



UNIVERSIDAD TÉCNICA  
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA  
INGENIERÍA EN INFORMÁTICA

## Plataforma Tecnológica Inclusiva para el Agendamiento de Horas Médicas: Investigación y Desarrollo Front-End

Maciel Ripetti castillo

[mripetti@usm.cl](mailto:mripetti@usm.cl)

Pamela Gatica Caballero  
Profesor Guía

Franco Jorquera  
Profesor Correferente

**Resumen:** Se describe el desarrollo *Front-End*, implementación y validación de la plataforma para el agendamiento de horas medicas SuHorita, una aplicación web orientada a reducir las barreras tecnológicas y cognitivas que enfrentan numerosos usuarios al momento de agendar horas médicas. El proyecto surge de la necesidad de facilitar el acceso al sistema de salud mediante una plataforma intuitiva, accesible, moderna y centrada en el usuario.

A lo largo del documento se aborda el contexto teórico, la metodología aplicada y las decisiones de diseño basadas en principios de usabilidad y accesibilidad. Se detalla el proceso de desarrollo del *Front-End*, considerando componentes interactivos, flujos de navegación y la integración de estándares WCAG y atributos ARIA. Posteriormente, se presenta una evaluación exhaustiva mediante pruebas funcionales, de usabilidad, de accesibilidad y de rendimiento, asegurando que la solución cumpla con los requisitos técnicos y la experiencia de uso esperada.

**Palabras Clave:** Accesibilidad, Usabilidad, Agendamiento médico, Plataforma web.



## CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

### 1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

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

**Título del trabajo:** PLATAFORMA TECNOLÓGICA INCLUSIVA PARA EL AGENDAMIENTO DE HORAS MEDICAS: INVESTIGACIÓN Y DESARROLLO FRONT-END

**Nombre del candidato(a):** Maciel Daly Ripetti Castillo

**Carrera / Grado:** Ingeniería en Informática

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

### 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

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

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

### 3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

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

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

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

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

---

---

---

### 4.- FIRMAS

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

**Fecha:** 11 - 03 - 2026

**Firma:**

**Estudiante o Candidato(a):**

**Fecha:** 11 - 03 - 2026

**Firma:**

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

# 1 Introducción

## 1.1 Contexto

El acceso a la salud es un derecho fundamental que, en la era digital, depende cada vez más de plataformas tecnológicas eficientes y accesibles. Si bien estas soluciones han optimizado la gestión de procesos médicos como el agendamiento, la mayoría presenta interfaces complejas que excluyen a usuarios con barreras cognitivas, tecnológicas o con baja alfabetización digital, ampliando las desigualdades en el acceso a la atención médica.

En la Corporación Municipal de Viña del Mar (CMVM) se evidencia la ausencia de un sistema de agendamiento inclusivo y confiable. Esta carencia limita el acceso a la salud de usuarios que requieren un entorno digital simple, accesible y centrado en la experiencia de uso, lo que subraya la necesidad de un *Front-End* inclusivo capaz de responder a distintos perfiles de pacientes.

## 1.2 Definición del problema

El problema central radica en la ausencia de una plataforma de agendamiento con un *Front-End* inclusivo y robusto, en la CMVM, afectando tanto al usuario paciente como al usuario administrativo.

El módulo de Agenda AVIS, que es el utilizado actualmente por la CMVN, presenta fallas críticas que generan ineficiencia operativa y desigualdad en el acceso:

### 1.2.1 Problema de Gestión Administrativa

El sistema actual, AVIS, carece de validaciones adecuadas que generan ineficiencias:

1. Registros vacíos: ingreso de fichas con datos incompletos.
2. Duplicidad de solicitudes: múltiples ingresos para una misma cita, generando registros redundantes y desorden en la base de datos.

Esto se traduce en procesos administrativos poco confiables y en una gestión de citas ineficiente.

### 1.2.2. Problema de Acceso e Inclusión

La falta de un sistema digital inclusivo, sumada a la alta demanda de pacientes, las limitaciones de acceso a tecnología y la diversidad en los niveles de alfabetización digital, genera consecuencias críticas:

1. Barreras de acceso oportuno: dificultades para adultos mayores y personas con discapacidad al interactuar con sistemas tradicionales.
2. Riesgo en la continuidad de tratamientos: deficiencias en la gestión de citas interrumpen la adherencia médica.
3. Déficit de comunicación: ausencia de recordatorios automáticos y de canales efectivos de confirmación.

En conjunto, estos factores afectan la calidad del servicio y profundizan la brecha digital en la atención de salud de la comunidad de Viña del Mar.

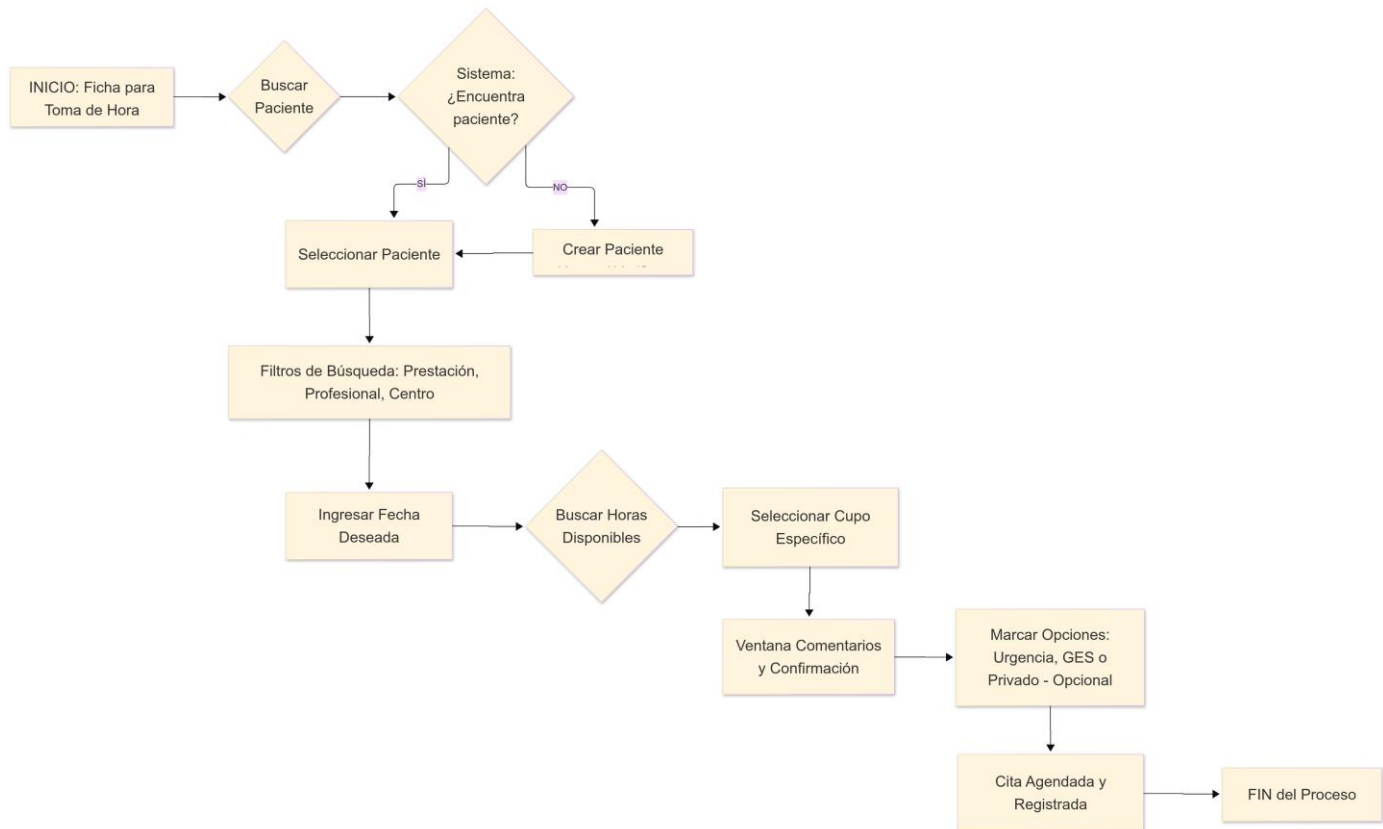


**Figura 1: Flujo actual de la ficha para agendar una hora médica**

Fuente: Elaboración propia

### 1.3 Descripción sobre la propuesta de solución

La propuesta consiste en el desarrollo del *Front-End* inclusivo de una plataforma de agendamiento médico para la CMVM, la cual se separa en una aplicación web para el paciente y una extensión web para el personal administrativo. En esta tesina nos enfocaremos en solo en la aplicación web inclusiva para el paciente.

El diseño seguirá los principios del Diseño Centrado en el Usuario (DCU) y cumplirá con las Pautas de Accesibilidad para el Contenido Web (WCAG), para así de e incorporar una interfaz de bajo esfuerzo cognitivo, con navegación simplificada, íconos claros, tipografía legible y contraste adecuado, para atender las necesidades de adultos mayores, personas con discapacidades visuales como daltonismo, o dificultades específicas del aprendizaje (DEA) como dislexia. Su diseño será responsivo, optimizado para teléfonos móviles, considerando que este es el dispositivo más utilizado por la población para acceder a internet.

El *Front-End* incorporará mecanismos de validación automática de datos esenciales, como RUN y campos obligatorios, lo que evitará registros incompletos. Además, se incluirá un control de duplicidad de citas, que no permitirá agendar una solicitud repetida, garantizando mayor orden y confiabilidad en la información.

### 1.3.1 Dependencias de Desarrollo

Para garantizar un desarrollo robusto, mantenible y accesible, se seleccionó un *stack* tecnológico moderno.

El lenguaje *TypeScript* permitirá un tipado estático que reduzca errores y facilite la escalabilidad del proyecto. En conjunto con *React*, se desarrollará una interfaz modular y reutilizable, optimizada para la creación de componentes accesibles.

En cuanto a la capa visual, se empleará *TailwindCSS* como *framework* de estilos para construir diseños consistentes, responsivos y adaptables a diferentes dispositivos, priorizando la experiencia en teléfonos móviles. Se complementará con *PostCSS* y *Autoprefixer*, que aseguran compatibilidad en diversos navegadores y mejoran la mantenibilidad de los estilos.

Este *stack* fue seleccionado no solo por su solidez técnica, sino porque facilita la implementación de buenas prácticas de accesibilidad, esenciales para un *Front-End* orientado a la inclusión digital en salud.

### 1.3.2 Características específicas del sistema

En términos de accesibilidad, se desarrollará una interfaz de bajo esfuerzo cognitivo, con navegación simplificada, iconografía clara, tipografía legible y bajo contraste, pensada especialmente para adultos mayores y personas con discapacidad. Además, el diseño será responsivo, priorizando la visualización en smartphones.

Se implementarán validaciones automáticas de datos esenciales por *Front-End*, como el RUN, pasaporte, NIP y campos obligatorios, evitando el registro de fichas incompletas y aumentando la confiabilidad de la base de datos.

### 1.3.3 Casos de éxito

La viabilidad de esta propuesta se sustenta en experiencias previas, tanto a nivel público como privado. Las plataformas de telemedicina implementadas por el MINSAL y en otros países han demostrado que el cumplimiento de los estándares WCAG permite que usuarios con discapacidad visual o que utilizan lectores de pantalla puedan acceder a funciones críticas como el agendamiento.

En el ámbito privado, diversos sistemas de agendamiento que incorporan recordatorios automáticos por SMS o WhatsApp han logrado disminuir el ausentismo en rangos del 20% al 40%, mostrando la efectividad de estas herramientas en la continuidad de tratamientos. Asimismo, los sistemas de identificación unificada utilizados por instituciones como el Registro Civil muestran cómo las validaciones en tiempo real fortalecen la confiabilidad de los datos.

Estos referentes evidencian que la integración de accesibilidad, comunicación automatizada y validación de datos son estrategias efectivas y replicables. La propuesta de esta tesina busca adaptar dichas prácticas al contexto de la CMVM, construyendo un *Front-End* inclusivo, confiable y socialmente pertinente.

## 1.4 Objetivos Generales y Específicos de la Tesina

**Objetivo General:** Implementar el *Front-End* de la plataforma de agendamiento de horas médicas "SuHorita", priorizando la accesibilidad y la usabilidad para optimizar la experiencia del usuario (UX) y reducir las barreras digitales en el acceso a la salud.

### Objetivos Específicos

1. Construir la arquitectura *Front-End* basada en componentes reutilizables, siguiendo las directrices de diseño inclusivo entregadas por el equipo de UX.
2. Desarrollar los flujos principales del paciente en la aplicación móvil: agendamiento, confirmación y cancelación de citas.
3. Incorporar mecanismos de accesibilidad en el *Front-End*, incluyendo colores amigables, tipografías adaptadas para dislexia y ajuste dinámico de tamaño de fuente, cumpliendo con los estándares.
4. Validar la accesibilidad y usabilidad de la interfaz mediante pruebas con usuarios representativos, evaluando si la solución reduce barreras digitales en comparación con el sistema actual.

## 1.5 Justificación de proyecto

El proyecto "SuHorita" se justifica en tres dimensiones claves: social-ética, práctica-organizacional y académica-tecnológica. En el ámbito social y ético, esta se aborda la urgencia de reducir la brecha digital en el acceso a la salud, un derecho fundamental. La falta de un sistema de agendamiento inclusivo en la CMVM perpetúa la exclusión de usuarios vulnerables, como adultos mayores o personas con barreras visuales o cognitivas al depender de interfaces y sistemas complejos. El desarrollo de un *Front-End* accesible, alineado con el objetivo general, busca eliminar estas barreras mediante la implementación de funcionalidades como la tipografía adaptable.

Desde una perspectiva práctica y organizacional, la solución propuesta es crítica para optimizar la eficiencia interna. El sistema actual adolece de fallas que generan duplicidad de registros e ineficiencia administrativa, impactando directamente la gestión de citas. La implementación de la extensión web para el ejecutivo y médico proporcionará herramientas de gestión robustas y un sistema de notificaciones automáticas que se espera reducirá drásticamente las tasas de ausentismo, liberando horas médicas y mejorando la continuidad de los tratamientos.

Finalmente, en el plano académico-tecnológico, el proyecto sirve como un caso de estudio riguroso en la aplicación de principios de Diseño Centrado en el Usuario (DCU) y estándares de accesibilidad WCAG en un entorno de alta criticidad social. La implementación de una arquitectura *Front-End* moderna basada en componentes reutilizables demostrará la viabilidad de construir sistemas escalables, aportando conocimiento tangible a las prácticas de desarrollo de *software* público.

