



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Desarrollo de Front-End para el Sistema de Logística de Contenedores en Almacenaje Extraportuario Aexsa

Raúl Cabrera Espinoza

raul.cabrera@usm.cl

Carlos Alten
Profesor Guía

Resumen: En esta tesina se plantea la implementación de frontend para la empresa de almacenaje extraportuario Aexsa, que enfrenta el desafío de optimizar sus operaciones relacionadas al movimiento de contenedores. Ante esto surge el proyecto Optainer, que busca implementar una aplicación web que les proporcione las herramientas y funcionalidades necesarias para optimizar sus tareas. Se apunta a crear interfaces que generen muy buenas experiencias a los distintos usuarios. Para evaluar la propuesta, se presentaron prototipos a empleados de la empresa, recibiendo retroalimentación para mejorar la interactividad con la aplicación. Se espera que, si la aplicación es adquirida por la empresa, pueda agilizar y optimizar la gestión de sus contenedores.

Palabras Clave: contenedores, frontend, componente, optimización, almacenaje extraportuario

1 Introducción

1.1 Contexto y relevancia del desarrollo de componentes específicos en el proyecto

Los almacenes extraportuarios son instalaciones de almacenamiento ubicadas fuera de las áreas portuarias, diseñadas para manejar y resguardar mercancías importadas o exportadas. Estas instalaciones pueden estar situadas en zonas industriales, cerca de centros de distribución o incluso en áreas metropolitanas estratégicas [1]. Las mercancías son dispuestas dentro de contenedores de distintos tamaños y dependiendo de la carga estos pueden necesitar refrigeración para mantener una temperatura específica en su interior [2].

Aexsa es una empresa de almacenaje Extraportuario que gestiona un patio de contenedores. Esta enfrenta el desafío de optimizar sus operaciones relacionadas con el movimiento de contenedores. La empresa actualmente administra la ubicación de los contenedores mediante Excel, lo que solo presenta limitaciones en cuanto a eficiencia y usabilidad. Esta ya contaba con un sistema web que les permitía revisar el inventario, además de los movimientos que debe realizar el operador de grúa, pero se nos mencionó en reiteradas ocasiones por distintos usuarios que dicho sistema era demasiado arcaico y que no ofrecía las funciones más potentes que les permitieran optimizar su trabajo. Ante esta necesidad, nuestro proyecto “Optainer” busca crear un sistema más avanzado, que permita una gestión de información y visualización de los contenedores eficiente, además de proporcionar herramientas útiles para la toma de decisiones en tiempo real.

1.2 Definición del problema.

En la actualidad, Aexsa gestiona la ubicación de los contenedores utilizando planillas de Excel. El operador de la grúa recibe una planilla con los detalles del contenedor que se debe mover, como su



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Desarrollo de Front-End para el Sistema de Logística de Contenedores en Almacenaje Extraportuario Aexsa

Nombre del candidato(a): Raúl Genaro Cabrera Espinoza

Carrera / Grado: Ingeniería en Informática

Campus: Viña del Mar ; **Departamento:** Electrotecnia e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Carlos Alten, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 25-06-2025

; Firma:

Estudiante o Candidato(a):

Fecha: 19-06-2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

identificador y otras características importantes. Cada movimiento que es realizado, así como cualquier otra información relevante, debe ser registrada manualmente en el archivo Excel. Este proceso es lento y propenso a errores, lo que aumenta de forma considerable el tiempo que requiere completar las tareas. Además, las instrucciones para mover los contenedores de ubicación son proporcionadas por el jefe de patio a través de walkie-talkie, basándose principalmente en su intuición y experiencia, sin contar con un sistema estandarizado o cálculos precisos.

Dada la gran cantidad de contenedores que maneja Aexsa actualmente, es común que algunos resulten extraviados, lo que provoca una pérdida significativa de tiempo en la localización de estos. Esta falta de un sistema eficiente para gestionar y monitorear la ubicación de los contenedores en tiempo real limita gravemente la capacidad que tiene la empresa para optimizar sus recursos, agilizar sus operaciones y reducir sus costos asociados a la gestión de contenedores.

1.3 Breve descripción sobre la propuesta de solución

Nuestra solución consiste en un sitio web responsivo, que presentará distintas interfaces entre las que se encuentran:

- Inicio/Mapa General: Incluye un mapa en 3D del patio de contenedores de Aexsa, que permite una visualización más cómoda de este, además de actualizar la ubicación actual de los contenedores cuando sean movidos por el operador de la grúa de forma manual. Contará además con distintas herramientas de filtrado para los contenedores.
- Inventario avanzado: Visualización avanzada de los bloques de contenedores individualmente.
- Tareas: Permite a un administrador asignar tareas relacionadas al movimiento de contenedores a un operador de grúa.

Como se mencionó anteriormente, el sitio será responsivo, esto debido a que en la empresa es empleado el uso de dispositivos de escritorio y tableta para los operadores de grúa.

Al autor de este documento le corresponde realizar el desarrollo y documentación de los diferentes componentes visuales, así como las distintas herramientas y librerías usadas para el sistema propuesto.

1.4 Objetivos Generales y Específicos de la Tesina

Objetivo General:

Desarrollar una interfaz web interactiva y responsiva para la gestión logística eficiente en un patio de contenedores, que permita a los distintos usuarios visualizar de forma cómoda los distintos bloques, además de consultar información detallada sobre los contenedores, implementando un diseño intuitivo y eficiente para brindar una alta experiencia de usuario.

Objetivos Específicos:

- Diseñar un mapa 3D que permita visualizar de forma cómoda e intuitiva un patio de contenedores utilizando librerías modernas.
- Integrar una funcionalidad que permita la reubicación manual de contenedores en tiempo real dentro del mapa 3D del patio de contenedores.
- Diseñar y programar una interfaz de usuario Responsiva para los distintos tipos de dispositivos manejados en la empresa.
- Integrar funcionalidades de filtrado y búsqueda al sitio web para agilizar procesos.

- Desarrollar una interfaz de usuario que permita a administradores el designar tareas a los operadores de grúa.

1.5 Justificación de proyecto

Este proyecto agilizará y facilitará el trabajo realizado por los distintos usuarios de la empresa, debido a que se proporcionarán diversas funcionalidades para optimizar sus procesos laborales cotidianos. El correcto desarrollo del frontend es importante en el proyecto, ya que está estrechamente relacionado con la usabilidad e interacción de los usuarios finales, que son los que evaluarán si la solución aporta realmente a solucionar sus problemáticas.

Beneficios:

- La integración del proyecto reemplaza los ineficientes procesos antiguos de la empresa, integrando funcionalidades que resultan de alta utilidad.
- La visualización del patio de contenedores en 3D con Three.js, una librería de JavaScript diseñada específicamente para crear gráficos tridimensionales en aplicaciones web, permitirá a los usuarios obtener una mayor comodidad al momento de realizar sus actividades y tomar decisiones.
- El sistema de “tareas”, que involucra los movimientos específicos que debe realizar un operador de grúa, brindará a los usuarios interesados la certeza de que el trabajo realizado por los operadores es el correcto.
- El uso de una biblioteca moderna como React permite crear una experiencia atractiva y escalable, lo que resulta en mejores experiencias de usuario y usabilidad.

1.6 Metodología

Se utilizará la metodología de desarrollo ágil Scrum, que divide al equipo por roles y el trabajo en sprints, que poseen objetivos específicos. Al finalizar cada sprint el progreso de cada uno de los integrantes es evaluado, lo que permite tener un avance y retroalimentación constante durante el desarrollo.

Para la construcción del frontend se empleará la metodología ágil Scrum debido a la versatilidad y experiencia previa obtenida en proyectos universitarios anteriores, lo que proporciona una base familiar sobre la cual realizar el desarrollo del proyecto. En esta metodología el autor de esta tesina es parte del equipo de desarrolladores, enfocándose en la programación del frontend.

La metodología ágil Scrum, proporciona beneficios como el ritmo de trabajo sostenible, ya que se adapta a los cambios de manera inmediata consiguiendo un ritmo constante, tanto en la duración de cada sprint como de esfuerzo. Otro beneficio que proporciona Scrum en la gestión de proyectos es la capacidad de solapar las fases de desarrollo, lo que da una gran ventaja frente a metodologías de desarrollo de cascada, en las que una fase comienza solo si se terminó la anterior [3].

Casos de éxito de empresas utilizando Scrum:

- 1.- Google: Ha logrado acelerar el tiempo de lanzamiento de sus nuevos productos al adoptar scrum en sus equipos de desarrollo.
- 2.- Spotify: Plataforma que ha tenido un enorme crecimiento en la industria del streaming en gran parte gracias a su enfoque ágil. Spotify ha adoptado Scrum en todos los aspectos de su negocio, desde el desarrollo de productos hasta la gestión de proyectos [4].
- 3.- Amazon: Empresa de e-commerce que ha adoptado con éxito Scrum en sus procesos de trabajo y lineamientos [5].



