PROYECTO AYTODEPORTE

TRABAJO FIN DE CICLO

DOCUMENTACIÓN TÉCNICA Y FUNCIONAL

DIEGO SAELICES SAELICES

DICIEMBRE 2025

ÍNDICE

1. Introducción.

a. Presentación del proyecto.

Este proyecto, denominado **AytoDeporte**, consiste en el desarrollo de una aplicación web destinada a facilitar y automatizar la gestión de reservas de instalaciones deportivas municipales. La plataforma ha sido diseñada como una herramienta moderna, accesible y disponible para todos los ciudadanos, con el objetivo de sustituir los procesos manuales y poco eficientes que actualmente se utilizan en el municipio.

AytoDeporte permite a los usuarios consultar horarios disponibles, realizar reservas de manera autónoma y gestionar sus datos personales desde cualquier dispositivo con conexión a internet. Para los administradores municipales, la aplicación incluye funcionalidades avanzadas como el control de usuarios, la gestión de instalaciones y la planificación de bloqueos horarios, así como herramientas específicas adaptadas a las particularidades del polideportivo local, como la diferenciación entre pistas de pádel viejas y nuevas o el bloqueo de fechas por eventos.

El sistema se ha implementado mediante una arquitectura **full-stack**, utilizando tecnologías como **HTML/CSS/JS** en el frontend y **Java Spring Boot + MySQL** en el backend, garantizando una estructura escalable, segura y fácil de mantener.

b. Objetivos del proyecto.

El proyecto persigue una serie de objetivos claramente definidos, divididos en **objetivo general** y **objetivos específicos**.

Objetivo general

Desarrollar una aplicación web completa que permita gestionar de forma digital, rápida y eficiente las reservas de instalaciones deportivas del ayuntamiento, eliminando procesos manuales y mejorando la experiencia tanto para los ciudadanos como para los administradores municipales.

Objetivos específicos

- **Digitalizar el sistema actual de reservas**, sustituyendo llamadas telefónicas y trámites presenciales.
- **Ofrecer una interfaz intuitiva y sencilla**, accesible para usuarios con conocimientos tecnológicos básicos.
- **Diseñar e implementar una API REST segura**, basada en Spring Boot, que centralice todas las operaciones relacionadas con reservas, usuarios e instalaciones.
- **Integrar un sistema de autenticación JWT** para garantizar la privacidad y la seguridad de los datos.
- **Automatizar reglas municipales reales**, como la distinción entre pistas de pádel viejas y nuevas, horarios específicos o periodos de bloqueo.

- **Permitir al ayuntamiento controlar la actividad deportiva**, gestionar usuarios, administrar el calendario y configurar bloqueos masivos cuando sea necesario.
- **Crear una aplicación extensible**, preparada para evolucionar e incorporar nuevas funcionalidades en el futuro, como estadísticas, notificaciones o sistemas de pago.

c. Justificación del proyecto.

La motivación principal de este proyecto surge de una necesidad real del municipio. Actualmente, **el proceso para reservar una pista deportiva es lento, incómodo y requiere múltiples desplazamientos físicos**, lo que supone una barrera tanto para los ciudadanos como para el propio personal del ayuntamiento. El procedimiento tradicional implica:

1. **Llamar al polideportivo municipal** para consultar si existe una hora libre.
2. En caso afirmativo, **acudir presencialmente al ayuntamiento** para comprar un ticket de reserva.
3. Finalmente, **volver al polideportivo** para entregar dicho ticket, momento en el que la reserva se hace oficial.

Este método, además de ineficiente, provoca pérdidas de tiempo, aglomeraciones innecesarias y una gestión caótica de horarios, especialmente en días de mayor demanda.

Ante esta situación, **AytoDeporte ofrece una solución integral**, permitiendo que los ciudadanos puedan reservar instalaciones desde su casa en cuestión de segundos y que el ayuntamiento disponga de un sistema centralizado, seguro y automatizado para la gestión deportiva.

El proyecto no solo moderniza el servicio municipal, sino que también mejora la transparencia, reduce errores humanos, evita duplicidad de tareas y optimiza el trabajo de los empleados públicos. En definitiva, se trata de un paso fundamental hacia la digitalización del municipio y la mejora de la calidad de vida de sus habitantes.

2. Análisis de requerimientos

a. Identificación de necesidades y requerimientos

El análisis de requerimientos constituye la base sobre la cual se diseña y desarrolla una aplicación eficiente. En el caso de *AytoDeporte*, las necesidades detectadas provienen tanto de los ciudadanos como del personal municipal encargado de la gestión de las instalaciones deportivas.

Necesidades detectadas

Necesidades de los ciudadanos

- Poder consultar horarios libres sin necesidad de realizar llamadas telefónicas.
- Realizar reservas de manera rápida y autónoma.
- Evitar desplazamientos innecesarios al ayuntamiento o al polideportivo.
- Tener un historial claro de reservas realizadas y futuras.

- Disponer de una interfaz intuitiva, accesible desde cualquier dispositivo.
- Recibir confirmación inmediata de cada reserva sin depender de terceros.