## 1.6 Metodología

El desarrollo de la plataforma "SuHorita" se gestionará mediante la metodología *Scrum*, un *framework* ágil e iterativo que permite entregar valor incremental y adaptarse continuamente al diseño inclusivo. Dado el enfoque en UX y accesibilidad, la naturaleza iterativa de *Scrum* es esencial para integrar y validar criterios de usabilidad y accesibilidad en ciclos cortos.

### 1.6.1 Roles y Artefactos Centrales

El proceso se organiza a partir de historias de usuario que representen tanto la funcionalidad general como las necesidades inclusivas. El *Product Backlog* contendrá todas las funcionalidades priorizadas, incluyendo aquellas de accesibilidad.

Ejemplo de historia de usuario orientada a inclusión:

- 1 "Como paciente con visión reducida quiero ampliar el tamaño de la letra para agendar cómodamente mi cita médica."

El *Sprint Backlog* seleccionará las historias prioritarias para el desarrollo del *Front-End* en cada ciclo, asegurando que los criterios de accesibilidad estén presentes desde los primeros *Sprints*. Cada incremento será una porción funcional, probada y usable de la plataforma.

### 1.6.2 Ciclos y Fases de Desarrollo

El desarrollo se segmentará en *Sprints* de duración fija y breve (máximo 2 semanas). Cada *Sprint* se centrará en un módulo funcional completo, integrando accesibilidad desde el inicio y estableciendo validaciones específicas (funcionalidad, usabilidad y rendimiento) en cada ciclo.

1. *Sprint* 1 – Cimentación y Arquitectura
  - A. Objetivo: Definir la arquitectura *Front-End* y fundamentos tecnológicos.
  - B. Accesibilidad transversal: Configuración del sistema de diseño inicial con criterios de accesibilidad.
  - C. Validación:
    - Pruebas unitarias de los componentes iniciales.
    - Chequeo de accesibilidad básica.
    - Revisión de consistencia del sistema de diseño inclusivo.
2. *Sprint* 2–3 – Módulo del Paciente (App Móvil)
  - A. Objetivo: Desarrollar componentes base y flujos de registro, agendamiento, cancelación y notificaciones.
  - B. Accesibilidad transversal: Implementación de colores amigables, ajuste de tipografía y ampliación de letra, navegación simple y elementos de lectura fácil.
  - C. Validación:
    - Pruebas unitarias de flujos y componentes interactivos.
    - Validación de usabilidad con usuarios para verificar comprensión, navegación y accesibilidad de los flujos.

## 1.7 Estructura del informe

El informe se estructurará en seis capítulos secuenciales para exponer de manera lógica y coherente el proceso completo, desde la justificación teórica hasta la implementación práctica.

- **Capítulo 1: Introducción:** Establece el contexto del problema, la justificación social y técnica del proyecto, y presenta los objetivos generales y específicos que guían la investigación y el desarrollo.
- **Capítulo 2: Marco teórico:** Documenta el estado del arte y los fundamentos teóricos. Se revisarán los principios del DCU, y se analizarán los sistemas de arquitectura *Front-End* modernos.
- **Capítulo 3: Implementación del Módulo Paciente.** Se describe el desarrollo del *Front-End* de la aplicación móvil. Aquí se evidenciará la implementación de los flujos de registro, agendamiento, confirmación de citas, la vista inicial y se presenta la integración de todas las funcionalidades de accesibilidad para el paciente.
- **Capítulo 4: Pruebas y Validación:** Presenta el proceso de evaluación del módulo desarrollado, con el objetivo de garantizar su correcto funcionamiento, accesibilidad y alineación con los requerimientos definidos. Se describen las metodologías de prueba aplicadas, los escenarios considerados y los resultados obtenidos.
- **Capítulo 5: Conclusiones y Trabajo Futuro:** Se evalúa el cumplimiento de los objetivos específicos y el impacto del proyecto en la mejora de la usabilidad y la reducción de barreras. Se identifican las limitaciones del desarrollo y se proponen futuras líneas de investigación y expansión.

## 2 Marco Teórico

### 2.1 Fundamentos del desarrollo *Front-End*

El desarrollo *Front-End* ha evolucionado de la simple presentación de información a la gestión de aplicaciones complejas de una sola página (SPA), donde la interactividad, rendimiento y accesibilidad son fundamentales. Este segmento establece los pilares tecnológicos y de diseño que sustentan el desarrollo de SuHorita, asegurando que la interfaz no solo sea funcional sino también inclusiva y escalable.

#### 2.1.1 Plataformas Web Progresivas (PWA)

La implementación como PWA permite ofrecer una experiencia similar a aplicaciones nativas, sin barreras de instalación, beneficiando a usuarios con baja alfabetización digital o dispositivos de gama baja.

- Inclusión tecnológica: Acceso *offline* y consumo reducido de recursos, reduciendo brechas por conectividad limitada.
- Resiliencia: *Service Worker* permite acceso a información crítica incluso con red inestable.
- Herramientas de implementación: Módulos como *next-pwa* garantizan la correcta configuración y despliegue de estas capacidades.

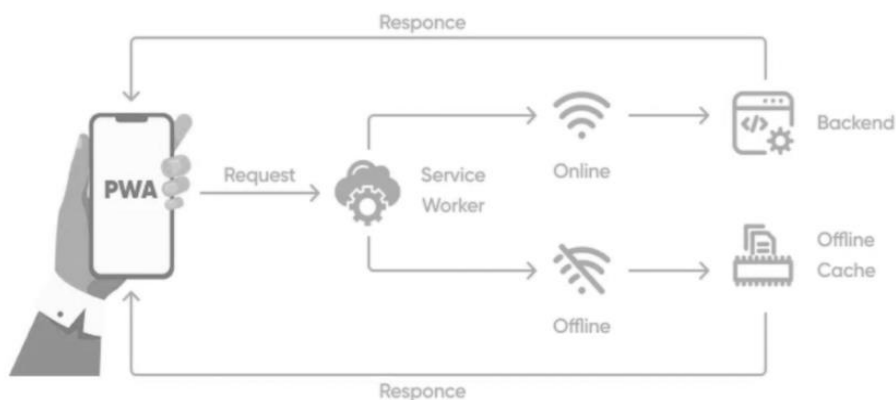


**Ilustración 1: Diagrama PWA**

Fuente: Extraída de [1]

#### 2.1.2 Ecosistema tecnológico moderno y tipado estático

El *stack* seleccionado respalda robustez, mantenibilidad y accesibilidad:

- *React*: Biblioteca declarativa que facilita la gestión de estado y la reutilización de componentes accesibles.
- *TypeScript*: Tipado estático que reduce errores y asegura la manipulación precisa de datos sensibles.
- *Tailwind CSS: Utility-first* que facilita estilos responsivos, alto contraste y ajustes de tipografía para accesibilidad cognitiva.
- Iconografía clara (*react-icons*): Mejora la usabilidad y reconocimiento visual, importante para usuarios con baja alfabetización digital.

### 2.1.3 Optimización de rendimiento en el Front-End

El rendimiento es un factor crítico en aplicaciones web orientadas a servicios esenciales, especialmente cuando son utilizadas desde dispositivos móviles de gama baja o con conectividad limitada. Una de las técnicas más utilizadas para optimizar el rendimiento en *Front-End* es el *lazy loading*, que consiste en cargar recursos (componentes, imágenes o módulos) únicamente cuando son necesarios, lo que permite reducir el tiempo de carga inicial de la aplicación. *Frameworks* modernos como Next.js incorporan soporte nativo para *lazy loading*, facilitando su implementación y su integración con estrategias de optimización orientadas a PWA y rendimiento general.

## 2.2 Barreras de accesibilidad a abordar

### 2.2.1 Identificación de barreras

El proceso de agendamiento de horas médicas mediante plataformas digitales presenta múltiples barreras que afectan de forma directa a la población usuaria del sistema público de salud. Estas barreras no solo están asociadas a discapacidades permanentes, sino también a condiciones temporales, contextuales o socioeducativas, como la baja alfabetización digital o el uso de dispositivos móviles de gama baja.

En el contexto de la CMVM, se identifican problemáticas recurrentes relacionadas con la complejidad para tomar agendar horas médicas, la falta de validaciones oportunas y la ausencia de retroalimentación clara para el usuario. Estas dificultades impactan especialmente a adultos mayores y usuarios con limitada experiencia en el uso de tecnologías digitales.

### 2.2.2 Definición del conjunto final de barreras

A partir del análisis del problema y las recomendaciones establecidas por las Pautas de Accesibilidad para el Contenido Web (WCAG), se define un conjunto acotado y justificado de barreras prioritarias a abordar en el desarrollo del Front-End de SuHorita.

Las barreras seleccionadas responden a criterios de frecuencia, impacto en el acceso a la salud y viabilidad técnica de mitigación mediante Front-End, permitiendo enfocar los esfuerzos de diseño e implementación en soluciones concretas y medibles.

Las barreras abordadas son las siguientes:

- Barreras visuales: baja visión y dificultades de discriminación de color.
- Barreras cognitivas: Dificultad de comprensión lectora y dislexia.
- Baja alfabetización digital: desconocimiento de flujos digitales complejos y temor a cometer errores.
- Barreras motrices: dificultad para interactuar con elementos pequeños o precisos en pantallas táctiles.

### 2.2.3 Justificación de las barreras seleccionadas

La selección de estas barreras se fundamenta en tres aspectos principales:

1. Relevancia social: los grupos afectados por estas barreras representan una proporción significativa de los usuarios del sistema público de salud, especialmente adultos mayores y personas con menor acceso a formación digital.

2. Impacto directo en la continuidad de la atención médica: dificultades en el agendamiento, confirmación o cancelación de horas médicas pueden derivar en ausentismo, retrasos en tratamientos y sobrecarga administrativa.
3. Capacidad de mitigación desde el Front-End: todas las barreras definidas pueden ser abordadas mediante decisiones de diseño, estructura de navegación, validaciones, estilos visuales y uso de estándares de accesibilidad, sin depender de infraestructura externa.

Este enfoque permite que SuHorita no intente resolver todas las problemáticas posibles, sino que se concentre en aquellas donde el *Front-End* inclusivo puede generar un impacto real y verificable.

## 2.2.4 Microcops y reducción de la carga cognitiva

Los *microcops* son a textos breves y específicos que guían al usuario durante la interacción con una interfaz, como mensajes de ayuda, validaciones, confirmaciones o instrucciones contextuales. En el contexto de la accesibilidad cognitiva, los *microcops* juegan un rol fundamental al reducir la incertidumbre, prevenir errores y facilitar la comprensión de los flujos digitales. [2]

Para usuarios con baja alfabetización digital o dificultades cognitivas, el uso de *microcops* claros, concisos y en lenguaje simple permite disminuir la carga mental asociada a la toma de decisiones, mejorando la eficiencia y la confianza durante el uso de la aplicación. Su correcta implementación es especialmente relevante en formularios y procesos críticos, como el agendamiento de horas médicas.

## 2.3 Diseño Centrado en el Usuario (DCU)

Las interfaces se diseñan con retroalimentación continua de usuarios reales y pruebas de usabilidad, asegurando que cada módulo sea intuitivo y accesible desde el primer *sprint*. Esto permite incorporar criterios de accesibilidad de forma transversal en todos los *sprints*, en lugar de dejarlos como ajustes finales.

### 2.3.1 Call To Action (CTA) [3]

Dentro del enfoque de DCU, los CTA representan elementos clave para guiar al usuario hacia la acción esperada en cada etapa del flujo. Ya que estos son mensajes o elementos visuales (como un botón) que incitan al usuario a realizar una acción específica e inmediata. Un CTA bien diseñado debe ser visible, comprensible y coherente con el contexto, evitando ambigüedades o sobrecarga de opciones.

En aplicaciones orientadas a usuarios con baja experiencia digital, es recomendable limitar la cantidad de CTA por pantalla, utilizar verbos de acción claros y reforzar visualmente el CTA principal. Estas prácticas reducen errores de interacción y contribuyen a una experiencia más intuitiva y accesible.

## 2.4 Estado del Arte en Sistemas Inclusivos

La accesibilidad en *Front-End* es un requisito fundamental para evitar exclusión de usuarios con discapacidades. Los estudios recientes destacan que la accesibilidad debe integrarse desde el inicio del proyecto [4] [5].

### 2.4.1 Accesibilidad en Aplicaciones *React*

El uso de *frameworks* como *React* requiere atención especial a la accesibilidad, ya que la interfaz se construye mediante JSX y componentes reutilizables.

La herramienta clave en el contexto actual que nos permitirá medir rendimiento y accesibilidad es Google *Lighthouse*, una herramienta gratuita y de código abierto de Google que permite auditar y mejorar la calidad de las páginas web [6].

### 2.4.2 Beneficios y Normativas

La accesibilidad mejora la experiencia de usuarios con discapacidades visuales, cognitivas o motoras y también beneficia a todos los usuarios mediante:

- Mejora de contraste, navegación clara y etiquetas descriptivas [7].
- Reducción de barreras tecnológicas y cumplimiento legal [5].
- Cumplimiento de estándares WCAG, accesibilidad cognitiva y atributos ARIA.

Algunos de los criterios inclusivos más relevantes son:

- 1.1.1 Contenido no textual: Todo elemento visual (iconos, botones, imágenes) debe tener texto alternativo significativo.
- 1.3.1 Información y relaciones: La estructura semántica (encabezados, listas, formularios) debe mantenerse en el código, permitiendo la comprensión mediante lectores de pantalla.
- 1.4.3 Contraste mínimo: El texto y los elementos visuales deben tener un contraste mínimo de 4.5:1 respecto al fondo. [8]
- 1.4.4 Tamaño del texto: El contenido debe ser legible y escalable hasta un 200 % sin pérdida de funcionalidad.
- 3.1.5 Nivel de lectura: El lenguaje debe ser claro y comprensible para usuarios con baja alfabetización digital. [8]

## 3 Implementación Plataforma Paciente

### 3.1 Estructura General del Proyecto

Se definió una arquitectura y una estructura de carpetas organizada de manera lógica, permitiendo ordenar y separar correctamente los distintos componentes, la estructura de directorios presentada corresponde a un entorno de desarrollo basado en contenedores para la aplicación "SuHorita", compuesta por un *frontend*, un *backend* y un *reverse proxy* con soporte para certificados TLS. Cada carpeta y archivo cumple una función específica dentro del despliegue, además el uso de un repositorio separado y manejado como submódulo permite una arquitectura desacoplada, donde el *frontend* puede evolucionar de forma independiente del *backend*, simplificar pruebas, mantener una estructura limpia y facilitar despliegues diferenciados en entornos de desarrollo, pruebas y producción.

