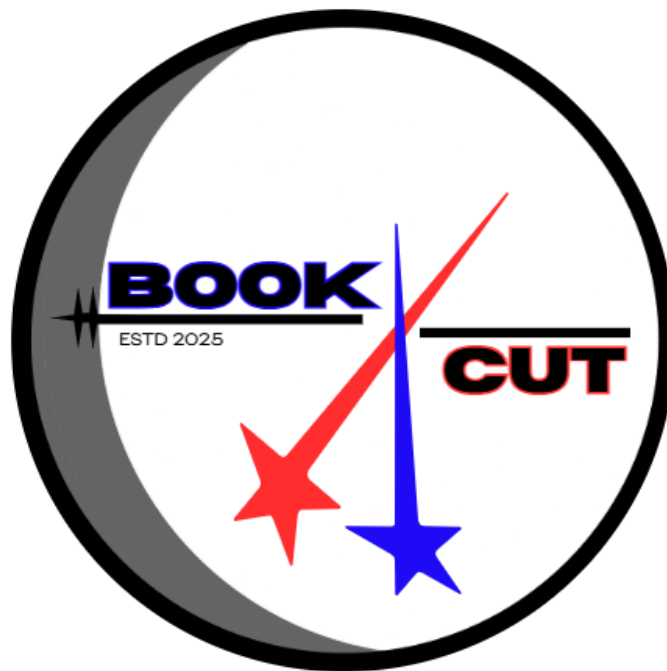


# Book&Cut

*Sistema integral de citas y agendas*



Alumnos: Iván Rubio Murillo , Daniel Moñino García , Luis José Marcano Fuentes

Grupo: DarkMatter



DARKMATTER

Centro: I.E.S Albarregas

Módulo 0492 · Proyecto DAM 2B · 2025/2026

Docente tutor: María Mercedes Martínez Frago

Versión y Fecha: 04/11/2025 v.0.1

Licencia y avisos legales: GPLv3

Enlaces:

[Git-Book&Cut](#)

[Tablero-Trello](#)

[Figma-Book&Cut](#)

[Documentación-Book&Cut](#)

## 2. Resumen Ejecutivo y Alcance de la Demostración

### 2.1. Problema/Oportunidad

**Problema:** Numerosas barberías y peluquerías en Extremadura, especialmente micro-pymes y emprendedores, gestionan sus citas mediante métodos manuales (llamadas, agendas físicas o aplicaciones de mensajería). Esto genera ineficiencias como confusiones, errores, solapamientos y la consecuente **pérdida de tiempo** y oportunidades de negocio.

**Oportunidad:** Desarrollar una aplicación **móvil/web sencilla y accesible** que automatice la gestión de citas. Esta solución permitirá optimizar la agenda, mejorar la comunicación con el cliente y facilitar la **modernización digital** de estos negocios extremeños para aumentar su competitividad.

### 2.2. Propuesta de Solución

La solución es una **Aplicación móvil/web multiplataforma** que ofrece al cliente la capacidad de **reservar, modificar y cancelar citas 24/7 de forma autónoma**. Para el negocio, proporciona un **Panel de Gestión** centralizado de la agenda, servicios, y personal. La aplicación incluye recordatorios automáticos (vía email/push) para **reducir el absentismo** (*no-shows*). El foco principal es la **facilidad de uso y la rapidez operativa** para la pequeña y mediana empresa.

### 2.3. Objetivo de Producto y Objetivo de E1

- **Objetivo de Producto (General):** Ser la herramienta digital de referencia en Extremadura para la gestión de citas en el sector de la belleza, aumentando la eficiencia operativa y la fidelización de clientes locales.
- **Objetivo de E1 (Primera Iteración):** Implementar la **funcionalidad crítica de Reserva de Cita** (HU-005) y construir la **API REST del Panel de Agenda Diaria** (HU-006) en el *backend*, asegurando la persistencia de datos y el cumplimiento de los Requisitos No Funcionales (RNF) mínimos.

### 2.4. Alcance de la Demo de E1 (Qué se muestra y qué no)

**Se muestra:** La funcionalidad básica completa de “**Reserva de Cita**” (flujo de selección de servicio, profesional y hora) y la “**Visualización de la Agenda del Día**” para el administrador. Incluye la Portada, el Resumen y la Justificación completados.

