PT-MTC-01. Manual Técnico de Configuración

SPORT-FULL
1.0

HISTORIAL DE REVISIÓN

Versión	Fecha Elaboración	Responsable Elaboración	Fecha Aprobación	Responsable Aprobación
1.0	04/12/2024	Andres Felipe Facundo		

CAMBIOS RESPECTO A LA VERSIÓN ANTERIOR

VERSIÓN	MODIFICACIÓN RESPECTO VERSIÓN ANTERIOR
1.0	Creación de documento

Tabla de Contenido.

SPORT-FULL
1.0
Tabla de Contenido.

1. Introducción

Este documento detalla los aspectos técnicos, funcionales y metodológicos necesarios para el desarrollo del proyecto
Sportfull, una plataforma destinada a la reserva de canchas sintéticas en línea. El documento incluye información sobre el alcance del proyecto, los requerimientos técnicos, la arquitectura utilizada, y los procedimientos para configuración y despliegue. El objetivo principal es proporcionar una guía completa para comprender y trabajar en el sistema de forma eficiente.

2. Alcance

El proyecto **Sportfull** tiene como alcance el desarrollo de una aplicación web y móvil que facilite la reserva de canchas deportivas. Los principales usuarios del sistema son jugadores, administradores de canchas, y los gestores o empleados

El documento abarca los siguientes aspectos:

- Requisitos técnicos para el sistema.
- Procedimientos de instalación y despliegue.
- Aspectos metodológicos para su desarrollo.

Este sistema afecta principalmente a los usuarios que buscan una experiencia rápida y confiable para reservar canchas, además de los administradores que necesitan gestionar horarios y precios de manera eficiente.

3. Definiciones, siglas y abreviaturas

SCRUM	Marco de trabajo ágil que se utiliza para la gestión y desarrollo de proyectos complejos.
SDK	*Software Development Kit*, conjunto de herramientas para desarrollo de software.
JDK	*Java Development Kit*, entorno necesario para desarrollar aplicaciones Java.
BD	Base de Datos.
API	Interfaz de Programación de Aplicaciones (*Application Programming Interface*).

4. Aspectos Técnicos

Requerimientos técnicos mínimos

El sistema **Sportfull** requiere los siguientes aspectos técnicos para su correcto funcionamiento:

Aspecto	Requerimiento Mínimo
Memoria RAM	8 GB (mínimo) para desarrollo, 4 GB (mínimo) para el entorno de ejecución.
Sistema Operativo	Windows 10 o superior, macOS 10.14+, o distribuciones Linux basadas en Ubuntu 20.04+.
Espacio en Disco	10 GB de almacenamiento disponible (mínimo).

Lenguajes de Programación	Java para el backend, jsx para el frontend
Servidor de Aplicaciones	Render o cualquier servidor compatible con Springboot (Railway, Render, etc.).
Motor de Base de Datos	MySQL
Frameworks	Springboot (backend), React.js (frontend).
Navegadores Compatibles	Chrome, Firefox, Microsoft Edge (últimas versiones).

4.1. Metodología de trabajo usada.

Gestión de proyectos:

SCRUM: Para integrar procesos ágiles con sprints quincenales, revisiones de sprint y reuniones diarias (*dailies*).

4.2. Tipo de Software o Enfoque tecnológico.

El sistema **Sportfull** está enfocado principalmente en el desarrollo de una **aplicación móvil**, diseñada para permitir a los usuarios gestionar la reserva de canchas deportivas desde cualquier lugar. Además, cuenta con una versión web complementaria orientada a tareas administrativas.

4.2. Arquitectura tecnológica.

El proyecto utiliza una **arquitectura cliente-servidor** con separación clara entre frontend y backend.

- **Frontend:** Construido con React, organizado en módulos como `components`, `layouts`, `pages`, y `UI`. Este enfoque modular permite el desarrollo y mantenimiento ágil de la interfaz.
- **Backend:** Desarrollado en Spring Boot, siguiendo una estructura en capas que incluye modelos, controladores, servicios y repositorios.
- **Estrategia general:** Aunque no utiliza microservicios, el backend se encuentra modularizado y preparado para escalar horizontalmente (adición de instancias del servidor).

