



UNIVERSIDAD  
NACIONAL  
DE LA PLATA

## FACULTAD DE INFORMÁTICA

# TESINA DE LICENCIATURA

**Programa de Apoyo al Egreso para Alumnos con Práctica Profesional Supervisada**

**TÍTULO:** Rediseño de la aplicación web Rightboat para compra y venta de embarcaciones

**AUTOR/A:** Escudero Victoria

**DIRECTOR/A ACADÉMICO:** Marrero Luciano

**DIRECTOR/A PROFESIONAL:** Martínez Fernando

**CODIRECTOR/A ACADÉMICO:**

**CARRERA:** Licenciatura en Sistemas

### RESUMEN

*Rightboat es una aplicación de compra venta de diversas categorías de embarcaciones en la que los usuarios localizados geográficamente en diferentes partes del mundo publican sus embarcaciones con el objetivo de recibir ofertas. El paso del tiempo y la evolución tecnológica ha llevado a que las herramientas utilizadas en la aplicación queden obsoletas. En este trabajo se llevó a cabo el rediseño integral del Front-End de Rightboat. El objetivo principal ha sido la integración de nuevas funcionalidades utilizando tecnología y herramientas actuales.*

### Palabras Claves

*Metodologías ágiles, aplicaciones web, desarrollo Front-End, Ingeniería de Software y Bases de Datos, Integración de proyectos y Diseño web.*

### Conclusiones

*El rediseño de la aplicación representa un proyecto que incluye desafíos diversos y constantes. La integración de potentes herramientas en el marco del Front-End y del Back-End ha dado lugar a una aplicación totalmente renovada. El proyecto no sólo cumple con las expectativas, redefine la esencia de la aplicación y ha transformado significativamente la experiencia del usuario en Rightboat, sentando las bases sólidas para futuras innovaciones en otras funcionalidades de la plataforma.*

### Trabajos Realizados

*Rediseño de aplicación web para compra venta de embarcaciones localizadas en todo el mundo. Dividiendo las tareas en subproyectos organizados por secciones de la aplicación, comenzando por tareas más sencillas hasta las más complejas. Integrando nuevas herramientas y afrontando los desafíos que fueron surgiendo.*

### Trabajos Futuros

*Se proponen a futuro una serie de líneas de posibles trabajos a desarrollar dentro de la aplicación teniendo en cuenta algunas falencias que la misma ha presentado. Entre ellas, mejoras en el motor de búsqueda utilizado actualmente, implementar cambios en el sistema de guardado de búsquedas (Saved Searches), cachear las vistas con el objetivo de mejorar el rendimiento de la aplicación y migrar las plantillas de emails.*



## Índice General

<b>1. Exposición de lo realizado en la PPS .....</b>	<b>2</b>
<b>2. Informe Técnico .....</b>	<b>10</b>
2.1 Resumen .....	10
2.2 Palabras Claves .....	11
2.3 Marco Conceptual .....	11
2.4 Trabajo Realizado .....	14
2.5 Trabajo Futuro .....	24
<b>Referencias Bibliográficas .....</b>	<b>26</b>

## Índice de Figuras

<b>Figura 1.</b> Secuencia de pasos para la publicación de una embarcación .....	<b>3</b>
<b>Figura 2.</b> Formulario para la carga de una embarcación (parte 1).....	<b>4</b>
<b>Figura 3.</b> Formulario para la carga de una embarcación (parte 2).....	<b>4</b>
<b>Figura 4.</b> Combo de selección del país .....	<b>5</b>
<b>Figura 5.</b> Combo de selección de estado/provincia .....	<b>5</b>
<b>Figura 6.</b> Captura de pantalla de la carga de imágenes de una embarcación .....	<b>6</b>
<b>Figura 7.</b> Captura de pantalla de la selección del plan de suscripción .....	<b>7</b>
<b>Figura 8.</b> Captura de pantalla perteneciente al proceso de pago .....	<b>8</b>
<b>Figura 9.</b> Ventana modal de recupero de contraseña .....	<b>9</b>
<b>Figura 10.</b> Ventana modal de inicio de sesión .....	<b>9</b>
<b>Figura 11.</b> Ventana modal de registro de usuario .....	<b>9</b>
<b>Figura 12.</b> Estructura de componentes botones .....	<b>16</b>
<b>Figura 13.</b> Implementación de DarkButton .....	<b>16</b>
<b>Figura 14.</b> Ejemplos de DarkButton .....	<b>17</b>
<b>Figura 15.</b> Código React Js para utilizar un DarkButton .....	<b>17</b>
<b>Figura 16.</b> Componente PrimaryButton .....	<b>17</b>
<b>Figura 17.</b> Instancia de PrimaryButton .....	<b>17</b>
<b>Figura 18.</b> Código de React Js para el llamado al hook personalizado .....	<b>18</b>
<b>Figura 19.</b> Envío de información del formulario de recupero de contraseña .....	<b>19</b>
<b>Figura 20.</b> Combo de selección del fabricante de una embarcación .....	<b>20</b>
<b>Figura 21.</b> Combo de selección de categorías de una embarcación .....	<b>20</b>
<b>Figura 22.</b> Dropzone de carga de imágenes .....	<b>22</b>

## ANEXO 4 – Formato tesina PAE-PPS

### 1. Exposición de lo realizado en la PPS

Durante la Práctica Profesional Supervisada se llevó a cabo el desarrollo de un proyecto de rediseño del “front end” de la aplicación web Rightboat. Esta aplicación permite la compra y venta de diversas categorías de embarcaciones en las que los usuarios localizados geográficamente en cualquier parte del mundo (en su mayoría en Gran Bretaña), publican sus embarcaciones con el objetivo de recibir ofertas de futuros compradores.

La publicación de una embarcación en Rightboat consiste en seguir una serie de 3 pasos para completar un proceso de negocio (wizard).

En el primer paso, se debe cargar la información principal de la embarcación, esto implica cargar los siguientes datos: modelo, año, categoría, estado, dimensiones, localización, precio, tipo de combustible, descripción e información extra, tal como, marca del motor, modelo del motor, recuento de motores, horas del motor y caballos de fuerza.

En el segundo paso, se accede a un sección de carga de imágenes (dropzone), en donde los usuarios cuentan con la posibilidad de añadir imágenes relacionadas con la embarcación que están publicando. Además, en esta sección se pueden reemplazar las imágenes existentes y/o cambiar el orden de visualización.

En el tercer paso, el usuario selecciona uno de los tres planes disponibles para la publicación de su embarcación. Las opciones disponibles son las siguientes:

- **Premium:** Esta opción consiste en una suscripción mensual, con carga de imágenes de manera ilimitada, la visualización de la publicación en la página principal (home page) y publicidad de la embarcación en los correos electrónicos de difusión.

- **Standard:** Consiste en una suscripción mensual con la posibilidad de cargar imágenes de manera ilimitada.

- **Free Trial:** Comienza con una prueba gratuita que permite cargar imágenes de manera ilimitada por un período de 30 días. Transcurrido este plazo de prueba, automáticamente se actualizará a Standard.

Finalmente, se accede a una página de pagos predefinida y segura, la cual proporciona una experiencia sencilla para el manejo de suscripciones. Desde esta página se hará efectivo el pago de la publicación una vez que el usuario haya ingresado los datos de su tarjeta.

El proyecto estuvo diagramado por partes directamente vinculadas a una vista o funcionalidad de la aplicación, dividiendo en tareas cada etapa del mismo. Se llevaron

a cabo reuniones en conjunto con el equipo de desarrolladores, gerentes de producto y especialista en SEO (Search Engine Optimization), las cuales fueron denominadas “cycle planning”. Estas “cycle planning” consisten en la planificación de sprints (plazo limitado breve en el que un equipo trabaja para completar una cantidad de trabajo establecida) y en donde cada sprint posee una duración de 15 días. Al finalizar cada uno se lleva a cabo una reunión denominada “cycle review”, en donde se analizan aspectos positivos y se plantean mejoras para el próximo ciclo.

En el primer sprint realizado, se abordan tareas directamente vinculadas a rediseñar tanto la barra de navegación (navbar) como el pie de página (footer) de la aplicación. Al contar con dos entornos de tecnologías diferentes, tanto el actual como el implementado para llevar adelante el rediseño, fue necesario desarrollar ambas secciones (navbar y footer) utilizando dos plantillas diferentes (*html.slim* [20] y *html.erb* [21]). El objetivo de esto fue garantizar una transición fluida y coherente a medida que avanzó el proceso de rediseño. Además, la aplicación mantiene una apariencia uniforme, evitando una discrepancia entre las vistas y logra una experiencia consistente para el usuario.

Durante el segundo sprint, se trabajó en los pasos a seguir (wizard) para la creación de anuncios. A este subproyecto, dentro del rediseño general de la aplicación, se lo denominó “For Sale By Owner” (FSBO).

