parte superior de la tabla, desembarcaremos a toda la dotación (Ilustración 4-12).



[Volver a Inicio](#)

### Seleccionar Persona

Dotacion	Desembarcar Todos
MRO Marinero 1	<a href="#">Desembarcar</a>
MRO Marinero 2	<a href="#">Desembarcar</a>
MRO Marinero 3	<a href="#">Desembarcar</a>
CB Cabo 1	<a href="#">Desembarcar</a>
AF Joaquín Garat Loureiro	<a href="#">Desembarcar</a>

Ilustración 4-10 : Desembarcar persona.



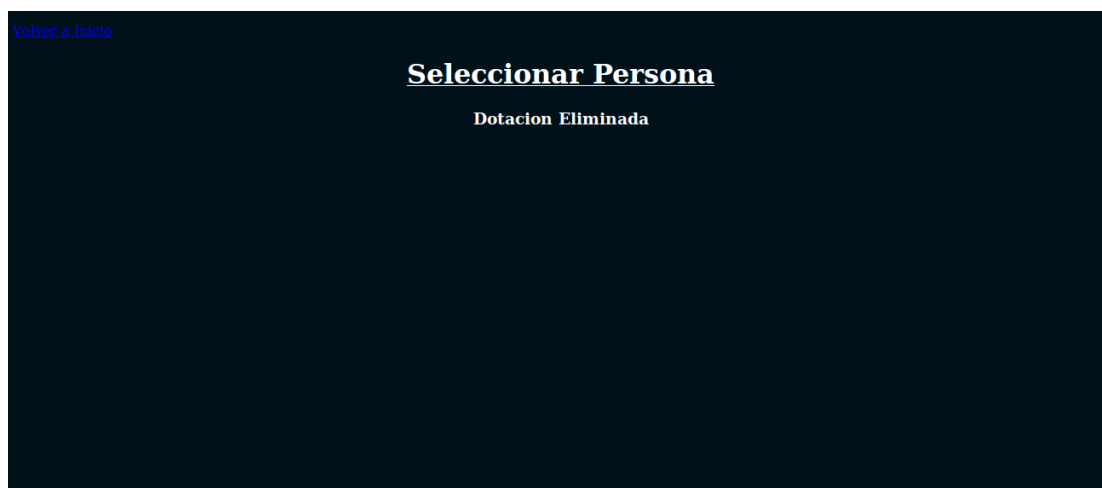
[Volver a Inicio](#)

### Seleccionar Persona

Dotacion	
MRO Marinero 1	<a href="#">Desembarcar</a>
MRO Marinero 2	<a href="#">Desembarcar</a>
MRO Marinero 3	<a href="#">Desembarcar</a>
CB Cabo 1	<a href="#">Desembarcar</a>

Proceso realizado con éxito.

Ilustración 4-11 : Validación interna - 3ª Comprobación



[Volver a Inicio](#)

### Seleccionar Persona

Dotacion Eliminada

Ilustración 4-12 : Desembarcar dotación completa.

Por otra parte, en “vista1.php”, tras seleccionar un grado de alistamiento, se nos mostrará un formulario con el cual podemos introducir una nueva situación operativa dentro del grado de alistamiento que hayamos seleccionado, o bien, eliminar una de las ya existentes. Si introducimos un nombre en el campo denominado “Nueva situación operativa con nombre:” y accionamos el botón “Añadir” (Ilustración 4-13) se volverán a cargar las situaciones operativas del grado de alistamiento en el que estuviésemos y podremos observar como se ha añadido una nueva con el nombre que hubiésemos indicado (Ilustración 4-14). Por el contrario, si introducimos el nombre de una situación ya existente en el campo del formulario denominado “Borrar situación con nombre:” y accionamos el botón “Borrar” (Ilustración 4-15), una vez más, se volverán a cargar las situaciones y podremos observar como la situación que introdujimos en el formulario ya no está (Ilustración 4-16).

Ilustración 4-13 : Introducir nueva situación.

Ilustración 4-14 : Validación interna - 4ª Comprobación.

Ilustración 4-15 : Borrar situación existente.

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Babor y Estribor de Guardia</a>	<a href="#">3 Vigilancias</a>
----------------------------------------	---------------------------------------------	-------------------------------

### Seleccionar Situación Operativa

Añadir situación con nombre:

[Añadir](#)

Borrar situación con nombre:

[Borrar](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Operaciones de Vuelo</a>	<a href="#">Hombre al Agua</a>	<a href="#">Escuadras de Socorro</a>	<a href="#">FPL-1</a>
----------------------------------------	--------------------------------------	--------------------------------	--------------------------------------	-----------------------

Ilustración 4-16 : Validación interna - 5ª Comprobación.

De manera complementaria, si al acceder a los puestos de una situación determinada accionamos el enlace “Gestionar Puestos” (Ilustración 4-17), la aplicación nos redirigirá a “vista2.php” donde añadiremos nuevos puestos a la situación previamente seleccionada, o eliminaremos alguno de los puestos ya existentes. Si, cuando la aplicación nos devuelva el archivo “vista2.php”, accionamos el enlace “Añadir Puestos” (Ilustración 4-18), la aplicación nos devolverá un formulario para introducir el nombre del puesto que queramos añadir. Una vez introducido el nombre y accionado el botón “Añadir” (Ilustración 4-19), la aplicación nos dirá si el proceso se ha realizado correctamente (Ilustración 4-20). Si ahora volvemos al plan de combate, podremos observar, al final de la tabla de puestos, el puesto que acabamos de añadir (Ilustración 4-21). Por otra parte, al acceder al archivo “vista2.php”, también se nos devolverá una tabla con todos los puestos de la situación operativa. Si accionamos el enlace “Borrar Puesto” que se encuentra al lado de cada puesto (Ilustración 4-22), eliminaremos el puesto que estuviese al lado de dicho enlace (Ilustración 4-23). Si ahora volvemos a cargar el plan de combate, podremos observar que el puesto que acabamos de borrar ya no se encuentra al final de la tabla (Ilustración 4-24).

[Volver a Inicio](#)

## Seleccionar Grado de Alistamiento

[Vaciar Todos los Puestos](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Babor y Estribor de Guardia</a>	<a href="#">3 Vigilancias</a>
----------------------------------------	---------------------------------------------	-------------------------------

### Seleccionar Situación Operativa

Añadir situación con nombre:

[Añadir](#)

Borrar situación con nombre:

[Borrar](#)

<a href="#">Zafarrancho de Combate</a>	<a href="#">Operaciones de Vuelo</a>	<a href="#">Hombre al Agua</a>	<a href="#">Escuadras de Socorro</a>	<a href="#">FPL-1</a>
----------------------------------------	--------------------------------------	--------------------------------	--------------------------------------	-----------------------

[Gestionar Puestos](#)

Puesto	Subcontrol	Asignado a	
Supervisor Propulsion	ENPRO	Marinero 1	<a href="#">Vaciar Puesto</a>
Supervisor Electricidad	ENPRO		<a href="#">Asignar Puesto</a>
Operador DCC	ENPRO		<a href="#">Asignar Puesto</a>
Suboficial SI	ENPRO		<a href="#">Asignar Puesto</a>
Operador CAPES	ENPRO		<a href="#">Asignar Puesto</a>

Ilustración 4-17 : Gestionar puestos.

[Volver a Inicio](#)  
[Volver al Plan de Combate](#)

### Gestionar Puestos

[Añadir Puesto](#)

Puesto	Subcontrol	
Supervisor Propulsion	11	<a href="#">Borrar Puesto</a>
Supervisor Electricidad	11	<a href="#">Borrar Puesto</a>
Operador DCC	11	<a href="#">Borrar Puesto</a>
Suboficial SI	11	<a href="#">Borrar Puesto</a>
Operador CAPES	11	<a href="#">Borrar Puesto</a>
Jefe de Escena PROA	11	<a href="#">Borrar Puesto</a>
Telefonista PROA	11	<a href="#">Borrar Puesto</a>
Electricista PROA	11	<a href="#">Borrar Puesto</a>
Controlador EERR PROA	11	<a href="#">Borrar Puesto</a>
Investigador PROA	11	<a href="#">Borrar Puesto</a>
Ayudante Investigador PROA	11	<a href="#">Borrar Puesto</a>
Mensajero PROA	11	<a href="#">Borrar Puesto</a>
Jefe Grupo CI PROA	11	<a href="#">Borrar Puesto</a>
Manguera PROA	11	<a href="#">Borrar Puesto</a>

Ilustración 4-18 : Añadir puesto.

[Volver a Inicio](#)  
[Volver al Plan de Combate](#)

### Gestionar Puestos

Puesto Nuevo:

Añadir

Ilustración 4-19 : Formulario de nuevo puesto.

[Volver a Inicio](#)  
[Volver al Plan de Combate](#)

### Gestionar Puestos

Puesto Añadido

Ilustración 4-20 : Puesto añadido.

Cronica	INFO	<a href="#">Asignar Puesto</a>
INTEL	INFO	<a href="#">Asignar Puesto</a>
MHS	INFO	<a href="#">Asignar Puesto</a>
RSC	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 1	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 2	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 3	S/C	<a href="#">Asignar Puesto</a>
Jefe Reparacion AWS	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 1	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 2	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 3	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 4	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 5	S/C	<a href="#">Asignar Puesto</a>
Auxiliar CSOOW	S/C	<a href="#">Asignar Puesto</a>
Supervisor VLS	S/C	<a href="#">Asignar Puesto</a>
Operador VLS	S/C	<a href="#">Asignar Puesto</a>
Operador ORTS	S/C	<a href="#">Asignar Puesto</a>
FCS POPA	S/C	<a href="#">Asignar Puesto</a>
FCS PROA	S/C	<a href="#">Asignar Puesto</a>
Mantenimiento DORNA/HARPOON	S/C	<a href="#">Asignar Puesto</a>
Operador Regulus	Comunicaciones	<a href="#">Asignar Puesto</a>
Supervisor Radio	Comunicaciones	<a href="#">Asignar Puesto</a>
Nuevo puesto		<a href="#">Asignar Puesto</a>

Ilustración 4-21 : Validación interna - 6ª Comprobación.