Necesidades del ayuntamiento

- Reducir la carga de trabajo administrativo asociada a la gestión manual de instalaciones.
- Evitar errores, duplicidades y reservas solapadas.
- Administrar todas las instalaciones desde una única plataforma.
- Controlar de forma sencilla los horarios, festividades, torneos o bloqueos masivos.
- Gestionar usuarios y roles de forma segura.
- Tener trazabilidad completa de la actividad deportiva del municipio.
- Disponer de un sistema escalable, fácil de mantener y preparado para futuras ampliaciones.

Requerimientos del sistema

Los requerimientos se clasifican en **funcionales** y **no funcionales**.

Requerimientos funcionales

El sistema debe ser capaz de:

1. Gestionar usuarios (registro, inicio de sesión, roles y autenticación).
2. Permitir a los ciudadanos:
    - Consultar instalaciones disponibles.
    - Ver días y horarios libres.
    - Realizar reservas.
    - Cancelar reservas si la normativa lo permite.
3. Permitir al administrador:
    - Crear, editar o eliminar instalaciones.
    - Configurar horarios y reglas específicas.
    - Crear bloqueos masivos (festivos, mantenimiento, torneos…).
    - Gestionar usuarios y roles.
    - Ver listados completos de reservas.
4. Aplicar reglas reales del municipio:
    - Diferenciar pistas de pádel viejas y nuevas.
    - Evitar solapamientos.
    - Controlar eventos de días especiales.

5. Ofrecer una API REST para comunicación entre frontend y backend.

Requerimientos no funcionales

El sistema debe cumplir:

- **Seguridad:** autenticación mediante JWT y protección de rutas por roles.
- **Escalabilidad:** arquitectura modular preparada para futuros módulos.
- **Rendimiento:** cargas rápidas, consultas optimizadas y API eficiente.
- **Usabilidad:** interfaz clara, accesible y responsive.
- **Compatibilidad:** funcionamiento en móviles, tablets y ordenadores.
- **Mantenibilidad:** código organizado, comentado y con estructura limpia.
- **Disponibilidad:** estar accesible en cualquier momento sin depender de la presencia de un empleado.

b. Identificación del público

El proyecto está dirigido a dos grupos principales:

1. Usuarios ciudadanos

Personas que desean reservar instalaciones deportivas. Se caracterizan por:

- Niveles distintos de experiencia digital.
- Necesidad de una interfaz sencilla y directa.
- Preferencia por evitar trámites presenciales.
- Uso frecuente desde dispositivos móviles.

Para este público, la aplicación debe priorizar **simplicidad**, **claridad visual** y **rapidez**.

2. Administradores municipales

Trabajadores del ayuntamiento encargados de:

- Gestionar el polideportivo.
- Controlar instalaciones, usuarios y reservas.
- Realizar bloqueos masivos.
- Monitorear la actividad.

Este perfil requiere una plataforma más completa, con vistas avanzadas y herramientas de control, pero igualmente accesible y cómoda de usar.

c. Estudio de mercado y competencia

Para analizar la viabilidad del proyecto, se ha estudiado la situación actual del mercado, evaluando las principales soluciones existentes y comparando sus características con las necesidades del municipio.

1. Soluciones comerciales existentes

Entre las plataformas más conocidas destacan:

• Playtomic

- Muy extendida en clubes de pádel y tenis.
- Potente, moderna y con app móvil.
- Inconveniente: cobra comisiones por reserva y no es totalmente personalizable.
- No está diseñada para reglas específicas municipales.

• Reservadeportes / Bookitit / BookGym

- Soluciones de reserva genéricas.
- Permiten digitalizar instalaciones, pero están pensadas para gimnasios o múltiples servicios.
- Menos flexibles en configuración de reglas.
- Coste mensual para el ayuntamiento.

• Sistemas municipales propios (cada pueblo tiene el suyo)

- En algunos casos tienen aplicaciones básicas de reserva.
- Suelen ser limitadas, antiguas y poco adaptables.

2. Comparativa con AytoDeporte

Característica	Otras soluciones	AytoDeporte
Coste por reserva	En muchos casos lo hay	Únicamente el precio de la pista
Personalización por municipio	Limitada	Total
Control del Ayuntamiento	Parcial	Completo
Reglas especiales (pádel viejo/nuevo, torneos)	Difíciles de implementar	Totalmente integradas
Propiedad de los datos	Privada	Municipal
Accesibilidad	Buena	Excelente y adaptada a los vecinos del municipio
Evolución futura	Depende de la empresa	Libre y ampliable

3. Conclusión del estudio de mercado

El análisis demuestra que, aunque existen plataformas potentes, **ninguna se adapta completamente a las necesidades reales del municipio ni ofrece un control total sin comisiones.**
AytoDeporte se presenta como una solución **personalizada, económica, escalable y alineada con las particularidades del polideportivo local**, con capacidad para crecer y evolucionar según las necesidades del ayuntamiento.

3. Diseño y planificación

a. Definición de la arquitectura del proyecto

La arquitectura de *AytoDeporte* se ha diseñado siguiendo un enfoque **modular, escalable y orientado a servicios**, basado en el patrón cliente-servidor. El objetivo principal es mantener una separación clara entre la capa de presentación (frontend), la lógica de negocio (backend) y la persistencia de datos (base de datos), permitiendo que cada parte evolucione de manera independiente.