1.7 Breve descripción de la organización del informe en capítulos.

En los capítulos siguientes se profundizará en:

- **Capítulo 2:** En este capítulo se abordan los fundamentos del frontend, explicando las tecnologías principales que se utilizan, la evolución historia del frontend, las tendencias actuales de la industria y una descripción general de la estructura del software. Además, se detallan las tecnologías y frameworks utilizados en el proyecto, dando el conocimiento necesario para comprender los conceptos y herramientas clave para el desarrollo frontend.
- **Capítulo 3:** En este capítulo se detalla la arquitectura y estructura de los componentes de la aplicación, dividiéndolos en componentes principales y comunes, explicando su organización. Se introduce el concepto de Clean Architecture, describiendo como se estructuran las carpetas del proyecto. También se abordan las librerías instaladas, la implementación de la responsividad utilizando Bootstrap, y las buenas prácticas y patrones de diseño aplicados en el desarrollo. Además, se explica la integración del frontend con el sistema general, detallando las solicitudes HTTP y Websockets. También se detalla la metodología de trabajo empleada en el desarrollo del proyecto, incluyendo los sprints y las tareas asociadas a cada uno. Finalmente, se abordan las estrategias de testing, pruebas y validación de los componentes, junto con las técnicas de resolución de problemas y optimización.
- **Capítulo 4:** En este capítulo se presentan las conclusiones obtenidas a lo largo del proyecto.
- **Capítulo 5:** Contiene las referencias bibliográficas y fuentes consultadas durante la investigación y desarrollo del proyecto.
- **Capítulo 6:** Se incluyen materiales complementarios relevantes que permiten ayudar a entender mejor el proyecto.

2 Marco Teórico

En este capítulo se explorarán los fundamentos principales del frontend, desde su evolución histórica hasta las tendencias que han predominado en la actualidad. Además, se detallará la arquitectura que se utilizará para la construcción del frontend para el proyecto Optainer, junto con las distintas tecnologías que se implementarán en su desarrollo.

2.1 Fundamentos del desarrollo Frontend

El Frontend es todo lo que constituye el apartado visual en una aplicación ya sea web o móvil, incluyendo la estructura del sitio, colores, fondos, tipografías y otros elementos con los que interactúa un usuario directamente. Para el éxito de un proyecto web es esencial la construcción de una presentación visual atractiva y una interacción que ofrezca una experiencia de usuario satisfactoria y agradable [6].

2.1.1 Tecnologías principales:

- **HTML (HyperText Markup Language):** Es el lenguaje de marcado que se utiliza para estructurar y organizar una página web y sus contenidos. Responde a la pregunta ¿Qué es? [7]
- **CSS (Cascading Style Sheets):** Es el lenguaje utilizado para controlar el diseño, las tipografías, los colores, el alineamiento y otros aspectos visuales que posee una página web. Responde a la pregunta ¿Cómo es? [7]
- **JavaScript:** Lenguaje de programación utilizado para agregar interactividad y funcionalidad a una página web. Responde a la pregunta ¿Qué hace? [7]

2.1.2 Evolución histórica del Frontend:

El Frontend tiene su nacimiento en la década de 1990, con la creación de la World Wide Web y HTML, donde aparecen las primeras páginas web. Los primeros sitios web eran proyecciones estáticas de información en un navegador, donde las páginas eran muy limitadas y solo mostraban texto e imágenes, con opciones de personalización y diseño muy simple [8].

A medida que avanzaban los años se introducen nuevas tecnologías como CSS y JavaScript que permitían tener una mayor personalización en los sitios web [8].

Con el paso del tiempo empezaron a surgir nuevas tecnologías que facilitarían técnicamente el desarrollo Frontend como backbone.js, que permitió organizar y modularizar el código de Frontend y Backend mediante su modelo de arquitectura MVC (modelo vista controlador). Pronto aparecerían frameworks por todos lados, como React, Angular y más recientemente Vue [9].

El próximo gran paso para el Frontend fue el modelo de arquitectura Spa (single-page application), que simula ser una página web de varias páginas, aunque en realidad es una aplicación de solo una página, diseñada de tal forma que solo se cargue una vez y se actualiza dinámicamente a medidas que el usuario interactúa sin necesidad de recargarla [9].

2.1.3 Tendencias Actuales de la industria:

- **Dark mode:** Siendo incorporado por Facebook, Twitter y otras aplicaciones comúnmente usadas, el modo oscuro permite a los usuarios otra opción al momento de visualizar una interfaz, ofreciendo mayor legibilidad y claridad al contenido, mejorando la experiencia de usuario [10].

- **PWAs (Progressive Web Apps):** Aplicaciones accesibles a través de un navegador con ventajas sobre una aplicación nativa, como tener opción offline, velocidad de carga aumentada y no necesitar actualizarse manualmente [10].

El Frontend se encuentra presente en diversos tipos de sistemas, entre los que se encuentran:

- **Aplicaciones web:** Aplicaciones que son accedidas a través de un navegador de internet y no requieren instalación. Estas son programadas con tecnologías de desarrollo web [11].
- **Aplicaciones nativas:** Aplicaciones desarrolladas para un sistema operativo o plataforma en específico, programados con sus respectivos SDK (kit de Desarrollo de Software). Estas aprovechan al máximo las capacidades del dispositivo [11].
- **Aplicaciones híbridas:** Combinación entre aplicaciones web y nativas, ejecutables en distintas plataformas. Estas son programadas con tecnologías de desarrollo web y empaquetadas en un contenedor nativo, lo que les permite funcionar como una aplicación nativa en sus respectivas plataformas [11].

2.2 Arquitectura de software

React es una biblioteca de JavaScript basada en la construcción de interfaces de usuario con una arquitectura basada en componentes. Las distintas vistas y funcionalidades de la aplicación web se descomponen en componentes más pequeños que pueden ser reutilizados [12].

La arquitectura basada en componentes proporciona organización y modularización en el código, lo que facilita el mantenimiento y gestión de aplicaciones [12].

Beneficios de arquitectura basada en componentes en React:

- **Mejor organización de código:** Al dividir la interfaz de usuario en componentes proporciona un código más limpio y fácil de entender [12].
- **Mantenimiento más fácil:** Los cambios pueden ser realizados en componentes específicos sin afectar al resto de la aplicación [13].

A continuación, se describe la estructura y disposición de los componentes, tomando como ejemplo la vista del inventario avanzado. En este caso, el componente principal (MapaAvanzadoView) actúa como contenedor de los componentes más pequeños (componentes comunes), que desempeñan funciones específicas o muestran datos que no cambian. El componente principal MapaAvanzadoView, representa la vista del inventario avanzado de la aplicación web real. Este componente actúa como contenedor para otros elementos, como el componente de búsqueda, que permite localizar un contenedor específico, los componentes de botón, que facilitan el agregar o eliminar un contenedor al sistema respectivamente, y los componentes encargados de mostrar la altura y columna en la que se encuentran los contenedores, entre otros.

Componente principal: MapaAvanzadoView



Componentes comunes:

- BarraTorres
- SideBarLateral
- BuscadorContenedor
- BotonAgregarContenedor
- BotonEliminarContenedor
- GrillaContenedores
- NumerosLateralIzquierda
- BotonesProfundidad
- NumerosAbajo

Figura 2-1. Estructura y disposición de los componentes de la vista de inventario avanzado
Fuente: Elaboración propia

2.3 Tecnologías y frameworks utilizados.

Para la construcción del Front-End y sus componentes se emplearán las siguientes herramientas:

- React: Biblioteca de JavaScript centrada en las interfaces de usuario con componentes reutilizables [14]. Se utilizará React para el desarrollo de los distintos componentes y vistas de la aplicación web.
- HTML (HyperText Markup Language): Lenguaje de marcado que se utiliza para estructurar y organizar una página web y sus contenidos [7].
- CSS (Cascading Style Sheets): Lenguaje utilizado para controlar el diseño, las tipografías, los colores, el alineamiento y otros aspectos visuales que posee una página web [7].
- JavaScript: Lenguaje de programación utilizado para agregar interactividad y funcionalidad a una página web [7].
- Three.js: Biblioteca de JavaScript utilizada para crear y mostrar contenido 3D en un navegador [15].
- Vite: Herramienta de compilación que tiene como objetivo proporcionar una experiencia de desarrollo más ágil para proyectos web modernos [16].
- JSX: Extensión de sintaxis para JavaScript que permite escribir marcado similar a HTML en un archivo JavaScript [17].

3 Diseño e Implementación

En este capítulo se presentarán los distintos diagramas que proporcionan una comprensión más detallada del proyecto Optainer. Además, se explicarán los tipos, la disposición y los componentes individuales que componen el frontend, así como las buenas prácticas y patrones de diseño empleados. También se abordará la metodología de desarrollo aplicada, junto con las pruebas y validaciones realizadas para asegurar el correcto funcionamiento del proyecto.

3.1 Diagramas UML:

Se presentarán los distintos diagramas realizados con UML para representar de manera clara los distintos aspectos presentes en el sistema.

Diagrama de casos de uso:

A continuación, se muestra un diagrama de casos de uso, que sirve para mostrar a los diferentes actores que utilizan la aplicación y las acciones que pueden ejecutar en él. Cuando se mencione al mantenedor de contenedores, se refiere a las acciones de agregar y eliminar contenedores del sistema. Los actores del diagrama corresponden a los usuarios que utilizarán la aplicación, siendo estos el administrador y el operador de grúa, quienes pueden realizar las distintas tareas relacionadas al movimiento de los contenedores en el patio de la empresa extraportuaria. Los casos de uso identificados para este diagrama están presentes en las distintas burbujas del diagrama para el sistema Optainer. Adicionalmente, han sido incluidos casos de uso relacionados a las tareas que deben ser realizadas por el operador de grúa.