IC	INFO	<a href="#">Borrar Puesto</a>
IDS	INFO	<a href="#">Borrar Puesto</a>
Cronica	INFO	<a href="#">Borrar Puesto</a>
INTEL	INFO	<a href="#">Borrar Puesto</a>
MHS	INFO	<a href="#">Borrar Puesto</a>
RSC	S/C	<a href="#">Borrar Puesto</a>
Grupo Reparacion Electr. 1	S/C	<a href="#">Borrar Puesto</a>
Grupo Reparacion Electr. 2	S/C	<a href="#">Borrar Puesto</a>
Grupo Reparacion Electr. 3	S/C	<a href="#">Borrar Puesto</a>
Jefe Reparacion AWS	S/C	<a href="#">Borrar Puesto</a>
Reparacion AWS 1	S/C	<a href="#">Borrar Puesto</a>
Reparacion AWS 2	S/C	<a href="#">Borrar Puesto</a>
Reparacion AWS 3	S/C	<a href="#">Borrar Puesto</a>
Reparacion AWS 4	S/C	<a href="#">Borrar Puesto</a>
Reparacion AWS 5	S/C	<a href="#">Borrar Puesto</a>
Auxiliar CSOOW	S/C	<a href="#">Borrar Puesto</a>
Supervisor VLS	S/C	<a href="#">Borrar Puesto</a>
Operador VLS	S/C	<a href="#">Borrar Puesto</a>
Operador ORTS	S/C	<a href="#">Borrar Puesto</a>
FCS POPA	S/C	<a href="#">Borrar Puesto</a>
FCS PROA	S/C	<a href="#">Borrar Puesto</a>
Mantenimiento DORNA/HARPOON	S/C	<a href="#">Borrar Puesto</a>
Operador Regulus	Comunicaciones	<a href="#">Borrar Puesto</a>
Supervisor Radio	Comunicaciones	<a href="#">Borrar Puesto</a>
Nuevo puesto		<a href="#">Borrar Puesto</a>

Ilustración 4-22 : Borrar puesto.

[Volver a Inicio](#)

[Volver al Plan de Combate](#)

## Gestionar Puestos

Puesto Borrado

Ilustración 4-23 : Puesto borrado.

INTEL	INFO	<a href="#">Asignar Puesto</a>
MHS	INFO	<a href="#">Asignar Puesto</a>
RSC	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 1	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 2	S/C	<a href="#">Asignar Puesto</a>
Grupo Reparacion Electr. 3	S/C	<a href="#">Asignar Puesto</a>
Jefe Reparacion AWS	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 1	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 2	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 3	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 4	S/C	<a href="#">Asignar Puesto</a>
Reparacion AWS 5	S/C	<a href="#">Asignar Puesto</a>
Auxiliar CSOOW	S/C	<a href="#">Asignar Puesto</a>
Supervisor VLS	S/C	<a href="#">Asignar Puesto</a>
Operador VLS	S/C	<a href="#">Asignar Puesto</a>
Operador ORTS	S/C	<a href="#">Asignar Puesto</a>
FCS POPA	S/C	<a href="#">Asignar Puesto</a>
FCS PROA	S/C	<a href="#">Asignar Puesto</a>
Mantenimiento DORNA/HARPOON	S/C	<a href="#">Asignar Puesto</a>
Operador Regulus	Comunicaciones	<a href="#">Asignar Puesto</a>
Supervisor Radio	Comunicaciones	<a href="#">Asignar Puesto</a>

Ilustración 4-24 : Validación interna - 7ª Comprobación.

Finalmente, para que la aplicación nos genere archivos PDF, deberemos ir a la “vista5.php” a través del enlace “Generar Archivo PDF” que se encuentra en la página de inicio. Una vez la aplicación nos haya devuelto el contenido de la nueva vista, se nos mostrará una tabla con las distintas opciones de archivos PDF que podemos generar (Ilustración 4-25). Si seleccionamos la opción “Listado de Dotación” la aplicación nos devolverá un archivo PDF con toda la dotación listada por nombre y rango (Ilustración 4-26). Si seleccionamos la opción “Situación Operativa”, primero se nos pedirá que seleccionemos un grado de alistamiento (Ilustración 4-27), y después una situación operativa dentro de dicho grado de alistamiento (Ilustración 4-28). De este modo la aplicación nos devolverá un nuevo archivo PDF con la información de todos los puestos de la situación operativa seleccionada (Ilustración 4-29). De manera similar, si seleccionamos la opción “Control o Subcontrol”, primero seleccionaremos uno de los dos controles (Ilustración 4-30) y después un subcontrol dentro del control seleccionado (Ilustración 4-31). Como resultado la aplicación nos devolverá un archivo PDF con todos los puestos que dependan de este subcontrol (Ilustración 4-32). Por último, si seleccionamos la opción “Ficha Personal”, se nos devolverá un listado de la dotación (Ilustración 4-33). Si seleccionamos a un miembro de la dotación de este listado, la aplicación nos generará un archivo PDF con todos los puestos ocupados por la persona seleccionada en cada situación en la que ocupe un puesto (Ilustración 4-34).

[Volver a Inicio](#)

## Generar Archivo PDF

<a href="#">Listado de Dotacion</a>
<a href="#">Situación Operativa</a>
<a href="#">Control o Subcontrol</a>
<a href="#">Ficha Personal</a>

Ilustración 4-25 : Generar archivo PDF.

### Dotación

Nombre	Rango
Marinero 1	MRO
Marinero 2	MRO
Marinero 3	MRO
Cabo 1	CB
Joaquin Garat Loureiro	AF



Ilustración 4-26 : Validación interna - 8ª Comprobación.

[Volver a Inicio](#)

## Generar Archivo PDF

<a href="#">Zafarrancho de Combate</a>
<a href="#">Babor y Estribor de Guardia</a>
<a href="#">3 Vigilancias</a>

Ilustración 4-27 : Seleccionar grado de alistamiento.

[Volver a Inicio](#)

## Generar Archivo PDF

Situaciones
<a href="#">Zafarrancho de Combate</a>
<a href="#">Operaciones de Vuelo</a>
<a href="#">Hombre al Agua</a>
<a href="#">Escuadras de Socorro</a>
<a href="#">EPL-1</a>

Ilustración 4-28 : Seleccionar situación operativa.

**Zafarrancho de Combate**

Puesto	Subcontrol	Asignado a
Supervisor Propulsion	ENPRO	Marinero 1
Supervisor Electricidad	ENPRO	Marinero 2
Operador DCC	ENPRO	Marinero 3
Suboficial SI	ENPRO	Cabo 1
Operador CAPES	ENPRO	Joaquin Garat Loureiro
Jefe de Escena PROA	ENPRO	
Telefonista PROA	ENPRO	
Electricista PROA	ENPRO	
Controlador EERR PROA	ENPRO	
Investigador PROA	ENPRO	
Ayudante Investigador PROA	ENPRO	
Mensajero PROA	ENPRO	

**Ilustración 4-29 : Validación interna - 9ª Comprobación.**[Volver a Inicio](#)**Generar Archivo PDF**[Control de Operaciones](#)[Control de Plataforma](#)**Ilustración 4-30 : Seleccionar control.**[Volver a Inicio](#)**Generar Archivo PDF****Operaciones**[AAW](#)[Coordinación Aérea](#)[ASUW](#)[ASW](#)[EW](#)[Comunicaciones](#)[INFO](#)[MFP](#)[S/C](#)**Ilustración 4-31 : Seleccionar subcontrol.**

### Control de Operaciones - AAW



Situación Operativa	Puesto	Asignado a
Zafarrancho de Combate	MSS	
3 Vigilancias	MSS 1ª	
3 Vigilancias	MSS 2ª	
3 Vigilancias	MSS 3ª	

Ilustración 4-32 Validación interna - 10ª Comprobación.

[Volver a Inicio](#)

### Generar Archivo PDF

Dotacion
<a href="#">MRO Marinero 1</a>
<a href="#">MRO Marinero 2</a>
<a href="#">MRO Marinero 3</a>
<a href="#">CB Cabo 1</a>
<a href="#">AF Joaquín Garat Loureiro</a>

Ilustración 4-33 : Seleccionar miembro de la dotación.

### Fichas de Personal



Joaquín Garat Loureiro	
Estado	Puesto
Babor y Estribor de Guardia	Supervisor Propulsion
3 Vigilancias	Supervisor Maquinas 1ª
Zafarrancho de Combate	Operador CAPES
Operaciones de Vuelo	HCO/Jefe de Trozo
Hombre al Agua	Electricista
FPL-2	Supervisor 20mm Er

Ilustración 4-34 : Validación interna - 11ª Comprobación.

## 4.2 Validación Externa

Tras comprobar que la aplicación funciona correctamente, se somete a validación externa. Este periodo fue llevado a cabo por el Teniente de Navío del Cuerpo General de la Armada, D. Luis Miranda García, con 9 años de experiencia en la creación y gestión de planes de combate a bordo de buques de la Armada.

Este periodo de validación consistió en el uso de la aplicación para llevar a cabo todas las modificaciones que se puedan dar sobre un plan de combate. El TN. Miranda llevó a cabo la incorporación de personal a la dotación del buque, la asignación de este personal con puestos en el plan de combate, la creación y eliminación de puestos, la sustitución de un miembro por otro en los puestos del plan de combate y la impresión del plan de combate en archivos PDF.

La aplicación fue considerada como una sustitución válida, pero incompleta, del actual método de desarrollo de planes de combate. Este periodo de evaluación concluyó con la asimilación de la necesidad de incorporar en la aplicación, la capacidad de crear relaciones entre puestos de manera que se puedan asignar varios puestos a una misma persona de manera automática. El objetivo sería conseguir que, tras asignarle a dicha persona uno de los puestos que se desee que ocupe, el resto de puestos se le asignen automáticamente. Esta capacidad sería de gran ayuda a la hora de desembarcar a una persona del buque y embarcar a otra que ocupe todos los puestos que la primera persona ha dejado vacantes.

## 5 CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se presentan las conclusiones obtenidas tras y durante la realización de este TFG así como las líneas futuras a seguir para su posible mejora.

### 5.1 Conclusiones

Como conclusión principal, se considera que este TFG ha cumplido los objetivos de desarrollar una aplicación *web* que realice las funciones necesarias para crear y gestionar planes de combate. Por otra parte, se puede considerar que la eficiencia y utilidad de la aplicación no han alcanzado los niveles suficientes para que, el hecho de sustituir el actual método de desarrollo de planes de combate por la aplicación, realmente suponga un avance notable en la rapidez con la que se desarrollan.

A continuación se establecen una serie de líneas futuras con las cuales se podría conseguir alcanzar esos niveles de utilidad y eficiencia que podrían llevar a considerar esta aplicación como una mejora indiscutible sobre la metodología actual de desarrollo de los planes de combate.