1. Frontend

El frontend está construido con **HTML, CSS y JavaScript**, aplicando un diseño totalmente responsive. La interfaz utiliza componentes reutilizables como header, footer, paneles dinámicos de información y estructuras de carga basadas en **fetch API**.

Las principales responsabilidades del frontend son:

- Mostrar la información al usuario de manera clara y accesible.
- Consumir la API REST del backend para obtener datos actualizados.
- Controlar los estados de sesión a través de JWT (almacenado en localStorage).
- Gestionar la navegación y las interacciones (reservas, cancelaciones, formularios).

2. Backend

El backend se ha implementado en **Java Spring Boot**, siguiendo una estructura clara basada en:

- **Controladores (Controllers)**: gestionan las peticiones HTTP y exponen los endpoints de la API REST.
- **Servicios (Services)**: centralizan la lógica de negocio, validaciones y procesos internos.
- **Repositorios (Repositories)**: se encargan de la comunicación con la base de datos mediante JPA/Hibernate.
- **DTOs y entidades**: gestionan la estructura de los datos que viajan entre capas.

La API sigue los principios REST, utilizando métodos como GET, POST, PUT y DELETE.
Además, incorpora un sistema de autenticación mediante **JWT**, permitiendo proteger rutas y diferenciar permisos entre usuarios y administradores.

3. Base de datos

La base de datos está implementada en **MySQL** y diseñada conforme a la estructura del polideportivo municipal. Incluye tablas como:

- usuarios
- instalaciones
- reservas
- bloqueos

Se han definido claves foráneas, restricciones y relaciones necesarias (1:N y N:1), garantizando consistencia e integridad.
Además, se han incorporado datos iniciales (seeders) y triggers relacionados con reglas reales del municipio, como impedir cancelaciones con menos de X horas de antelación.

4. Flujo general

1. El cliente realiza una petición HTTP desde el navegador.
2. El backend recibe la petición, valida el JWT si es necesario y ejecuta la lógica correspondiente.
3. Se consulta la base de datos si procede.
4. El backend devuelve una respuesta JSON estructurada.
5. El frontend actualiza la interfaz según los datos recibidos.

Este modelo garantiza una comunicación eficiente, buen rendimiento y facilidad de mantenimiento.

b. Diseño de la interfaz de usuario

El diseño de la interfaz es un aspecto clave del proyecto, ya que la herramienta está orientada a ciudadanos de todas las edades y niveles tecnológicos. La prioridad ha sido crear una **experiencia sencilla, clara y visualmente agradable** con una clara inspiración del entorno de Apple.

1. Principios de diseño aplicados

- **Simplicidad**: evitar elementos innecesarios para no abrumar al usuario.
- **Claridad**: tipografías legibles, botones visibles, iconografía intuitiva.
- **Consistencia visual**: uso de un mismo estilo en colores, tamaños y espaciados mediante variables CSS.
- **Accesibilidad**: contraste adecuado, botones grandes y textos explicativos.
- **Responsive design**: adaptación completa a móviles, tablets y PC.

2. Distribución de la interfaz

El proyecto sigue una estructura modular para mantener limpieza y facilitar la reutilización:

- **Header**:
  Navegación simplificada, acceso a login/logout e información del usuario.
- **Footer**:
  Información institucional, enlaces útiles y año dinámico.

- **Página principal (index)**:
  Presentación del sistema con acceso directo a la sección de instalaciones.
- **Listado de instalaciones**:
  Vista clara con tarjetas o bloques donde se muestra cada instalación.
- **Calendario y horarios**:
  Selección visual de días disponibles y horarios libres.
- **Panel de usuario**:
  Historial de reservas, datos personales y opciones de gestión.
- **Panel de administrador**:
    - Gestión de usuarios
    - Gestión de instalaciones
    - Bloqueos masivos
    - Listado global de reservas

3. Identidad visual

Se ha utilizado una paleta de colores suaves y modernos basada en los tonos azulados presentes en los diseños institucionales del proyecto.
Algunas variables CSS clave:

--color-bg: #e8e8ee;

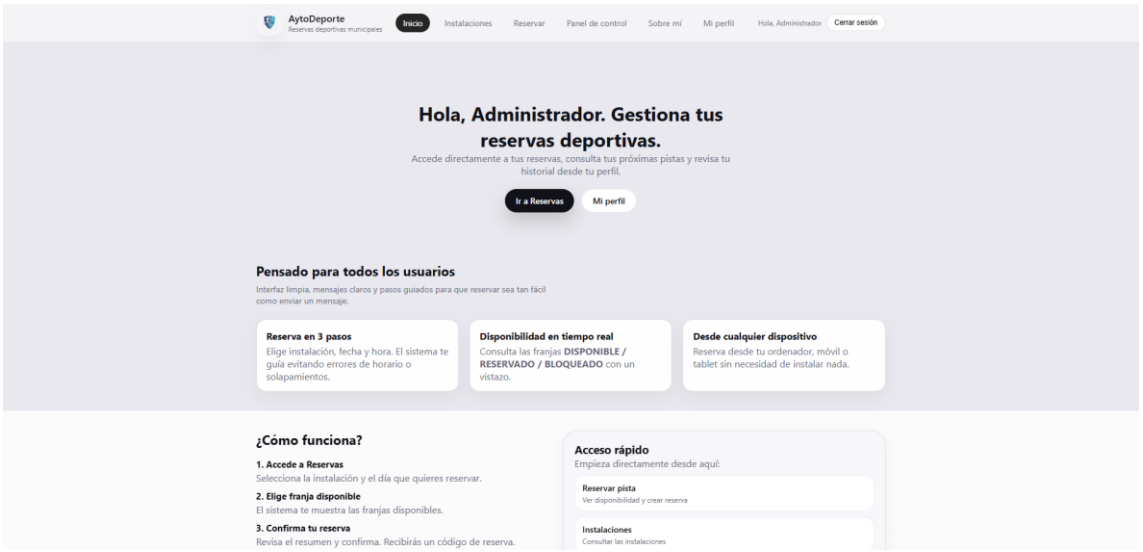
--color-primary: #007aff;

