BarberManager: Diseño y desarrollo de una aplicación para la gestión digital de barberías

Josep Molina Eirin

1 de febrero de 2026

Resumen– Este trabajo presenta el diseño y desarrollo de BarberManager, una aplicación móvil híbrida orientada a la digitalización de la gestión de pequeñas barberías.
El objetivo principal es ofrecer una herramienta sencilla y accesible que permita gestionar citas, servicios y perfiles de usuario, reduciendo la carga administrativa del profesional. Además, el proyecto hace especial hincapié en la experiencia de usuario y accesibilidad, teniendo en cuenta perfiles con poca experiencia tecnológica o dificultades de diversa naturaleza.
La solución ha sido desarrollada haciendo uso de Vue e Ionic en el frontend, y NestJS para el backend. El proceso ha incluido pruebas de prototipado, pruebas de usabilidad y testing técnico para seguir el proceso real de un proyecto.

Palabras clave– gestión de barberías, aplicación móvil, experiencia de usuario, accesibilidad, Vue, Ionic

Abstract– This project presents the design and development of BarberManager, a hybrid mobile application aimed at supporting the digitalisation of small barbershop management.
The main goal is to offer an easy and accessible tool for managing appointments, services and user profiles, reducing the administrative workload of the barber. Additionally, the project especially focuses on the User Experience and accessibility, considering users with limited technical skills or difficulties of various kinds.
The application has been developed using Vue and Ionic for the frontend development, and NestJS for the backend. The process has also included prototyping techniques, usability tests and code testing to follow the real process of a commercial project.

Keywords– barbershop management, mobile application, user experience, accessibility, Vue, Ionic

1 Introducción: Cuestiones a resolver

La evolución del panorama tecnológico está transformando profundamente nuestra manera de comunicarnos, trabajar y gestionar nuestras actividades cotidianas.
Desde simples actos como pagar un billete de transporte público, hasta aspectos más notorios como la gestión bancaria a través del móvil, estamos cambiando la forma de interactuar hasta límites nunca vistos. Esto trae consigo varios retos, tanto a nivel técnico como social, que garanticen la inclusión de todos los usuarios de un servicio, garantizando que se ofrece la misma cobertura que al método que se sustituye.

1.1. Reto Económico

Actualmente gran parte de los grandes negocios y/o multinacionales se encuentran en un proceso de cambio en el que cuentan con la modernización tecnológica como motor central de esta, con tal de atraer nuevos clientes familiarizados con los dispositivos electrónicos. Esto se

• E-mail de contacto: 1558589@uab.cat
• Mención: Ingeniería del Software
• Trabajo tutorizado por: Emanuel Adolfo Vallejo Cañizares (departament)
• Curso 2025/26

traduce en grandes inversiones en estudios de mercado, diseño de soluciones tecnológicas, contratación de asesores tecnológicos o utilizar soluciones ya existentes en el mercado, pero sólo sostenibles para empresas con cierto volumen de facturación.

Todo este movimiento conlleva unas ventajas, pero deja por el camino a damnificados: **los pequeños negocios**. Estos, a diferencia de lo comentado anteriormente, basan su negocio en la atención directa y la fidelización del cliente mediante un trato personalizado y directo, y su apuesta por la transformación tecnológica no se basa en el atractivo que el marketing pudiese generar, sino más bien para la automatización o gestión de tareas adjuntas a la naturaleza del negocio como, en el caso que nos ocupa, la gestión de las citas en una barbería.

Tradicionalmente, si uno desea ir a la barbería, basta con acercarse al establecimiento o simplemente llamar para acordar una cita con el barbero. Este método es efectivo, pero **carga al trabajador con una responsabilidad adicional** que es externa a su oficio, y lo interrumpe al tener que dejar esperando al cliente al que está haciendo un servicio para atender la gestión de citas. Por tanto, la digitalización tiene un impacto práctico inmediato: el barbero puede ofrecer a sus clientes la posibilidad de reservar citas en línea mediante una aplicación. El cliente, desde su dispositivo móvil, puede seleccionar la hora disponible que más le convenga y escoger de antemano los servicios que desea recibir, agilizando así la gestión del establecimiento y mejorando la experiencia del usuario en la barbería.

Esto sería una idealización de la realidad, ya que no contamos con soluciones sencillas que cubran estas situaciones, situándose más como opciones enfocadas al negocio entero, es decir, soluciones CRM [1][2]. Por tanto, uno de los objetivos es la simplificación de la funcionalidad al servicio, de manera que personas con pocos conocimientos técnicos, y sin tener que realizar grandes inversiones, puedan gozar de la automatización de sus negocios, y centrarse en su seña de identidad: el trato al cliente.

1.2. Reto Social

Esta modernización también descuida a otro actor importante: los **usuarios inexpertos en tecnología**. Está bien puede ser gente de edad avanzada que no se encuentra en disposición de un teléfono inteligente, o gente que no es habilidosa en este sentido, y encuentra en estos cambios obstáculos que los alejan de las corrientes actuales. En otros casos, contamos con personas que se han visto obligadas a adaptarse a los tiempos, pero que en ningún caso se encuentran cómodas haciendo un uso más allá de enviar y recibir mensajes, o consultar Internet.

Por tanto, para la realización de este proyecto se tendrá en cuenta aspectos de **accesibilidad** que faciliten la integración de las personas en estas nuevas soluciones tecnológicas. No consiste en adaptar la solución resultante, sino pensar en todo momento del desarrollo en el emplazamiento de botones, pantallas u opciones propias, o incluso en la manera de acceder al servicio [3].

2 Estado del Arte

Actualmente, tal y como hemos comentado en el apartado anterior, contamos con múltiples propuestas en el mercado que vienen a intentar cubrir las necesidades previamente expuestas: **aplicaciones destinadas a barberías de particulares** que necesitan de una digitalización para su negocio. Pese a ser **propuestas muy sólidas**, podemos sacar varios puntos clave.

Por una parte, nos ofrecen plataformas con un sinfín de opciones y servicios orientados a la gestión de citas, a la generación de estadísticas o la automatización de mensajes de recordatorio para clientes. Soluciones que si bien son potentes, **carecen de un enfoque suficiente** para el mercado al que se destinan. Podemos ver cómo estas soluciones actúan como una **solución genérica** a la gestión de un negocio, sea cual sea su naturaleza, al que se le añaden ciertos matices para intentar darle esa utilidad de aplicación de gestión de barbería. Esto puede decepcionar al barbero, que espera que esta solución tenga un impacto notable en su negocio. Por supuesto que lo puede tener, pero si queremos que algo satisfaga las necesidades de los barberos en su totalidad, debemos desarrollar una solución que tenga un **uso exclusivo para la gestión de barberías** y en la que el barbero encuentre todo aquello que espera.

Por otra parte, y continuando la idea anterior, estas plataformas cuentan con interfaces atractivas, de diseño moderno y aparentemente funcional. No obstante, su **naturaleza genérica e inflexible** penaliza su uso y destierra a personas con poco conocimiento de las tecnologías, también conocidos como *analfabetos digitales* [4]. Este grupo de personas son aquellos que no cuentan con los conocimientos ni soltura suficientes para orientarse con dispositivos electrónicos, muchas veces obligados por la sociedad actual a adaptarse o caer en el ostracismo, y que por tanto al encontrarse con aplicaciones de este tipo no se encuentran cómodos y les cuesta encontrar botones o menús a los que necesitan acceder.