### 5.2 Líneas Futuras

Esta aplicación se puede considerar como el inicio de un camino que culmine con el desarrollo de una aplicación moderna, funcional y operativa que se pueda integrar en los buques de la Armada y que no solo permita gestionar los planes de combate sino controlar su aplicación.

En tanto al código de la aplicación, el principal camino a seguir para su mejora sería la actualización de los procesos desarrollados en lenguaje PHP y su posterior sustitución, en los casos que corresponda, por los mismos procesos desarrollados en JavaScript. La principal razón por la que el cambio a JavaScript sería considerado como una mejora es la diferencia entre el modo que utiliza PHP para presentar información al usuario y el modo que utiliza JavaScript. PHP es un lenguaje que se ejecuta en el lado del servidor, es decir, cuando se accede a un archivo PHP, el código es ejecutado en el servidor y el resultado es enviado al usuario que accedió al archivo. JavaScript, por el contrario, es un lenguaje del lado del cliente, significando esto que cuando el cliente accede a la aplicación *web*, se le envía el código una vez y lo podrá ejecutar tantas veces y de tantas maneras como quiera. Podemos concluir por lo tanto que JavaScript ahorra muchos envíos entre el usuario y el servidor y para aplicaciones con alto contenido en formularios, animaciones y más contenido del tipo dinámico es recomendable.

En tanto a la funcionalidad y las capacidades de la propia aplicación, el principal camino a seguir para alcanzar los objetivos que se pretenden con su desarrollo, que no es otro que su implementación en los buques de la Armada, sería añadir la posibilidad de crear relaciones entre puestos de tal manera que al asignar a un miembro de la dotación un puesto que esté relacionado con otros, este miembro

automáticamente pase a ocupar todos los puestos que están relacionados con el primero que se le asignó. Lo que se pretende conseguir con esta mejora es agilizar el proceso de sustitución de miembros de la dotación, mediante el cual un miembro desembarca dejando todos sus puestos vacíos, y otro miembro embarca para ocupar todos los puestos que dejó libre el primero. Con la creación de las relaciones entre puestos ya no sería necesario ir asignándole todos los puestos, uno a uno, al nuevo miembro de la dotación. Más bien, si existe la relación entre los puestos, con asignarle solo uno de ellos, automáticamente pasaría a ocuparlos todos.

Otras líneas futuras para mejorar la aplicación podrían incluir la posibilidad de que la propia aplicación recomiende que personas son las más idóneas para ocupar un puesto, o que la aplicación pueda distinguir en qué tipo de buque está siendo integrada para adaptar los puestos a los necesarios para ese tipo de buque.

Con respecto a las recomendaciones por parte de la aplicación, si añadiésemos una serie de requerimientos de rango, especialidad, antigüedad y cursos a los puestos del plan de combate y detallásemos estos datos al embarcar a cada miembro, la aplicación podría ser capaz de comprobar que miembros de la dotación cumplen con los requisitos mínimos de un puesto y darnos una recomendación de las personas más aptas para ocupar dicho puesto.

Para finalizar, con respecto a la personalización de los puestos según el tipo de buque, si le añadiésemos a los puestos información acerca del tipo de buque en el que existe dicho puesto y creásemos una base de datos con los tipos de buques de la armada, la aplicación podría llegar a pedirnos especificaciones acerca de en qué buque se encuentra y adaptar los puestos a los mínimos y necesarios que requiere el tipo de buque que hayamos especificado, permitiendo la modificación por parte del usuario de dichos puestos.

## 6 BIBLIOGRAFÍA

- [1] “Psicología y Empresa. Renovando empresas con talento humano”  
<http://psicologiayempresa.com/la-gestion-de-recursos-humanosimportancia-y-objetivos.html>  
[Último acceso: 18/01/2016].
- [2] “Wikipedia. La enciclopedia libre. Armada Española”  
[https://es.wikipedia.org/wiki/Armada\\_Espa%C3%B1ola#Buques\\_en\\_construcci.C3.B3n](https://es.wikipedia.org/wiki/Armada_Espa%C3%B1ola#Buques_en_construcci.C3.B3n)  
[Último acceso: 18/01/2016].
- [3] CC. Francisco J. Ramos Meca, TN. Salvador Pinar Gris y CC. Celestino S. López de Castro García. Guía para la redacción de los manuales de organización en los buques. (D-CP-03) Marzo 2004.
- [4] “El blog de WorkMeter. Optimiza la eficiencia de tu empresa”  
<http://es.workmeter.com/blog/bid/192132/5-sofwarees-para-gestion-de-recursos-humanos>  
[Último acceso: 26/01/2016]
- [5] “NEGOCIOS Y EMPRENDIMIENTO. Ideas y Herramientas para Emprender”  
<http://www.negociosyemprendimiento.org/2011/07/programas-gestion-de-recursos-humanos.html>  
[Último acceso: 26/01/2016]
- [6] “Autotask” <http://www.autotask.com/es/footer-pages/acerca-de-autotask>  
[Último acceso: 26/01/2016].
- [7] “OPTIMA IT, expertos en servicios gestionados” <http://www.optimait.es/autotask>  
[Último acceso: 26/01/2016]
- [8] “Sistemas en Recursos Humanos” <http://sistemasenrecursoshumanos.jimdo.com/gestpeople/>  
[Último acceso: 01/02/2016].
- [9] “ABCdatos. Programas y tutoriales en castellano” <http://www.abcdatos.com/programa/recursos-humanos-gestion.html> [Último acceso: 01/02/2016]

- [10] “WorkMeter. Buen trabajo” <http://web.workmeter.com/es/software-productividad.html> [Último acceso: 01/02/2016].
- [11] “WorkMeter. Buen trabajo” <http://web.workmeter.com/es/faq-medir-productividad.html> [Último acceso: 01/02/2016]
- [12] “e-volution. Gestión de Personas” <http://www.evolution.com.ec/producto/e-volution-gestion-de-personas> [Último acceso: 01/02/2016]
- [13] “e-volution. Gestión de Personas” <http://www.evolution.com.ec/producto/e-volution> [Último acceso: 01/02/2016]
- [14] “e-volution. Gestión de Personas” <http://www.evolution.com.ec/empresa/quienes-somos> [Último acceso: 01/02/2016]
- [15] “SimpleHRM” <http://www.simplehrmlive.com/about.html> [Último acceso: 01/02/2016]
- [16] “Tecnologías de Información en la empresa. Soluciones de TI en los negocios, pymes, organizaciones” <http://innovando-ti.blogspot.com.es/2013/09/simple-hrm-software-open-source-para-la.html> [Último acceso: 01/02/2016]
- [17] “RRHH de Recursos Humanos.net. Comunidad de los profesionales en Recursos Humanos & RRHH 2.0” <http://www.derrhh.net/video/emagister-rrhh-tu-herramienta> [Último acceso: 01/02/2016]
- [18] “Wikipedia. La enciclopedia libre. Lenguaje de marcado” [https://es.wikipedia.org/wiki/Lenguaje\\_de\\_marcado](https://es.wikipedia.org/wiki/Lenguaje_de_marcado) [Último acceso: 03/02/2016]
- [19] “Wikipedia. La enciclopedia libre. HTML” <https://es.wikipedia.org/wiki/HTML> [Último acceso: 03/02/2016]
- [20] “Wikipedia. La enciclopedia libre. Lenguaje de programación” [https://es.wikipedia.org/wiki/Lenguaje\\_de\\_programaci%C3%B3n](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [Último acceso: 03/02/2016]
- [21] Mark Diver. “Gartner Blog Network” [http://blogs.gartner.com/mark\\_driver/2014/10/02/gartner-programming-language-index-for-2014/](http://blogs.gartner.com/mark_driver/2014/10/02/gartner-programming-language-index-for-2014/) [Último acceso: 03/02/2016]
- [22] Luis Miguel Cabezas Granado y Francisco José González Lozano, Desarrollo Web con PHP y MySQL, Editorial Anaya, 2015. ISBN: 978-84-415-3568-8
- [23] “Wikipedia. La enciclopedia libre. Base de datos” [https://es.wikipedia.org/wiki/Base\\_de\\_datos](https://es.wikipedia.org/wiki/Base_de_datos) [Último acceso: 03/02/2016]
- [24] Gustavo Nipas. “in SlideShare” <http://es.slideshare.net/nipas/10-sgbd> [Último acceso: 03/02/2016]
- [25] Victor Zevallos Palacios. “in SlideShare” <http://es.slideshare.net/VictorZevallos/comparacion-de-gestores-de-base-de-datos> [Último acceso: 03/02/2016]

- [26] “EcuRed. Conocimiento con todos y para todos” [http://www.ecured.cu/Servidor\\_Web](http://www.ecured.cu/Servidor_Web)  
[Último acceso: 04/02/2016]
- [27] “masadelante.com / servicios y recursos para tener éxito en internet”  
<https://www.masadelante.com/faqs/servidor-web> [Último acceso: 04/02/2016]
- [28] “Digital Learning” <http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/> [Último acceso: 04/02/2016]
- [29] “Wikipedia. La enciclopedia libre. Servidor HTTP Apache”  
[https://es.wikipedia.org/wiki/Servidor\\_HTTP\\_Apache#Uso](https://es.wikipedia.org/wiki/Servidor_HTTP_Apache#Uso) [Último acceso: 04/02/2016]
- [30] Cristian Zapata. “MANTENIMIENTO DE UNA COMPUTADORA”  
<http://mantenimientosdeunapc.blogspot.com.es/2011/11/que-es-xampp-y-para-que-sirve.html>  
[Último acceso: 04/02/2016]
- [31] “ANDROS fenoll.sa. Diseño y Programación Web” [https://programadorwebvalencia.com/cual-es-el-mejor-sistema-operativo/#.Vrn92\\_nhDIV](https://programadorwebvalencia.com/cual-es-el-mejor-sistema-operativo/#.Vrn92_nhDIV) [Último acceso: 04/02/2016]
- [32] “VirtualBox” <https://www.virtualbox.org/wiki/Downloads> [Último acceso: 18/02/2016]
- [33] “How to Ubuntu” <http://howtoubuntu.org/how-to-install-lamp-on-ubuntu>  
[Último acceso: 22/02/2016]
- [34] Olivier Heurtel, PHP y MySQL. Domine el desarrollo de un sitio web dinámico e interactivo,  
Editorial Ediciones ENI, 2014. ISBN: 978-2-7460-9037-8
- [35] “tomando notas” <https://tomatoma.wordpress.com/manual-de-php/2-estructuras-de-control-if-for-for-each-while/> [Último acceso: 22/02/2016]
- [36] “CODEHERO” <http://codehero.co/php-desde-cero-estructuras-de-control/>  
[Último acceso: 22/02/2016]
- [37] “tutorialspoint. SimpleEasyLearning” [http://www.tutorialspoint.com/php/php\\_sessions.htm](http://www.tutorialspoint.com/php/php_sessions.htm)  
[Último acceso: 22/02/2016]
- [38] “w3schools.com The World’s Largest Web Developer Site”  
<http://www.w3schools.com/html/default.asp> [Último acceso: 22/02/2016]
- [39] “w3schools.com The World’s Largest Web Developer Site”  
<http://www.w3schools.com/css/default.asp> [Último acceso: 22/02/2016]
- [40] “GestPeople” <http://gestpeople.uptodown.com/> [Último acceso: 01/03/2016]
- [41] “Wikipedia. La enciclopedia libre. HTML5” <https://en.wikipedia.org/wiki/HTML5>  
[Último acceso: 01/03/2016]
- [42] “prweb” <http://www.prweb.com/releases/2013/8/prweb11024341.htm>