El primer paso de este subproyecto, fue el desarrollo de una página inicial con objetivos publicitarios y de SEO, es decir, se identifican e incluyen en la vista palabras estratégicas relacionadas al contenido de la aplicación (“boat”, “buy”, “sell”) que impactan en el posicionamiento de Rightboat en los motores de búsqueda. Además, posee comentarios de los usuarios. Estos comentarios se visualizan en una sección especial en la cual se puede iterar sobre ellos. Dependiendo de si el usuario ha iniciado sesión, existe la posibilidad de realizar el registro correspondiente previo acceso al wizard.

Para mostrar los pasos que el usuario debe seguir al momento de realizar la publicación de una embarcación y su estado actual, se creó una barra de progreso que incluye las tres etapas. A continuación, en la figura 1, se presenta una captura de pantalla de la misma.

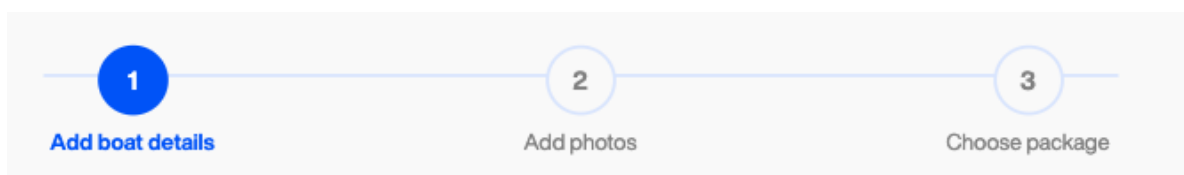


Figura 1. Secuencia de pasos para la publicación de una embarcación.

El primer paso consiste en la carga de la información sobre la embarcación a publicar “Add boat details”. A continuación, en la figura 2 y figura 3, se presentan dos capturas

de pantalla del formulario que brinda el sistema para cargar los datos de una embarcación.

The screenshot shows the 'Rightboat' website interface for 'Sell Your Boat'. At the top, there are navigation links: 'Boats For Sale', 'Research & Advice', and 'Sell Your Boat' (highlighted). A progress indicator shows three steps: 1. Add boat details (active), 2. Add photos, and 3. Choose package. Below the progress indicator, the heading 'Step 1: Add boat details' is followed by a sub-heading 'Main boat information'. A note states: 'Your boat will be advertised to our worldwide audience. Make the very best impression, and give as much information as you can about your boat.' The form contains the following fields:

- Make \***: Select make (dropdown)
- Model \***: Select model (dropdown)
- Boat type \***: Select boat type (dropdown)
- Boat category \***: Select boat categories (dropdown)
- Length \***: Enter length (text input) and meters (dropdown)
- Hull material \***: Select hull material (dropdown)
- Engine type \***: Select engine type (dropdown)
- Fuel type \***: Select fuel type (dropdown)
- Boat location (Country) \***: Select boat location (country) (dropdown)
- Boat location (State / City) \***: Select boat location (state / city) (dropdown)

Figura 2. Formulario para la carga de una embarcación (parte 1).

The screenshot shows the 'Rightboat' website interface for 'Sell Your Boat', continuing from Step 1. The heading 'Step 2: Add boat details' is followed by a sub-heading 'Extra information'. A note states: 'Add a competitive sale price. Price is a key element on the decision to buy a boat.' The form contains the following fields:

- Price \***: Enter price (text input)
- Currency \***: EUR (dropdown)
- Year built \***: Enter year built (text input)
- Description \***: Large text area for description. A note below states: 'Limit 20000 characters. This short accurate description will appear on the advert details page, it will be the first thing people see, so make a great first impression!'.
- Engine (optional)**: Type, Fuel, Mode, Horsepower and Hours. (dropdown)
- Engine make**: Select engine manufacturer (dropdown)
- Engine model**: Select engine model (dropdown)
- Engine count**: Enter engine count (text input)
- Engine hours**: Enter engine hours (text input)
- Engine horse power**: Enter engine horse power (text input)
- Other information (optional)**: Accommodation, Equipment, Electronics. (dropdown)

Figura 3. Formulario para la carga de una embarcación (parte 2).

Dentro del formulario de carga, el usuario debe incluir información en campos obligatorios, los cuales están debidamente señalados, otros son de libre elección.

A continuación, en las figuras 4 y 5, se muestran dos campos de selección para cargar la ubicación de la embarcación. El usuario realiza la selección del país correspondiente (por ejemplo Argentina), lo que desencadena una subconsulta en la base de datos para obtener la división geográfica de dicho país. Posteriormente, se deberá seleccionar de manera obligatoria la provincia o estado según corresponda.

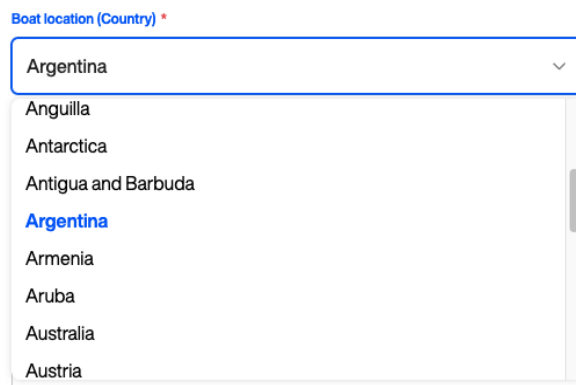


Figura 4. Combo de selección del país.

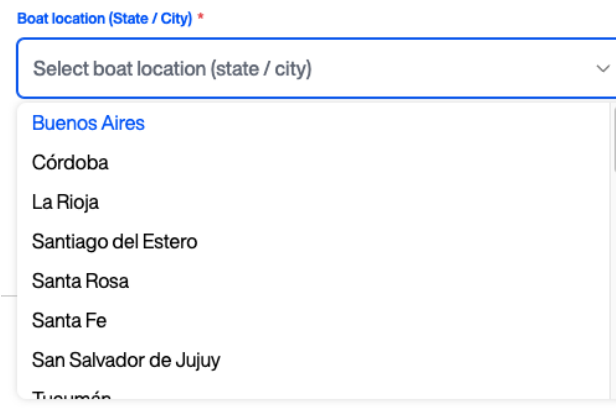


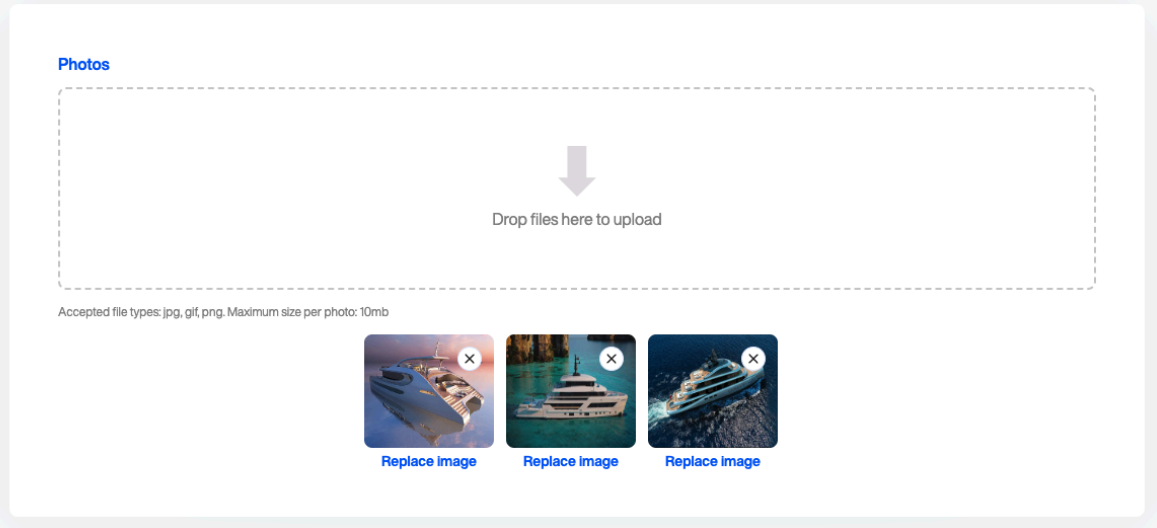
Figura 5. Combo de selección del estado/provincia.

Estos componentes fueron implementados en React Js [1] utilizando la librería Headless UI [2] y estilos con clases Tailwind CSS [3][6].

En el segundo paso “Add photos”, el usuario puede añadir imágenes de la embarcación a publicar. Esta tarea se desarrolló utilizando la librería de carga de archivos de JavaScript [22] llamada Uppy [4]. A continuación, en la figura 6, se presenta una captura de pantalla perteneciente a este segundo paso.

**Step 2: Add your photos**

Let's connect you with the buyer. Great photos are key in increasing inquiries to your listing.