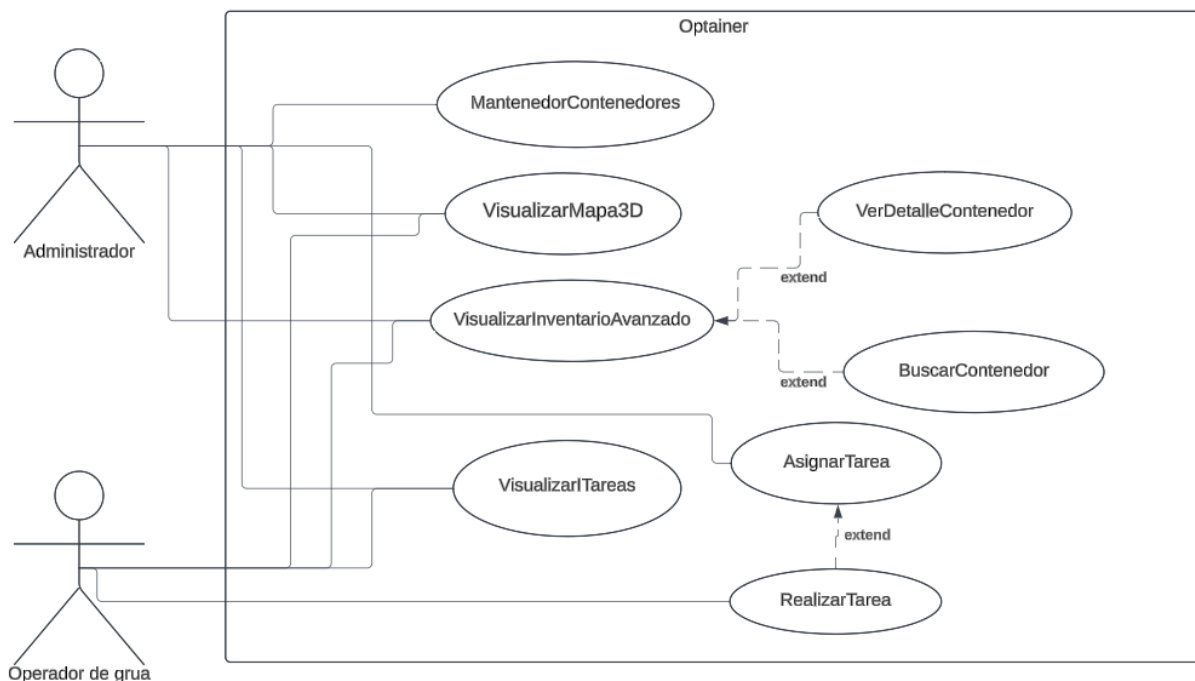


Figura 3-1. Diagrama de casos de uso del proyecto
Fuente: Elaboración propia

Diagrama de clases:

A continuación, se muestran los distintos diagramas de clase para cada una de las vistas, adaptados para que las clases sean los componentes de React. En este caso, los nombres de las clases son los nombres de los componentes, los atributos son los states y variables, y los métodos son las funciones y hooks. Hay componentes que reciben funciones o states como props, por lo que esto se especifica en el campo de tipo de dato si corresponde. Hay componentes que no tienen ni reciben states o funciones, por lo que se representan con un cuadro simple.

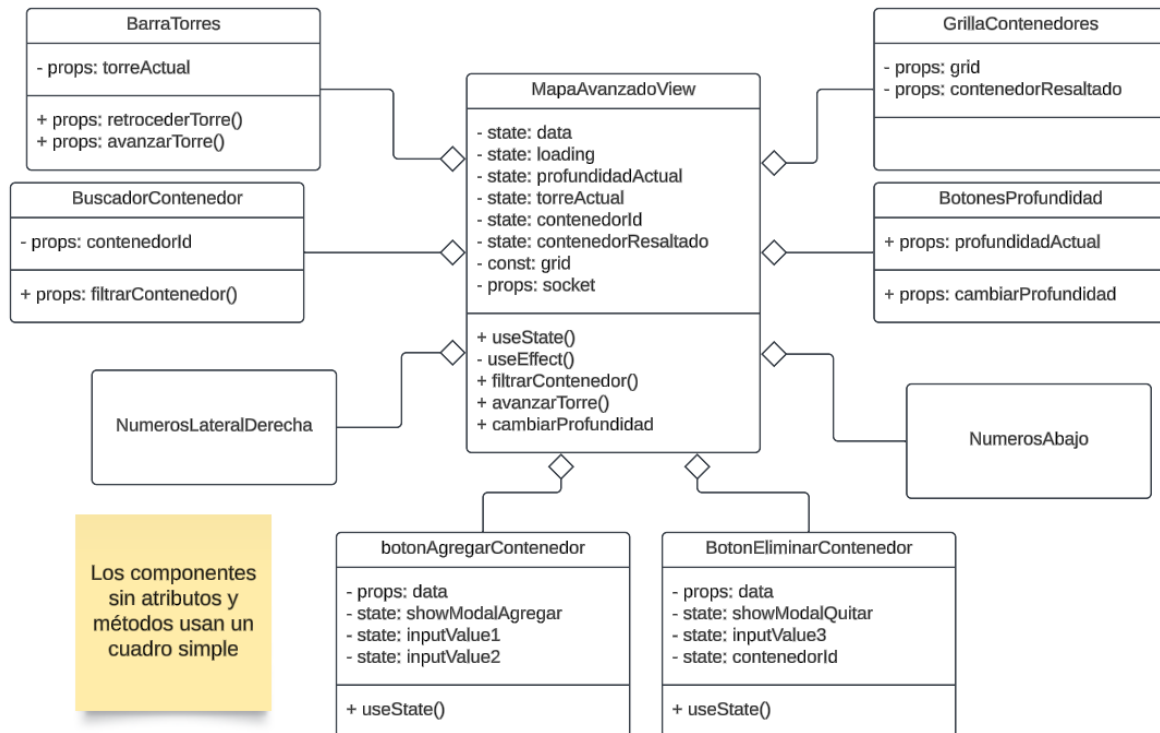


Figura 3-2. Diagrama de clases para vista de inventario avanzado

Fuente: Elaboración propia

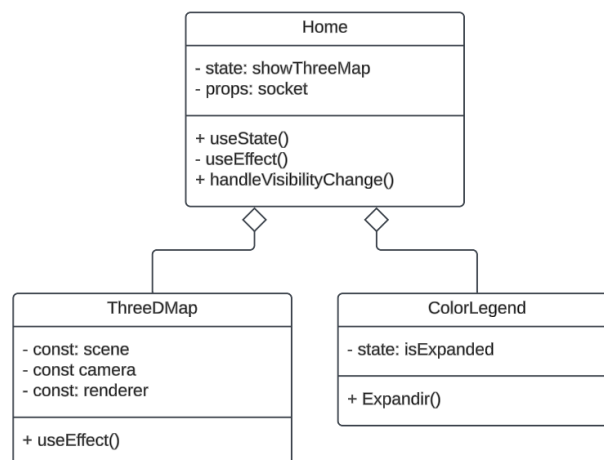


Figura 3-3. Diagrama de clases para la vista del mapa 3D

Fuente: Elaboración propia

Diagrama de secuencia:

A continuación, se muestran diagramas de secuencia para las dos funcionalidades que han sido las más representativas de la aplicación, siendo estas, mostrar los contenedores en el inventario avanzado y agregar un nuevo contenedor al sistema. Para el diagrama que muestra el proceso de mostrar los contenedores, se cuenta con los respectivos componentes que participan (la vista avanzada de contenedores, la API y la base de datos) y al actor que interactúa con la aplicación (el operador de la grúa). Para el diagrama que presenta la secuencia que permite agregar un contenedor nuevo al sistema, se cuentan con los mismos componentes, pero con el actor administrador (puede agregar y eliminar contenedores). El diagrama para eliminar un contenedor sería bastante similar al de agregar, pero cambiando el método de POST a DELETE

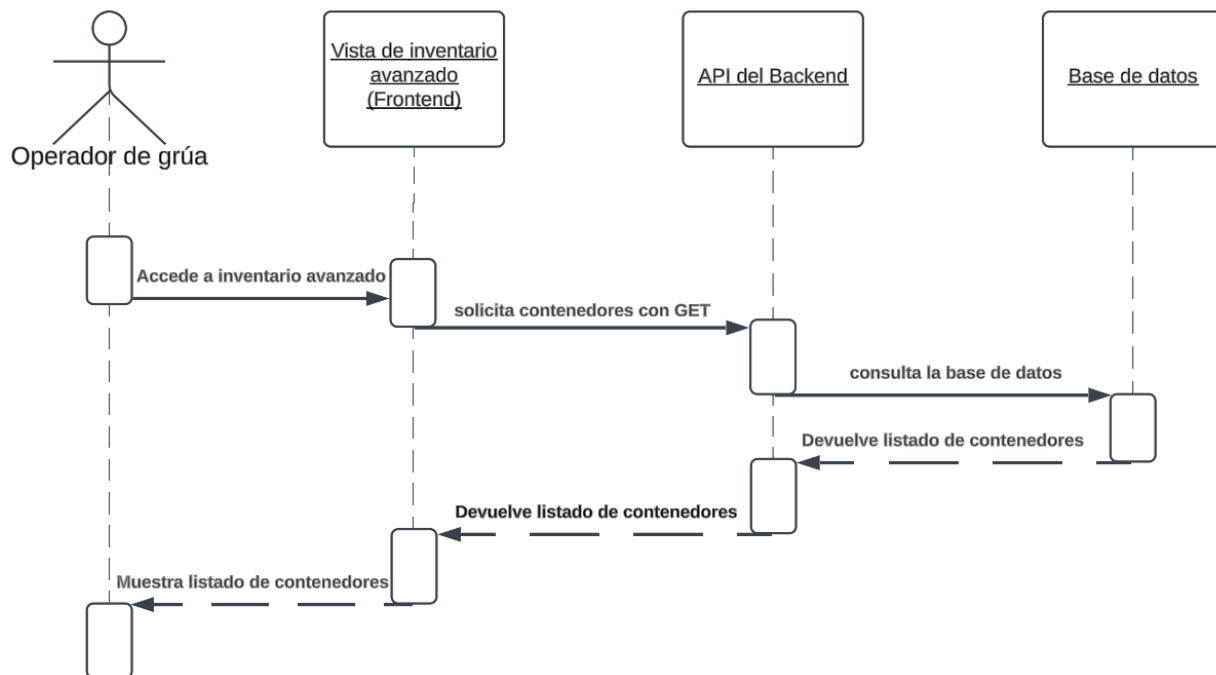


Figura 3-4. Diagrama de secuencia para visualizar contenedores en el inventario avanzado