[Último acceso: 01/03/2016]

[43] “neuHRona. Noticias Tecnológicas para los Profesionales de los RRHH”  
<https://josepvila.wordpress.com/2011/09/20/gestpeople-rrhh-para-pymes/>  
[Último acceso: 01/03/2016]

[44] David Torné. “productividad desarrollo davidtorné”  
<http://blog.davidtorne.com/es/2012/02/analiza-tu-actividad-con-workmeter/>  
[Último acceso: 01/03/2016]

[45] “GNOME HELP” <https://help.gnome.org/misc/release-notes/2.20/index.html.es>  
[Último acceso: 01/03/2016]

[46] “free(code)” <http://freecode.com/projects/simplehrm> [Último acceso: 01/03/2016]

[47] “softonic. La guía de software más completa del mundo” <http://emagister-rrhh.softonic.com/>  
[Último acceso: 01/03/2016]

[48] “w3” <https://www.w3.org/html/logo/> [Último acceso: 01/03/2016]

[49] “TARINGA!” <http://www.taringa.net/post/ciencia-educacion/19201418/PHP-y-POO-Programacion-Orientada-a-Objetos-Ejemplos.html> [Último acceso: 01/03/2016]

[50] “PROTOINFORMÁTICO” <http://protoinformatico.com/2015/09/mysql-server-configurar-para-conectar-desde-cliente/> [Último acceso: 01/03/2016]

[51] “NEOTEO” <http://www.neoteo.com/servidor-http-apache-los-mejores-desarrolladores>  
[Último acceso: 01/03/2016]

[52] “Dustin Meredith” <http://dustinmeredith.ca/> [Último acceso: 01/03/2016]

[53] “linuxadictos” <http://www.linuxadictos.com/llega-la-primera-build-de-ubuntu-next-con-snappy.html> [Último acceso: 01/03/2016]

[54] “Doctores del PC. Servicio Técnico Informático”  
<http://www.doctoresdelpc.com/noticias/virtualbox-version-4-x-desde-la-consola/>  
[Último acceso: 01/03/2016]

[55] “.net y punto” <https://pontonetypunto.wordpress.com/> [Último acceso: 01/03/2016]

[56] “The APACHE software foundation” <http://www.apache.org/> [Último acceso: 01/03/2016]

[57] “MySQL. The world’s most popular open source database” <https://www.mysql.com/>  
[Último acceso: 01/03/2016]

[58] “php” <http://php.net/> [Último acceso: 01/03/2016]

[59] “ubuntu” <http://www.ubuntu.com/> [Último acceso: 01/03/2016]

[60] “VirtualBox” <https://www.virtualbox.org/> [Último acceso: 01/03/2016]

[61] “FPDF Library. PDF Generator” <http://www.fpdf.org/> [Último acceso: 02/03/2016]

## ANEXO I: BASE DE DATOS

**\*Primera vez que se acceda a la herramienta\***

```
mysql> create database PLANCO;
mysql> create user "tfg"@"localhost" identified by "contraseña";
mysql> grant all on PLANCO.* to "tfg"@"localhost";
```

**\*Accediendo con el usuario creado\***

```
mysql> use PLANCO;
mysql> create table estados (id int, alistamiento int, estado varchar(128),
primary key (id)) engine=InnoDB;
mysql> create table subcontroles (id int, control varchar(128), subcontrol
varchar(128), primary key(id)) engine=InnoDB;
mysql> create table dotacion (id varchar(128), nombre varchar(128), rango
varchar(128), antigüedad date, especialidad varchar(128), id_estado int,
id_asignacion tinyint(1), vigilancia int, primary key(id, id_estado))
engine=InnoDB;
mysql> create table puestos (id varchar(128) auto_increment, puesto
varchar(128), id_estado int, id_dotacion int, id_subcontrol int, primary
key(id), foreign key(id_estado) references estados(id), foreign
key(id_dotacion) references dotacion(id), foreign key(id_subcontrol)
references subcontroles(id)) engine=InnoDB;
```

**\*Ejemplo de introducción de datos\***

```
mysql> insert into estados (id, alistamiento, estado) values
(1,1,'Zafarrancho de Combate')
```

## ANEXO II: INDEX.PHP

```
<?php
session_name("TFG");
session_start();
unset($_SESSION["array_estados"]);
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Plan de Combate Web</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 table, tr, th {background-color: #f0ffff;border:1px solid
black;border-collapse:collapse;width:50%;margin:auto;
 height:50px;}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 </style>
 </head>
 <body>

 <h1><u>Plan de Combate WEB</u></h1>

 <table>
 <tr>
 <th>Plan de Combate</th>
 </tr>
 <tr>
 <th>Embarcar Nuevo
Personal</th>
 </tr>
 <tr>
 <th>Desembarcar Personal</th>
 </tr>
 <tr>
 <th>Generar Archivo PDF</th>
 </tr>
 </table>
 </body>
</html>
```

## ANEXO III: VISTA1.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Plan de Combate</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 h2 {color:white;text-align:center;}
 h3 {color:white;text-align:left;}
 table, tr, th {background-color: #f0ffff;border:1px solid
black;border-collapse:collapse}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 div#d00 { background-color: #f0ffff; padding:0px;
width:10%; border-radius:5px;}
 div#d01 { background-color: #f0ffff; padding:5px;
width:43%; border-radius:5px;}
 div#d02 {overflow-x:auto;}
 div#d03 { background-color: #f0ffff; padding:0px;
width:17%; border-radius:5px;}
 div#d04 { background-color: #f0ffff; padding:0px;
width:12%; border-radius:5px;}
 </style>
 </head>
 <body>
 <div id='d00'><p>Volver a
Inicio</p></div>
 <h1><u>Seleccionar Grado de Alistamiento</u></h1>
 <div id='d03'>
 <p>Vaciar Todos los
Puestos</p>
 </div>
 <table style="width:50%; margin:auto; height: 50px;">
 <tr>
 <th>Zafarrancho de
Combate</th><th>Babor y Estribor de
Guardia</th><th>3
Vigilancias</th>
 </tr>
 </table>


```

```

<?php
if (isset($_SESSION["array_estados"])) {
 $estados = $_SESSION["array_estados"];
 if (count($estados) > 0) {
 echo "<h2>Seleccionar Situacion Operativa</h2>";
 echo "<div id='d01'><form method='post'
action='control.php'>AÑadir situacion con nombre:<input type='text'
name='situacion' required><input type='hidden' name='accion'
value='nueva_situacion'><input type='submit' name='enviar'
value='AÑadir'></form><form method='post' action='control.php'>Borrar
situacion con nombre:<input type='text' name='situacion' required><input
type='hidden' name='accion' value='borrar_situacion'><input type='submit'
name='enviar' value='Borrar'></form></div>";
 echo "<div id='d02'><table style='width:100%;
height:25px;'>";
 echo "<tr>";
 foreach ($estados as $key => $value) {
 echo "<th>$value</th>";
 }
 echo "</tr>";
 echo "</table></div>";
 } else {
 echo "<h2>No hay situaciones</h2></br>";
 }
}
?>

<?php
$accion = $_REQUEST["accion"];
if ($accion == "puestos") {
 $alistamiento = $_SESSION["array_alistamiento"];
 if (isset($_SESSION["array_puestos"])) {
 $puestos = $_SESSION["array_puestos"];
 $alistamiento = $_SESSION["array_alistamiento"];
 $estado = $_SESSION["estado"];
 $nombres = $_SESSION["nombres"];
 $nombredotacion = $_SESSION["nombredotacion"];
 $longitud = count($nombres);
 for ($i=0; $i<$longitud; $i++) {

 $puestos[$i]["id_subcontrol"]=$nombres[$i]["subcontrol"];
 }
 $longitud = count($nombredotacion);
 for ($i=0; $i<$longitud; $i++) {

 $puestos[$i]["id_dotacion"]=$nombredotacion[$i]["nombre"];
 }
 if ($alistamiento == "3") {
 echo "<table style='width:40%; height:25px;'>";
 echo "<tr><th><a
href='control.php?accion=listar_vigilancia&vigilancia=1&estado=$estado&vi
g=1'>1ª Vigilancia</th><th><a
href='control.php?accion=listar_vigilancia&vigilancia=2&estado=$estado&vi
g=2'>2ª Vigilancia</th><th><a

```

