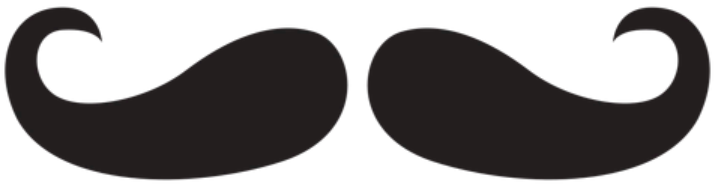
BookMyCut

PROYECTO DE DESARROLLO DE APLICACIONES WEB

Curso académico 2025/2026 - CFGS 2°DAW

Daniel Vilar Martínez



Índice

Descripción del proyecto

Como propuesta de proyecto tengo el objetivo de desarrollar una aplicación web llamada **BookMyCut** se trata de una aplicación pensada para que peluquerías o centros de estética sean capaces de gestionar sus reservas de manera online y también proporcionar información sobre los servicios que ofrezcan,

La aplicación permitirá a los clientes reservar citas con estilistas, gestionar su perfil, recibir recordatorios de citas próximas y etc. El sistema contará con diferentes roles, como clientes, estilistas y administradores, cada uno con acceso a funcionalidades específicas. Los administradores tendrán la capacidad de gestionar al resto de usuarios, las citas y la disponibilidad de los estilistas, las fechas disponibles de la peluquería y más. Los clientes podrán gestionar sus citas, cancelarlas y modificar sus datos personales. Los estilistas podrán especificar los servicios que ofrecen y consultar las citas programadas en el calendario.

El sistema estará diseñado para ser sencillo, fácil de usar y práctico. La interfaz de usuario tendrá un diseño responsive y se basará en un estilo minimalista y elegante utilizando sobre todo los colores blanco y negro. Para mejorar la experiencia del usuario se añadirán notificaciones para recordar las citas programadas y más funcionalidades para mejorar la accesibilidad.

He elegido está idea para mi proyecto final porque mi principal objetivo era crear una aplicación que pudiera usarse en una empresa real, que fuera sencilla y sin mucha complejidad. Muchas de las aplicaciones con aplicaciones reales ya existen de una manera profesional, y la mayoría se escapan de la complejidad de este proyecto. El concepto de una aplicación para reservar citas, en este caso una peluquería, es simple y podría utilizarse en algunos pequeños negocios particulares que requieren poca cosa.

Stack tecnológico

Para este proyecto decidí usar un stack que ya me era familiar por los trabajos y prácticas que hicimos en clase. **Spring Boo**t como es obvio lo he utilizado porque es con lo que hemos trabajado durante diversas prácticas. **Angular** me resultó cómodo para el frontend aunque en las prácticas he estado con **React** y tengo que decir que me he ido confundiendo un poco al trabajar por un lado con uno y por otro lado con otro.

También opté por **Bootstrap** y **MySQL** porque son herramientas que ya habíamos usado bastante en clase. **Bootstrap** me permitió hacer el diseño y que la app fuera responsive sin perder tiempo en CSS, y **MySQL** me parecía lo más natural para gestionar los datos. Para la autenticación utilicé **JWT** y la comunicación entre frontend y backend la hice mediante una **API REST**, ya que es la arquitectura que llevamos viendo todo el curso y encajaba con mi aplicación. Además, utilice **WebSockets** para enviar notificaciones en tiempo real, evitando que el usuario tenga que refrescar la página constantemente. Por último, elegí Docker para el despliegue, porque facilita ejecutar todos los servicios juntos y asegura que la aplicación funcione igual en cualquier entorno.

Requisitos funcionales

RF01 - El sistema deberá tener un sistema de login mediante username/email y contraseña, autenticando al usuario y generando un token JWT para sesiones seguras.

RF02 - El sistema deberá tener un sistema de logout que invalida el token de autenticación del cliente.

RF03 - El sistema deberá permitir el registro de usuarios, asignando automáticamente el rol de CLIENTE a los usuarios registrados mediante el formulario público. Los administradores pueden crear usuarios con cualquier rol desde el panel de administración.

RF04 - El sistema deberá permitir editar el perfil de usuario, permitiendo visualizar y editar la información personal (nombre) y cambiar la contraseña requiriendo la contraseña actual como validación.

