Descripción Arquitectónica

Desarrollo de Software

Curso 2022-2023

3º Ingeniería Informática GP6

UNIVERSIDAD
DE GRANADA

Enunciado: Sistema de Gestión de Barcos (Astillero)

Nuestro sistema va a ser una red social, en la que los usuarios simularán que tienen un barco de pesca. Habrá un foro donde podrán ver que pesca cada barco. Además cada usuario podrá crear un barco dependiendo de lo que se quiera gastar. Al registrarse en la aplicación, al usuario se le entregará una cantidad de puntos con los que podrán construir su primer barco y podrá ganar puntos pescando y así construir otros barcos. En el foro se aplicarán una serie de filtros para evitar comentarios no deseados. Por último, la aplicación contará con un menú donde podrá construir los barcos

1. Análisis de requisitos

El objetivo general de nuestra aplicación es proporcionar una red social sobre barcos donde los usuarios puedan pasar el tiempo y relacionándose entre sí

1.1 Requisitos no funcionales

Seguridad (Aspecto fundamental en la gestión de la base de datos) : Implementando medidas de protección como firewalls, sistema de detección de intrusiones o autenticación de usuarios. Además de medidas de protección que prevengan posibles ataques externos, tales como hackers o intrusos malintencionados. Garantizando la confidencialidad de los datos personales haciendo que solo las personas autorizadas puedan acceder a la información sensible almacenada en la base de datos.

Disponibilidad: Realizaremos la monitorización del sistema una plataforma (Firebase) para asegurar que las caídas del servidor sean las mínimas posibles y que los problemas se puedan reparar rápidamente.

Escalabilidad: Nuestro sistema puede crecer y aumentar la cantidad de usuarios pescando a la vez en la aplicación.

Ética y **compromiso**: Nos comprometemos a garantizar que nuestros productos y servicios sean utilizados de una manera responsable y segura. Hemos implementado políticas estrictas que prohíben el acceso a nuestros juegos a menores de edad. Comprendemos que la componente aleatoria de los juegos de azar puede resultar adictiva en algunos casos, y nos tomamos en serio nuestra responsabilidad de proteger a los jóvenes de los posibles efectos negativos

Regulable: Cumpliendo con la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.

Eficiencia: Dando respuestas rápidas en cualquier tipo de situaciones, incluso en situaciones de sobrecarga de uso

Evolutivo:

- Se podrán comprar y vender barcos a otros usuarios
- Se podrán comprar puntos a través de un plataforma de pago para poder conseguir más puntos
- Poder expandir la aplicación a otros países y otros idiomas

Por último sobre la interfaz de usuario podemos destacar que es **amigable** ya que es intuitiva y sencilla de usar

1.2 Requisitos funcionales

- Dar de alta un usuario
- Gestión de cuenta
- Añadir comentario
- Crear barco
- Gestión de barco
- Gestión de filtro
- Gestion de peces
- Gestión de puntos

Detalle	Descripción
RF#	1
Nombre	Dar de alta un usuario
Descripción	Permitir que un usuario se pueda registrar en nuestra página web, rellenando los datos que sean necesario
Entrada	Nombre, Apellidos, Correo, Nick, Contraseña y Fecha Nacimiento
Procesamiento	[Prerrequisito: el correo no haya sido usado anteriormente] [Prerrequisito: el nick no hay sido usado anteriormente] Añadir a la BD el nuevo usuario con todos sus datos
Salida	Confirmación del alta del usuario

Detalle	Descripción
RF#	2
Nombre	Gestión de cuenta
Descripción	Permite la modificación de alguno de los datos de la cuenta de usuario (email,contraseña,nick) e incluso la eliminación de dicha cuenta
Entrada	Nuevo datos del cliente
Procesamiento	[Prerrequisito: acceso identificado] Actualización de los datos en la BD externa al sistema
Salida	Confirmación de los cambios realizados

Detalle	Descripción
RF#	3
Nombre	Añadir Comentario
Descripción	Permite añadir un comentario al foro de comentarios al sistema
Entrada	Comentario, Usuario
Procesamiento	[Prerrequisito: acceso identificado] Añadir a la BD el comentario que el usuario acaba de añadir
Salida	Confirmación de que el comentario se ha enviado correctamente

Detalle	Descripción
RF#	4
Nombre	Crear Barco
Descripción	Permite crear un barco (objeto principal del juego) con el que se puede jugar y obtener recompensas con el
Entrada	Tipo de Barco que queremos construir
Procesamiento	[Prerrequisito: acceso identificado] [Prerrequisito: el tipo de barco sea correcto] [Prerrequisito: tener los puntos suficientes para poder construir los barcos] Crear un nuevo barco en la BD externa al sistema
Salida	El barco que se acaba de construir

Detalle	Descripción
RF#	5
Nombre	Gestión de Barco
Descripción	Permite la baja/modificación de los atributos del barco (tipo de barco, nivel…)
Entrada	Nuevos atributos del barco
Procesamiento	[Prerrequisito: acceso identificado] [Prerrequisito: tener los puntos suficientes para poder realizar ciertos tipos de cambios] Actualización de los datos en la BD externa al sistema
Salida	Confirmación de los cambios realizados

Detalle	Descripción
RF#	6
Nombre	Gestión de peces
Descripción	Permitir añadir/eliminar peces (objetos que serán pescados por los objetos barcos)
Entrada	Los datos del pez
Procesamiento	[Prerrequisito: acceso identificado como gestor de la BD] [Prerrequisito: los datos introducidos del pez sean correctos] Actualización de los datos en la BD externa al sistema
Salida	Confirmación de lo cambios realizados

Detalle	Descripción
RF#	7
Nombre	Gestión de puntos
Descripción	Permite añadir/modificar el atributo puntos de la cuenta de los usuarios con los que dichos usuarios pueden comprar o mejorar los barcos
Entrada	Los datos del usuario del que vamos a modificar los puntos
Procesamiento	[Prerrequisito: acceso identificado como gestor de la BD] [Prerrequisito: los datos introducidos del usuario sean correctos] Actualización de los datos en la BD externa al sistema
Salida	Confirmación de lo cambios realizados

2. Listado y descripción de las partes interesadas en el sistema

- **Arquitecto**: Es el responsable de diseñar la estructura y la arquitectura de la aplicación. Su función principal es diseñar una solución técnica que sea escalable, eficiente, fácil de mantener y que cumpla con los requisitos funcionales y no funcionales de la aplicación.

- **Desarrollador**: Es el responsable de escribir, probar y mantener el código de una aplicación. Su función principal es transformar el diseño y los requisitos definidos por el arquitecto de software en una aplicación funcional.

- **Director de proyecto**: Es responsable de liderar y supervisar el equipo de desarrollo de software para garantizar que la aplicación se entregue dentro del plazo y presupuesto acordados, cumpliendo con los requisitos y objetivos del negocio. La función principal es planificar, coordinar y supervisar el trabajo del equipo de desarrollo, incluyendo la definición de tareas, asignación de recursos y seguimiento del progreso del proyecto. También es responsable de la gestión del presupuesto, la identificación y gestión de riesgos, la comunicación con los stakeholders del proyecto y la resolución de problemas y conflictos que puedan surgir durante el desarrollo de la aplicación.

- **Experto:** Es un profesional que tiene un conocimiento profundo y especializado sobre un área específica de la aplicación, como la experiencia del usuario, entre otros. La función principal de un experto en una aplicación es proporcionar asesoramiento y guía técnica especializada en su área de experiencia. Puede ayudar a mejorar la usabilidad y la accesibilidad de la aplicación, mientras que un experto en seguridad puede ayudar a asegurar que la aplicación cumpla con los estándares de seguridad requeridos.

- **Administrador del sistema**: Es el responsable de la configuración, el mantenimiento y la supervisión del sistema informático que aloja la aplicación. El administrador del sistema es esencial para garantizar que la aplicación esté disponible y funcione correctamente para los usuarios. La función principal en una aplicación es asegurarse de que la infraestructura de la aplicación esté en buen estado. También es responsable de garantizar que la aplicación esté en línea y disponible para los usuarios las 24 horas del día, los 7 días de la semana. Además es responsable de mantener la documentación y los procedimientos de la aplicación actualizados y realizar las pruebas de los sistemas de seguridad de manera regular.

- **Técnico de pruebas:** Es el responsable de diseñar, ejecutar y automatizar pruebas para garantizar que la aplicación cumpla con los requisitos de calidad y

funcionamiento. La función principal es detectar errores, bugs y problemas de rendimiento antes de que la aplicación sea lanzada al mercado o se implemente en producción. Además, debe ejecutar las pruebas y documentar los resultados, así como identificar y comunicar los errores y problemas encontrados al equipo de desarrollo para su corrección.

- **Usuario**: Es la persona que utiliza la aplicación para llevar a cabo una tarea específica. Pueden ser cualquier persona que esté registrada en la aplicación. Realiza tareas como pescar, construir barco, obtener puntos y comentar en el foro.

3. Listado y descripción de las inquietudes o intereses en el sistema

Al presentar los requisitos a los usuarios, expresan cierta inquietud sobre algunos aspectos del sistema.

Inquietudes de los usuarios
U1: Espero que la letra y los botones no sean demasiado pequeños y no pueda verlos con claridad en mi móvil.
U2: Temo que en el foro se puedan dar conversaciones inadecuadas o que se pueda ejercer acoso sobre otro usuario, además del uso de palabras inadecuadas. Creo que ciertas palabras deberían ser censuradas en el foro.
U3: Creo que sería conveniente que para los botones que realicen acciones terminales, como borrar o crear barco, se avise de la acciones que se va a realizar.
Experto Estructura interna, cualidades del diseño básica, filosofía del diseño funcional.

3.1. Inquietudes

A partir de las inquietudes generales recogidas al presentar los requisitos a las partes interesadas se obtienen las siguientes inquietudes conectadas con los requisitos no

funcionales y funcionales. Las inquietudes U1 y U3 conforman un nuevo requisito no funcional y la inquietud de U2 es incorporada como un nuevo requisito funcional.

3.1.1. Inquietudes relacionadas con requisitos no funcionales

Añadimos un nuevo requisito ya que el usuario espera que la información sea clara y entendible, además de que las palabras sean legibles y con un tamaño de fuente adecuado. El nuevo requisito no funcional es:

- (V2) Accesibilidad: el sistema deberá cumplir con la especificación W3C para todos los textos que contenga (diseño, estilos y Web claros, contraste fuerte en colores; botones/letras grandes o adaptables,notificaciones y retroalimentación claras).

3.1.2. Inquietudes relacionadas con requisitos funcionales

Se añade un requisito funcional más para atender a la inquietud de U2:

Detalle	Descripción
RF#	8-v2
Nombre	Gestión de Filtro
Descripción	Permite añadir/modificar/eliminar filtros (palabras que posteriormente se censuran en los comentarios)
Entrada	La palabra que queremos añadir, modificar o modificar
Procesamiento	[Prerrequisito: acceso identificado como gestor de la BD] Actualización de los datos en la BD externa al sistema
Salida	Confirmación de los cambios realizados

4. Descripción detallada del punto de vista contextual

Describe las relaciones de la aplicación con las entidades externas y con los sistemas internos y externos con los que interactúa.

4.1. Inquietudes

4.1.1. Ámbito del sistema y responsabilidades

Las responsabilidades del sistema son las siguientes:

Red social ambientada en pesca
● Comentar en el foro de la red social
● Presentar las diferentes opciones que puedes hacer a tu barco
● Utilizar tu barco para pescar y conseguir puntos
Capacidades excluidas en una primera versión del sistema:
● Poder comprar y vender barcos a otros usuarios
● Comprar puntos a través de una plataforma de pago

4.1.2. Identificación de las entidades externas y servicios y datos usados

- **Usuarios del sistema**: Personas que utilicen o contribuyan a la red social como usuarios, administradores del sistema…
- **La BD**: Donde se almacenan todos los datos que maneja nuestra aplicación
- **Sistema interno de filtros**: Donde se tendrán en cuenta las palabras que tendremos que censurar y se encargará de cambiar dichas palabras por asteriscos e incluso restarle puntos al usuario que haga ese comentario

4.1.3. Naturaleza y caracterısticas de las entidades externas

- La **BD de la aplicación**: en ella se almacenan todos los datos de la aplicación. Alojada en los servidores de la empresa. También se realizan las garantías de seguridad, confidencialidad, disponibilidad, escalabilidad y legalidad
- **Usuarios del sistema:** Algunos clientes no pueden registrarse en la aplicación ya que solo puede ser usadas por personas mayores de 18 años
- **Sistema interno de filtros:** Está integrado en el foro de comentarios y censura ciertas palabras malsonante u ofensivas que puede introducir el usuario

4.1.4. Identificación y responsabilidades de las interfaces externas

Interfaces de servicios:

- Tabla interfaz de usuario.

Detalle	Descripción
Nombre	Interfaz de usuario
Semántica	A través de esta interfaz el usuario realiza operaciones en el sistema como: identificarse/registrarse, comentar, comprar barcos, pescar…
Parámetros	Nombre de usuario, contraseña para el caso de registro/identificarse Comentario para la operación de comentar Información de barco para la operación de compra El barco específico para la operación de pescar
Acciones a realizar	Registrar al usuario en la operación de registro Iniciar sesión del usuario en la operación en que se identifica Capturar y registrar el comentario en la operación de comentar Capturar la información del nuevo barco y registrarlo para el usuario en la operación de compra de barco Realizar una sesión de pesca para la operación de pescar.
Excepciones (1. Usuario ya existe para la operación de registro)	1. Se devuelve un mensaje de error por pantalla.

Excepciones (2. Usuario no existe o contraseña errónea en la operación de inicio de sesión) Excepciones (3. Operación no autorizada para la compra del barco)	2. Se devuelve un mensaje de error por pantalla. 3. Se devuelve un mensaje de error por pantalla
Errores (El usuario interrumpe alguna de las operaciones)	A tratar por el sistema con su implementación.