Fuente: Elaboración propia

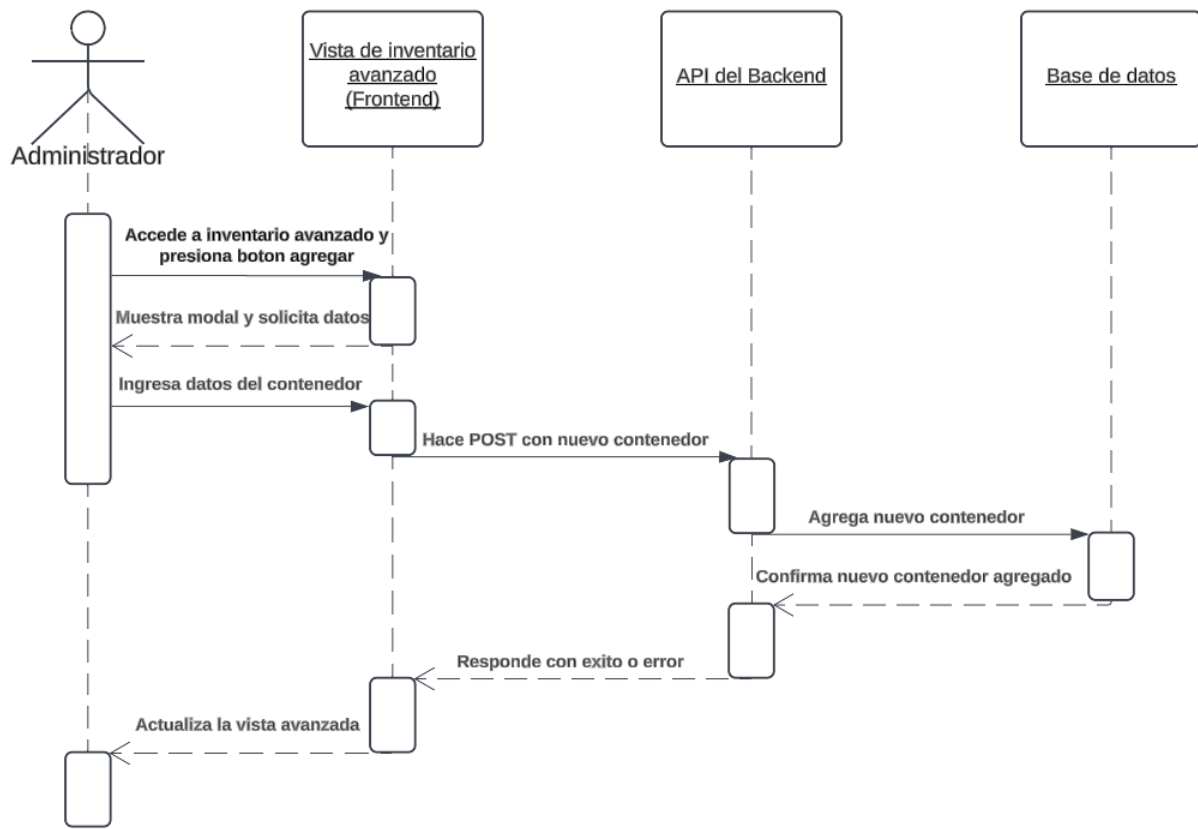


Figura 3-5. Diagrama de secuencia para agregar un nuevo contenedor al sistema
Fuente: Elaboración propia

Diagrama de componentes:

A continuación, se muestra un diagrama de componentes que presenta los distintos componentes de la aplicación dados por su enfoque, siendo estos contenidos en el frontend, Backend o el componente de la base de datos. El frontend tiene a los componentes que representan a las distintas vistas de la aplicación, junto con sus interacciones con las interfaces del Backend. El Backend incluye a los componentes que representan a la API y a los Websockets, que disponen de interfaces que permiten gestionar contenedores (obtener, agregar y eliminar) y proporcionar actualizaciones en tiempo real sobre los contenedores respectivamente. Además, el componente base de datos proporciona la interfaz que conecta con el Backend y permite administrar los contenedores del sistema.

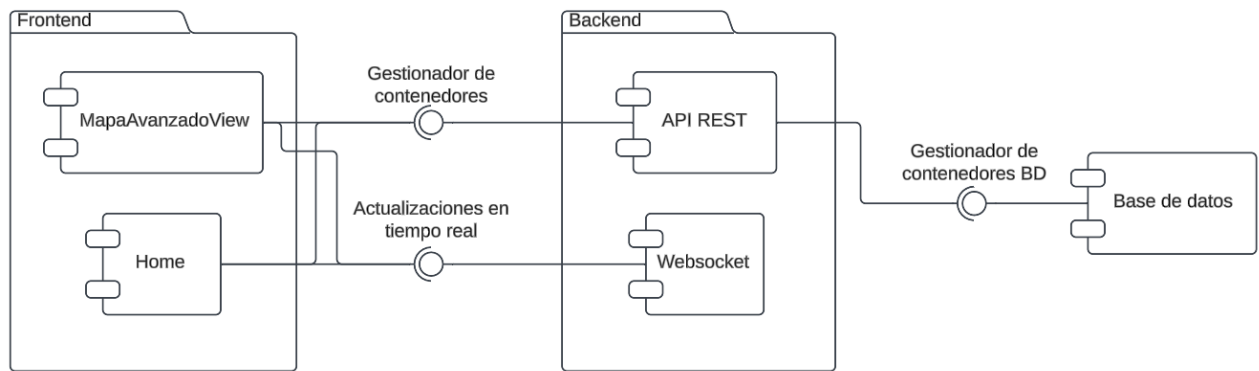


Figura 3-6. Diagrama de componentes del proyecto
Fuente: Elaboración propia

Diagrama de despliegue:

A continuación, es presentado el diagrama de despliegue, el cual nos indica los dispositivos físicos de la aplicación. La arquitectura del proyecto es monolítica modular, por lo que la aplicación está contenida en dos nodos. El nodo del servidor de la aplicación contiene tanto el lado del cliente (frontend) como el servidor (backend). Además, el servidor que contiene la API se comunica mediante protocolo TCP/IP para gestionar los contenedores de la base de datos.

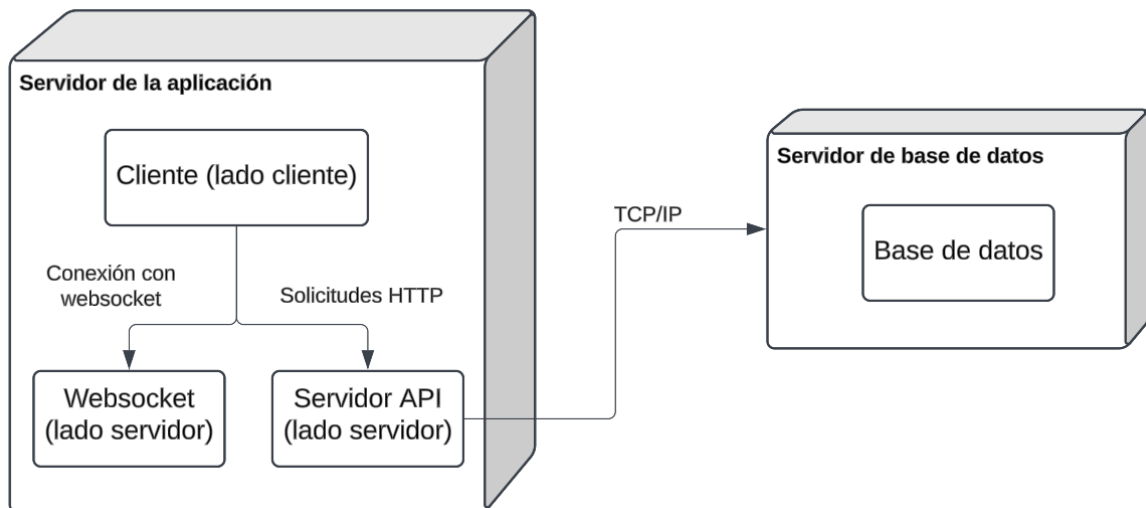


Figura 3-7. Diagrama de despliegue del proyecto
Fuente: Elaboración propia

3.2 Diseño de Componentes

En esta sección se abordará en la arquitectura y estructura de los distintos componentes en React, detallando como se organizan dentro de la aplicación web. Se profundizará en las buenas prácticas y

patrones utilizados durante el desarrollo. Finalmente, se explicará cómo se comunica el frontend con el backend.

3.2.1 Arquitectura y estructura de los componentes.

Arquitectura basada en componentes:

Las distintas vistas y funcionalidades de la aplicación web se descomponen en componentes más pequeños que pueden ser reutilizados.

La arquitectura basada en componentes proporciona organización y modularización en el código, lo que facilita el mantenimiento y gestión de aplicaciones.

