UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Recreación del sistema macOS mediante tecnologías web reactivas

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Hernández Perales, Álvaro

Tutor/a: Albert Albiol, Manuela

CURSO ACADÉMICO: 2022/2023

Resumen

Las aplicaciones web son, cada vez más, populares por su facilidad de uso, desarrollo y los escasos requisitos de hardware necesarios para ejecutarlas. Por esto las interfaces web se han convertido en una de las áreas más activas del desarrollo web. Estas aplicaciones, mediante el uso de interfaces de usuario modernas, pretenden alcanzar una productividad similar a las aplicaciones nativas, reduciendo los requisitos del sistema y con la ventaja de ser multiplataforma.

En este trabajo, abordamos la cuestión de las interfaces y experiencias de usuario creadas en lenguajes web. En concreto, desarrollaremos una réplica del sistema operativo de escritorio de Apple, tratando de imitar patrones, interfaces y experiencias con el objetivo de experimentar cómo de nativa puede ser percibida una aplicación web.

Palabras clave: Sistema operativo, Lenguajes de programación web, ReactJS, CSS, Experiencia de usuario, Interfaces de usuario.

Resum

Les aplicacions web són, cada vegada més, populars per la seua facilitat d'ús, desenvolupament i els escassos requisits de sistema necessari per a executar-les. Per això les interfícies web s'han convertit en una de les àrees més actives de la programació web. Aquestes aplicacions, mitjançant l'ús d'interfícies d'usuari modernes, pretenen aconseguir una productivitat similar a les aplicacions natives, reduint els requisits del sistema i amb l'avantatge de ser multi plataforma.

En aquest treball, abordem la qüestió de les interfícies i experiències d'usuari creades en llenguatges web. En concret, desenvoluparem una rèplica del sistema operatiu d'escriptori d'Apple, tractant d'imitar patrons, interfícies i experiències amb l'objectiu d'experimentar com de nadiua pot ser percebuda una aplicació web.

Paraules clau: Sistema operatiu, Llenguatges de programació web, ReactJS, CSS, Experiència d'usuari, Interfícies d'usuari.

Abstract

Web applications are becoming increasingly popular because of their ease of use, ease of development and the low hardware requirements needed to run them. This is why web interfaces have become one of the most active areas of web development. These applications, using modern user interfaces, aim to achieve similar productivity to native applications, reducing system requirements and with the advantage of cross-platform.

In this paper, we address the issue of user interfaces and user experiences created in web languages. Specifically, we will develop a replica of Apple's desktop operating system, trying to mimic patterns, interfaces, and experiences with the goal of experimenting how native a web application can be perceived.

Keywords: Operative System, Web programming languages, ReactJS, CSS, User experience, User interfaces.

Índice

Índice de figuras

Índice de tablas

1 Introducción

En este primer capítulo se presenta una breve explicación del proyecto y se describe la motivación, objetivos, metodología del proyecto y estructura del documento.

1.1 Motivación

El diseño de aplicaciones que logren una interacción efectiva entre el usuario y el sistema es un área de investigación muy activa en la comunidad de diseño y programación web. Estas investigaciones involucran a expertos en ciencias de la computación, así como a diseñadores, científicos, psicólogos y otros profesionales.

En 2023, aproximadamente 5.160 mil millones de personas son usuarios activos de internet [1], lo que equivale a un 64.4% de la población mundial. Estos usuarios abarcan un rango de edad considerable por lo que los patrones, interfaces y accesibilidad web se han de adaptar a las necesidades de cada grupo de usuarios al igual que los sistemas operativos permiten la personalización, control y accesibilidad.

Durante las últimas décadas, la evolución de internet y los lenguajes de programación web han permitido que aplicaciones que requerían ser instaladas en un ordenador con altas prestaciones se puedan ejecutar hoy en día como una aplicación web. Parte de esta evolución se debe al cloud computing [2], un modelo de computación que posibilita el acceso flexible y bajo demanda a recursos informáticos, como almacenamiento, servidores y aplicaciones, a través de Internet. Estos recursos facilitan el desarrollo y alojamiento del *backend*. El backend es la parte de la aplicación web que se encarga de la lógica y el procesamiento de datos. Por otro lado, el *frontend* es la capa o parte de la aplicación web que se encarga de mostrar toda la información en pantalla, con la que el usuario puede interactuar. El presente Trabajo de Final de Grado (TFG) se enfocará en esta capa.

En la actualidad, la mayoría de las aplicaciones que usamos diariamente se han visto beneficiadas por las tecnologías y protocolos web, algunas de ellas incluso son ejecutadas directamente como aplicaciones web sobre una ventana en nuestro escritorio, lo que se conoce como aplicación híbrida.

Algunos ejemplos de dichas aplicaciones híbridas pueden ser Discord[1], una plataforma de mensajería y videollamadas, o Figma[2], una herramienta de diseño que ha revolucionado y democratizado el acceso al diseño colaborativo de calidad o Visual

[1] https://discord.com/
[2] https://www.figma.com/

Studio Code[3], un IDE de código escrito puramente en TypeScript[4] que se sitúa como el editor de código más popular del mundo[5]. Estas aplicaciones híbridas están ejecutadas sobre Electron[6], una plataforma que permite desarrollar y ejecutar aplicaciones web en entornos de escritorio.

La línea entre la web y el escritorio virtual del dispositivo es cada vez más difusa y por eso ha empezado a tomar más importancia que nunca las interfaces e interacciones persona-computador. En este trabajo investigaremos cómo de cerca pueden llegar a estar y cómo se pueden desarrollar experiencias e interfaces cercanas a las nativas usando lenguajes web.

1.2 Objetivos

El principal objetivo de este TFG es la recreación de la interfaz del sistema operativo de escritorio macOS Ventura[7], usando lenguajes de programación web. Esta recreación simulará desde el más mínimo detalle, como los colores, iconos o efectos, hasta el comportamiento, los atajos de teclado, animaciones y lógica del entorno de escritorio. Se ha desglosado el alcance del proyecto en varios sub-objetivos:

- Recreación del entorno de escritorio, con ventanas, Dock[8] y barra de menú de aplicaciones.
- Desarrollo de varios servicios nativos del sistema como reproducción de música, almacenamiento de archivos o sistema de notificaciones.
- Simulación fidedigna del comportamiento y aspecto nativo mediante lenguaje de maquetación web CSS.

Para lograr estos objetivos es necesario un conocimiento profundo de cómo funciona una aplicación web a bajo nivel (maquetación CSS avanzada, funcionamiento del DOM, ciclo de vida de los componentes de ReactJS, etc.) como del comportamiento nativo del sistema operativo y las diferentes relaciones con patrones web necesarias para traducir el funcionamiento nativo a lenguajes web.

[3] https://code.visualstudio.com/

[4] TypeScript es un lenguaje de programación de alto nivel, gratuito y de código abierto, desarrollado por Microsoft, que complementa a JavaScript y agrega la capacidad de realizar anotaciones de tipos estáticos opcionales.

[5] Datos oficiales de cada empresa, recogidos y visualizados en el siguiente video: https://www.youtube.com/watch?v=qwrgwS-K3Uk&ab_channel=EricCode

[6] https://www.electronjs.org/

[7] macOS Ventura es la versión 13.0.0. del sistema operativo de escritorio de Apple.

[8] El Dock es la barra que agrupa las aplicaciones en uso o fijadas en el sistema operativo macOS, lo que se conoce en Windows como barra de tareas.

1.3 Metodología

Las metodologías de desarrollo de software proponen maneras de crear y organizar las tareas en los proyectos software. Las dos corrientes metodológicas más conocidas son la tradicional y la ágil. A continuación, se detallan las ventajas e inconvenientes de cada una y se decidirá aquella que más se adecue a las necesidades del proyecto.

Por una parte, las metodologías tradicionales son apropiadas para proyectos en los que se cuenta con una amplia experiencia o se llevan a cabo en condiciones predecibles y estables, donde la estructura suele ser más rígida e inflexible.

Entre sus principales ventajas destaca la capacidad de realizar un seguimiento exhaustivo y detallado de cada fase del proyecto, al asignar de manera específica y concentrada los recursos y la atención en cada una de ellas, por lo que es más sencillo prevenir errores.

Una limitación significativa de las metodologías tradicionales es la complejidad asociada con la recopilación inicial de requisitos, la cual puede dificultar la adaptación a los cambios o necesidades emergentes a medida que avanza el proyecto y por tanto resultar en retrasos en su desarrollo.

Por otra parte, las metodologías ágiles se caracterizan por su flexibilidad y adaptabilidad a los posibles cambios en el proyecto, lo que lo hace ideal para proyectos propensos a presentar más cambios durante su desarrollo. Una de sus principales ventajas es que permite dividir el trabajo más libremente de forma que se obtiene un producto funcional lo antes posible. En cambio, las metodologías ágiles requieren de una involucración activa por parte del cliente.

El presente proyecto software se desarrollará conforme la metodología ágil Kanban[9], dado que el problema propuesto se puede dividir en iteraciones cada cual con más detalle y funcionalidad que la anterior. Es importante destacar que Kanban no es una técnica de planificación sino de organización de las tareas.

La forma en la que se dividirán las tareas será útil a la hora de medir el progreso y los objetivos logrados durante todo el proceso de desarrollo. Para la división y gestión de las tareas utilizaremos la herramienta Jira (Figura 1).

[9] Kanban es una metodología visual de gestión de proyectos que posibilita a los equipos visualizar y controlar sus flujos de trabajo y carga de trabajo de manera efectiva.

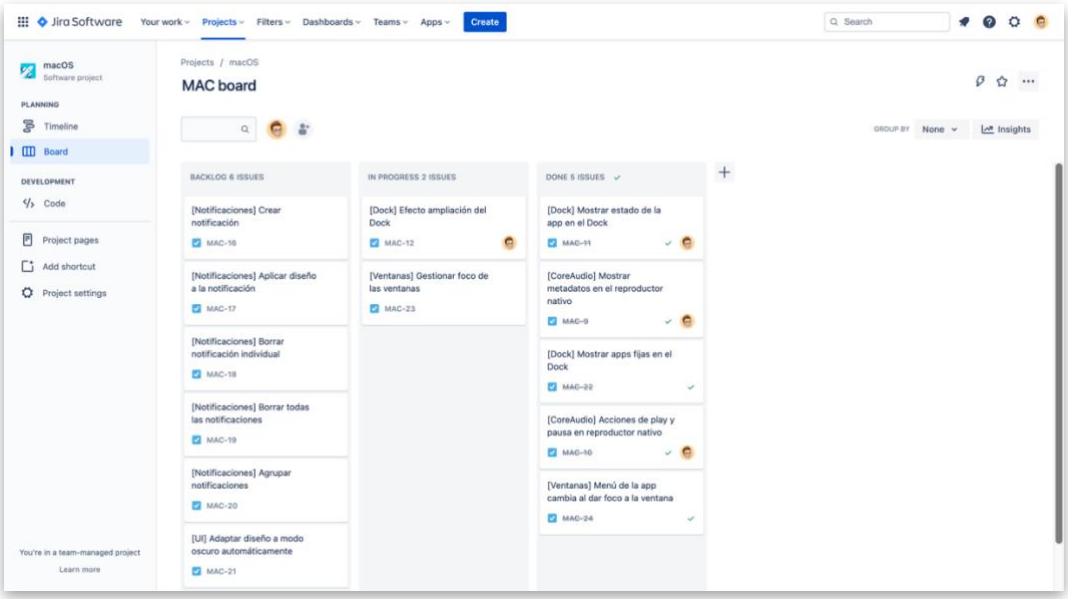

Figura 1: Tablero Kanban en Jira

1.4 Estructura del documento

El presente documento se divide en varios capítulos, los cuales se presentan y desarrollan de la siguiente forma:

- **Estado del arte.** Descripción de manera concisa las diferentes aproximaciones de distintos desarrolladores respecto a la recreación de interfaces de sistemas operativos populares. Además, se propondrá la solución a desarrollar en este proyecto, la cual pretende no solo mejorar las soluciones estudiadas anteriormente si no innovar en cuanto a interacción con el usuario y calidad del nivel de detalle.
- **Análisis del problema.** Análisis las posibles soluciones y sus peculiaridades, así como definir brevemente la solución propuesta y cómo se desarrollará.
- **Diseño de la solución.** Desglose a nivel técnico de la arquitectura del proyecto y las tecnologías y herramientas utilizadas para poder llevarlo a cabo.
- **Desarrollo de la solución.** Exposición de la implementación dividida en tres grupos principales, servicios, interfaces y aplicaciones. En esta sección se hará hincapié en las diferentes limitaciones de las tecnologías web y cómo se han resuelto.

- **Implantación.** Guía de la publicación del proyecto en la plataforma Vercel[10], para poder ser accedida de forma pública desde cualquier dispositivo.
- **Conclusiones y trabajo futuro.** Resumen de las conclusiones del proyecto y revisión de los objetivos logrados, así como descripción de posible trabajo futuro a modo de mejora.
- **Referencias.** Registro de las fuentes consultadas para la redacción del trabajo.

[10] Vercel es la compañía americana detrás del *framework* de desarrollo NextJS, que se vende como plataforma en la nube como servicio.

2 Estado del arte

El campo de las interfaces web se encuentra en constante evolución y es por eso por lo que desarrolladores y diseñadores han experimentado con lenguajes de maquetación como CSS y distintos *frameworks* con el objetivo de replicar las interfaces nativas de diferentes sistemas operativos.

En este capítulo se presentan algunas propuestas desarrolladas por diferentes programadores, realizando un desglose de sus funcionalidades y puntos a mejorar, y se expone la conclusión obtenida de esta exploración.

2.1 Aproximaciones similares a la propuesta

En esta sección analizaremos en detalle la implementación y diseño de diferentes propuestas desarrolladas por distintos programadores, así como su semejanza a la interfaz y comportamiento nativos de los diferentes sistemas operativos de Apple.