--color-text: #111118;

--radius-large: 24px;

--shadow-soft: 0 12px 30px rgba(0, 0, 0, 0.05);

Estas constantes ayudan a mantener uniformidad, mejorar el rendimiento y facilitar el mantenimiento del estilo.



c. Planificación de las tareas y los recursos necesarios

El proceso de desarrollo se ha organizado siguiendo una planificación estructurada en fases, lo que ha permitido avanzar de manera ordenada y controlar cada parte del proyecto.

1. Fases del proyecto

Fase 1: Análisis de requisitos

- Estudio del proceso actual de reservas en el municipio.
- Identificación de problemas y áreas de mejora.
- Recopilación de reglas específicas (pádel vieja/nueva, torneos, festivos...).

Fase 2: Diseño de la base de datos

- Modelo entidad-relación.
- Definición de tablas, atributos y relaciones.
- Creación del script inicial (scriptSaelices.sql).

Fase 3: Diseño del frontend

- Estructura HTML base.
- Definición del estilo visual (CSS).
- Maquetación responsive y componentes reutilizables.

Fase 4: Desarrollo del backend

- Creación del proyecto Spring Boot.
- Controladores, servicios y repositorios.
- Implementación de seguridad JWT.
- Endpoints REST completos.

Fase 5: Conexión frontend-backend

- Consumo de la API mediante fetch.
- Gestión del token y roles.
- Pruebas reales de reservas, cancelaciones y bloqueos.

Fase 6: Pruebas y correcciones

- Pruebas unitarias.
- Pruebas de integración y uso real.
- Corrección de errores y optimización.

Fase 7: Documentación y entrega

- Redacción de la memoria.

- Generación del manual técnico y del usuario.
- Preparación de la presentación final.

2. Recursos necesarios

Recursos materiales

- Ordenador personal capaz de ejecutar entornos de desarrollo.
- Software necesario:
    - Visual Studio Code
    - MySQL / XAMPP
    - Navegador actualizado
    - Git y GitHub

Recursos humanos

- Desarrollador del proyecto.

Recursos técnicos

- Framework: Spring Boot.
- Lenguajes: Java, HTML, CSS, JavaScript.
- Base de datos: MySQL.
- Control de versiones: GitHub.

4. Implementación y pruebas

a. Desarrollo de las funcionalidades del proyecto

La fase de implementación es el núcleo del proyecto, donde se materializan todos los requerimientos establecidos en las etapas anteriores. En *AytoDeporte*, el desarrollo se ha dividido en distintos módulos funcionales, cada uno de ellos diseñado para cumplir una función concreta dentro del sistema.

1. Implementación del backend (API REST)

La API REST del proyecto se ha desarrollado utilizando **Spring Boot**, organizando el código en tres capas principales: **controladores**, **servicios** y **repositorios**.

Controladores (Controllers)

Exponen los endpoints principales del sistema, como:

- /api/auth/login → Autenticación y generación de token JWT
- /api/usuarios → Gestión de usuarios
- /api/instalaciones → Listado y mantenimiento de instalaciones

- /api/reservas → Creación, consulta y cancelación de reservas
- /api/bloqueos → Gestión de bloqueos administrativos

Cada controlador se encarga de recibir las peticiones, validarlas y delegarlas a la capa de servicios.

Servicios (Services)

Contienen la lógica de negocio, incluyendo:

- Validación de horarios y disponibilidad.
- Prevención de solapamientos en reservas.
- Aplicación de reglas municipales (pádel viejo vs nuevo).
- Restricciones para cancelaciones con menos de X horas de antelación.
- Control de acceso según roles (ADMIN/USUARIO).

Repositorios (Repositories)

Encargados del acceso a la base de datos mediante JPA y consultas personalizadas cuando son necesarias.

Sistema de seguridad

El backend incluye:

- Autenticación **JWT**.
- Filtros para validar tokens.
- Protección de rutas según rol.
- Cifrado de contraseñas.

Esto garantiza que solo los usuarios autorizados acceden a las funciones sensibles.

2. Implementación del frontend

El frontend está desarrollado con **HTML, CSS y JavaScript** puro, organizado en componentes reutilizables para facilitar el mantenimiento.