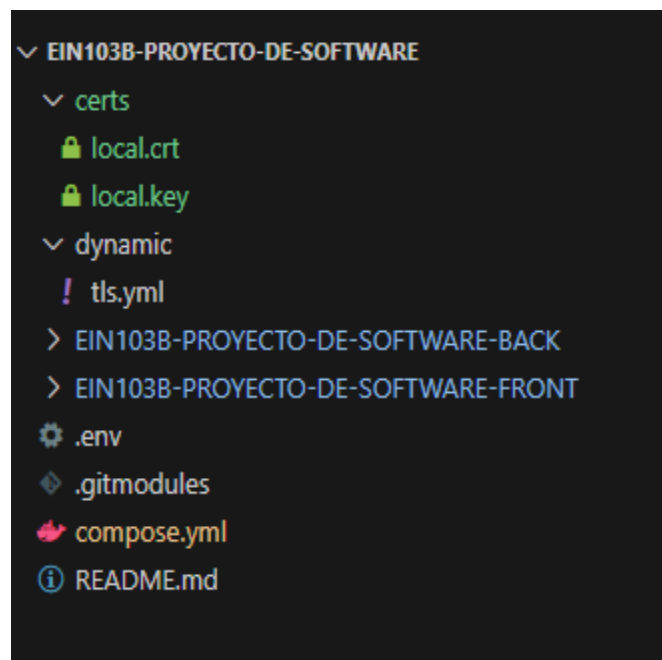


**Ilustración 2: Estructura general de directorios**

Fuente: Elaboración propia

#### 3.1.1 certs/

Este directorio almacena los certificados utilizados para habilitar comunicación segura mediante HTTPS en el entorno local, los archivos son utilizados por el *reverse proxy* para establecer conexiones TLS durante el desarrollo.

- local.crt: Es el certificado público autoafirmado para el dominio local.
- local.key: La llave privada asociada al certificado.

### 3.1.2 dynamic/

Carpeta destinada a archivos de configuración dinámica del *reverse proxy*. Incluye el archivo `tls.yml`, donde se definen parámetros de seguridad TLS, tales como certificados, opciones de cifrado, y políticas de manejo de conexiones seguras. Este archivo permite modificar la configuración sin reiniciar completamente los servicios.

### 3.1.3 EIN103B-PROYECTO-DE-SOFTWARE-BACK/

Este submódulo corresponde al repositorio del *backend* de la aplicación y representa el núcleo funcional del sistema. En él se implementa toda la lógica de negocio, los controladores, la gestión de datos, la integración con servicios externos y los mecanismos de seguridad necesarios para el funcionamiento correcto de la plataforma.

### 3.1.4 EIN103B-PROYECTO-DE-SOFTWARE-FRONT/

Repositorio y submódulo correspondiente al *frontend*. Este componente es el encargado de la experiencia de usuario, la presentación de la información y la interacción directa con el usuario final. Su estructura y funcionamiento serán detallados más adelante, dado que constituyen el foco principal de esta tesina.

### 3.1.5 .env

Este archivo almacena variables de entorno utilizadas por el *docker-compose* y los servicios del proyecto. Permite configurar el entorno sin modificar directamente los archivos del proyecto.

Se suelen incluir: Configuración de puertos, rutas de certificados, claves o *tokens* de desarrollo y la configuración de base de datos.

### 3.1.6 .gitmodules

Nos indica que el proyecto utiliza submódulos de Git. Especifica la ubicación de los repositorios del *frontend* y *backend*, permitiendo mantenerlos como módulos independientes pero vinculados al proyecto principal.

### 3.1.7 compose.yml

Archivo principal de orquestación con *Docker Compose*. Define los servicios (*frontend*, *backend*, *proxy*), sus dependencias, redes internas, volúmenes, montaje de certificados y mapeos de puertos. Permite levantar la arquitectura completa mediante un solo comando (*docker compose up*).

### 3.1.8 README.md

Documento base del proyecto aquí se describen instrucciones de instalación, despliegue, requisitos y otros detalles útiles para el equipo de desarrollo.

## 3.2 Estructura General del *Front-end*

El directorio **EIN103B-PROYECTO-DE-SOFTWARE-FRONT** corresponde al módulo *frontend* de la aplicación y en el que nos vamos a enfocar en esta tesina, este se gestiona mediante un entorno contenerizado. Además, la estructura está organizada de manera modular para facilitar el desarrollo, el mantenimiento y el despliegue. A continuación, se describen los elementos más relevantes:

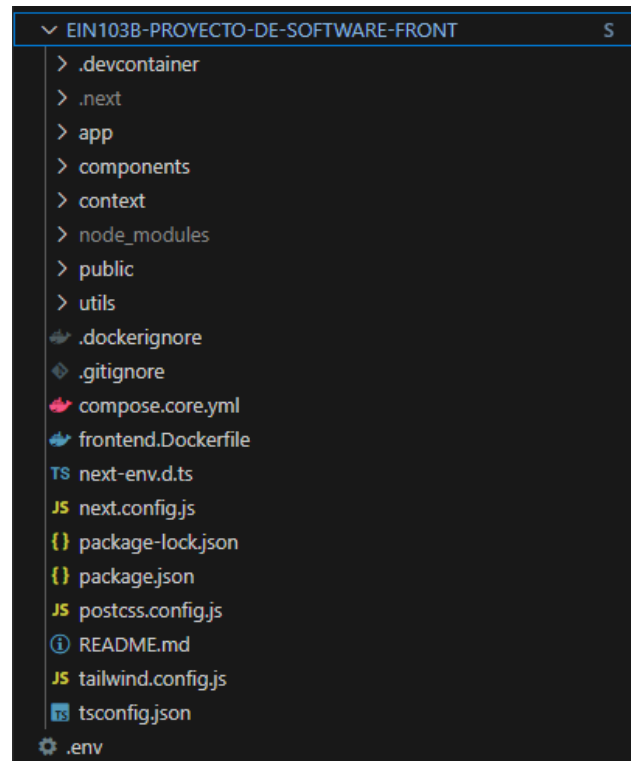


**Ilustración 3: Estructura de directorios *Front-end***

Fuente: Elaboración propia

### 3.2.1 Directorios principales

#### 3.2.1.1 .devcontainer/

Este archivo define la configuración del entorno de desarrollo en contenedores, especifica que el entorno utilizará la imagen `node:18`, ejecutará `npm install` automáticamente después de crearse el contenedor, montará el proyecto local dentro del contenedor en la ruta `/workspace`, configurará el `shell` integrado como `Bash` y establecerá al usuario del contenedor como `node`. En conjunto, permite trabajar con un entorno Node.js aislado, reproducible y listo para desarrollo sin necesidad de instalar Node localmente.

#### 3.2.1.2 .next/

Esta carpeta es generada automáticamente por Next.js. Contiene artefactos internos como: Código compilado del *frontend*, archivos de caché, optimización de imágenes y páginas pre-renderizadas. Cabe destacar que no debe modificarse manualmente.

### 3.2.1.3 context/

Contiene los archivos para la gestión de estados globales en *React* mediante un *StepsProvider*, encargado de administrar toda la información del flujo de pasos dentro de la aplicación, incluyendo qué pasos han sido validados, los datos seleccionados en el proceso y la información del paciente cuando es necesario registrarlo. Además, utiliza *sessionStorage* para mantener los datos del paciente incluso si el usuario recarga la página, y expone funciones para actualizar cada parte del estado sin necesidad de pasar *props* entre componentes. A través del *hook* personalizado *useSteps*, cualquier componente puede acceder fácilmente a este estado centralizado, permitiendo construir un flujo de navegación más ordenado, consistente y escalable.

### 3.2.1.4 node\_modules/

Este es un directorio generado automáticamente que contiene todas las librerías y dependencias del proyecto (*React*, *Next.js*, *TailwindCSS*, etc.), representa el *stack* tecnológico seleccionado para el desarrollo.

### 3.2.1.5 components/

Almacena los bloques de construcción o componentes reutilizables de la interfaz como Navbar, Botones, formularios, etc, estos son esenciales para la reutilización de código y para implementar los criterios de accesibilidad de forma consistente en todos los módulos, este punto será abordado en el punto 3.3 Arquitectura de los componentes.

### 3.2.1.6 public/

Esta carpeta contiene los recursos estáticos como imágenes, íconos y archivos accesibles directamente desde el navegador. Dentro también podemos encontrar otra carpeta llamada */fonts* con la fuente *OpenDyslexic* [9], que es la que será usada en la plataforma, además podemos encontrar el *manifest.json* necesario para la implementación como PWA.



**Ilustración 4: Logo SuHorita**

Fuente: Elaboración Grupal

### 3.2.1.7 utils/

Contiene funciones auxiliares que son utilizadas por distintas partes del proyecto. Contiene el archivo *validators.js*, que implementa las validaciones automáticas de datos esenciales (RUN) para prevenir registros incompletos.

### 3.2.1.8 app/

Este es el directorio raíz del enrutamiento. Aquí se organizan las páginas, *layouts*, *endpoints* y lógica de navegación del *frontend*. Cada subdirectorío dentro de *app/* representa una ruta o un módulo funcional de la aplicación del paciente.

**Tabla 1: Detalle de la Estructura de archivos en App**

Esta tabla busca describir cómo se organiza el enrutamiento y los módulos funcionales del frontend, relacionando cada ruta con su rol dentro del flujo de agendamiento del paciente y con los objetivos específicos de la tesina.

| Carpetas dentro de /app | Descripción Funcional                                                                                                          | Objetivo de la Tesina Relacionado                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| /pedir-hora1            | Contiene las pantallas para la solicitud inicial de la hora, es el primer paso donde se ingresa el Rut del paciente.           | Desarrollo del flujo de agendamiento (OE2).                                                        |
| /seleccionar-servicio2  | Interfaz para la selección de la especialidad médica requerida, es el segundo paso del flujo para agendar hora.                | Desarrollo del flujo de agendamiento (OE2).                                                        |
| /confirmar-reserva3     | Presenta el resumen detallado para que el paciente revise y confirme lo seleccionado, es el tercer paso y ultimo para agendar. | Desarrollo del flujo de agendamiento (OE2).                                                        |
| /Reserva-exitosa        | Pantalla de confirmación final de la cita, aquí se muestra el detalle de la hora solicitada.                                   | Proporciona <i>feedback</i> claro y necesario para el usuario (WCAG 2.4.4 - Propósito del enlace). |
| /registro-paciente      | Flujo para la inscripción de nuevos pacientes que no están en el sistema.                                                      | Recopilación eficiente y confiable de datos personales.                                            |
| /checkeo_rut            | Maneja la pantalla de identificación del paciente mediante su RUN, implementando la validación de datos.                       | Implementación de validaciones automáticas para evitar duplicidad/registros incompletos.           |
| /confirmar-hora         | Módulo para que el paciente confirme su reserva 24 horas antes de su hora agendada.                                            | Desarrollo del flujo de confirmación (OE2).                                                        |
| /anular-hora            | Módulo para que el paciente pueda cancelar una cita previamente agendada.                                                      | Desarrollo del flujo de cancelación (OE2).                                                         |

**Tabla 2: Archivos de Configuración y Estilos en App**

Se explica el propósito de los principales archivos de configuración y estilos en App, destacando cómo soportan la estructura visual, la accesibilidad y la gestión de estado de la aplicación.

| Archivo                  | Tipo                                           | Función en el Proyecto                                                                                                                                                   |
|--------------------------|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>layout.tsx</b>        | Componente de <i>Layout</i> (TypeScript/React) | Define la estructura principal de la página (cabecera, pie de página), donde se integran los componentes esenciales como la barra de navegación y los <i>providers</i> . |
| <b>page.tsx</b>          | Componente de Página (TypeScript/React)        | Representa la página de inicio o <i>Home</i> de la aplicación, el punto de entrada para el usuario.                                                                      |
| <b>Providers.tsx</b>     | Componente de Provisión (TypeScript/React)     | Envuelve la aplicación para proporcionar el estado global a todos los demás componentes. Clave para el OE3.                                                              |
| <b>accesibilidad.css</b> | Hoja de Estilos (CSS)                          | Contiene las reglas CSS específicas para la adaptación de la interfaz, como el contraste, estilos y tamaño para fuentes.                                                 |
| <b>globals.css</b>       | Hoja de Estilos (CSS)                          | Define los estilos base de la aplicación, tipografía general, y la configuración de Tailwind CSS.                                                                        |

### 3.2.2 Archivos de configuración esenciales

**Tabla 3: Archivos de configuración**

Se identifican los archivos clave de configuración del proyecto y su rol técnico, evidenciando el stack tecnológico utilizado, la calidad del desarrollo y la preparación para despliegue y escalabilidad.

| Archivo             | Descripción                                                                                     | Rol en el Proyecto                                                    |
|---------------------|-------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <b>package.json</b> | Define los metadatos del proyecto y lista todas las dependencias y <i>scripts</i> de ejecución. | Confirma el uso del <i>stack React, Next.js</i> y <i>TypeScript</i> . |

|                                               |                                                                                             |                                                                                                                                           |
|-----------------------------------------------|---------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <b>tsconfig.json</b>                          | Archivo de configuración para el compilador de <i>TypeScript</i> .                          | Habilita el tipado estático del proyecto, reduciendo errores en tiempo de desarrollo y facilitando la escalabilidad.                      |
| <b>next.config.js</b>                         | Archivo de configuración de Next.js.                                                        | Permite optimizar la aplicación (por ejemplo, para PWA o para <i>performance</i> ).                                                       |
| <b>tailwind.config.js / postcss.config.js</b> | Archivos de configuración de Tailwind CSS.                                                  | Confirman el uso del <i>framework</i> de estilos, esencial para implementar diseños responsivos y con alto contraste de manera eficiente. |
| <b>frontend.Dockerfile</b>                    | Instrucciones para la creación de la imagen del contenedor Docker para el <i>frontend</i> . | Indica que el proyecto está preparado para su despliegue en contenedores, garantizando un entorno consistente y reproducible.             |

### 3.3 Arquitectura de los componentes

El sistema fue desarrollado utilizando Next.js como *framework* principal, lo que permitió contar desde el inicio con una arquitectura organizada y basada en componentes reutilizables. Esta estructura favorece la separación entre lógica, presentación y comportamiento, facilitando la mantenibilidad del código y permitiendo una evolución controlada del proyecto.

Cada vista del sistema está construida a partir de componentes independientes, los cuales encapsulan funcionalidades específicas como navegación, formularios, botones, barras de progreso y elementos estructurales. Esta modularidad permite componer interfaces complejas mediante piezas pequeñas, coherentes y fácilmente *testeables*.

La figura presentada ilustra la arquitectura general de navegación y la interacción entre componentes dentro de las distintas interfaces del sistema: interfaz de inicio, reserva de hora, confirmación, cancelación y registro. Cada etapa del flujo está compuesta por componentes que se integran entre sí para construir la experiencia completa del usuario.