De entre las aproximaciones descubiertas en el estudio del mercado, se procede a presentar las 3 más completas. Cabe destacar que todos los proyectos estudiados solo incluyen la parte *frontend*, en ningún momento se comunican con APIs[11] o servicios de lado *backend* para obtener información.

2.1.1 macos-web

Desarrollado en Preact[12] y Typescript junto a módulos CSS y SCSS, macos-web[13] (Figura 2) es una aproximación muy reciente al sistema operativo de escritorio de Apple. Entre sus puntos fuertes encontramos el correcto uso de glassmorphism [3], un detalle de interfaz muy conocido en todo el diseño de Apple, además del uso de la paleta de colores oficiales de la plataforma documentada en Human Interface Guidelines [4] de Apple.

En cuanto a la apariencia, trata de parecerse al aspecto nativo del escritorio, pero se nota la falta de detalle en los estados como *:hover*[14] en la barra de menús de aplicaciones arriba a la izquierda. También el autor optó por aplicar sus propias animaciones en vez de tratar de imitar las nativas, por lo que no da la sensación de que sea una copia exacta.

---

[11] Una API o una interfaz de programación de aplicaciones es un conjunto de definiciones y protocolos que se usa para diseñar e integrar el software de las aplicaciones.

[12] Preact es una biblioteca de JavaScript, considerada la alternativa liviana (tan solo 3kb) a ReactJS.

[13] https://github.com/an0n7os/macos-web

[14] La pseudo-clase :hover de CSS se activa cuando el usuario interactúa con un elemento mediante un dispositivo señalador, sin necesariamente activarlo.

En cuanto a nivel de funcionalidad, podemos ver que la única aplicación funcional en pantalla es la calculadora, junto con el calendario que se puede ver en el Dock. Mientras, el resto de las aplicaciones no son funcionales ni muestran una interfaz, solo un mensaje de que está en construcción.

Se puede afirmar que este proyecto está cerca de hacerse pasar por una interfaz nativa, pero queda delatado por la interacción y la experiencia de usuario, donde usa patrones típicos de páginas web en vez de tratar de imitar y adaptar los patrones de escritorio.

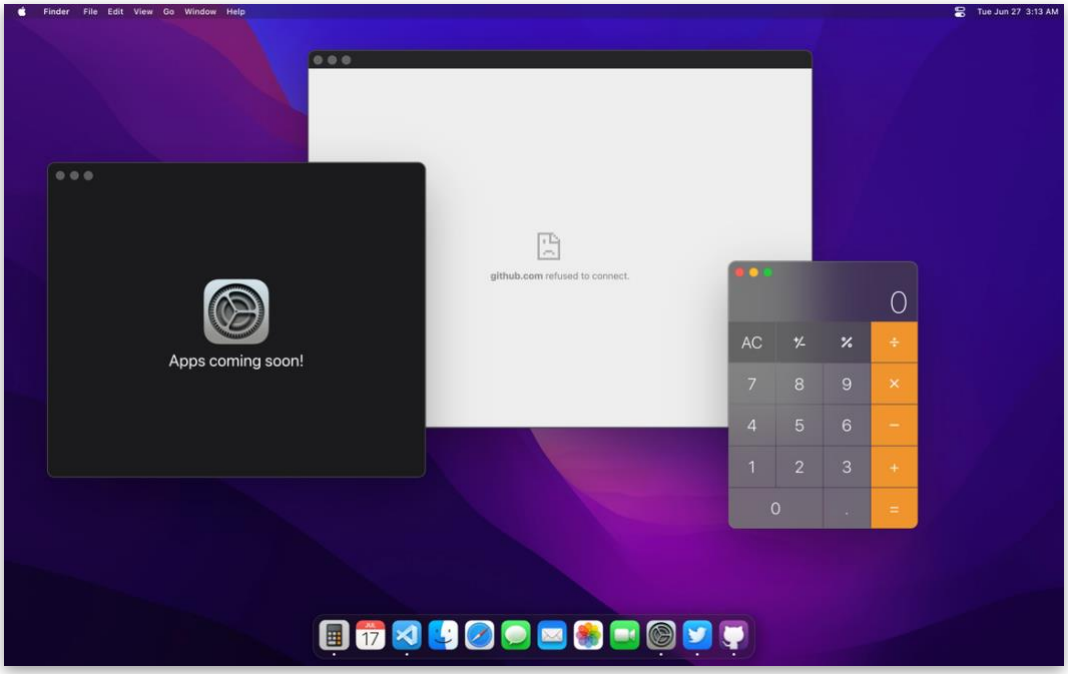

Figura 2: Captura de pantalla del proyecto macos-web

2.1.2 Giant Sur

Gian Sur[15] pretende ser un clon de UI[16] de macOS Big Sur (Figura 3). En ese sentido se le puede dar crédito por usar la iconografía oficial de la plataforma, aunque falla al no aplicar glassmorphism en las interfaces y mezclar iconos de las aplicaciones con versiones anteriores o incluso versiones de iconos de aplicaciones alteradas.

En cuanto a la experiencia y la interacción no podemos encontrar nada, ya que no hay ningún tipo de funcionalidad implementada más allá de la selección de iconos del

[15] https://github.com/JJewell2k/big-fat-liar
[16] UI son las siglas en inglés de “User Interface”, es decir, Interfaz de usuario.

escritorio. Además, la transición de ampliación de los iconos de las aplicaciones tan icónica del Dock de macOS realiza la transición de forma muy abrupta.

Este proyecto pierde interacción en relación con el anterior estudiado, por lo que no podría hacerse pasar por el sistema operativo ya que ni siquiera tiene ventanas de aplicaciones.

Figura 3: Captura de pantalla del proyecto giant-sur

2.1.3 macOS Web Recreation

Por último, se analiza el proyecto macOS Web Recreation[17], que, aunque data de hace más de 7 años, es el proyecto más completo a nivel de funcionalidad de los 3 estudiados. Mientras, la interfaz es demasiado antigua para saber con certeza si era fidedigna (Figura 4), nos ayuda a ver con cierta lente cuánto ha cambiado el diseño de interfaces estos últimos años para sentirse más fluida, simple, ligera y más interactivo.

En cuanto a la experiencia se nota cómo ha intentado adaptar los patrones nativos al web, dando una experiencia satisfactoria en la mayoría de los casos. Mientras los iconos y colores no son los oficiales, por lo que no costaría mucho diferenciar el original de la copia.

Se puede observar que parte de las imágenes no aparecen debido a que, al ser importadas, algunos enlaces se han roto y han dejado de funcionar, lo cual complica el

[17] https://codepen.io/Varo/pen/BypqMJ

análisis para saber si el diseño se acerca la versión de macos que intenta recrear visualmente, la cual es macOS Mojave 10.14.0.

En este caso la versión del sistema operativo no es la misma que la que se quiere desarrollar en este TFG (Ventura 13.14.0), pero se ha decidido incluir en el estudio por la precisión de los detalles y efectos de las interfaces con dicha versión, lo cual forma parte de uno de los objetivos del presente TFG.

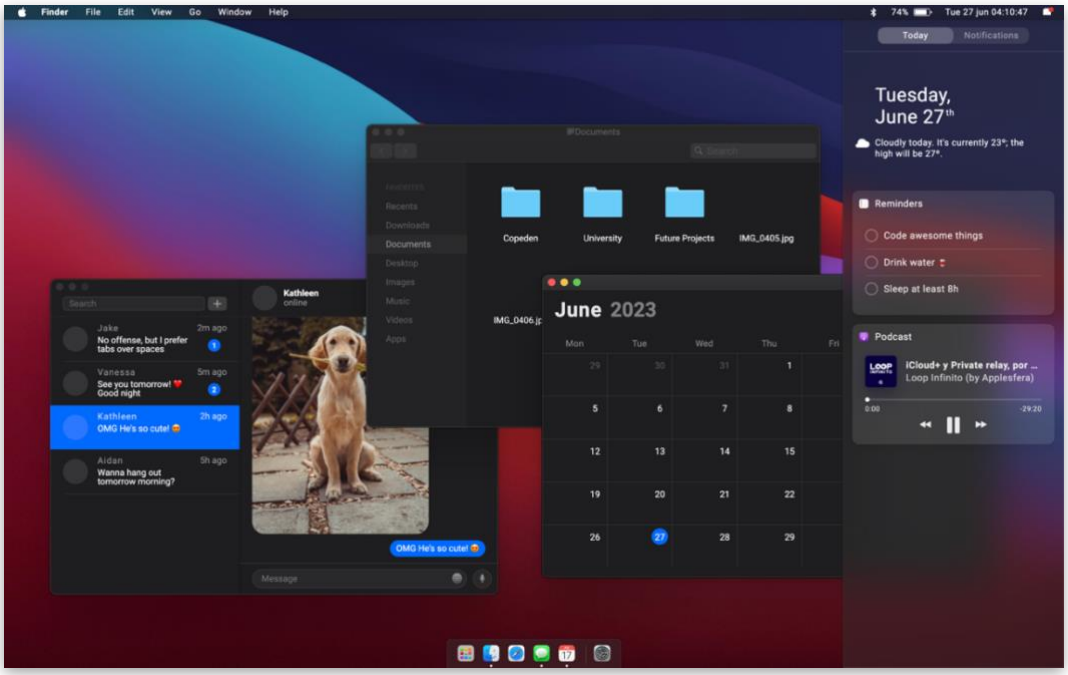

Figura 4: Captura de pantalla del proyecto macos-web-recreation

2.2 Conclusión

La inmensa mayoría de los proyectos que tratan de recrear un sistema operativo en lenguajes web se centran en las interfaces, descuidando los patrones de escritorio, la experiencia de usuario, la interacción con dicha interfaz y la funcionalidad de la implementación. Todos estos proyectos están enfocados en la parte visual, únicamente en la maquetación de ciertas interfaces, incluso evitando implementar cualquier tipo de interacción como las animaciones basadas en el estado del cursor, las cuales son un patrón habitual tanto en las interfaces de escritorio como en las interfaces web.

En conclusión, los proyectos analizados usan patrones e interfaces web, no llegando a el nivel de detalle e implementación de los nativos del sistema operativo. Esto hace que el usuario tenga que crear una relación entre los patrones de escritorio y los patrones web antes de poder interactuar, reduciendo la efectividad de las interfaces y el rendimiento de las tareas desarrolladas en estas.

2.3 Solución propuesta

En el presente TFG se abordará la creación fidedigna más completa posible del sistema operativo macOS, centrándose no solo en las interfaces principales, sino en todos los detalles que componen la experiencia de usuario (interacciones, animaciones, iconos, colores, funcionalidades, etc.). El reto a cumplir es alcanzar un mayor nivel de detalle en la experiencia, creando servicios de sistema, atajos de teclado, aplicaciones funcionales y animaciones e interacciones a la altura de la experiencia de escritorio.

3 Análisis del problema

En este capítulo se analizan las posibles soluciones al problema planteado, examinando las ventajas e inconvenientes de cada posible solución, estableciendo un criterio de selección y eligiendo la idea final a desarrollar.

3.1 Identificación y análisis de soluciones posibles

Tras realizar el análisis de los 3 proyectos presentados en el capítulo 2, se han extraído las funcionalidades que estos proyectos ofrecen y se han recogido en la Tabla 1. Esta tabla presenta las funcionalidades aplicando el método MoSCoW, que refleja la priorización de las funcionalidades, estableciendo su importancia de mayor a menor. Esta clasificación permite determinar la relevancia de las funcionalidades dentro del proyecto.

MUST Debe tener	• Escritorio. • Dock (que muestre aplicaciones para poder abrir). • Barra de Menús. • Spotlight (Búsqueda de aplicaciones). • Control de ventanas. • Ventanas de aplicaciones. • Abrir aplicaciones. • Cerrar aplicaciones. • Cerrar sesión. • Iniciar sesión.
SHOULD Debería tener	• Aplicación de Música. • Aplicación de Notas. • Menús de aplicaciones. • Gestor de aplicaciones abiertas. • Modo oscuro. • Fondo de pantalla dinámico.
COULD Podría tener	• Aplicación de Finder. • Aplicación de Mensajes. • Aplicación de Figma. • Mini app de música en la barra de menús.

	• Acceso a periféricos del dispositivo. • Aplicación de Recordatorios. • Widgets funcionales (Tiempo, recordatorios, etc.). • Minimizar ventana de aplicaciones. • Maximizar ventana de aplicaciones. • Aplicación de Preferencias del sistema. • Cálculos matemáticos básicos en Spotlight. • Definiciones de palabras en Spotlight. • Cambio de idioma del sistema.
WON'T No tendrá	• Previsualización de documentos. • Edición de documentos. • WebDAV o conexiones cloud con la aplicación Finder. • Papelera. • Aplicación de Terminal. • Creación de varios usuarios. • Acceso a internet y páginas web. • Acceso a periféricos del dispositivo. • Sugerencias de búsqueda web en Spotlight. • Conversión de divisas en Spotlight.

Tabla 1: Priorización de tareas mediante método MoSCoW

Como conclusión, se pretende disponer de las funcionalidades que el usuario medio usa en su día a día, creando una experiencia mínima usable e interactiva.

3.2 Solución propuesta

Como se puede observar en la tabla anterior, se han seleccionado como funcionalidades prioritarias aquellas que son estrictamente necesarias para obtener un mínimo producto viable, tanto en interfaz, como en experiencia o funcionalidad.

Una vez seleccionadas las funcionalidades se propone como solución la recreación fidedigna de las interfaces de macOS Ventura, junto con la creación de servicios de música y persistencia en los que se apoyaran las aplicaciones seleccionadas para desarrollar, que son Música, Finder y Notas.

Dividiremos en TFG en tareas para poder medir el progreso de forma más precisa. Una vez se tengan todas las tareas y sub-tareas listas en Jira se empezará a desarrollar

por la tarea más prioritaria (según la tabla de MoSCoW) e implementando antes las tareas que tengan dependencias en otras tareas.