Principales funcionalidades implementadas:

- Sistema de inicio de sesión con validación visual.
- Consumo de la API mediante fetch() para todas las operaciones.
- Panel de usuario con historial de reservas.
- Tarjetas dinámicas que representan cada instalación del polideportivo.
- Selector de días y horarios disponibles en tiempo real.
- Panel de administración con:
    - Gestión de instalaciones

- Gestión de usuarios
- Gestión de bloqueos masivos
- Visualización global del calendario

Manejo del token

Al iniciar sesión:

- El sistema almacena el token JWT en localStorage.
- Cada petición protegida incluye automáticamente el header:
- Authorization: Bearer <token>
- Los scripts detectan el rol del usuario y muestran opciones diferentes en el menú.

3. Base de datos

La base de datos MySQL contiene las tablas principales:

- **usuarios**
- **instalaciones**
- **reservas**
- **bloqueos**

Incluye:

- Relaciones definidas con claves foráneas.
- Índices en campos críticos para mejorar el rendimiento.
- Triggers opcionales para validar lógica de negocio especial.
- Datos iniciales insertados para pruebas.

b. Pruebas unitarias y de integración

La fase de pruebas garantiza que todas las funcionalidades del sistema se ejecuten correctamente y que el sistema sea robusto ante errores o situaciones imprevistas. En este proyecto se han realizado **pruebas unitarias**, **pruebas de integración** y **pruebas de usuario**.

1. Pruebas unitarias

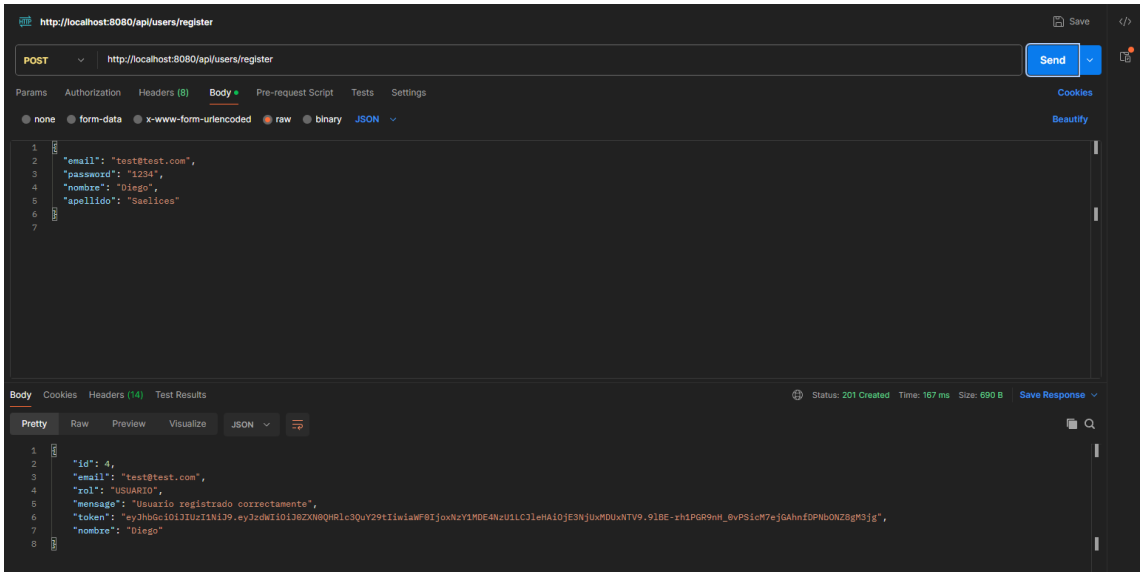
Se han aplicado pruebas a las principales funciones de la capa de servicios.
Estas pruebas verificaron que:

- Las reservas no puedan solaparse con otras ya existentes.
- Las cancelaciones fuera de tiempo sean rechazadas correctamente.
- Los bloqueos afecten a los horarios y días adecuados.
- Los usuarios no puedan acceder a recursos restringidos sin el rol correspondiente.

- El sistema responda con los códigos HTTP adecuados (200, 400, 401, 403, 404).

2. Pruebas de integración

Para probar la API completa se utilizó **Postman**, realizando peticiones reales al backend.



Pruebas realizadas:

- Login correcto e incorrecto (JWT válido e inválido).
- Listado de instalaciones.
- Creación de reservas válidas.
- Intentos de reserva en horarios bloqueados.
- Cancelación de reservas con y sin las condiciones necesarias.
- Comprobación de funcionamiento del panel de administración:
    - Crear bloqueos.
    - Consultar reservas globales.
    - Crear y editar instalaciones.

Estas pruebas permitieron verificar que el flujo completo del sistema es correcto y que la API REST funciona adecuadamente.

3. Pruebas con usuarios reales

Para validar la usabilidad del sistema, se realizaron pruebas con algunos familiares.

Se comprobó:

- Claridad de la interfaz.
- Facilidad de reservar una pista.
- Funcionamiento del calendario.
- Comprensión de mensajes de error.

- Flujo de login y logout.

Gracias a estas pruebas, se realizaron mejoras en el diseño antes de la versión final.

c. Corrección de errores y optimización del rendimiento

Durante la fase final del desarrollo se identificaron y corrigieron diversos errores y se optimizó el rendimiento tanto del backend como del frontend.

1. Corrección de errores comunes

Algunos de los problemas detectados fueron:

- Errores de CORS al conectar el frontend con el backend.
- Problemas con consultas SQL que no devolvían los horarios correctos.
- Peticiones repetidas innecesarias en el frontend.
- Formularios que no validaban correctamente los datos del usuario.
- Fallos al mostrar el rol del usuario en el header.
- Errores visuales en CSS y problemas de responsive.