- Tabla interfaz sistema con la BD

Detalle	Descripción
Nombre	Interfaz de la Base de Datos
Semántica	Se realizan operaciones de consulta/inserción sobre credenciales de usuario, barcos, peces y puntos en la Base de Datos a través de Sentencias SQL.
Parámetros	Nombre de la base de datos, nombre de usuario y contraseña encriptados para las consultas/modificaciones respectivas a los usuarios. Nombre de la base de datos, nombre de usuario e información del barco para las consultas/modificaciones respectivas a los barcos. Nombre de la base de datos para las consultas/modificaciones respectivas a los peces. Nombre base de datos, nombre de usuario y cantidad de puntos para consultas/modificaciones respectivas a los puntos Nombre base de datos, nombre de usuario para consultas/modificaciones respectivas a los puntos
Acciones a realizar	Conectarse al SGBD, realizar la operación de consulta/inserción correspondiente mediante la ejecución del código SQL y desconectarse del SGBD.
Excepciones (1. error sintáctico código	Se devuelve un código de excepción.

SQL) Excepciones (2. operación no autorizada)	Se devuelve un código de excepción.
Errores (sistema externo no responde)	A tratar por el sistema con su implementación.

- Tabla interfaz Administrador

Detalle	Descripción
Nombre	Interfaz de administrador
Semántica	A través de esta interfaz el administrador realiza tareas administrativas como gestión de puntos de usuarios, añadir/eliminar usuarios…
Parámetros	Nombre de administrador, contraseña para identificarse. [Si procede] Nombre del nuevo usuario [Si procede] Nombre del usuario a eliminar [Si procede] Cantidad de puntos a sumar o restar
Acciones a realizar	Iniciar sesión del administrador en la operación en que se identifica. Añadir un nuevo usuario a la base de datos. Eliminar un usuario existente. Modificar los puntos de un usuario existente.
Excepciones (1. Nombre administrador no existe o contraseña errónea en la operación de inicio de sesión). Excepciones (2. Nombre de usuario existente en la base de datos, a la hora de crear uno nuevo). Excepciones (3. Nombre de usuario no valido para eliminar). Excepciones (4. Cantidad negativa [por debajo de 0] de puntos en el usuario). Excepciones (5. No existe usuario para modificar los puntos).	1. Se devuelve un mensaje de error por pantalla. 2. Se devuelve un mensaje de error por pantalla. 3. Se devuelve un mensaje de error por pantalla 4. Se devuelve un mensaje de error por pantalla 5. Se devuelve un mensaje de error por pantalla
Errores (El administrador interrumpe alguna de las operaciones)	A tratar por el sistema con su implementación.

4.1.5. Naturaleza y características de las interfaces externas

Las características generales son:

- Totalmente automatizadas
- Transaccionales

- **Interfaz de Usuario:**
    - **Eficiente:** Debe responder rápidamente
    - **Disponible:** Alta disponibilidad
    - **Segura y confidencial:** Datos encriptados de los usuarios, protocolo https
    - **Escalable:** Preparada para aceptar un gran número de usuarios

- **Interfaz de Base de Datos**
    - **Eficiente:** Debe responder muy rápidamente
    - **Disponible:** Alta disponibilidad, servidores en espejo
    - **Segura y confidencial:** Acceso identificado a base de datos, protocolo https, datos de acceso encriptados
    - **Escalable:** Preparada para aceptar un gran volumen de datos

- **Interfaz de Administrador:**
    - **Eficiente:** Debe responder rápidamente
    - **Disponible:** Alta disponibilidad
    - **Segura y confidencial:** Acceso identificado a la interfaz, datos encriptados, protocolo https

4.1.6. Otras interdependencias externas

Debido a que no utilizamos sistemas externos nuestra aplicación no presenta interdependencias externas.

4.1.7. Impacto del sistema en su entorno

El sistema interno de filtros puede filtrar tanto palabras como fechas pero en esta primera versión de nuestro sistema solo utilizaremos el filtro de palabras dejando el filtro de fechas para futuras versiones.

4.1.8 Completitud, consistencia y coherencia global

Se garantiza que los procesos más importantes tengan una cobertura adecuada y todos los datos que requieran serán almacenados en la base de datos y accesibles a estos sistemas

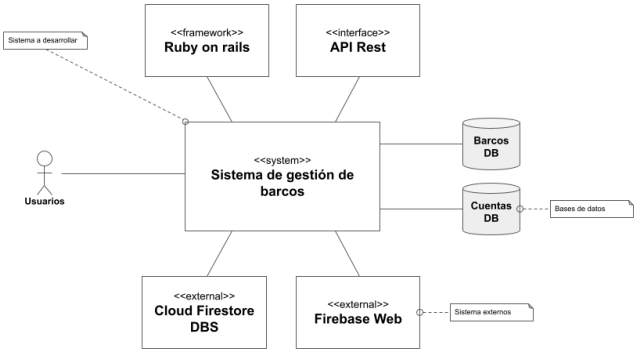
4.1.9 Inquietudes para cada interesado del sistema

Tipo de Interesado	Inquietudes
Arquitecto	Todas las inquietudes
Desarrollador	Todas las inquietudes
Director de Proyecto	Todas las inquietudes
Experto	Ámbito y responsabilidades del sistema, Impacto del sistema en su entorno, Completitud,consistencia y coherencia global
Administrador del Sistema	Todas las inquietudes
Técnico de Pruebas	Todas las inquietudes
Usuario	Ámbito y responsabilidades del sistema; identificación´on de las entidades externas y servicios y datos usados; completitud, consistencia y coherencia global

4.2 Modelos: modelo de contexto

4.2.1. Notación

Diagrama UML para las partes interesadas más técnicas:



4.3. Modelos: escenarios de interacción

ESCENARIO FUNCIONAL DE CONTEXTO 1	
Descripción	Creación de barco
Estado del sistema	El usuario se ha identificado en el sistema con sus credenciales
Entorno del sistema	El entorno de implementación funciona normalmente, sin problemas
Estímulo externo	Click sobre el botón “crear barco”
Respuesta requerida del sistema	El sistema muestra muestra los tipos de barcos que se pueden seleccionar
Estímulo externo	Click sobre el botón “barco *tipo*”
Respuesta requerida del sistema	El sistema mostrará un aviso con la información que tiene la acción de crear barco
Estímulo externo	Click sobre botón “aceptar”
Respuesta requerida del sistema	El sistema conectará con la base de datos y añadirá el nuevo barco
Estímulo externo	El servidor envía *OK*
Respuesta requerida del sistema	El sistema mostrará un aviso de que la creación se ha realizado correctamente

ESCENARIO FUNCIONAL DE CONTEXTO 2	
Descripción	Fallo en la creación de barco
Estado del sistema	El usuario se ha identificado en el sistema con sus credenciales
Entorno del sistema	El entorno de implementación funciona normalmente, sin problemas
Estímulo externo	Click sobre el botón “barco ‘tipo’ ”
Respuesta requerida del sistema	El sistema mostrará un aviso con la información que tiene la acción de crear barco
Estímulo externo	Click sobre botón “aceptar”
Respuesta requerida del sistema	El sistema conecta con la base de datos y comprueba que el usuario no tiene los puntos suficientes para crear el barco
Estímulo externo	El sistema envía un fallo en la inserción
Respuesta requerida del sistema	El sistema mostrará una notificación que avisará de que no se podido realizar la creación

ESCENARIO FUNCIONAL DE CONTEXTO 3	
Descripción	Selección de barco para pescar
Estado del sistema	El usuario se ha identificado en el sistema con sus credenciales u ha accedido al foro
Entorno del sistema	El entorno de implementación funciona normalmente, sin problemas
Estímulo externo	Click sobre el botón “barco ‘tipo’ ‘nivel’ ”
Respuesta requerida del sistema	El sistema conecta con la base de datos y enviará como referente de barco actual el barco con el que se quiere pescar

ESCENARIO FUNCIONAL DE CONTEXTO 4	
Descripción	Pescar
Estado del sistema	El usuario se ha identificado en el sistema con sus credenciales, ha accedido al foro y ha seleccionado el barco
Entorno del sistema	El entorno de implementación funciona normalmente, sin problemas
Estímulo externo	Click sobre el botón “pescar”

Respuesta requerida del sistema	El sistema conecta con la base de datos y sumará los puntos que da la pesca al usuario
Estímulo externo	El sistema envía *OK*
Respuesta requerida del sistema	El sistema conectará con la base de datos y añadirá como nuevo comentario al foro “ ‘usuario’ ha pescado”

4.4 Problemas y errores comunes

Algunos de los problemas y errores que hemos detectado. Se pone sólo asterisco (*) cuando se haya considerado el problema y se crea que está bajo control.

- Entidades externas ausentes o incorrectas: *
- Dependencias implícitas entre entidades externas no consideradas, y que pueden afec-
- tar al propio sistema: *
- Descripciones imprecisas o ausentes de una interfaz externa: *
- Nivel de detalle inapropiado: *
- Degeneración del entorno, o efectos progresivos de cambios no controlados que pueden no apreciar las partes interesadas: *
- Contexto o ámbito implícito o asumido: *
- Complicación de interacciones: este problema será considerado en etapas posteriores del desarrollo.
- Abuso de jergas tecnológicas que pueden no entender las partes interesadas: *

4.5 Lista de verificación

1. ¿Has consultado con todos las partes interesadas quienes están interesados en el punto de vista contextual? Sí
2. ¿Has identificado todas las entidades externas al sistema y sus responsabilidades más relevantes? Sí
3. ¿Comprendes bien la naturaleza de cada interfaz con cada entidad externa y está documentada en un nivel apropiado de detalle? Sí
4. ¿Has considerado las posibles dependencias entre las entidades externas con las que hay que interactuar? ¿Están documentadas estas dependencias implícitas en la DA? Sí
5. ¿Ilustra de forma adecuada el diagrama de contexto todas las interfaces entre el sistema y su entorno con las suficientes definiciones aclaratorias en el diagrama? Sí
6. ¿Han acordado formalmente todos las partes interesadas con los contenidos del modelo contextual? ¿Está documentado en algún lugar? Sí
7. ¿Está situado el modelo contextual bajo algún sistema formal de control de cambios? Sí
8. ¿Se sigue el proceso de control de cambios? ¿Se consulta a las partes interesadas para que den su consentimiento formal? Sí

9. ¿Has identificado todas las capacidades o requerimientos básicos del sistema y están documentados en el nivel de detalle apropiado? Sí
10. ¿Es la definición del ámbito consistente internamente? Sí
11. ¿Está el entorno especificado en un nivel apropiado de detalle, equilibrando brevedad con claridad y completitud? Sí
12. ¿Has explorado un conjunto de escenarios realistas de interacciones entre el sistema y actores externos? Sí
13. ¿Les parece claro el contexto, el ámbito y posibles implicaciones a otros equipos con los que debes interactuar? Sí
14. ¿Has comprobado si en el modelo contextual existe alguna información que parezca obvia y debería ser explícitamente descrita pero se ha omitido? Sí
15. ¿Están todos los datos requeridos para los procesos principales almacenados en algún lugar, interna o externamente? Sí
16. ¿Está formulada de forma coherente la solución global? Sí

5. Descripción detallada del punto de vista funcional

5.1. Inquietudes

5.1.1. Capacidades funcionales

Las funciones de las que se hará cargo el Sistema de Gestión de Barcos son:

- Dar de alta un usuario
- Gestión de cuenta
- Añadir comentario
- Crear barco
- Gestión de barco
- Eliminar un barco
- Gestión de filtro
- Gestion de peces
- Gestión de puntos

Otras funciones que la responsabilidad no recae en el sistema:

- Recompensas por experiencia (en futuras versiones)
- Compraventa de barcos entre usuarios (en futuras versiones)
- Compra de puntos con tarjeta de crédito (entidad externa y en futuras versiones)

5.1.2. Interfaces externas

Las interfaces externas son los datos, eventos y flujos de control entre tu sistema y otros. Creemos que debemos expresar dichos flujos desde Usuarios/Sistemas Externos hacia el Sistema de Gestión de Barcos y viceversa. Tenemos dudas sobre la expresión desde Usuarios/Sistemas Externos hacia el Sistema, ya que en el ejemplo no lo hace pero intuimos del libro que sí habría que hacerlo.

En la sección anterior, Identificación y responsabilidades de las interfaces externas, definimos las siguientes interfaces externas

- Tabla interfaz de usuario.
- Tabla interfaz sistema con la BD**.**
- Tabla interfaz Administrador

5.1.3. Estructura interna

Nuestra aplicación tendrá un sistema con software propio, donde habrá que desarrollar gran parte del mismo. Se asume un estilo arquitectónico general de tipo “Cliente/Servidor”. La gran parte de las funcionalidades del lado del cliente y el controlador de conexiones con el servidor está implementado en código Dart (concretamente con el framework Flutter).

La parte del servidor estará implementada exclusivamente usando el framework Ruby on Rails, proveyendo una interfaz web del sistema, una API a la que conectan los clientes flutter para la obtención de los datos e interacción con el modelo.

También añadir que en este lado se asume un estilo arquitectónico de “Abstracción de datos y organización OO”. Además, para la parte de la lógica de la aplicación y cómo gestionar su conexión con las demás partes del sistema emplearemos el estilo arquitectónico de **Modelo-Vista-Controlador.** La parte del servidor dispondrá de múltiples paquetes:

- **Modelo**: En este paquete estará implementado la lógica de la aplicación, es decir, las clases Mar, Barco, Peces… Con sus respectivas funciones. Estará implementado en Ruby On Rails.
- **Controlador**: Este paquete se encargará de la obtención/paso de datos hacia/desde el modelo, así como de conectar con el módulo de conexiones de la Base de Datos y de proporcionar la información a las vistas de las páginas Web. Estará implementado en Ruby On Rails.
- **Vista**: Este paquete es el encargado de proveer las vistas Web de la aplicación. Estarán implementadas en HTML.
- **API:** Paquete encargado de proveer las funciones a los clientes de la aplicación móvil.

- **Conexión a la Base de Datos:** Este paquete se encargará de gestionar las conexiones son el servidor de la base de datos.

Para garantizar una cohesión consistente, se utilizarán módulos para encapsular el software necesario para cada capacidad funcional. Para tener un mínimo acoplamiento, se juntaran en mismos módulos funciones que tengan fuerte interacción entre ambas.