Esta situación motiva a desarrollar una solución que tenga en cuenta la **experiencia de usuario**, y que incluya al máximo número de personas disponibles, añadiendo funcionalidades que incluyen a **usuarios con dificultades físicas**, y facilitando la experiencia lo máximo posible para personas sin experiencia previa o limitada, **evitando caer en excesos visuales** o propuestas rompedoras **sin un sentido práctico** que pueden hacernos caer en el problema descrito como las *puertas de Norman* [5].

3 Objetivos

El principal objetivo es el desarrollo de una aplicación móvil que permita la gestión de Barbería, y que permita al barbero una digitalización rápida y con una curva de aprendizaje reducida. Para ello la aplicación se desarrollará **haciendo foco en el barbero** y aquellas funciones que más utiliza, ofreciendo también un **grado de personalización útil** que permita simplificar la experiencia dentro de la aplicación. Además, se aplican técnicas de estudio de experiencia de usuario, en específico user testing para

garantizar que la aplicación cumple con su vocación de satisfacer a un diverso número de perfiles diferentes, teniendo en cuenta a **colectivos con dificultades** en el manejo de las tecnologías y a **grupos con problemas físicos** que puedan interferir con operar la aplicación.

Por otra parte, y tomando un **perspectiva académica**, este trabajo tiene como objetivo la puesta en práctica de tecnologías anteriormente desconocidas para mi, unidas al desarrollo de código utilizando técnicas de diseño y patrones utilizados ampliamente en el mundo comercial, y desarrollo de frontend y backend simultáneamente.

Por la mención a la que pertenezco dispongo de conocimientos desarrollando una aplicación para dispositivos móviles, pero en este proyecto se intenta realizar el proceso completo de desarrollo: recogida de requisitos, creación de documentación y prototipo o sketch representativo, desarrollo de la aplicación, test del código, aplicación de técnicas de experiencia de usuario para la mejora continua del producto y garantizar su usabilidad, y finalmente la **entrega de una aplicación funcional en un entorno real**, es decir, que la aplicación consume recursos del backend durante su uso en una plataforma móvil real.

4 Metodología

Después de revisar múltiples opciones, la que mejor encaja en este proyecto sería la metodología **Kanban**. Este enfoque de trabajo de tipo *Agile* originario de la industria automovilística japonesa [6] encaja con las necesidades y tipología del proyecto a desarrollar.

- Contamos con un tablero con las columnas **To Do, In Progress, Pull Request, Merged y Update** con tal de ordenar las tareas según su estado actual. Esto nos permite tener un control sobre la situación de cada porción de la aplicación.
- No se nos limita el número de tareas totales, sino el control del flujo a un determinado número de tareas para evitar sobrecarga, y garantizar un flujo de trabajo uniforme
- Se permite **añadir tareas** en cualquier punto del desarrollo
- Es más flexible que otras metodologías como el **Modelo en espiral**, al no contar con ciclos estructurados previamente, y al permitir introducir cambios/mejoras en plazos de tiempo menores.
- Al contar con técnicas de experiencia de usuario involucradas en el desarrollo de la aplicación, como es el user testing, permite poder realizar las pruebas pertinentes cuando se estime, sin depender de un calendario rígido

Dentro de las soluciones comerciales encontradas para la gestión del proyecto mediante esta tecnología, y según experiencias previas, se hará uso de **Trello** como herramienta para la gestión del tablero de requisitos.

5 Plan de Proyecto

Este proyecto se estima que toma **aproximadamente unas 300 horas**, y por tanto para cumplir con los objetivos planteados, debemos llevar a cabo una planificación temporal rigurosa que nos permita organizar las tareas, especificar su orden y estimar la carga de trabajo que conlleva cada una de ellas. Para ello se ha utilizado la herramienta *Microsoft Project* [7], en la que se ha redactado una hoja de tareas a realizar, con las fechas estimadas de realización, y esto acompañado del correspondiente **diagrama de Gantt** [8].

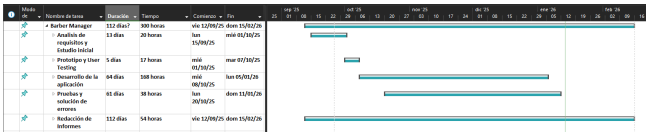

Fig. 1: Lista de tareas y Diagrama de Gantt asociado

5.1. Requisitos de Sistema

Primero de todo se realizó una **búsqueda de requisitos** funcionales y no funcionales que permitiera sentar las **bases** de lo que sería el **proyecto a desarrollar**. Después de una lista inicial y su posterior filtrado, damos con la lista con los objetivos.

5.2. Requisitos Funcionales

RF-1	Crear cita
RF-2	Eliminar cita
RF-3	Añadir servicio
RF-4	Modificar servicio
RF-5	Eliminar servicio
RF-6	Gestionar precios y duración de cada servicio.
RF-7	Consultar ingresos diarios, semanales y mensuales.
RF-8	Ver los servicios más solicitados.
RF-9	Registro de horarios y turnos.
RF-10	Gestión del perfil de usuario.

5.3. Requisitos No Funcionales

RNF-1	La app debe responder rápidamente.
RNF-2	Soportar múltiples usuarios simultáneamente sin caídas.
RNF-3	Interfaz intuitiva y clara, fácil de usar.
RNF-4	Autenticación segura
RNF-5	Protección de datos de clientes
RNF-6	Debe garantizar la usabilidad a personas con daltonismo.

5.4. Fases de desarrollo

- Obtención de Requisitos Funcionales y No Funcionales
- Preparación de prototipos interactivos, mediante el uso de herramientas como ***Figma***.

- Programación del **frontend** de la aplicación haciendo uso del framework de desarrollo **Vue + Ionic**, directamente enfocados en el desarrollo de aplicaciones móviles y web progresiva (PWA) [9]
- Programación del **backend** haciendo uso de **NestJS**.
- Uso de técnicas de experiencia de usuario, como el ***user testing*** [10] para recibir feedback sobre la interfaz, y su consiguiente análisis.
- Ajuste del frontend en base a los datos recogidos.
- Testing a nivel de código de ciertos componentes del frontend.

6 Tecnologías del Proyecto

- **NestJS:** Framework de Node.Js para la construcción del backend. Amplía las funciones de Node.js al permitir una mayor escalabilidad y modularidad, garantizando una mantenibilidad a largo plazo más sencilla.
  - **Prettier:** Herramienta para **formatear el código** de manera uniforme, aplicando un estilo, mejorando su legibilidad y consistencia.
  - **JSON Web Tokens:** Método de autenticación que se basa en un token que contiene la información de un usuario. Permite conexiones más cómodas, sin necesidad de hacer un login cada vez que se accede, es *stateless* [11], y seguro.
  - **Mongoose:** Facilita el modelar y manejar datos para MongoDB, realizando consultas y validaciones de manera más sencilla.
  - **Vitest:** Framework para el desarrollo de **pruebas unitarias** en Vite, framework incluido en Vue para el desarrollo de aplicaciones. [12]
- **Figma:** Herramienta para el diseño de prototipos de **alta fidelidad**, pudiéndose realizar también prototipos **interactivos** para dar soporte a métodos como el *user testing*.
- **Vue:** Framework de Javascript orientado al diseño del *frontend*. Permite crear componentes reactivos y reutilizables, que facilitan la construcción de interfaces interactivas y modernas.
- **Ionic:** Framework que utiliza a Vue como base. Permite desarrollar aplicaciones híbridas para múltiples entornos como **Android, iOS o PWA**. Además, ofrece componentes de interfaz ya preparados, llamadas a elementos de sistema y CLIs propia para manejar despliegues o generación de ejecutables, entre otros.
- **Capacitor:** Paquete incorporado en la librería de Ionic que permite el **empaquetado y generación** de archivos **instalables** en dispositivos con sistema operativo **iOS o Android** basándose en una aplicación web. Permite la gestión de notificaciones, cookies, sesiones o memorias de Pinia de manera automática.
- **Docker:** Plataforma de creación de contenedores, creando un entorno en el que la aplicación se ejecuta independientemente del sistema.
- **TypeScript:** Variante de JavaScript respaldada por Microsoft que incorpora tipado estático, mejorando la detección de errores y la legibilidad del código.
- **Azure App Service:** Es un servicio de tipo **PaaS (Platform as a Service)** [13], que permite el despliegue de manera sencilla y escalable de aplicaciones web y APIs sin requerir una gestión activa de servidor [14].