- **Comunicación:** Utiliza API REST para la comunicación entre frontend y backend.
- **Despliegue:** El backend y frontend están configurados para funcionar de manera independiente, facilitando la integración con balanceadores de carga en caso de aumento en la demanda.

4.3. Patrones de Diseño usados.

- **Backend (Spring Boot):**
  - **MVC (Model-View-Controller):** El backend implementa el patrón MVC, separando la lógica de negocio (servicios), la gestión de datos (repositorios) y el control de solicitudes (controladores).
  - **DAO (Data Access Object):** Utilizado para interactuar con la base de datos mediante repositorios.
- **Frontend (React):**
  - **Componentes Reutilizables:** Diseñados siguiendo el principio de componentes desacoplados, permitiendo su reutilización en diferentes partes de la aplicación.
  - **Single Page Application (SPA):** Gestiona rutas dinámicamente, cargando únicamente el contenido necesario.

4.4. Gestión de Datos.

El sistema utiliza una **base de datos relacional** (por ejemplo, MySQL o PostgreSQL) debido a la necesidad de mantener integridad referencial y relaciones claras entre las tablas.

- **Persistencia:** La información es gestionada mediante JPA (Java Persistence API) en Spring Boot.
- **Justificación:** Las bases de datos relacionales son ideales para sistemas que requieren consistencia en transacciones, como reservas de campos deportivos, donde es crucial evitar conflictos en los horarios de reserva.

5. Requisitos de Configuración

Hardware:

- Procesador: Dual-core o superior.

- RAM: 4GB mínimo (8GB recomendado).
- Espacio en Disco: 500MB para la aplicación y base de datos.

Software:

- **Frontend:**
  - ReactJS (Node.js 16+ y npm).
  - Librerías: React Router, Axios,google-maps
- **Backend:**
  - Spring Boot (JDK 11 o superior).
  - Motor de base de datos: MySQL
  - Dependencias: Spring Data JPA, Spring Web, Lombok.

Entorno de desarrollo:

- IDEs: Visual Studio Code para el frontend, IntelliJ IDEA o Eclipse para el backend.
- Sistema operativo: Windows 10+ o Ubuntu 20.04+.

6. Proceso de Configuración o Despliegue

- **Frontend:**
  - Clonar el repositorio: `git clone <url-frontend>`.
  - Instalar dependencias: `npm install`.
  - Ejecutar en desarrollo: `npm start`.
  - Desplegar en producción: Usar `npm run build` para generar archivos estáticos y subirlos a un servidor como Netlify o Vercel.
- **Backend:**
  - Clonar el repositorio: `git clone <url-backend>`.
  - Configurar el archivo `application.properties` con las credenciales de la base de datos.
  - Ejecutar el servidor: Usar `mvn spring-boot:run` o empaquetar el proyecto (`.jar`) con Maven.

    - Desplegar en producción: Subir el archivo `.jar` a un servidor como AWS EC2 o Render.
- **Base de Datos:**
    - Configurar la base de datos MySQL/PostgreSQL con las tablas iniciales.
    - Ejecutar scripts SQL incluidos en el repositorio si es necesario.

7. Ingreso al Sistema

- **Inicio de la aplicación:**
    - **Frontend:** Abrir el navegador y acceder al enlace proporcionado para la aplicación (localhost o dominio en producción).
    - **Backend:** Verificar que el servidor está corriendo y las API están disponibles (puerto predeterminado: `8080`).
- **Credenciales de acceso:**
    - Usuario inicial: `admin@example.com`
    - Contraseña inicial: `admin123`
- **Resultado esperado:** El usuario debería ver una interfaz inicial con opciones para registrarse, iniciar sesión y navegar por las funcionalidades del sistema.

8. Otras Consideraciones

- **Variables de entorno:**
    - Configurar claves sensibles como credenciales de la base de datos y tokens API en un archivo `.env` o gestor de secretos.
- **Manejo de errores:**
    - Asegurarse de manejar correctamente los errores 404 y 500 en el frontend.
    - En backend, habilitar logs para rastrear errores en producción.
- **Compatibilidad:**
    - Probar en diferentes navegadores para asegurar la funcionalidad completa en Chrome, Firefox y Edge.
- **Escalabilidad futura:**

- Preparar configuraciones para la integración con un balanceador de carga y orquestación de contenedores como Docker.