Una vez desarrollada una funcionalidad se realizarán pruebas a nivel de interacción y funcionalidad para asegurar que funciona exactamente como lo haría en el sistema operativo. Si la tarea pasase las pruebas se dará por terminada y se pasará a la siguiente, en caso de que se encuentre algún fallo se documentará en la tarea y se resolverá el error.

Para terminar, cuando las funcionalidades mínimas se hayan finalizado de desarrollar, se subirá la última versión del código a Vercel en forma de página web (se expondrán los pasos sobre cómo hacerlo en el capítulo 6) para poder acceder a la aplicación desde cualquier dispositivo de escritorio.

4 Diseño de la solución

En este capítulo se presenta el diseño de la solución, el cual se compone de una breve presentación de la arquitectura del sistema, seguida por una explicación más detallada sobre las relaciones dentro de la arquitectura.

También se presentarán las herramientas y tecnologías utilizadas para llevar a cabo el presente TFG, incluidas las aplicaciones y lenguajes de programación usadas durante todo el proceso del trabajo.

4.1 Arquitectura del sistema

La arquitectura de este proyecto se conoce como arquitectura de cliente serverless, donde no existe la parte del servidor (a diferencia de la arquitectura cliente-servidor) y la capa del cliente contiene todo el código de la aplicación.

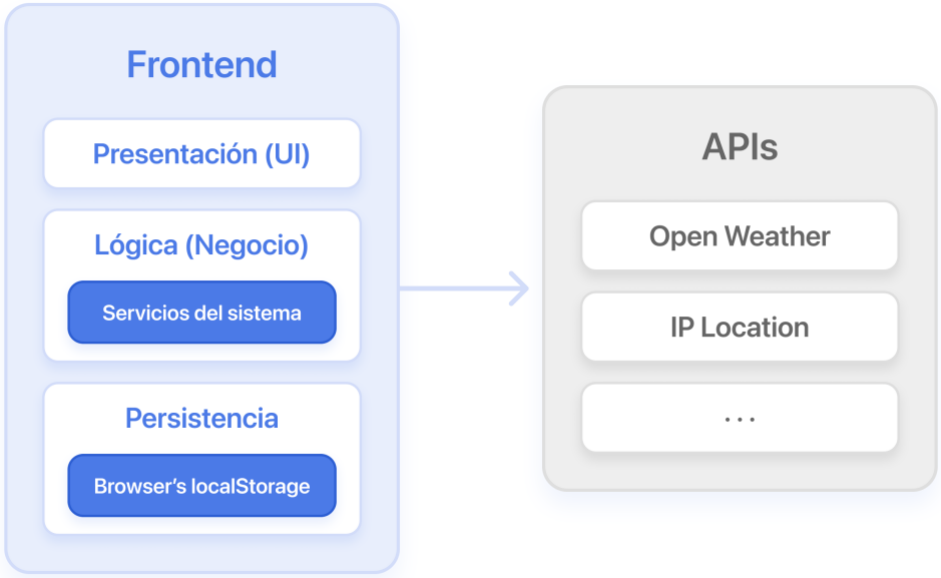

Figura 5: Arquitectura Cliente Serverless.

Como se puede observar en la Figura 5, al aislar todo el código en un mismo repositorio, conseguimos una arquitectura monolítica, donde se agrupan todas las tareas relacionadas con la lógica de negocio, presentación de la interfaz de usuario y comunicación con servicios externos en una misma capa.

En la capa de presentación ocurren todas las interacciones con el usuario, desde la disposición de la información hasta la entrada de datos pasando por los controles interactivos. La interfaz de usuario obtiene datos de la capa de persistencia, mientras los servicios del sistema, de la capa de lógica, le otorgan funcionalidad.

La capa lógica agrupa los servicios del sistema, que más adelante se presentarán en el capítulo 5.1. Estos brindan funcionalidad a la aplicación, se ocupan de toda la parte de tareas y operaciones y se comunican directamente con la capa de persistencia para crear, leer, actualizar y eliminar datos.

Por último, la capa de persistencia es la encargada de mantener los datos accesibles y persistentes. En este caso particular, no usa una base de datos per se, sino que almacena los datos en la caché localStorage [5] del propio navegador. La particularidad de localStorage es que está limitada a 5MB y los datos se mantendrán persistentes hasta que el usuario borre la caché. Aunque es una desventaja es más que suficiente para el uso que se le va a dar y, por otra parte, no requiere la creación y mantenimiento de un servicio dedicado al almacenamiento de datos.

4.1.1 Patrones de diseño

Dado que la base de la aplicación está desarrollada íntegramente en ReactJS es relevante destacar el conjunto de patrones estructurales y creacionales que rigen el comportamiento de este lenguaje de programación y lo hacen una gran opción para cualquier proyecto de gran envergadura:

- **Patrón Composición** [6]**:** Una de las mayores bondades de React es poder crear interfaces compuestas de varios componentes, los cuales pueden estar a su vez compuestos por varios más. Esta capacidad hace que podamos reutilizar el mismo componente en varias vistas (por ejemplo, una barra de navegación) o usar el mismo componente varias veces dentro de una misma vista (por ejemplo, una lista de libros, donde se usa el mismo componente para cada resultado de la lista adaptándolo a la información de cada libro).
- **Patrón Estado** [7]**:** Los estados son uno de los pilares que definen React y este patrón es el actor principal de su comportamiento reactivo, ya que dicha reactividad viene dada por los cambios en los estados internos de la aplicación. Esta característica permite que partes de la interfaz de usuario o la lógica de negocio actúen de manera condicional basada en el valor de un estado y por tanto su comportamiento se podría representar como una máquina de estados.
- **Patrón Observador** [8]**:** Este patrón permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que están observando. En React este es el fundamento de múltiples hooks [9], como useEffect [10].

Además de los patrones fundamentales que rigen el comportamiento de React, también hemos hecho uso de otros patrones para mejorar la mantenibilidad del repositorio y facilitar ciertas implementaciones:

- **Patrón Singleton** [11]**:** Asegura que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia. Este

patrón de diseño creacional permite que la aplicación pueda tener siempre una referencia única sobre variables globales como se detallará más adelante en el capítulo 5.1. Este es el único patrón creacional usado en el proyecto de final de grado.

- **Patrón Puente** [12]**:** Este patrón permite dividir una clase grande, o un grupo de clases estrechamente relacionadas, en dos jerarquías separadas (abstracción e implementación) que pueden desarrollarse de forma independiente la una de la otra. El apartado de servicios del sistema dentro de la aplicación se beneficia enormemente de este patrón como se verá más adelante en el capítulo 5.1.

4.1.2 Modelo de reactividad

Dentro de la capa del cliente, las diferentes sub-capas se relacionan entre ellas, enviando o recibiendo eventos y actualizaciones. A continuación, ilustraremos el modelo de reactividad de ReactJS y expondremos porqué es una buena opción para este proyecto de fin de grado.

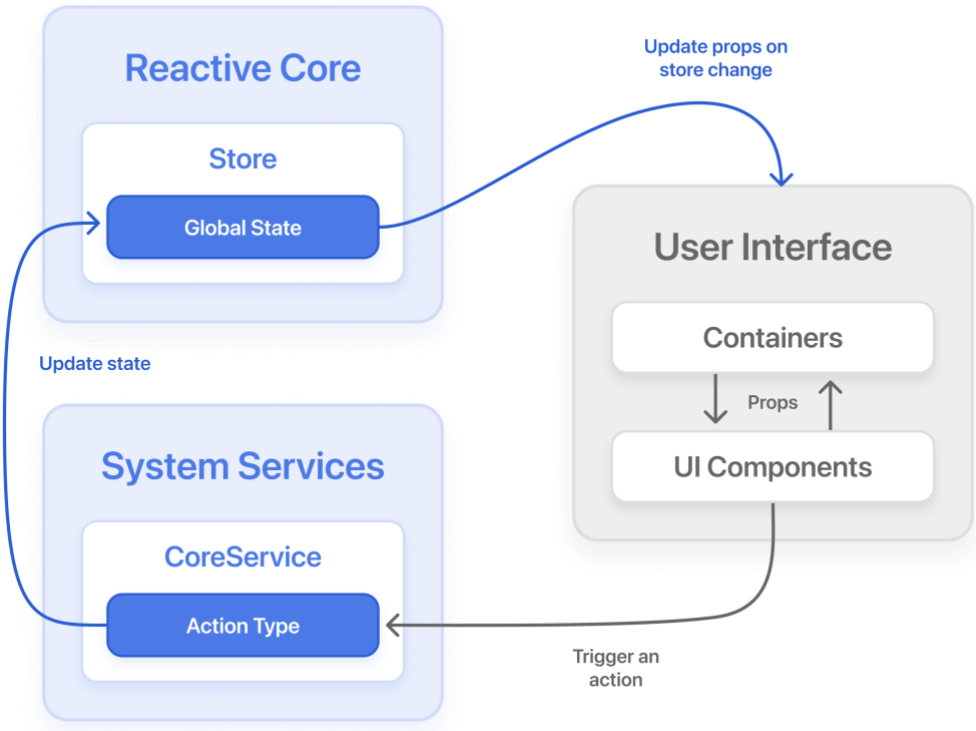

Figura 6: Modelo de reactividad de ReactJS

Como se muestra en la Figura 6, el modelo de reactividad de React permite actualizar la interfaz del usuario mediante cambios de estado. A su vez estos cambios de estado pueden ser activados mediante la interacción con la propia interfaz de usuario.

La reactividad de React brinda múltiples ventajas derivadas de su modelo. A continuación, se exponen las ventajas más relevantes:

- **Eficiencia en rendimiento:** La reactividad en React posibilita una actualización eficiente y optimizada de los componentes. Solo se renderizan aquellos componentes que se ven afectados por cambios en el estado o las props [13], evitando actualizaciones innecesarias en otros componentes y mejorando así el rendimiento general de la aplicación.
- **Mantenimiento simplificado:** La utilización de la reactividad en React permite centralizar la lógica de actualización y renderización dentro de los propios componentes. Esto simplifica la comprensión y el mantenimiento del código, ya que los cambios y actualizaciones en el estado pueden ser rastreados y gestionados de manera clara y ordenada.
- **Programación declarativa:** La reactividad en React se basa en una programación declarativa, lo que implica enfocarse en describir qué acciones debe tomar la interfaz de usuario en función del estado actual. Esto simplifica la lógica de programación y mejora la legibilidad del código.
- **Mayor flexibilidad y reutilización de código:** La reactividad en React permite la reutilización de componentes y lógica a través de props y contextos y, por tanto, componer y combinar componentes de manera flexible, lo que facilita el desarrollo de interfaces de usuario más complejas y la gestión de estados compartidos.
- **Experiencia de usuario fluida:** La reactividad en React permite una experiencia de usuario fluida y dinámica. Los componentes pueden actualizarse de manera inmediata y reactiva a las interacciones del usuario, brindando una respuesta rápida y una sensación de fluidez en la interfaz.

Estas ventajas contribuyen a un desarrollo más eficiente y una mejor experiencia general para los usuarios de la aplicación.

4.2 Tecnologías utilizadas

En esta sección se detallan en profundidad las aplicaciones, herramientas y lenguajes de programación usados en este trabajo de fin de grado, presentando sus cualidades principales que la hacen indispensable para el desarrollo de la solución propuesta, la curva de aprendizaje de estas y los conocimientos personales.

4.2.1 Figma

Figma [14] es una herramienta de diseño y prototipado profesional que permite a los usuarios crear y colaborar en proyectos de diseño de manera eficiente. Es una aplicación basada en la nube, lo que significa que no requiere instalación y se puede acceder desde cualquier dispositivo con conexión a Internet.

Con Figma, se pueden crear diseños de interfaces de usuario, tanto estáticos como interactivos, de manera intuitiva y fácil de usar. Ofrece una amplia gama de herramientas y funciones para la creación de elementos gráficos, como formas, iconos, imágenes y texto. Además, permite la importación de recursos y la integración con otras herramientas de diseño.

Esta herramienta se ha usado para replicar algunas de las interfaces nativas visualmente, antes de programarlas para medir dimensiones, pesos tipográficos, tamaño de fuente, color, etc. y así crear una réplica precisa de la interfaz del sistema operativo de manera más ágil. Una vez se han obtenido dichos diseños estos se han programado usando TailwindCSS siguiendo las medidas especificadas.

En cuanto a la curva de aprendizaje, esta aplicación es adecuada tanto para principiantes como para profesionales, ya que su interfaz es extremadamente simple y su funcionamiento bastante intuitivo. Los conocimientos personales sobre esta herramienta son de nivel experto, ya que se usa en el trabajo como jefe de diseño y para proyectos personales.

4.2.2 Visual Studio Code

Visual Studio Code [15] es un potente editor de código fuente desarrollado por Microsoft. Es una herramienta extremadamente popular entre los programadores debido a su flexibilidad, funcionalidad y facilidad de uso. Aunque se puede utilizar para una amplia variedad de lenguajes de programación, está especialmente diseñado para trabajar con tecnologías web, como HTML, CSS y JavaScript.

Una de las características destacadas de Visual Studio Code es su extensibilidad. Los usuarios pueden personalizar y ampliar el editor instalando una amplia variedad de extensiones, que proporcionan funcionalidades adicionales y soporte para diferentes lenguajes y marcos de trabajo. Esto permite adaptar la experiencia de desarrollo a las necesidades específicas de cada programador.

Visual Studio Code también ofrece características prácticas como el resaltado de sintaxis, autocompletado inteligente, depuración integrada y control de versiones. Su interfaz intuitiva y su diseño minimalista facilitan la navegación y la organización de proyectos complejos.

Esta herramienta ofrece una variedad inmensa de opciones y llegar a manejarla con soltura teniendo conocimiento total sobre esta es complejo. En cambio, gracias a su interfaz simple es una aplicación apta para principiantes y muy potente para profesionales, llegando a ser el editor de código más usado del mercado. El conocimiento personal de esta herramienta es profesional, ya que se usa en el trabajo y para proyectos de la universidad y personales.