**Figura 6. Captura de pantalla de la carga de imágenes de una embarcación.**

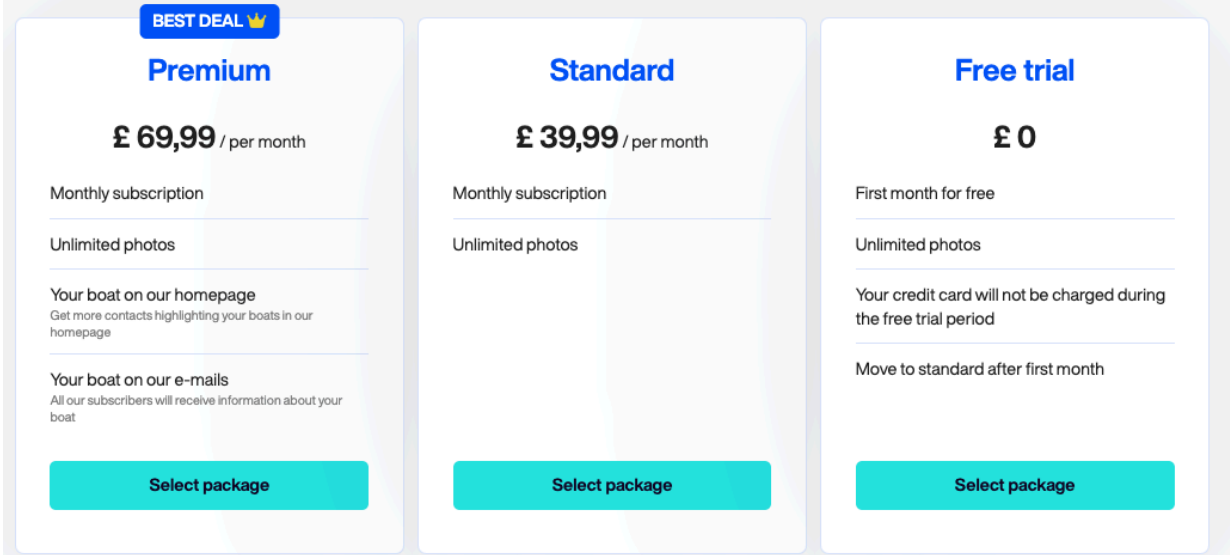
A través de esta librería (Uppy) se puede crear de manera simple una zona de carga de imágenes. Además, brinda la posibilidad de administrar los tipos de archivos permitidos y su tamaño máximo.

En la parte inferior se implementó una lista de archivos que mapea todas las imágenes que han sido cargadas. También existe la posibilidad de eliminarlas, reemplazarlas o cambiar su orden, para esta última alternativa, se utilizó una librería de JavaScript [22] llamada Sortable [5], la cual es una herramienta para crear listas ordenables mediante la técnica de arrastrar y soltar en navegadores modernos y dispositivos táctiles.

En el tercer y último paso, el usuario debe seleccionar el plan de suscripción a la aplicación. A continuación, en la figura 7, se presenta una captura de pantalla correspondiente a la selección del plan de suscripción.

### Step 3: Choose your package

You will be charged on a montly basis until you unpublish your advert.



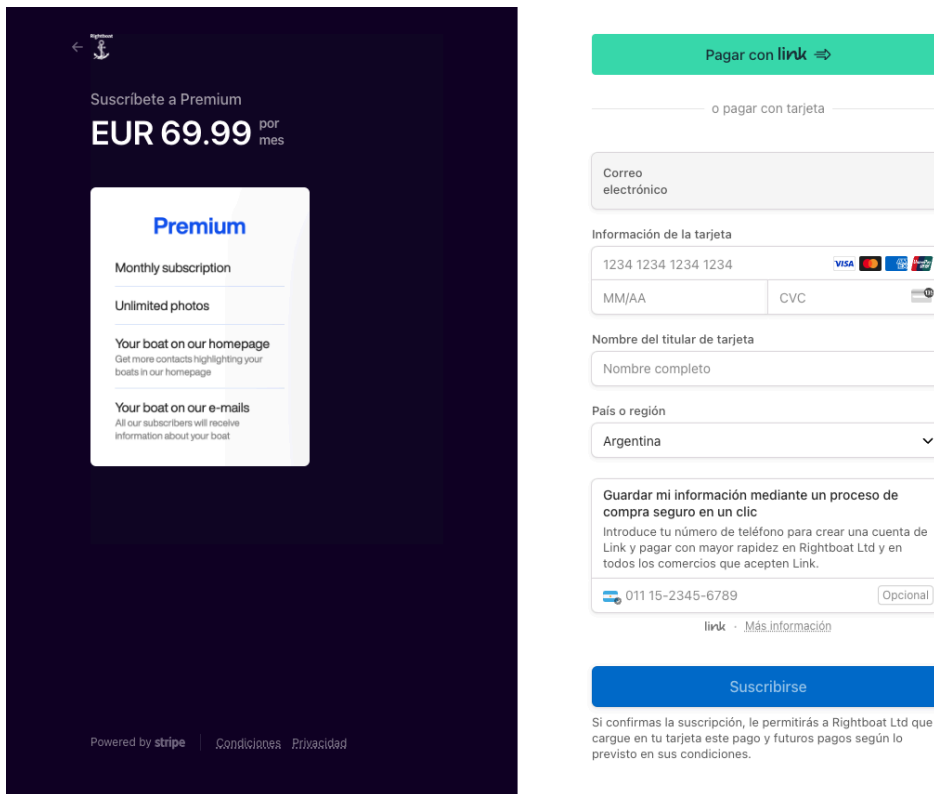
Package	Price	Key Features
<b>Premium</b> (BEST DEAL)	£ 69,99 / per month	<ul style="list-style-type: none"> <li>Monthly subscription</li> <li>Unlimited photos</li> <li>Your boat on our homepage (Get more contacts highlighting your boats in our homepage)</li> <li>Your boat on our e-mails (All our subscribers will receive information about your boat)</li> </ul>
<b>Standard</b>	£ 39,99 / per month	<ul style="list-style-type: none"> <li>Monthly subscription</li> <li>Unlimited photos</li> </ul>
<b>Free trial</b>	£ 0	<ul style="list-style-type: none"> <li>First month for free</li> <li>Unlimited photos</li> <li>Your credit card will not be charged during the free trial period</li> <li>Move to standard after first month</li> </ul>

Figura 7. Captura de pantalla de la selección del plan de suscripción.

Luego de seleccionar un paquete, el usuario será redirigido a un proceso de pago (checkout de Stripe<sup>1</sup>) [17]. A continuación, en la figura 8, se presenta una captura de pantalla con la suscripción seleccionada. Se hará efectiva la publicación del barco una vez que el usuario ingrese los datos de su tarjeta y presione el botón “Suscribirse”.

<sup>1</sup> Stripe Checkout es un formulario de pago pre diseñado y optimizado proporcionado por Stripe. Se integra de manera sencilla y segura brindando opciones de pago en sitios web o aplicaciones.



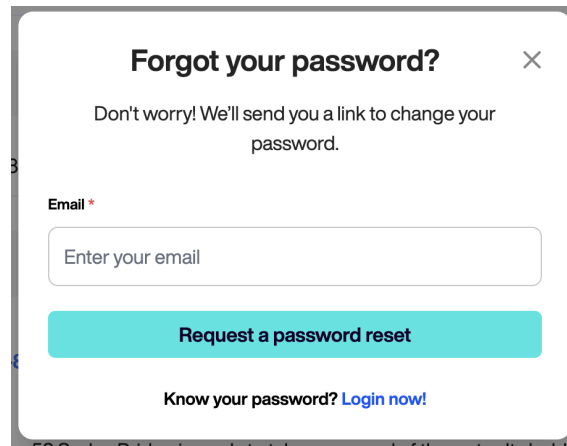


**Figura 8. Captura de pantalla perteneciente al proceso de pago (Checkout de Stripe).**

Durante el tercer sprint se llevó a cabo la implementación de un componente modal (elemento que aparece sobre el contenido de la página o pantalla que se está visualizando) genérico en React Js [1] que fue reutilizado en los distintos componentes.

En la figura 9, se presenta una captura de pantalla para recuperar contraseña. En la figura 10, se presenta un captura de pantalla de inicio de sesión. En la figura 11, se presenta una captura de pantalla para el registro de un nuevo usuario.

Cuando el usuario, sin iniciar sesión, haga click sobre el botón “*Sign In / Join for free*” en la barra de navegación, se despliega el modal de inicio de sesión el cual permite navegar sobre las distintas opciones, tales como, registro de usuario o recupero de contraseña.



**Forgot your password?** ✕

Don't worry! We'll send you a link to change your password.