Tabla representativa:

Componente	Requisito	Descripción
Cliente Dart (Flutter)	1. Interfaz de usuario 2. Navegación y flujo de trabajo. 3. Comunicación con el servidor 4. Lógica del negocio	→ Implementa la interfaz de usuario para los clientes.. → Controla la navegación y el flujo de trabajo dentro de la aplicación. → Gestiona las conexiones y la comunicación con el servidor .→ Implementa la lógica del negocio como son las clases barcos, pesca, etc.
Modelo en el servidor	5. Replicar lógica de negocio en servidor	→Replica el modelo del cliente flutter pero con distintas funcionalidades para comunicarse con el controlador de cara a guardar datos en la Base de Datos
Controlador	6. Gestión de usuarios, barcos… etc. 7. Extracción de datos del servidor	→ Controla la autenticación y autorización de usuarios. → Extrae información del servidor para inicializar las vistas.
Vista	8. Interfaz gráfica mediante HTML del usuario	→ Implementa la interfaz gráfica de usuario.
API	9. API y conexión con clientes	→ Provee la API y gestiona las conexiones con los clientes.
Conexión a BD	10. Gestión de la base de	→ Administra las

	datos	conexiones…

5.1.4. Filosofía del diseño funcional

Los principios de diseño cumplidos por el sistema son:

- **Cohesión**: Gracias a la aplicación del Modelo-Vista-Controlador así como de la arquitectura cliente-servidor y estilo orientado a objetos. Todos los paquetes del sistema contienen componentes muy relacionados en funcionalidad.

- **Bajo acoplamiento**: Sobre todo seguido gracias al estilo Modelo-Vista-Controlador, pero también logrado mediante el uso de Cliente/Servidor ya que podemos cambiar cualquier parte del sistema sin que afecte a las demás. Por ejemplo, el cliente móvil sólo llamará a la función de la API sin tener que tener en cuenta la implementación de esta.

- **Coherencia**: se puede deducir de la aplicación de los principios previamente mencionados, como son la máxima cohesión y el mínimo acoplamiento.

- **Consistencia**: las decisiones de diseño se aplican teniendo en cuenta a todo el sistema en su conjunto como entidad.

- **Modularidad**: con la capacidad de dividirse en módulos gracias al Modelo-Vista-Controlador, lo que permite una mayor flexibilidad y escalabilidad del sistema.

- **Extensibilidad**: El sistema será fácilmente extensible ya que las nuevas posibles funcionalidades no tienen porqué afectar a las demás. Está logrado en gran parte gracias al bajo acoplamiento logrado en el sistema.

En general, gracias a la aplicación de Modelo-Vista-Controlador conseguimos un gran número de características de diseño. Otras características que cumple son: Flexibilidad Funcional, Generalidad, baja interdependencia…

5.1.5. Inquietudes para cada interesado del sistema

Tipo de interesado	Inquietudes
Arquitecto	Diseño de una arquitectura, patrones de diseño, seguridad y mantenimiento del

	software.
Desarrollador	Calidad del diseño y de la estructura interna, además de las capacidades funcionales e interfaces externas.
Director de proyecto	Planificación del proyecto, gestión de recursos y costes, comunicación con el cliente, general…
Experto	Estructura interna, cualidades del diseño básica, filosofía del diseño funcional,
Administrador del sistema	La filosofía de diseño funcional básica, interfaces y estructura interna.
Técnico de pruebas	Estructura interna, capacidad funcional, cualidades de diseño e interfaces externas.
Usuario	Capacidad funcional e interfaces externas

5.2. Modelos

5.2.1. Modelo de la estructura funcional

Diagrama de componentes(Se ha usado una notación UML.)

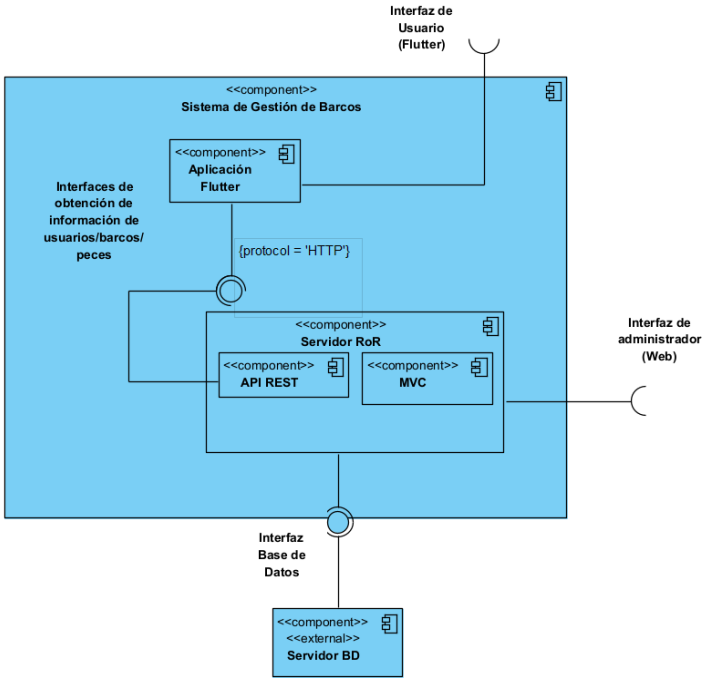
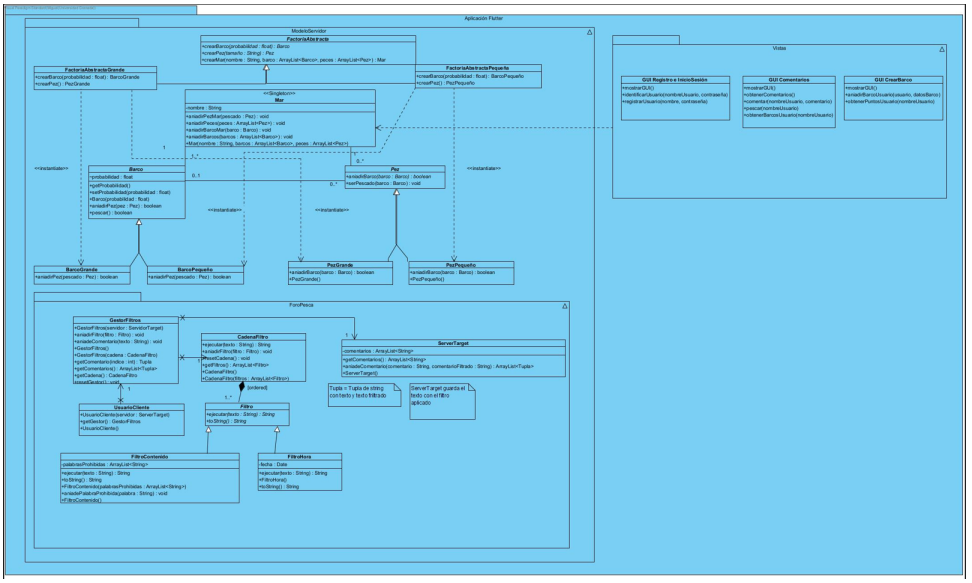
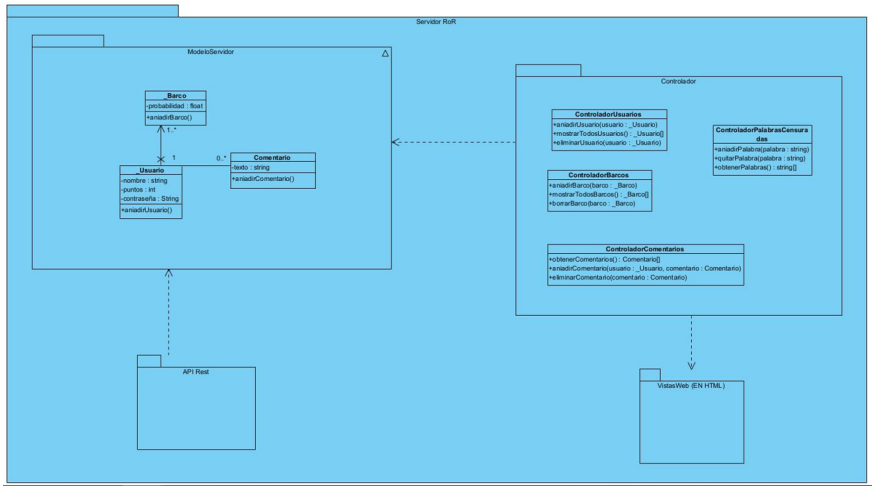


Diagrama de clases





5.3. Actividades

5.3.1. Identificar los elementos

En la elaboración del diagrama de clases, se sigue una metodología que parte de las funcionalidades necesarias para identificar los conceptos más relevantes y agruparlos en clases. Además, se ha aplicado el estilo arquitectónico Modelo Vista Controlador para estructurar gran parte de las clases en paquetes. Para otras tareas de refinamiento, como la descomposición o la herencia, se han dejado para la fase de diseño posterior.
Además, para la elaboración del modelo de la perspectiva funcional hemos desarrollado un **diagrama de componentes**. Para el desarrollo de dicho diagrama hemos tenido que identificar previamente los siguientes elementos: **modelo**, **vista, controlador, Aplicación**

Flutter, API REST, Conexión a BD. A continuación profundizaremos sobre las responsabilidades de dichos elementos.

5.3.2. Asignar responsabilidades a los elementos

Las responsabilidades de dos elementos del diagrama de componentes pueden verse a continuación:

Responsabilidades de los elementos	
Aplicación Flutter	● Se encarga de implementar la interfaz del usuario del sistema mediante una aplicación móvil. ● Gestiona las conexiones entre la aplicación Flutter y la API REST.
Modelo (Aplicación Flutter)	● Implementa la lógica del sistema, como son las clases Pez, PezGrande, PezPequeño, Barco, BarcoPequeño, BarcoGrande, Mar y sus respectivos métodos detallados en el diagrama de clases anterior.
Modelo (Servidor RoR)	● Implementa parte de la lógica del sistema, como son las clases Barco, Usuario y Comentario junto con los respectivos métodos para interactuar con el Controlador
Vista (Servidor RoR)	● Provee de interfaz gráfica a los usuarios mediante Web.
Controlador (Servidor RoR)	● Se encarga de unir el Modelo con la Vista y la API REST. ● Implementa los métodos básicos para el funcionamiento de nuestra aplicación como son: identificar/registrarUsuario, pescar(), insertar/consultar Barcos, insertar Peces… Estos servirán de apoyo para los demás elementos detallados anteriormente

API Rest (Servidor RoR)	● Da un servicio mediante protocolo HTTP a las aplicaciones móviles de manera que estas puedan conectar con el servidor y obtener los datos necesarios para el funcionamiento de la aplicación.

5.3.3. Diseñar las interfaces

En la arquitectura, las interfaces entre el servidor y los clientes son fundamentales y requieren un enfoque orientado a datos debido a que los lenguajes de programación son diferentes y la conexión basada en mensajería (REST).

Nombre: Interfaz de Usuario	
Semántica	Interfaz del usuario con el sistema. Se emplea para operaciones de identificación/registro/comentar/comprar barcos/pesca
Tipo	REST/HTTP
Prerrequisitos	El sistema funciona correctamente para todas las operaciones. Para el caso de comentar/comprar barcos/pescar el usuario debe haberse identificado previamente.
Entrada	Nombre de usuario y contraseña para el caso de identificación/registro Para el caso de comentar la entrada es el texto del comentario Para el caso de comprar barcos la entrada es la información del barco. No es necesario ninguna entrada para pescar.
Salida	Ninguna salida para la identificación Mensaje de operación realizada con éxito para el registro/comentar/compra de barco.. Mensaje con los puntos ganados para la pesca.
Efecto	Se capturaron por el sistema las entradas requeridas.

	Se insertaron tuplas en las operaciones de registro, comentarios, compras de barco. Se modificaron tuplas para el caso de pesca.

Nombre: Interfaz del Servidor con BD	
Semántica	Interfaz del servidor con la BD. Se emplea para operaciones de registro/consultar comentarios, usuarios, barcos, pescas…
Tipo	
Prerrequisitos	El sistema funciona correctamente para todas las operaciones.
Entrada	Sentencia SQL para la inserción de una fila con el nombre de usuario y su contraseña para la identificación y registro. Sentencia SQL para la inserción/consulta de los comentarios. Sentencia SQL para la inserción/consulta de los barcos. Sentencia SQL para la consulta de la pesca.
Salida	Ninguna salida para la identificación Mensaje de operación realizada con éxito para el registro/comentar/compra de barco.. Mensaje con los puntos ganados para la pesca.
Efecto	Se capturaron por el sistema las entradas requeridas. Se insertaron tuplas en las operaciones de registro, comentarios, compras de barco. Se modificaron tuplas para el caso de pesca.

Nombre: Interfaz de Administrador	
Semántica	Interfaz del administrador con el sistema. Se emplea para operaciones de identificación/modificación/inserción de Barcos, Usuarios y Comentarios.
Tipo	HTTP
Prerrequisitos	El sistema funciona correctamente para todas las operaciones. Para el caso de identificación/modificación/inserción el usuario debe haberse identificado previamente.
Entrada	Nombre de administrador y contraseña para el caso de identificación. Para el caso de modificación/inserción se necesitará: - Usuario. - Barco. - Comentario y Usuario.
Salida	Ninguna salida para la identificación Mensaje de operación realizada con éxito para el modificación/inserción de barco, usuario y comentario..
Efecto	Se capturaron por el sistema las entradas requeridas. Se insertaron tuplas en las operaciones de usuarios, comentarios y barcos. Se modificaron tuplas para el caso de usuarios, comentarios y barcos.

5.3.4. Diseñar los conectores

Todos los conectores entre cliente y servidor serán manejados por Ruby on Rails, utilizando un modelo de arquitectura de servicios web RESTful. Esto permite que los datos se manejen de manera eficiente y flexible, lo que permite un procesamiento rápido y eficaz de los mismos. Además, las conexiones con los clientes web se hacen también mediante protocolo HTTP, por lo que estos conectores nos sirven para ambos tipos de clientes.

Conectores respecto a los Usuarios:

Detalle	Descripción
Nombre	“registroUsuario”
Semántica	Se envían datos de usuario y se registra el alta
Tipo	REST (mensajería): PUT
Prerrequisitos	No estar identificado como usuario
Entrada	Nombre usuario y contraseña
Salida	Confirmación de registro

Detalle	Descripción
Nombre	“consultarUsuario”
Semántica	Se envían datos del usuario para permitir iniciar sesión
Tipo	REST (mensajería): GET
Prerrequisitos	El sistema funciona correctamente
Entrada	Identificador del usuario
Salida	[Nombre de usuario existe y la contraseña es correcta] El usuario se identificó correctamente por lo que se creó una conexión con el servidor y se descargaron los datos del usuario, con los cuales se inicializó el modelo.