4.2.3 GitHub

Github [16] es una plataforma de desarrollo de software basada en Git que permite a los desarrolladores colaborar en proyectos, gestionar versiones y compartir su trabajo de forma eficiente. Su enfoque en la comunidad, las herramientas de gestión de proyectos y las características de seguridad lo convierten en una herramienta esencial para el desarrollo de software en equipo.

En concreto esta herramienta se ha utilizado para la gestión de versiones del código, ayudando a mantener las versiones organizadas y transparencia en las modificaciones del código. Github es una plataforma fácil de usar, con una interfaz intuitiva y una variedad de herramientas accesibles para principiantes y muy completas para programadores en un nivel avanzado.

4.2.4 GitHub Desktop

GitHub Desktop [17] es una aplicación de escritorio que proporciona una interfaz amigable para interactuar con los repositorios de GitHub. Permite clonar, crear y administrar repositorios, así como colaborar de manera efectiva en proyectos de desarrollo de software. Su enfoque intuitivo y su integración con GitHub la convierten en una herramienta valiosa para los desarrolladores que desean gestionar sus proyectos de forma eficiente.

Esta herramienta ayuda con la parte de la gestión de las versiones y repositorios, ya que sustituye a la línea de comandos por una interfaz clara y simple. Su curva de aprendizaje es extremadamente baja, casi no se requieren conocimientos para empezar a usarla.

4.2.5 SF Symbols

SF Symbols [18] es una biblioteca de iconos vectoriales diseñada por Apple que está disponible para desarrolladores en sus plataformas. Ofrece una amplia gama de iconos escalables y accesibles que se adaptan a la estética y estilo de Apple. Con SF Symbols, los desarrolladores pueden crear interfaces de usuario coherentes y visualmente atractivas en sus aplicaciones y dispositivos.

Se usa esta aplicación para obtener los iconos del paquete de iconos vectoriales de Apple para mayor realismo y nivel de detalle de las interfaces.

4.2.6 TypeScript

TypeScript [19] es un lenguaje de programación que mejora JavaScript al agregar tipado estático y características avanzadas. Proporciona un sistema de tipos sólido que ayuda a prevenir errores y mejora la productividad en el desarrollo de software. Con su capacidad de transpilación a JavaScript estándar y su amplia comunidad de desarrolladores, TypeScript se ha convertido en una opción popular para proyectos de gran escala y aplicaciones web y móviles.

La curva de aprendizaje de un lenguaje como TypeScript es normal, baja al empezar gracias al tipado y media conforme se van adquiriendo más conocimientos del lenguaje y las posibilidades que brinda sobre JavaScript [20] aumentan.

4.2.7 TailwindCSS

TailwindCSS [21] es un framework de diseño de interfaz de usuario que se basa en una metodología utilitaria. Proporciona una amplia gama de clases CSS reutilizables que permiten a los desarrolladores construir sitios web y aplicaciones web personalizadas y eficientes. Con su enfoque en el rendimiento y la flexibilidad, TailwindCSS se ha convertido en una opción popular para proyectos de diseño web modernos y adaptables.

La metodología utilitaria de TailwindCSS se basa en la creación de clases CSS pequeñas y reutilizables que se aplican directamente a los elementos HTML. Estas clases representan utilidades específicas, como márgenes, espaciado, tamaños de texto, colores y diseños. Al combinar y reutilizar estas clases, los desarrolladores pueden crear rápidamente diseños y estilos personalizados sin tener que escribir CSS adicional.

Una de las ventajas de TailwindCSS es su enfoque en el rendimiento. Utiliza una metodología de diseño modular que permite optimizar el tamaño de los archivos CSS generados, lo que se traduce en una carga más rápida de los sitios web y una mejor experiencia para los usuarios.

Se ha elegido TailwindCSS por su flexibilidad y sencillez de uso y lectura comparado con CSS tradicional, evitando así tener una hoja de estilos a parte y con la ventaja de poder leer el diseño directamente en la implementación del componente.

Este framework de CSS tiene una curva de aprendizaje medianamente elevada, dado que ya proporciona los nombres de las utilidades, el usuario tiene que aprenderlos de memoria para poder coger velocidad programando interfaces. Una vez se tiene cierta soltura con el lenguaje, resulta mucho más rápido y potente que el CSS tradicional.

4.2.8 Radix UI

Radix UI es una biblioteca de componentes de interfaz de usuario (UI) desarrollada para la creación de aplicaciones web modernas y accesibles. Proporciona un conjunto de componentes reutilizables y personalizables que se adhieren a las mejores prácticas de diseño y están optimizados para la accesibilidad.

La biblioteca Radix UI se enfoca en la construcción de componentes fundamentales y flexibles, como botones, menús desplegables, modales, pestañas y mucho más. Estos componentes se pueden combinar y personalizar fácilmente para adaptarse a las necesidades de diseño y funcionalidad de cada proyecto.

Una de las características distintivas de Radix UI es su enfoque en la accesibilidad. Todos los componentes están diseñados con un enfoque en cumplir con los estándares de accesibilidad web y se esfuerzan por proporcionar una experiencia de usuario inclusiva para personas con discapacidades.

Además, Radix UI permite una fácil integración con frameworks populares como ReactJS, NextJS y TailwindCSS, lo que facilita su uso en proyectos existentes. La biblioteca también proporciona una documentación completa y ejemplos de código, lo que ayuda a los desarrolladores a aprender y utilizar los componentes de manera efectiva.

4.2.9 ReactJS

ReactJS [22] es una biblioteca de JavaScript de código abierto desarrollada por Facebook, Meta que se utiliza para construir interfaces de usuario interactivas y reactivas [23]. Se enfoca en la creación de componentes reutilizables que permiten el desarrollo eficiente y modular de aplicaciones web.

La principal característica de React JS es su enfoque en la construcción de interfaces de usuario declarativas. Con React, los desarrolladores pueden crear componentes individuales que representan partes específicas de la interfaz y luego combinarlos para formar la aplicación completa. Estos componentes son altamente reutilizables y actualizan automáticamente la interfaz en función de los cambios en los datos, lo que facilita la creación de aplicaciones interactivas y de respuesta rápida.

React JS utiliza una técnica llamada "Virtual DOM" [24] (DOM virtual) para lograr un rendimiento óptimo. En lugar de realizar cambios directamente en el DOM del navegador, React genera una representación virtual del mismo y realiza comparaciones eficientes para determinar los cambios que deben aplicarse. Esto minimiza las operaciones costosas en el DOM real y mejora el rendimiento general de la aplicación.

Este framework es usado por grandes empresas como Facebook, Netflix, Dropbox, Microsoft o Slack, lo que verifica que es una muy buena opción para desarrollar proyectos de gran envergadura. La curva de aprendizaje es mediana, hay numerosas diferencias con JavaScript y el cambio de paradigma a declarativo, aunque una vez se

entiende el paradigma, crear componentes con React resulta increíblemente más rápido que con JavaScript puro.

4.2.10 NextJS

Next.js [25] es un framework de desarrollo web de código abierto y basado en React. Está diseñado para crear aplicaciones web modernas y rápidas, brindando una experiencia de desarrollo eficiente y un rendimiento óptimo. Next.js combina la potencia de React con características adicionales que facilitan la creación de aplicaciones escalables y renderizadas tanto en el servidor como en el cliente.

Una de las principales características de Next.js es el renderizado del lado del servidor (SSR) y la representación en el lado del cliente (CSR) híbrida. Esto significa que se puede generar el contenido HTML en el servidor para una carga inicial rápida y una mejor optimización para los motores de búsqueda, y luego continuar la renderización y la interactividad en el lado del cliente. Esta combinación permite una experiencia de usuario fluida y velocidad de carga de página alta.

Además, Next.js ofrece enrutamiento dinámico automático, lo que significa que las rutas se generan de forma automática en función de la estructura del proyecto, lo que ahorra tiempo y esfuerzo en la configuración manual del enrutamiento. También proporciona pre-renderización estática, donde las páginas pueden generarse por adelantado en tiempo de compilación y servirse como archivos HTML estáticos, lo que mejora aún más el rendimiento y la velocidad de carga.

Next.js tiene una integración sencilla con APIs y servicios externos, lo que permite obtener y procesar datos de forma eficiente. También es compatible con el desarrollo de aplicaciones web progresivas (PWA) y proporciona funcionalidades como la gestión del estado global y la optimización automática del código para un mejor rendimiento.

Este framework ha sido elegido por todas sus bondades y rapidez para programar interfaces modernas, además de funcionar muy bien junto con TailwindCSS. En concreto es muy útil en tareas repetitivas como el enrutamiento. La curva de aprendizaje es baja, una vez se sabe React solo se requiere aprender las funcionalidades necesarias que permiten optimizar nuestro desarrollo.

4.2.11 Vercel

Vercel [26] es una plataforma en la nube que permite el despliegue y alojamiento de aplicaciones web de forma rápida y sencilla. Está diseñada especialmente para proyectos construidos con tecnologías modernas, como React, Next.js y Vue.js. Vercel simplifica el proceso de implementación y ofrece una infraestructura global que garantiza un rendimiento rápido y confiable de las aplicaciones.

Una de las características destacadas de Vercel es su enfoque en la implementación sin configuración. Proporciona una experiencia de implementación sencilla, donde los desarrolladores pueden conectar directamente sus repositorios de código y Vercel se

encarga automáticamente de compilar y desplegar la aplicación. Esto ahorra tiempo y esfuerzo en configuraciones complejas y permite un flujo de trabajo ágil.

Vercel también ofrece una funcionalidad de pre-renderización estática y generación de sitios estáticos, lo que permite la creación de sitios web altamente optimizados y de carga rápida. Además, cuenta con integraciones nativas para la gestión de dominios, certificados SSL y redirecciones, lo que facilita la configuración de la infraestructura necesaria para el despliegue de aplicaciones.

Una ventaja adicional de Vercel es su capacidad de escalar automáticamente las aplicaciones en función de la demanda. La plataforma detecta automáticamente los picos de tráfico y ajusta los recursos de manera dinámica para garantizar un rendimiento óptimo en todo momento. Esto permite que las aplicaciones se mantengan rápidas y estables incluso en momentos de alta concurrencia.

Esta plataforma PaaS [27] ofrece lo necesario, de forma gratuita, para alojar la aplicación desarrollada en este TFG. Además, al ser los creadores de NextJS, su plataforma está optimizada para tomar ventaja de todas las capacidades del framework NextJS y así asegurar una experiencia rápida y estable. Su curva de aprendizaje es muy baja, su interfaz es muy sencilla y todos los flujos están muy bien guiados, por lo que cualquier usuario sin apenas conocimientos de programación es capaz de usarla.

4.2.12 Jira

Jira [28] es una herramienta de gestión de proyectos y seguimiento de problemas ampliamente utilizada en el desarrollo de software y otros campos de trabajo. Proporciona un entorno centralizado para la planificación, seguimiento y colaboración en proyectos, lo que ayuda a los equipos a mantenerse organizados y enfocados en sus tareas.

Con Jira, los usuarios pueden crear y asignar tareas, establecer fechas límite, realizar un seguimiento del progreso y gestionar las dependencias entre las diferentes actividades. Además, ofrece herramientas de gestión de problemas (issues) que permiten a los equipos registrar, priorizar y solucionar problemas o incidencias que surjan durante el desarrollo del proyecto.

En el caso de este proyecto, esta herramienta cumple con creces con las necesidades expuestas en el apartado 1.3, facilitando la gestión de las tareas en el tablero Kanban. La curva de aprendizaje de esta herramienta es muy baja, aunque también ofrece herramientas más avanzadas según las necesidades del proyecto.

4.2.13 Microsoft Word

Microsoft Word [29] es un procesador de textos potente y versátil que proporciona herramientas completas para la creación, edición y formato de documentos de texto. Con su amplia gama de funciones de revisión y corrección de textos, opciones de personalización de formato, y capacidad de colaborar en tiempo real, Microsoft Word es

una herramienta confiable para la creación de documentos profesionales en diversos entornos, como el trabajo, la educación y el hogar.

Esta conocida herramienta ha sido utilizada para la redacción de la memoria del TFG, su curva de aprendizaje es media, para realizar documentos simples no se requieren conocimientos previos, sin embargo, conforme los requisitos del documento crecen, también lo hace la complejidad de los menús y opciones.

5 Desarrollo de la solución propuesta

En este capítulo se detalla la implementación final del trabajo, haciendo hincapié en aquellas partes del código con mayor importancia y complejidad. Se explicarán varias partes de las interfaces, funcionalidades y aplicaciones creadas.

5.1 Servicios del sistema

Los servicios del sistema son programas o procesos en un sistema operativo que brindan funcionalidades específicas a otros programas. Estos servicios facilitan tareas como la gestión de archivos, la comunicación en red y el control de hardware. Los servicios del sistema son componentes esenciales para el funcionamiento de un sistema operativo y permiten que las aplicaciones interactúen con el hardware y otros recursos del sistema de manera eficiente.

En este caso, dado que el TFG se desarrolla en el dominio web, se entenderá servicio de sistema como todo módulo o clase en la que las aplicaciones o interfaces se apoyan para desarrollar una funcionalidad o interactuar con recursos del sistema. A diferencia de las funciones que pueden haber dentro de los componentes, estos módulos proporcionan métodos abstractos y constructores reutilizables.

El tipado y documentación de estos servicios permiten saber en todo momento qué tipo de dato va a devolver y el tipo de los parámetros de entrada, lo que agiliza el proceso de desarrollo y ayuda a minimizar los errores en tiempo de ejecución, ya que se pueden detectar de forma estática en el código. Al ser una estructura mono repositorio, estos tipos son compartidos a lo largo de todo el proyecto, por lo que es muy sencillo de implementar y no requiere más pasos extra que definir los tipos de las entradas y salidas de cada método cuando se implementan.