```
href='control.php?accion=listar_vigilancia&vigilancia=3^&estado=$estado&vi
g=3'>3^ Vigilancia</th></tr>";
 echo "</table>";
 } else {
 if (count($puestos) > 0) {
 echo "<div id='d04'><p>Gestionar
Puestos</p></div>";
 echo "<table style='width:100%; height:25px;'>";
 echo
"<tr><th>Puesto</th><th>Subcontrol</th><th>Asignado a</th></tr>";
 foreach ($puestos as $row) {
 if($row["id_dotacion"]!=NULL) {
 echo
"<tr><th>".$row["puesto"]."</th><th>".$row["id_subcontrol"]."</th><th>".$ro
w["id_dotacion"]."</th><th><a
href='control.php?accion=vaciar_puesto&id_puesto=".$row["id"]."&id_dotacion
=".$row["id_dotacion"]."'">Vaciar Puesto</th></tr>";
 } else {
 echo
"<tr><th>".$row["puesto"]."</th><th>".$row["id_subcontrol"]."</th><th>".$ro
w["id_dotacion"]."</th><th>Asign
ar Puesto</th></tr>";
 }
 }
 echo "</table>";
 unset($_SESSION["array_puestos"]);
 }
 }
 unset($_SESSION["array_puestos"]);
}
if (!isset($puestos)) {
 if ($alistamiento == "3") {
 echo "<table style='width:40%; height:25px;'>";
 echo "<tr><th><a
href='control.php?accion=listar_vigilancia&vigilancia=1^&estado=$estado&vi
g=1'>1^ Vigilancia</th><th><a
href='control.php?accion=listar_vigilancia&vigilancia=2^&estado=$estado&vi
g=2'>2^ Vigilancia</th><th><a
href='control.php?accion=listar_vigilancia&vigilancia=3^&estado=$estado&vi
g=3'>3^ Vigilancia</th></tr>";
 echo "</table>";
 } else {
 $estado = $_SESSION["estado"];
 echo "<div id='d04'><p>Gestionar
Puestos</p></div>";
 echo "<h3>No se han encontrado puestos</h3></br>";
 }
}
?>

<?php
```

```

$accion = $_REQUEST["accion"];
if ($accion == "puestos_vig") {
$estado = $_SESSION["estado"];
if (isset($_SESSION["array_puestos_vigilancia"])) {
 $estado = $_SESSION["estado"];
 echo "<table style='width:40%; height:25px;'>";
 echo "<tr><th><a
href='control.php?accion=listar_vigilancia&vigilancia=1&estado=$estado&vi
g=1'>1ª Vigilancia</th><th><a
href='control.php?accion=listar_vigilancia&vigilancia=2&estado=$estado&vi
g=2'>2ª Vigilancia</th><th><a
href='control.php?accion=listar_vigilancia&vigilancia=3&estado=$estado&vi
g=3'>3ª Vigilancia</th></tr>";
 echo "</table>";
 $puestos_vigilancia =
$_SESSION["array_puestos_vigilancia"];
 $nombres = $_SESSION["nombres"];
 $nombredotacion = $_SESSION["nombredotacion"];
 $longitud = count($nombres);
 for ($i=0; $i<=$longitud; $i++) {

 $puestos_vigilancia[$i]["id_subcontrol"]=$nombres[$i]["subcontrol"];
 }
 $longitud = count($nombredotacion);
 for ($i=0; $i<$longitud; $i++) {

 $puestos_vigilancia[$i]["id_dotacion"]=$nombredotacion[$i]["nombre"];
 }
 if (count($puestos_vigilancia) > 0) {
 echo "<div id='d04'><p>Gesti
onar Puestos</p></div>";
 echo "<table style='width:100%; height:25px;'>";
 echo
"<tr><th>Puesto</th><th>Subcontrol</th><th>Asignado a</th></tr>";
 foreach ($puestos_vigilancia as $row) {
 if($row["id_dotacion"]!=NULL) {
 echo
"<tr><th>".$row["puesto"]."</th><th>".$row["id_subcontrol"]."</th><th>".$ro
w["id_dotacion"]."</th><th>Vaciar
Puesto</th></tr>";
 } else {
 echo
"<tr><th>".$row["puesto"]."</th><th>".$row["id_subcontrol"]."</th><th>".$ro
w["id_dotacion"]."</th><th><a
href='control.php?accion=listar_disponibles_vigilancia&id_puesto=".$row["id
"]."'">Asignar Puesto</th></tr>";
 }
 }
 echo "</table>";
 } else {
 echo "<h3>No se han encontrado puestos</h3></br>";
 }
 unset($_SESSION["array_puestos_vigilancia"]);
 } else {

```

```
 echo "<div id='d04'><p>Gesti
onar Puestos</p></div>";
 echo "<h3>No se han encontrado puestos</h3></br>";
 }
}
?>

<?php
if (isset($_SESSION["disponibles"])) {
 $disponibles = $_SESSION["disponibles"];
 if (count($disponibles) > 0) {
 echo "<table style='width:50%; height:25px;'>";
 echo "<tr><th>Disponibles</th></tr>";
 foreach ($disponibles as $row) {
 echo "<tr><th>".$row["rango"]."
".$row["nombre"]."</th><th>Asignar</th></tr>";
 }
 echo "</table>";
 } else {
 echo "No hay nadie disponible";
 }
 unset($_SESSION["disponibles"]);
}
?>

<?php
$accion = $_REQUEST["accion"];

if (isset($accion)) {
 if ($accion == "asignado") {
 echo "<h3>Puesto Asignado</h3>";
 }

 if ($accion == "vaciado") {
 echo "<h3>Puesto Vaciado</h3>";
 }

 if ($accion == "todos") {
 echo "<h3>Puestos Vaciados</h3>";
 }
}
?>
</body>
</html>
```

## ANEXO IV: VISTA2.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Gestionar Puestos</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 h2 {color:white;text-align:center;}
 h3 {color:white;text-align:left;}
 table, tr, th {background-color: #f0ffff;border:1px solid
black;border-collapse:collapse}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 div#d01 { background-color: #f0ffff; padding:5px;
width:25%; border-radius:5px;}
 </style>
 </head>
 <body>
 <p>Volver a Inicio</p>
 <p>Volver al Plan de Combate</p>
 <h1><u>Gestionar Puestos</u></h1>
 <?php
 $accion = $_REQUEST["accion"];
 $alistamiento = $_SESSION["array_alistamiento"];
 if (isset($accion)) {
 if ($accion == "añadir_puesto"){
 if ($alistamiento == "3") {
 echo "<div id='d01'>
 <form method='post'
action='control.php'>
 Puesto Nuevo:

 <input type='text' name='puesto'
required>

 <input type='hidden' name='accion'
value='nuevo_puesto_vigilancia'>
 <input type='submit' name='enviar'
value='Añadir'>
 </form>
 </div>";
 } else {
 echo "<div id='d01'>
 <form method='post'
action='control.php'>
```

```
Puesto Nuevo:

<input type='text' name='puesto'
required>

<input type='hidden' name='accion'
value='nuevo_puesto'>
<input type='submit' name='enviar'
value='AÑadir'>

</form>
</div>";
 }
}
if ($accion == "borrado") {
 echo "<h3>Puesto Borrado</h3>";
}
if ($accion == "añadido") {
 echo "<h3>Puesto Añadido</h3>";
}
} else {
 if (isset($_SESSION["array_puestos"])) {
 $puestos = $_SESSION["array_puestos"];
 $nombres = $_SESSION["nombres"];
 $longitud = count($nombres);
 for ($i=0; $i<=$longitud; $i++) {

 $puestos[$i]["id_subcontrol"]=$nombres[$i]["subcontrol"];
 }
 if (count($puestos) > 0) {
 echo "<p>Añadir Puesto</p>";
 echo "<table style='width:70%;
height:25px;' margin:auto;>";
 echo
 "<tr><th>Puesto</th><th>Subcontrol</th></tr>";
 foreach ($puestos as $row) {
 echo
 "<tr><th>".$row["puesto"]."</th><th>".$row["id_subcontrol"]."</th><th>Borrar
Puesto</th></tr>";
 }
 }
 unset($_SESSION["array_puestos"]);
 } else {
 echo "<p>Añadir Puesto</p>";
 }
}
?>
</body>
</html>
```

## ANEXO V: VISTA3.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Embarcar Persona</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 h2 {color:white;text-align:center;}
 h3 {color:white;text-align:center;}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 div { background-color: #f0ffff; padding:20px; width:30%;
border-radius:5px; margin: auto; }
 </style>
 </head>
 <body>
 <p>Volver a Inicio</p>
 <h1><u>Nueva Persona</u></h1>
 <?php
 $accion = $_REQUEST["accion"];

 if (isset($accion)) {
 if ($accion == "exito") {
 echo "<h2>Introducido Correctamente</h2>";
 echo "<p>Seguir
Introduciendo</p>";
 }
 } else {
 echo "
 <div>
 <form method='post' action='control.php'>
 Nombre completo:

 <input type='text' name='nombre' required>

 TIM:

 <input type='text' name='id' required>

 Rango:

 <input type='text' name='rango' required>

 Antigüedad:

 <input type='date' name='antigüedad' placeholder='AAAA-MM-
DD'>


```

```
 Especialidad:

 <input type='text' name='especialidad'>

 Vigilancia:

 <input type='number' name='vigilancia'>

 <input type='hidden' name='accion' value='nueva_persona'>
 <input type='submit' name='enviar' value='Introducir
Datos'>
 </form>
 </div>";
 }
 ?>
</body>
</html>
```

## ANEXO VI: VISTA4.PHP

```
<?php
session_name("TFG");
session_start();
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Desembarcar Persona</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 h2 {color:white;text-align:center;}
 h3 {color:white;text-align:center;}
 table, tr, th {background-color: #f0ffff;border:1px solid
black;border-collapse:collapse; height: 25px;}
 table {width:40%; margin:auto; height: 50px;}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 </style>
 </head>
 <body>
 <p>Volver a Inicio</p>
 <h1><u>Seleccionar Persona</u></h1>

 <?php

 if (isset($_SESSION["array_desembarcar"])) {
 $desembarcar = $_SESSION["array_desembarcar"];
 if (count($desembarcar) > 0) {
 echo "<table>";
 echo "<tr><th>Dotacion</th><th>Desembarcar
Todos<th></tr>";
 foreach ($desembarcar as $row) {
 echo "<tr><th>".$row["rango"]."
".$row["nombre"]."</th><th>Desembarcar</th></tr>";
 }
 echo "</table>";
 } else {
 echo "<h3>No hay dotacion</h3>";
 }
 }
 unset($_SESSION["array_desembarcar"]);
 }

 if (isset($_SESSION["eliminar_persona"])) {
 $eliminar_persona = $_SESSION["eliminar_persona"];
```

```
 if (count($eliminar_persona) > 0) {
 echo "<table>";
 echo "<tr><th>Dotacion</th></tr>";
 foreach ($eliminar_persona as $row) {
 echo "<tr><th>".$row["rango"]."
".$row["nombre"]."</th><th>Desembarcar</th></tr>";
 }
 echo "</table>";
 echo "
<h3>Proceso realizado con exito.</h3>";
 } else {
 echo "Error al desembarcar la persona";
 }
 unset($_SESSION["eliminar_persona"]);
 }
 $accion = $_REQUEST["accion"];
 if (isset($accion)) {
 if ($accion == "todos"){
 echo "<h3>Dotacion Eliminada</h3>";
 }
 }
 ?>
</body>
</html>
```

## ANEXO VII: VISTA5.PHP