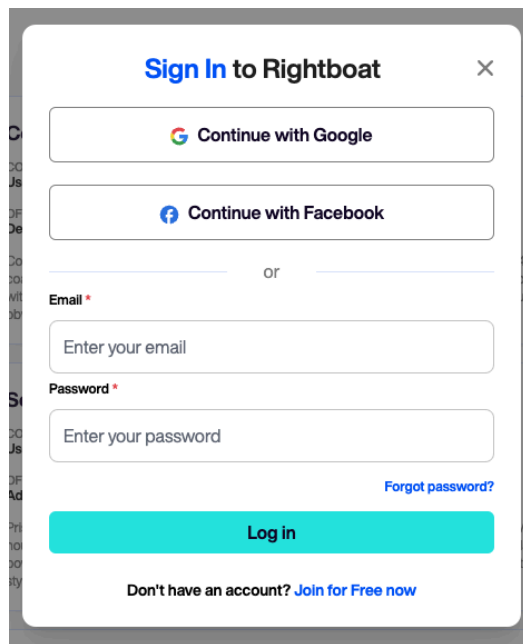
Email \*

Enter your email


**Request a password reset**


Know your password? [Login now!](#)

Figura 9. Ventana modal de recupero de contraseña.



**Sign In to Rightboat** ✕

 Continue with Google

 Continue with Facebook

or

Email \*

Enter your email

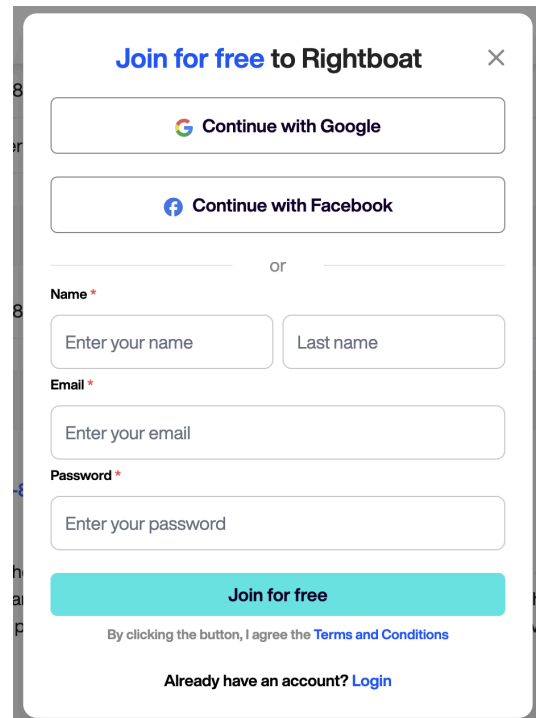
Password \*

Enter your password


[Forgot password?](#)


**Log in**

Don't have an account? [Join for Free now](#)



**Join for free to Rightboat** ✕

 Continue with Google

 Continue with Facebook

or

Name \*

Enter your name Last name

Email \*

Enter your email

Password \*

Enter your password

**Join for free**

By clicking the button, I agree the [Terms and Conditions](#)

Already have an account? [Login](#)

Figura 10. Ventana modal de inicio de sesión.

Figura 11. Ventana modal de registro de usuario.

Todos los formularios cuentan con campos opcionales u obligatorios, los cuales se encuentran señalados con un asterisco en color naranja.

Finalmente, se desarrolló una vista para visualizar los detalles de una embarcación (Boat Details Page). Los usuarios interesados pueden ver la información previamente cargada en el wizard. Esta vista cuenta con un conjunto de imágenes del barco.

## 2. Informe Técnico

### 2.1 Resumen

A inicios del año 2023, comienza el rediseño integral de la aplicación, dando prioridad a las mejoras urgentes. Conforme pasaron los meses, surge el proyecto del rediseño del "front end" titulado como "Rediseño de la aplicación web Rightboat para compra y venta de embarcaciones".

El proyecto se compone de varios subproyectos, página de detalles del barco (Boat Details Page), wizard de creación de anuncios (FSBO), barra de navegación (Navbar), pie de página (Footer), página de Inicio (Home Page), formularios modales de registro e inicio de sesión (Sign In/Up).

Durante cada sprint, los miembros del equipo completan una serie de tareas asignadas a cada uno. En colaboración, los desarrolladores tanto del "front end" como del "back end" siguen las indicaciones de los gerentes del producto y abordan los errores reportados por el asegurador de calidad.

Debido a su menor complejidad los primeros elementos en ser implementados dentro del proyecto fueron la barra de navegación (navbar) y el pie de página (footer). Estos componentes, al ser más simples en su estructura y funcionalidad se convirtieron en los subproyectos iniciales en ser desarrollados. Lo que permitió a los integrantes de Rightboat familiarizarse con el proceso de trabajo colaborativo, establecer pautas de diseño y desarrollo, y sentar las bases para abordar etapas posteriores más complejas en el proyecto.

Seguido a esto, se puso en marcha el rediseño del proceso de publicación de embarcaciones (FSBO), el cual plantea desafíos más complejos. Esta fase del proyecto demandó un esfuerzo adicional por parte de los desarrolladores, quienes tuvieron que dedicar tiempo a investigar nuevas metodologías y herramientas para abordar con éxito cada tarea.

Se integraron tecnologías modernas y se introdujo un nuevo sistema de pagos, el cual requirió un detallado análisis de la plataforma Stripe [17]. En ciertas tareas, fue necesario contactar al equipo de soporte para resolver ciertos aspectos técnicos, como la gestión de claves de acceso o fallos en algunos mecanismos de pagos.

Como última tarea se desarrolla la vista de detalles del barco junto con los distintos formularios modales de la aplicación.

El rediseño de Rightboat representa un proyecto que incluye desafíos diversos y constantes. La integración de potentes herramientas en el marco del "front end" y del "back end" ha dado lugar a una aplicación totalmente renovada. El proyecto no sólo cumple con las expectativas, redefine la esencia de la aplicación y ha transformado

significativamente la experiencia del usuario en Rightboat, sentando las bases sólidas para futuras innovaciones en otras funcionalidades de la plataforma.

## 2.2 Palabras Claves

Metodologías ágiles, aplicaciones web, desarrollo Front-End, Ingeniería de Software y Bases de Datos, Integración de proyectos y Diseño web.

## 2.3 Marco Conceptual

El proyecto “Rediseño de la aplicación web Rightboat para compra y venta de embarcaciones” representa un desafío dinámico dentro del entorno laboral de la empresa con el objetivo de rediseñar una aplicación desarrollada con tecnologías que han quedado obsoletas.

La elección de migrar de CoffeeScript [7] a React Js [1] surge de la necesidad de mejorar la eficiencia y el rendimiento de la aplicación. Las métricas de velocidad de cada página se tienen en cuenta y son monitoreadas permanentemente por el equipo técnico durante las distintas etapas del rediseño, utilizando herramientas como gtmatrix [8] y la gema<sup>2</sup> de rails, rack-mini-profiler [9].

React Js [1], con su arquitectura basada en componentes, permite una mejor organización del código y una mayor reutilización, lo que a su vez facilita la mantenibilidad y escalabilidad. En el caso específico de la transición de plantillas *html.slim* [20] a *html.erb* [21], se consideró mejorar la flexibilidad y la compatibilidad del sistema con un enfoque estándar. En este caso, *html.erb* [21], como motor de plantillas embebidas en Ruby [11] [18], garantiza una mejor integración con el ecosistema de Ruby on Rails [11] [18], facilitando el mantenimiento y la extensión del código.

El equipo del proyecto es integrado por desarrolladores “front end”, los cuales poseen como función principal el rediseño de las vistas e interfaz de usuario siguiendo los lineamientos previstos. Además, hay desarrolladores de “back end” encargados de mantener la implementación, la lógica del servidor y la base de datos.

Un asegurador de calidad (Quality Assurance), se encarga de realizar pruebas exhaustivas e identificar posibles problemas y fallas, asegurando que el producto final cumpla con lo establecido y que ofrezca una experiencia sin errores a los usuarios. Además, un gerente de proyecto (Project Manager) está a cargo de la coordinación general, supervisando la planificación de los diferentes sprints y su ejecución.

---

<sup>2</sup> Una gema es un conjunto de código Ruby empaquetado que te da acceso a métodos personalizados que otra persona escribió.

El gerente de producto (Product Manager) define la visión general del producto Rightboat. Colabora con el equipo de desarrollo, emite revisiones (Product review) para el estado de las tareas y garantiza que el producto final cumpla con las expectativas del mercado y de los clientes.

Además, el equipo cuenta con un especialista en SEO (Search Engine Optimization), cuya función principal es optimizar la estructura de la aplicación para mejorar su clasificación en los motores de búsqueda, logrando un aumento en la cantidad de visitas a la página. Un líder tecnológico (Tech Lead) brinda una dirección técnica sólida asegurando el uso efectivo de las nuevas herramientas.

Todos los participantes del proyecto, se encuentran presentes en las reuniones de planificación del ciclo, revisión del ciclo, refinamiento y revisión tecnológica.

Las reuniones de planificación del ciclo marcan el comienzo de un sprint. Durante la reunión, el gerente del producto (Product Manager) selecciona las tareas que se abordarán y se definen claramente los objetivos que se esperan al final del sprint. Se decide qué elementos del backlog<sup>3</sup> se priorizan y se asignan tareas específicas a los miembros del equipo.