Se exponen a continuación los servicios del sistema elegidos para implementar para el presente proyecto de fin de grado.

5.1.1 AppLibrary

Como en todo sistema operativo, las aplicaciones deben compartir una estructura estándar que las defina, por ello se crea AppLibrary, una librería para definir de forma centralizada el modelo de las aplicaciones y sus metadatos y funciones. Además, esta librería incluye varios métodos que permiten consultas sobre la colección de aplicaciones definidas en el sistema como se detalla a continuación.

```
import * as apps from '../store/apps';

/**
 * Get the AppID of every app
 * @returns {string[]} Returns an array with all the AppIDs
 */
const appIDs: string[] = Object.getOwnPropertyNames(apps)
  .map((key) => [key, Object.getOwnPropertyDescriptor(apps, key)])
  .filter(([key, descriptor]) => typeof descriptor['get'] === 'function')
  .map(([key]) => key.toString());
```

Figura 7: Fragmento de código de AppLibrary, detalle de función agrupa todas las IDs de las aplicaciones del sistema

En la Figura 7 se define la función que recopila todos los identificadores de las aplicaciones declaradas en el sistema. Se denomina identificador a la cadena de texto que permite identificar de manera inequívoca una aplicación.

La función recoge todas las exportaciones de la colección de aplicaciones del sistema, así conseguimos una lista de los modelos de las aplicaciones. A continuación, se mapea el objeto filtrándolo por el método getter del modelo, en otras palabras, se obtiene el conjunto de valores posibles dentro del objeto que devuelven una función, en este caso la aplicación. Una vez se ha obtenido esta colección filtrada para obtener las aplicaciones, se mapea el objeto de nuevo para obtener de cada aplicación su ID.

Con esta función resulta más sencillo y eficiente consultar o filtrar aplicaciones en el resto de los métodos de la librería. Vista esta función se pueden definir el resto de los métodos de AppLibrary en la tabla siguiente.

get(…) **Parámetros:** AppID: string	A partir de la ID de una aplicación, devuelve el modelo de la aplicación cuyo ID sea equivalente al del parámetro de la función. Este método es especialmente útil para no tener que importar siempre en todos los documentos toda la biblioteca de aplicaciones y, lo más importante, poder obtener aplicaciones de forma dinámica, sin necesitar importarla antes explícitamente en el documento que la consulte.
list()	Devuelve un array de todas las IDs de las aplicaciones en forma de cadena de texto, hace uso directo de la función descrita en la Figura 7.
appKeywords()	Devuelve una lista de duplas que contienen, la ID de la aplicación y el conjunto de palabras clave de la aplicación. Este método es especialmente útil para búsquedas de aplicaciones

	por su uso o nombre, es por esto por lo que se utiliza en la función descrita a continuación.
search(…) **Parámetros:** keyword: string, exclude?: string[]	En los parámetros de esta función se incluye una “keyword” (palabra clave), la cual es el término para buscar, normalmente suele ser una cadena de texto. También tiene un parámetro opcional llamado “exclude”, que es una lista de IDs de aplicaciones a evitar en el resultado. Este último parámetro es especialmente útil en situaciones como el campo de búsqueda de aplicaciones dentro del Launchpad, donde queremos evitar que la propia aplicación de Launchpad aparezca como resultado. Dada una cadena de texto esta función busca coincidencias que empiecen, incluyan o terminen en con la “keyword”. En el caso de que la “keyword” tenga espacios, cada palabra separada por espacios se usará en la búsqueda de forma independiente. Para el algoritmo de la búsqueda se hace uso de la función **appKeywords** que devuelve una lista con todas las palabras clave de todas las aplicaciones, a modo de diccionario. Este diccionario se filtra en búsqueda de coincidencias con alguna de las sub-cadenas de texto como se muestra en la Figura 8 y devuelve una lista de las aplicaciones que contienen alguna de estas palabras clave.

Tabla 2: Métodos de AppLibrary

```
/**
   * Return AppIDs of the apps which name or keywords starts with the keyword
   * @param {string} keyword - Keyword, term or word you want to search
   * @param {string[]} exclude - Exclude, array of IDs to exclude from the search
   * @returns {string[]} Array with all AppIDs that match with the keyword search
   */
  search(keyword: string, exclude?: string[]): string[] {
    return appKeywords
      .filter((app) => (exclude ? !exclude.includes(app[0]) : app))
      .filter(
        (app) =>
          app[1].filter((string) => string.startsWith(keyword)).length > 0
      )
      .map((results) => results[0]);
  }
```

Figura 8: Fragmento de código de la función search() de AppLibrary

Más tarde, en el apartado 5.2.1, 5.2.4 y 5.2.5, se hará uso de este método para brindar funcionalidad a las interfaces del sistema.

5.1.2 CoreAudio

CoreAudio se encarga de toda la gestión del contenido de audio del sistema y su reproducción. En concreto se hace uso del patrón Singleton [11] para crear una única instancia del componente del reproductor, al que se puede acceder globalmente con los métodos de este módulo, los cuales se definen en la siguiente tabla.

play()	Como el propio nombre de la función indica, inicia la reproducción de la pista de audio seleccionada en ese instante.
pause()	Al contrario que play(), esta función pausa la reproducción de la pista de audio. En el caso de que no se esté reproduciendo nada, la función no tendrá efectos.
next()	Dada una lista de reproducción predefinida next() permite pasar a la siguiente pista de audio de la lista. Si la pista de audio actual era la última por defecto se volverá a la primera pista de audio de la lista al ejecutar la función. Esta función se ejecuta automáticamente al finalizar cada pista de audio, consiguiendo así la experiencia de un reproductor estándar que permite reproducción continua de contenido.
prev()	Al contrario que next(), esta función pasa a la pista de audio anterior y en el caso de que la pista actual fuera la primera, salta a la última pista de audio, siguiendo en mismo comportamiento que un buffer circular.

Tabla 3: Métodos de CoreAudio

Los métodos definidos en esta librería permiten abstraer el acceso y la mutación del componente de audio de HTML, permitiendo acceder al servicio del sistema de reproducción de audio de manera funcional.

5.1.3 CoreData

Para acceder a la capa de persistencia se necesita un módulo o librería capaz de realizar operaciones CRUD una base de datos, la cual en el caso de este proyecto es la caché del propio navegador.

set(…) **Parámetros:** keyName: string, keyValue: any	Este método proporciona una abstracción de localStorage [5] para poder escribir en caché. Los parámetros de la función son keyName, el cual es la clave por la cual queremos identificar al dato, y keyValue, que es el dato en sí. La caché permite cualquier estructura de datos dentro de keyValue, por lo que se pueden guardar desde cadenas de texto hasta imágenes en formato Base64 [30], con la condición de que la totalidad de la caché no supere los 5MB. En caso de que se intente insertar un ítem y la caché esté llena saltará un error.
get(…) **Parámetros:** keyName: string	Este método permite acceder a un dato almacenado mediante una clave que se debe añadir como parámetro. En el caso de que la clave no exista o no haya ningún dato asociado a dicha clave el método devolverá “null”.
delete(…) **Parámetros:** keyName: string	El método delete() permite eliminar de la caché la entrada asociada a la clave proporcionada por el parámetro keyName. En el caso de que no se proporcione ninguna clave el método no tendrá ningún efecto sobre la caché.
clear()	En caso de que se desee limpiar todos los datos de la cache, este método permite vaciar la caché por completo, eliminando todos los pares clave-valor de la caché.

Tabla 4: Métodos de CoreData

CoreData es fundamental para aplicaciones como Notas o Recordatorios, donde se debe almacenar y modificar información.

5.1.4 NotificationKit

NotificationKit es un componente de software que permite enviar notificaciones y mensajes a los usuarios para informarles sobre eventos o cambios importantes en una aplicación o estado del sistema operativo.

A medida que las interacciones del usuario con la interfaz aumentan también lo hacen los métodos de entrada y los patrones de comunicación para la presentación de información al usuario. Tener una variedad de patrones de comunicación como las notificaciones, alertas, diálogos o los modales ayudan a crear una experiencia más completa y enfocada en el usuario.

En este caso, empezaremos con las notificaciones y para ello se ha creado una librería específica con los siguientes métodos.

notify(…) **Parámetros:** app: string, title: string, message: string, action?: function	Crea una notificación que aparecerá arriba a la derecha de la pantalla y en el centro de notificaciones. La notificación usa por debajo la librería react-hot-toast [31], ya que nos permite una gran personalización del diseño y una implementación limpia, sencilla y eficiente. Para hacer su uso más sencillo se ha creado una plantilla con el diseño de una notificación, la cual se completa con los valores introducidos en los parámetros. El parámetro “action” es opcional y permite ejecutar una acción al hacer click en la notificación. El método devuelve la ID de la notificación, que será útil en el siguiente método para poder identificarla inequívocamente.
dismiss(…) **Parámetros:** id?: string	Este método permite hacer desaparecer una notificación si se incluye el parámetro “id” y si no se incluye ningún parámetro se eliminan todas las notificaciones activas y del cetro de notificaciones.

Tabla 5: Métodos del Sistema de notificaciones

5.1.5 WindowManager y AppSwitcher

Un gestor de ventanas (WindowManager) es un componente del sistema operativo o un programa independiente que controla la apariencia y disposición de las ventanas en una interfaz gráfica de usuario. Su función principal es permitir la manipulación y organización de las ventanas, como su tamaño, posición, superposición y cierre.

Este componente se instancia en el escritorio con tal de que tenga acceso a todas las ventanas de aplicaciones para su gestión. Además de proporcionar los métodos necesarios para gestionar las ventanas de una forma eficiente, añade una interfaz de forma implícita a la que se denomina dentro del dominio como AppSwitcher.

Figura 9: Escritorio con AppSwitcher activo mostrando las aplicaciones abiertas entre las cuales se puede mover el foco

AppSwitcher es el atajo para moverse entre ventanas que aparece cuando pulsamos en el teclado las teclas “Option” + Tabulador (macOS) o “Alt” + Tabulador (Windows y Linux). Esta interfaz permite visualizar las aplicaciones abiertas y mover el foco entre ellas. Cuando el usuario presiona este comando la interfaz aparece permitiendo moverse por la lista usando la tecla de tabulador y cuando el usuario deja de pulsar las teclas se le da foco a la aplicación seleccionada.

El gestor ve ventanas incluye varios métodos para que el usuario pueda abrir, cerrar, mover, redimensionar y ocultar las ventanas del escritorio. También tiene varios estados internos que le permiten saber qué aplicaciones están abiertas y qué aplicación tiene el foco. A continuación, se presentan estos métodos.

openApp(…) **Parámetros:** AppID: string	Cuando el usuario quiere abrir una aplicación y hace click en el icono de la app en el Dock o Launchpad se ejecuta este método pasando como parámetro la ID de la aplicación que se desea abrir. Esta función se beneficia del estado interno para saber qué aplicaciones están abiertas y dependiendo de si el ID de la aplicación está en esta lista o no se ejecutará una función o la otra. En el caso de que la aplicación no estaba abierta, se abre y muestra en pantalla, en el caso contrario se le da foco si no lo tenía, para ello se usará el método siguiente focusApp().

	Para que este método sufra efecto de forma global actualiza el estado interno en el que se almacena las aplicaciones abiertas (en este caso un array) y lo añade en la cola, que indica que es la última aplicación con la que se ha hecho interacción y por tanto la que tiene el foco. Este método se ocupa de actualizar el estado global que define si cierta aplicación está abierta, del cual se benefician otros componentes como el Dock o Spotlight, que indican si una aplicación está abierta con una marca debajo de su icono como se puede apreciar en la Figura 9.
focusApp(…) **Parámetros:** AppID: string	Este método enfoca la ventana de la aplicación cuyo ID se pasa por parámetro a la función. Si la aplicación está abierta pero la ventana está oculta, focusApp() vuelve a mostrar la ventana. Si la aplicación ya tiene foco no realiza ningún cambio. Para que la ventana obtenga el foco se mueve la aplicación al final del array de la variable interna que controla qué aplicaciones están abiertas. Además, esta función se encarga de la gestión de qué contenido mostrar en la barra de menús de aplicaciones, actualizando su contenido cada vez que el foco de la ventana cambia.
quitApp(…) **Parámetros:** AppID: string	Cuando el usuario selecciona la opción “Quit” y el nombre de la app, la app llama a este método que se encarga de cerrar la aplicación y eliminar la ventana del escritorio. De la misma manera que en los anteriores métodos, para eliminar la ventana del escritorio se actualiza la variable interna que almacena las aplicaciones abiertas y elimina del array, obteniendo el foco entonces la última aplicación con la que el usuario había interactuado previamente.

Tabla 6: Métodos del Gestor de ventanas

5.2 Interfaces de usuario

Una interfaz de usuario es el medio a través del cual un usuario interactúa con un sistema o una aplicación. Proporciona los elementos visuales y funcionales necesarios para que el usuario pueda comunicarse con la tecnología de manera efectiva. Una interfaz de usuario puede incluir elementos como botones, menús, formularios, iconos y ventanas, que permiten al usuario realizar acciones y recibir retroalimentación del sistema.

El objetivo de una interfaz de usuario es ser intuitiva, fácil de usar y proporcionar una experiencia agradable al usuario. Una buena interfaz de usuario mejora la usabilidad y la accesibilidad de un sistema, lo que a su vez facilita la interacción y el logro de tareas por parte del usuario.