```
<?php
session_name("TFG");
session_start();
unset($_SESSION["array_estados"]);
?>
<!DOCTYPE html>
<html lang="en">
 <head>
 <link href="Logo.ico" type="image/x-icon" rel="shortcut icon"/>
 <meta charset="utf-8">
 <title>Generar PDF</title>
 <style>
 body {background-color: #00111a;}
 h1 {color:white;text-align:center;}
 h3 {color:white;text-align:center;}
 table, tr, th {background-color: #f0ffff;border:1px solid
black;border-collapse:collapse;width:30%;margin:auto;
 height:25px;}
 a:link {color:blue;}
 a:visited {color:blue;}
 a:hover {color:green;}
 a:active {color:green;}
 </style>
 </head>
 <body>
 <p>Volver a Inicio</p>
 <h1><u>Generar Archivo PDF</u></h1>

 <?php
 $accion = $_REQUEST["accion"];

 if (isset($accion)) {
 if ($accion == "fichas") {
 $listar_dotacion = $_SESSION["listar_dotacion"];
 if (count($listar_dotacion) > 0) {
 echo "<table>";
 echo "<tr><th>Dotacion</th></tr>";
 foreach ($listar_dotacion as $row) {
 echo "<tr><th>".$row["rango"]."
".$row["nombre"]."</th></tr>";
 }
 echo "</table>";
 }
 unset($_SESSION["listar_dotacion"]);
 }

 if ($accion == "situacion") {
 echo "<table>";
 echo "<tr>";
```

```
 echo "<th>Zafarrancho de
Combate</th></tr><tr><th>Babor y Estribor de
Guardia</th></tr><tr><th>3
Vigilancias</th>";
 echo "</tr>";
 echo "</table>";
 }

 if ($accion == "situaciones") {
 $alistamiento = $_SESSION["alistamiento"];
 $situaciones = $_SESSION["situaciones"];
 if ($alistamiento == "3") {
 if (count($situaciones) > 0) {
 echo "<table>";
 echo "<tr><th>Situaciones</th></tr>";
 foreach ($situaciones as $row) {
 echo "<tr><th>.$r
ow["estado"]."</th></tr>";
 }
 echo "</table>";
 } else {
 echo "<h3>No hay situaciones<h3>";
 }
 } else {
 if (count($situaciones) > 0) {
 echo "<table>";
 echo "<tr><th>Situaciones</th></tr>";
 foreach ($situaciones as $row) {
 echo "<tr><th><a
href='generarpdf.php?id=".$row["id"]."&estado=".$row["estado"]."&alistamien
to=$alistamiento&accion=situacion'">.$row["estado"]."</th></tr>";
 }
 echo "</table>";
 } else {
 echo "<h3>No hay situaciones<h3>";
 }
 }
 }

 if ($accion == "3") {
 $id = $_REQUEST["id"];
 $estado = $_REQUEST["estado"];
 echo "<table><tr><th><a
href='generarpdf.php?id=$id&estado=$estado&alistamiento=3&vigilancia=1âª&ac
cion=situacion'">1âª Vigilancia</th></tr><tr><th><a
href='generarpdf.php?id=$id&estado=$estado&alistamiento=3&vigilancia=2âª&ac
cion=situacion'">2âª Vigilancia</th></tr><tr><th><a
href='generarpdf.php?id=$id&estado=$estado&alistamiento=3&vigilancia=3âª&ac
cion=situacion'">3âª Vigilancia</th></tr></table>";
 }

 if ($accion == "control") {
```

```

 echo "<table><tr><th>Control de
Operaciones</th></tr><tr><th>Control de
Plataforma</th></tr>";
 }

 if ($accion == "subcontroles") {
 $control = $_SESSION["control"];
 $subcontroles = $_SESSION["subcontroles"];
 if (count($subcontroles) > 0) {
 echo "<table><tr><th>$control</th></tr>";
 foreach ($subcontroles as $row) {
 echo "<tr><th>".$row["subcontro
l"]."</th></tr>";
 }
 }
 }

 } else {
 echo "<table>
<tr>
<th>Listado de Dotacion</th>
</tr>
<tr>
<th>Situación Operativa</th>
</tr>
<tr>
<th>Control
o Subcontrol</th>
</tr>
<tr>
<th>Ficha
Personal</th>
</tr>
</table>";
 }
 ?>
</body>
</html>

```

## ANEXO VIII: GENERARPDF.PHP

```
<?php
session_name("TFG");
session_start();
include('fpdf.php');
$servername = "localhost";
$username = "tfg";
$password = "joaquin-01";
$dbname = "PLANCO";
$datos = array();
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
}
mysqli_set_charset($conn, "utf8");
$action = $_REQUEST["accion"];
if (isset($action)) {
 if ($action == "dotacion") {
 $sql = "SELECT nombre, rango FROM dotacion WHERE id_estado=0";
 $result = mysqli_query($conn, $sql);
 $pdf = new FPDF('P','mm','A4');
 $pdf->SetAutoPageBreak(true);
 $pdf->AddPage();
 $pdf->SetTitle('Dotacion');
 $pdf->SetAuthor('Joaquin Garat');
 $pdf->SetSubject('TFG');
 $pdf->SetFont('Arial','B',16);
 $pdf->Image('Logo_Armada.png',175,5,20,40,'PNG');
 $pdf->Write(5,utf8_decode('Dotaci3n'));
 $pdf->Ln(10);
 $pdf->SetFontSize(12);
 $pdf->SetTextColor(0,0,0);
 $pdf->SetFillColor(200);
 $pdf->Cell(80,7,'Nombre',1,0,'C',1);
 $pdf->Cell(40,7,'Rango',1,1,'C',1);
 $pdf->SetFont('Arial','',12);
 $pdf->SetTextColor(0,0,0);
 while ($row = mysqli_fetch_array($result)) {
 $pdf->Cell(80,7,utf8_decode($row['nombre']),1);
 $pdf->Cell(40,7,utf8_decode($row['rango']),1,1,'R');
 }
 $pdf->SetXY(10,-10);
 $pdf->SetFontSize(10);
 $pdf->Cell(0,0,utf8_decode('P3gina ').$pdf->PageNo(),0,0,'R');
 $pdf->Output();
 }
 if ($action == "situacion") {
 $alistamiento = $_REQUEST["alistamiento"];
 $puestos = array();
 if ($alistamiento == "1") {
 $GA = "Zafarrancho de Combate";
 }
 if ($alistamiento == "2") {
 $GA = "Babor y Estribor de Guardia";
 }
 }
}
```

```
}
if ($alistamiento == "3") {
 $GA = "3 Vigilancia";
}
$estado = $_REQUEST["estado"];
$id = $_REQUEST["id"];
$vigilancia = $_REQUEST["vigilancia"];
$sql = "SELECT puesto, id_subcontrol, id_dotacion FROM puestos
WHERE id_estado=$id AND puesto LIKE '%$vigilancia'";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $puestos[] = $row;
 }
}
$subcontroles = array_column($puestos, "id_subcontrol");
$i = 0;
foreach ($subcontroles as $row) {
 $sql = "SELECT control, subcontrol FROM subcontroles WHERE
id=$row";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $subcontrol[$i] = $row;
 }
 }
 $i++;
}
$longitud = count($subcontroles);
for ($i = 0; $i < $longitud; $i++) {
 $puestos[$i]["id_subcontrol"] = $subcontrol[$i]["subcontrol"];
}
$dotacion = array_column($puestos, "id_dotacion");
$i = 0;
foreach ($dotacion as $row) {
 $sql = "SELECT id, nombre FROM dotacion WHERE id='$row' AND
id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $nombre[$i] = $row;
 }
 }
 $i++;
}
$longitud = count($dotacion);
for ($i = 0; $i < $longitud; $i++) {
 $puestos[$i]["id_dotacion"] = $nombre[$i]["nombre"];
}
$pdf = new FPDF ('P', 'mm', 'A4');
$pdf->SetAutoPageBreak(true);
$pdf->AddPage();
$pdf->SetTitle('Situacion');
$pdf->SetAuthor('Joaquin Garat');
$pdf->SetSubject('TFG');
```

```
$pdf->SetFont('Arial','B',16);
$pdf->Image('Logo_Armada.png',175,5,5,10,'PNG');
if ($alistamiento == "3") {
 if ($GA == $estado) {
 $pdf->Write(5,utf8_decode($estado));
 $pdf->Write(5,utf8_decode(' ('));
 $pdf->Write(5,utf8_decode($vigilancia));
 $pdf->Write(5,utf8_decode(' Vigilancia')));
 } else {
 $pdf->Write(5,utf8_decode($estado));
 $pdf->Write(5,utf8_decode(' a '));
 $pdf->Write(5,utf8_decode($GA));
 $pdf->Write(5,utf8_decode(' ('));
 $pdf->Write(5,utf8_decode($vigilancia));
 $pdf->Write(5,utf8_decode(' Vigilancia')));
 }
} else {
 if ($GA == $estado) {
 $pdf->Write(5,utf8_decode($estado));
 } else {
 $pdf->Write(5,utf8_decode($estado));
 $pdf->Write(5,utf8_decode(' en '));
 $pdf->Write(5,utf8_decode($GA));
 }
}
$pdf->Ln(10);
$pdf->SetFontSize(12);
$pdf->SetTextColor(0,0,0);
$pdf->SetFillColor(200);
$pdf->Cell(90,7,'Puesto',1,0,'C',1);
$pdf->Cell(40,7,'Subcontrol',1,0,'C',1);
$pdf->Cell(50,7,'Asignado a',1,1,'C',1);
$pdf->SetFont('Arial','',12);
$pdf->SetTextColor(0,0,0);
foreach ($puestos as $row) {
 $pdf->Cell(90,7,utf8_decode($row['puesto']),1);
 $pdf->Cell(40,7,utf8_decode($row['id_subcontrol']),1);
 $pdf->Cell(50,7,utf8_decode($row['id_dotacion']),1,1);
}
$pdf->SetXY(10,-10);
$pdf->SetFontSize(10);
$pdf->Cell(0,0,utf8_decode('Página ').$pdf->PageNo(),0,0,'R');
$pdf->Output();
}
if ($accion == "subcontrol") {
 $control = $_SESSION["control"];
 $id = $_REQUEST["id"];
 $sql = "SELECT puesto, id_estado, id_dotacion FROM puestos WHERE
id_subcontrol='$id'";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $puestos[]=$row;
 }
 }
 $estados = array_column($puestos, "id_estado");
 $i=0;
```