Las reuniones de revisión del ciclo se llevan a cabo al final de cada sprint, el objetivo de esta reunión es a partir de los tickets<sup>4</sup> completados poder determinar posibles ajustes tanto a nivel de equipo para mejorar la eficiencia, así como a nivel de desarrollo para optimizar los procesos. A raíz de esta información, se plantean las tareas a cumplir para el próximo sprint.

Las reuniones de refinamiento no están limitadas a un sprint específico, se llevan a cabo regularmente para revisar y ajustar los elementos del backlog. Durante las reuniones de refinamiento, el equipo revisa, prioriza y puntúa los tickets del backlog, aclara requisitos y se asegura de que estén listos para ser incluidos en los sprints futuros.

Finalmente, las reuniones de revisión tecnológica se centran en revisar y discutir aspectos técnicos específicos del desarrollo y resolver posibles problemas. Permiten fomentar la colaboración entre los miembros del equipo técnico. En estas reuniones sólo participan el líder tecnológico y los desarrolladores.

La comunicación constante y fluida, así como el intercambio de ideas entre los desarrolladores, resultan fundamentales. El equipo encargado del “front end” de la aplicación interactúa directamente con el equipo encargado del “back end” ya existente

---

<sup>3</sup> El backlog es una lista de trabajo ordenado por prioridades para el equipo de desarrollo que se obtiene de la hoja de ruta y sus requisitos.

<sup>4</sup> Un ticket representa una tarea de proyecto. En Jira Software, los tickets suelen representar cosas como funcionalidades importantes, requisitos de los usuarios y errores de software.

en la aplicación. Una buena comunicación entre ambos equipos de desarrolladores es importante para adaptar los cambios necesarios y lograr una implementación exitosa de las nuevas vistas de Rightboat.

Durante el proyecto, se aplican diversas metodologías y procesos para garantizar un enfoque estructurado y eficiente. Entre estas metodologías, se destaca la adopción de un enfoque ágil, en particular la metodología Scrum<sup>5</sup>, que facilita a la organización y colaboración del equipo, permitiendo hacer ajustes rápidos en respuesta a los cambios en los requisitos y sugerencias del cliente.

Para la gestión de tareas, se utiliza la herramienta Jira [10]. En esta herramienta se cargan previamente los tickets a realizar durante cada sprint teniendo en cuenta la capacidad del equipo, asegurando así una distribución efectiva de las responsabilidades y un seguimiento preciso del progreso. Se utilizó un tablero de kanban<sup>6</sup> que se encuentra conformado por ocho columnas (Blocked, To Do, In Progress, Product Review, In Review, QA, Ready for Release, Done), cada una representando un estado particular en el flujo de trabajo.

La columna con estado “Blocked” contiene tickets con obstáculos que impiden su progreso, como puede ser la dependencia de otra tarea. La columna “To Do” posee tareas pendientes de abordar, ordenadas por prioridad. La columna “In Progress” indica las tareas en desarrollo. La columna “Product Review” contiene tickets listos para la revisión del equipo de gestión del producto. La columna “In Review” representa tareas en revisión por otro desarrollador asignado. La columna “QA” contiene tickets a la espera de ser testeados por el asegurador de calidad. La columna “Ready for Release” indica elementos listos para ser subidos a producción y la columna “Done” indica las tareas que fueron completadas.

Se establece una correspondencia entre las columnas y las responsabilidades de cada integrante del equipo, lo que permite a cada miembro de Rightboat comprender claramente cuándo una tarea específica le corresponde. Así, el equipo logra mantener una sincronización eficiente y una coordinación efectiva para avanzar en el flujo de trabajo de manera organizada. Esta distribución de responsabilidades basada en la ubicación de las tareas en el tablero facilita una comprensión clara de las prioridades individuales.

---

<sup>5</sup> Scrum es un marco de gestión de proyectos de metodología ágil que ayuda a los equipos a estructurar y gestionar el trabajo mediante un conjunto de valores, principios y prácticas.

<sup>6</sup> Un tablero de kanban es una herramienta ágil de gestión de proyectos diseñada para ayudar a visualizar el trabajo, limitar el trabajo en curso y maximizar la eficiencia (o el flujo).

## 2.4 Trabajo Realizado

Para llevar adelante el proyecto “Rediseño de la aplicación web Rightboat para compra y venta de embarcaciones” se realizó un trabajo de investigación previo al inicio del mismo. Las tecnologías utilizadas requieren invertir tiempo de estudio para lograr adquirir la base del conocimiento necesario para una integración efectiva de cada una.

Además, en el transcurso del desarrollo del rediseño de la aplicación, se llevó a cabo un análisis del mercado, centrándose en el estudio de diversas aplicaciones ya establecidas en el sector y altamente posicionadas. Este análisis incluye una evaluación de las características funcionales y la usabilidad de aplicaciones similares, así como de las preferencias y patrones de comportamiento de los usuarios.

Se realizaron comparaciones de las diferentes estrategias implementadas por las plataformas competidoras, con el fin de identificar posibles oportunidades y áreas de mejora para nuestra propia aplicación. Se consideraron varios aspectos, tales como la organización de los listados de embarcaciones en cada categoría, la optimización de la experiencia de usuario haciendo foco en la diversidad de filtros disponibles para el usuario en las páginas de listados, la simplicidad de los pasos en el proceso de negocio para la carga de una embarcación y la relevancia de los campos vinculados a cada tipo de barco. Esto sirvió para la toma de decisiones estratégicas y el diseño de características innovadoras en Rightboat.

Para este análisis se utilizaron YachtWorld [24] y Boat Trader [25], dos plataformas de compraventa de embarcaciones en el mercado europeo. Se detectan similitudes significativas entre ambas, tanto en diseño como en usabilidad. Se observa que comparten una paleta de colores análoga, donde el azul predomina en ambas. Asimismo, una disposición similar de los botones, un diseño semejante en las tarjetas de información de los barcos, en la distribución de las vistas al navegar y organización de los formularios de ingreso de datos al realizar una publicación.

Algunas de las decisiones que surgen luego del análisis fueron:

- Al listar los resultados, se utilizan tarjetas con la información de los barcos en posición horizontal en lugar de vertical. El objetivo es proporcionar más detalles visuales de forma inmediata. Esta disposición permite presentar una mayor cantidad de información esencial sobre cada barco sin que el usuario necesite acceder a la página de detalles del barco (Boat Details Page) para obtenerlos. Se mantiene una disposición vertical en las vistas en donde los usuarios no acceden directamente en búsqueda de embarcaciones, como por ejemplo, la página de inicio. En esta página, además de ofrecer información sobre las embarcaciones disponibles, se proporcionan detalles sobre la empresa de compra y venta, así como su funcionamiento.

- Simplificar los filtros en la sección principal de la página de búsqueda con el objetivo de hacer la aplicación más intuitiva para usuarios menos familiarizados con el mercado. Se quitaron opciones de búsqueda irrelevantes en el número final de resultados. Se proporciona una experiencia directa al buscar embarcaciones que permite encontrar lo que el usuario necesita de manera rápida y sencilla.
- Reducir la cantidad de pasos en el proceso de negocio (wizard), logrando diferenciarse de otras plataformas que requieren más pasos. En el desarrollo de una publicación es fundamental disminuir el abandono del sitio sin éxito. Simplificar los pasos en el wizard emerge como una estrategia. Esto se logra mediante la consolidación de la información en etapas menos fragmentadas, el diseño intuitivo de la interfaz, así como la implementación de funciones que permiten el almacenamiento automático del progreso. Almacenar la información cargada por el usuario en un estado “draft” o borrador para brindarle la posibilidad de continuar con la publicación más tarde, cualquiera haya sido el estado en el que haya abandonado el proceso.
- Destacar una única publicidad inteligente. Anteriormente, la página de inicio presentaba hasta 8 barcos destacados aleatoriamente. Ahora se centra en una publicidad clara y directa, que asegura que la información relevante esté inmediatamente disponible según la geolocalización del usuario.

Para comenzar con el rediseño de “front end” de Rightboat se integra una librería destacada por su capacidad para crear interfaces de usuario web y nativas. React Js [1] brinda un enfoque moderno y la capacidad de desarrollar aplicaciones web dinámicas y de alto rendimiento. La naturaleza modular de React Js [1] y la posibilidad de crear componentes reutilizables permiten crear una estructura de código limpia y un desarrollo eficiente. La documentación oficial de la librería destaca su potencial, enfocándose en su capacidad modular como un aspecto fundamental que impulsa su eficacia.

Desde la perspectiva del desarrollo, el uso de React Js [1] ofrece una experiencia coherente y adaptable. Su principal ventaja radica en la libertad de cada miembro del equipo para crear componentes que luego pueden reutilizarse en diversas secciones del proyecto, facilitando la colaboración entre distintos desarrolladores. Un componente, como un botón o un formulario, puede ser creado por un desarrollador y luego ser utilizado por otro en su propia implementación o tarea sin necesidad de conocer su estructura interna. Esta flexibilidad se logra al invocar el componente y enviar las propiedades (props) correspondientes, las cuales son esencialmente atributos o parámetros personalizables. De esta manera, es posible definir distintos aspectos del componente, como su apariencia, comportamiento e incluso funcionalidad.