Conectores respecto a los comentarios:

Detalle	Descripción
Nombre	“nuevoComentario”
Semántica	Operaciones de gestión de comentarios en la BD (hacia el servidor)
Tipo	REST (mensajería): PUT
Prerrequisitos	Estar identificado como usuario
Entrada	Nombre usuario y Texto comentario
Salida	Operación realizada.

Detalle	Descripción
Nombre	“consultarComentarios”
Semántica	Muestra los comentarios del foro.
Tipo	REST (mensajería): GET
Prerrequisitos	Estar identificado como usuario
Entrada	
Salida	Listado de comentarios

Conectores respecto a los barcos:

Detalle	Descripción
Nombre	“nuevoBarco”
Semántica	Operaciones de gestión de barcos en la BD (hacia el servidor)
Tipo	REST (mensajería): PUT
Prerrequisitos	Estar identificado como usuario. Tener suficientes puntos.
Entrada	Nombre usuario y tipo de barco.
Salida	Operación realizada.

Detalle	Descripción
Nombre	“consultarBarcos”
Semántica	Muestra los barcos del usuario.
Tipo	REST (mensajería): GET
Prerrequisitos	Estar identificado como usuario
Entrada	Nombre de usuario
Salida	Listado de barcos

Conectores respecto a los puntos:

Detalle	Descripción
Nombre	“registarPuntos”
Semántica	Operaciones de gestión de puntos en la BD (hacia el servidor)
Tipo	HTTP PUT
Prerrequisitos	Estar identificado como administrador
Entrada	Nombre de usuario y Cantidad de puntos
Salida	Operación realizada.

Detalle	Descripción
Nombre	“consultarPuntos”
Semántica	Muestra los puntos obtenidos por un usuario.
Tipo	HTTP GET
Prerrequisitos	Estar identificado como usuario
Entrada	Nombre de usuario
Salida	Cantidad de puntos del usuario.

5.3.5. Comprobar la trazabilidad funcional

Para comprobar la trazabilidad funcional, podemos consultar la estructura interna elaborada respecto a la tabla representativa → Tabla representativa.

Esta tabla de trazabilidad ayuda a garantizar que todos los componentes y módulos del sistema estén relacionados con sus respectivas funcionalidades y requisitos, lo que facilita el seguimiento de las implementaciones y el mantenimiento del sistema.

5.3.6. Recorrer escenarios comunes

Los distintos escenarios que comprobaremos seguirán una metodología muy similar. A continuación detallamos algunos:
Cliente se identifica en aplicación móvil. Cuando el cliente se identifica inserta sus credenciales en la vista que proporciona la aplicación móvil para ella y pulsa el botón "Identificarse" se produce una llamada al método comprobarUsuario del controlador de conexiones flutter. Este método lo que hace es llamar mediante el protocolo HTTP a la API REST, la cual a su vez llama al mismo método del controlador implementado en el servidor. Por último éste lanza otra llamada al módulo de conexión con la BD el cual comprueba que las credenciales sean correctas y le devuelve un booleano según sea o no correcto. Esta respuesta se irá devolviendo en cascada hasta llegar de nuevo a la interfaz de la aplicación flutter, mostrando esta un mensaje de error o de éxito según corresponda.

Cliente registra un nuevo barco. Cuando el cliente haya insertado la información de dicho barco en la vista proporcionada por la aplicación móvil implementada en Flutter y pulse el botón "Comprar", se llama al método añadirBarco del controlador de conexiones de la aplicación flutter quien a su vez invoca un método de la API REST, la cual lanza una llamada al controlador. Este lanza otra llamada al gestor de conexiones de la BD para comprobar que el usuario tenga los puntos suficientes y si los tiene inserta el barco tanto en el modelo como en la base de datos. Si no dispone de los puntos suficientes devolverá el error. (Aunque previamente podríamos haberlo comprobado en la aplicación flutter). Tras esto, se inserta el barco en el modelo local.

5.3.7. Analizar las interacciones

Hemos podido comprobar el bajo acoplamiento en el diagrama de clases ya que no hay enlaces en exceso entre las distintas clases del diagrama, si no los estrictamente necesarios. De la misma manera, en el diagrama de componentes sólo están las conexiones estrictamente necesarias y básicas, sin componentes excesivamente complejos que podrían afectar a esta vertiente.

5.3.8. Analizar la flexibilidad

El empleo de Cliente/Servidor, MVC y una visión Orientada a Objetos son muy eficientes en este aspecto, pues una nueva funcionalidad en el modelo supondría añadir el respectivo método al controlador, API, conexión a la BD (si fuese necesario) y las respectivas nuevas vistas en la aplicación móvil y web. Es decir, los cambios serían los básicos para poder añadir dicha funcionalidad.

5.4. Problemas y errores comunes

Se pone una (x) cuando se haya considerado el problema y se crea que está bajo control.

Problema	Bajo Control
Interfaces pobremente definidas	X
Dificultades en la autenticación y autorización de usuarios.	X
Interfaces de usuario poco intuitivas o difíciles de usar.	
Vista Sobrecargada (No se han introducido elementos de otros puntos de vista)	
Diagramas sin definiciones de elementos: el diagrama de clases no presenta conectores ni interfaces.	X
Dificultades para reconciliar las necesidades de distintas partes interesadas	X
Nivel de detalle erróneo	X
Rendimiento insuficiente al manejar un gran volumen de registros de capturas	
Demasiadas dependencias	X
Inadecuada gestión de errores y excepciones en la aplicación.	X

5.5. Lista de verificación

- ¿Tiene menos de 15 a 20 elementos de nivel superior? **Si**
- ¿Tienen todos los elementos un nombre, responsabilidades claras e interfaces claramente definidas? **Si**
- ¿Tienen lugar todas las interacciones de elementos a través de interfaces y conectores bien definidos que unen las interfaces? **Si**
- ¿Exhiben los elementos un nivel apropiado de cohesión? **Si**
- ¿Exhiben los elementos un nivel apropiado de acoplamiento? **Si**
- ¿Ha identificado los escenarios de uso importantes y los ha utilizado para validar la estructura funcional del sistema? **No, se ha decidido dejarlo para la fase de diseño, pero los más importantes han sido considerados en el punto de vista del contexto.**
- ¿Has comprobado que tu sistema cubre todos los requisitos funcionales? **Sí**
- ¿Has definido y documentado un conjunto apropiado de principios de diseño arquitectónico los cuales cumple tu sistema? **Sí**
- ¿La arquitectura permite la escalabilidad necesaria para manejar un aumento en el número de usuarios y la cantidad de datos de pesca? **Si, hasta cierto punto.**
- ¿Se ha establecido una estrategia para la gestión de errores y excepciones en la arquitectura del sistema? **Si**
- ¿Se ha previsto la posibilidad de implementar nuevas funcionalidades y características en la arquitectura del sistema sin afectar negativamente a las existentes? **Si, a través de módulos.**
- ¿Se ha considerado la experiencia del usuario y la facilidad de uso en el diseño de la interfaz de usuario y la arquitectura del sistema? **Si**
- ¿Se ha establecido una estrategia para la gestión de errores y excepciones en la arquitectura del sistema? **No, se ha dejado para otras fases del diseño.**

6. Descripción detallada, punto de vista de información

6.1. Descripción

Detalle	Descripción
Definición	Describe la forma que la arquitectura almacena, manipula, gestiona y distribuye la información
Inquietudes	Estructura y contenido de la información, flujo de información, propietarios de los datos, volúmenes de datos y normativa
Modelos	Modelos estáticos de los datos, modelo de

	flujo de información y modelos de propiedad de datos.
Problemas y errores comunes	Incompatibilidad de datos, mala calidad de los mismos, múltiples fuentes de actualización inevitables y complejidad de la interfaz

6.2. Inquietudes

6.2.1. Estructura y contenido de la información

Fijándonos en las entidades de tipo importantes que conforman nuestro sistema, debemos contar con las siguientes estructuras de información:

- Tabla de base de datos con información de los **usuarios**:

Atributo	Descripción
id_usuario	Identificador de un usuario único, puede ser un índice de tipo entero o un atributo clave primaria como el DNI o correo de un usuario
nombre	Nombre con el que el usuario se registra en el sistema
puntos	Contador de puntos del usuario
contraseña	Clave cifrada con la que el usuario se registra en el sistema y que permite iniciar sesión en las posteriores conexiones
barcos (arraylist)	Lista de los todos los barcos que son propiedad del usuario
comentarios (arraylist)	Lista de todos los comentarios creados por el usuario

- Tabla de base de datos con información de los **barcos**:

Atributo	Descripción
id_barco	Identificador de un barco único, puede ser un índice de tipo entero

probabilidad	Probabilidad con la que el barco pescará todos los peces del mar cuando realice una pesca

- Tabla de base de datos con información de los **comentarios**:

Atributo	Descripción
id_comentario	Identificador de un comentario único, puede ser un índice de tipo entero
texto	Cadena con el mensaje que contiene el comentario
usuario	Objeto usuario que ha escrito dicho comentario, con el que está relacionado.

6.2.2. Flujo de información

Es importante identificar los flujos de datos críticos que deben ser monitoreados y controlados. Más adelante, se presentan los modelos de flujo de información que muestran con más detalle, la interacción entre entidades y cómo se mueve la información a través del sistema.

Los usuarios van a ser los que creen sus propios barcos a través de la interfaz web del sistema. Además, serán los usuarios los que iniciarán el proceso de pesca para un barco concreto que sea de su propiedad.

Los usuarios van a ser los que creen comentarios, también pueden leer comentarios de otros usuarios. Todas las operaciones se realizan a través de la interfaz, que interactúa directamente con la base de datos donde se almacenará toda la información de barcos y comentarios que crea cada usuario.

6.2.3. Calidad de los datos

El sistema, en esta primera versión de su implementación, no tiene un sistema de base de datos distribuido, puesto que tiene una sola base de datos centralizada donde se encuentran todos los datos, ya que se prevé un bajo número de usuarios, y por tanto un bajo volumen de datos. Se realizarán análisis en más detalle cuando aumente el número de usuarios que lleve probablemente al uso de bases de datos distribuidas.

Al tener una única base de datos nos podemos asegurar de la consistencia y calidad de los mismos, ya que no necesitará una actualización compleja de los datos cuando haya cambios.

Sin embargo, puesto que hay datos como los puntos que pueden ser intervenidos por los administradores del sistema, se estudia la posibilidad de añadir una zona de espera con estos cambios, donde se espera a que sean verificados por el equipo y una vez completado el proceso realizar la actualización, para evitar fallos o incongruencias en la vista o puntos de los usuarios.

6.2.4. Propietarios de los datos

Para no crear inconsistencias, ya que los usuarios pueden crear barcos de forma simultánea, se formará un id para los barcos creados por un usuario que siempre contendrá:
el id del usuario creador, más un entero que lo identifique de forma única.

6.2.5. Volúmenes de datos

El volumen de datos del sistema no se prevé que sea muy grande en su inicio, porque tendrá un número de usuarios pequeño, y ya que no albergará datos de tamaño pesado, sino que en su totalidad se va a tratar de objetos representados con texto.

Al no tratarse de un volumen de datos grande, no deben presentarse problemas en cuanto a copias de seguridad, ni tiempo de extracción para la búsqueda en la base de datos, o por restricciones de almacenamiento.

6.2.6. Inquietudes de las partes interesadas

Tipo de interesado	Inquietudes
Arquitecto	Diseño de una arquitectura, patrones de diseño, seguridad, estructura y diseño de la información
Desarrollador	Interesados en saber cómo se traducirán los modelos del arquitecto de datos en bases de datos reales e interfaces de información (en tiempo real, por lotes...)
Director de proyecto	Gestión de recursos y costes involucrados en el mantenimiento y calidad de los datos.
Experto	Estructura interna, cualidades de datos imprescindibles, filosofía del diseño funcional,

Administrador del sistema	Interesados en saber cómo se gestionarán y apoyarán estos componentes del sistema en el mundo real
Técnico de pruebas	No se centran demasiado en los detalles de la arquitectura de la información pero pueden encontrar útil una comprensión básica de los principios y estrategias clave. Se centran más en la calidad de los datos.
Usuario	Preocupados por los aspectos funcionales de la arquitectura de la información y por las cualidades visibles para el usuario como la latencia, antigüedad, calidad de los datos y la gestión y recuperación de las transacciones

6.3 Modelos

6.3.1 Modelos estáticos de los datos

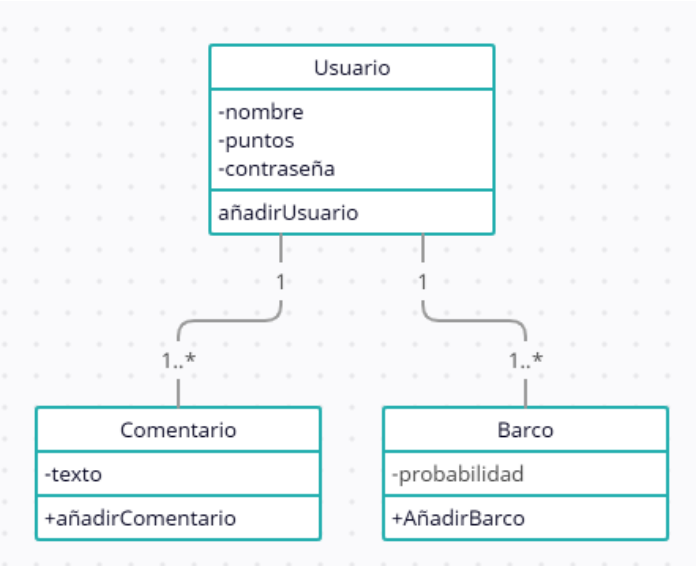
Los modelos estáticos de estructura de datos analizan la estructura estática de los datos, teniendo en cuenta los elementos de datos importantes y las relaciones entre ellos.
Los modelos de clases desempeñan un papel similar al de los modelos entidad-relación, pero para el mundo orientado a objetos. Modelan elementos de datos y sus partes constituyentes y las relaciones estáticas entre ellos.