```
foreach ($estados as $row) {
 $sql = "SELECT id, estado FROM estados WHERE id=$row";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $estado[$i] = $row;
 }
 }
 $i++;
}
$longitud = count($estados);
for ($i=0; $i<$longitud; $i++) {
 $puestos[$i]["id_estado"]=$estado[$i]["estado"];
}
$dotacion = array_column($puestos, "id_dotacion");
$i=0;
foreach ($dotacion as $row) {
 $sql = "SELECT id, nombre FROM dotacion WHERE id='$row' AND
id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $nombre[$i] = $row;
 }
 }
 $i++;
}
$longitud = count($dotacion);
for ($i=0; $i<$longitud; $i++) {
 $puestos[$i]["id_dotacion"]=$nombre[$i]["nombre"];
}
$sql2 = "SELECT subcontrol FROM subcontroles WHERE id=$id";
$result2 = mysqli_query($conn, $sql2);
$pdf = new FPDF ('P','mm','A4');
$pdf->SetAutoPageBreak(true);
$pdf->AddPage();
$pdf->SetTitle('Subcontrol');
$pdf->SetAuthor('Joaquin Garat');
$pdf->SetSubject('TFG');
$pdf->SetFont('Arial','B',16);
$pdf->Image('Logo_Armada.png',175,5,5,10,'PNG');
$pdf->Write(5,'Control de ');
$pdf->Write(5,utf8_decode($control));
$pdf->Write(5,' - ');
while ($row = mysqli_fetch_array($result2)) {
 $pdf->Write(5,utf8_decode($row["subcontrol"]));
}
$pdf->Ln(10);
$pdf->SetFontSize(12);
$pdf->SetTextColor(0,0,0);
$pdf->SetFillColor(200);
$pdf->Cell(90,7,'Situacion Operativa',1,0,'C',1);
$pdf->Cell(40,7,'Puesto',1,0,'C',1);
$pdf->Cell(50,7,'Asignado a',1,1,'C',1);
foreach ($puestos as $row) {
 $pdf->Cell(90,7,utf8_decode($row['id_estado']),1);
```

```
$pdf->Cell(40,7,utf8_decode($row['puesto']),1);
$pdf->Cell(50,7,utf8_decode($row['id_dotacion']),1,1);
}
$pdf->SetXY(10,-10);
$pdf->SetFontSize(10);
$pdf->Cell(0,0,utf8_decode('Página ').$pdf->PageNo(),0,0,'R');
$pdf->Output();
}

if ($accion == "ficha") {
 $id = $_REQUEST["id"];
 $sql = "SELECT puesto, id_estado, id_dotacion FROM puestos WHERE
id_dotacion='$id'";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $puestos[]=$row;
 }
 }
 $estados = array_column($puestos, "id_estado");
 $i=0;
 foreach ($estados as $row) {
 $sql = "SELECT id, estado FROM estados WHERE id=$row";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $estado[$i] = $row;
 }
 }
 $i++;
 }
 $longitud = count($estados);
 for ($i=0; $i<$longitud; $i++) {
 $puestos[$i]["id_estado"]=$estado[$i]["estado"];
 }
 $sql2 = "SELECT nombre FROM dotacion WHERE id='$id' AND
id_estado=0";
 $result2 = mysqli_query($conn, $sql2);
 $pdf = new FPDF ('P','mm','A4');
 $pdf->SetAutoPageBreak(true);
 $pdf->AddPage();
 $pdf->SetTitle('Fichas de Personal');
 $pdf->SetAuthor('Joaquin Garat');
 $pdf->SetSubject('TFG');
 $pdf->SetFont('Arial','B',16);
 $pdf->Image('Logo_Armada.png',175,5,20,40,'PNG');
 $pdf->Write(5,'Fichas de Personal');
 $pdf->Ln(10);
 $pdf->SetFontSize(12);
 $pdf->SetTextColor(0,0,0);
 $pdf->SetFillColor(200);
 while ($row = mysqli_fetch_array($result2)){
 $pdf->Cell(140,7,utf8_decode($row['nombre']),1,1,'C',1);
 }
 $pdf->Cell(60,7,utf8_decode('Estado'),1,0,'C',1);
 $pdf->Cell(80,7,utf8_decode('Puesto'),1,1,'C',1);
 $pdf->SetFont('Arial','',12);
```

```
$pdf->SetTextColor(0,0,0);
foreach ($puestos as $row) {
 $pdf->Cell(60,7,utf8_decode($row['id_estado']),1);
 $pdf->Cell(80,7,utf8_decode($row['puesto']),1,1,'R');
}
$pdf->SetXY(10,-10);
$pdf->SetFontSize(10);
$pdf->Cell(0,0,utf8_decode('Página ').$pdf->PageNo(),0,0,'R');
$pdf->Output();
}
mysqli_close($conn);
}
?>
```

## ANEXO IX: CONTROL.PHP

```
<?php
session_name("TFG");
session_start();

function redirect($url, $statusCode = 303) {
 header('Location: ' . $url, true, $statusCode);
 die();
}

require ("modelo.php");

$accion = $_REQUEST["accion"];

if (isset($accion)) {

 if ($accion == "listar_estados") {
 $alistamiento = $_REQUEST["alistamiento"];
 $estados = listar_estados($alistamiento);
 $_SESSION["array_alistamiento"] = $alistamiento;
 $_SESSION["array_estados"] = $estados;
 redirect("vista1.php");
 }

 if ($accion == "nueva_situacion") {
 $alistamiento = $_SESSION["array_alistamiento"];
 $situacion = $_REQUEST["situacion"];
 $nueva_situacion = nueva_situacion($alistamiento, $situacion);

 redirect("control.php?accion=listar_estados&alistamiento=$alistamient
o");
 }

 if ($accion == "borrar_situacion") {
 $alistamiento = $_SESSION["array_alistamiento"];
 $situacion = $_REQUEST["situacion"];
 $borrar_situacion = borrar_situacion($alistamiento, $situacion);

 redirect("control.php?accion=listar_estados&alistamiento=$alistamient
o");
 }

 if ($accion == "listar_puestos") {
 $estado = $_REQUEST["estado"];
 $puestos = listar_puestos($estado);
 $_SESSION["estado"] = $estado;
 $_SESSION["array_puestos"] = $puestos;
 $subcontroles = array_column($puestos, "id_subcontrol");
 $nombres = nombressub($subcontroles);
 $_SESSION["nombres"] = $nombres;
 $personas = array_column($puestos, "id_dotacion");
 $nombredotacion = nombredotacion($personas);
 $_SESSION["nombredotacion"] = $nombredotacion;
 redirect("vista1.php?accion=puestos");
 }
}
```

```
}

if ($accion == "listar_vigilancia") {
 $vigilancia = $_REQUEST["vigilancia"];
 $_SESSION["vigilancia"] = $vigilancia;
 $vig = $_REQUEST["vig"];
 $_SESSION["vig"] = $vig;
 $estado = $_REQUEST["estado"];
 $puestos_vigilancia = listar_vigilancia($vigilancia,$estado);
 $_SESSION["array_puestos_vigilancia"] = $puestos_vigilancia;
 $subcontroles =
array_column($puestos_vigilancia,"id_subcontrol");
 $nombres = nombressub($subcontroles);
 $_SESSION["nombres"] = $nombres;
 $personas = array_column($puestos_vigilancia,"id_dotacion");
 $nombredotacion = nombredotacion($personas);
 $_SESSION["nombredotacion"] = $nombredotacion;
 redirect("vista1.php?accion=puestos_vig");
}

if ($accion == "gestionar_puestos") {
 $estado = $_REQUEST["estado"];
 $puestos = listar_puestos($estado);
 $_SESSION["estado"] = $estado;
 $_SESSION["array_puestos"] = $puestos;
 $subcontroles = array_column($puestos,"id_subcontrol");
 $nombres = nombressub($subcontroles);
 $_SESSION["nombres"] = $nombres;
 redirect("vista2.php");
}

if ($accion == "gestionar_puestos_vigilancia") {
 $estado = $_REQUEST["estado"];
 $vigilancia = $_SESSION["vigilancia"];
 $puestos = listar_vigilancia($vigilancia,$estado);
 $_SESSION["estado"] = $estado;
 $_SESSION["array_puestos"] = $puestos;
 $subcontroles = array_column($puestos,"id_subcontrol");
 $nombres = nombressub($subcontroles);
 $_SESSION["nombres"] = $nombres;
 redirect("vista2.php");
}

if ($accion == "nuevo_puesto") {
 $estado = $_SESSION["estado"];
 $puesto = $_REQUEST["puesto"];
 $nuevo_puesto = nuevo_puesto($puesto, $estado);
 redirect("vista2.php?accion=añadir");
}

if ($accion == "nuevo_puesto_vigilancia") {
 $vigilancia = $_SESSION["vigilancia"];
 $estado = $_SESSION["estado"];
 $puesto = $_REQUEST["puesto"];
 $nuevo_puesto = nuevo_puesto_vigilancia($puesto, $vigilancia,
$estado);
```

```
 redirect("vista2.php?accion=aÃ±adido");
 }

 if ($accion == "borrar_puesto") {
 $id = $_REQUEST["id"];
 $borrar_puesto = borrar_puesto($id);
 redirect("vista2.php?accion=borrado");
 }

 if ($accion == "listar_disponibles") {
 $id_puesto = $_REQUEST["id_puesto"];
 $_SESSION["id_puesto"] = $id_puesto;
 $estado = $_SESSION["estado"];
 $disponibles = listar_disponibles($estado);
 $_SESSION["disponibles"] = $disponibles;
 redirect("vista1.php");
 }

 if ($accion == "listar_disponibles_vigilancia") {
 $id_puesto = $_REQUEST["id_puesto"];
 $_SESSION["id_puesto"] = $id_puesto;
 $vig = $_SESSION["vig"];
 $estado = $_SESSION["estado"];
 $disponibles = listar_disponibles_vigilancia($estado, $vig);
 $_SESSION["disponibles"] = $disponibles;
 redirect("vista1.php");
 }

 if ($accion == "asignar") {
 $id = $_REQUEST["id"];
 $id_puesto = $_SESSION["id_puesto"];
 $estado = $_SESSION["estado"];
 $asignar = asignar_puesto($id, $id_puesto, $estado);
 redirect("vista1.php?accion=asignado");
 }

 if ($accion == "vaciar_puesto") {
 $id_puesto = $_REQUEST["id_puesto"];
 $id_dotacion = $_REQUEST["id_dotacion"];
 $estado = $_SESSION["estado"];
 $vaciar = vaciar_puesto($id_puesto, $id_dotacion, $estado);
 redirect("vista1.php?accion=vaciado");
 }

 if ($accion == "vaciar_todos") {
 $vaciar_todos = vaciar_todos();
 redirect("vista1.php?accion=todos");
 }

 if ($accion == "nueva_persona") {
 $nombre = $_REQUEST["nombre"];
 $id = $_REQUEST["id"];
 $rango = $_REQUEST["rango"];
 $antiguedad = $_REQUEST["antiguedad"];
 $especialidad = $_REQUEST["especialidad"];
 $vigilancia = $_REQUEST["vigilancia"];
```