6.1. Patrones de diseño

6.1.1. Patrón Hexagonal

También conocida como **Ports and Adapters**, permite separar la lógica de negocio del mundo exterior.

- **Core:**
  - Lógica central de la aplicación, aglutina el funcionamiento de las funcionalidades
  - No depende de ningún framework ni de detalles técnicos
- **Puertos:**
  - Se definen dentro de los métodos de una interfaz. **No son la implementación del método, solo la declaración.**
  - Se colocan dentro del Core
  - Actúan como contratos, puesto que dictan qué es lo que el *Core* necesita del exterior.
- **Adaptadores:**
  - Se sitúan fuera del core, en los **módulos**.
  - Se realiza la implementación de los métodos declarados en los puertos.
  - **Tipos:**
    - **Entrada (Driving Adapters)**: Reciben peticiones del exterior, y las traducen al core.
      Ejemplo: Traducir una petición HTTP en una llamada al caso de uso.
    - **Salida (Driven Adapters)**: A petición del Core se ejecutan acciones hacia sistemas externos.
      Ejemplo: Consultas SQL a una base de datos relacional [15].

6.1.2. CQRS (Command Query Responsibility Segregation)

Se encarga de separar las operaciones de lectura de las operaciones de escritura.

- La idea inicial es que las consultas y las modificaciones de datos no deben compartir la misma lógica ni mismos modelos de datos
- Permite separar la parte de lectura y la de escritura de forma independiente.
- Separamos responsabilidades, cosa que facilita el mantenimiento y escalabilidad futuros [16].

6.1.3. Repository

Patrón que actúa como intermediario entre la lógica de negocio, y la fuente de datos (base de datos, API, etc.)
Objetivo: Ocultar los detalles de acceso a datos, de manera que la lógica de negocio trabaja con **objetos en memoria** sin preocuparse de cómo se almacenan o recuperan.

- Funciona como una caja negra en la que entran o salen datos, pero en ningún momento se sabe que hay detrás de cada método.
- Facilita cambios en la fuente de datos sin afectar la aplicación.
- Permite pruebas unitarias más sencillas, especialmente en el uso de *Mock objects* [26].
- Centraliza el acceso a datos, evitando duplicidad de información y carga innecesaria de esta en la memoria, ralentizando el funcionamiento del programa [17].

6.2. Buenas Prácticas

6.2.1. Inyección de Dependencias

Se utiliza para proveer los objetos que una clase necesita. Esto consiste en construir las dependencias que necesita la clase de manera externa, y posteriormente se le facilitan.

- Utilizamos el constructor de la clase para, mediante el paso de parámetros, inicializar las variables.
- En caso de modificar el método que se pasa como parámetro, no necesitaremos adaptar nuestra clase, ya que es ajena al modo de generar ese valor, solo lo recibe [18].

6.2.2. Value Object

Representa un concepto del dominio, es decir, un objeto viene definido por sus valores, y no tanto por su identidad.

- Es inmutable, puesto que una vez creado sus valores no cambian posteriormente (si se le puede asignar tantas veces como queramos).
- No tiene entidad propia. Esto quiere decir que no necesita un identificador único para diferenciarse de los otros, sino que lo que importa es su contenido o valores que almacena.
- Útil si deseas evitar errores por cambios accidentales de valores, a la vez que genera código más limpio.
- **Ejemplo:** En una función, pasamos una variable por valor. De esta manera garantizamos que la función no puede modificar su valor a menos que sea asignando el resultado a la misma variable [19].

6.3. UX Design

6.3.1. Prototipado

El prototipado es una técnica muy utilizada en el software actual. Consiste en crear un modelo del sistema que puede ser utilizado para entender los objetivos de este, y acabar de clarificar los requisitos finales. Esta representación bien puede ser un esbozo a papel, algo más profesional a color, o realizado utilizando herramientas informatizadas especializadas (como en nuestro caso lo es **Figma**) [20].

6.3.2. User Testing

Técnica utilizada para realizar pruebas sobre una interfaz de usuario diseñada. Este tipo de test cuenta con **dos personas:** un facilitador y un apuntador. El **facilitador** se encarga de plantear al usuario de prueba un posible caso de uso real de la aplicación, y mediante el uso de un **prototipo** la persona debe interactuar con la interfaz. El apuntador debe estar atento a los movimientos del usuario, apuntar cualquier atisbo de duda o error al usar la interfaz.

- Puede haber más de una persona anotando, estos se llaman **observadores**.
- Debemos garantizar que la fluidez entre pantallas sea rápida para que la interacción no se vea afectada por el método en sí.
- Debemos hacer foco en nuestro prototipo, y en todo momento basarnos en el test de este, no de la persona que lo prueba[20].

7 Desarrollo de la aplicación

7.1. Etapa inicial: Análisis de requisitos y estudio inicial

Al comenzar el proyecto, partía de la base de contar con una estructura de backend ya definida por mi tutor Emanuel, y varios endpoints realizados. Esto se debe a que este backend sirve de **base para dos frontends**: uno que va destinado a barberías, que es objeto de este trabajo final; y otro destinado a los clientes de la barbería, que se encontraba ya comenzado, y ha sido desarrollado paralelamente por Emanuel.

Para saber cómo materializar la idea inicial primero de todo se realizó un estudio de requisitos, usando técnicas como el brainstorming con Emanuel, búsqueda de aplicaciones con el mismo objetivo y estudiar sus carencias o lagunas no desarrolladas; o mediante una lista de tareas a realizar en una barbería y que puedan ser digitalizadas. Finalmente se realizó una lista inicial de requisitos, de la cual pudimos sacar una conclusión inicial sobre qué tecnologías utilizar.

Desde un principio la idea era basarnos en un framework conocido, de los cuales destacó Vue por su versatilidad a la hora de generar componentes de manera sencilla y poder aplicarlos en el proyecto. Por otra parte, su curva de aprendizaje es más reducida que en opciones como Angular, y mucho más estable en el aspecto de versiones que React, especialmente desde que se dio el salto a Vue 3 [22].
Una vez escogida la base, me debatí entre dos frameworks a añadir a Vue como son Quasar o Ionic. Estos están pensados para facilitar bibliotecas de componentes nativos para dispositivos móviles, compatibles con iOS y Android,

CASO DE USO	Crear cita
Versión	1.0
Fecha	06/10/2025
Autores	Josep Molina Eirin
Descripción	Permite al barbero crear nuevas citas en base a la disponibilidad o a las peticiones de los clientes.
Actores	Barbero
Pre-condición	El barbero o administrador debe haber iniciado sesión con su cuenta previamente.
Flujo principal	1. El usuario accede al calendario de citas 2. Selecciona una fecha y hora disponible. 3. Introducimos la información del cliente y el servicio 4. Guardar cambios
Subflujos	—
Flujos alternativos	Si la hora seleccionada ya esta ocupada, derivamos un mensaje de error por pantalla.
Postcondición	Registramos correctamente la cita
Requisitos no funcionales	RNF-1, RNF-2, RNF-3, RNF-4, RNF-5
Prioridades	Alta
Comentarios	Funcionalidad esencial para el correcto funcionamiento de la barbería.