Todos fueron solventados mediante:

- Ajustes en la configuración de Spring Security.
- Refactorización de consultas.
- Limpieza de código JavaScript.
- Nuevas funciones de validación y control de estado.

2. Optimización del rendimiento

Se aplicaron varias estrategias de optimización:

En el backend

- Creación de índices en tablas de reservas e instalaciones.
- Reducción de consultas pesadas mediante consultas personalizadas.
- Mejora del tiempo de respuesta de la API.
- Cacheo temporal de datos poco cambiantes (roles, instalaciones…).

En el frontend

- Minimización del acceso al DOM.
- Optimización del manejo de eventos.
- Reducción de cargas innecesarias.
- Reutilización de componentes UI.
- Carga gradual de contenido según la navegación.

3. Mantenibilidad y calidad del código

Para mantener un código limpio y fácil de escalar se aplicaron:

- Principios de arquitectura limpia.
- División en módulos lógicos.
- Comentarios detallados en secciones complejas.
- Uso de variables CSS para coherencia visual.
- Control estricto del flujo de autenticación.

5. Documentación

a. Documentación técnica

La documentación técnica describe en detalle la estructura interna del proyecto, sus componentes principales, la API REST, la base de datos y la lógica de negocio implementada. Su objetivo es permitir que cualquier desarrollador pueda comprender, mantener y ampliar la aplicación sin depender del autor original.

1. Estructura del backend (Spring Boot)

El backend sigue una arquitectura basada en capas claramente separadas:

- **controller** → Endpoints REST.

- **service** → Lógica de negocio.

- **repository** → Gestión de datos mediante JPA.

- **config** → Configuraciones específicas del proyecto.

- **security** → Filtros JWT, configuraciones y autenticación.

- **model** → Entidades y DTOs.

2. Algunos endpoints principales de la API REST

Método	Endpoint	Descripción
POST	/api/users/login	Inicia sesión y devuelve el token JWT
GET	/api/reservations/availability	Obtener disponibilidad horaria de una instalación un día concreto
DELETE	/api/blocks/{id}	Eliminar un bloqueo por su id
POST	/api/reservations/{id}/cancel	Cancelar una reserva
GET	/api/installatiosn/active	Obtener instalaciones activas
DELETE	/api/installations/{id}	Eliminar una instalacion
PUT	/api/installations/{id}	Actualizar una instalación

3. Lógica de autenticación JWT

La aplicación utiliza tokens JWT para:

- Validar identidad del usuario.
- Gestionar roles (USUARIO/ADMIN).
- Proteger rutas sensibles.

Cada petición protegida incluye el header:

Authorization: Bearer <token>

El backend valida:

- Integridad del token.
- Expiración.
- Rol correspondiente.

4. Base de datos

La base de datos MySQL incluye las tablas:

- **usuarios(id, nombre, email, password, rol)**
- **instalaciones(id, nombre, tipo, activo)**
- **reservas(id, usuario_id, instalacion_id, fecha, hora)**
- **bloqueos(id, instalacion_id, fecha_inicio, fecha_fin)**

Relaciones:

- 1 usuario → N reservas
- 1 instalación → N reservas
- 1 instalación → N bloqueos

Se añaden índices y restricciones para evitar duplicidades y mejorar rendimiento.

5. Lógica de negocio clave

- Evitar reservas en horas ya ocupadas.
- Impedir reservas dentro de instalaciones bloqueadas.
- Evitar solapamientos.
- Impedir cancelaciones con menos de X horas de antelación.
- Permitir bloqueos masivos para torneos, festividades o mantenimiento.

b. Documentación de usuario

La documentación de usuario está dirigida principalmente a dos grupos:
1) Los ciudadanos que reservan instalaciones.
2) El personal del ayuntamiento que utiliza el panel de administración.

1. Manual para ciudadanos

Acceder al sistema

- Entrar en la URL oficial del ayuntamiento.
- Registrarse o iniciar sesión con usuario y contraseña.

Reservar una instalación

1. Acceder al menú **Instalaciones**.
2. Seleccionar la pista deseada.
3. Elegir un día disponible.
4. Seleccionar la hora libre.
5. Confirmar la reserva.
6. Recibir el mensaje “Reserva realizada con éxito”.

Ver mis reservas

- Acceder al apartado **Mi perfil**.
- Consultar el historial y reservas futuras.
- Cancelar reservas permitidas.

Cerrar sesión

- Seleccionar **Cerrar sesión**.

2. Manual para administradores

Acceso al panel de administración

- Iniciar sesión con un usuario con rol **ADMIN**.
- El menú mostrará una nueva opción llamada Control.

Funciones disponibles

- **Gestión de usuarios** → crear, editar o eliminar usuarios.
- **Gestión de instalaciones** → alta/baja y edición de pistas.
- **Bloqueos** → creación de bloqueos masivos, días festivos y torneos.
- **Ver todas las reservas** → supervisión completa de la actividad.