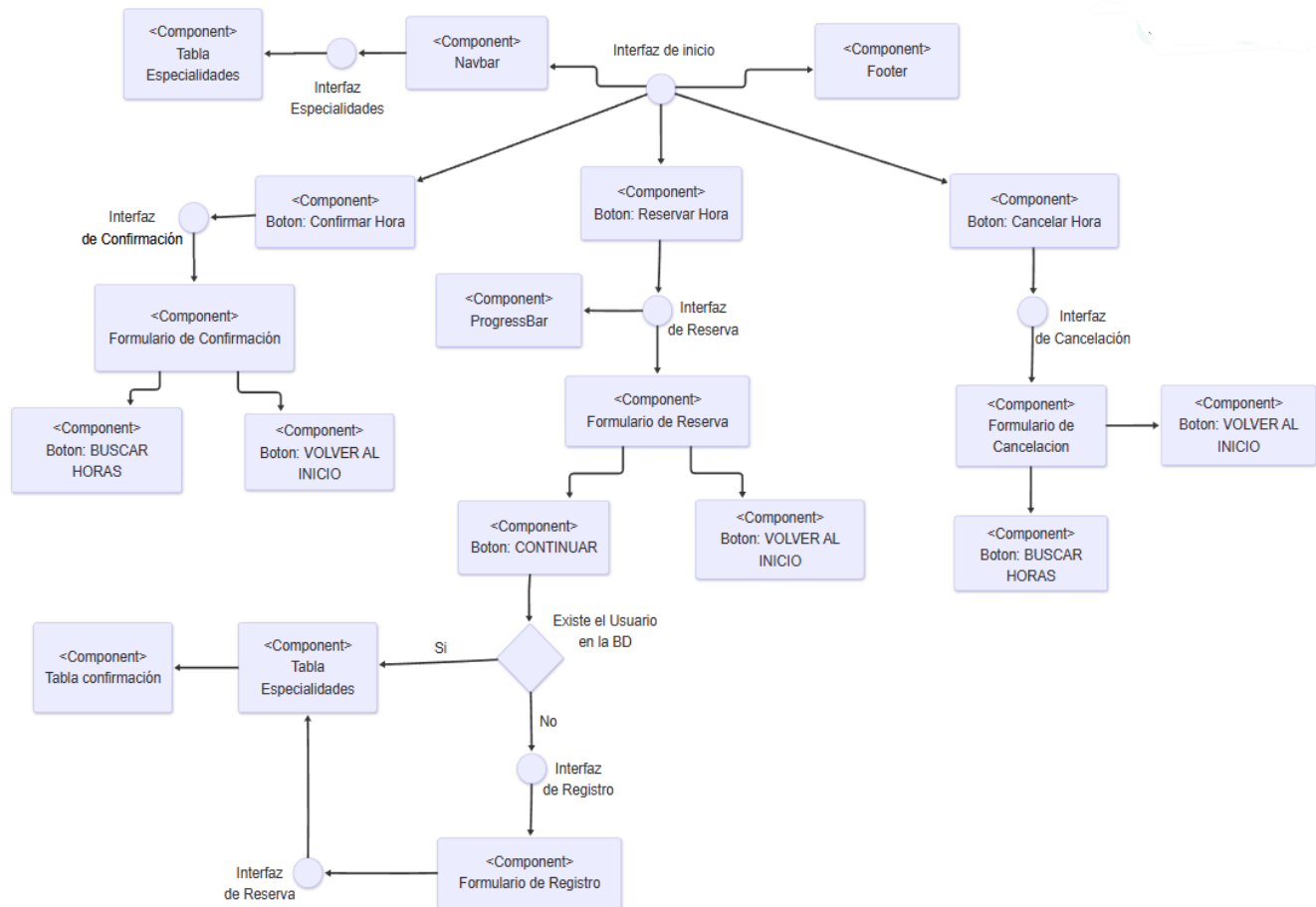


**Figura 2: Componentes**

Fuente: Elaboración propia

### 3.3.1 Reutilización de componentes

La arquitectura basada en componentes implementada en el *frontend* permite maximizar la reutilización de elementos, disminuir la duplicación de código y asegurar una experiencia de usuario consistente en todas las etapas del sistema. Esta estrategia, alineada con las prácticas recomendadas en React y Next.js, facilita la mantenibilidad del proyecto, mejora la escalabilidad y permite incorporar nuevas funcionalidades sin afectar la estructura existente.

Dentro del directorio `components/`, se agrupan los elementos reutilizables que forman la base de la interfaz gráfica y la lógica interactiva del sistema. Estos componentes no son exclusivos de una vista específica, sino que fueron diseñados para integrarse en múltiples pantallas según sea necesario. A continuación, se describe con mayor detalle el rol y la reutilización de cada uno:

#### ActionButtons.jsx

Contiene un conjunto de botones que encapsulan acciones frecuentes dentro de la aplicación (confirmar, cancelar, avanzar, agendar, etc.). Centralizar estos elementos permite mantener coherencia visual y funcional a lo largo de todas las interfaces.

## **Footer.jsx**

Componente correspondiente al pie de página de la aplicación. Incluye elementos comunes como información institucional, enlaces de navegación secundaria y políticas de uso. Se renderiza en las diferentes vistas como parte del *layout* general.

## **Navbar.jsx**

Representa la barra de navegación superior. Gestiona elementos clave como menú principal, enlaces a secciones del sistema, logotipo y opciones de usuario. Es un componente esencial para la estructura y navegación global del *frontend*.

## **ProgressBar.jsx**

Implementa una barra de progreso utilizada para indicar el avance en el proceso o etapas dentro del flujo de la aplicación. En este caso el flujo de reserva de hora médica, esta le entrega *feedback* visual al usuario, mejorando la experiencia y comprensión del estado actual del proceso.

## **FormularioReserva.jsx**

Encapsula toda la lógica y campos necesarios para la selección de especialidad, fecha y hora. Se integra con varios botones definidos en *ActionButtons.jsx*.

## **FormularioConfirmación.jsx**

Reutiliza componentes comunes como botones, tabla de especialidades y lógica compartida para mostrar la información ya seleccionada por el usuario.

## **FormularioCancelación.jsx**

Comparte estructura y lógica con los otros formularios, sobrescribiendo solo los elementos específicos del proceso de cancelación.

## **TablaEspecialidades.jsx**

Permite mostrar las especialidades médicas u opciones disponibles al usuario, es utilizada en la Navbar donde podemos apretar un botón de especialidades y nos lleva a un apartado con estas y también al momento de seleccionar servicio en el flujo de solicitar hora médica.

## **TablaConfirmación.jsx**

Se utiliza para mostrar de manera uniforme los datos finales de la reserva antes de enviarlos.

## **FormularioRegistro.jsx**

Implementado como un componente independiente reutilizado en cualquier flujo donde sea necesario registrar o actualizar los datos del paciente.

### 3.4 Diseño de la Interfaz

El diseño de la interfaz se desarrolló bajo un enfoque centrado en accesibilidad, inclusión y cumplimiento estricto de las pautas WCAG 2.2, incorporando diferentes modos visuales y de interacción para usuarios con diversas necesidades cognitivas, visuales y motrices. La aplicación implementa un sistema de estilos avanzado que adapta tipografías, colores, tamaños de elementos interactivos y comportamientos de navegación según el perfil del usuario.

Desde el inicio se adoptó un enfoque de DCU, donde las interfaces se construyen con retroalimentación continua de usuarios reales y pruebas de usabilidad iterativas. Esto asegura que cada módulo sea intuitivo y accesible desde el primer *sprint*, incorporando criterios de accesibilidad de manera transversal durante todo el ciclo de desarrollo, en lugar de considerarlos únicamente como ajustes finales.

#### 3.4.1 Tipografía

Se integró la familia tipográfica OpenDyslexic [9], específicamente diseñada para mejorar la lectura de personas con dislexia. Esta fuente se aplica de manera progresiva, permitiendo al usuario activar o desactivar su uso según preferencia personal. Su incorporación contribuye a disminuir la confusión entre caracteres y facilita la lectura continua.

#### 3.4.2 Ajustes de Accesibilidad

La interfaz incorpora dos modos principales que pueden activarse individualmente según las necesidades del usuario:

##### a. Ajustes de Visión (baja visión / ceguera parcial)

- Aumenta el tamaño del texto y el interlineado.
- Potencia el contraste mediante colores validados con ratios superiores a 7:1.
- Ajusta títulos, párrafos y elementos interactivos para minimizar el esfuerzo visual.
- Mejora el estado *hover/focus* para hacerlo más perceptible.

##### b. Ajustes de Motricidad

- Incrementa considerablemente el tamaño de botones y texto.
- Amplía las áreas de interacción táctil.
- Mantiene lineamientos estrictos del tamaño mínimo ( $\geq 60\text{px}$  en la mayoría de los controles).

Está diseñado para usuarios con dificultades de movilidad, temblores o artritis.

Se añadieron colores alternativos que no dependen exclusivamente del color para transmitir información. Se incorporaron iconos semánticos ( $\checkmark$ ,  $\Delta$ , **i**) como indicadores de éxito, error o información. Además, se definieron paletas de colores amigables, utilizando azules, rojos y verdes.

### 3.4.3 Adaptaciones Basadas en WCAG 2.2

Con el objetivo de asegurar una experiencia verdaderamente accesible, se realizaron ajustes globales en toda la interfaz relacionados con las pautas de la WCAG, un estándar internacional creado por el W3C (*World Wide Web Consortium*) para asegurar que sitios *web*, aplicaciones y contenidos digitales sean accesibles para todas las personas, incluyendo:

- Personas con discapacidad visual (ceguera, baja visión)
- Personas con discapacidad motriz
- Personas con discapacidad cognitiva
- Adultos mayores
- Usuarios que dependen de tecnologías de asistencia (lectores de pantalla, dispositivos alternativos de entrada, *switches*, entre otros)

**Tabla 4: Pautas WCAG aplicadas**

| Pauta WCAG 2.2 | Nombre de la pauta                          | Cómo se cumple en el código                                                                |
|----------------|---------------------------------------------|--------------------------------------------------------------------------------------------|
| 1.4.3 - 1.4.6  | Contraste mínimo / Contraste mejorado       | <i>.high-contrast usa colores con contraste 7:1.</i>                                       |
| 1.4.4          | Redimensionamiento del texto                | <i>html {font-size: 16px;} permite escalar hasta 200% sin romper layout.</i>               |
| 1.4.10         | Reflujo responsivo                          | <i>Media query &lt; 320px evita scroll horizontal y ajusta contenido.</i>                  |
| 1.4.12         | Espaciado de texto                          | <i>line-height: 1.5, margin-bottom: 2em, letter-spacing: 0.12em, word-spacing: 0.16em.</i> |
| 1.4.13         | Contenido al pasar el cursor o enfocar      | <i>.tooltip:hover .tooltiptext { visibility: visible; }.</i>                               |
| 2.1.1 - 2.1.2  | Acceso por teclado / Sin trampas de teclado | <i>Estilos de: focus visibles en botones, inputs y enlaces.</i>                            |
| 2.3.3          | Animación a pedido (reducir movimiento)     | <i>.animation-mode elimina todas las animaciones.</i>                                      |
| 2.5.5 / 2.5.7  | Tamaño objetivo mínimo (44×44px)            | <i>button, a, input {min-height: 44px; min-width: 44px; }.</i>                             |
| 3.3.1          | Identificación de errores                   | <i>.error-indicator::before { content: "⚠"; }.</i>                                         |
| 3.3.3          | Sugerencias ante errores                    | <i>Clases .error-indicator, .info-indicator, .success-indicator.</i>                       |
| 1.4.1          | Uso del color — no depender solo de color   | <i>Se agregan iconos: success-indicator::before, error-indicator::before.</i>              |

|       |                                                       |                                                                                               |
|-------|-------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| 2.4.7 | Focus visible                                         | <i>outline: 2px solid #005fcc; outline-offset: 2px;.</i>                                      |
| 4.1.2 | Nombre, rol, valor<br>(para componentes interactivos) | Estilos compatibles con atributos ARIA<br>(en los archivos JSX<br>mencionados anteriormente). |

Estos estándares se aplican tanto a botones como a enlaces, formularios y controles interactivos en todas las pantallas.

### 3.4.4 Estructura Semántica y Navegación

Para garantizar navegabilidad por teclado y compatibilidad con lectores de pantalla se integraron los atributos ARIA (*Accessible Rich Internet Applications*), un conjunto de atributos diseñados para mejorar la accesibilidad de interfaces *web*, especialmente para personas que utilizan lectores de pantalla u otras tecnologías de asistencia, ya que permiten describir elementos, roles y estados que no son evidentes para tecnologías de asistencia cuando la interfaz usa componentes complejos (menús dinámicos, modales, *sliders*, *tabs*, etc.).

- Se integraron roles ARIA (*main*, *banner*, *region*, *menubar*, *progressbar*, *radiogroup*).
- Se añadieron descripciones *aria-label* y *aria-describedby* para todos los campos.
- Mensajes dinámicos utilizan *aria-live="polite"* para anunciar cambios.
- Se implementó navegación por teclado con flechas, *Enter*, *Space*, *Home* y *End*.

Esto permite un recorrido claro, predecible y comprensible incluso sin apoyo visual.

### 3.4.5 Diseño responsivo

En dispositivos móviles se aumentó el tamaño de texto, botones y enlaces para mejorar la interacción táctil y evitar el *scroll* horizontal, cumpliendo con WCAG 1.4.10.

Cada control mantiene una dimensión mínima que garantiza que usuarios con dificultades motrices puedan utilizar la aplicación de forma cómoda.

## 3.5 Diseño de accesibilidad e inclusión

Este apartado complementa el diseño de la interfaz descrito en la sección 3.4, profundizando en la accesibilidad e inclusión como elementos estructurales del *Front-End* y no únicamente como características visuales. A diferencia del diseño de la interfaz, que aborda la presentación y experiencia general del usuario, esta sección se centra en cómo las barreras identificadas en el Capítulo 2 fueron abordadas mediante decisiones técnicas concretas, integradas directamente en la arquitectura del sistema.

Las estrategias aquí descritas se orientan a garantizar un acceso equitativo al proceso de agendamiento digital de horas médicas, considerando barreras visuales, cognitivas, tecnológicas y de usabilidad, y asegurando su mitigación mediante componentes reutilizables, estilos globales, validaciones automáticas y manejo adecuado de estados de la interfaz.

### 3.5.1 Estrategias de accesibilidad implementadas en el Front-End

Las estrategias de accesibilidad implementadas en SuHorita fueron definidas a partir del conjunto de barreras priorizadas en el Capítulo 2 y se materializan mediante decisiones de diseño UX/UI y su correspondiente implementación técnica en el *Front-End*. Cada estrategia fue pensada para ser verificable, medible y reutilizable a lo largo de todo el flujo de la aplicación.