Actividades:

- Un proceso llamado normalización reduce el modelo a su forma más pura, en la que no hay información repetida, redundante o duplicada. Es raro que los modelos relacionales se lleven más allá de la tercera forma normal, y desde el punto de vista del arquitecto, a menudo es más útil modelar algunos de los datos sin normalizar.

- El análisis de dominio examina los atributos de los elementos de datos y las reglas que definen sus valores permitidos.

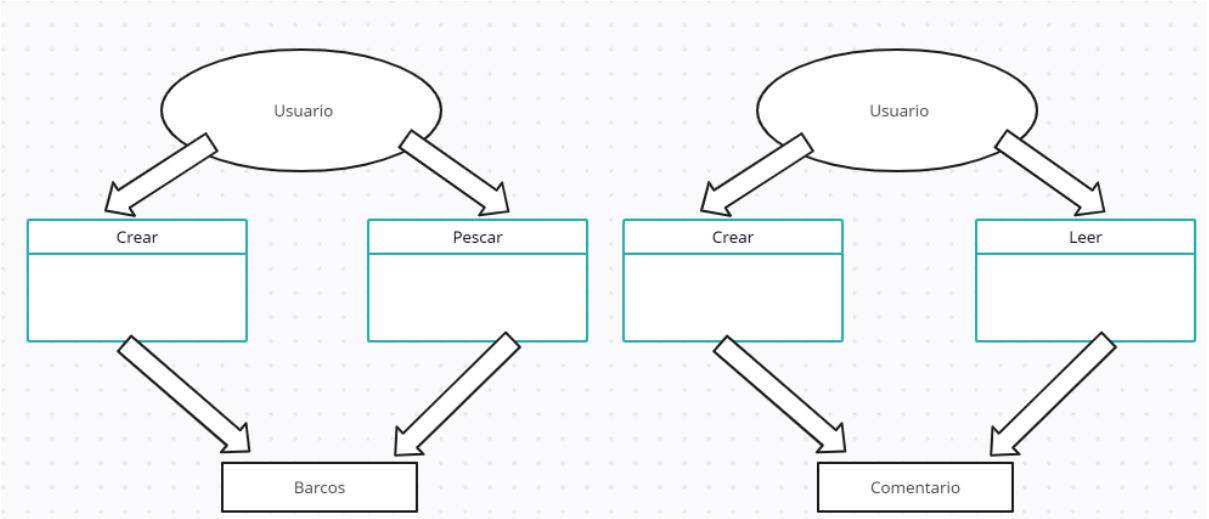
- Para derivar modelos de clase se utilizan técnicas como la descomposición estructural o la agregación. La descomposición estructural consiste en dividir un elemento en piezas coherentes más pequeñas, mientras que la agregación es el proceso inverso, crear un nuevo elemento combinando otros elementos similares



6.3.2 Modelo de flujo de la información

Los modelos de flujo de información analizan el movimiento dinámico de la información entre los elementos del sistema y con el mundo exterior. Estos modelos identifican los principales elementos arquitectónicos y los flujos de información entre ellos. Cada flujo representa datos transferidos de un componente a otro, es decir una interfaz de datos. A cada flujo se asocia una dirección, el alcance de los datos transferidos, la información volumétrica y el medio por el que se intercambian los datos.

- Los rectángulos grandes representan procesos que manipulan datos
- Los rectángulos estrechos representan almacenes de datos
- Las flechas representan flujos de datos
- Las elipses representan entidades externas



- Los Usuarios pueden crear sus propios barcos y se informa a la base de datos de ello.

- Los usuarios pueden crear y leer comentarios, escribiendo en la base de datos o obteniendo información de ella.

6.3.3 Modelos de propiedad de datos

Los modelos de propiedad de datos definen el propietario de cada elemento de datos de la arquitectura. En este contexto, “elemento de datos” suele significar entidad u ocasionalmente atributo, aunque se pueden modelar partes más complejas.
Cuando dos sistemas pueden modificar el mismo dato, hay que desarrollar estrategias de resolución de conflictos, como las que se describen a continuación, para garantizar que se respetan las reglas de negocio y que la información queda en un estado coherente.

- Aceptar siempre la última actualización
- Mantener varias copias del mismo elemento de datos, etiquetadas con sus fuentes
- Mantener un historial de cambios en los datos en lugar de limitarse a la última versión de los mismos
- Crear reglas más complejas en función de los datos modificados y de la naturaleza del cambio
- Cuando se registran varios valores, requerir la intervención manual para solucionar el conflicto
- Rechazar por completo las actualizaciones conflictivas
- Utilizar una combinación de estrategias

6.4 Problemas y errores comunes

6.4.1. Incompatibilidad de datos

Las incompatibilidades de datos surgen porque los distintos sistemas codifican la información a nivel de campo de maneras distintas. Nuestro sistema usará la misma codificación en todos los sistemas, respetando la tipología de todos los atributos
En cambio, en el ámbito de modelos de negocio es un poco más complejo la resolución de estos problemas, ya que tendríamos que hacer un tratamiento que puede resultar bastante complejo. Tendríamos que desarrollar un modelo común de alto nivel de la estructura de datos, los atributos de datos clave y sus dominios, además de validar en todas las partes del sistema.

6.4.2. Mala calidad de datos

Si los datos son incoherentes, imprecisos o incompletos no importa lo bueno que sea su modelo de datos, tendrá problemas en el funcionamiento. Para prevenir estos inconvenientes tomaremos las siguientes medidas:

- Validamos pronto su clave sobre la calidad de los datos
- Nos aseguraremos de saber qué datos son importantes y cuáles no.
- Utilizaremos herramientas de calidad de datos para analizar la calidad de los datos existentes
- Identificamos los lugares en los que pueden aparecer datos de mala calidad y desarrollaremos estrategias para tratarlos.

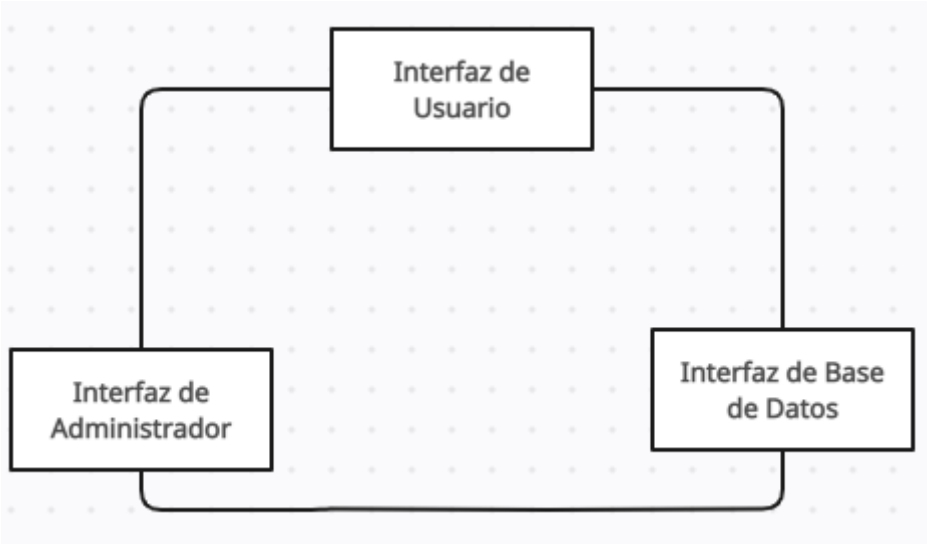
6.4.3. Múltiples fuentes de actualización

Al crear arquitecturas distintas, intentamos que todos los modelos se actualicen en un lugar único, pero esto es prácticamente imposible en la realidad. Como medidas de seguridad se tomarán las siguientes medidas:

- Nuestro modelo de propiedad de los datos es completo y preciso y todos los elementos de datos con múltiples actualizadores están identificados
- Comprendemos donde pueden surgir incoherencias a través de múltiples actualizadores y tenemos en cuenta los puntos críticos en los que se encuentra el elemento de datos incompatibles.
- Llevaremos a cabo estrategias para resolver los problemas, como sustituir las actualizaciones antiguas por otras más recientes, manejar dos copias de datos, etc…

6.4.4. Complejidad de la interfaz

La complejidad de la interfaz crece de forma exponencial a medida que se implementan interfaces nuevas ya que todas las interfaces tienen que estar conectadas entre sí, por esto nosotros apostamos por una interfaz sencilla en la que solo hay 3 interfaces internas y ninguna externa:



6.5 Checklist

- ¿Tiene un nivel de detalle adecuado en sus modelos de datos(no más de 20 entidades)?**Si.**
- ¿Están claramente identificadas las claves de todas las entidades importantes?**Si.**
- Cuando una entidad está distribuida en varios sistemas o ubicaciones con claves diferentes, ¿Están definidas las correspondencias entre las claves?**Si.**¿Dispone de procesos para mantener estas correspondencias cuando se crean elementos de datos?**Si**
- ¿Ha definido estrategias para resolver los conflictos de propiedad de los datos, en particular cuando hay varios creadores o actuadores?**Si.**
- ¿Están claramente identificados los requisitos de latencia y existen mecanismos para garantizar su comportamiento?**Si.**
- ¿Dispone de estrategias claras para garantizar la coherencia de las transacciones en los almacenes de datos distribuidos y equilibra esta necesidad con el coste en términos de rendimiento y complejidad?**Si.**
- ¿Dispone de mecanismos para validar los datos migrados y tratar adecuadamente los errores?**No.**
- ¿Ha definido la capacidad de almacenamiento y procesamiento suficiente para el archivado?**No.**¿Y para restaurar los datos archivados?**No.**
- ¿Se ha realizado una evaluación de la calidad de los datos?**Si.**¿Ha creado estrategias para hacer frente a los datos de mala calidad?**Si.**

7. Descripción detallada de la perspectiva de seguridad

7.1 Aplicabilidad

Veremos los puntos de vista que son más susceptibles de verse afectados por una vulneración de seguridad hacia una capacidad concreta de la aplicación.

Punto de vista	Aplicabilidad
Contextual	El acceso al sistema es por parte de actores. Solo para responder peticiones, otros sistemas van a acceder al nuestro.
Funcional	Permite identificar los elementos funcionales del sistema que se deben proteger. Se van a especificar en el apartado actividades.
Informacional	Permite identificar los datos del sistema que se deben proteger.
De desarrollo	Los desarrolladores software deben conocer restricciones que garanticen la política de seguridad del sistema, tales como no compartición de datos privados entre usuarios, no exponer información de pagos, de ubicación de usuarios, etc.
De despliegue	Utilizando una herramienta externa de Bases de Datos, hacemos un uso seguro y protegido de dicho servicio sin exponer nuestro sistema.
Operacional	Se especificarán operaciones dentro de la aplicación que no supongan una vulneración de la seguridad de los usuarios ni de la propia aplicación, para ello se realizará un mantenimiento, que puede incluir nuevas instalaciones y mejoras, migraciones, monitorización, etc.

7.1.1 Actividades

Paso 1: Identificación de los recursos

Partiendo del punto de vista funcional, los recursos que pueden identificarse según su función del acceso son:

- Gestión de las cuentas de usuarios.
- Gestión de barcos de los usuarios.
- Gestión de peces.
- Gestión de puntos.
- Gestión de comentarios.

Partiendo del punto de vista informacional, los recursos que pueden identificarse según sensibilidad son:

- Registros de cuentas de usuarios.
- Precio de creación de barcos.
- Registro de comentarios.
- Registro de puntos.
- Registro de barcos comprados por los usuarios.

Los principales pueden también clasificarse (en roles) en función de los recursos a los que pueden acceder:

- Usuarios (clientes)
- Gestor de peces.
- Administrador
- Superusuario
- Gestor de base de datos

En esta tabla los propietarios conforman un grupo de personas, los cuales estan dirigidos por un principal:

- Equipo gestión - Administrador y/o gestor de base de datos.
- Equipo de precios - Gestor de base de datos.
- Cliente - Es un usuario individual.

Recursos	Sensibilidad	Propietario	Control de acceso
Registros de cuentas de usuarios.	Robo de información personal o invasión de la privacidad.	Cliente Equipo gestión	Sin acceso directo a los datos.
Precio de creación de barcos (puntos).	Si se modifica de forma indebida puede ocasionar problemas en la experiencia de los usuarios y la propia aplicación.	Equipo de precios	Sin acceso directo a los datos.
Gestión de comentarios.	Si se comenta en el foro incumpliendo las normas puede deteriorar la experiencia de los usuarios y la propia aplicación.	Equipo gestión	Sin acceso directo a los datos.

Registro de barcos comprados por los usuarios.	Se debe controlar para proteger la integridad de los datos.	Equipo gestión	Sin acceso directo a los datos.
Operaciones con cuentas de usuario	Se debe controlar para proteger integridad y acceso a datos privados.	Cliente Equipo gestión	Acceso al expediente individual. Necesita autenticación.
Operaciones de modificación de puntos, comentarios, filtros, barcos o peces.	Es necesario que se controle para proteger la integridad de los datos.	Equipo gestión	Acceso a la modificación de puntos, comentarios, filtros, barcos o peces.. Necesita autenticación. En el que recae la responsabilidad de los cambios.

Paso 2: Definición de la política de seguridad

	Registros de cuentas de usuarios.	Precio de creación de barcos (puntos).	Gestión de comentarios.	Registro de barcos comprados por los usuarios.	Operaciones con cuentas de usuario.	Operaciones de modificación de puntos, comentarios, filtros, barcos o peces.
Administrador de datos	Todo con auditoría.	Todo con auditoría	Todo con auditoría	Todo con auditoría	Todo con auditoría	Todo con auditoría
Gestor de barcos	–	Todo con auditoría	–	Todo con auditoría	–	–
Administrador de precios de barcos (puntos)	–	Todo con auditoría	–	Operaciones de solo lectura	–	–
Empleado de atención al cliente	–	–	–	–	Todo con auditoría	Operaciones de solo lectura
Empleado de gestión	–	–	Todo con auditoría	Operaciones de solo lectura	Todo con auditoría	Todo con auditoría
Usuario registrado	–	Solo lectura	–	Operaciones de solo lectura	Todo	–

Paso 3: Identificación de los amenazas del sistema

Construimos un “modelo de amenazas” en base a cada recurso identificado, considerando cada posible amenaza, las consecuencias que tendría sobre el sistema y la probabilidad de que ocurra dicha amenaza.