Descritas las interfaces se muestran ahora varios ejemplos recreados en React, incluyendo algunas capturas de la vista de escritorio de la aplicación web (no confundir con el escritorio real del sistema operativo). Todos los componentes de la interfaz han sido diseñados o personalizados usando las clases de utilidades [32] de TailwindCSS.

```
<div
  onMouseMoveCapture={dockMagnification}
  onMouseLeave={dockMagnificationStop}
  className=
"group/dock absolute left-1/2 bottom-[5px] min-w-fit -translate-x-1/2 rounded-xl bg-gray-6/30 p-1 pb-1.5 before:absolute
before:-inset-px before:-z-1 before:rounded-xl before:border before:border-white/15 before:shadow-dock before:ring-0.5 be
fore:ring-black/25 before:backdrop-blur-3xl before:backdrop-brightness-105 before:backdrop-saturate-[1.1] after:absolute
after:-inset-x-1 after:top-full after:-z-1 after:h-5 dark:bg-gray-6/25 dark:ring-black/60 dark:before:backdrop-brightness
-90"

>
  {/* APP ICONS */}
</div>
```

Figura 10: Fragmento de código del componente Dock, mostrando cómo se usan las clases de utilidades de TailwindCSS

Como se puede ver en la Figura 10, mediante la sintaxis de TailwindCSS se consigue describir mediante clases de utilidades de CSS parte del estilo del Dock, uno de los componentes más complejos de las interfaces implementadas. Gracias a esto se evita tener que crear dos documentos, uno para la maquetación en JSX y otro para la definición de las clases asociadas a dicha maquetación, lo que en algunos casos puede generar gran confusión por la jerarquía de nombres, hacer más costoso su mantenimiento y generar código repetido.

TailwindCSS permite, mediante una colección de clases predefinidas, crear interfaces complejas y autodefinidas en un mismo componente, lo que ayuda enormemente a la reutilización de los componentes y permite, de forma sencilla, crear componentes con estilos variables dependiendo de los estados del componente.

Este framework también permite la creación y modificación de nuevas clases de utilidad en su archivo de configuración, lo que ayuda a poder seguir un sistema de

diseño[18] consistente a lo largo de todo el desarrollo. Dicho archivo de configuración se introduce en un JSON[19], lo cual hace más sencillo exportar e importar configuraciones entre proyectos o aplicaciones y, por tanto, mantener el mismo sistema de diseño incluso entre diferentes desarrollos.

5.2.1 Dock

El Dock proporciona acceso rápido y conveniente a aplicaciones, Launchpad y otras funciones del sistema. Se encuentra ubicado en la parte inferior de la pantalla y se compone de iconos de aplicaciones que se animan en un efecto de ampliación cuando el usuario mueve el cursor por encima de estos.

Al hacer clic en los iconos de las aplicaciones estas se abren utilizando el método openApp() de la librería WindowManager explicada anteriormente. El Dock proporciona acceso rápido a las aplicaciones que se están abiertas actualmente, mostrando un indicador visual debajo de sus respectivos iconos.

Figura 11: Captura del escritorio de la aplicación web, detalle del Dock mostrando el efecto de ampliación característico de macOS

El diseño del Dock es simple, con un fondo que imita el efecto de glassmorphism tan característico de las interfaces de Apple. Es relevante entender que los efectos que ocurren en una página web se ejecutan mediante CPU, al contrario que en el dispositivo donde estas animaciones se vena aceleradas por GPU y por tanto de forma más eficiente. Además, el lenguaje de maquetación CSS tiene ciertas limitaciones técnicas con las que choca este efecto, en concreto no se permiten efectos los nodos hijos si el nodo padre ya tiene efectos aplicados.

Esta limitación ha sido de gran dificultad de eludir de forma eficiente, ya que los efectos se comportan de forma ligeramente diferente en cada navegador web. Se ha

[18] Un sistema de diseño es un conjunto coherente de principios, componentes y directrices que se utilizan para crear y mantener una apariencia y experiencia visual unificada y consistente en una plataforma, aplicación o producto digital.

[19] JSON es un formato de archivo estándar abierto de intercambio de datos que utiliza texto legible para almacenar y transmitir objetos de datos consistentes en duplas clave-valor y arrays.

conseguido una solución eficiente a este problema aplicando el efecto de glassmorphism al pseudo elemento “before” del Dock como se puede observar mediante las clases de utilidades de la Figura 10. Esta solución se ha aplicado a todos los elementos de la interfaz recreada del sistema operativo que presentaban esta limitación, incluido en este caso hasta detalles como la etiqueta que muestra el nombre encima del icono de la aplicación del Dock.

Para logar el efecto de ampliación se ha introducido en el Dock un evento de puntero de ratón, que ejecuta un callback al moverse por encima del Dock. Este evento proporciona datos necesarios para cálculos como las coordenadas del cursor en ese instante en el tiempo. Usando estas coordenadas en la pantalla y calculando el centro de los iconos del Dock en coordenadas relativas a la pantalla, que se define como la suma de la mitad del ancho del icono y la distancia del icono por la izquierda hasta el borde izquierdo de la pantalla. Con estos datos, mientras se detecta el cursor moviéndose por encima del Dock, se calcula la siguiente función de curva $(-x^2 / 200)$ que determina, según el valor del eje X, el aumento de escala de los iconos cercanos al cursor en ese momento.

```
const dockMagnificaitonCurve = (x, w, multiple = 1) =>
  (55 * multiple * Math.exp((-x * x) / 200 / 100) + w).toFixed(2);

const dockMagnification = (e) => {
  e.currentTarget.querySelectorAll('#dockItem').forEach((item) => {
    const centroid =
      item.offsetParent.offsetLeft -
      item.offsetParent.offsetWidth / 2 +
      item.offsetLeft +
      item.offsetWidth / 2;

    item.style.setProperty(
      'min-width',
      `${dockMagnificaitonCurve(e.clientX - centroid, item.offsetHeight)}px`
    );
  });
};
```

Figura 12: Fragmento de código que calcula el efecto de ampliación de los iconos del Dock

5.2.2 Barra de menú de aplicaciones

La barra de menú de aplicaciones es un componente común en muchas interfaces de usuario, especialmente en sistemas operativos de escritorio como macOS, que ofrece una forma estandarizada de organizar y proporcionar acceso a las funciones principales de una aplicación. Esta barra se encuentra en la parte superior de la pantalla y muestra las opciones y funciones de la aplicación activa.

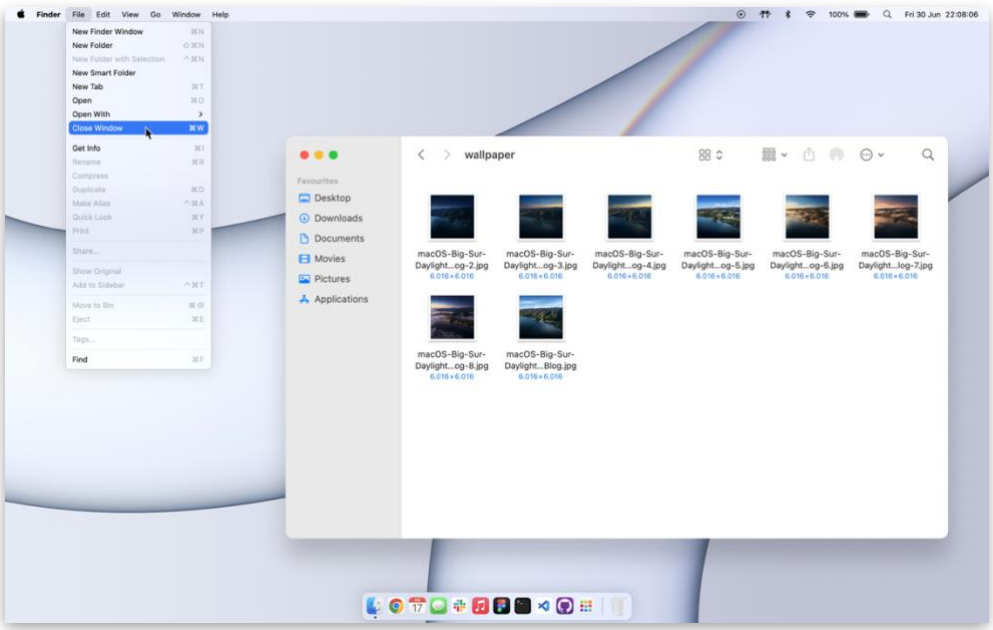

Figura 13: Captura del escritorio de la aplicación web con la aplicación Finder abierta, en la esquina superior izquierda se puede observar la barra de menú de aplicaciones con la opción "File" abierta

Como se observa en la Figura 13, la barra de menú aplica también el efecto de glassmorphism visto antes en el Dock, el cual heredan los submenús desplegables con diferentes funciones y opciones relacionadas que varían dependiendo del contexto y las características de la aplicación.

Este componente usa por debajo la librería Radix UI[20], en concreto el componente Menu Bar[21], que se adapta por defecto a los patrones de interacción de la barra de menús.

5.2.3 Centro de notificaciones

El centro de notificaciones centraliza y muestra de forma organizada las notificaciones y alertas que se generan en el dispositivo. Este permite mostrar varios widgets, diseñados para mostrar información actualizada y fácil de escanear a un golpe de vista.

[20] Radix UI es una biblioteca de componentes reutilizables y personalizables de interfaz de usuario que están optimizados para la accesibilidad y usan las mejores prácticas de diseño.

[21] https://www.radix-ui.com/docs/primitives/components/menubar

Las notificaciones listadas dentro del centro de notificaciones y las que aparecen deslizándose en el escritorio cuando el centro de notificaciones está cerrado han sido creadas con la librería de NotificationKit.

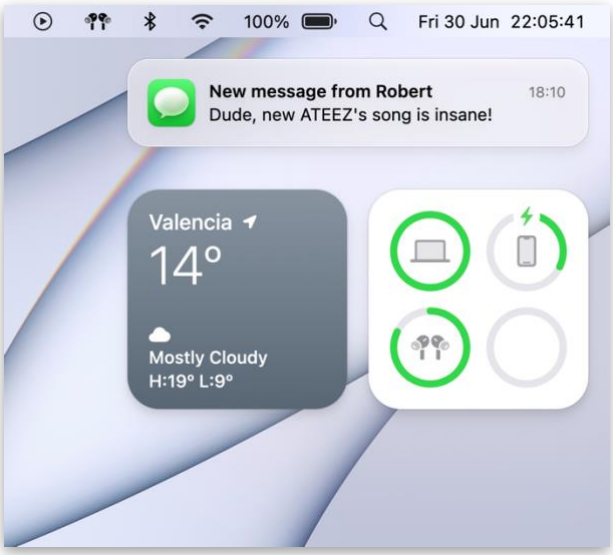

Figura 14: Captura del escritorio de la aplicación web, detalle de Centro de notificaciones abierto mostrando una notificación y dos widgets

En la Figura 14 se pude ver un ejemplo de notificación de la aplicación Mensajes y dos widgets. Ambos, notificación y widgets, permiten realizar acciones rápidas, como responder a mensajes o correos electrónicos directamente desde la propia notificación, marcar tareas como completadas, posponer recordatorios, entre otras opciones.

En el caso de esta implementación se ha tenido que prescindir del efecto de gradiente de opacidad en la parte inferior del centro de notificaciones, ya que no era posible aplicarlo a ningún pseudo-elemento porque debe contener los elementos en su interior para poder aplicar la máscara de gradiente de CSS. Si se hubiera aplicado ese efecto al padre se perdería la capacidad de añadir glassmorphism a las notificaciones o widgets.

5.2.4 Launchpad

Launchpad es un lanzador de aplicaciones visual y organizado. Se asemeja a la interfaz de un dispositivo móvil, como un iPhone o iPad, y permite a los usuarios acceder y abrir rápidamente aplicaciones instaladas en su Mac. Esto proporciona una forma rápida y conveniente de buscar y abrir aplicaciones sin tener que buscarlas en el Finder o en el Dock.

Cuando se activa el Launchpad, se muestra una pantalla completa que muestra los íconos de todas las aplicaciones disponibles en el sistema. Estos íconos se organizan

en múltiples páginas, y el usuario puede desplazarse horizontalmente para navegar entre ellas.

Figura 15: Captura del escritorio de la aplicación web con Launchpad abierto, mostrando algunas aplicaciones del sistema

Se puede ver en la parte superior de la Figura 15 un campo de texto, el cual hace la función de buscador, el cual implementa el método search() de la AppLibrary visto anteriormente. Conforme el usuario teclea en este campo de texto, se van mostrando los resultados de la búsqueda de la misma forma que se presentan las aplicaciones en la Figura 15 pero mostrando solo los resultados de la búsqueda.

Para la disposición de los iconos se ha utilizado un grid de CSS, que define en forma de tablero las filas y columnas donde se pueden introducir los elementos para conseguir un diseño más responsive y adaptable a todos los tamaños de pantalla.

5.2.5 Spotlight

Spotlight es una función de búsqueda integrada en macOS que permite a los usuarios encontrar rápidamente archivos, aplicaciones, configuraciones y otra información dentro del dispositivo. Funciona como un motor de búsqueda interno para el sistema operativo.