Fig. 2: Ejemplo del caso de uso de **Crear cita**

y una herramienta llamada Capacitor, que permite la generación de paquetes instalables para estos sistemas operativos, y que por tanto permiten hacer aplicaciones móviles. Después de comparar sus bondades y desventajas, se acabó optando por Ionic al contar con una biblioteca de componentes más versátil y amplia, y por su popularidad a la hora de encontrar documentación de calidad sobre el uso de ciertas funcionalidades.

Con esto ya definido, se procedió a hacer una planificación temporal con un diagrama de Gantt, y a la realización del primer informe entregado.

7.2. Inicio y desarrollo del proyecto

Una vez contrastados los requisitos y teniendo decidido el stack tecnológico a utilizar, podemos comenzar con el desarrollo propiamente dicho del proyecto.
Primeramente y antes de programar nada, debemos definir un estilo visual para la aplicación, así como las pantallas a diseñar y la distribución de los elementos que las componen, como botones o barras de búsqueda, entre otros. Para ello hice uso de la plataforma de desarrollo de prototipos Figma, en la que planteé un diseño inicial que ha sido tomado como un wireframe a color con la agrupación de las funciones por pantallas según su naturaleza, además de sentar las bases de cómo se debe ver visualmente la aplicación.

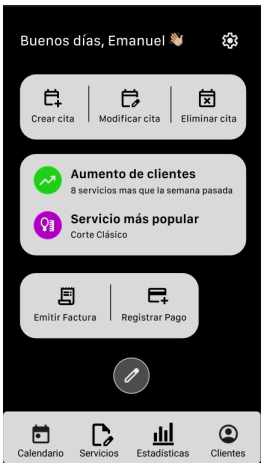

Fig. 3: Pantalla de inicio planteada en el prototipo inicial

Con esto ya definido y discutido con Emanuel, me encargué de crear el proyecto en WebStorm para poder iniciar el desarrollo del proyecto. Primero de todo creé una plantilla, sobre la que instalé todos los frameworks mencionados anteriormente. Una vez realizada una prueba inicial de que todo funcionaba correctamente implementé una estructura de carpetas con la que separar cada funcionalidad. En base a esto se realizó el desarrollo de la aplicación para las siguientes funcionalidades:

- Creación de la estructura de pantallas, en la que contamos con una barra de navegación con cinco accesos rápidos: Principal, Calendario, Servicios, Estadísticas y Perfil.
- Se crea en la pantalla inicial un menú rápido en el que se podrá desarrollar a futuro una biblioteca de Widgets [23] que ofrecerán accesos rápidos o información de interés como nivel de ocupación de los barberos, por ejemplo.
- Se crea el caso de uso de Crear cita, que consta de tres pantallas: una que abre una lista de todos los clientes disponibles en la aplicación, una vez seleccionado uno accedemos a la carta de servicios para seleccionar que quiere el cliente, acompañado de un subtotal en la parte inferior. Haciendo uso del botón **Continuar** llegamos a la pantalla final, donde contamos con un calendario mensual donde escoger el día, para posteriormente poder ver los huecos disponibles y los barberos que realizan ese servicio.
- Se crea la pantalla de usuario, inicialmente solo se mostraba la información básica del usuario y un botón para **cerrar sesión.**
- Se crea la pantalla de Servicios, en la que se muestra la lista de servicios ofrecidos por la barbería. A esto se añade posteriormente en la siguiente iteración las funcionalidades de **refrescar la página** mediante un servicio de polling, crear servicios, modificarlos o eliminarlos.
- Se crea la pantalla de Calendario, en la que contamos con un calendario de tipo carrusel en el que, al seleccionar un día determinado, podemos consultar las reservas a nuestro nombre, con la opción de poderse eliminar o marcar como reserva ya completada.
- Se modifica la pantalla de usuario para permitir editar toda la información referente a él, inclusive la disponibilidad de horas que tiene semanales y los servicios que domina.
- Se desarrolla un modo para daltónicos activable nada más abrir la aplicación o bien desde el menú de Perfil, con tal de facilitar su uso visual a personas con esta enfermedad.

Para cada uno de estos puntos se ha seguido el mismo procedimiento de desarrollo: primero se crea el endpoint cumpliendo con aquello que se especifica en el caso de uso. Una vez contamos con esta parte ya desarrollada y probada mediante **Postman** [24], comenzamos con la creación de los elementos del frontend para gestionar y mostrar la información del backend.

7.3. Etapa final: User Testing, cambios finales, Testing y Deploy

Una vez contamos con el desarrollo de la aplicación completado, preparé y realicé unas sesiones de user testing. Esto me permitió recoger opiniones e información sobre la facilidad de uso y la accesibilidad de los casos de uso básicos de la aplicación. Basándonos en esta información se realizaron ciertos ajustes visuales a la aplicación que subsanan las problemáticas encontradas.

Con todo desarrollado, falta probar a nivel de código la robustez de la aplicación. Debido a que el proyecto abarca una cantidad de horas determinada y se ha intentado llegar a todas las áreas del desarrollo, decidí realizar test de funciones críticas, y acabé desarrollando test para el AuthStore, es decir, para la gestión de la información del usuario, aspecto que considero que destaca por la importancia y confidencialidad de los datos que maneja. Por tanto, se procede a usar la plataforma **Vitest** para la generación de **trece pruebas unitarias**. Para el desarrollo de cada una de ellas se usa la **estructura AAA** [25]. Esta divide el desarrollo en tres partes:

- **Arange:** Preparamos todos los elementos necesarios para la prueba, como variables con datos de prueba o Mock Objects.
- **Act:** Se ejecuta el elemento sujeto a prueba, y se almacena el resultado de la función a probar
- **Assert:** Se comprueba que el resultado es el esperado inicialmente.

Basándonos en esta estructura, queda claro en todo momento qué fase del test estamos realizando, ordenado de manera lógica, siendo fácil de leer y entender.

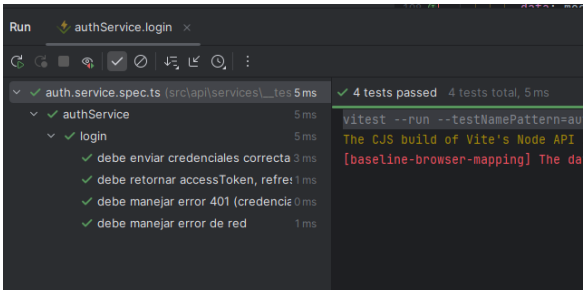

Fig. 4: Captura que muestra los tests realizados, y marcados como correctos

Contando también con la validación de las pruebas implementadas, pasamos al campo final. Para ello se utiliza una instancia de Azure App Service [14] configurada con un App Service que se encarga de alojar el servicio corriendo en la nube, y por otro lado un Azure Cosmos DB que aloja la **base de datos no relacional**. Con esto configurado y probado, se cambió de nuevo la configuración del frontend desde el fichero **.env** para apuntar a la nueva URL del servicio en la nube. Ya contamos con una aplicación funcional, aun no como propuesta de mercado pero sí como paso previo a su lanzamiento.