**No se muestra:** Funcionalidades avanzadas como la pasarela de pagos online, gestión de stock/inventario, analíticas de negocio, notificaciones *push* automáticas, o el panel completo de configuración de servicios y personal.

### 3. Justificación

#### 3.1. Beneficios Esperados

- **Técnicos:** Experiencia en desarrollo multiplataforma (Flutter/Spring Boot), integración de APIs de calendario y dominio del diseño arquitectónico monolítico.
- **Educativos:** Aplicación práctica de metodologías ágiles (Scrum/Kanban) y gestión avanzada de repositorios (Git).
- **De Uso:** Ahorro de tiempo significativo para el negocio, reducción de pérdidas por absentismo (*no-shows*) y mejora de la imagen profesional y accesibilidad para el cliente.

#### 3.2. Integración con la Industria Extremeña

El proyecto está enfocado en el sector de la belleza y cuidado personal en Extremadura.

- **Casos de Uso:**
  - **Barbería Emergente (Mérida/Badajoz):** Gestionar el alto flujo de citas en horas pico y distribuir la carga entre varios profesionales.
  - **Peluquería Rural (Pueblo):** Ofrecer un canal digital simple a clientela local, ampliando su alcance.

La aplicación busca ser una **herramienta de digitalización a bajo coste** adaptada a las necesidades de las pequeñas empresas extremeñas.

#### 3.3. Análisis de Productos Similares (Benchmark Breve)

El análisis de mercado posiciona a la aplicación como una alternativa más económica y sencilla que sus competidores:

Producto Similar	Modelo de Negocio	Debilidades (Ventaja Competitiva para nuestro proyecto)
Booksy	Suscripción mensual.	<b>Alto Coste de Suscripción</b> y exceso de funciones innecesarias para una micro-pyme.
Fresha	Freemium (Comisiones por pagos/clientes nuevos).	<b>Comisiones Ocultas</b> por transacciones, que impactan el margen de beneficio.

<b>GoBarber</b>	Precio fijo mensual bajo.	<b>Limitación a apps móviles</b> , haciendo la gestión administrativa desde PC menos accesible.
-----------------	---------------------------	---

### 3.4. Participación en ODS

- **ODS 8 (Trabajo decente y crecimiento económico):** Al digitalizar y optimizar la gestión, ayuda a aumentar la productividad y la sostenibilidad económica de las micro-pymes en Extremadura.
- **ODS 9 (Industria, innovación e infraestructura):** Fomenta la adopción de tecnología en un sector tradicional, impulsando la innovación digital a nivel local.

## 4. Historias de Usuario

### 4.2. Ejemplos de Historias (3–5) con Criterios de Aceptación

#### Búsqueda y Reserva de Cita (HU-005) - MUST HAVE

- **Como Cliente**, quiero poder seleccionar el servicio y el profesional deseado para ver los horarios disponibles y reservar mi cita.
- **Criterios de Aceptación (Ajustados al Flujo):**
  1. **GIVEN** estoy en la **Pantalla de Búsqueda**, **WHEN** selecciono "Corte Masculino" y al Barbero "Javier", **THEN** el sistema me muestra solo las fechas y horas libres de Javier para ese servicio.
  2. **GIVEN** selecciono una hora, **WHEN** confirmo la reserva, **THEN** recibo una confirmación en pantalla y por email.

#### Visualización de Agenda Diaria (HU-006) - MUST HAVE

- **Como Administrador/a**, quiero ver todas las citas reservadas para el día de hoy para tener un control rápido de mi jornada.
- **Criterios de Aceptación (Ajustados al Flujo):** **GIVEN** que accedo al **Panel de Administración (Sección Agenda)**, **WHEN** selecciono la fecha de hoy, **THEN** veo una lista cronológica con la hora, el nombre del cliente y el servicio de cada cita.

#### Cancelación de Cita (HU-007) - SHOULD HAVE

- **Como Cliente**, quiero poder cancelar mi cita a través de un enlace en el email de confirmación para notificar al negocio sin tener que llamar.
- **Como Administrador**, quiero poder cancelar las citas que se soliciten a través de la aplicación.
- **Criterios de Aceptación (Ajustados a Lógica y Flujo):**
  1. **Cliente: GIVEN** pulso el enlace de cancelación, **WHEN** la cita **no ha excedido el límite de cancelación** (política de la barbería), **THEN** el negocio es notificado por email y el hueco se libera automáticamente. (Si se ha excedido el límite, muestra error y no permite la cancelación).