**Tabla 5: Estrategias de Accesibilidad**

Presenta una síntesis de las principales barreras abordadas, las decisiones de diseño adoptadas, su implementación técnica concreta y los mecanismos de verificación utilizados para asegurar su correcto funcionamiento.

| Barrera                                          | Decisión de diseño (UX/UI)                                  | Implementación técnica                                                                                                                            | Verificación (métrica)                                                      | Pantallas / Componentes                         |
|--------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|-------------------------------------------------|
| <b>Daltonismo / baja discriminación de color</b> | No depender exclusivamente del color para comunicar estados | Uso de iconos semánticos (✓, △, i) junto a texto; clases .success, .error y .info definidas en accesibilidad.css; paleta con contraste $\geq 7:1$ | <i>Lighthouse Accessibility</i> $\geq 90$ ;                                 | Botones, mensajes de validación, confirmaciones |
| <b>Dislexia / dificultad lectora</b>             | Tipografía especializada y control de lectura               | Integración de fuente OpenDyslexic; ajustes de <i>line-height</i> , <i>letter-spacing</i> y ancho de línea mediante clases CSS                    | Prueba de comprensión de instrucciones $\geq 85\%$                          | Formularios y pantallas informativas            |
| <b>Baja alfabetización digital</b>               | Flujo guiado y simplificado                                 | Flujo paso a paso con ProgressBar.jsx; CTA único por pantalla; Etiquetas ARIA; textos breves y lenguaje simple                                    | Tasa de éxito en tarea "Agendar hora" $\geq 85\%$ ; tiempo $\leq 2$ minutos | Flujo completo de agendamiento                  |
| <b>Sobrecarga cognitiva</b>                      | Reducción de opciones simultáneas                           | Máximo 2–3 opciones por vista; confirmaciones explícitas antes de acciones críticas; prevención de errores                                        | Errores críticos por tarea $\leq 1$ ; tasa de abandono $\leq 25\%$          | Selección de servicio, confirmación             |

|                                   |                                   |                                                                                                                         |                                                           |                                               |
|-----------------------------------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------|
| <b>Visión reducida</b>            | Escalabilidad y alto contraste    | Unidades rem; control de tamaño de fuente (A-/A+); modo .high-contrast; <i>zoom</i> hasta 200% sin romper <i>layout</i> | Revisión manual de legibilidad y <i>layout</i>            | Tipografía global, <i>cards</i> , formularios |
| <b>Dispositivos de asistencia</b> | Navegación completamente operable | Orden lógico de tabulación; foco visible; roles y atributos ARIA; soporte <i>Enter</i> , <i>Space</i> y flechas         | <i>Checklist</i> : flujo completo usable solo con teclado | Todo el flujo crítico                         |

### 3.6 Presentación de la Interfaz

Este apartado describe las pantallas principales que conforman la interfaz de usuario del sistema de reserva de horas médicas “*SuHorita*”. Cada vista fue diseñada bajo criterios de accesibilidad, claridad y simplicidad, priorizando la experiencia de personas con distintos niveles de alfabetización digital. La interfaz integra componentes consistentes como la barra de progreso, Navbar y Footer, y presenta formularios guiados que validan en tiempo real la información ingresada por el usuario, como el tipo de documento y su identificador correspondiente. El diseño se centra en la usabilidad: elementos grandes y legibles, retroalimentación inmediata, navegación asistida y una organización visual limpia que facilita la comprensión de cada paso del proceso. En su conjunto, estas pantallas permiten avanzar de forma clara, segura y ordenada a través del flujo de solicitud de atención médica.

Por temas de espacio no todas las imágenes de las interfaces se adjuntaron completas.

### 3.6.1 Pantalla principal SuHorita

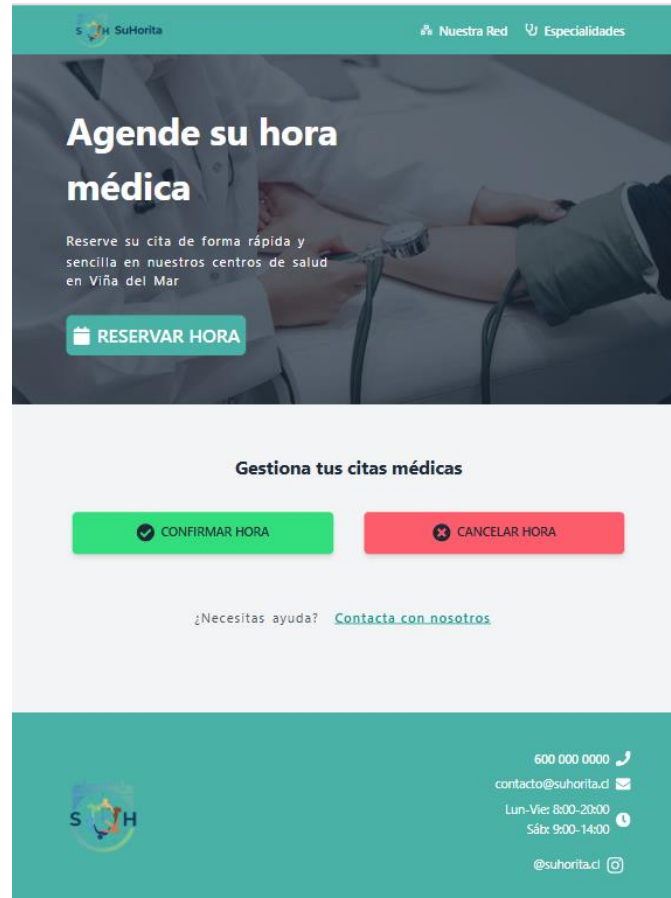


**Ilustración 5: Pantalla principal SuHorita**

Fuente: Elaboración Grupal

Esta Ilustración corresponde a la pantalla de inicio de SuHorita, actúa como punto de entrada a todo el flujo de reserva de horas médicas. La interfaz presenta una estructura basada en diseño responsivo mediante Tailwind CSS e incorpora elementos globales como *Navbar*, *Footer* y componentes (*ActionButtons*). La sección principal está compuesta por un encabezado visual de gran formato, construido con una imagen de fondo y un *overlay* semitransparente, lo que permite destacar el mensaje central de la plataforma. Dentro de este *banner* se despliega un título jerárquico y un subtítulo descriptivo que explican la función del sistema, acompañados de un botón principal de llamada a la acción ("Reservar Hora"), diseñado con un alto nivel de accesibilidad: tamaños mínimos táctiles, soporte para lector de pantalla mediante atributos ARIA y retroalimentación visual en estados de foco y *hover*. Además, se integra un ícono contextual que refuerza la función del botón sin interferir con la legibilidad. En conjunto, esta pantalla cumple la función de guía inicial para el usuario, ofreciendo un punto de navegación claro, estético y consistente con la identidad visual del sistema.

### 3.6.2 Reserva de hora médica paso 1

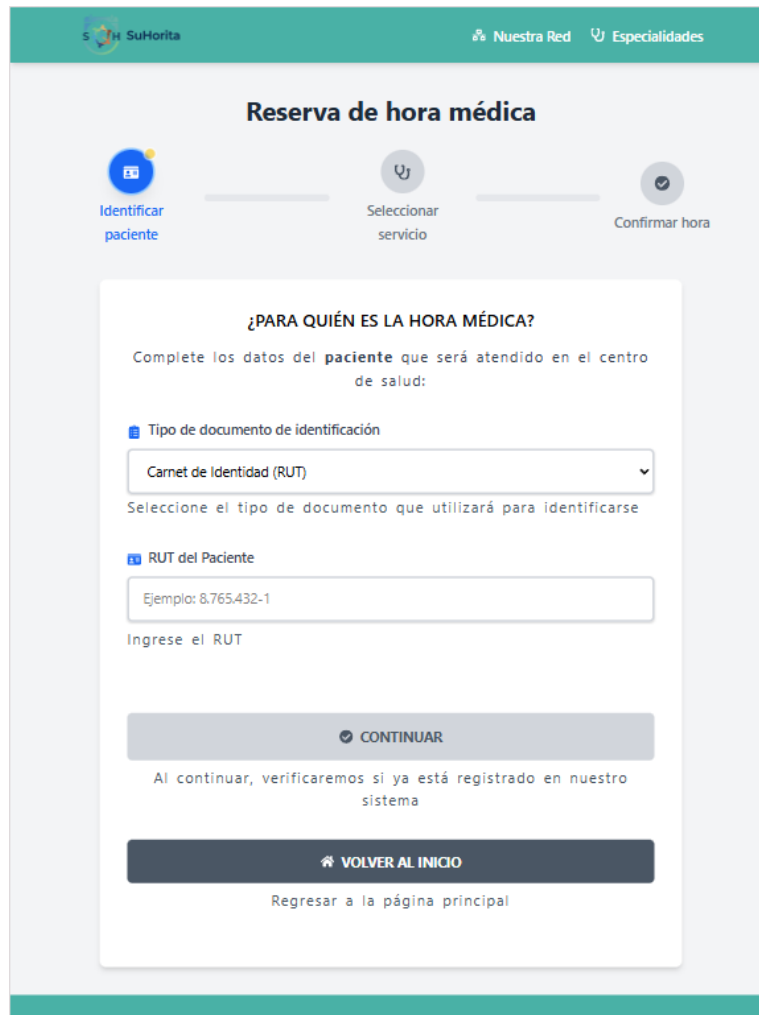


**Ilustración 6: Reserva de hora médica paso 1**

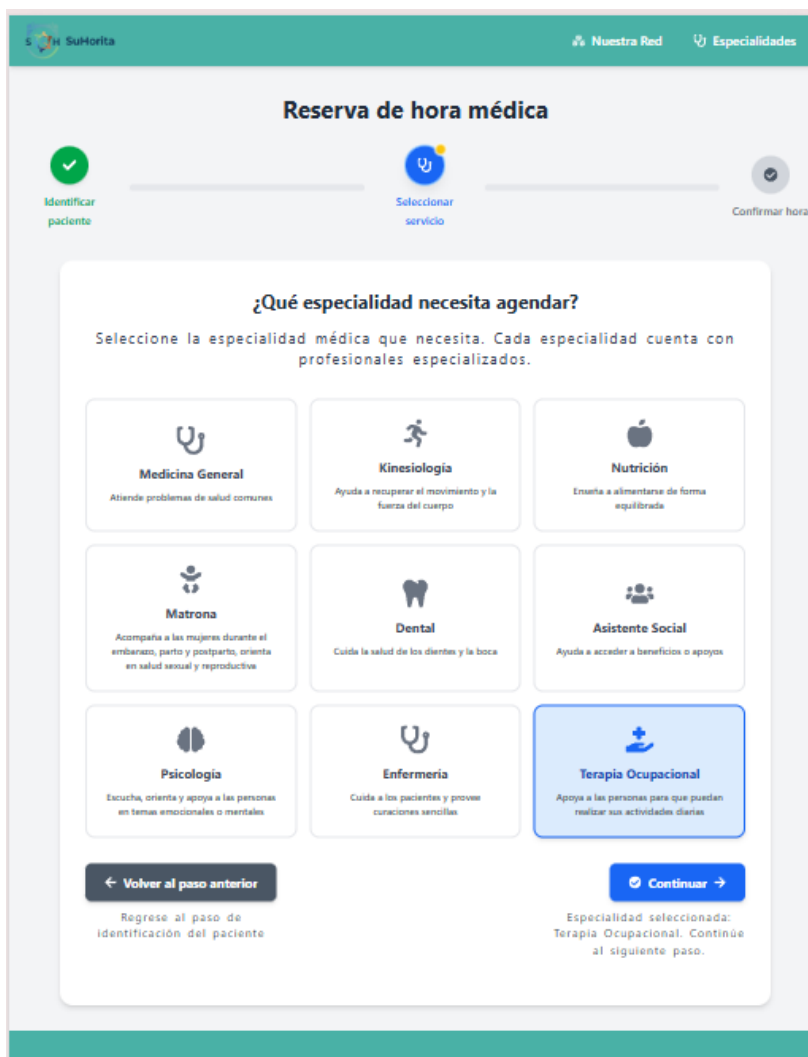
Fuente: Elaboración Grupal

La interfaz presenta un formulario accesible y guiado, donde el usuario selecciona el tipo de documento (RUT, Pasaporte o NIP) y luego ingresa el identificador correspondiente, aplicando validación dinámica según el formato requerido y entregando retroalimentación inmediata con mensajes de error, descripciones ARIA y cambios visuales en el campo. El diseño incorpora principios de DCU, con textos claros, iconografía contextual, controles suficientemente grandes y soporte para eventos de teclado, lo que facilita su uso por personas con distintos niveles de alfabetización digital. Además, el botón de avance gestiona estados de carga, deshabilitación y accesibilidad, asegurando que el usuario solo pueda continuar cuando el documento es válido. La interfaz organiza la información mediante encabezados jerárquicos, contenedores con sombra y espaciado consistente, ofreciendo una experiencia limpia, ordenada y coherente con los lineamientos visuales del sistema.

La lógica detrás gestiona la validación del documento de identidad del paciente, aplicando formatos y reglas específicas según el tipo seleccionado y actualizando el estado de validez en tiempo real. Utiliza el contexto global (*StepsContext*) para marcar

el paso como completado y almacenar temporalmente los datos del paciente. Al continuar, realiza una consulta al *backend* mediante una petición HEAD para verificar si el paciente ya existe en la base de datos; dependiendo de la respuesta, redirige automáticamente al formulario de registro o al módulo de selección de servicio. Además, incorpora controles de accesibilidad, eventos de teclado (Enter), retroalimentación visual mediante estados de carga y componentes compartidos como Navbar, Footer y la barra de progreso.

### 3.6.3 Reserva de hora médica paso 2



**Reserva de hora médica**

Identificar paciente Seleccionar servicio Confirmar hora

**¿Qué especialidad necesita agendar?**

Seleccione la especialidad médica que necesita. Cada especialidad cuenta con profesionales especializados.

|                                                                                                                                                                                                                |                                                                                                                                                                    |                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <br><b>Medicina General</b><br>Atiende problemas de salud comunes                                                             | <br><b>Kinesiología</b><br>Ayuda a recuperar el movimiento y la fuerza del cuerpo | <br><b>Nutrición</b><br>Enseña a alimentarse de forma equilibrada                                         |
| <br><b>Matrona</b><br>Acompaña a las mujeres durante el embarazo, parto y postparto, orienta en salud sexual y reproductiva | <br><b>Dental</b><br>Cuida la salud de los dientes y la boca                    | <br><b>Asistente Social</b><br>Ayuda a acceder a beneficios o apoyos                                    |
| <br><b>Psicología</b><br>Escucha, orienta y apoya a las personas en temas emocionales o mentales                            | <br><b>Enfermería</b><br>Cuida a los pacientes y provee curaciones sencillas    | <br><b>Terapia Ocupacional</b><br>Apoya a las personas para que puedan realizar sus actividades diarias |

[← Volver al paso anterior](#) [Continuar →](#)

Regrese al paso de identificación del paciente

Especialidad seleccionada: Terapia Ocupacional. Continúe al siguiente paso.