8 Aspectos Técnicos

8.1. Estructura de directorios del proyecto

Para la correcta compartimentación de los archivos de manera que quedan agrupados según su funcionalidad:

- **/api**: Cuenta con todo aquello encargado de la gestión de la comunicación con el backend mediante el uso de nuestra API RESTful. Dentro alojamos varios archivos y un directorio:
  - **/services:** Contiene los servicios agrupados según el dominio, que tal y como figuran en la API son:
    - **auth.service.ts**: contiene los endpoints para realizar login, logout, generación del JWT, y su renovación mediante el uso de un Refresh Token.
    - **client.service.ts**: contiene el endpoint getAllClients, que se encarga de obtener a todos los usuarios clientes de la base de datos. Útil para el caso de uso Crear Cita, para poder asignar esta a un cliente.
    - **company.service.ts**: Engloba todas las operaciones CRUD sobre los servicios (Crear, modificar y eliminar servicios), obtener los servicios asociados a una barbería, y otro para obtener toda la información de una barbería dado su id.
    - **reservation.service.ts**: Contiene los endpoints relacionados con la gestión de las reservas.
      - **getAvailableDays**: dadas una fecha de inicio y una fecha final, sumado a los servicios a realizar, devuelve una lista de los días con algún hueco horario disponible. Útil para la generación del calendario en el caso de uso Crear Cita.
      - **getAvailableSlots**: dado un día, y una lista de servicios a realizar, devuelve los huecos horarios disponibles ese día, además de información del barbero disponible.
      - **createReservation**: Facilitando el id de cliente, una fecha y hora determinada y los servicios a realizar, crea una reserva en la base de datos.
      - **getReservationsByEmployee**: Dada una fecha de inicio y otra de final, busca todas las reservas asociadas a un barbero en el intervalo especificado.
      - **deleteReservation**: Facilitando un identificador de reserva, se encarga de eliminarla de la base de datos.
      - **completeReservation**: Una vez completada una reserva, esta puede ser marcada por el barbero como realizada. Este endpoint, dado un identificador de reserva, se encarga de cambiar el campo de status a 'Completed'.

    - **user.service.ts**: Contamos con los endpoints para la gestión de la información de usuario. Por una parte, tenemos un endpoint mediante el cual obtener toda la información de usuario facilitando un identificador de usuario. Finalmente, para la actualización de los datos tenemos un endpoint para poder alterar los valores.
  - **client.ts**: Configura una instancia de Axios que se utiliza desde la aplicación para poder enviar las peticiones HTTP.
  - **interceptor.ts**: Interceptor que se encarga de gestionar las peticiones, y de la gestión y renovación de los tokens, e interpretación de códigos de error específicos para poder ser procesados.

- **/components:** agrupa todos los componentes de Vue desarrollados para la aplicación. Esta es una funcionalidad potente, que permite que mediante una sola creación, instanciar un mismo elemento múltiples veces, con tal de reutilizar elementos a la vez que ahorramos código redundante.
- **/composables:**
  - **useServiceRefresh.ts:** Se encarga de encapsular toda la lógica detrás de la función de refresh de la pantalla **servicios**. Allí, al hacer scroll hacia abajo la pantalla llama a un endpoint dedicado que devuelve la lista de servicios, y por tanto actualiza el frontend. En caso de no refrescarse manualmente, se ha implementado un sistema de polling establecido para que cada 10 minutos, y si se está utilizando la aplicación la lista se actualiza.
- **/router:** Contiene el Vue Router de la aplicación. En Vue, para la gestión de las pantallas se establece un árbol que contiene la estructura jerárquica de pantallas para ser accedidas mediante dominio.
- **/stores:** Dentro de Vue, se utiliza una tecnología llamada Pinia. Esta se encarga de gestionar el almacenamiento temporal de la aplicación para mantener la información a través de las pantallas. Para nuestro caso, contamos con dos de ellos: CompanyStore para almacenar toda la información referente a la compañía; y AuthStore para almacenar todos los datos del usuario que inicia sesión.
- **/types:** Encontramos las interfaces de datos para toda la aplicación. Por tanto, tenemos la estructura de la compañía, del usuario usa la aplicación y del cliente.
- **/utils:** Configura una funcionalidad de Capacitor para el almacenamiento de información local en el dispositivo, que nosotros hemos llamado secureStorage.
- **/views:** Almacena todas las pantallas de la aplicación. Cada uno de los archivos contiene la parte visual, que combina elementos de HTML con etiquetas propias de Vue o de Ionic para generar la vista, y una etiqueta de script donde se encuentra toda la lógica de la pantalla, como es la gestión del routing, la funcionalidad de los elementos dinámicos de las pantallas, o las llamadas asíncronas a la API.

8.2. Estructura de la base de Datos

La base de datos escogida es la que ya contábamos desarrollada por parte de Emanuel para el backend. Contamos con una base de datos no relacional de **MongoDB** que consta de **tres colecciones:** users, reservations y companies.

- **users:** Contiene la información de los usuarios, tanto trabajadores como clientes. Cabe destacar que para realizar esta separación contamos con un atributo **role** que los encasilla como uno u otro. A su vez, dependiendo del tipo contienen campos diferentes. Es el caso del trabajador, que cuenta con availability, services, isDefault y companyId como campos añadidos.


```
_id: "2b3f0e9b-a4c8-49c4-9346-8c7e5c80b870"
name : "Josep"
status : "active"
password : "$2b$10$MGYtehYZCvwwzY4.RS0OvehuZnovCYCqG8cEA9fegSaSC9dBy6.dG"
email : "juanpablo@ejemplo.com"
phone : "+34653541778"
isDefault : false
▾ services : Array (2)
    0: "f029fbe9-b7ff-4b08-8c28-042a0fa0bd82"
    1: "59f49e05-46d8-40a9-89df-c64eb7554dcd"
▾ availability : Object
  ▸ monday : Object
  ▸ tuesday : Object
  ▸ wednesday : Object
  ▸ thursday : Object
  ▾ friday : Object
    ▾ slots : Array (2)
      ▾ 0: Object
          start : "09:00"
          end : "09:30"
      ▾ 1: Object
          start : "09:30"
          end : "10:00"
  ▾ saturday : Object
    ▾ slots : Array (empty)
  ▾ sunday : Object
    ▾ slots : Array (empty)
companyId : "069e8bf8-a505-4581-908c-84e55ad05466"
role : "employee"
createdAt : 2025-11-30T16:41:28.205+00:00
updatedAt : 2026-01-15T17:54:26.701+00:00
__v : 0
refreshToken : "$2b$10$ygrUKj.JXFKf4CYaqkxiIuCCbgYgIK17ehjji6XS/rIY3Sk4wO/l."
```


Fig. 5: Ejemplo de un usuario de tipo **Empleado**

- **reservations:** Cuenta con las reservas realizadas. Cada reserva lleva asociado su id de compañía, el id de cliente, los servicios a realizar, la fecha y hora concertada, la fecha y hora de la realización de la reserva, la duración total, el empleado agendado y un campo de **status**. Este al crearse la reserva por defecto figura como **confirmed**, pero una vez el servicio se realiza, el barbero puede cambiar el status a **completed** para gestionar el flujo de las reservas correctamente.
- **companies:** Contiene la información básica de la barbería. Aquí se almacenan los servicios que se realizan, los empleados asociados, fotos del negocio, fecha de creación del perfil, id y **status** del negocio.

9 Resultados

9.1. Aplicación desarrollada

El desarrollo del proyecto BarberManager ha dado lugar a una aplicación móvil funcional orientada a la gestión de barberías para negocios que quieran una adaptación lo más suave posible, y tal y como se ha dado por acabado el desarrollo, podemos decir que se han cumplido los objetivos planteados en las fases iniciales del proyecto.