Crear bloqueos masivos

1. Entrar en **Bloqueos**.
2. Seleccionar instalación o “todas”.
3. Elegir fecha y hora.
4. Confirmar.

El sistema bloqueará automáticamente las horas seleccionadas para todos los ciudadanos.

c. Manual de instalación y configuración

Este manual permite desplegar el sistema *AytoDeporte* desde cero.

1. Requisitos previos

Software necesario

- Java 17 o superior
- MySQL 8
- XAMPP o Workbench (opcional)
- Visual Studio Code

Hardware

- PC estándar con conexión a internet.

2. Instalación de la base de datos

1. **1.** Abrir panel de XAMPP e iniciar MySQL y Apache
2. Acceder a localhost/phpmyadmin/
3. Para crear la base de datos, ejecutar el script proporcionado (scriptSaelices.sql).
4. Confirmar que se han creado todas las tablas.
5. Verificar datos iniciales como usuarios de prueba, instalaciones y bloqueos.

3. Inicio del backend

1. Abrir el proyecto con VSCode.
2. Instalar la extensión live server.
3. Abrir un terminal y ejecutar cd backend y después, .\mvnw spring-boot:run
4. Comprobar que funciona yendo al index.html botón derecho, “Open with live server”.

6. Mantenimiento y evolución

El mantenimiento y la evolución del sistema son aspectos fundamentales para garantizar que *AytoDeporte* continúe siendo una herramienta útil, segura y adaptada a las necesidades del ayuntamiento a lo largo del tiempo. Una aplicación de este tipo no debe considerarse un producto cerrado, sino un sistema vivo que requiere revisiones periódicas, correcciones y mejoras continuas.

a. Plan de mantenimiento y soporte

El plan de mantenimiento de *AytoDeporte* se ha diseñado con el objetivo de asegurar su correcta operatividad, prevenir fallos y garantizar la seguridad de los datos de los usuarios.

1. Mantenimiento correctivo

Este tipo de mantenimiento se centra en la corrección de errores que puedan aparecer durante el uso real de la aplicación:

- Corrección de fallos detectados por los usuarios.
- Solución de errores en reservas, cancelaciones o bloqueos.
- Ajustes en el sistema de autenticación.
- Corrección de errores visuales en la interfaz.
- Solución de incompatibilidades con nuevos navegadores o dispositivos.

Estas incidencias se registrarán y se solucionarán en el menor tiempo posible para garantizar la continuidad del servicio.

2. Mantenimiento preventivo

Su objetivo es evitar que aparezcan problemas antes de que se produzcan:

- Revisión periódica de logs del sistema.
- Actualización de dependencias del backend (Spring Boot, librerías de seguridad).
- Comprobación del rendimiento de la base de datos.
- Limpieza de datos antiguos o reservas obsoletas.
- Revisión del sistema de copias de seguridad.

3. Mantenimiento adaptativo

Permite adaptar el sistema a cambios externos:

- Adaptación a nuevas normativas municipales.
- Cambios en la organización del polideportivo.
- Incorporación de nuevas instalaciones deportivas.
- Cambios en los horarios o en las reglas de uso.

4. Soporte al usuario

Se contempla un sistema de soporte básico mediante:

- Correo electrónico de contacto.
- Manual de usuario.
- Ayuda contextual dentro de la propia aplicación.
- Resolución de incidencias por parte del personal municipal.

b. Identificación de posibles mejoras y evolución del proyecto

Aunque *AytoDeporte* cumple correctamente los objetivos iniciales, existen múltiples líneas de evolución que pueden ampliar notablemente su funcionalidad en el futuro.

1. Mejoras funcionales para los ciudadanos

- Envío de correos electrónicos de confirmación de reserva.
- Recordatorios automáticos antes del inicio de la actividad.
- Histórico detallado con estadísticas personales.
- Sistema de valoraciones sobre las instalaciones.
- Posibilidad de repetir reservas semanales automáticamente.

2. Mejoras funcionales para el ayuntamiento

- Panel de estadísticas con gráficas de ocupación por meses.
- Informes automáticos para la toma de decisiones.
- Gestión avanzada de turnos, torneos y ligas.
- Control de ingresos si se integran sistemas de pago.
- Gestión de incidencias reportadas por los usuarios.

3. Mejoras técnicas

- Implementación de sistema de cache para acelerar consultas.
- Optimización avanzada de consultas SQL.
- Sistema de notificaciones en tiempo real.
- Refuerzo de la seguridad con doble factor (2FA).
- Migración futura a arquitectura basada en microservicios.

c. Actualizaciones y mejoras futuras

El sistema *AytoDeporte* está preparado para crecer y evolucionar mediante un sistema de actualizaciones estructurado y controlado.

1. Sistema de versiones

Cada nueva versión del sistema podrá incluir:

- Corrección de errores detectados.
- Nuevas funcionalidades solicitadas por el ayuntamiento.
- Mejoras visuales y de experiencia de usuario.
- Optimizaciones de rendimiento.

Las versiones se documentarán correctamente indicando:

- Cambios introducidos.
- Errores corregidos.

- Funcionalidades añadidas.
- Posibles incompatibilidades.

2. Integración de nuevas tecnologías

A medio y largo plazo, se puede plantear:

- Creación de una **aplicación móvil nativa** para Android e iOS.
- Integración de **pasarelas de pago online**.
- Uso de **servicios en la nube** para mejorar la disponibilidad.
- Incorporación de **inteligencia artificial** para predicción de ocupación.
- Integración con plataformas municipales existentes.