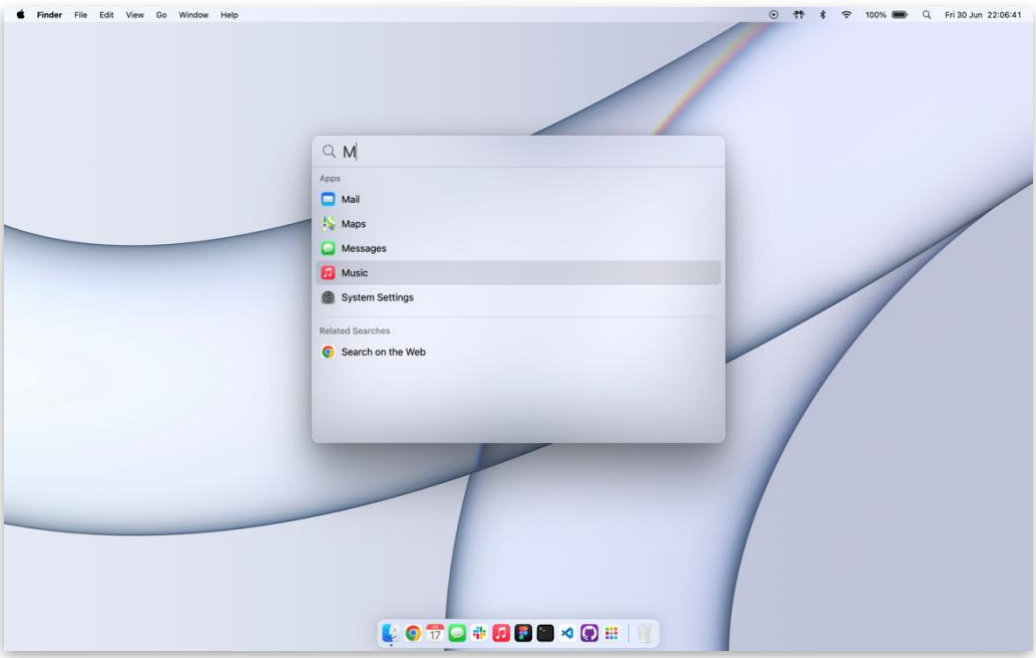

Figura 16: Captura del escritorio de la aplicación web con Spotlight abierto, se pueden ver los resultados de una búsqueda de prueba

Al activar Spotlight, se muestra una ventana de búsqueda en la parte superior del escritorio como se presenta en la Figura 16. El usuario puede escribir palabras clave, frases o nombres de aplicaciones para iniciar una búsqueda, la cual se obtiene mediante la función search() de AppLibrary. A medida que se escriben los términos de búsqueda, Spotlight ofrece resultados en tiempo real, mostrando coincidencias relevantes de aplicaciones del sistema.

Spotlight incluye varias características que mejoran la accesibilidad y eficiencia de su uso, como poder invocarlo con un atajo de teclado personalizado (“Option” + Espacio para Mac o “Alt” + Espacio para Windows o Linux) o poder seleccionar los resultados con las flechas de “arriba" y “abajo” del teclado.

5.3 Aplicaciones

Se ha decidido implementar tres aplicaciones que ilustran el uso de los servicios del sistema detallados anteriormente, donde Finder y Notas hacen uso de CoreData y la aplicación Música de CoreAudio, usando una lista reproducción predefinida.

5.3.1 Finder

Finder es el administrador de archivos predeterminado en el sistema operativo macOS. Permite a los usuarios navegar y gestionar archivos y carpetas en su dispositivo mediante una interfaz de usuario que muestra la estructura de carpetas y archivos. Los usuarios pueden explorar diferentes ubicaciones en su computadora, como el escritorio, documentos, descargas, aplicaciones y otras carpetas personalizadas.

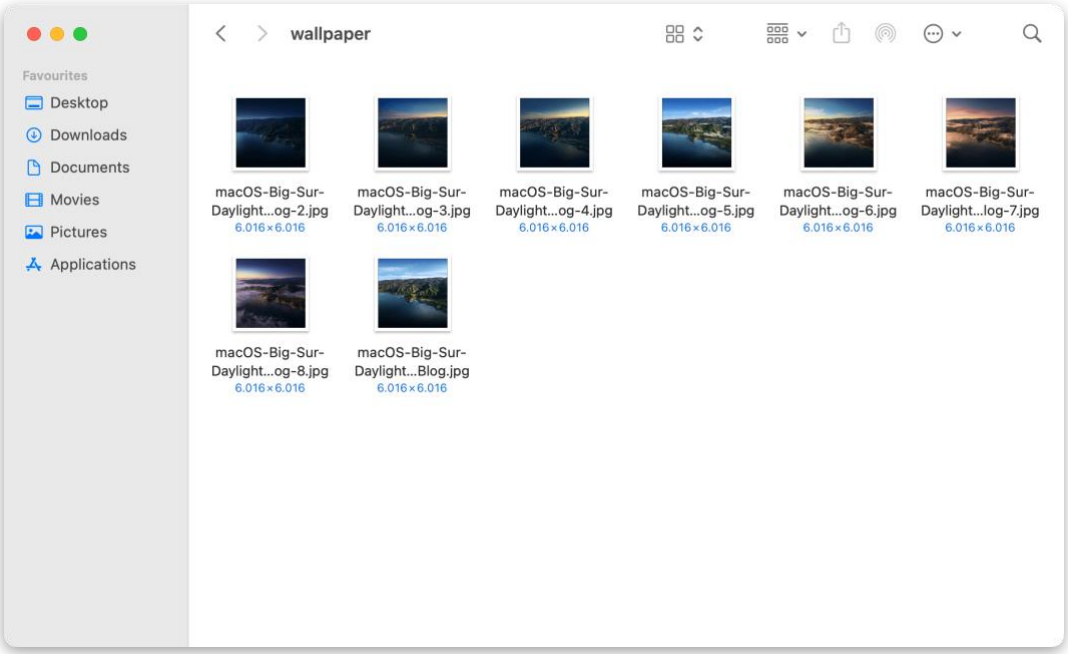

Figura 17: Captura de la aplicación web Finder recreada en lenguajes reactivos

En la Figura 17 se presenta la interfaz del Finder, que se divide en tres partes:

- **Barra lateral:** En el lateral izquierdo de la ventana de Finder, se encuentra la barra lateral, que proporciona acceso rápido a ubicaciones comunes, como escritorio, descargas, documentos, videos, imágenes y aplicaciones.
- **Área de contenido principal:** En el centro de la ventana se muestran los archivos y carpetas de la ubicación seleccionada. Se puede cambiar entre vista de iconos y vista de lista.
- **Barra de herramientas:** Encima del área de contenido principal, hay una barra de herramientas que contiene botones y opciones para realizar

acciones comunes, como copiar, pegar, eliminar, crear nuevas carpetas y realizar búsquedas.

Finder implementa CoreData para obtener el árbol del directorio, modificar y eliminar archivos y carpetas además de las funciones dedicadas de la aplicación como navegar por el árbol del directorio o buscar archivos y carpetas entre otras.

5.3.2 Música

Música, servicio nativo de Apple Music, muestra un álbum que permite reproducir las canciones de este mediante el uso de la librería del sistema CoreAudio. Esta implementación permite además la creación de un mini reproductor ubicado en la barra de menú de aplicaciones, en la esquina derecha superior de la pantalla.

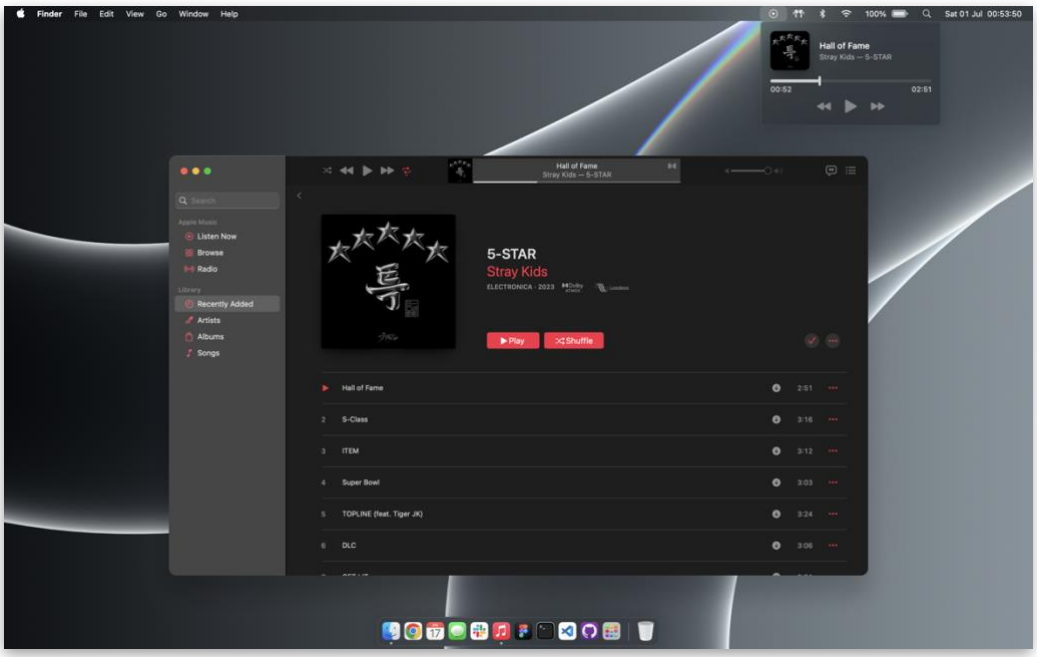

Figura 18: Captura de la aplicación web Música recreada en lenguajes reactivos

En la Figura 18 se puede observar cómo los estados de los dos reproductores están en sincronía CoreAudio. Además, cualquier acción sobre el mini reproductor sufrirá efecto en la aplicación de Música y viceversa, como si fueran una sola.

Al hacer uso de CoreAudio el reproductor puede realizar las acciones de reproducir y pausar, las cuales se ejecutan sobre la instancia de CoreAudio y actualiza globalmente el estado de reproducción que se verá reflejado en ambas interfaces.

También es posible de saltar directamente a cualquier canción de la lista o usar los controles de anterior y siguiente que obtienen su funcionalidad de los métodos prev() y next() de CoreAudio.

5.3.3 Notas

Notas es el ejemplo más sencillo y atómico de funcionalidad de la librería CoreData. Esta aplicación es capaz de cargar las notas que hayan guardadas y mostrarlas en la interfaz, sobrescribir el contenido en caché con las modificaciones de la nota y eliminar notas.

Para no sobrecargar la caché con operaciones de escritura y lectura por cada letra escrita se ha implementado un timeout[22] de 250 milisegundos, que se reestablece cada vez que el usuario presiona una tecla. En el momento que el usuario deja de teclear por más de 250 milisegundos se verifica si existen cambios en el contenido y, si procede, se actualiza el contenido en la caché.

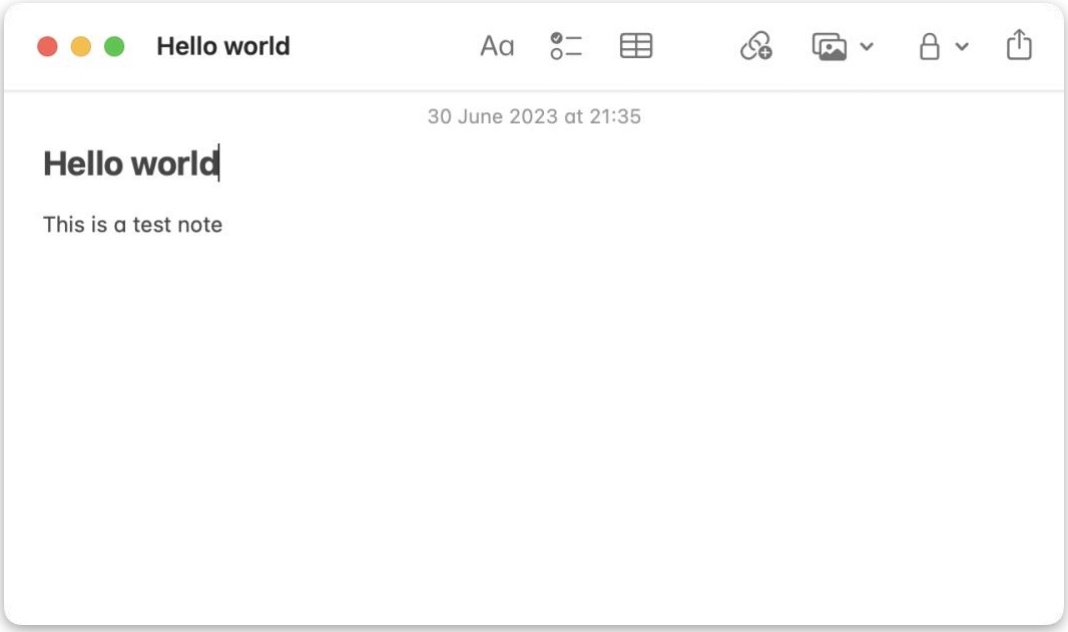

Figura 19: Captura de la aplicación Notas, recreada en lenguajes reactivos

En la Figura 19 se encuentra, justo debajo de la barra de herramientas, la fecha de última actualización, la cual se actualiza cada vez que se reescribe el contenido de la nota. Como la fecha, parte de los metadatos de la nota también son el título de la nota, el cual se extrae de la primera línea de contenido de la nota.

[22] El término "timeout" se refiere a un período de tiempo predefinido durante el cual un proceso debe completarse. Si el proceso excede ese tiempo establecido, se ejecuta un callback.

6 Implantación

En este capítulo se detalla el proceso para hacer accesible el proyecto desarrollado en este TFG de forma pública. Con este objetivo se ha escogido usar la plataforma Vercel, de los creadores de NextJS.

Las cualidades por las que Vercel ha sido elegido para alojar la aplicación web han sido, entre muchas otras, el hosting gratuito y de calidad, la integración completa con NextJS, dado que son sus creadores y el fácil uso y gestión del contenido y dominio desde su panel de control del sitio web.

Para empezar, se deberá tener una cuenta de Vercel o acceder mediante GitHub, en este caso crearemos la cuenta usando GitHub. A continuación, se deben seguir los pasos en pantalla para autorizar el uso de los datos de GitHub por parte de GitHub.

Una vez registrada la cuenta, se crea un nuevo proyecto y se selecciona el repositorio donde está almacenado el código del proyecto. Tras elegir la rama que se desea ejecutar, en este caso “main”, se procede a compilar el proyecto y publicarlo de manera automática.

Cuando la publicación del proyecto se ha realizado con éxito, Vercel nos proporcionará una URL pública única con la que podremos acceder al proyecto desde cualquier dispositivo.

Si, por otra parte, queremos ejecutar la aplicación de manera local, los requisitos del entorno de trabajo serán tener Yarn, NPM y la última versión de NodeJS instalados en el sistema. Una vez cumplidos los requisitos del entorno, simplemente se deberá descargar el repositorio del proyecto y actualizar las librerías usadas con el comando de terminal **yarn** y ejecutar el proyecto usando **yarn run dev**. El proyecto se levantará en localhost:3001, por lo que para visualizarlo se deberá acceder a dicha dirección desde un navegador web.