**Ilustración 7: Reserva de hora médica paso 2**

Fuente: Elaboración Grupal

Esta interfaz permite al usuario seleccionar la especialidad del servicio que desea solicitar. Para ello, combina lógica de negocio, accesibilidad avanzada y comunicación con el *backend*. Al cargarse, el componente obtiene desde la API la lista de especialidades, profesionales y centros de salud, incorporando dinámicamente los íconos asociados mediante un mapeo interno (*iconMap*). Cada especialidad se presenta como una tarjeta interactiva con ícono, nombre y descripción, cambiando de estilo al ser seleccionada, se muestra como un botón accesible que funciona bajo un patrón

*radiogroup*, permitiendo navegación por teclado (flechas, *Enter* y *Space*), manejo de *tabIndex* y mensajes ARIA en tiempo real, lo que garantiza la usabilidad para personas con distintas capacidades.

La selección realizada por el usuario se almacena en el contexto global (*StepsContext*), marcando el paso como válido y guardando datos estructurados que serán utilizados en la pantalla de confirmación. Si el usuario intenta continuar sin elegir una especialidad, el sistema muestra mensajes de error y emite un anuncio accesible para lectores de pantalla. Cuando la selección es válida, el botón de avance habilita la navegación hacia la siguiente etapa del flujo (confirmar hora). El componente también administra estados de carga, errores y retroalimentación inmediata, integrándose de forma fluida con *Navbar*, *Footer* y la barra de progreso para mantener coherencia visual y de navegación dentro del sistema.

### 3.6.4 Reserva de hora médica paso 3

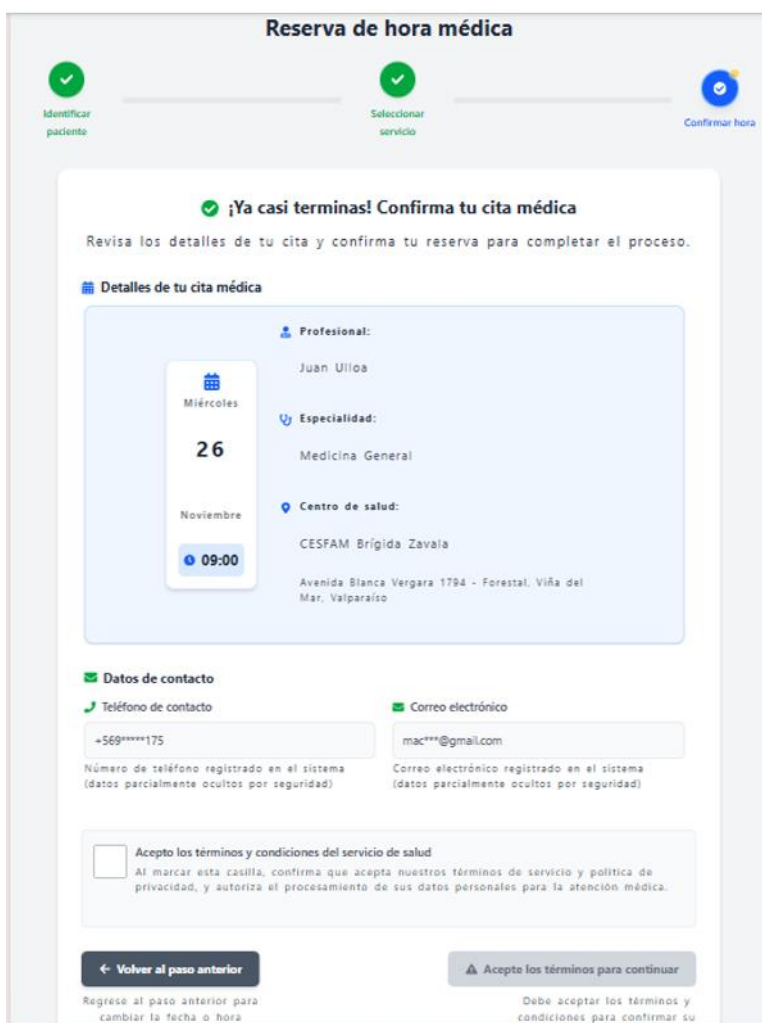


**Ilustración 8: Reserva de hora médica paso 3**

Fuente: Elaboración Grupal

La imagen constituye el último paso para la reserva de hora, donde el usuario revisa visualmente toda la información de su cita antes de confirmar la reserva. Organiza los datos en módulos claros y accesibles: encabezado con barra de progreso, tarjeta con los detalles de la cita (fecha, hora, profesional y centro médico) y sección destinada a los datos de contacto del paciente, presentada con campos de solo lectura y estados de carga para mantener la seguridad. También integra mensajes ARIA en vivo para mejorar la navegación mediante lectores de pantalla, así como mecanismos de validación y control del flujo, como la aceptación obligatoria de los términos antes de habilitar el botón de confirmación. En conjunto, el diseño guía al usuario de manera intuitiva, ofreciendo retroalimentación visual constante y una estructura que facilita validar la información previo a finalizar el proceso.

La lógica gestiona la carga de datos del paciente y la interacción con los servicios del *backend*. A partir de la información almacenada en *sessionStorage*, el módulo valida la estructura de los datos, consulta la API para obtener el teléfono y correo ofuscados del usuario, y recupera la hora asignada según la disponibilidad del profesional. Además, controla estados de carga, errores y anuncios ARIA, asegurando que las actualizaciones importantes sean comunicadas de forma accesible. Una vez que el usuario acepta los términos y confirma la acción, el componente envía la solicitud de reserva mediante una petición POST y, tras recibir la respuesta del servidor, redirige a la pantalla de éxito. En síntesis, este módulo coordina la comunicación con la API, el manejo de estados internos y la lógica final del flujo de reserva.

### 3.6.5 Resumen de la reserva

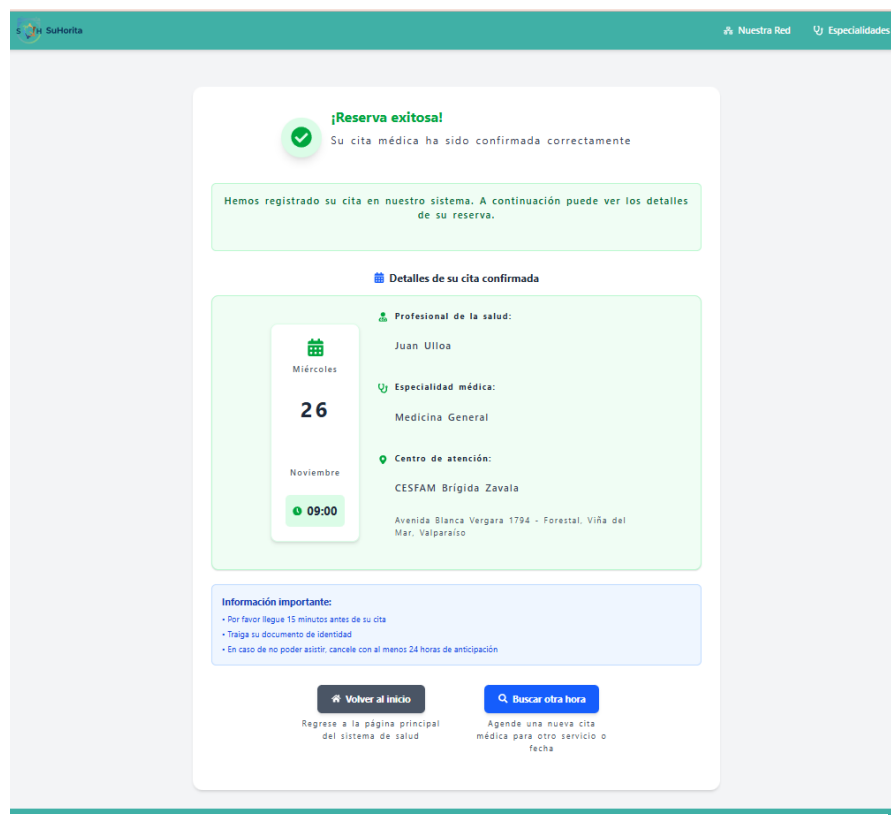


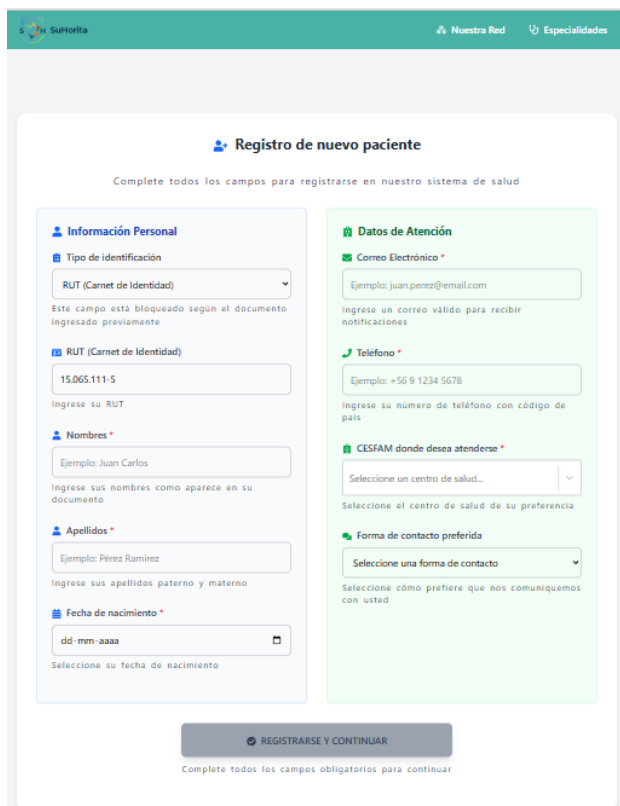
**Ilustración 9: Resumen de la reserva**

Fuente: Elaboración Grupal

La ilustración muestra la interfaz encargada de presentar al usuario el resumen del proceso de agendamiento médico. La vista está diseñada para entregar retroalimentación inmediata y comprensible, utilizando iconos, colores y mensajes claros que refuerzan el éxito de la operación. La estructura visual organiza la información en tarjetas responsivas, optimizadas para dispositivos móviles y escritorio, manteniendo una jerarquía clara entre los elementos relevantes. Además, se incorporan pautas de accesibilidad, como regiones con *aria-live*, roles semánticos y elementos de navegación accesibles, garantizando que la interfaz pueda ser interpretada correctamente por lectores de pantalla. La pantalla también ofrece rutas de continuidad, como volver al inicio o agendar una nueva cita, orientando al usuario de forma intuitiva hacia sus próximas acciones dentro del sistema.

En cuanto a la lógica funcional, es responsable de recuperar, validar y formatear los datos asociados a la reserva médica previamente confirmada. La información se obtiene tanto desde *sessionStorage* como desde el contexto global de pasos (*StepsContext*), permitiendo mantener consistencia entre pantallas y asegurar que los datos persistan tras el flujo de navegación. El código contempla múltiples estructuras posibles para los datos del paciente y de la cita, manejando variaciones en los atributos recibidos y aplicando validaciones que previenen errores de interpretación. Además, incorpora funciones de formateo de fecha y hora ajustadas al huso horario GMT-3, garantizando coherencia temporal dentro del sistema. Durante la carga se gestionan estados como *loading* y mensajes accesibles mediante ARIA, asegurando una experiencia robusta.

### 3.6.6 Registro nuevo paciente



**Registro de nuevo paciente**

Complete todos los campos para registrarse en nuestro sistema de salud

**Información Personal**

**Tipo de identificación**

RUT (Carnet de Identidad)

Este campo está bloqueado según el documento ingresado previamente

**RUT (Carnet de Identidad)**

15.065.111-5

Ingrese su RUT

**Nombres \***

Ejemplo: Juan Carlos

Ingrese sus nombres como aparece en su documento

**Apellidos \***

Ejemplo: Pérez Ramírez

Ingrese sus apellidos paterno y materno

**Fecha de nacimiento \***

dd-mm-aaaa

Seleccione su fecha de nacimiento

**Datos de Atención**

**Correo Electrónico \***

Ejemplo: juan.perez@email.com

Ingrese un correo válido para recibir notificaciones

**Teléfono \***

Ejemplo: +56 9 1234 5678

Ingrese su número de teléfono con código de país

**CESFAM donde desea atenderse \***

Seleccione un centro de salud...

Seleccione el centro de salud de su preferencia

**Forma de contacto preferida**

Seleccione una forma de contacto

Seleccione cómo prefiere que nos comuniquemos con usted

**REGISTRARSE Y CONTINUAR**

Complete todos los campos obligatorios para continuar

**Ilustración 10: Registro nuevo paciente**

Fuente: Elaboración Grupal

Esta ilustración muestra el módulo de registro de pacientes dentro del sistema, estructurado en dos secciones principales: Información personal y datos de atención. El diseño emplea una grilla responsiva que organiza los campos de forma clara y adaptable. Los datos se capturan mediante campos dinámicos que varían según el tipo de documento seleccionado (RUT, Pasaporte o NIP), los cuales se actualizan en tiempo real y pueden bloquearse cuando el sistema detecta información previamente validada.

El formulario incorpora validaciones locales como la verificación del formato del correo electrónico, mensajes de error accesibles, ayudas contextuales mediante *aria-describedby* y restricciones para fechas y campos obligatorios. Para la selección del centro de salud, se utiliza un selector avanzado (*react-select*) que facilita la búsqueda interna y mejora la accesibilidad del control. En la sección de contacto, el usuario puede elegir su canal preferido de comunicación y, en caso de seleccionar "Otro", se despliega un campo adicional para especificar el medio alternativo, manteniendo el formulario limpio y eficiente.

Además, se integra iconografía semántica que refuerza la comprensión visual de cada bloque. El botón de envío permanece deshabilitado hasta que se cumplan todas las validaciones, asegurando un flujo coherente y evitando errores de ingreso.

En cuanto a la lógica, gestiona la captura, validación y envío de los datos del paciente. A través de estados locales, valida campos críticos como el correo electrónico, aplica reglas específicas según el tipo de documento y controla el bloqueo de ciertos inputs cuando existe información previa en el contexto global (*StepsContext*). También consume datos del *backend* para cargar dinámicamente la lista de centros de salud y utiliza *react-select* para facilitar su elección.

Al momento de enviar el formulario (*handleSubmit*), construye el objeto final de registro y realiza una petición POST al *backend*. Si el registro es exitoso, los datos se almacenan en el contexto global, el paso correspondiente se marca como completado y el usuario es redirigido automáticamente al módulo de selección de servicio. A lo largo del proceso, se aplican medidas de accesibilidad mediante mensajes claros, retroalimentación inmediata y controles consistentes.