3. Proyección del proyecto

AytoDeporte no solo puede utilizarse en el municipio para el que ha sido creado, sino que puede replicarse en otros pueblos o ayuntamientos con mínimas adaptaciones, convirtiéndose en un producto escalable, reutilizable y comercializable en el futuro.

7. Conclusiones

a. Evaluación del proyecto

El proyecto **AytoDeporte** ha permitido desarrollar una solución real, funcional y adaptada a una necesidad concreta del municipio: la gestión eficiente de las reservas deportivas. A lo largo del desarrollo se ha conseguido transformar un proceso manual, lento y fragmentado en un sistema digital accesible, rápido y centralizado.

Desde el punto de vista técnico, el sistema presenta una arquitectura sólida basada en una **API REST con Spring Boot**, un **frontend ligero en HTML, CSS y JavaScript** y una **base de datos MySQL correctamente estructurada**. La integración de la autenticación mediante **JWT** garantiza la seguridad de los datos y el control de accesos, uno de los aspectos más críticos en cualquier sistema de gestión.

A nivel funcional, la aplicación responde correctamente a los casos de uso principales:

- Consulta de instalaciones.
- Reservas en tiempo real.
- Cancelaciones controladas.
- Gestión de bloqueos.
- Panel de administración completo.

Además, el proyecto ha sido desarrollado teniendo en cuenta la **usabilidad**, el **diseño responsive**, la **claridad visual** y la **facilidad de uso** para personas con distintos niveles de conocimiento tecnológico, lo que aumenta notablemente su valor práctico.

En conjunto, la evaluación global del proyecto es **muy positiva**, ya que cumple con su propósito principal, ofrece una solución real a un problema existente y se presenta como una herramienta lista para ser utilizada en un entorno municipal real.

b. Cumplimiento de objetivos y requisitos

Los objetivos planteados al inicio del proyecto se han cumplido de forma satisfactoria:

Objetivo general

El objetivo de **crear una plataforma digital para gestionar reservas deportivas municipales** se ha alcanzado plenamente, sustituyendo con éxito el proceso tradicional basado en llamadas, desplazamientos y tickets físicos.

Objetivos específicos alcanzados

- Digitalización completa del sistema de reservas.
- Creación de una interfaz intuitiva, clara y accesible.
- Implementación de una **API REST segura**.
- Integración de **autenticación JWT**.
- Automatización de reglas municipales reales.
- Desarrollo de un **panel de administración completo**.
- Sistema escalable y preparado para futuras ampliaciones.

Requisitos funcionales

Todos los requisitos funcionales definidos en la fase de análisis han sido implementados:

- Gestión de usuarios.
- Gestión de instalaciones.
- Gestión de reservas.
- Gestión de bloqueos.
- Control de permisos por roles.

Requisitos no funcionales

También se han cumplido los principales requisitos no funcionales:

- Seguridad.
- Rendimiento adecuado.
- Diseño responsive.
- Mantenibilidad del código.

- Escalabilidad del sistema.

Por tanto, puede afirmarse que **el proyecto cumple completamente con los objetivos y requisitos establecidos** en su fase inicial.

c. Lecciones aprendidas y recomendaciones para futuros proyectos

El desarrollo de *AytoDeporte* ha supuesto una experiencia de aprendizaje muy completa tanto a nivel técnico como organizativo. Entre las principales lecciones aprendidas destacan las siguientes:

1. Importancia de una buena planificación

Una planificación correcta desde el inicio evita muchos problemas durante el desarrollo. Definir bien los requisitos, las reglas de negocio y la estructura del sistema es clave para no tener que rehacer partes importantes del proyecto más adelante.

2. Valor de la modularidad

Separar correctamente el proyecto en frontend, backend y base de datos facilita el mantenimiento, las pruebas y la ampliación futura. La arquitectura modular permite evolucionar el sistema sin afectar al conjunto.

3. Seguridad desde el primer momento

Implementar la autenticación JWT desde las primeras fases ha sido clave para garantizar un sistema seguro. La seguridad no debe dejarse para el final en ningún proyecto serio.

4. Pruebas constantes

Realizar pruebas de forma continua evita que los errores se acumulen. Probar cada módulo conforme se desarrolla ahorra mucho tiempo en las fases finales.

5. Importancia de la experiencia de usuario

Un sistema puede ser técnicamente correcto, pero si no es cómodo de usar, pierde gran parte de su valor. Las pruebas con usuarios reales han sido fundamentales para mejorar la interfaz.

Recomendaciones para futuros proyectos

- Definir desde el principio la documentación.
- Usar siempre control de versiones.
- Priorizar la seguridad.
- Diseñar pensando en el usuario final.
- No dejar las pruebas para el final.
- Diseñar la arquitectura pensando en el crecimiento futuro.

8. Bibliografía y referencias

a. Fuentes utilizadas en el proyecto y enlaces de interés

-https://start.spring.io/

-https://www.youtube.com/playlist?list=PLkVpKYNT_U9cbo0XywpRGl-DFJEHDi9g5

-https://www.youtube.com/watch?v=BAXc4Cznzlo

-Apuntes y código de otros proyectos realizados