2. **Administrador: GIVEN** pulso la notificación de cancelación que emerge en la aplicación, **WHEN** Confirmo la anulación, **THEN** el cliente es notificado de la cancelación y el hueco queda libre.

#### **Aprobación de Citas Solicitadas (HU-008) - SHOULD HAVE**

- **Como Administrador**, quiero poder aceptar o rechazar las citas que sean solicitadas a través de la aplicación, para tener un control final de la agenda.
- **Criterios de Aceptación (Ajustados al Flujo): GIVEN** accedo a la sección “**Citas pendientes**” en el Panel Admin, **WHEN** selecciono una solicitud y pulso el botón “Aprobar”, **THEN** la cita se integra en la agenda diaria (HU-006) y el cliente recibe una notificación de confirmación.

#### **4.3. Backlog Priorizado (MoSCoW)**

- **MUST HAVE (Imprescindible):** HU-005 (Reserva), HU-006 (Agenda Diaria).
- **SHOULD HAVE (Importante si hay tiempo):** HU-007 (Cancelación de Cita), HU-008 (Aprobación/Rechazo de Citas).

## **5. Arquitectura**

Nuestro objetivo es construir una solución simple pero robusta con una arquitectura minimalista que permita la entrega rápida de valor y mantenga la simplicidad operativa. Todo el sistema funciona localmente utilizando Docker Compose, lo que garantiza un entorno de desarrollo consistente y fácil de replicar.

### **5.1. Diagrama (C1/C2: Contexto y Contenedores)**

El sistema se compone de tres contenedores principales orquestados por Docker Compose y la aplicación cliente. La API REST Spring Boot actúa como un monolito manejando toda la lógica de negocio y accediendo directamente a MySQL mediante JDBC.

- **Usuario:** Interactúa con la aplicación.
- **App Flutter/Android:** Interfaz de usuario nativa encargada de realizar llamadas REST sobre HTTPS + JSON a la API.
- **API REST Spring Boot Monolito:** Expone los endpoints y contiene la lógica de negocio (validación de citas).
- **MySQL:** Base de datos relacional para almacenar citas, profesionales y servicios.
- **Adminer (Opcional Dev):** Herramienta para visualizar y editar la base de datos durante el desarrollo.

## 5.2. Decisiones de Arquitectura (ADR)

Hemos definido 10 ADRs fundamentales para guiar el desarrollo , garantizando la coherencia arquitectónica y proporcionando pasos concretos para la implementación.

### 5.2.1. ADR-001: Arquitectura Monolítica con Spring Boot

- Decisión: Construir un único proyecto Spring Boot que expone endpoints REST y maneja todas las operaciones en un solo artefacto desplegable.
- Motivo: Es la opción más simple de desplegar, entender y mantener para un equipo pequeño de DAM. Este enfoque no impide una futura migración a microservicios.
- Implementación (Ejemplo): Organizar el código en paquetes claros: controller/, service/, dao/, model/ y dto/.

### 5.2.2. ADR-002: Seguridad Simplificada con API Key

- Decisión: Para la Iteración 1, usar un sistema de autenticación simple basado en una clave de API fija (X-API-KEY).
- Motivo: Evita el bloqueo por la complejidad de implementar JWT completo, permitiendo dedicar el tiempo a la funcionalidad core (Reserva de Citas, HU-005).
- Implementación (Ejemplo): Implementar un filtro servlet que valide la cabecera X-API-KEY y retornar código HTTP 401 si la clave es inválida.

### 5.2.3. ADR-003: Acceso a Datos con Spring JDBC Template

- Decisión: Utilizar Spring JDBC Template para el acceso a datos en lugar de JPA/Hibernate.
- Motivo: Permite escribir SQL explícito , lo que es sencillo para equipos en formación y evita consultas ineficientes generadas automáticamente.
- Implementación (Ejemplo): Crear clases DAO (Data Access Object) con métodos CRUD y usar SQL parametrizado para prevenir inyección SQL.