Implementación de la arquitectura:

- **Componentes principales:** Componentes que representan las diferentes vistas de la aplicación. Estos componentes actúan como contenedores de componentes comunes.
- **Componentes comunes:** Componentes que poseen funcionalidades específicas dentro de las vistas.

Clean Architecture:

Patrón arquitectónico basado en la separación de responsabilidades. Es un enfoque para organizar y estructurar el código en un proyecto de software de manera que sea flexible, mantenible, escalable y fácil de probar [18].

Estructura de carpetas:

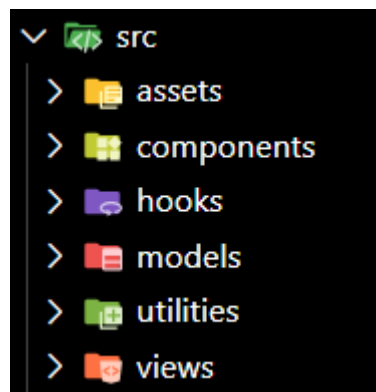


Figura 3-8. Estructura de carpetas de la aplicación con Clean Architecture

El proyecto en React se divide en las carpetas:

- **assets:** Carpeta con recursos de la aplicación (imágenes, iconos, texturas, etc....)
- **components:** Contiene a los distintos componentes comunes de la aplicación. Cada componente tiene una carpeta con su archivo de lógica y estilos.
- **hooks:** Contiene los diferentes hooks de la aplicación.
- **models:** Contiene los archivos para representar entidades.
- **utilities:** Contiene los archivos que permiten dar formato a texto.

- **views:** Contiene los componentes con las distintas vistas de la aplicación (componentes principales). Cada vista tiene su archivo de lógica y estilos.

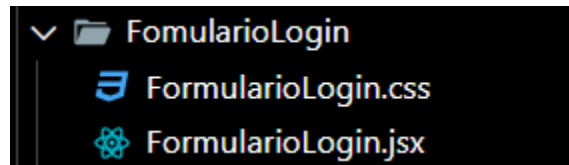


Figura 3-9. Carpeta con el contenedor FormularioLogin dentro de la carpeta components

Librerías instaladas en el proyecto:

- **"react-bootstrap":** Librería que nos permite usar elementos de Bootstrap.
- **"react-hot-toast":** Librería que nos permite crear componentes "toast" de forma rápida y personalizada.
- **"react-icons":** Librería que nos provee una gran variedad de iconos para nuestros componentes.
- **"socket.io-client":** Librería usada para todo lo que tenga que ver con los Websockets.
- **"three":** Librería para crear gráficos 3D en una aplicación web.

Responsividad de la aplicación:

Fue implementada responsividad para la página con Bootstrap. La aplicación cambia la disposición de sus componentes adaptándose a pantallas medianas y grandes (tabletas y dispositivos de escritorio).

Componentes principales:

A continuación, se listarán los componentes contenedores, que representan las distintas vistas de la aplicación.

- **LoginView (Vista de Login):**
 - **Descripción:** Vista inicial de la aplicación web que contiene el nombre del proyecto y un formulario de inicio de sesión, que solicita nombre de usuario y contraseña para los usuarios.
 - **Componentes comunes utilizados:** FormularioLogin
- **Home (Vista 3D del patio de contenedores):**
 - **Descripción:** Vista que presenta el mapa en 3D del patio de contenedores de Aexsa. Las torres de contenedores son presentadas en grillas de 5 por 5 contenedores de la A hasta la G. Adicionalmente se dispone de un recuadro expandible que nos indica la disponibilidad de contenedores que tiene una columna de alguna de las distintas torres.
 - **Componentes comunes utilizados:** ThreeMap y ColorLegend
- **MapaAvanzadoView (Vista de inventario avanzado de contenedores):**
 - **Descripción:** Vista que dispone de una grilla en 2D de las distintas torres de contenedores dado por su profundidad. Además de la profundidad, los contenedores están ubicados en una fila y altura específica, dada por los números que se encuentran a la izquierda y abajo de la grilla. Se puede navegar por las torres de contenedores usando los botones de avanzar y retroceder de la barra superior de torres. Se puede localizar un contenedor en específico

ingresando su identificador en el buscador, que se destacará en color amarillo. Además, es posible agregar o eliminar contenedores usando sus respectivos botones ubicados a la derecha del buscador. Esta vista también presenta la hora actual dependiendo de la hora configurada en el dispositivo.

- **Componentes comunes utilizados:** BotonAgregarContenedor, BotonEliminarContenedor, BarraTorres, BuscadorContenedor, NumerosLaterallzquierda, NumerosAbajo, BotonesProfundidad y GrillaContenedores.
- **TareasView (Vista de tareas):** La vista de tareas contiene una tabla con las distintas tareas que debe realizar un operador de grúa. Al seleccionar una tarea, se despliega una checklist con los movimientos que este debe realizar para completarla.

Componentes comunes:

A continuación, se listarán los componentes comunes, que se encuentran dentro de los componentes contenedores.

Componente: BarraTorres

- **Descripción:** Este componente es una barra que indica la torre actual de contenedores que se ve actualmente, tiene un botón para avanzar y otro para retroceder en las torres. Las torres de contenedores van de la letra A hasta la G.
- **Funcionalidades principales:**
 - **Cambio de torres:** Los botones permiten navegar entre las distintas torres de contenedores, con la opción de avanzar o retroceder. El contenido de la grilla de contenedores en el inventario avanzado se actualiza al moverse de torre.
 - **Torre actual:** El texto centrado de la barra nos permite saber en qué torre se está posicionado.
- **Diseño y herramientas implementadas:**
 - **React-icons:** Para representar los botones para avanzar y retroceder, usando los componentes FaAngleLeft y FaAngleRight.
 - **Estilos:** El componente usa un archivo CSS para el diseño y animaciones de la barra de torres.



Figura 3-10. Componente BarraTorres

Componente: BotonAgregarContenedor

- **Descripción:** Este componente es un botón que permite agregar un nuevo contenedor al sistema. Al presionarlo se abre una ventana emergente, en la que el usuario puede ingresar el identificador y la ubicación del nuevo contenedor. El componente comprueba si el contenedor ya existe antes de añadirlo.

- **Funcionalidades principales:**

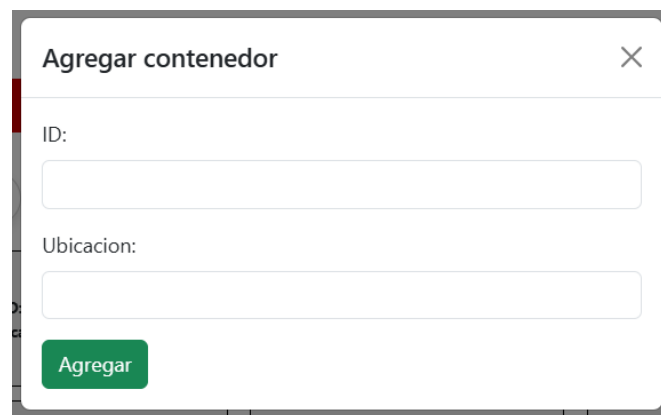
- **Mostrar modal:** Al oprimir el botón se despliega una ventana emergente que solicita los datos de un contenedor nuevo para agregar al sistema.
- **Agregar contenedor:** La ventana emergente del botón, nos permite ingresar los datos de un nuevo contenedor para agregarlo al sistema.
- **Verificar existencia de contenedor:** Al intentar agregar un contenedor, se comprueba si el contenedor ya existe (el identificador o ubicación ya existen en el sistema). Si el contenedor existe, se muestra un mensaje de error y el contenedor no se agrega.

- **Diseño y herramientas implementadas:**

- **React-icons:** Para representar el botón de agregar contenedores, usando el componente FaPlus.
- **React-hot-toast:** Para mostrar el mensaje de error que aparece brevemente.



Figura 3-11. Componente BotonAgregarContenedor



La imagen muestra una ventana modal con el título "Agregar contenedor" y un botón de cerrar (X) en la esquina superior derecha. Dentro de la ventana, hay dos campos de entrada de texto: "ID:" y "Ubicacion:". Debajo de estos campos, hay un botón verde con el texto "Agregar".

Figura 3-12. Ventana emergente que solicita los datos del nuevo contenedor

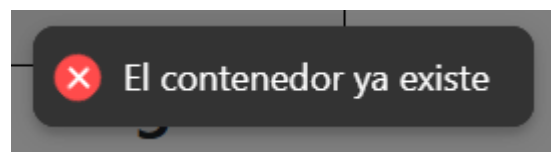


Figura 3-13. Mensaje de error arrojado si el contenedor que se intenta agregar ya existe

Componente: BotonEliminarContenedor

- **Descripción:** Este componente es un botón que permite eliminar un contenedor del sistema. Al presionarlo se abre un modal en el que el usuario puede ingresar el identificador del contenedor que requiera eliminar. El componente comprueba si el contenedor no existe antes de eliminarlo.

- **Funcionalidades principales:**

- **Mostrar modal:** Al oprimir el botón se despliega una ventana emergente que solicita el identificador de un contenedor para ser eliminado.
- **Eliminar contenedor:** La ventana emergente del botón, nos permite ingresar el identificador de un contenedor que será eliminado del sistema.
- **Verificar existencia de contenedor:** Al intentar eliminar un contenedor, se comprueba si el contenedor no existe (el identificador no existe en el sistema). Si el contenedor no existe, se muestra un mensaje de error indicando que el contenedor no existe.

- **Diseño y herramientas implementadas:**

- **React-icons:** Para representar el botón de eliminar contenedores, usando el componente FaTruckMoving
- **React-hot-toast:** Para mostrar el mensaje de error que aparece brevemente.