Primeramente y como resultado principal, se ha desarrollado una aplicación híbrida multiplataforma (Android, iOS y PWA) que permite a los profesionales de una barbería gestionar de manera centralizada aspectos de su

negocio como las citas, servicios horarios y fotos de perfil. Además, para el desarrollo se han utilizado tecnologías y frameworks ampliamente utilizados actualmente en el sector industrial y con un amplio recorrido comercial, como Ionic y Vue con un backend basado en NestJS, consumiendo una API REST desplegada en un entorno cloud real. Esto, unido con técnicas de diseño de código avanzadas, como la arquitectura hexagonal, que permiten contar con una arquitectura con un grado de acople bajo, fácilmente mantenible y abierta a una escalabilidad futura, y por tanto asemejándose lo máximo posible a un desarrollo real en una empresa actual, siempre teniendo en cuenta del tiempo del que se ha dispuesto.

Todas las funcionalidades críticas definidas en el plan inicial han sido implementadas y probadas con un nivel de éxito aceptable. Podemos definir estas funcionalidades como las siguientes:

- Creación y eliminación de citas
- Gestión de servicios
- Visualización de un calendario de citas
- Gestión del perfil de usuario del trabajador
- Modo de accesibilidad para personas con daltonismo

9.2. User Testing

Dentro del desarrollo y aunque muchas veces no es apreciado como la herramienta que acaba siendo, se realizaron sesiones de user testing con varios usuarios externos al desarrollo del proyecto, alguno con un perfil similar al objetivo (una peluquera puede ser considerada del mismo ámbito profesional que un barbero) y otros no, pero permitió comprobar aquellos elementos visuales que se desarrollaron por la mentalidad de un ingeniero, y que pese a parecer en muchas veces sencillos o auto explicativos, nos evidencia que un producto siempre está sujeto al trasfondo de los clientes.

Tal y como se especifica en el curso **Introducción a la UX design ofrecido por Google** [27], debemos establecer un público objetivo de lo más variado posible, que cubren múltiples perfiles y que por tanto nos permitan ver aquellos puntos que pueden ser un escollo, como por ejemplo para personas de una edad más avanzada que no tienen la facilidad tecnológica que una persona joven podría tener. O para personas con daltonismo, que pueden confundir colores que por defecto asociamos a ciertas acciones, como que un botón de **eliminación suele ser rojo**.

Los resultados obtenidos evidenciaron que:

- La navegación entre pantallas resulta intuitiva para usuarios sin experiencia técnica
- Los casos de uso principales se completan sin necesidad de ayuda externa
- La disposición de los elementos visuales facilita la comprensión de la aplicación

No obstante, hubo ciertos casos en los que la realización de estas tareas fue un poco más larga o rebuscada de lo debido, y como consecuencia se realizaron ajustes visuales y de interacción, mejorando la claridad de los botones, lo que contribuye a una experiencia de usuario más accesible, amigable y coherente.

9.3. Accesibilidad a través del desarrollo

Dentro del desarrollo de esta aplicación, la accesibilidad no se ha tratado como una capa final añadida al conjunto del desarrollo, sino como un criterio transversal durante todas las fases de desarrollo. Para ello, si bien no se ha seguido una guía de estándares, ha sido una oportunidad inmejorable para aplicar todos aquellos principios que durante el grado se han visto, en especial en la asignatura **Disseny del Software**. Esto se traduce en decisiones concretas como las siguientes:

- Uso de tamaños de fuente diferenciados, y seguir una agrupación lógica de los elementos según la pantalla
- Uso de elementos como **ion-toast** mediante los cuales mostrar confirmaciones de acciones, para que el usuario entienda que el sistema ha respondido correctamente
- Igualmente en el caso contrario, se evita mostrar mensajes de **logging** con lenguaje altamente técnico para el usuario, y en su defecto se muestran mensajes cortos, claros y explicativos
- Uso de iconos autoexplicativos y homogeneización del estilo visual de las pantallas para no mover en la medida de lo posible los accesos de sitio.
- Uso de botones grandes, espaciados entre si.

Aún tomadas estas medidas, y como se esperaba, del User Testing derivaron ciertos aspectos a mejorar, que han sido solventados a continuación:

- En el caso de uso de **Crear Cita**, al seleccionar una fecha dentro del calendario, este no se cierra automáticamente. Esto confunde a varios usuarios, que esperan volver a la pantalla de selección de huecos horarios. Por tanto, se ha corregido y el calendario cierra al seleccionarse un día
- En la pantalla de servicios, para acceder a la pantalla de **Crear Servicio** se creó un boton flotante en la esquina inferior derecha, conteniendo un símbolo positivo. Pese a parecer autoexplicativo desde la perspectiva de desarrollador, el usuario de mayor edad registrado tuvo problemas para entender su significado. Se ha corregido añadiendo un texto donde explicitamente se muestra un ***Crear Servicio*** con tal de evitar posibles confusiones.
- Al acceder a la pantalla de edición del usuario, algunos usuarios realizaban cambios sin presionar el botón de ***Aplicar Cambios*** y volvían a la pantalla anterior. Esto ha sido solucionado cambiando el botón a uno de tipo flotante, y que sea visible durante el **scroll** de toda la pantalla, y por otra parte en caso de detectarse cambios

realizados, se muestra un mensaje que advierte de la pérdida de cambios, y que requiere de volver a realizar de la acción de volver hacia atrás para efectuarse.

9.4. Resultados Técnicos y despliegue

Desde un punto de vista técnico, este proyecto demuestra la viabilidad del desarrollo de una aplicación basándose en un arquitectura moderna con la separaciones de responsabilidades frontend/backend, haciendo uso de patrones de diseño como la **arquitectura hexagonal** o el **patrón Repository**, gestionar de manera segura la autenticación mediante el uso de **JSON Web Tokens** y el almacenamiento local seguro mediante Capacitor.

Asimismo, se han desarrollado 13 pruebas unitarias utilizando **Vitest** para validar el correcto funcionamiento del **módulo de autenticación**, reforzando la robustez del sistema en una de sus áreas más **críticas**.

Por otra parte, esta separación de responsabilidades permite que el backend haya podido ser desplegado como aplicación de manera sencilla en **Azure App Service** [14], permitiendo simular un **entorno de producción real** en el que la aplicación consume recursos directamente desde un servidor alojado en el cloud. Este despliegue nos confirma que la aplicación **no se limita a un simple prototipo académico** trabajando en local, sino que permite conexiones reales de dispositivos móviles y por tanto su **futura escalabilidad** para convertirse en una solución con un recorrido comercial viable.

9.5. Documentación generada

Durante todo el desarrollo del proyecto se ha generado documentación que ha respaldado el desarrollo real de la aplicación. Como tal, toda la información de este desarrollo se ha intentado aglutinar dentro de los informes de progreso facilitados en las entregas correspondientes. Estos recogen tablas de requisitos funcionales, no funcionales, casos de uso con flujos principales y subflujos desarrollados, el prototipo desarrollado en Figma y explicaciones más desglosadas a nivel técnico de las decisiones y cambios tomados durante el desarrollo. Por otra parte, debemos contar con que la aplicación de user testing requiere de una generación adicional de información, tanto para su realización como la que se deriva de su procesado después de haberse realizado. Por tanto, y como documentación externa a la especificada anteriormente, se añaden:

- Plantilla del documento de consentimiento informado necesario para poder tomar grabación de uno de las pruebas.
- Transcripción de la conversación llevada a cabo durante una prueba.
- Guión utilizado como entrevistador para conducir las sesiones
- Hojas de anotaciones con los resultados obtenidos de las pruebas realizadas

10 CONCLUSIONES

El proyecto Barber Manager nos ha permitido desarrollar una solución tecnológica que responde a las necesidades de digitalización de pequeños negocios, concretamente barberías, ofreciendo una alternativa sencilla accesible y enfocada al usuario final.