Para elaborarlo,respondemos a las siguientes preguntas:

- ¿Quién es probable que quiera saltarse la política de seguridad?:
    - Hackers que quieran conseguir información referente a tarjetas de crédito o información personal de valor para poder suplantar la identidad de otro usuario.
    - Un administrador del sistema que quiera ventajas ilegítimas en su usuario de la aplicación.

- ¿Qué motivación tiene el atacante para atacar el sistema?
    - Hackers: uso fraudulento de una tarjeta de crédito o suplantación de identidad.
    - Vendedores: apropiación indebida de objetos virtuales (barcos / peces / puntos) en la aplicación.

- ¿Cómo tratará de saltarse la política de seguridad?
    - Hackers:
        - Obteniendo datos de la base de datos
        - Obteniendo los datos desde la interfaz Web
        - Obteniendo los datos de un sistema externo
    - Administrador del sistema: robando contraseña / incumpliendo normas de comportamiento legal.

- ¿Cuales son las principales características del atacante (sofisticación, compromiso, recursos, etc.?
    - Hackers: mucha experiencia, atacan a otros sistemas parecidos, dedicación quasi-profesional.
    - Vendedor: poca experiencia y baja motivación, solo por interés lúdico o beneficios monetarios casi imperceptibles.

- ¿Cuáles son las consecuencias de que se salte la política de esta forma?
    - Hackers: la reputación del negocio bajará tanto que lo podría hacer caer en bancarrota.
    - Vendedores: disminución (baja) de las ganancias para el negocio.

Árbol de ataque

A continuación, vamos a hacer el ejemplo de un posible árbol de ataque con el objetivo de suplantar la identidad de un usuario registrado en la aplicación.

1. Extraer detalles de la base de datos del sistema.
    a. Acceder directamente a la base de datos
        i. Mecanismos para descifrar o corromper contraseñas de bases de datos o aprovechar debilidad conocida por información del sistema.
    b. Engaño o soborno a un administrador de la base de datos, para conseguir acceso.
2. Extraer detalles de la interfaz web de la aplicación.
    a. Con ataques a usuarios registrados por medio de “phishing”, para emplear técnica de usurpación de identidad.
    b. Mecanismos para descifrar o corromper contraseñas de usuario.
    c. Acatar el software del servidor del sitio para encontrar vulnerabilidades o aprovechar aspectos software conocidos con los que atacar.

Paso 4: Diseño de los mecanismos de seguridad a implementar

Medidas de seguridad a adoptar (teniendo en cuenta el ejemplo del árbol de ataque).

- No compartir información del software o base de datos sobre los que se encuentra la aplicación.
- Revisiones periódicas del código para detectar vulnerabilidades de seguridad.
- Formación reiterada de seguridad y ética para los usuarios y trabajadores.
- Intentos de ataque al propio sistema a fin de encontrar fallas en la seguridad.
- Los servidores de bases de datos pueden aislarse de la red pública y protegerlos con firewalls y estructuras de red contundentes y seguras, como granjas web.
- Ante pérdida de datos, sería recomendable disponer de copias de seguridad para recuperar la información y que no suponga cambios irrevocables en el sistema.
- Preparar la aplicación web para que no se exponga de forma total ninguna información confidencial (como datos bancarios).

Paso 5: Evaluación de los riesgos de seguridad

Para estimar el impacto económico real que pueden ocasionar las vulnerabilidades de nuestro sistema consultaremos a especialistas, como por ejemplo una gestoría de ciberseguridad. Podemos realizar una aproximación como sigue:

Riesgo	Coste estimado	Probabilidad estimada	Coste teórico
El hacker (o atacante) puede acceder a la base de datos	50000	0,02%	18000
Fallo que permite a los usuarios crear barcos sin coste	10000	0,1%	4500
Suplantación de identidad de un cliente por otro	1000	5%	560

Tras la evaluación podemos concluir que en principio se trata de un nivel aceptable de riesgo de seguridad. Los riesgos no deben suponer una crisis total del sistema.

7.1.2 Tácticas arquitectónicas

Problemas y errores comunes
Lista para capturar todos los requerimientos de seguridad:

- ¿Has identificado los recursos sensibles del sistema? Sí
- ¿Has identificado los conjuntos de principales que necesitan acceder a los recursos? Sí
- ¿Has identificado las necesidades que tiene el sistema de garantizar la integridad de la información? Sí
- ¿Has identificado las necesidades de disponibilidad del sistema? Sí
- ¿Has establecido una política de seguridad para definir las necesidades de seguridad del sistema, junto con los principales y los permisos de acceso que tiene cada uno a cada recurso, y cuándo debe comprobarse la integridad de la información? Sí
- ¿Es la política de seguridad lo más simple posible? Sí
- ¿Has recorrido un modelo formal de amenazas para identificar los riesgos de seguridad del sistema? Sí
- ¿Has considerado tanto las amenazas externas como las internas al sistema? Sí
- ¿Has considerado cómo el entorno de despliegue del sistema alterará las amenazas del sistema? No
- ¿Has recorrido escenarios de ejemplos junto con las partes interesadas en el sistema de forma que puedan entender la política de seguridad planificada y los riesgos del sistema? No
- ¿Has revisado los requisitos de seguridad con expertos externos en seguridad? No

La otra lista proporcionada es directamente aplicable en la DA:

- ¿Has abordado cada amenaza del modelo de amenazas con la profundidad necesaria? Sí
- ¿Has usado lo más posible las tecnologías de seguridad de terceras partes? Sí

- ¿Has realizado un diseñno global integrado de la solución dada para garantizar la
- seguridad? No
- ¿Has considerado los principios estándares de seguridad a la hora de diseñar la infra-estructura de seguridad? Sí
- ¿Es tu infraestructura de seguridad lo más simple posible? Sí
- ¿Has definido una forma de identificar brechas de seguridad y de recuperar el sistema frente a ellas? No
- ¿Has aplicado los resultados de la perspectiva de seguridad a todos los puntos de vista afectados? No
- ¿Han revisado tu diseño de seguridad expertos externos en seguridad? No

8. Descripción detallada, punto de vista de despliegue

PV despliegue → de moda diseño continuo (CI/CD) / DevOps

8.1 Descripción

Detalle	Descripción
Definición	Describe el entorno en el que el sistema será desplegado, incluyendo las dependencias que tenga el sistema en tiempo de ejecución.
Inquietudes	Tipo de hardware requerido, especificaciones y cantidad de hardware, requerimientos de software de terceros, compatibilidad de tecnología, requerimientos network y restricciones físicas.
Modelos	Plataformas de modelos a tiempo real,dependencias de modelos tecnológicos y modelos de red.
Problemas y errores comunes	Independencias no claras, tecnología no probada, consideraciones tardías del sistema.

8.2 Inquietudes

8.2.1 Tipo de hardware requerido

Como hardware necesario requerido suponemos que trabajos con unos 80 usuarios de media de forma paralela, aunque es obvio que esta cifra, tanto de usuarios como al de características hardware ha de ser escalable, ya que en cualquier momento puede crecer de forma exponencial el número de usuarios:

Las características hardware con las que contamos en un principio para gestionar la aplicación con un supuesto de 80 usuarios:

- El hardware donde se ejecutará el sistema de nuestra aplicación de barcos:
    - Procesador: Procesador de unos 4 núcleos con al menos 2.5GHz de ciclo de reloj, para poder gestionar la carga de trabajo.
    - Memoria RAM: Unos 8GB de memoria para poder ejecutar la aplicación y manejar la carga de trabajo de forma eficiente.
    - Almacenamiento: Contaremos con unos 128GB de capacidad para alojar el sistema operativo, la aplicación y los datos asociados.
- El hardware donde se ejecutará nuestra base de datos en mysql con docker será muy semejantes, partiendo de la base propuesta antes sobre la escalabilidad:
    - Procesador: Procesador de unos 4 núcleos con al menos 2.5GHz de ciclo de reloj, para poder gestionar la carga de trabajo.
    - Memoria RAM: Unos 8GB de memoria para poder ejecutar la aplicación y manejar la carga de trabajo de forma eficiente.
    - Almacenamiento: Contaremos con unos 128GB de capacidad para alojar el sistema operativo, la aplicación y los datos asociados.
- Otras consideraciones a tener en cuenta:
    - Red: Tendremos una conexión de red de alta velocidad con un ancho de banda que nos pueda garantizar también una respuesta en tiempo real que se acomode a la aplicación.

8.2.2 Especificación y cantidad de hardware requerido

Para la especificación, se ejecuta en servidores con contenedores docker con un sistema operativo Linux.

En cuanto a la cantidad de hardware, en este caso dependerá mucho de las interacciones que supongamos con nuestro sistema, dependerá principalmente del número de clientes que estimemos por temporadas, como se especificó en el anterior apartado.

8.2.3 Requisitos de software de terceros

En cuanto a los requisitos de terceros no son necesarios, ya que no contamos con software de terceros para nuestra aplicación actualmente, lo unico que podriamos llegar a contemplar es el hecho de que se pueda acceder a nuestra API por otras aplicaciones por otros motivos varios:

- Tratamiento de datos de nuestra red social por otras aplicaciones
- Vender los datos a otras organizaciones
- etc…

8.2.4 Compatibilidad tecnológica

Para la compatibilidad tecnológica deberemos tener en cuenta que a la hora de obtener datos de la base de datos debemos contar con compatibilidad de versiones tanto para el propio acceso a la base de datos como para la trata de estos, ya que se utilizaran diferentes bibliotecas para la gestión de usuarios por ejemplo.

Además que la aplicación ha de poder ejecutarse desde distintos dispositivos, los cuales deben contar con algunas garantías para evitar posibles fallos asociados con diferentes versiones de software.

8.2.5 Requisitos de red

Para lograr una concurrencia real dentro del sistema y darle la sensación de chat a los usuarios necesitamos diferentes requisitos de red para alcanzar la interacción a tiempo real que experimentan los usuarios de nuestro sistema.

También a la hora de definir los enlaces requeridos entre máquinas usaremos equilibrio de carga en caso de necesidad de contar con más de una máquina que ofrezca servicios para no sobrecargar ninguna y gestión de firewall para una mayor seguridad de paquetes.

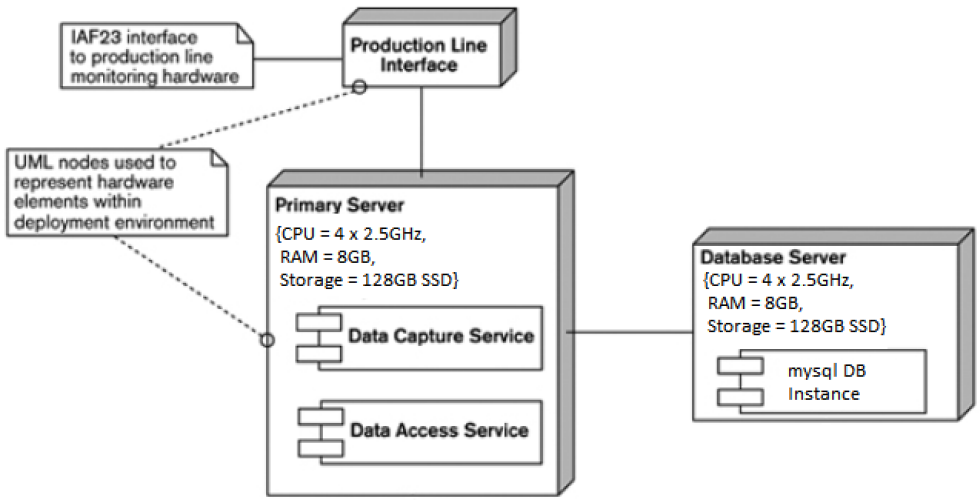
8.2.6 Inquietudes de partes intersados

Tipo de interesado	Inquietudes
Arquitecto	Diseño de una arquitectura, patrones de diseño, seguridad y mantenimiento del software.
Desarrollador	Calidad del diseño y de la estructura interna, además de las capacidades funcionales e interfaces externas.
Director de proyecto	Planificación del proyecto, gestión de recursos y costes, comunicación con el cliente, general…
Experto	Estructura interna, cualidades del diseño básica, filosofía del diseño funcional,
Administrador del sistema	La filosofía de diseño funcional básica, interfaces y estructura interna.
Técnico de pruebas	Estructura interna, capacidad funcional, cualidades de diseño e interfaces externas.

8.3 Modelos

8.3.1 Modelos de plataforma de tiempo de ejecución

Diagrama de modelo de plataforma de tiempo de ejecución:



UML no asocia semántica detallada con los nodos o asociaciones y no proporciona diferentes tipos de cada uno. Por lo tanto, el uso efectivo de este tipo de diagrama normalmente se basa en gran medida en estereotipos, valores etiquetados y comentarios para distinguir entre diferentes tipos de nodos y enlaces.

Actividades

- **Diseñar el entorno de despliegue,** hay que identificar los servidores clave del sistema, los requisitos de los sistemas propiamente dichos y enlaces de red necesarios.

  Como servidor clave de nuestro sistema en principio tendremos solo el principal, ya que no necesitamos de más para controlar la red social con 80 usuarios.

- **Asigne los elementos al hardware,** una vez que tenga un entorno de implementación propuesto, debe encontrar un lugar de propia acomodación en él para cada uno de sus elementos del software.

  A la hora de asignar elementos software a nuestros componentes hardware, contaremos con nuestro servidor con sus características hardware para implementar nuestra aplicación, mientras que contaremos con nuestro servidor de base de datos con su hardware especificado también que contendrá nuestro sistema de contenedores docker, donde implementaremos nuestra base de datos.

- **Calcule los requisitos de hardware,** esta actividad normalmente comienza con una estimación inicial antes del diseño del entorno de implementación inicial, seguida de un proceso iterativo de refinamiento a medida que avanza la arquitectura y el diseño.

  Ya especificado en los apartados anteriores.

- **Llevar a cabo una evaluación técnica,** para diseñar y estimar el entorno de implementación, es posible que deba realizar una serie de ejercicios de evaluación técnica, como el desarrollo de elementos de prototipo, puntos de referencia y pruebas de compatibilidad.