Figura 3-14. Componente BotonEliminarContenedor

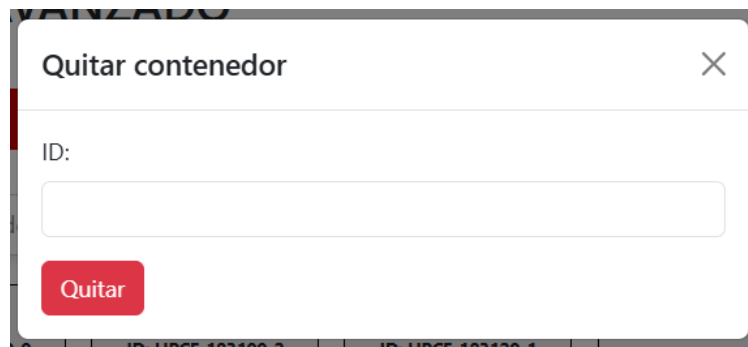


Figura 3-15. Ventana emergente que solicita los datos de un contenedor a eliminar

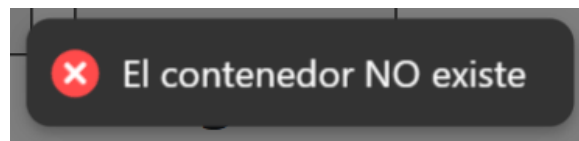


Figura 3-16. Mensaje de error arrojado si el contenedor que se intenta eliminar no existe

Componente: BotonesProfundidad

- **Descripción:** Este componente muestra 3 botones que permiten cambiar la profundidad de la grilla 2D de los contenedores. Existen 3 profundidades: contenedores de adelante, contenedores del medio y contenedores de atrás.

- **Funcionalidades principales:**
 - **Cambiar profundidad:** Es posible cambiar la profundidad actual presionando su respectivo botón.
- **Diseño y herramientas implementadas:**
 - **Estilos:** El componente usa un archivo CSS para su diseño y animaciones.



Figura 3-17. Componente BotonesProfundidad

Componente: BuscadorContenedor:

- **Descripción:** Este componente es una barra de búsqueda que permite buscar un contenedor en específico ingresando su identificador y presionando el botón para buscar. Si el contenedor existe, este se resalta.
- **Funcionalidades principales:**
 - **Contenedor específico:** El buscador permite buscar un contenedor en específico que se desee localizar, usando su identificador. Al encontrar el contenedor, la grilla de contenedores se posiciona en la localización exacta de este, cambiando la torre, y profundidad exacta de los componentes respectivos. Si se busca un contenedor que no existe, no ocurre nada.
 - **Contenedor resaltado:** Al buscar un contenedor con el buscador, este se resaltará en un color amarillo si existe.
- **Diseño y herramientas implementadas:**
 - **React-icons:** Para representar el botón que busca contenedores, usando el componente FaSearch.
 - **Estilos:** El componente usa un archivo CSS para su diseño y animaciones.

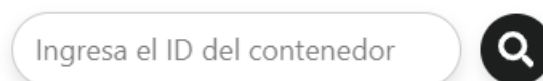


Figura 3-18. Componente BuscadorContenedor

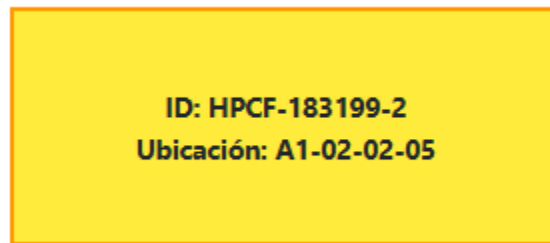


Figura 3-19. Contenedor buscado con BuscadorContedor resaltado

Componente: ColorLegend

- **Descripción:** Este componente es un botón que, al presionarlo, despliega un cuadro que indica la cantidad de contenedores que se pueden ingresar a una columna dado por color.
- **Funcionalidades principales:**
 - **Colores de disponibilidad:** Al presionar el botón, se presentan las distintas disponibilidades que presentan las columnas de contenedores, indicadas por colores.
- **Diseño y herramientas implementadas:**
 - **React-icons:** Para representar el botón que se expande y muestra la leyenda, usando los componentes FaChevronUp, FaChevronDown
 - **Estilos:** El componente usa un archivo CSS para su diseño y animaciones.



Figura 3-20. Componente ColorLegend

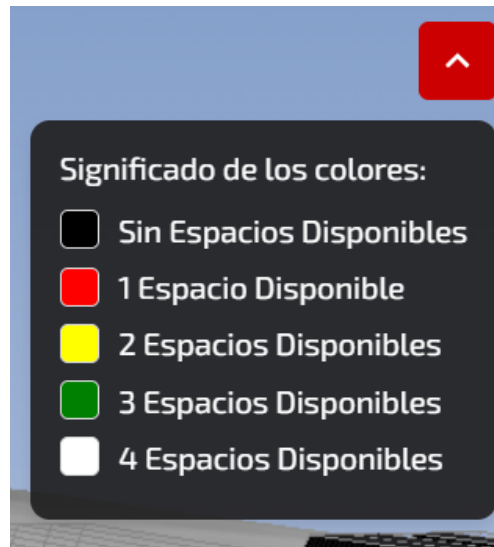


Figura 3-21. Ventana que se muestra al presionar el botón y tiene las disponibilidades de contenedores

Componente: GrillaContenedores

- **Descripción:** Este componente es una grilla de 5 por 5 que muestra los contenedores que existen en el sistema. Si el contenedor existe, se muestra su identificador y ubicación, si no existe se muestra “vacío”.
- **Funcionalidades principales:**
 - **Crear contenedor:** Si existe un contenedor, se crea y es representado su contenido con su identificador y su ubicación. En caso de que un contenedor no exista, se muestra un contenedor que dice “Vacío”.
 - **Crear grilla 2D:** Se crea una grilla de 5 por 5 de contenedores (representando las columnas y alturas que puede tener un contenedor).
- **Diseño y herramientas implementadas:**
 - **Estilos:** El componente usa un archivo CSS para el diseño y animaciones de la grilla de contenedores.

ID: HPCF-183199-9 Ubicación: A1-02-01-05	ID: HPCF-183199-2 Ubicación: A1-02-02-05	ID: HPCF-183129-1 Ubicación: A1-02-03-05	Vacio	Vacio
ID: ZRAF-183300-2 Ubicación: A1-02-01-04	ID: ZRAF-183300-3 Ubicación: A1-02-02-04	ID: HPCF-083002-1 Ubicación: A1-02-03-04	ID: HPCF-783802-1 Ubicación: A1-02-04-04	ID: HPCF-083100-1 Ubicación: A1-02-05-04
ID: ZRCF-183392-2 Ubicación: A1-02-01-03	ID: LDGN-232393-7 Ubicación: A1-02-02-03	ID: HPCF-183102-8 Ubicación: A1-02-03-03	ID: HPCF-183102-9 Ubicación: A1-02-04-03	ID: HPCF-083707-1 Ubicación: A1-02-05-03
ID: ZRCF-183392-1 Ubicación: A1-02-01-02	ID: MSCU-528135-5 Ubicación: A1-02-02-02	ID: QBJV-161208-3 Ubicación: A1-02-03-02	ID: HPCF-183102-4 Ubicación: A1-02-04-02	ID: HPCF-183102-6 Ubicación: A1-02-05-02
ID: MSCU-328115-3 Ubicación: A1-02-01-01	ID: MZCP-915101-9 Ubicación: A1-02-02-01	ID: PPJV-183208-3 Ubicación: A1-02-03-01	ID: EEEE-111111-2 Ubicación: A1-02-04-01	ID: HPCF-183102-1 Ubicación: A1-02-05-01

Figura 3-22. Componente GrillaContenedores

Componente: HoraActual

- **Descripción:** Este componente obtiene la hora actual del dispositivo, le otorga el formato “horas:minutos” y la retorna para ser usada.
- **Funcionalidades principales:**
 - **Hora dinámica:** Es mostrada la hora actual dependiendo del dispositivo, actualizándose en tiempo real.
- **Diseño y herramientas implementadas:**
 - **Hooks de React:** Es utilizado el hook useState para cambiar actualizar la hora actual de manera periódica.

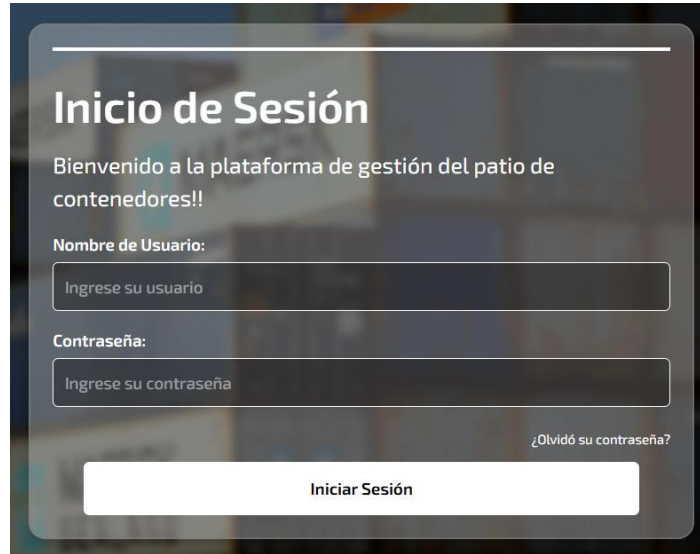
15:08

Figura 3-23. Componente HoraActual

Componente: FormularioLogin

- **Descripción:** Este componente es un formulario de inicio de sesión, que solicita usuario y contraseña que, al ingresarlos y oprimir su botón “Iniciar Sesión”, se nos dirigirá a la vista del home con el mapa en 3D del patio de contenedores.
- **Funcionalidades principales:**
 - **Iniciar sesión:** Al llenar el formulario de Login con los datos solicitados, ingresamos a la vista home.