### 5.2.4. ADR-004: Versionado de Esquema con Flyway

- Decisión: Implementar Flyway para la migración de la base de datos.
- Motivo: Garantiza que todos los miembros del equipo trabajen con exactamente la misma estructura de base de datos, eliminando problemas de desincronización.
- Implementación (Ejemplo): Documentar los cambios de esquema en archivos SQL secuenciales ubicados en src/main/resources/db/migration/.

### 5.2.5. ADR-010: Despliegue Local con Docker Compose

- Decisión: Crear un archivo docker-compose.yml para orquestar los servicios MySQL, la API Spring Boot y, opcionalmente, Adminer.

- Motivo: Permite levantar todo el stack completo con un solo comando (docker-compose up) , asegurando un entorno de desarrollo idéntico para todos.
- Implementación (Ejemplo): Configurar el servicio MySQL con un volumen persistente para los datos y utilizar un usuario app\_user con permisos limitado

### 5.3. Integraciones, Datos y Dependencias

- Tecnologías de Stack: Spring Boot (Backend), MySQL (BD) , App Flutter (Frontend).
- Datos Sensibles:
- Versionado de API: Todas las rutas usarán el prefijo /api/v1/ para permitir futuras evoluciones sin romper clientes existentes.

Riesgo Técnico	Impacto	Mitigación
<b>Desincronización de BD</b>	Fallos en el arranque o errores de SQL en entornos de desarrollo.	Uso obligatorio de <b>Flyway (ADR-004)</b> para aplicar migraciones automáticamente al inicio.
<b>Complejidad de Seguridad</b>	Bloqueo en la implementación de la funcionalidad <i>core</i> .	Uso de <b>API Key fija (ADR-002)</b> para simplificar la seguridad inicial.
<b>Inyección SQL</b>	Vulnerabilidad de la aplicación debido al uso de SQL explícito.	Uso de <b>SQL parametrizado</b> en todos los métodos del DAO con Spring JDBC Template (ADR-003).
<b>Exposición de Credenciales</b>	Fuga de datos sensibles al subir a repositorios públicos.	Uso estricto de <b>Variables de Entorno</b> (ADR-008) y no registrar contraseñas/tokens en los logs (ADR-009).

## 6. Requisitos No Funcionales (RNF) Definidos y Verificables

RNF	Descripción y Categoría	Métrica, Umbral y Método de Verificación	Trazabilidad (Vínculo a HU / ADR)
<b>RNF-001</b>	Latencia de Reserva (Rendimiento). La operación crítica de reserva de cita debe ser rápida para el usuario final.	<b>Métrica:</b> Tiempo de respuesta del <i>endpoint</i> POST /api/v1/reservas. <b>Umbral:</b> Debe ser < 500 ms. <b>Método:</b> Medición con herramientas de rendimiento ( <i>benchmarking</i> ) en pruebas de integración.	<b>HU-005</b> (Búsqueda y Reserva)
<b>RNF-002</b>	Autenticación (Seguridad). Toda petición al <i>backend</i> debe ser validada.	<b>Métrica:</b> Número de peticiones bloqueadas por falta de clave / Peticiones totales. <b>Umbral:</b> 100% de las llamadas API deben validar la clave. <b>Método:</b> Pruebas funcionales negativas sobre el filtro del <i>Controller</i> .	<b>ADR-002</b> (Seguridad API Key)
<b>RNF-003</b>	Carga de Agenda (Rendimiento). El administrador necesita visualizar rápidamente la agenda diaria.	<b>Métrica:</b> Tiempo de carga del <i>endpoint</i> GET /api/v1/agenda/hoy. <b>Umbral:</b> Debe ser < 2.0 segundos. <b>Método:</b> Pruebas funcionales en el entorno de desarrollo.	<b>HU-006</b> (Visualización Agenda)



<b>RNF-004</b>	Integridad del Esquema (Mantenibilidad). El esquema de la base de datos debe ser consistente entre entornos.	<b>Métrica:</b> Estado de la verificación de Flyway. Umbral: La aplicación debe arrancar con el mensaje Flyway Succeeded. Método:	<b>ADR-004</b> (Flyway)
----------------	--	---	-------------------------