A continuación, en la figura 12, se presenta un ejemplo de los distintos componentes referentes a botones que se utilizan en Rightboat.



Figura 12. Estructura de componentes botones.

Distintas variantes que invocan al componente principal `Button.jsx` se pueden observar en la estructura.

A continuación, en la figura 13, se presenta el fragmento de código React Js [1] para implementar `DarkButton`.

```
import React from 'react'
import Button from './Button'

export default function DarkButton ({ ...props }) {
  return (
    <Button variant='dark' {...props} />
  )
}
```

Figura 13. Implementación de `DarkButton`.

El componente `DarkButton` instancia a `Button` enviando a través de la propiedad “variant” la variante a la que pertenece (dark). `Button` centraliza las características principales de las distintas variantes. Además, es posible enviarle clases adicionales de estilos de Tailwind CSS [3] para una instancia específica y también, incluir un elemento children en el interior del botón, como un checkbox o un ícono.

A continuación, en la figura 14, se presenta un ejemplo de la imagen del botón que se ubica dentro del carousel en la vista de detalles de un barco (Boat Details Page) y en la figura 15, se presenta el código React Js [1] para incluir el ícono “Share” en el interior del componente y enviarle propiedades con clases adicionales para una invocación en particular.

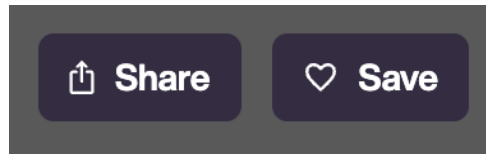


Figura 14. Ejemplos de DarkButton.

```
<DarkButton aria-label='share'>
  · <Share />
  · <p className='md:flex'>Share</p>
</DarkButton>
```

Figura 15. Código React Js para utilizar un DarkButton.

A continuación, en la figura 16, se presenta otra variante, el `PrimaryButton.jsx`.

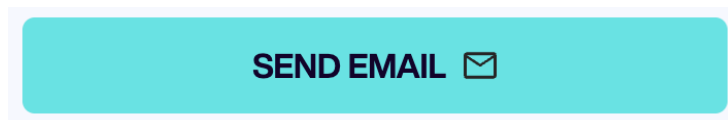


Figura 16. Componente PrimaryButton.

A continuación, en la figura 17, se presenta el código React Js [1] que renderiza dentro del botón un ícono de mail o un spinner de carga dependiendo del estado almacenado en `loadingForm`.

```
<PrimaryButton
  · className='w-full py-3 mt-1'
  · >
  · <span>SEND EMAIL</span>
  · {loadingForm
  · · ? <Loading className='w-4 h-4' />
  · · : <Mail />}
</PrimaryButton>
```

Figura 17. Instancia de PrimaryButton.

React Js [1] proporciona también la increíble funcionalidad de incluir hooks en los componentes, permitiendo realizar tareas más avanzadas. La utilización de los mismos facilita la escritura de código limpio y comprensible en comparación con el uso de clases. Los hooks son funciones que te permiten “engancharte” el estado de React Js [1] y el ciclo de vida desde componentes de función.

Algunos hooks utilizados son los siguientes:

1. **useState** es un React Hook que permite agregar un estado local a un componente funcional. Proporciona una forma de declarar una variable de estado y una función para actualizarla, lo que te permite manipular el estado de un componente sin tener que escribir una clase. Ejemplo de su sintaxis:

```
const [state, setState] = useState(initialState);
```

2. **useEffect** es un React Hook que permite sincronizar un componente con un sistema externo. Se puede utilizar este hook para ejecutar código después de que el componente se haya renderizado. Es útil para realizar tareas como suscribirse a eventos o realizar llamadas a una API. Ejemplo de su sintaxis:

```
useEffect(() => {
  document.title = `You clicked ${count} times`;
});
```

3. **useRef** es un React Hook que te permite hacer referencia a un valor que no es necesario renderizar. Ejemplo de su sintaxis:

```
const ref = useRef(initialValue)
```

React Js [1] posee varios hooks integrados como los mencionados “useState”, “useRef” y “useEffect”. En algunos casos se necesita que haya un hook para cumplir con algún propósito más específico. Por ejemplo, obtener datos, para mantener un seguimiento determinado. Además, es posible crear hooks propios según la necesidad de la aplicación. El uso de hooks personalizados brinda flexibilidad para compartir lógica que no era posible antes. En Rightboat se han implementado con el objetivo de simplificar componentes, quitar lógica de su interior y aprovechar su reutilización.

Por ejemplo, para realizar las solicitudes a la base de datos y comunicar el “back end” con el “front end” se implementó un hook personalizado. Este hook fue denominado `useRailsAjax()` y está diseñado para simplificar el manejo de solicitudes enviadas al “back end” (Ruby on Rails [11] [18]).

Para utilizar el hook personalizado `useRailsAjax()` en un componente de React Js [1], simplemente se debe importar al principio y luego desestructurar los estados y funciones que proporciona. A continuación, en la figura 18 se presenta un ejemplo de su sintaxis.

```
const { data, send, error } = useRailsAjax();
```

Figura 18. Código de React Js para el llamado al hook personalizado.

Esta estructura permite acceder fácilmente a posibles errores y datos de respuesta en el componente. A continuación, en la figura 19, se presenta un ejemplo de solicitud en React Js [1].

```
const handleFormSubmit = (event) => {  
  event.preventDefault();  
  const data = { email: email, origin: "resend-modal" };  
  send({ method: "POST", url: urlEmail, dataForm: data });  
};
```

Figura 19. Envío de información del formulario de recupero de contraseña.

Los hooks en React Js [1], no sólo son útiles por sí mismos, sino que también actúan como un enlace hacia la expansión de funcionalidades a través de librerías de JavaScript [22]. Una librería en JavaScript [22] representa un conjunto de funciones y herramientas predefinidas. Las librerías ofrecen soluciones listas para usar, desde manipulación del DOM hasta gestión de estado o manejo de peticiones HTTP.

En el desarrollo de la interfaz, principalmente en el primer paso de carga de información de una embarcación, la elección de las herramientas a utilizar resulta muy importante para garantizar una experiencia de usuario efectiva. El primer paso permite al usuario ingresar manualmente múltiples campos de información que conforman el modelo del barco en la base de datos. Esta información es utilizada para renderizar las vistas de detalles y generar textos dinámicos. Los textos dinámicos, se utilizan como estrategia, para potenciar la visibilidad de una publicación en los motores de búsqueda. Permiten adaptar el contenido ingresado a diferentes audiencias o situaciones, mejorando la relevancia de la información, atrayendo más usuarios y generando un mayor interés, lo que puede traducirse en un mejor posicionamiento en el ranking del sitio en los resultados de la mayoría de los buscadores en internet.

En ese sentido, se optó por integrar al formulario de creación del modelo de barco ComboBox [14] y ListBox [15] de Headless UI [2].

Headless UI [2] es una biblioteca que permite crear componentes de interfaz de usuario accesibles sin imponer estilos visuales predefinidos, lo que brinda un alto grado de control sobre la apariencia y comportamiento de los elementos. Para los campos de entrada del formulario, esta herramienta permite diseñar y personalizar de manera flexible, manteniendo la accesibilidad y la usabilidad.

El ComboBox [14], es un elemento que combina una caja de texto con una lista desplegable de opciones. Permite al usuario seleccionar una opción de una lista predefinida o ingresar su propia opción si no está disponible en la lista.

A continuación, en la figura 20, se presenta una captura de pantalla para la elección del fabricante de la embarcación.

Make \*

Select make

- Axopar
- Azimut Yachts
- Bavaria Yachts
- Bayliner
- Beneteau
- Boston Whaler
- Chaparral
- Chris Craft

Figura 20. Combo de selección del fabricante de una embarcación.

Un ListBox [15] es un elemento que muestra una lista para que el usuario pueda seleccionar una o más opciones. En el caso de Headless UI [2], el componente ofrece una estructura para crear listas interactivas y accesibles, brinda la libertad de personalizar el diseño y funcionalidad según las necesidades de la aplicación.

En la figura 21, se presenta una captura de pantalla para la elección de las categorías de una embarcación.

Boat category \*

Airboats, Barges

- Aft Cabin Boats
- Airboats
- Aluminum Fishing Boats
- Antique And Classic Boats
- Barges
- Bass Boats
- Bay Boats
- Bowrider Boats

Figura 21. Combo de selección de categorías de una embarcación.