- **Diseño y herramientas implementadas:**
 - **Estilos:** El componente usa un archivo CSS para el diseño y animaciones del formulario.



Inicio de Sesión

Bienvenido a la plataforma de gestión del patio de contenedores!!

Nombre de Usuario:

Contraseña:

[¿Olvidó su contraseña?](#)

Iniciar Sesión

Figura 3-24. Componente FormularioLogin

Componente: NumerosAbajo

- **Descripción:** Componente que contiene los números que representan las columnas de los contenedores en la parte inferior de la grilla.
- **Funcionalidades principales:**
 - Este componente solo muestra las distintas columnas de contenedores.

1 2 3 4 5

Figura 3-25. Componente NumerosAbajo

Componente: NumerosLaterallzquierda

- **Descripción:** Componente que contiene los números que representan las alturas de los contenedores en la parte lateral izquierda a la grilla 2D del inventario avanzado de contenedores.
- **Funcionalidades principales:**
 - Este componente solo muestra las distintas alturas de contenedores.

5

4

3

2

1

Figura 3-26. Componente NumerosLaterallzquierda

Componente: SideBarLateral

- **Descripción:** Este componente es un botón del tipo hamburguesa, que al presionarlo se expande y despliega botones para ir a la vista del home, inventario avanzado, Tareas y para salir de la aplicación.
- **Funcionalidades principales:**
 - **Botón expansible:** El botón se expande al presionarlo, desplegando más botones con opciones.
 - **Navegación:** Los botones del botón expandido permiten navegar entre las distintas vistas de la aplicación.
- **Diseño y herramientas implementadas:**
 - **React-icons:** Para representar los botones que representan las distintas vistas, usando los componentes FaSignOutAlt, FaBars, FaTasks, FaHouse y FaBox.
 - **Estilos:** El componente usa un archivo CSS para el diseño y animaciones del botón de la barra lateral.



Figura 3-27. Componente SideBarLateral



Figura 3-28. Componente SidebarLateral Expandido

Componente: ThreeMap

- **Descripción:** Este componente es un mapa en 3D del patio de contenedores del patio de Aexsa. El mapa cuenta con torres de contenedores que van de la A hasta la G, con grillas de contenedores. La cámara puede desplazarse, acercarse, alejarse y rotar en el mapa.
- **Funcionalidades principales:**
 - **Mostrar contenedores:** Se muestran todos los contenedores ingresados en el sistema.
 - **Colores de disponibilidad:** Las columnas de contenedores presentan distintos colores que indican, cuantos contenedores más pueden ser ingresados.
 - **Movilidad:** Es posible desplazarse, rotar, acercarse y alejarse, usando la cámara del mapa.
- **Diseño y herramientas implementadas:**
 - **Texturas:** El mapa en 3D, presenta imágenes usadas como textura para el piso y contenedores.

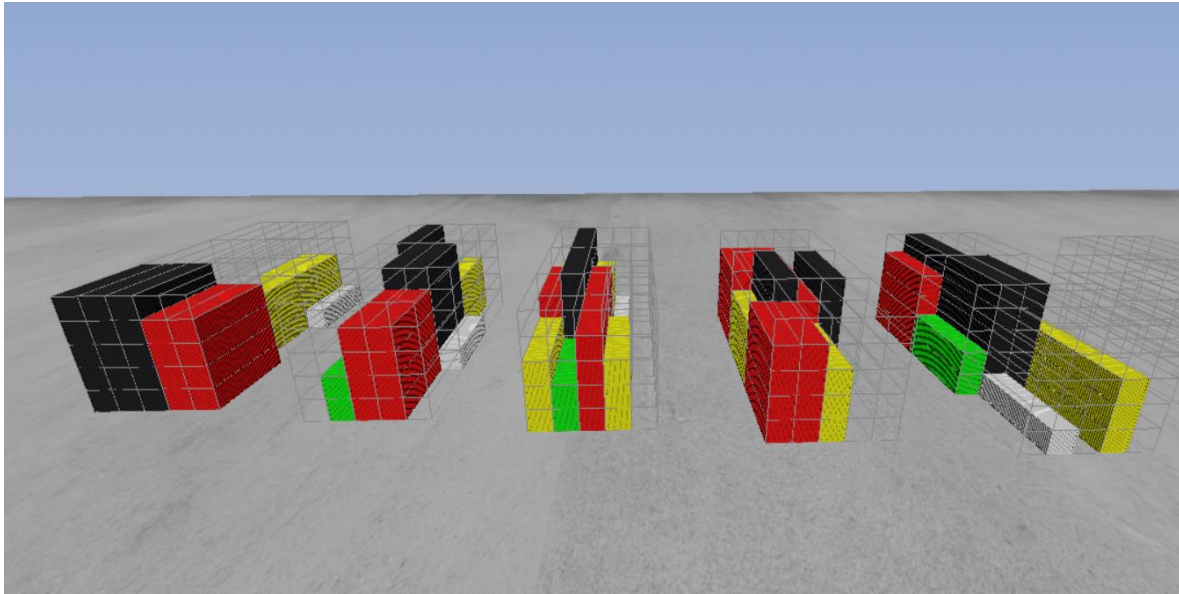


Figura 3-29. Componente ThreeMap

3.2.2 Uso de buenas prácticas y patrones de diseño.

Buenas prácticas:

- **Uso de estructura modular:** Con Clean Architecture se estructura el proyecto en carpetas lógicas, lo que permite una mejor legibilidad y organización de código.
- **Índice para los componentes:** Existe un archivo que sirve como índice para los componentes, que evita redundancia y permite un mejor entendimiento del código.
- **Mensajes de error:** Siempre que ocurre un error, se muestran mensajes claros para los usuarios (sin términos técnicos como referencia a la base de datos).

Patrones de diseño:

- **Custom hooks:** Se implementará el uso de custom hooks para los componentes que comparten lógica entre sí. Por ejemplo, tanto el botón para agregar como eliminar contenedores usan la misma lógica para verificar si un contenedor ya existe en el sistema, por lo que fue creado un hook con dicha función (useVerificarContenedor en la carpeta de hooks).

3.2.3 Integración con el sistema general.

El frontend se integra al resto del sistema mediante solicitudes HTTP y Websockets.

Solicitudes HTTP hacia el backend:

Son empleados métodos http para la comunicación con el Backend, principalmente para el manejo de los datos de los contenedores. Se realizan solicitudes GET, para obtener la lista completa de los contenedores de la base de datos. Se realizan solicitudes POST, para agregar un contenedor nuevo a la base de datos. Se realizan solicitudes DELETE para eliminar un contenedor de la base de datos.

Es utilizado el método fetch de React para realizar las operaciones CRUD en la base de datos.

Al momento de solicitar los datos con GET se obtiene un JSON que se debe transformar a un arreglo de objetos, siendo los objetos, los contenedores con sus distintos datos:

```
[
  {
    "contenedor": "MSCU-988171-3",
    "ubicacion": "G1-02-04-02",
    "zona": "COMERCIAL",
    "visado": false
  },
  {
    "contenedor": "AASA-663552-1",
    "ubicacion": "E1-01-04-02",
    "zona": "COMERCIAL",
    "visado": false
  },
  {
    "contenedor": "ERSA-654552-1",
    "ubicacion": "E2-01-02-02",
    "zona": "COMERCIAL",
    "visado": false
  },
  {
    "contenedor": "HGDA-553552-1",
    "ubicacion": "E3-01-02-01",
    "zona": "COMERCIAL",
    "visado": false
  },
  {
    "contenedor": "MRKU-432105-1",
    "ubicacion": "D3-02-02-03",
    "zona": "COMERCIAL",
    "visado": false
  }
],
```

Figura 3-30. Arreglo de contenedores recibidos con GET

```
{
  "contenedor": "MSCU-988171-3",
  "ubicacion": "G1-02-04-02",
  "zona": "COMERCIAL",
  "visado": false
},
```

Figura 3-31. Contenedor recibido con GET

Websockets:

La aplicación tiene la característica de mostrar, agregar y eliminar contenedores en tiempo real del sistema utilizando el protocolo websocket.

Este protocolo comienza una comunicación constante entre el cliente y el servidor, lo que permite enviar y recibir datos con este, en tiempo real.

En la aplicación principal se comienza la comunicación con el websocket, que es entregado a las vistas correspondientes. En la aplicación se cuenta con un evento que permite actualizar los contenedores en el inventario avanzado de contenedores, al momento de agregar uno con los botones respectivos.

3.3 Detalles de la codificación y desarrollo.

Como fue mencionado antes, el proyecto fue realizado implementando la metodología ágil Scrum.

El proyecto cuenta con las siguientes historias de usuario en total:

- Como administrador y operador quiero ver las ubicaciones de los contenedores en el mapa 3D.
- Como administrador y Operador quiero que la base de datos de contenedores de la empresa se muestre en el mapa.
- Como administrador y operador necesito poder navegar por las páginas de la web.
- Como administrador y Operador quiero poder localizar un contenedor en específico.
- Como administrador quiero asignar tareas a los operadores.
- Como operador quiero ver la ubicación de los contenedores en tiempo real.
- Como administrador quiero poder agregar, eliminar y editar contenedores del mapa.
- Como operador quiero que los contenedores que muevo queden registrados en el mapa con su ubicación nueva.