Los objetivos planteados al inicio del trabajo se han cumplido satisfactoriamente, y de misma manera se ha demostrado que es posible implementar una solución personalizada, sin recurrir a plataformas genéricas complejas ni a grandes inversiones económicas para obtener unos buenos resultados.

Desde un punto de vista académico, este proyecto me ha permitido la oportunidad de poder explorar un paradigma para mí desconocido anteriormente como era el desarrollo como full stack developer, utilizando un framework de desarrollo para el frontend como es Vue, unido a la suma de un conjunto de patrones de diseño de código que han requerido de toda mi capacidad de concentración con tal de entender su funcionamiento, pero que me han dotado de las capacidades de desarrollar código de alta calidad, haciendo uso de arquitecturas limpias y acorde a todo lo visto al anteriormente durante los estudios del grado de ingeniería informática.

Por otra parte, cabe destacar la importancia de la accesibilidad y la usabilidad, aspectos que han influido directamente en el diseño de la aplicación y que en muchas veces no se toman tan en cuenta cómo se deberían. Hablo desde mi perspectiva cuando digo que desde que tuve contacto con este tipo de técnicas, no dejo de ver su potencial como herramientas para el refinamiento de producto y para la adecuación del resultado final a lo que espera el cliente final, y por tanto poder ofrecer un producto exitoso y que satisfacen las necesidades de un público cada vez más diverso, con múltiples necesidades, gustos y capacidades. En este proyecto se ha querido desarrollar tal y como especifica Don Norman en su exitoso libro *El diseño de las cosas cotidianas* [28], partiendo de la premisa de que los errores no deben atribuirse al usuario, sino al propio diseño, y que este debe anticiparse y prevenirlos siempre que sea posible.Para ello se ha trabajado para que la percepción ayude a las personas a orientarse dentro de los casos de uso, añadiendo significantes allí donde sean necesarios para dotar de aún más literalidad a ciertas funcionalidades, y ofreciendo siempre una respuesta a las acciones realizadas por el usuario.

11 USO DE LA IA GENERATIVA

Mentiría si dijese que durante el desarrollo de esta aplicación no he sido un usuario asiduo a esta tecnología, principalmente durante las etapas de desarrollo técnico. Debido al limitado tiempo del que se dispone, y a mi nulo conocimiento de aspectos como el lenguaje TypeScript, frameworks como Ionic o Vue, poder contar con una herramientas tan potente como ChatGPT o Claude AI hace que la adaptación y el aprendizaje sean lo más llevaderos posibles. El uso de la inteligencia artificial se ha centrado principalmente en

tareas de apoyo a la realización del trabajo, para consulta y aprendizaje. Concretamente, estas herramientas se han utilizado para:

- La generación y estructuración de documentación relacionada con el user testing, ayudando a definir preguntas y criterios de evaluación
- La orientación en el desarrollo del backend, permitiéndole comprender la integración de los distintos patrones de diseño
- La generación de ejemplos en TypeScript utilizados como referencia para entender el uso de ciertas funciones
- La integración de componentes y funcionalidades propias de Ionic en Vue
- La resolución de errores de configuración del proyecto

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a mi familia, quien me ha apoyado desde el primer momento durante esta etapa universitaria hasta la consecución de este proyecto. Puedo decir con seguridad que, sin el apoyo que me han prestado, todo este proceso habría sido imposible de completar. Quiero agradecer especialmente a mi madre, quien me ha apoyado incondicionalmente, por ser crítica en los momentos necesarios y por mostrarme su apoyo en los momentos de mayor dificultad. Sin una persona así, el conformismo es algo que acecha a menudo y su ejemplo me ha enseñado a no dejar de luchar nunca por cumplir un objetivo, por muy improbable que parezca.

Gracias, por otra parte, a mi tutor Emanuel Adolfo Vallejo Cañizares. Desde un primer momento vi en este proyecto una propuesta atractiva, respaldada por una persona con una gran motivación y conocimiento, lo que me llevó a un proyecto en el que, pese a partir de un conocimiento limitado de las tecnologías utilizadas, he podido alcanzar un gran aprendizaje y dar como resultado una propuesta real. Gracias por todo el apoyo prestado durante el proyecto, por la disponibilidad ante cualquier duda surgida y por la ayuda recibida para enfocar correctamente el desarrollo.

Finalmente, gracias a mi pareja Claudia. Ella ha sido la mejor persona con la que he podido compartir esta etapa, y sin ella todo este camino recorrido habría sido mucho más duro. Su compañía durante todas las etapas atravesadas ha hecho que incluso los momentos más críticos fueran más llevaderos. Su paciencia, comprensión y ánimo constante han sido un pilar fundamental para no perder la motivación y culminar con la entrega de este proyecto.

REFERENCIAS

[1] GoBarber, *GoBarber: App de gestión de citas para barberías*. [Online]. Disponible en: https://www.gobarber.es/ [Accedido: 16 de enero de 2026].

[2] Timp, *Timp Barberías*. [Online]. Disponible en: https://timp.pro/barberias/ [Accedido: 16 de enero de 2026].

[3] B. Bos, *Design Principles*, World Wide Web Consortium (W3C), 2003. [Online]. Disponible en: https://www.w3.org/People/Bos/DesignGuide/design.html [Accedido: 23 de septiembre de 2025].

[4] Andrea Rosales, *Cuando vinieron a buscar a los analfabetos digitales...*, CCCBLAB [Online]. Disponible en: https://lab.cccb.org/es/cuando-vinieron-a-buscar-a-los-analfabetos-di

[5] BBC Noticias Mundo, *Qué son las puertas Norman, las que hacen que te equivoques cuando quieres entrar o salir*, BBC [Online]. Disponible en: https://www.bbc.com/mundo/noticias-41587542

[6] KanbanTool, *La historia de Kanban*. [Online]. Disponible en: https://kanbantool.com/es/guia-kanban/historia-de-kanban [Accedido: 24 de septiembre de 2025].

[7] Microsoft, *Microsoft Project* [Online]. Disponible en: https://www.microsoft.com/es-es/microsoft-365/project/project-management

[8] Wikipedia, *Diagrama de Gantt* [Online]. Disponible en: https://es.wikipedia.org/wiki/Diagrama_de_Gantt

[9] Microsoft Learn, *Progressive Web Apps (PWA)*. [Online]. Disponible en: https://learn.microsoft.com/es-es/microsoft-edge/progressive-web-apps/ [Accedido: 16 de enero de 2026].

[10] Content Square, *Cómo hacer pruebas de usuario eficaces en 7 sencillos pasos* [Online]. Disponible en: https://contentsquare.com/es-es/guias/user-testing/how-to/

[11] RedHat, *Stateless Systems: definición*. [Online]. Disponible en: https://www.redhat.com/es/topics/cloud-native-apps/stateful-vs-stateless [Accedido: 16 de enero de 2026].

[12] Vitest, *Vitest: Features* [Online]. Disponible en: https://vitest.dev/guide/features.html

[13] Stephanie Susnjara, Ian Smalley, *¿Qué es la plataforma como servicio (PaaS)?*, IBM, [Online]. Disponible en: https://www.ibm.com/es-es/think/topics/paas

[14] Microsoft, *Azure App Service* [Online]. Disponible en: https://azure.microsoft.com/es-es/products/app-service

[15] Wikipedia, *Arquitectura hexagonal (software)*. [Online]. Disponible en: https://es.wikipedia.

org/wiki/Arquitectura_hexagonal_(software) [Accedido: 24 de septiembre de 2025].