La estructura de ventanas emergentes (modales) de la aplicación fue desarrollada utilizando el componente Dialog [13] de Headless UI [2], este representa una herramienta fundamental al ofrecer una solución sólida y flexible para la implementación de ventanas modales. Su versatilidad radica en su capacidad para gestionar la accesibilidad de manera integral permitiendo una navegación coherente y amigable. Además, su enfoque headless, es decir, sin estilos predeterminados, brinda la libertad de personalizar completamente el aspecto visual de acuerdo con las necesidades y el diseño específico de la aplicación.

Se integró también el componente Transition [16] de Headless UI [2] para crear un efecto durante la apertura y cierre de las ventanas emergentes. Este componente envuelve cada ventana modal permitiendo añadir transiciones de entrada/salida a elementos que se renderizan de manera condicional.

La integración de la biblioteca Uppy [4] en el proyecto constituye otro gran ejemplo de cómo aprovechar el potencial de las librerías. Uppy [4] se destaca por su capacidad para ofrecer funcionalidades altamente personalizables y flexibles para la carga de archivos en aplicaciones web. Se destaca por su eficiencia en la gestión de la carga de archivos y su sencilla interacción.

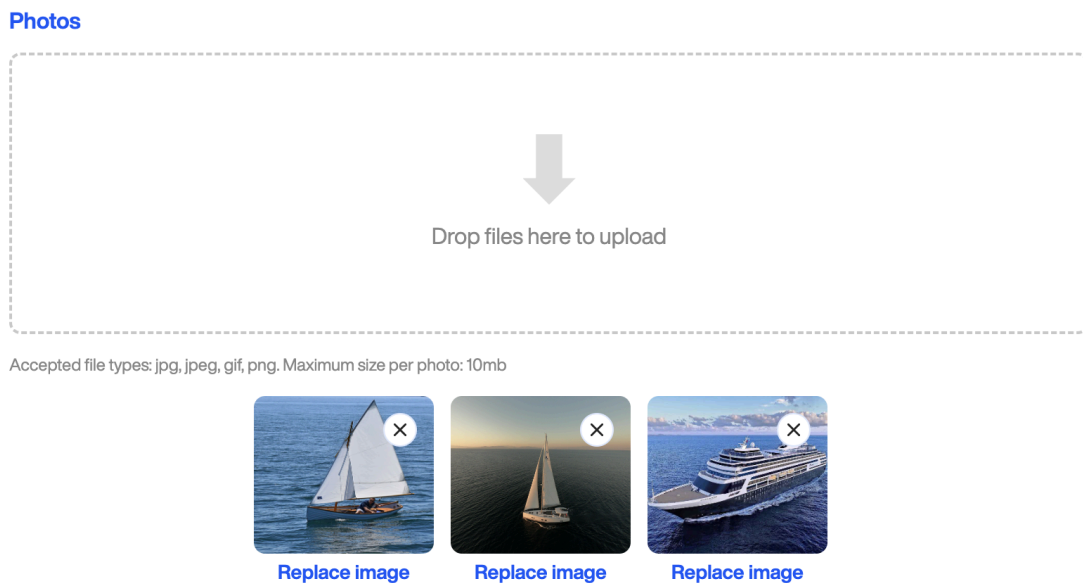
En el segundo paso del proceso de publicación de un barco en el formulario FSBO (For Sale By Owner), se visualiza una zona destinada a la carga de imágenes. Inicialmente, para su desarrollo, se integra la librería Dropzone Js [12]. La elección de incorporar Dropzone [12] se fundamenta en la intención de mantener la librería utilizada hasta el momento, pero actualizada a una versión más reciente. Sin embargo, hacer esta actualización no fue suficiente. Debido a limitaciones y la necesidad de funcionalidades más avanzadas, se plantea la posibilidad de migrar esta tarea a la librería de carga de imágenes Uppy [4]. Esto implica dedicación de tiempo de estudio adicional por parte del equipo para comprender el funcionamiento y las capacidades de esta nueva librería.

Uppy [4] ofrece un conjunto más completo de herramientas, incluyendo soporte para múltiples fuentes de archivos, previsualización en tiempo real, manipulación de imágenes y una mayor capacidad de personalización a través de sus complementos.

Al integrar Uppy [4] en un componente de React Js [1], se aprovechan sus funciones de carga de archivos, de arrastrar y de soltar. La integración de Uppy [4] en la aplicación se realiza mediante la instalación de la herramienta @uppy/react [23], que proporciona componentes específicos para la interacción. Al hacerlo, se crea una experiencia de carga de archivos intuitiva para los usuarios de la aplicación. Se

personaliza la zona de dropzone<sup>7</sup> siguiendo los prototipos de la aplicación implementados en figma [26] por el equipo de diseño.

La creación de una instancia de Uppy [4], brinda la posibilidad de configurar distintos aspectos. A continuación, en la figura 22, se presenta una captura de pantalla para la sección de carga de imágenes de la aplicación.



**Figura 22. Dropzone de carga de imágenes.**

Las imágenes subidas se logran visualizar en miniaturas bajo la zona de carga y cuentan con la posibilidad de ser reordenadas gracias a la integración de la librería Sortable Js [5].

Para determinar cómo implementar la funcionalidad de arrastrar y soltar imágenes, se realizó un proceso de investigación. Se identificaron y evaluaron diversas herramientas disponibles para el arrastre de elementos visuales tales como:

- Dragula Js: Proporciona una API simple para hacer que arrastrar y soltar en varios navegadores en las aplicaciones sea muy sencillo.
- Interact Js: Ofrece una amplia gama de funcionalidades para interactuar con elementos en la interfaz de usuario, incluyendo el arrastre de la imagen.
- Sortable Js: Es una biblioteca independiente de JavaScript [22] para hacer elementos arrastrables y ordenables.

<sup>7</sup> Dropzone es una herramienta destinada a mejorar tu productividad aumentando las acciones posibles al arrastrar y soltar elementos.

Se optó por Sortable Js [5] como la solución más adecuada para satisfacer los requisitos establecidos. Esta elección se fundamentó principalmente en su sencilla integración con las herramientas que estaban siendo utilizadas. La documentación oficial de esta biblioteca ofrece ejemplos prácticos que facilitan la prueba y comprensión de su funcionalidad.

A medida que el proyecto fue adquiriendo mayor complejidad y alcance, la organización de la estructura del mismo se convirtió en un factor crucial para mantener la claridad y escalabilidad. Desde sus inicios hasta su actual fase de rediseño, la adaptación y crecimiento del proyecto se reflejaron directamente en la manera en que se estructuró el código y los componentes.

La estructura de directorios está organizada en distintas carpetas. Cada directorio alberga componentes específicos que se agrupan por funcionalidad. Por ejemplo, la carpeta "alerts" contiene componentes relacionados con mensajes de alerta a los usuarios, mientras que en la carpeta "buttons", "cards" y "form" se ubican los componentes correspondientes a estos elementos visuales.

Además, se tiene una carpeta "inputs" para los diferentes tipos de campos de entrada, "utils" para funciones de utilidad compartidas, "navbar" para componentes de la barra de navegación y "registration" para elementos relacionados con el registro de usuarios. Esto facilita la navegación y mantenimiento del código, permitiendo una gestión eficiente de los componentes dentro del proyecto.

El uso de herramientas como Linear [19] y Jira [10] fue fundamental para optimizar la planificación y ejecución de tareas. Inicialmente, Linear [19] fue utilizada para planificar, crear flujos de trabajo definidos, asignar problemas o tickets a los integrantes del equipo y visualizar el estado general del proyecto.

Linear [19] es una herramienta de gestión de proyectos que puede integrarse con metodologías ágiles como scrum<sup>8</sup>. Proporciona una estructura flexible para organizar y visualizar el flujo de trabajo en equipo.

Se trabajó en iteraciones cortas llamadas sprints, que duraron dos semanas cada uno. Durante cada sprint, se seleccionaron y abordaron una serie de tareas o elementos de trabajo, llamados "tickets". Linear [19] se adapta a esta metodología pero no lo suficiente. Con el objetivo de mejorar la comunicación dentro del equipo surge la migración hacia la herramienta de gestión de proyectos, Jira [10]. Esta transición tiene como objetivo optimizar la eficiencia en el desarrollo al facilitar una conexión más

---

<sup>8</sup> Scrum es un marco de gestión de proyectos de metodología ágil que ayuda a los equipos a estructurar y gestionar el trabajo mediante un conjunto de valores, principios y prácticas.



fluida con Slack<sup>9</sup>, la herramienta de comunicación utilizada por el equipo. La integración entre ambas plataformas busca maximizar la colaboración, permitiendo una interacción más ágil y coordinada entre los miembros.

La implementación de un bot en Slack resulta también un factor clave en el proceso de comunicación entre integrantes. Este bot<sup>10</sup>, se encarga de notificar al equipo instantáneamente cada vez que se producen cambios en el estado de los tickets en Jira [10]. Mejorando significativamente la visibilidad y la agilidad en la comunicación, permitiendo una respuesta más rápida y precisa ante cualquier modificación o actualización en el desarrollo del proyecto.