RF05 - El sistema deberá permitir a un cliente reservar una cita con antelación, seleccionando estilista, servicios, fecha y hora disponible.

RF06 - El sistema deberá mostrar el calendario a la hora de reservar con los días disponibles y actualizarse automáticamente, basándose en la disponibilidad del estilista seleccionado y excluyendo horas ya ocupadas.

RF07 - El sistema deberá permitir a los clientes cancelar o modificar sus citas siempre que no hayan pasado.

RF08 - El sistema deberá enviar un recordatorio para una reserva que esté próxima mediante notificaciones automáticas programadas.

RF09 - El sistema deberá permitir a un estilista/administrador ver las citas según unos filtros (fecha, estilista, cliente, servicio, estado).

RF10 - El sistema deberá permitir a un administrador cancelar/modificar citas, bloquear fechas mediante excepciones de horario, y además poder añadir o eliminar a otros usuarios (estilistas o clientes).

RF11 - El sistema deberá permitir gestionar la disponibilidad de los estilistas por día de la semana con horarios de inicio y fin.

RF12 - El sistema deberá permitir a los estilistas gestionar los servicios que ofrecen, asociándose a servicios existentes o creando nuevos.

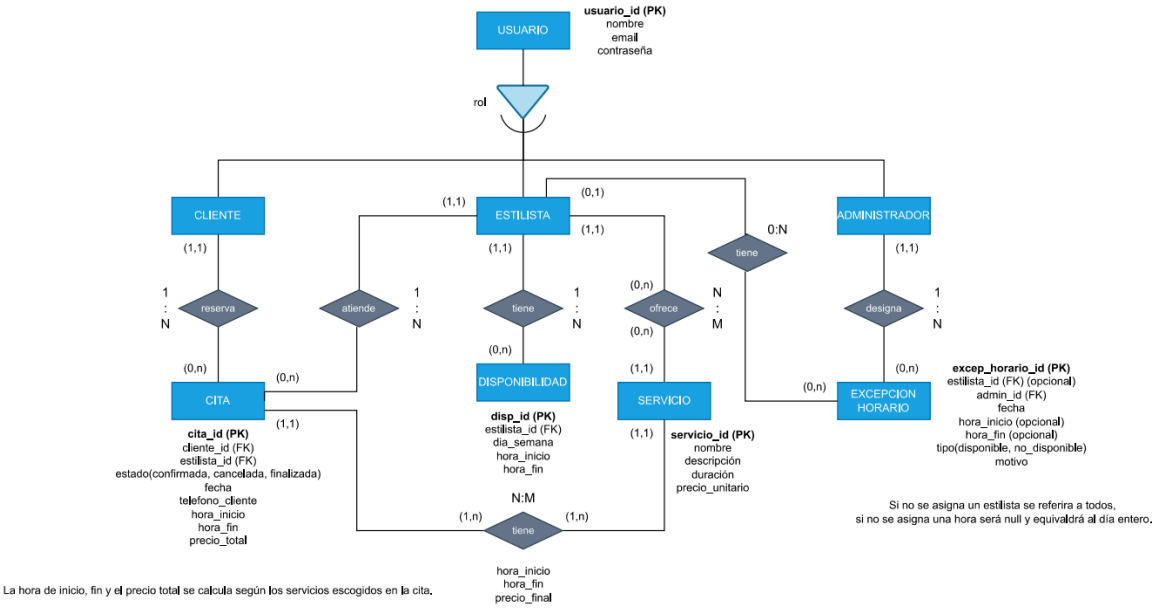
RF13 - El sistema deberá permitir a un cliente ver su historial de citas anteriores con información completa.
RF14 - El sistema deberá permitir alternar entre el modo claro y oscuro mediante un sistema de temas implementado en la aplicación.