6.1 Despliegue automático

Tanto Vercel como GitHub tienen webhooks[23] asociados a acciones como nuevos commits o modificaciones en el código de una rama Git. Esto abre la puerta a automatizaciones como CI/CD y notificaciones de nuevas actualizaciones.

En este caso se ha aprovechado esta característica para automatizar el despliegue de la rama principal de la aplicación, de este modo ante cualquier cambio en la rama

[23] Un webhook es una forma de integración y comunicación entre aplicaciones y servicios web. Es un mecanismo mediante el cual una aplicación puede enviar automáticamente datos o notificaciones a otra aplicación cuando ocurre un evento o se produce una acción específica.

“main”, Vercel descargará el código y lo compilará. Si no hay ningún error en tiempo de compilación, los archivos estáticos generados a partir del árbol de NextJS se servirán en la dirección web del proyecto como la nueva versión del proyecto y por tanto se actualizará el contenido web al de la última versión del código.

En resumen, cada vez que un cambio de código se suba a la rama “main”, este se compilará y se servirá en la misma URL en la que está alojado el código, mostrando de manera automática siempre la última versión públicamente.

7 Conclusiones

En este último capítulo se analiza el cumplimiento de los objetivos, se presentan algunas líneas de trabajo para impulsar el desarrollo de interacciones con interfaces de usuarios modernas y se reflexiona sobre la relación del trabajo con asignaturas del grado de Ingeniería Informática.

7.1 Cumplimiento de los objetivos

El objetivo principal de este proyecto ha sido abordar uno de los desafíos más activos en el campo de la ingeniería del software y el diseño web: el desarrollo de interfaces eficientes e interacciones naturales. Para lograr este objetivo, se emplearon técnicas de maquetado web, ingeniería del software y diseño de interfaces.

Se puede observar cómo el trabajo ha cumplido con los objetivos propuestos, ya que ha cumplido con todos los objetivos planteados en la sección 1.2. de este documento, y el resultado se puede ver en apartado 5.

Con este trabajo se ha demostrado que las interfaces web han llegado, gracias a la evolución de los lenguajes de maquetado y desarrollo web, a un nivel de detalle y capacidades nunca vistas, que permiten una interacción fluida entre el usuario y el sistema, creando una relación similar en eficiencia y modelos mentales entre un usuario y el sistema operativo.

El área de las interfaces web seguirá evolucionando y se topará con cambios de paradigma que requerirán repensar las bases que conocemos hoy con tal de adaptarlas a las nuevas tecnologías y patrones de diseño de interfaces. Sin duda seguirá siendo un tema muy activo en comunidades de diseño y desarrollo.

7.2 Trabajo futuro

A modo de reflexión sobre las posibilidades de expansión del presente TFG se expone una lista de posibles mejoras y funcionalidades a modo de trabajo futuro. Esta colección de tareas se inspira en el uso cotidiano de un usuario medio, en este caso, al tratarse de un sistema operativo priorizaremos los servicios, funcionalidades y experiencias del propio sistema operativo, en el cual se apoyan el resto de las aplicaciones del sistema.

Como todo sistema operativo, los controles y métodos de entrada son la base de cualquier experiencia interactiva y deben ser consistentes a lo largo de toda la experiencia, así como familiares con los patrones y modelos mentales más comunes. De esta idea presentamos los primeros trabajos futuros:

- Creación de un sistema de diseño.
    - Crear y documentar tokens, moléculas y organismos.

    - Crear librería de componentes reutilizables en ReactJS y TailwindCSS.
    - Crear librería de iconos del sistema para asegurar la consistencia.
- Adaptar los componentes al estándar de accesibilidad WCAG [33].

A continuación, teniendo ya una base de componentes consistente, se presentan las tareas orientadas a dar mayor soporte de servicios del sistema para así aumentar las capacidades de este y de las diferentes aplicaciones.

- Crear CoreWeather — Obtener la previsión del tiempo dada una localización.
- Crear CoreLocation — Obtener la localización del usuario, en dos variantes:
    - Precisa: usando permisos del navegador, para ubicación exacta.
    - General: usando la IP del dispositivo, para ubicación aproximada.
- Crear CoreMedia — Gestión y lectura de periféricos como cámara, micrófono y altavoces accesibles desde el dispositivo.
- Crear CoreMotion — Procesar acelerómetro, giroscopio y eventos relacionados con el ambiente y sensores.
- Extender funcionalidades y opciones de NotificationKit.
- Crear PDFKit — Visualización, marcado y edición de archivos PDF.
- Extender CoreWindow — Ofrecer más flexibilidad y añadir funcionalidades como diálogos y alertas.

Seguidamente, con el gran conjunto de nuevos componentes y servicios, nos centraremos en las aplicaciones, teniendo en cuenta las limitaciones técnicas y de seguridad de las aplicaciones web.

- Creación de Preferencias del Sistema
    - Ajustes de Cuenta.
    - Ajustes de Red.
    - Ajustes de Bluetooth.
    - Ajustes de Notificaciones — Gestión de notificaciones global y por aplicación para un mayor control.
    - Ajustes de Seguridad y Privacidad — Gestión de permisos globales y de aplicaciones.
    - Ajustes de Escritorio y Dock.
    - Ajustes de Fondo de pantalla — Dar una selección de fondos de pantalla de calidad al usuario, en variantes de color sólido e imágenes estáticas, dinámicas o modo día y noche. Además de

ofrecer la posibilidad de usar una imagen del dispositivo como fondo de pantalla.

  - Ajustes de Contraseña — Permite al usuario cambiar la contraseña.
- Mejorar de Finder — Permitir todas las acciones CRUD y ampliar las capacidades de visualización y gestión de carpetas y archivos.
- Mejorar Música — Permitir buscar música, guardar en la biblioteca, filtros de visualización, cola de reproducción, letras de la canción y creación y gestión de listas.
- Mejorar aplicación de Spotlight — Añadir funcionalidades no implementadas, como conversión de divisas, sugerencias de búsquedas web, operaciones matemáticas básicas, definiciones de palabras y más.
- Añadir la aplicación Preview — Permite abrir y previsualizar archivos del sistema de archivos.
- Añadir aplicación Recordatorios — Con el uso de CoreData, poder crear y gestionar listas de recordatorios.
- Añadir la aplicación Calendario — Visualizar los días, semanas y meses del año siguiendo los patrones del calendario del sistema, permitiendo además crear y gestionar eventos.

Se puede observar que las posibilidades de ampliación son infinitas al tratarse de un programa tan complejo como un sistema operativo, es por eso por lo que las tareas se han seleccionado pensando en lo esencial para el uso del sistema operativo como un usuario avanzado.

7.3 Relación con las asignaturas

En esta sección se analizan las diferentes habilidades y conocimientos aplicados en el trabajo que se han adquirido a lo largo del grado de Ingeniería Informática. A continuación, se detallan las asignaturas que más relación han tenido con el proyecto:

- **Interfaces Persona-Computador (IPC):** en esta asignatura se han desarrollado conceptos relacionados con las interfaces, experiencia de usuario, accesibilidad y diseño centrado en el usuario. Ha sido realmente de ayuda ya que amplía conceptos realmente importantes para conseguir unas interacciones entre persona y máquina con la menor fricción posible.
- **Ingeniería del Software (ISW):** en esta asignatura se han explicado las bases generales de la ingeniería de la programación, tales como el ciclo del desarrollo, la integración o las pruebas unitarias. Ha sido realmente útil para entender las relaciones técnicas entre los artefactos software y en la parte práctica de “picar” código.
- **Diseño de Software (DDS):** en esta asignatura se han enseñado los patrones básicos de la programación y se han aplicado en proyectos de

clase para ver su utilidad y practicidad a la hora de lidiar con problemas recurrentes que los patrones pueden resolver.

- **Integración e Interoperabilidad (IEI):** en esta asignatura se han explicado la reutilización de servicios distribuidos y la integración con estos para crear sistemas que puedan trabajar de forma conjunta.
- **Proyecto de Ingeniería de Software (PSW):** en esta asignatura se desarrolló un proyecto software en grupo desde cero, pasando por todas las etapas necesarias para un desarrollo ágil. Está asignatura ha ayudado muy positivamente en la parte práctica del desarrollo de proyectos software, donde al vivirlo en primera persona se aprende mucho más.

8 Referencias

[1] Simon Kemp, «DIGITAL 2023: GLOBAL OVERVIEW REPORT», *https://datareportal.com/reports/digital-2023-global-overview-report*, Jan. 26, 2023.

[2] S. P. Mirashe and N. V. Kalyankar, «Cloud Computing», Mar. 2010, [Online]. Available: http://arxiv.org/abs/1003.4074

[3] Michal Malewicz, «Glassmorphism in user interfaces», *UX Collective*, Nov. 2020, Accessed: Jun. 28, 2023. [Online]. Available: https://uxdesign.cc/glassmorphism-in-user-interfaces-1f39bb1308c9

[4] Apple Inc., «Human Interface Guidelines», *https://developer.apple.com/design/human-interface-guidelines*.

[5] MDN Contributors, «Window: localStorage property», *https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage*.

[6] Alexander Shvets, «Composite Pattern», *https://refactoring.guru/es/design-patterns/composite*.

[7] Alexander Shvets, «State Pattern», *https://refactoring.guru/es/design-patterns/state*.

[8] Alexander Shvets, «Observer Pattern», *https://refactoring.guru/es/design-patterns/observer*.

[9] Meta Inc. and React, «Built-in React Hooks», *https://react.dev/reference/react*.

[10] Meta Inc. and React, «useEffect Hook», *https://react.dev/reference/react/useEffect*.

[11] Alexander Shvets, «Singleton Pattern», *https://refactoring.guru/es/design-patterns/singleton*.

[12] Alexander Shvets, «Bridge Pattern», *https://refactoring.guru/es/design-patterns/bridge*.

[13] Meta Inc. and React, «Passing Props to a Component», *https://react.dev/learn/passing-props-to-a-component*.

[14] Figma, «Figma», *https://www.figma.com/*.

[15] Microsoft, «Visual Studio Code», *https://code.visualstudio.com/*.

[16] GitHub Inc., «GitHub», *https://github.com/*.

[17] GitHub Inc., «GitHub Desktop», *https://desktop.github.com/*.

[18] Apple Inc., «SF Symbols», *https://developer.apple.com/sf-symbols/*.

[19] Microsoft, «TypeScript», *https://www.typescriptlang.org/*.

[20] Brendan Eich, «JavaScript», *https://developer.mozilla.org/en-US/docs/Web/JavaScript*.

[21] Adam Wathan and Steve Schoger, «TailwindCSS», *https://tailwindcss.com/*.

[22] Facebook, «React», *https://react.dev/*.

[23] Felipe Restrepo-Calle, «Programación Reactiva», *http://ferestrepoca.github.io/paradigmas-de-programacion/reactive/reactive_teoria/index.html*.

[24] Inc. Meta Platforms, «Virtual DOM and Internals», *https://legacy.reactjs.org/docs/faq-internals.html*.

[25] Vercel Inc., «NextJS», *https://nextjs.org/*.

[26] Vercel Inc., «Vercel», *https://vercel.com/*.

[27] IBM, «What is Platform-as-a-Service (PaaS)?», *https://www.ibm.com/topics/paas*.

[28] Atlassian, «Jira», *https://www.atlassian.com/software/jira*.

[29] Microsoft, «Microsoft Word», *https://www.microsoft.com/es-es/microsoft-365/word*.

[30] MDN Contributors, «Base64», *https://developer.mozilla.org/en-US/docs/Glossary/Base64*.

[31] Timo Lins, «React hot toast», *https://react-hot-toast.com/*.

[32] Tailwind CSS, Adam Wathan, Jonathan Reinink, David Hemphill, and Steve Schoger, «Utility-First Fundamentals», *https://tailwindcss.com/docs/utility-first*.

[33] Shawn Lawton Henry, «WCAG 2 Overview», *https://www.w3.org/WAI/standards-guidelines/wcag/*, Jun. 05, 2018.

[34] Naciones Unidas, «Objetivos de Desarrollo Sostenible», *https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/*.

[35] UN. General Assembly (71st sess. : 2016-2017), «Work of the Statistical Commission pertaining to the 2030 Agenda for Sustainable Development : resolution / adopted by the General Assembly», New York, Jul. 2017. Accessed:

Jun. 26, 2023. [Online]. Available: https://digitallibrary.un.org/record/1291226?ln=en

[36] Naciones Unidas, «Objetivo 4: Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos», *https://www.un.org/sustainabledevelopment/es/education/*.

[37] Naciones Unidas, «Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos», *https://www.un.org/sustainabledevelopment/es/economic-growth/*.

[38] Naciones Unidas, «Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación», *https://www.un.org/sustainabledevelopment/es/infrastructure/*.

9 Reflexión sobre los objetivos ODS

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**		X		
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Tabla 7: Objetivos de Desarrollo Sostenible (ODS)

Los Objetivos de Desarrollo Sostenible (ODS) [34] son 17 metas globales diseñadas para promover un futuro mejor y más sostenible en todo el mundo. Fueron establecidos en 2015 por la Organización de las Naciones Unidas [35].

Este trabajo está estrechamente relacionado con los siguientes ODS:

- **Educación de calidad (Objetivo 4)** [36]**:** ya que la creación de estas aplicaciones web hace que puedan ser más accesibles, tanto a nivel de interfaces como a nivel de dispositivo dado que son aplicaciones multiplataforma.
- **Trabajo decente y crecimiento económico (Objetivo 8)** [37]**:** la implementación de aplicaciones usables, rápidas y de bajo coste basadas

en interfaces e interacciones web que facilitan las tareas ayudan a crear un sistema de trabajo óptimo.

- **Industria, innovación e infraestructura (Objetivo 9)** [38]**:** este trabajo presenta una solución innovadora para facilitar las interacciones persona-computador, por tanto, contribuye a la innovación.