### 3.5.4 Confirmación de hora médica

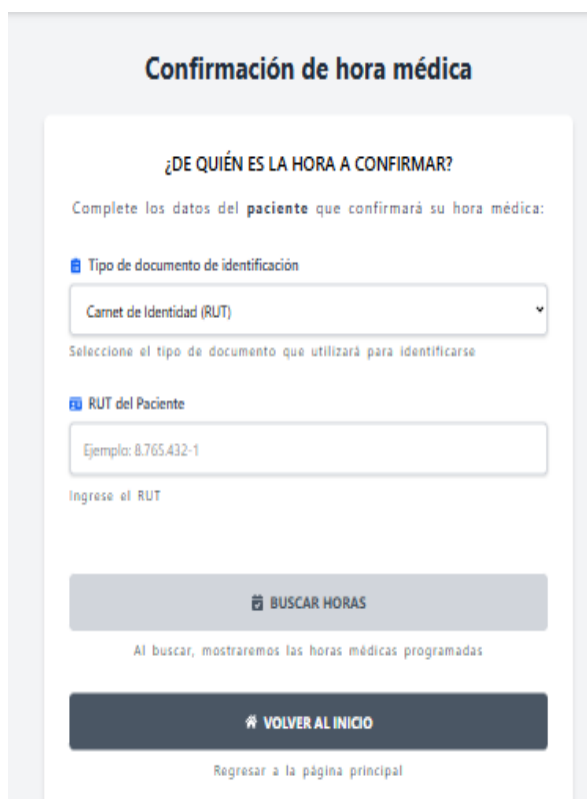


**Ilustración 11: Confirmar hora**

Fuente: Elaboración Grupal

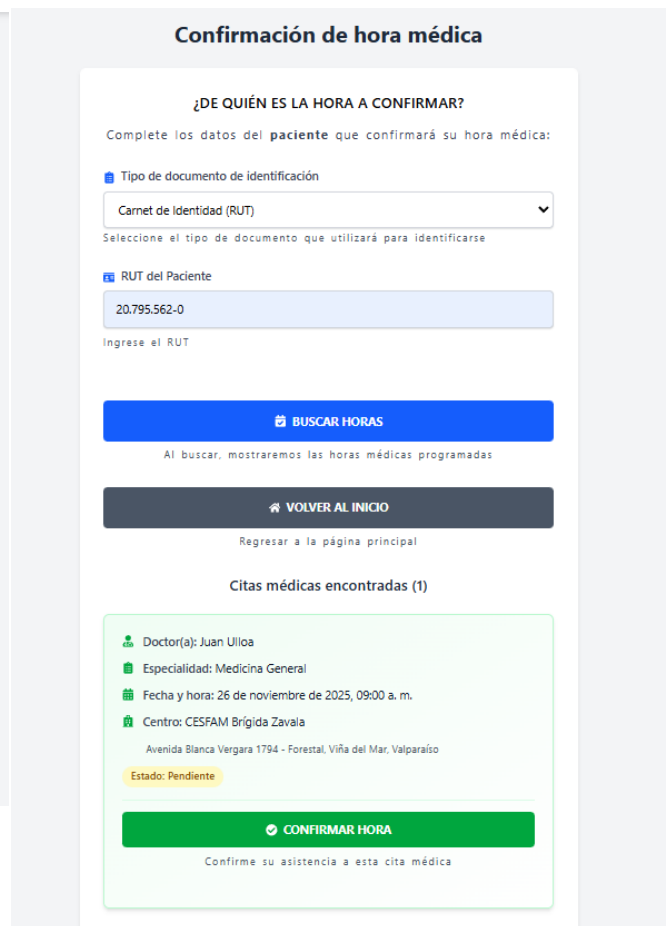


**Ilustración 12: Confirmar hora con Rut**

Fuente: Elaboración Grupal

Estas ilustraciones corresponden la pantalla de confirmación de hora médica, donde el usuario ingresa los datos del paciente para buscar y confirmar una cita previamente agendada. La interfaz guía al paciente paso a paso: primero selecciona el tipo de documento (RUT, Pasaporte o NIP) y luego ingresa el número correspondiente, aplicando validaciones dinámicas según el formato de cada documento. Cuando el valor ingresado es incorrecto, el sistema muestra mensajes de error accesibles con ARIA y estilos visuales que facilitan la identificación del problema. Una vez que el documento es válido, el botón "Buscar Horas" se habilita y ejecuta la búsqueda de citas médicas asociadas al paciente. El componente gestiona estados de carga, errores de búsqueda y desplazamiento automático hacia los resultados. Si la API devuelve citas, estas se muestran en tarjetas visuales que incluyen profesional, especialidad, fecha y hora, centro médico, dirección y estado de la cita. Cada tarjeta incorpora un botón "Confirmar Hora", que envía una solicitud para confirmar la asistencia del paciente; durante ese proceso, el botón muestra un indicador de carga para evitar acciones repetidas. La pantalla también incluye un botón para volver al inicio y asegura accesibilidad general mediante roles semánticos, aria-live, contrastes adecuados y elementos de tamaño táctil. En conjunto, este componente administra todo el flujo de ingreso de identificación, búsqueda de citas y confirmación, presentando la información de manera clara, validada y accesible.

La lógica detrás administra múltiples estados internos: el tipo de documento seleccionado, el valor ingresado, la validez del documento, el estado de carga, la lista de citas obtenidas, posibles errores y la cita actualmente en proceso de confirmación. Para garantizar que los datos ingresados sean correctos, aplica validaciones específicas según el tipo de documento utilizando funciones externas como `validateRut`, `validatePassport` y `validateNIP`, además de aplicar formatos automáticos cuando corresponde. Cuando se ejecuta la búsqueda, el componente realiza una llamada al *backend* utilizando los parámetros construidos dinámicamente en función del documento ingresado. Según la respuesta del servidor, muestra la lista de citas encontradas o un mensaje de error accesible. El sistema también desplaza automáticamente la vista hacia los resultados mediante *scrollIntoView*, mejorando la experiencia de navegación. La acción de confirmar cita se realiza mediante una petición POST al *endpoint* correspondiente, enviando tanto el ID de la hora seleccionada como el documento ingresado. El código contempla respuestas vacías del servidor (como 204 *No Content*), maneja errores devueltos por la API y actualiza la interfaz removiendo la cita ya confirmada para evitar acciones duplicadas. Finalmente, el componente incluye una función de formateo de fechas que transforma valores ISO en textos legibles para el usuario, utilizando convenciones locales de Chile. En conjunto, este código ofrece una interfaz robusta y accesible para consultar y confirmar horas médicas, integrando validación, comunicación con el *backend* y retroalimentación visual en un solo flujo coherente.

### 3.5.4 Cancelar hora médica

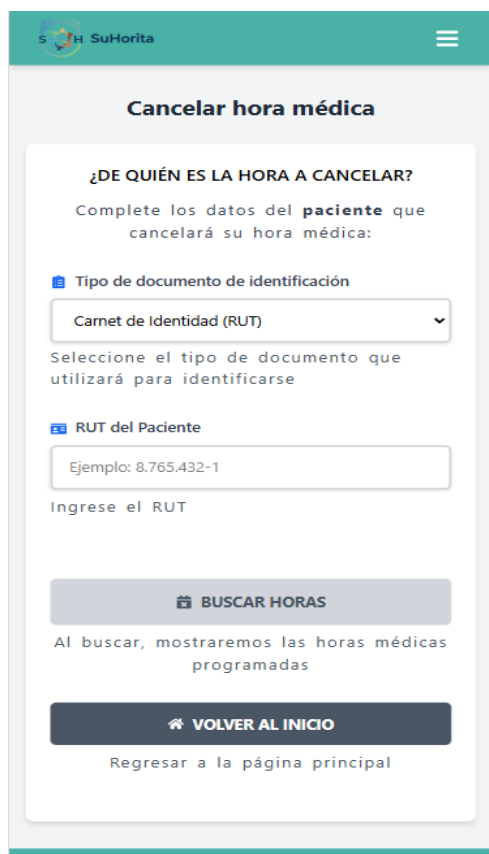


**Ilustración 13: Cancelar hora**

Fuente: Elaboración Grupal

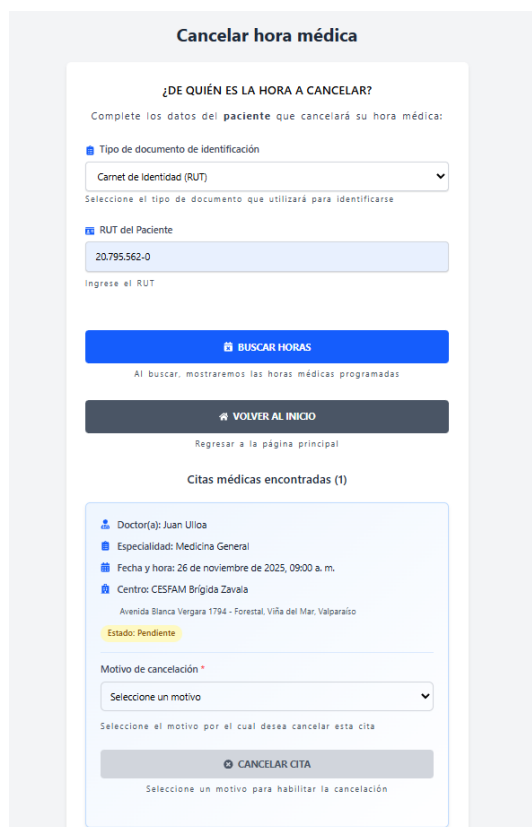


**Ilustración 14: Cancelar hora con Rut**

Fuente: Elaboración Grupal

La interfaz organiza los elementos visuales y accesibles que permiten al usuario buscar y anular una cita. La vista inicia con una estructura general responsiva que incorpora el *Navbar*, un título central y un contenedor principal con estilo uniforme. Dentro del formulario, el código presenta campos para seleccionar el tipo de documento e ingresar el identificador del paciente, aplicando mensajes de ayuda, iconografía contextual e indicadores visuales de error según la validez del dato ingresado.

La interfaz incluye un botón principal para ejecutar la búsqueda de horas asociadas al documento, el cual se habilita o deshabilita dinámicamente según el estado de validación y la actividad del sistema. También se incorpora un botón secundario para volver al inicio, pensado como una ruta de escape clara y accesible. Una vez realizada la búsqueda, el código muestra mensajes de error o resultados utilizando regiones con *aria-live*, referencias para posicionamiento automático (*scroll*), y componentes semánticos que refuerzan la accesibilidad.

Cuando existen citas disponibles, estas se renderizan como tarjetas informativas que incluyen datos del profesional, especialidad, fecha, centro de atención, dirección y estado de la cita, usando colores y jerarquías visuales consistentes. Cada tarjeta incorpora un selector obligatorio para elegir el motivo de cancelación y un botón que activa el proceso de anulación, el cual se habilita solo cuando el usuario ha seleccionado un motivo válido. Todo el diseño prioriza coherencia visual, accesibilidad, uso de iconos significativos, retroalimentación inmediata y una disposición que facilita la lectura en dispositivos móviles y escritorio.

La lógica administra estados como el tipo de documento seleccionado, el valor ingresado, la validez del dato, el estado de carga, las citas encontradas, errores de búsqueda, motivos de anulación y la cita recientemente cancelada. También mantiene una referencia (*resultadosRef*) para controlar el desplazamiento automático hacia los resultados o mensajes relevantes.

El componente valida la entrada según el tipo de documento (RUT, Pasaporte o NIP), aplicando formateos y restricciones propios de cada caso mediante funciones externas. A partir de estas validaciones, habilita dinámicamente la búsqueda. Al consultar las citas, construye los parámetros requeridos por la API, envía una solicitud GET y gestiona diversos escenarios de respuesta: coincidencias, listas vacías, errores HTTP y fallas de conexión. Tras obtener los resultados, ejecuta un desplazamiento automático hacia la sección correspondiente para mejorar la accesibilidad y la experiencia de uso.

Para anular una cita, la lógica verifica que el usuario haya seleccionado un motivo, compone la solicitud DELETE con los parámetros adecuados y maneja respuestas exitosas incluso cuando no incluyen cuerpo (código 204). Luego actualiza el estado eliminando la cita anulada, limpia el motivo asociado y gestiona los distintos estados de carga y error durante el proceso. Además, incorpora una función de formateo de fechas que transforma valores ISO en textos legibles para el contexto chileno, permitiendo mostrar la información temporal de forma consistente. En conjunto, esta lógica integra validación, comunicación con el *backend*, actualización de estados y manejo de errores para sostener el flujo completo de búsqueda y anulación de horas médicas.

## 4 Pruebas y Validación

Este capítulo detalla el proceso de evaluación de la plataforma SuHorita mediante una serie de pruebas diseñadas para corroborar su comportamiento, calidad y accesibilidad. Estas pruebas permiten validar no solo el correcto funcionamiento técnico, sino también el nivel de satisfacción, comprensión e interacción que experimenta un usuario real. La validación se abordó desde cuatro dimensiones fundamentales: funcionalidad, accesibilidad, usabilidad y rendimiento.

### 4.1 Pruebas funcionales

Las pruebas funcionales se desarrollaron con el objetivo de verificar que cada componente del sistema operara de acuerdo con los requerimientos definidos. Para ello, se elaboraron casos de prueba específicos para los flujos principales del módulo: registro de usuario, agendamiento de citas, confirmación de reserva y notificaciones asociadas.

Se empleó una metodología basada en *testing* de caja negra, evaluando la aplicación sin considerar su código interno. Los casos contemplaron entradas válidas, entradas inválidas y escenarios límite para asegurar un comportamiento robusto. Los resultados demostraron que el sistema responde correctamente ante entradas esperadas, mientras que ante entradas erróneas despliega mensajes claros y oportunos de error.

Durante esta etapa se identificaron algunas incidencias, principalmente relacionadas con validaciones incompletas, por ejemplo, Rut inexistentes o problemas de validación del NIP y el pasaporte, errores de diseño responsivo en algunos dispositivos y comportamientos no esperados en la navegación principalmente relacionados con el llamado a la base de datos, ya que no mostraba las opciones o los datos registrados. Todas las observaciones fueron corregidas en iteraciones posteriores, permitiendo fortalecer la estabilidad y coherencia del sistema antes de avanzar a nuevas etapas de evaluación.

### 4.2 Evaluación de accesibilidad

La accesibilidad fue evaluada mediante herramientas automáticas y revisión manual, con el objetivo de verificar el cumplimiento de las pautas WCAG 2.2 y las estrategias definidas en la sección 3.5, además se utilizó Google *Lighthouse* en modo Desktop y Mobile, enfocándose en el apartado de Accesibilidad.