```
$dotacion = nueva_persona($nombre, $id, $rango, $antiguedad,
$especialidad, $vigilancia);
 redirect("vista3.php?accion=exito");
}

if ($accion == "desembarcar") {
 $desembarcar = desembarcar();
 $_SESSION["array_desembarcar"] = $desembarcar;
 redirect("vista4.php");
}

if ($accion == "eliminar_persona") {
 $id = $_REQUEST["id"];
 $eliminar_persona = eliminar_persona($id);
 $_SESSION["eliminar_persona"] = $eliminar_persona;
 redirect("vista4.php");
}

if ($accion == "desembarcar_todos") {
 $desembarcar_todos = desembarcar_todos();
 redirect("vista4.php?accion=todos");
}

if ($accion == "situaciones") {
 $alistamiento = $_REQUEST["alistamiento"];
 $_SESSION["alistamiento"] = $alistamiento;
 $situaciones = situaciones($alistamiento);
 $_SESSION["situaciones"] = $situaciones;
 redirect("vista5.php?accion=situaciones");
}

if ($accion == "control") {
 $control = $_REQUEST["control"];
 $_SESSION["control"] = $control;
 $subcontroles = subcontroles($control);
 $_SESSION["subcontroles"] = $subcontroles;
 redirect ("vista5.php?accion=subcontroles");
}

if ($accion == "fichas") {
 $listar_dotacion = listar_dotacion();
 $_SESSION["listar_dotacion"] = $listar_dotacion;
 redirect("vista5.php?accion=fichas");
}
}
?>
```

## ANEXO X: MODELO.PHP

```
<?php
```

```
function listar_estados($alistamiento) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $estados = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, estado FROM estados WHERE
alistamiento=$alistamiento";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $estados[$row["id"]]=$row["estado"];
 }
 return $estados;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function nueva_situacion($alistamiento, $situacion) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $nueva_situacion = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "INSERT INTO estados (alistamiento, estado) VALUES (?, ?)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $alistamiento, $situacion);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function borrar_situacion($alistamiento, $situacion) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $borrar_situacion = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
```

```
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "DELETE FROM estados WHERE alistamiento=? AND estado=?";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $alistamiento, $situacion);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function listar_puestos($estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $puestos = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, puesto, id_subcontrol, id_dotacion FROM puestos
WHERE id_estado=$estado";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $puestos[]=$row;
 }
 return $puestos;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function listar_vigilancia($vigilancia,$estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $puestos_vigilancia = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, puesto, id_subcontrol, id_dotacion FROM puestos
WHERE puesto LIKE '%$vigilancia' AND id_estado=$estado";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $puestos_vigilancia[]=$row;
 }
 return $puestos_vigilancia;
 } else {
 return NULL;
 }
}
```

```
 }
 mysqli_close($conn);
}

function nombressub($subcontroles) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $nombres = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $i=0;
 foreach ($subcontroles as $row) {
 $sql = "SELECT control, subcontrol FROM subcontroles WHERE
id=$row";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $nombres[$i] = $row;
 }
 }
 $i++;
 }
 return $nombres;
 mysqli_close($conn);
}

function nombredotacion($personas) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $nombredotacion = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $i=0;
 foreach ($personas as $row) {
 $sql = "SELECT id, nombre FROM dotacion WHERE id='$row' AND
id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while ($row = mysqli_fetch_assoc($result)) {
 $nombredotacion[$i] = $row;
 }
 }
 $i++;
 }
 return $nombredotacion;
 mysqli_close($conn);
}
```

```
function nuevo_puesto($puesto, $estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $nueva_puesto = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "INSERT INTO puestos (puesto, id_estado) VALUES (?, ?)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $puesto, $estado);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function nuevo_puesto_vigilancia($puesto, $vigilancia, $estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $nueva_puesto_vigilancia = array();
 $puesto_vigilancia = $puesto." ".$vigilancia;
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "INSERT INTO puestos (puesto, id_estado) VALUES (?, ?)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $puesto_vigilancia, $estado);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function borrar_puesto($id) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $borrar_puesto = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "UPDATE puestos SET id_dotacion=NULL, id_subcontrol=NULL,
id_estado=NULL WHERE id=?";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "s", $id);
 $sql = "DELETE FROM puestos WHERE id=?";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "s", $id);
```

```
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
 }

function datos_persona($tim) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $persona = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT nombre, rango FROM dotacion WHERE id='$tim' AND
id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_array($result)) {
 $persona[]=$row;
 }
 return $persona;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function listar_disponibles($estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $disponibles = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, nombre, rango FROM dotacion WHERE
id_estado=$estado and id_asignacion=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $disponibles[] = $row;
 }
 return $disponibles;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function listar_disponibles_vigilancia($estado, $vig) {
 $servername = "localhost";
 $username = "tfg";
```

```
$password = "joaquin-01";
$dbname = "PLANCO";
$disponibles = array();
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
}
mysqli_set_charset($conn, "utf8");
$sql = "SELECT id, nombre, rango FROM dotacion WHERE
id_estado=$estado and id_asignacion=0 and vigilancia=$vig";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $disponibles[] = $row;
 }
 return $disponibles;
} else {
 return NULL;
}
mysqli_close($conn);
}

function asignar_puesto($id, $id_puesto, $estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $asignar = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "UPDATE puestos SET id_dotacion=? WHERE id=?";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $id, $id_puesto);
 mysqli_stmt_execute($stmt);
 $sql = "UPDATE dotacion SET id_asignacion=1 WHERE id=? and
id_estado=?";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ss", $id, $estado);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function vaciar_puesto($id_puesto, $id_dotacion, $estado) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $vaciar = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
```

```
$sql = "UPDATE puestos SET id_dotacion=NULL WHERE id=?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "s", $id_puesto);
mysqli_stmt_execute($stmt);
$sql = "UPDATE dotacion SET id_asignacion=0 WHERE nombre=? and
id_estado=?";
$stmt = mysqli_prepare($conn, $sql);
mysqli_stmt_bind_param($stmt, "ss", $id_dotacion, $estado);
mysqli_stmt_execute($stmt);
mysqli_close($conn);
}

function vaciar_todos() {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $vaciar_todos = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "UPDATE puestos SET id_dotacion=NULL";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_execute($stmt);
 $sql = "UPDATE dotacion SET id_asignacion=0";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
}

function nueva_persona($nombre, $id, $rango, $antiguedad, $especialidad,
$vigilancia) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $dotacion = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia) VALUES (?,?,?,?,?,?)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?,?,?,?,?,1)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
}
```

```
$sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 2)";
$stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 3)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 4)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 5)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 6)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 7)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 8)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 9)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 10)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 11)";
```

```
$stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 12)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 13)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 14)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 15)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 16)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 17)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 18)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 19)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 20)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
```

```
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 21)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 $sql = "INSERT INTO dotacion (id, nombre, rango, antiguedad,
especialidad, vigilancia, id_estado) VALUES (?, ?, ?, ?, ?, ?, 22)";
 $stmt = mysqli_prepare($conn, $sql);
 mysqli_stmt_bind_param($stmt, "ssssss", $id, $nombre, $rango,
$antiguedad, $especialidad, $vigilancia);
 mysqli_stmt_execute($stmt);
 mysqli_close($conn);
 }

function desembarcar() {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $desembarcar = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, nombre, rango FROM dotacion WHERE id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $desembarcar[] = $row;
 }
 return $desembarcar;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function eliminar_persona($id) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $eliminar_persona = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi3n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "UPDATE puestos SET id_dotacion=NULL WHERE id_dotacion='$id'";
 $result = mysqli_query($conn, $sql);
 $sql = "DELETE FROM dotacion WHERE id='$id'";
 $result = mysqli_query($conn, $sql);
 $sql = "SELECT id, nombre, rango FROM dotacion WHERE id_estado=0";
```

```
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $eliminar_persona[] = $row;
 }
 return $eliminar_persona;
} else {
 return NULL;
}
mysqli_close($conn);
}

function desembarcar_todos() {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $desembarcar_todos = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "UPDATE puestos SET id_dotacion=NULL";
 $result = mysqli_query($conn, $sql);
 $sql = "DELETE FROM dotacion";
 $result = mysqli_query($conn, $sql);
 mysqli_close($conn);
}

function situaciones($alistamiento) {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $situaciones = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexi n fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, estado FROM estados WHERE
alistamiento=$alistamiento";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $situaciones[] = $row;
 }
 return $situaciones;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}

function subcontroles($control) {
 $servername = "localhost";
```

```
$username = "tfg";
$password = "joaquin-01";
$dbname = "PLANCO";
$subcontroles = array();
$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
}
mysqli_set_charset($conn, "utf8");
$sql = "SELECT id, subcontrol FROM subcontroles WHERE
control='$control'";
$result = mysqli_query($conn, $sql);
if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $subcontroles[] = $row;
 }
 return $subcontroles;
} else {
 return NULL;
}
mysqli_close($conn);
}

function listar_dotacion() {
 $servername = "localhost";
 $username = "tfg";
 $password = "joaquin-01";
 $dbname = "PLANCO";
 $listar_dotacion = array();
 $conn = mysqli_connect($servername, $username, $password, $dbname);
 if (!$conn) {
 die("Conexión fallida:".mysqli_connect_error());
 }
 mysqli_set_charset($conn, "utf8");
 $sql = "SELECT id, nombre, rango FROM dotacion where id_estado=0";
 $result = mysqli_query($conn, $sql);
 if (mysqli_num_rows($result) > 0) {
 while($row = mysqli_fetch_assoc($result)) {
 $listar_dotacion[] = $row;
 }
 return $listar_dotacion;
 } else {
 return NULL;
 }
 mysqli_close($conn);
}
?>
```