[16] B. Nullpoint Excelsior, *Arquitectura hexagonal Parte IV: Patrones de arquitectura sobre la capa Application*. [Online]. Disponible en: https://nullpointer-excelsior.github.io/posts/implementando-hexagonal-con-nestjs-part4/ [Accedido: 24 de septiembre de 2025].

[17] J. DC, *¿Qué es el Patrón Repository para Arquitecturas Limpias?*, Platzi. [Online]. Disponible en: https://platzi.com/blog/patron-repository/ [Accedido: 24 de septiembre de 2025].

[18] D. Palmino, *Desarrollo ágil con Arquitectura limpia Hexa3*, Medium. [Online]. Disponible en: https://sl1nk.com/9uatt [Accedido: 24 de septiembre de 2025].

[19] M. Á. Sanchez, *Value Objects*, Medium. [Online]. Disponible en: https://medium.com/all-you-need-is-clean-code/value-objects-d4c24115fa69 [Accedido: 25 de septiembre de 2025].

[20] J. S. Gual, *Usability and UX*, Material de la asignatura Disseny de Software. [Online].

[21] D. P. Mussarra, *Tècniques d'assistència a l'elicitació per cooperar i consensuar*, Material de la asignatura Requisits de Software. [Online].

[22] Epicmax, *Vue 2 to Vue 3 Migration Guide: Risks, Benefits and Strategies* [Online]. Disponible en: https://epicmax.co/vue-3-migration-guide

[23] Wikipedia, *Widgets* [Online]. Disponible en: https://es.wikipedia.org/wiki/Widget

[24] Javier, *¿Qué es Postman? ¿Cuáles son sus principales ventajas?*, Formadores IT, 2023. [Online]. Disponible en: https://formadoresit.es/que-es-postman-cuales-son-sus-principales-ventajas/

[25] Microsoft Learn, *Conceptos básicos de pruebas unitarias* [Online]. Disponible en: https://learn.microsoft.com/es-es/visualstudio/test/unit-test-basics?view=visualstudio

[26] Jeremy Trujillo, *Mock Objects: uso en pruebas de software*, Medium, 2020 [Online]. Disponible en: https://jeremy-trujillo.medium.com/mock-objects-los-suplantadores-de-clases-7f357c17bce7

[27] Coursera, *Foundations of User Experience (UX) Design*, [Online]. Disponible en: https://www.coursera.org/learn/foundations-user-experience-design

[28] D. Norman, *The Design of Everyday Things*, Basic Books, 2013. [Online]. Disponible en: https://ia902800.us.archive.org/3/items/thedesignofeverydaythingsbydonnorman/The%20Design%20of%20Everyday%20Things%20by%20Don%20Norman.pdf

APÈNDICE: EVIDENCIAS VISUALES DE LA APLICACIÓN

A.1. Primeros pasos

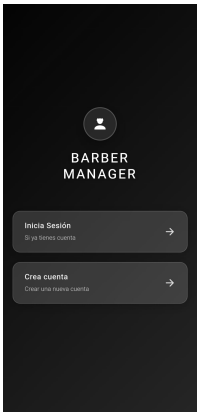

Fig. 6: Pantalla inicial al abrir la aplicación por primera vez

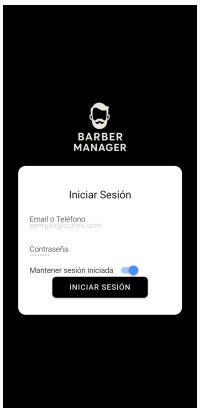

Fig. 7: Pantalla de inicio de sesión

Fig. 8: Pantalla principal de la aplicación

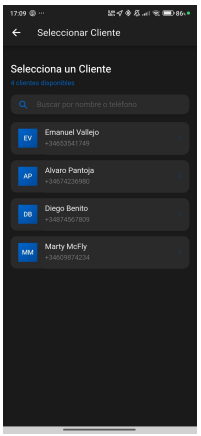

Fig. 9: Lista de clientes

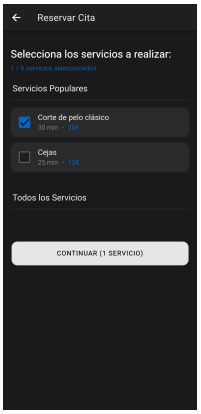

Fig. 10: Pantalla dónde seleccionar los servicios a realizar

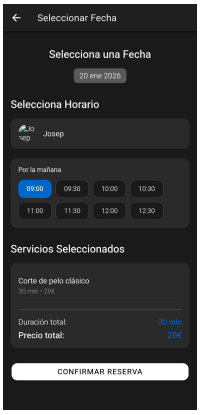

Fig. 11: Selección de dia, hora y barbero, acompañado de un subtotal

A.3. Calendario de citas

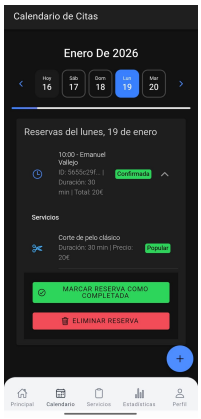

Fig. 12: Vista del calendario de citas, con una cita en estado Confirmed

A.4. Servicios

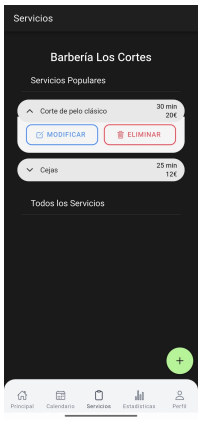

Fig. 13: Pantalla con la carta de servicios

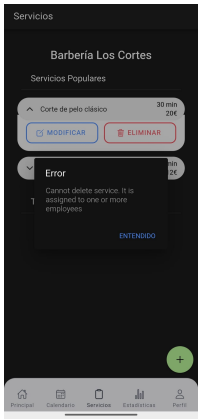

Fig. 14: Mensaje de error: el servicio no se puede eliminar porque esta asignado a un empleado

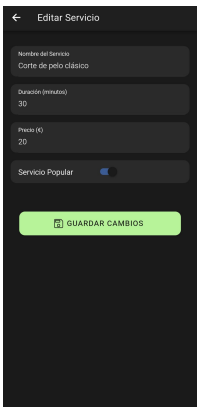

Fig. 15: Editar un servicio determinado

A.5. Gestión de usuario

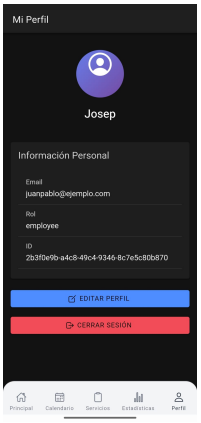

Fig. 16: Información resumida del usuario

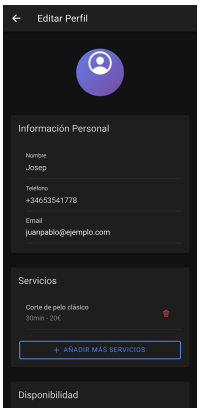

Fig. 17: Editar la información básica del usuario y los servicios que es capaz de realizar

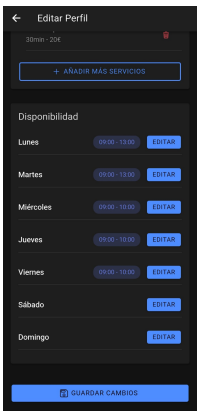

Fig. 18: Edición de calendario de disponibilidad del barbero