#### 4.2.1 Evaluación automática (Google *Lighthouse*)

Métrica objetivo:

- Puntaje *Lighthouse* (Accesibilidad)  $\geq 90$
- Ausencia de errores críticos de accesibilidad

Resultados obtenidos:

- Escritorio: Accesibilidad  $\geq 93$
- Celulares: Accesibilidad  $\geq 87$
- Se detectaron errores críticos relacionados con contraste.



Celulares



Escritorio



Accesibilidad

Estas comprobaciones destacan las oportunidades para [mejorar la accesibilidad de tu app web](#). La detección automática solo puede detectar un subconjunto de problemas y no garantiza la accesibilidad de tu app web, por lo que también se recomienda hacer [pruebas manuales](#).

### Ilustración 15: Medición de accesibilidad Móvil

Fuente: Elaboración Propia



Celulares



Escritorio



Accesibilidad

Estas comprobaciones destacan las oportunidades para [mejorar la accesibilidad de tu app web](#). La detección automática solo puede detectar un subconjunto de problemas y no garantiza la accesibilidad de tu app web, por lo que también se recomienda hacer [pruebas manuales](#).

### Ilustración 16: Medición de accesibilidad PC

Fuente: Elaboración Propia

#### 4.2.2 Checklist de accesibilidad (evaluación manual)

Se aplicó un *checklist* de accesibilidad de forma manual para validar aspectos que no siempre son detectados por herramientas automáticas. La evaluación se realizó sobre los flujos críticos del sistema, priorizando el proceso de agendamiento

**Tabla 6: Checklist de accesibilidad**

| Criterio evaluado                                            | Cumple | Observaciones              |
|--------------------------------------------------------------|--------|----------------------------|
| Orden lógico de tabulación                                   | ✓      | Secuencia predecible       |
| Foco visible en elementos interactivos                       | ✓      | <i>Outline</i> contrastado |
| Uso correcto de <i>labels</i> en formularios                 | ✓      | Asociados a inputs         |
| Mensajes de error comprensibles                              | ✓      | Texto + iconos             |
| Anuncio de cambios dinámicos ( <i>aria-live</i> )            | ✓      | Estados informados         |
| No dependencia exclusiva del color                           | ✓      | Iconos semánticos          |
| Tamaño mínimo de controles ( $\geq 44 \times 44 \text{px}$ ) | ✓      | Cumple WCAG 2.5.5          |
| Compatibilidad con lector de pantalla                        | ✓      | Roles ARIA correctos       |

#### 4.3 Pruebas de usabilidad

Las pruebas de usabilidad buscaron evaluar la experiencia real de los usuarios, considerando la facilidad para comprender, aprender y completar las tareas principales de la aplicación. Para ello, se realizaron sesiones con usuarios representativos del público objetivo, principalmente adultos mayores, no se consideró el registro del usuario en la aplicación en las pruebas, ya que se asume que el usuario ya está registrado en la base de datos de nuestro sistema, se registraron métricas como:

- Tiempo total para completar tareas.
- Nivel de comprensión de la interfaz.
- Satisfacción subjetiva frente al diseño y la navegabilidad.

**Tabla 7: Resultados de pruebas de tareas**

resultados obtenidos en las pruebas de usabilidad realizadas con usuarios representativos

| Usuario | Tarea          | ¿Completó? | Tiempo   | Observaciones                           |
|---------|----------------|------------|----------|-----------------------------------------|
| U1      | Agendar hora   | Sí         | 1:45 min | Flujo claro                             |
| U2      | Agendar hora   | Sí         | 2:05 min | Duda en botón de términos y condiciones |
| U3      | Agendar hora   | Sí         | 1:30 min | Sin errores                             |
| U1      | Confirmar hora | Sí         | 0:50 min | Confirmación clara                      |
| U2      | Cancelar hora  | Sí         | 1:10 min | Duda en bloque de hora                  |

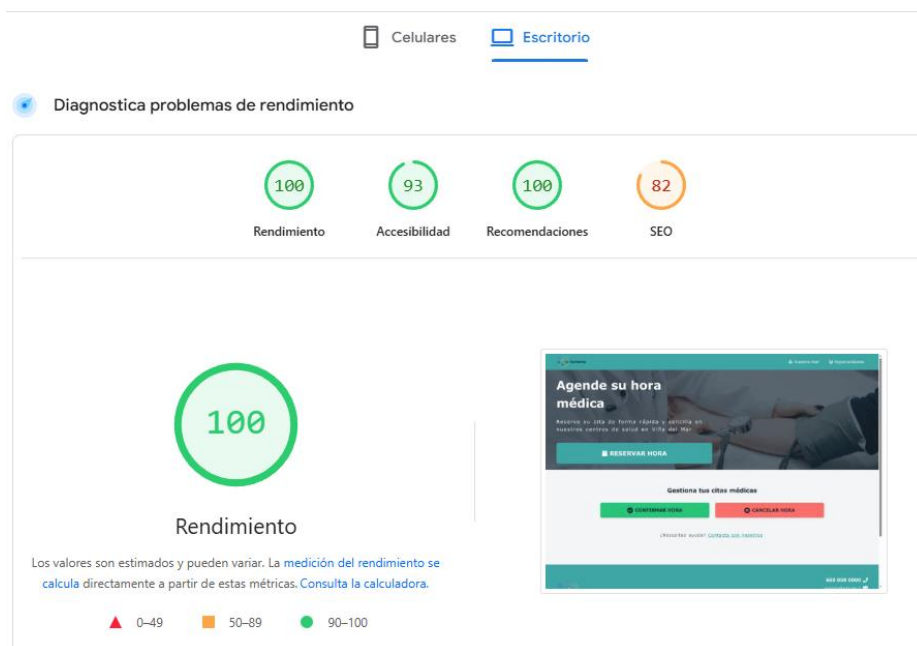
Se aplicaron técnicas como observación directa y pensamiento en voz alta para obtener información cuantitativa y cualitativa, además de la medición del tiempo. Los participantes expresaron una alta valoración de la claridad visual, el uso de íconos y la estructura lineal del agendamiento. Sin embargo, también se reportaron mejoras necesarias en la visibilidad de algunos botones y en la explicación de ciertos mensajes informativos.

Con base en estos hallazgos, se optimizó la jerarquía visual, se ajustaron colores y tamaños para mejorar la legibilidad, y se simplificaron textos para facilitar la comprensión inmediata. Estas mejoras aumentaron la eficiencia del flujo y redujeron la tasa de errores de interacciones.

#### 4.4 Pruebas de rendimiento y estabilidad

Con el fin de asegurar que la aplicación funcionara de manera fluida y estable en diversos entornos, se realizaron pruebas de rendimiento utilizando diferentes dispositivos (gama alta, media y baja) y condiciones de red (Wi-Fi, 4G, baja conectividad) además se utilizó Google *Lighthouse* (Performance) para medir el rendimiento. Las métricas analizadas incluyeron:

- Tiempo de carga inicial de la aplicación.
- Rendimiento en la navegación entre pantallas.
- Uso de memoria y CPU durante operaciones intensivas.
- Estabilidad ante uso prolongado y repetitivo.
- Puntaje *Lighthouse* Performance de rendimiento.



## Ilustración 17: Medición de rendimiento PC

Fuente: Elaboración Propia



## Ilustración 18: Medición de rendimiento Movil

Fuente: Elaboración Propia

Los resultados mostraron un desempeño notable, con tiempos de carga moderados y una navegación estable incluso en dispositivos de menor capacidad. Se realizaron optimizaciones adicionales, como reducción del peso de ciertos recursos, implementación de *Lazy Loading* y mejoras en el manejo de estados, lo que permitió disminuir el consumo de memoria y aumentar la fluidez general.

#### **4.5 Correcciones, mejoras e iteraciones**

El proceso de validación generó múltiples aprendizajes que derivaron en actualizaciones y refinamientos. Entre las mejoras implementadas destacan:

- Correcciones funcionales: validación más estricta de campos, manejo de errores con la BD y eliminación de fallas en flujos específicos.
- Ajustes de usabilidad: reubicación de botones, aumento de tamaños, iconografía más clara y mejora en la jerarquía visual.
- Optimización de accesibilidad: incorporación completa de atributos ARIA, mejora en contrastes, etiquetado correcto de botones y optimización para lectores de pantalla.
- Mejoras de rendimiento: compresión de imágenes, reducción de llamadas innecesarias y optimización de componentes para tiempos de respuesta más rápidos.

Estas iteraciones permitieron consolidar una versión más completa, estable y accesible del módulo, alineada con los objetivos definidos al inicio del proyecto.

## 5 Conclusiones

El desarrollo del módulo paciente de *SuHorita* permitió demostrar que es posible diseñar e implementar una plataforma de agendamiento médico que reduzca barreras tecnológicas, cognitivas y de accesibilidad, ofreciendo una experiencia inclusiva y alineada con las necesidades reales de los usuarios. A lo largo del proyecto se comprobó que la accesibilidad no es un complemento opcional, sino un componente estructural del diseño *Fronddend*, especialmente cuando el sistema está orientado a adultos mayores y a poblaciones con baja alfabetización digital, visión reducida o dificultades para interactuar con interfaces tradicionales.

El proceso de diseño e implementación evidenció la importancia del Diseño Centrado en el Usuario y la integración continua de retroalimentación, lo que permitió ajustar flujos, simplificar interacciones y reforzar patrones visuales para garantizar que la plataforma fuera comprensible, intuitiva y predecible desde el primer uso. Además, la modularidad del *stack* tecnológico facilitó el desarrollo de componentes reutilizables, seguros y escalables, asegurando una arquitectura mantenible a largo plazo.

Un aprendizaje clave del proyecto fue la correcta interpretación y aplicación de los estándares de accesibilidad WCAG 2.2, entendiendo no solo sus criterios técnicos, sino también su impacto directo en la experiencia del usuario. Se incorporaron buenas prácticas como contraste adecuado, semántica HTML consistente, manejo correcto del foco, textos alternativos, validaciones accesibles y uso adecuado de ARIA. Este proceso permitió comprender que los estándares WCAG no son una simple lista de verificación, sino un marco conceptual que orienta decisiones de diseño más humanas y responsables.

Las pruebas funcionales, de usabilidad y rendimiento permitieron validar el comportamiento del sistema en condiciones reales, demostrando que el módulo cumple con los objetivos propuestos: agilizar el proceso de búsqueda y confirmación de citas, reducir la complejidad cognitiva del flujo y brindar una experiencia coherente, estable y accesible. Estas iteraciones también permitieron corregir problemas, optimizar tiempos de carga y mejorar la presentación de la información según las recomendaciones de los usuarios.

Finalmente, el proyecto dejó como principal conclusión que construir software inclusivo requiere una combinación equilibrada de ingeniería, diseño, empatía y validación continua. La experiencia adquirida en WCAG, ARIA y buenas prácticas de accesibilidad constituye un aprendizaje significativo que fortalece la capacidad de desarrollar soluciones digitales más justas, universales y alineadas con los estándares de calidad. Esta tesina marca un paso sólido hacia plataformas de salud digitales más accesibles, orientadas a reducir brechas y garantizar que cualquier persona, sin importar su nivel de alfabetización tecnológica o condición física, pueda acceder de forma digna y eficiente a la atención médica.

## 6 Agradecimientos

Agradezco a todas las personas que hicieron posible llegar hasta aquí. En primer lugar, a mi abuela Margarita Quiroz Olivares quien ahora me cuida desde el cielo, pero fue mi mayor apoyo en mis primeros años de universidad junto a mi abuelo Ernesto Castillo Fuentes que aún sigue siendo un pilar fundamental en mi educación, a mi mamá por su preocupación, al resto de mi familia.

Asimismo, expreso mi gratitud a mis compañeros de proyecto y a todas las personas que, directa o indirectamente, contribuyeron con su tiempo, retroalimentación y apoyo durante cada etapa de este proceso. Su colaboración fue fundamental para alcanzar los objetivos planteados.

## 7 Referencias

- [1] S. Venkatarajalu, «researchgate,» junio 2024. [En línea]. Available: [https://www.researchgate.net/figure/Progressive-Web-Apps-Architecture\\_fig1\\_381643020](https://www.researchgate.net/figure/Progressive-Web-Apps-Architecture_fig1_381643020).
- [2] elisatorregrosa, «webpositer,» 8 10 2021. [En línea]. Available: <https://www.webpositer.com/blog/que-es-microcopy-ejemplos/>.
- [3] D. Stillgoe, «blend,» 13 6 2024. [En línea]. Available: <https://www.blendb2b.com/blog/website-call-to-action-cta-examples-and-best-practices>.
- [4] J. Mawa, «freecodecamp,» 27 09 2001. [En línea]. Available: <https://www.freecodecamp.org/news/react-accessibility-tools-build-accessible-react-apps/>.
- [5] N. Kaurav, «medium,» 16 06 2025. [En línea]. Available: <https://medium.com/kpmg-uk-engineering/building-accessible-frontend-applications-with-react-7f4b9d5bf9fe>.
- [6] chrome, «chrome for developers,» chrome , 2 06 2025. [En línea]. Available: <https://developer.chrome.com/docs/lighthouse/overview?hl=es-419>.
- [7] A. R. Martin, «caixabanktech,» 30 04 2025. [En línea]. Available: <https://www.caixabanktech.com/es/blogs/accesibilidad-digital-en-react-mejores-practicas-para-crear-aplicaciones-inclusivas-y-eficientes/>.
- [8] W3C, «W3C,» 06 05 2025. [En línea]. Available: <https://www.w3.org/TR/WCAG21/>.
- [9] E. d. Teleprompter.com, «Teleprompter,» 15 febrero 2025. [En línea]. Available: <https://www.teleprompter.com/blog/effectiveness-of-lexend-and-opendyslexic-fonts>.
- [10] P. P. Murray, «ehealthreporter,» 14 septiembre 2022. [En línea]. Available: <https://ehealthreporter.com/la-salud-digital-como-herramienta-de-inclusion-casos-de-exito-en-la-region/>.
- [11] G. d. T. d. E. y. D. (. y. e. G. d. T. d. I. P. d. A. (. WG)., «W3C,» 06 mayo 2025. [En línea]. Available: <https://www.w3.org/WAI/standards-guidelines/wcag/es>.
- [12] Á. Peredo, «Galernaestudio,» 21 Julio 2025. [En línea]. Available: <https://galernaestudio.com/disenio-centrado-usuario/>.



- [13] avislatam, «avislatam,» [En línea]. Available: <https://slide.avislatam.com/portal-ayuda/manuales/6825/AVIS-Manual-Agenda-REV6825.pdf>.
- [14] I. Sotelo, «sospnt,» 27 febrero 2023. [En línea]. Available: <https://sospnt.com/blog/306-de-que-se-trata-scrum>.
- [15] Microsoft, «learn microsoft,» 20 09 2025. [En línea]. Available: <https://learn.microsoft.com/es-es/microsoft-edge/progressive-web-apps/>.