El desarrollo fue dividido en 3 sprints de aproximadamente 3 a 4 semanas cada uno.

Sprints del proyecto:

Sprint 1 (duración: agosto 22 - septiembre 11)

Backlog:

- Como administrador y operador quiero ver las ubicaciones de los contenedores en el mapa 3D.

Tareas abordadas por quien escribe la tesina:

- Configurar entorno de desarrollo.
- Creación de la primera vista.
- Avance de la creación del mapa.

Sprint 2 (duración: octubre 01 - octubre 18)

Backlog:

- Como administrador y Operador quiero que la base de datos de contenedores de la empresa se muestre en el mapa.
- Como administrador y operador necesito poder navegar por las páginas de la web.
- Como administrador y Operador quiero poder localizar un contenedor en específico.
- Como administrador quiero asignar tareas a los operadores.

Tareas abordadas por quien escribe la tesina:

- Agregar vistas restantes.
- Integración de distintos componentes

Sprint 3 (duración: octubre 27 - noviembre 15)

Backlog:

- Como operador quiero ver la ubicación de los contenedores en tiempo real.
- Como administrador quiero poder agregar, eliminar y editar contenedores del mapa.
- Como operador quiero que los contenedores que muevo queden registrados en el mapa con su ubicación nueva.

Tareas abordadas por quien escribe la tesina:

- Implementar eliminar contenedores.

- Implementar lógica movimiento.
- Optimizar página.

3.4 Pruebas y Validación

Para la revisión de que el Frontend cumple con los requerimientos planteados se realizarán:

- **Validación con usuarios:** Para que los usuarios finales nos indiquen si la interfaz es fácil e intuitiva de usar.
- **Criterios de evaluación:** Se evaluará el comportamiento de la aplicación en distintos dispositivos (tabletas y dispositivos de escritorio).

3.4.1 Estrategias de testing aplicadas a los componentes.

Para asegurar el correcto funcionamiento de los diferentes componentes de la aplicación, se llevaron a cabo pruebas de integración al final de cada uno de los tres sprints en los que se divide el desarrollo del proyecto.

- **Pruebas de integración:** Estas pruebas fueron implementadas con el objetivo de verificar que los componentes interactúan correctamente entre sí. Un ejemplo de estas pruebas sería comprobar que un formulario envíe los datos de manera correcta al Backend, asegurando que la comunicación entre el frontend y el Backend se realice de forma correcta.

3.4.2 Resolución de problemas y optimización

Las estrategias empleadas para la resolución de problemas fueron técnicas de depuración (Debugging), que incluyeron las siguientes:

- **Uso de herramientas de desarrollo del Navegador:** Se utilizó la función console.log y sus variantes para imprimir mensajes y valores directamente en la consola proporcionada por el navegador, con el objetivo de identificar errores y comportamientos inesperados en el código.
- **Pruebas de validación de entrada:** Se realizaron pruebas en los formularios de agregar y eliminar contenedores utilizando distintos tipos de entradas erróneas (como campos vacíos que no deberían estar vacíos o formatos incorrectos) con el fin de asegurar que el sistema maneje de forma correcta estos casos, proporcionando los mensajes de error correspondientes.
- **Documentación de código:** Se añadieron comentarios en los distintos archivos para documentar y explicar el propósito, la lógica y la funcionalidad del código.

Las estrategias que se usarán para la optimización son:

- **Refactorización de código con Clean Architecture:** Se usará Clean Architecture para simplificar, mejorar legibilidad y eliminar redundancias en el código.
- **Optimización del mapa:** El mapa puede bajar el rendimiento de la aplicación en general, por lo que se implementa la funcionalidad de desmontarlo cuando no se use, lo que permite a la aplicación general no tener problemas de ralentización adicionales.



4 Conclusiones

A lo largo del desarrollo de este proyecto para una aplicación web de una empresa de almacenaje extraportuario, el autor de esta tesina ha aprendido una gran cantidad de conceptos y herramientas claves para la creación de interfaces web. Uno de los aspectos más valiosos fue el profundizar en las buenas prácticas que se deben llevar a cabo si es que se quiere asegurar que una aplicación sea escalable y fácil de mantener.

El desafío más importante que enfrentó el autor de esta tesina fue el adaptarse al proceso de aprender a la par que programaba. Al utilizar tecnologías nuevas, tuvo que aprender sobre la marcha y resolver problemas en la medida que surgían.

La comunicación constante con el equipo resultó en un factor clave para que todos los aspectos del frontend estuvieran alineados a las expectativas del cliente.

De cara al futuro, se considera crucial el implementar por completo la vista de tareas para operadores de grúa, que permita gestionar de manera más eficiente las actividades diarias que estos llevan a cabo. No obstante, debido a la priorización de otras funcionalidades de mayor importancia, quedará fuera la implementación completa tanto de la vista de tareas como el inicio de sesión.

5 Referencias

- [1] Sitio web KLog, «Almacenes extraportuarios: una solución logística estratégica,» *KLog.co*, 10 julio 2024. [En línea]. Available: <https://klog.co/blog/almacenes-extraportuarios-una-solucion-logistica-a-estrategica#:~:text=de%20una%20empresa.,%C2%BFQu%C3%A9%20son%20los%20almacenes%20extraportuarios?,flexibilidad%20para%20gestionar%20las%20cargas>.
- [2] C. Loyola, «¿Qué es un Contenedor Refrigerado o Reefer?,» *REEFET Spa*, 18 noviembre 2019. [En línea]. Available: <https://www.reefet.cl/post/que-es-un-contenedor-refrigerado>.
- [3] G. Bailon, «¿Por qué utilizar Scrum en la gestión de proyectos?,» *TAKTIC*, 7 mayo 2019. [En línea]. Available: <https://taktic.es/blog/por-que-utilizar-scrum-en-la-gestion-de-proyectos/>.
- [4] E. Rodh, «Casos de Éxito de Scrum,» *Cohesiva*, sin fecha. [En línea]. Available: <https://www.cohesivaempresarial.com/post/casos-de-%C3%A9xito-de-scrum-la-f%C3%B3rmula-secreta-para-el-%C3%A9xito-empresarial>. [Último acceso: 01 12 2024].
- [5] Sitio web eClass, «Revisa los casos de éxito de 3 empresas utilizando Scrum,» *eClass*, sin fecha. [En línea]. Available: <https://blog.ecclass.com/revisa-el-caso-de-exito-de-3-empresas-utilizando-scrum>.
- [6] M. Coppola, «Frontend y backend: qué son, en qué se diferencian y ejemplos,» *HubSpot*, 5 noviembre 2021. [En línea]. Available: <https://blog.hubspot.es/website/frontend-y-backend#que-es>.
- [7] R. Ballerini, «HTML, CSS y Javascript, ¿cuáles son las diferencias?,» *Alura Latam Cursos*, 23 julio 2021. [En línea]. Available: <https://www.aluracursos.com/blog/html-css-javascript-cuales-son-las-diferencias>.
- [8] A. Casero, «Historia del desarrollo frontend,» *KeepCoding*, 15 julio 2024. [En línea]. Available: <https://keepcoding.io/blog/historia-del-desarrollo-frontend/#:~:text=La%20historia%20del%20desarrollo%20frontend%20se%20remonta%20a%20los%20primeros,por%20texto%20e%20im%C3%A1genes%20est%C3%A1ticas>.
- [9] C. Miller, «The Evolution of Front End Development,» *RemoteMode*, 19 enero 2022. [En línea]. Available: <https://remotemode.com/blog/the-evolution-of-front-end-development/>.
- [10] S. M., «Tendencias frontend en 2024: qué retendrá a su cliente,» *WeblinIndia*, sin fecha. [En línea]. Available: <https://www.weblinindia.com/es/blog/frontend-trends-2024.html>.
- [11] M. García, «¿Qué son las aplicaciones nativas, web e híbridas?,» *ESDESIGN*, 6 octubre 2023. [En línea]. Available: <https://www.esdesignbarcelona.com/actualidad/disenio-web/apps-nativas-vs-hibridas>.
- [12] L. Laurent, «Arquitectura basada en componentes de React: Un estudio de caso,» *AppMaster*, 31 julio 2023. [En línea]. Available: <https://appmaster.io/es/blog/arquitectura-basada-en-componentes-react>.
- [13] V. Aristizabal, «Arquitectura de Componentes,» *Medium*, 20 febrero 2024. [En línea]. Available: <https://vanessamarely.medium.com/arquitectura-de-componentes-e48816925d51>.
- [14] D. A., «Qué es React: definición, características y funcionamiento,» *Hostinger*, 29 junio 2023. [En línea]. Available: https://www.hostinger.es/tutoriales/que-es-react#Caracteristicas_de_React.
- [15] Sitio web Three.js, «Fundamentals,» *Three.js*, sin fecha. [En línea]. Available: <https://threejs.org/manual/#en/fundamentals>.
- [16] Sitio web Vite, «Getting Started,» *Vite*, sin fecha. [En línea]. Available: <https://vite.dev/guide/>.
- [17] Sitio web React, «Escribir marcado con JSX,» *React*, sin fecha. [En línea]. Available: <https://es.react.dev/learn/writing-markup-with-jsx>.



- [18] Sitio web CodeScouts, «Clean Architecture,» *CodeScouts*, sin fecha. [En línea]. Available: <https://library.codescouts.academy/docs/clean-architecture/#:~:text=La%20arquitectura%20limpia%20es%20un,puertos%20y%20adap%20tadores%20entre%20otros>.



6 Anexos

Enlace al repositorio de GitHub del proyecto para su correcta comprensión:

- <https://github.com/LookSkys/Optainer>