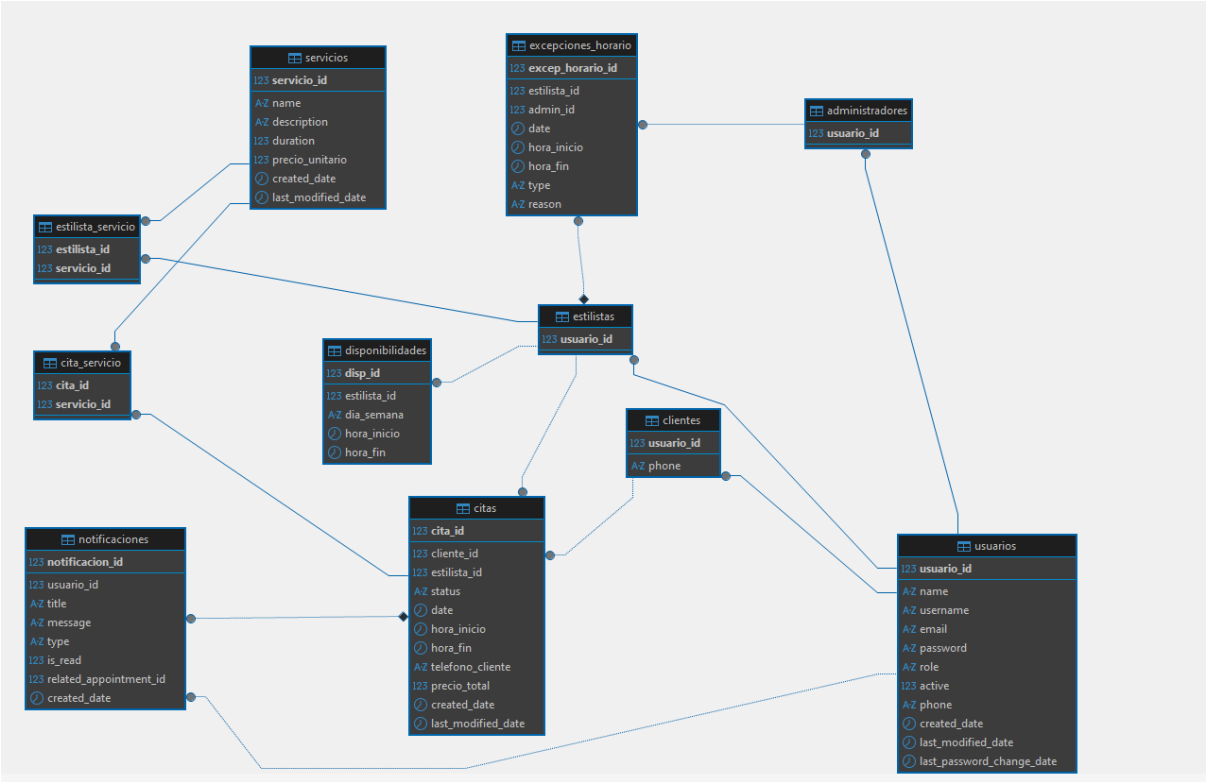
Modelo de datos

El modelo de datos de BookMyCut está diseñado siguiendo principios de normalización (3NF) y buenas prácticas. El esquema se compone de 10 tablas principales más 2 tablas de relación N:M: usuarios (tabla base con campo role), clientes, estilistas, administradores (tablas de relación), servicios, disponibilidades (horarios por día de semana), excepciones_horario (días festivos/vacaciones), citas, cita_servicio (relación N:M), estilista_servicio (relación N:M) y notificaciones.

El modelo implementa relaciones principales: Usuarios-Roles (1:1 mediante tablas de relación), Clientes-Citas (1:N), Estilistas-Citas (1:N), Estilistas-Disponibilidades (1:N), Estilistas-Servicios (N:M), Citas-Servicios (N:M), Usuarios-Notificaciones (1:N), y Administradores-Excepciones (1:N). Todas las relaciones están definidas con foreign keys y campos de auditoría (created_date, last_modified_date).

Esquema E-R





Análisis/Comparativa con Alternativas del Mercado

En el mercado hay varias aplicaciones comerciales para gestionar reservas en peluquerías, como Fresha (desde 20 € al mes), Booksy (que cobra comisiones por reserva) o Timely (desde 40 € al mes). Estas apps traen funciones avanzadas, como apps móviles, pagos online o herramientas de marketing, pero para negocios pequeños tienen algunos inconvenientes: son caras, no tienes control total sobre tus datos, dependen de servicios externos y no se pueden personalizar mucho.