## 2.5 Trabajo Futuro

Rightboat cuenta con un abanico de funcionalidades muy amplio, el proyecto de rediseño se llevó a cabo luego de trabajar en una serie de cambios y modificaciones destinados a abordar las necesidades más urgentes de la aplicación.

Una versión de la aplicación que fue lanzada a producción algunos años atrás, con el paso del tiempo, la evolución tecnológica ha llevado a que las herramientas utilizadas se vuelvan obsoletas. En respuesta a una necesidad de actualización, se realizó un proyecto integral de rediseño de funcionalidades. Dada la magnitud de los cambios requeridos, se optó por organizar las modificaciones por orden de prioridad, abordando de manera sistemática y eficiente las áreas más urgentes.

Algunas líneas de posibles trabajos futuros para la aplicación Rightboat:

- **Rediseño de las plantillas de emails de la aplicación de *html.slim* [20] a *html.erb* [21]:** Actualmente las plantillas de los mails continúan utilizando los diseños anteriores, a futuro será necesario aplicar los nuevos cambios. La necesidad de mejorar la compatibilidad con herramientas y bibliotecas que están más optimizadas para el formato *html.erb* [21] en el ecosistema Ruby on Rails [11] [18]. Utilizar *html.erb* [21] puede simplificar la integración de funcionalidades específicas de Rails y facilitar la colaboración con otros desarrolladores que estén más familiarizados con este formato.
- **Reestructuración y reimplementación del sistema de “Saved Searches”:** La aplicación cuenta actualmente con un sistema de guardado de búsquedas en el que los usuarios pueden guardar sus búsquedas realizadas con los filtros que hayan aplicado y recibir notificaciones por correo electrónico diariamente cada vez que se publique un nuevo barco con las características de la

---

<sup>9</sup> Slack es una aplicación de mensajería para empresas que conecta a las personas con la información que necesitan.

<sup>10</sup> Un bot es una aplicación de software automatizada que realiza tareas repetitivas en una red.



búsqueda realizada o que se actualice la información de una embarcación ya existente.

- **Migración del motor de búsqueda de Apache Solr [27] a Elasticsearch [28]:** El objetivo principal de la migración será lograr mejoras en el rendimiento, escalabilidad y funcionalidades específicas de búsqueda. Elasticsearch es conocido por su velocidad, distribución y capacidades de análisis avanzadas, lo que podría ofrecer un mejor soporte para las necesidades específicas de búsqueda en la aplicación, así como una integración más fluida con otras tecnologías.
- **Implementar caché en las vistas de la aplicación para mejorar la performance:** Guardando información de la aplicación en caché utilizando el motor de base de datos Redis [29] se puede lograr mejorar significativamente el rendimiento al reducir la carga en la base de datos y acelerar la entrega de contenido estático.

## REFERENCIAS BIBLIOGRÁFICAS

[1] Artemij Fedosejev (2015) *React.js Essentials*: A fast-paced guide to designing and building scalable and maintainable web apps with React Js.

[https://strukturnifondovi.hr/wp-content/uploads/2017/03/9781783551620-REACTJS\\_ESSENTIALS.pdf](https://strukturnifondovi.hr/wp-content/uploads/2017/03/9781783551620-REACTJS_ESSENTIALS.pdf)

<https://legacy.reactjs.org/docs/getting-started.html>

<https://es.react.dev/>

[2] *Headlessui*: Completely unstyled, fully accessible UI components, designed to integrate beautifully with Tailwind CSS.

<https://headlessui.com/>

[3] Adam Wathan, Steve Schoger (2017) Documentación de *Tailwind CSS*. La documentación oficial de Tailwind CSS es desarrollada y mantenida por el equipo de desarrollo de Tailwind Labs. Algunos de los miembros fundadores y principales colaboradores de Tailwind Labs, como Adam Wathan, Jonathan Reinink, Steve Schoger y David Hemphill.

<https://v2.tailwindcss.com/docs>

<https://github.com/tailwindlabs/tailwindcss>

[4] *Uppy*: Uppy is a sleek, modular JavaScript file uploader that integrates seamlessly with any application. It's fast, has a comprehensible API and lets you worry about more important problems than building a file uploader.

<https://uppy.io/docs/quick-start/>

[5] *Sortable Js*: JavaScript library for reorderable drag-and-drop lists.

<https://sortablejs.github.io/Sortable/>

[6] Noel Rappin (2021) *Modern CSS with Tailwind*: Flexible Styling without the Fuss.

<https://www.amazon.com/-/es/Noel-Rappin/dp/1680508180>

[7] Alex MacCaw (2012) The Little Book on *CoffeeScript*: The JavaScript Developer's Guide to Building Better Web Apps.

<https://www.amazon.com/-/es/Alex-MacCaw/dp/1449321054>

[8] *Gtmetrix*: See how your site performs, reveal why it's slow and discover optimization opportunities.

<https://gtmetrix.com/>



[9] *Rack-mini-profiler*: Middleware that displays speed badge for every HTML page, along with (optional) flamegraphs and memory profiling. Designed to work both in production and in development.

<https://github.com/MiniProfiler/rack-mini-profiler>

[10] Atlassian (2002) *Jira Software*: Jira Software is the #1 agile project management tool used by teams to plan, track, release and support world-class software with confidence.

<https://www.atlassian.com/es/software/jira/guides/getting-started/introduction>

[11] Michael Hartl (2022) *Ruby on Rails* Tutorial: Learn Web Development with Rails.

[https://www.amazon.com/-/es/Michael-Hartl-ebook/dp/B0BGDX5JM/ref=sr\\_1\\_1?crid=179IRQSP8QW3&keywords=ruby+on+rails+book&qid=1691711895&s\\_prefix=ruby+on+rail%2Caps%2C392&sr=8-1](https://www.amazon.com/-/es/Michael-Hartl-ebook/dp/B0BGDX5JM/ref=sr_1_1?crid=179IRQSP8QW3&keywords=ruby+on+rails+book&qid=1691711895&s_prefix=ruby+on+rail%2Caps%2C392&sr=8-1)

[12] *Dropzone.js* is one of the most popular drag and drop JavaScript libraries. It is free, fully open source, and makes it easy for you to handle dropped files on your website.

<https://www.dropzone.dev/>

[13] *Dialog (Modal)* A fully-managed, renderless dialog component jam-packed with accessibility and keyboard features, perfect for building completely custom modal and dialog windows for your next application.

<https://headlessui.com/react/dialog>

[14] *ComboBox* Comboboxes are the foundation of accessible auto-completes and command palettes for your app, complete with robust support for keyboard navigation.

<https://headlessui.com/react/combobox>

[15] *ListBox* Listboxes are a great foundation for building custom, accessible select menus for your app, complete with robust support for keyboard navigation.

<https://headlessui.com/react/listbox>

[16] *Transition* The Transition component lets you add enter/leave transitions to conditionally rendered elements, using CSS classes to control the actual transition styles in the different stages of the transition.

<https://headlessui.com/react/transition>

[17] *Stripe* powers online and in-person payment processing and financial solutions for businesses of all sizes.

<https://stripe.com/docs>

[18] *Ruby on Rails*: Rails is a full-stack framework. It ships with all the tools needed to build amazing web apps on both the front and back end.



<https://guides.rubyonrails.org/>

[19] Linear: Linear streamlines issues, sprints, and product roadmaps. It's the new standard for modern software development.

<https://linear.app/>

[20] Ruby on Rails: Template Engine with Slim and Haml: Learn how to use properly the two most famous template engine.

<https://www.amazon.com/Ruby-Rails-Template-properly-template-ebook/dp/B01HP5A40S?>

<https://github.com/slim-template/slim>

[21] Ruby on Rails layout and rendering documentation. Documentación mantenida por la comunidad de Ruby on Rails.

[https://guides.rubyonrails.org/layouts\\_and\\_rendering.html](https://guides.rubyonrails.org/layouts_and_rendering.html)

[22] JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions.

<https://developer.mozilla.org/es/docs/Web/JavaScript>

[23] @uppy/react React components for the Uppy UI plugins and a useUppyState hook.

<https://uppy.io/docs/react/>

[24] Yachtworld: Find Power boats for sale in your area & across the world on YachtWorld.

<https://www.yachtworld.es/>

[25] Boat Trader: Find thousands of New & Used Boats, Outboard Motors, Engines, Trailers. Sell your Boat fast online today, read our in-depth boating guides & more!

<https://www.boattrader.com/>

[26] Figma: Is the leading collaborative design tool for building meaningful products.

<https://www.figma.com/best-practices/guide-to-developer-handoff/>

[27] Apache Solr: Solr is the open source solution for search and analytics.

<https://solr.apache.org/guide/solr/latest/index.html>

[28] ElasticSearch: Elasticsearch is the search and analytics engine that powers the Elastic Stack.

<https://www.elastic.co/es/>

[29] Redis: The open-source, in-memory data store used by millions of developers as a cache.

<https://redis.io/docs/>