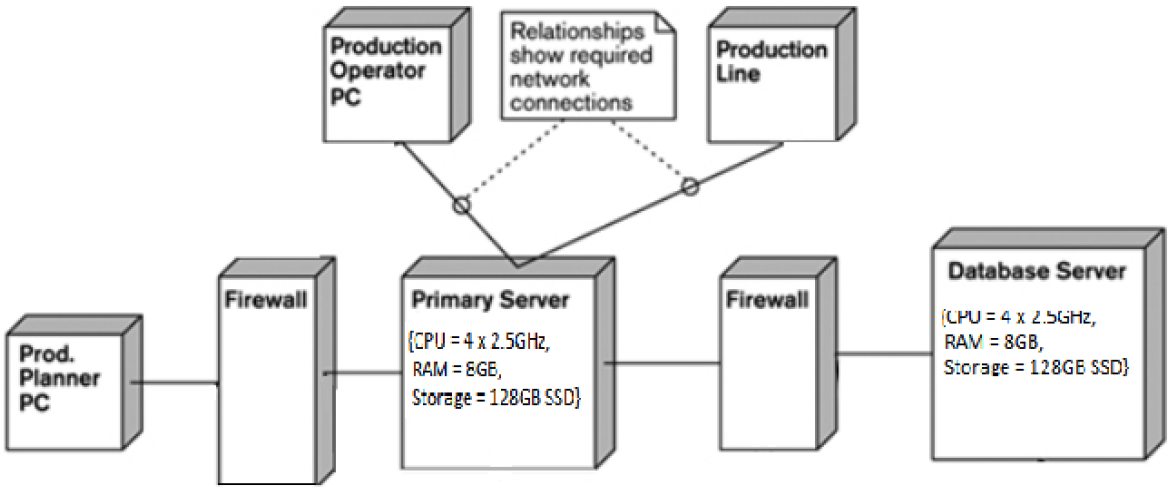
  Se realizarán los ejercicios propuestos tanto de evaluación técnica, como software con el prototipo que se proponga, para garantizar la calidad y estabilidad de nuestro sistema.

- **Evaluar las restricciones,** es raro que los arquitectos definen una vista de implementación sin restricciones externas. Las restricciones que encuentre pueden ser estándares formales, pautas informales o simplemente restricciones implícitas que sabe que existen.

Nosotros al tratar con una red social contamos con muchas restricciones por parte de estándares, como es por ejemplo el de trata de la información, ya que existen leyes que regulan de forma muy estricta todo el tratamiento de datos de los usuarios, como son las políticas de privacidad, etc…

8.3.2 Modelos de red

Diagrama de modelo de red:



Las notaciones comunes que se utilizan para capturar el modelo de red incluyen el uso de UML y los diagramas tradicionales de cajas y líneas. Estas opciones se analizan brevemente en esta subsección.

Actividades

- **Diseñar la Red,** el diseño de la red generalmente se maneja por separado del hardware de la computadora porque están involucrados diferentes especialistas.

  El diseño de red en nuestro caso lo implementaremos para establecer la conexión entre la base de datos y el servidor donde se ejecutará nuestra aplicación de chat de usuarios de barcos.

- **Estimar la Capacidad y Latencia,** parte del diseño de su red lógica es estimar la capacidad y la latencia que espera entre cada nodo. La precisión no es tan importante en esta etapa, pero es importante una estimación realista de la magnitud del tráfico que se transportará y el tiempo esperado de ida y vuelta.

  La capacidad la especificamos ya anteriormente de cada uno de los nodos, como son el del servidor donde se ejecuta la aplicación, la del servidor de base de datos y el de los clientes que accedan a la aplicación, cuya latencia será la necesaria como para poder sostener el sistema de chat de barcos en tiempo real.

8.3.3 modelos de dependencia tecnológica

Dependencias de software del nodo del servidor principal:

Component	Requires
Data Access Service	HP-UX 64-bit 11.23 + patch bundle B.11.23.0703 HP aCC C++ runtime A.03.73
Data Capture Service	HP-UX 64-bit 11.23 + patch bundle B.11.23.0703
	HP aCC C++ runtime A.03.73
	Oracle OCI libraries 11.1.0.7
HP aCC C++ Compiler & Runtime	HP patch PHSS_35102
	HP patch PHSS_35103
Oracle OCI 11.1.0.7	HP-UX optional package X11MotifDevKit.MOTIF21
	HP-UX patch PHSS_37958

Actividades

- **Analizar las dependencias en tiempo de ejecución,** este proceso pese a ser tedioso es necesario y ha de hacerse mayoritariamente de forma manual, aunque podemos ayudarnos con las especificaciones que se nos aporte por parte de cada uno de los componentes de terceros, para así asegurar compatibilidad con el software.

  Las dependencias en tiempo de ejecución como las que hemos puesto en la tabla serán totalmente necesarios (suponiendo que utilizaramos los mismos software y hardware de la tabla), para la correcta implementación de nuestro sistema.

- **Llevar a cabo una evaluación técnica,** para documentar correctamente las dependencias.

  Se evaluará de forma técnica los sistemas, para poder documentar como implica el punto de forma correcta todas sus dependencia.

- **Relación entre modelos,** para sistemas complejos, podemos contar con varios modelos estrechamente relacionados en lugar de solo un modelo.

  En nuestro caso contamos con un modelo en la aplicación flutter, además de otro modelo en la base de datos que es donde acceden los clientes para estar actualizados respecto a las actualizaciones del mismo.

8.4 Problemas y errores comunes

Problema	Bajo Control
Dependencias poco claras o inexactas	X
Topología no probada	
Acuerdos de nivel de servicio inadecuados o faltantes	
Falta de conocimientos técnico especializado	X
Consideración tardía del entorno de implementación	X
Ignorar las complejidad entre sitios	X
Provisión de espacio libre apropiado	X
No especificar un entorno de recuperación ante desastres	

8.5 Checklist

• ¿Ha asignado todos los elementos funcionales del sistema a un tipo de elemento en su plataforma de tiempo de ejecución?**Si** ¿Los ha asignado a dispositivos de hardware específicos si corresponde? **No, ya que se trata de elementos funcionales genéricos como el de inicio de sesión, que se encargará de controlar los inicios de sesión y puede ser ejecutado por cualquier servidor.**

• ¿Se comprende completamente la función de cada pieza de su plataforma de tiempo de ejecución? **Si** ¿El hardware o servicio especificado es adecuado para el rol?**Si**

• ¿Ha establecido especificaciones detalladas para los dispositivos de hardware del sistema o los servicios alojados que necesita? **Si** ¿Sabe exactamente cuántos de cada dispositivo o cuánto de cada servicio se requiere? **Si**

• ¿Tiene acuerdos de nivel de servicio para los elementos del entorno de tiempo de ejecución proporcionados por terceros? **Si**¿Las garantías en los acuerdos adecuados para su sistema? **Si**¿Puede probar si las garantías son creíbles o no? **Si**

• ¿Ha identificado todo el software de terceros necesario y ha documentado todas las dependencias entre los elementos del sistema y el software de terceros? **Si**

• ¿Se comprenden y documentan la topología de la red y los servicios requeridos por el sistema? **Si**

• ¿Ha estimado y validado la capacidad de red requerida? **No, ya que como se comenta dependerá estrictamente de la velocidad necesaria del momento, el flujo de clientes, etc… (nada de eso es predecible)** ¿Se puede construir la topología de red propuesta para soportar esta capacidad? **Si, ya que pese a no ser predecible en cualquier momento nos podemos adaptar.**

• ¿Han validado los especialistas en redes que se puede construir la red requerida? **Si**

• ¿Ha realizado pruebas de compatibilidad al evaluar sus opciones arquitectónicas para garantizar que los elementos del entorno de implementación propuesto se puedan combinar como se desee? **Si**

• ¿Ha utilizado suficientes prototipos, puntos de referencia y otras pruebas prácticas al evaluar sus opciones arquitectónicas para validar los aspectos críticos del entorno de implementación propuesto? **Si**

• ¿Puede crear un entorno de prueba realista que sea representativo del entorno de implementación propuesto? **No, por lo mismo que comentado anteriormente, dependerá del momento y del número de usuarios, aunque podría llegar a estimarse y hacerlo con ciertas aproximaciones.**

• ¿Confía en que el entorno de implementación funcionará según lo diseñado? **Si** ¿Ha obtenido una revisión externa para validar esta opinión? **No**

• ¿Están satisfechos los evaluadores de que el entorno de implementación cumple con sus requisitos en términos de estándares, riesgos y costos? **Si**

• ¿Ha verificado que se puedan cumplir las limitaciones físicas (como espacio físico, energía, refrigeración, etc.) implícitas en el entorno de implementación necesario? **Si**

• ¿Las especificaciones de su hardware y servicio incluyen una cantidad adecuada de espacio libre? **Si**

• ¿Su vista de implementación incluye una especificación de un entorno de recuperación ante desastres, si es necesario? **Si**

9. Perspectiva de evolución

En esta perspectiva atenderemos las posibles preocupaciones que dan lugar la cantidad de cambios a desarrollar

Aplicabilidad

Punto de vista	Aplicabilidad
Contextual	- Mismas interfaces. - Nuevas interacciones - Nuevas entidades externas *(middlewares para gestión de pagos, aumento de número de BD y servidores con mayor rendimiento)*
Funcional	En el caso de ampliar las bases de datos no hay ningún problema debido a que la API para facilitar la conexiones a BD SQL se efectúan de la misma forma, lo mismo sucede con los servidores ofrecen los mismos servicios. En el caso de la gestión de pagos habrá que añadir nuevas funciones para utilizar el middleware y modificar los puntos del cliente en la BD.
Informacional	No se requiere de añadir nueva información sólo modificar la existente.
Concurrente	La estructura concurrente será muy simple, segura.
Desarrollo	En el entorno de desarrollo el único impacto será el de gestionar pagos pero cómo se usa un middleware, sólo se deberá incluir la API y leer la respectiva documentación.
Despliegue	No afecta
Operacional	No afecta

9.1 Gestión de producto

La ruta de evolución que va a seguir de acuerdo a los puntos anteriormente mencionados es la siguiente:

Recompensas por experiencia → Gestión de pagos → Compra/Venta de barcos entre usuarios.

Esta ruta está revisada y planificada en base a los intereses de los clientes y atendiendo a la dificultad que puede suponer su desarrollo.

Si se desease añadir más requisitos para la evolución el algoritmo que se seguiría es teniendo en cuenta interés del cliente/dificultad de desarrollo.

9.1.1 Magnitud de cambio

La magnitud de los cambios actuales no va acarrear cambios extremos de todo el sistema pues sólo será añadir nuevas funcionalidades que si puede llegar a cambiar algo del modelo será muy minucioso

Podemos concluir que cómo los cambios no van a generar problemas tampoco generará costes muy elevados y sobreesfuerzos por parte del equipo.

9.1.2 Dimensiones del cambio

Las dimensiones de cambios que se van a experimentar o pueden llegar a experimentarse son:

- Evolución funcional: Se añadirán nuevos subsistemas que serán el resultado de aplicar los requisitos de evolución, más explícitamente, “recompensas por experiencia, compra/venta de barcos entre usuarios”.
- Evolución de integración: Debemos de tener en cuenta el uso de middlewares y cambios en estos por terceros puede llevar a cabo que tengamos que redefinir algunas partes en las que se utilice.
- Evolución de crecimiento: Durante la salida del producto puede darse el caso de que necesitemos de más recursos para ello seguiremos la planificación de despliegue.

9.1.3 Probabilidad de cambio

Los cambios más probables son los funcionales y en los que más nos vamos a centrar, después le sigue los de crecimiento y por último integración

La correspondencia quedaría de la siguiente manera:

1. Funcionales - Alta probabilidad.
2. Crecimiento - Baja probabilidad.
3. Integración - Baja probabilidad.

9.1.4 Temporización de cambio

De acuerdo a las probabilidades establecidas para cada uno de los cambios la temporización quedaría:

1. Funcionales - entrega por fases.
2. Crecimiento - vagas de cambios a futuros dependiendo de factores externos.
3. Integración - vagas por factores externos. *(dependiendo del interés del usuario final)*

9.1.5 Cuando pagar por el cambio

Desde la filosofía de nuestro equipo de trabajo apostamos por las 2 estrategias, sin embargo, si hacemos tuviésemos que apostar por una sólo sin duda sería la estrategia de sistema los más simple posible, pero, cómo somos conocedores de que en el futuro requerirá de numerosos cambios es el motivo de por qué utilizamos la estrategia de la flexibilidad.

9.1.6 Cambios guiados por factores externos

Los cambios que nos podemos enfrentar por motivos externos serían:

1. Cambios en las funcionalidades de middlewares.
2. Cambios en las interfaces de los servicios que ofrecen host.
3. Cambios en políticas de regulación de datos.
4. Movimientos consensuados por grupos de personas en contra del producto.
5. Cambios en la estrategias.
6. Cambios de organización y estructura de los equipos.

Estos cambios descritos son los que se ocurren la primera vez que se piensan en posibles factores externos aunque debemos de reconocer que existe multitud de factores externos que quizás no se tengan en cuenta en estos instantes y sean importantes.

9.2 Complejidad en el desarrollo

Nuestra planificación en la evolución del proyecto es muy estricta e importante para restar complejidad al sistema, en cada iteración a evolucionar del sistema tenemos unas partes mínimas a desarrollar que satisface con los requisitos de los stakeholders.