BookMyCut, en cambio, es gratis y te da control total sobre los datos, porque puedes alojarlo tú mismo. Tiene un diseño simple y práctico, es totalmente personalizable al ser código abierto y no depende de servicios de terceros. Por otro lado, todavía le faltan cosas que sí tienen las alternativas comerciales, como la gestión de inventario, pagos online, marketing automático o una app móvil, pero para un negocio pequeño que solo quiere gestionar citas básicas, estas funciones no son tan necesarias.

Prototipo de la aplicación web

El prototipo de BookMyCut se basa en un sistema de diseño minimalista y elegante, utilizando principalmente los colores blanco y negro. La paleta incluye variaciones de gris para jerarquía visual, con soporte para tema claro y oscuro. El diseño es completamente responsive: mobile (< 768px) con menú colapsable y tablas como tarjetas y desktop (> 992px) con layout completo.

Las principales páginas del sistema incluyen: página de inicio pública (Welcome) con hero section y características, página de login con formulario de autenticación, dashboard para estilistas y administradores con resumen de citas, formulario de reserva con calendario interactivo y selector de servicios, lista de citas con filtros y paginación, gestión de servicios para estilistas/administradores, y perfil de usuario con edición de datos personales y cambio de contraseña.

Documentación interactiva

La API está completamente documentada con Swagger/OpenAPI 3. Acceso a la documentación interactiva:

- **Swagger UI**: http://localhost:8080/api/swagger-ui.html
- **OpenAPI JSON**: http://localhost:8080/api/api-docs
- **OpenAPI YAM**L: http://localhost:8080/api/api-docs.yaml

Total de Endpoints: 40 endpoints REST
- **Controladores**: 8 controladores principales
- **Módulos**: Autenticación, Usuarios, Citas, Servicios, Disponibilidades, Excepciones, Notificaciones, Estilistas-Servicios

Manual de despliegue

Despliegue con Docker Compose (Recomendado)

Requisitos: Docker Desktop (20.10+), Docker Compose (incluido), Git.

1. Clonar repositorio: “git clone <URL> && cd dvilmar-proyecto-final”

2. Configurar variables: Crear archivo .env copiando el .env.example en la raíz del proyecto.

3. Construir y levantar: “docker-compose up -d --build” (construye imágenes, descarga MySQL, crea red/volúmenes, ejecuta scripts SQL, inicia contenedores)

4. Verificar estado: docker-compose ps (debe mostrar backend, frontend y mysql en estado "Up")

5. Acceder:
- Frontend: http://localhost:4200

- Backend: http://localhost:8080/api
- Swagger: http://localhost:8080/api/swagger-ui.html

También existen .env en cada directorio de los Dockerfiles para adaptar un posible despliegue manual, pero habría que crear la base de datos manualmente, actualmente se cree en el docker-compose.

Conclusiones y Postmortem

Creo que la aplicación cumple con lo que se esperaba del proyecto: permite gestionar citas, usuarios y servicios de manera funcional y sencilla. La aplicación es intuitiva y fácil de usar, y los diferentes roles tienen acceso a lo que necesitan. La elección de tecnologías creo que fue acertada y adecuada para mí Spring Boot, Angular, MySQL, JWT, WebSockets, sobre todo el despliegue en Docker hace que la aplicación sea estable, segura y fácil de mantener y además me ha gustado especialmente trabajar con contenedores..

Durante el desarrollo, lo que más me costó fue como organizar el proyecto, luego la gestión y el despliegue fui sacando poco a poco. Los problemas más grandes fueron configurar las notificaciones en tiempo real con WebSockets, sincronizar el calendario y hacer que el modo claro/oscuro funcionará en toda la web y el sistema de guardado de contraseña.

Lo que aprendí de este proyecto es que es muy importante planificar bien los requisitos y dejar claras las estructuras de datos desde el principio, porque facilita mucho el trabajo. Para futuras mejoras, se podrían, añadir pagos online, mejorar el sistema de contraseñas y login, etc. Podrían faltar validaciones por hacer, mejorar la seguridad en ciertos aspectos.

En resumen para ser mi primera app creo que está bien, es una aplicación sencilla, útil, perfecta para negocios pequeños, y sirve como base para añadir mejoras en el futuro según las necesidades reales.

Bibliografía

Enlace repositorio: https://github.com/dvilmar/dvilmar-proyecto-final

Daniel Vilar Martínez
Versión del Proyecto: 1.0.0