		Verificar los <i>logs</i> de inicio del servicio Spring Boot en Docker.	
<b>RNF-005</b>	Consistencia de Errores (Usabilidad/Mantenibilidad). Los errores de validación de entrada deben ser predecibles para Flutter.	<b>Métrica:</b> Formato de la respuesta JSON ante un error de validación (código HTTP y estructura JSON). Umbral: 100% de los errores de validación deben devolver un código HTTP 400 con formato JSON. Método: Pruebas de integración enviando DTOs inválidos.	<b>ADR-005</b> (Manejo de Errores JSON)
<b>RNF-006</b>	Cifrado de Credenciales (Seguridad). Las contraseñas no deben ser almacenadas en texto plano.	<b>Métrica:</b> Verificación de las contraseñas en la tabla de usuarios de MySQL. Umbral: 0 contraseñas deben estar visibles en texto plano. Método: Inspección manual de la base de datos.	<b>ADR-008</b> (Variables de Entorno)

## Plan de Pruebas para Historias de Usuarios E1

Este plan se centra en las funcionalidades básicas y críticas del flujo de la aplicación. **Asumiré las siguientes funcionalidades como imprescindibles** basándome en las capturas de pantalla, las cuales deberás validar o ajustar con las HUs reales de tu proyecto.

ID HU Asumida	Descripción Breve (Flujo)	Criterios de Aceptación (Ejemplo)	Casos de Prueba (ID)
HU-E1-001	Registro/Creación de Cuenta	El usuario puede registrarse completando todos los campos requeridos y recibe una confirmación exitosa.	CP-R-001, CP-R-002
HU-E1-002	Inicio de Sesión (Login)	El usuario puede ingresar exitosamente con credenciales válidas y es redirigido a la pantalla principal ("Me...").	CP-L-001, CP-L-002
HU-E1-003	Visualización de Perfil	El usuario puede acceder a su perfil y ver/editar sus datos personales.	CP-P-001, CP-P-002
HU-E1-004	Solicitar Cita (Flujo Principal)	El usuario puede completar el flujo de solicitud de cita y ver la cita programada en su lista de citas ("Cita...").	CP-C-001, CP-C-002

<b>HU-E1-005</b>	<b>Visualización de Listado de Citas</b>	El usuario puede ver un listado de sus citas pendientes y pasadas.	CP-LC-001
------------------	--	--	-----------

### Casos de Prueba Detallados (1+ por HU)

A continuación, se detallan ejemplos de casos de prueba para cada HU imprescindible:

ID Caso	HU Asociada	Módulo	Objetivo de la Prueba	Pasos de Ejecución	Resultado Esperado
<b>CP-R-001</b>	HU-E1-001	Registro	Verificar el registro exitoso con datos válidos.	1. Acceder a la pantalla de registro. 2. Rellenar todos los campos (nombre, email, contraseña, etc.). 3. Presionar el botón "Registrarse".	El sistema crea la cuenta y redirige al usuario al <i>dashboard</i> o muestra mensaje de éxito.

<b>CP-L-001</b>	HU-E1-002	Login	Verificar el inicio de sesión con credenciales correctas.	1. Acceder a la pantalla de Login. 2. Ingresar Email y Contraseña válidos. 3. Presionar el botón "Ingresar".	El usuario accede a la pantalla principal de la aplicación.
<b>CP-P-001</b>	HU-E1-003	Perfil	Verificar la visualización correcta de los datos del perfil.	1. Iniciar sesión. 2. Navegar a la sección "Me..." (Perfil).	Se muestran los datos del usuario (Nombre, Email, etc.) sin errores de formato.
<b>CP-C-001</b>	HU-E1-004	Citas	Verificar la programación exitosa de una nueva cita.	1. Iniciar sesión. 2. Navegar al flujo de creación de citas ("Cita..."). 3. Seleccionar servicio, fecha y hora. 4. Confirmar la cita.	La cita se guarda y el usuario recibe un mensaje de confirmación.

<b>CP-LC-001</b>	HU-E1-005	Citas	Verificar la correcta carga del listado de citas.	1. Iniciar sesión. 2. Navegar a la sección de listado de citas ("Cita...").	El listado muestra las citas programadas previamente, ordenadas y con el estado correcto.
------------------	-----------	-------	---	---	---