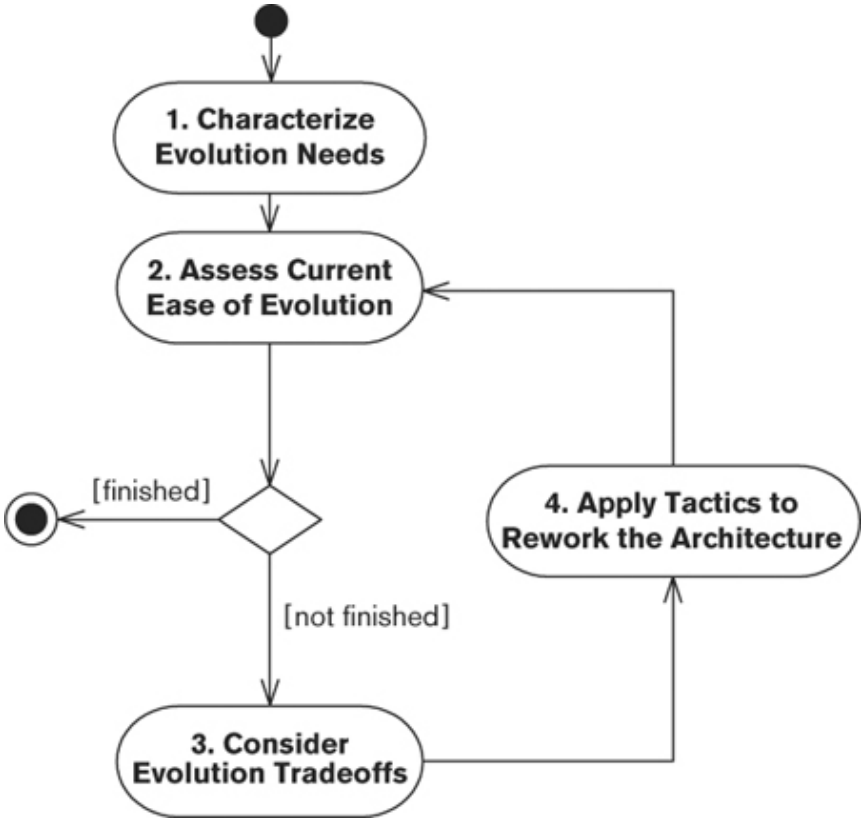
9.3 Preservación del conocimiento

El cambio de personal entre distintos proyectos puede hacer que se le vaya olvidando el conocimiento del proyecto en el que estaban para ello contamos con esquemas muy básicos que recoge todos los aspectos más importantes del proyecto para estimular los recuerdos de la persona que estaba trabajando o bien para facilitar a nuevo personal a entender las ideas del sistema.

9.4 Fiabilidad del cambio

Evitar que los arreglos de bugs generen un impacto negativo y hagan al sistema más complejo, en nuestro equipo contamos con una variedad de tests automáticos que ponen a prueba nuestro sistema, planificaciones de la evolución, pocos o cambios nulos sobre el entorno de trabajo durante la evolución.

9.5 Actividades



9.6 Caracterizar necesidades evolutivas

- Funciones diferidas:
    - Se podrán comprar y vender barcos a otros usuarios. *(Cambio funcional, escala de modelo, alta probabilidad de cambio, a corto plazo)*.
    - Se podrán comprar puntos a través de un plataforma de pago para poder conseguir más puntos *(Cambio funcional/integración, escala de modelo, media probabilidad de cambio, a medio plazo)*.
    - Poder expandir la aplicación a otros países y otros idiomas.(Cambio de extensión/plataforma, alta escala, poca-media probabilidad de cambio, a largo plazo).

- Huecos en los requisitos:
    - Implementación más gráfica de los objetos.(Cambio funcional, alta escala del modelo, media-alta probabilidad, a largo plazo).
    - Disposición de personal en la moderación del foro y gestión de filtros. (Cambio integración, poca probabilidad, inmediato).

- Requisitos vagos:

- Disponibilidad: Los servicios que utilizamos ofrecen alta disponibilidad de sus servidores(24/7) y tolerancia a fallos, a esto le debemos sumar la monitorización de los servidores para notificarnos de posibles fallos, reestructurar cargas, añadir más rendimiento.

- Requisitos abiertos: N/A.

9.7 Evaluación facilidad de evolución en la actualidad

Cada uno de los siguientes requisitos de evolución que han aparecido durante el análisis están ordenados en orden ascendente según su relevancia y probabilidad de cambio.

- Se podrán comprar y vender barcos a otros usuarios. → Dificultad fácil-media, magnitud pequeña, poco riesgo de conjunto de cambio.
- Se podrán comprar puntos a través de un plataforma de pago para poder conseguir más puntos → Dificultad media, magnitud pequeña, poco riesgo de conjunto de cambio.
- Implementación gráfica de los objetos. → Dificultad media-difícil, magnitud grande, alto riesgo de conjunto de cambio.
- Poder expandir la aplicación a otros países y otros idiomas. → Dificultad fácil-media, magnitud grande, poco riesgo de conjunto de cambio.
- Disposición de personal en la moderación del foro y gestión de filtros. Dificultad muy fácil, magnitud mediana, poco riesgo de conjunto de cambio.

9.7.1 Considerar las contrapartidas de la evolución

Tras la evaluación de cada uno de los requisitos de evolución podemos concluir que es un buen nivel de riesgo aceptable, el nivel de confianza ante estos cambios también es alto y se seguirá en el orden en el que se han evaluado.
En el caso de aparecer nuevos requisitos cómo son ordenados según relevancia y probabilidad se colocarán en el sitio correspondiente, puede darse el caso de que vaya surgiendo nuevos y los menos probables no se efectúen para solucionar esto cada que vez que se añada uno nuevo se deberá de analizar de nuevo en base a la situación.

9.8 Vistas afectadas

Las vistas afectadas por la perspectiva de evolución son funcional, integración y despliegue.

9.8.1 Tácticas arquitectónicas

Aislamiento

Los cambios son/serán en la medida de lo posible pequeños y sencillos, en caso de que no se pueda evitar no debería de existir mucho problema debido a que contamos con:

- Encapsulación: cada elemento está bien definido con sus respectivas interfaces flexibles, y con pocos datos internos relacionados con el usuario para evadir la propagación de errores hacia fuera.
- Bajo acoplamiento: cada cambio se asigna a elementos concretos en la medida de lo posible.
- Alta cohesión: cada una de las funciones debe estar fuertemente relacionada con el elemento en cuestión.
- Redundancia: cómo buena práctica de desarrollo la redundancia de código debería ser evitada.

9.8.2 Interfaces extensibles

Las interfaces son los que se llevan los puntos más negativos de los cambios porque una modificación en el modelo de un parámetro lleva a recodificar funciones que utilizaban el elemento ese del modelo con sus antiguos parámetros.

Con la idea de resolver este problema planteamos 2 soluciones:

1. Utilizar API's que trabajen sobre objetos así sería más fácil y flexible la gestión de los parámetros a través del objeto en vez simplemente los parámetros.
2. Utilizar tecnologías compatibles con las interfaces de información cómo XML, JSON… de forma que se puedan extender los mensajes, obtener parámetros esenciales y opcionales.

Optamos por la primera solución debido a que es la más sencilla de desarrollar y no requiere de mucho esfuerzo.

9.9 Técnicas de diseño que facilitan el cambio

Existen 3 patrones básicos para ayudar que el sistema sea más fácil de cambiar:

- Patrones de abstracción.
- Patrones de generalización.
- Patrones de inversión de control. *(No descartamos utilizarlo en el futuro)*

  Usamos 2 patrones, el de abstracción y generalización de forma que esto afectará incrementando la flexibilidad de nuestro sistema.

9.9.1 Aplicar estilos arquitectónicos basados en metamodelos

No aplicaremos este estilo *(consisten en un metamodelo desglosan el procesamiento y los datos del sistema en sus bloques de construcción fundamentales y usan configuraciones de tiempo de ejecución para ensamblarlos en componentes completamente funcionales)* debido a que preveemos que nuestro sistema no requerirá de muchos cambios, además de que este estilo podría afectar al rendimiento de la aplicación.

9.10 Construir puntos de cambio en el software

Desde nuestro equipo creemos que este estilo es el más conveniente para nuestro proyecto debido a que nos permite identificar que elementos del sistema son los más susceptibles a cambios y considerarlos cómo puntos críticos/cambio. Una medida menos extrema que la anterior y que además nos permite conocer qué elementos pueden evolucionar y cuáles no.

En nuestro modelo hacemos uso de un patrón de filtros en el que se puede comprobar el ejemplo de uso de interfaces por parte de otros elementos para aumentar la flexibilidad.

Mediante configuración de momento no tenemos pensado controlar el comportamiento con el uso de un flujo de datos autodescriptivo, tampoco separar el procesamiento de los datos físico y lógico pues los datos no se someten a mutaciones de formato.

Junto a la encapsulación del código y a la no redundancia añadiremos la división de los procesos en pasos de forma que se puedan interpretar los pasos cómo posibles puntos de cambios.

9.10.1 Usar puntos de extensión estándar

Este enfoque está directamente relacionado con el anterior, y es utilizado por el servidor que nos permite conectar con el servicio de hosting a través de la construcción de adaptadores personalizados para integración, de esta forma eludimos la creación de propia de mecanismos para la integración restándole esfuerzo a la hora de desarrollo/evolucionar sobre el sistema.

Hacer cambios fiables

Un cambio en alguno de los módulos que sea malo puede provocar grandes costes y fallos al sistema, para hacer que los cambios sean fiables contamos con:

1. Gestor de la configuración del software: la medida de controlar los cambios a un módulo será el uso de Git, una herramienta que nos permite almacenar en un contenedor el proyecto e ir subiendo cambios en línea, proporciona una gran cantidad de medidas para evitar un mal cambio.
2. Proceso de compilación automatizado: Docker es una buena herramienta que nos permite utilizar la abstracción de un sistema junto con las dependencias que use el proyecto para generar un entorno en el que siempre se va a poder ejecutar de la misma forma en diferentes máquinas.
3. Análisis de dependencias: Utilizaremos herramientas para poder encontrar las dependencias de las diferentes partes del sistema.
4. Proceso de lanzamiento automático, mecanismos para deshacer lanzamientos fallidos, gestión de la configuración del entorno: estas estrategias cómo ya se ha mencionado Docker es capaz de atender las necesidades que tienen por cómo funciona.
5. Pruebas automatizadas: Contamos con pruebas con diferentes niveles según la importancia de cada elemento.
6. Integración continua: Se irá integrando de forma continua y a un bajo ritmo para asegurarse el resultado satisfactorio de cada una de las partes cambiantes.

9.11 Preservar entornos de desarrollo

A la hora del desarrollo a lo largo del tiempo nuestro entorno puede dar lugar a que sea reemplazado y utilizamos numerosas herramientas en una amplia gama y de diferentes versiones haciendo que seamos menos productivos y pudiendo originar errores.

La solución directa a este problema no es más que utilizar una virtualización de un sistema y acoplar las herramientas, dependencias necesarias que vamos a utilizar todos.
De esta forma nos cercioramos que no existirán errores por problemas en el entorno de desarrollo.

9.12 Problemas y errores comunes

Clasificación	

Priorización incorrecta de las dimensiones	NO
Cambios que nunca suceden	POCO PROBABLE
Impactos de evolución en criterios críticos de calidad	NO
Dependencia excesiva de hardware o software específico	SI
Ambientes de desarrollo perdidos	NO
Gestión de lanzamiento ad hoc	SI

9.13 Lista de comprobación

1. ¿Has considerado qué dimensiones evolutivas son más importantes para tu sistema? **Sí**
2. ¿Confías en que has realizado un análisis suficiente para confirmar que tu priorización de las dimensiones evolutivas es válida? **Sí**
3. ¿Has identificado cambios específicos particulares que se requerirán y la magnitud de cada uno? **Sí**
4. ¿Has evaluado la probabilidad de que cada uno de esos cambios sea realmente necesario? **No, la probabilidad de que realmente sea efectuada (*algo similar)***

Lista de comprobación para la DA:

5. ¿Has realizado una evaluación arquitectónica para establecer si tu arquitectura es lo suficientemente flexible como para satisfacer las necesidades evolutivas del sistema? **Sí**
6. Cuando el cambio es probable, ¿tu diseño arquitectónico contiene el cambio en la medida de lo posible? **Sí**

7. ¿Has considerado elegir un estilo arquitectónico inherentemente orientado al cambio? **Sí**
8. Si es así, ¿has evaluado los costos de hacerlo? **Sí**
9. ¿Has cambiado los costos de tu apoyo a la evolución por las necesidades del sistema en su conjunto? **No**
10. ¿Alguna propiedad de calidad crítica se ve afectada negativamente por el diseño que has adoptado? **No**
11. ¿Has diseñado la arquitectura para acomodar sólo aquellos cambios que crees que serán necesarios? **Sí**
12. ¿Puedes recrear sus entornos de desarrollo y prueba de manera confiable? **Sí**
13. ¿Puedes construir, probar y lanzar de manera confiable y repetible el sistema, incluida la capacidad de revertir los cambios si salen mal? **Sí**
14. ¿Es tu enfoque evolutivo elegido la opción más barata y menos arriesgada de entregar el sistema inicial y la evolución futura requerida? **No**

10. Autoevaluación

En general, el grupo ha trabajado de manera adecuada y eficiente, pudiendo adoptar un gran nivel de coordinación, algo que es difícil en el desarrollo de un proyecto en el que cada parte depende de las demás, por lo que el trabajo en paralelo de cada integrante se tiene que gestionar lo mejor posible.

Los responsables nos hemos encargado de iniciar el entorno de conversación y debate sobre el proyecto a desarrollar para que el sistema final fuese resultado de un consenso entre todos los integrantes.

Gador Romero Prieto (experto): se ha encargado de hacer entender el trabajo que se iba a desarrollar, la metodología para el mismo y de ayudar al director y arquitecto a refinar ciertos apartados del proyecto, además de resolver cualquier duda que surgiese sobre el mismo.

Sergio Muñoz Gomez (arquitecto): ha desarrollado tanto el documento como la estructura del sistema que se iba a desarrollar. Además, ha revisado cada apartado, comprobando que no se desviase de la idea inicial de la arquitectura.

Diego Velázquez Ortuño (director): ha sido el encargado de iniciar las conversaciones y comprobar la documentación necesaria para desarrollar el proyecto. Además, ha asignado las tareas que ha considerado más adecuadas a las parejas y ha revisado el resultado final.

Pareja **Miguel Tirado Guzmán** y **David Martínez Díaz**:
Se han encargado de desarrollar el punto de vista **Funcional**, siendo éste el más importante ya que de él dependían los demás. Lo han desarrollado de manera rápida y eficiente, además de haber consultado y resuelto sus dudas con la profesora.

Pareja **Raúl Morgado Saravia** y **Elena Ortega Contreras**:
Se han encargado del desarrollo del punto de vista de **Información** y de la perspectiva de **Seguridad**. Ambos saben trabajar bien en conjunto y se coordinan bien entre sí y han podido resolver de manera adecuada la parte de trabajo que se les ha asignado.

Pareja **Jose Luis Rico Ramos** y **Ruben Rosales Tapia**:
Se han encargado de desarrollar el punto de vista de **Despliegue** y de la perspectiva de **Evolución**. Han resuelto todas sus dudas con el equipo y han desarrollado eficientemente su trabajo.