Anexo V. Documentación técnica

Sistema para la mejora de la movilidad articular basada en el uso de la estimación de posturas

Trabajo Fin de Máster

Ingeniería Informática

Febrero 2025

Autor

Sergio Salinero Santamaría

Tutores

André Fílipe Sales Mendes

Gabriel Villarrubia González

Í N D I C E

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

El presente documento proporciona la documentación técnica correspondiente al sistema desarrollado, con el objetivo de facilitar la compresión del código fuente funcional. El sistema se compone de tres subsistemas interconectados (frontend, backend, y un servidor que gestiona la base de datos). La estructura del anexo es la siguiente:

- Estructura del sistema
- Documentación técnica

2. ESTRUCTURA DEL SISTEMA

A continuación, se detalla la estructura del sistema, dividida en los tres subsistemas desarrollados.

2.1. SUBSISTEMA FRONTEND

El subsistema Frontend comprende todo aquello que interactúa directamente con el usuario, conformando la parte visible del sistema y expresando las funcionalidades del mismo.

Ha sido desarrollado con la tecnología React. React es una biblioteca de JavaScript de código abierto usada especialmente para aplicaciones web. Permite crear componentes reutilizables que gestionan su propio estado y luego los combina para formar una interfaz más compleja.

Dentro del proyecto, en el directorio */liveposes/src* se encuentra el código fuente distribuido de la siguiente manera:

- **/app:** Directorio dedicado a las configuraciones generales del proyecto. Se encuentra la página inicial (index) junto con la hoja de estilos global y el icono de la plataforma (favicon.ico).
- **/pages:** Aquí se hayan los ficheros JavaScript correspondientes a las páginas accesibles en la plataforma, siendo un total de 16, se explicarán más adelante.
- **/components:** La interfaz de usuario se divide en piezas pequeñas y reutilizables de código, facilitando el mantenimiento y la escalabilidad. Aquí se encuentran los componentes que son utilizados por las páginas.
- **/libraries:** Almacena los ficheros de funcionalidad que no se corresponden ni con las páginas ni con los componentes. Concretamente, se encuentra un único fichero encargado de ejecutar el algoritmo de detección de posturas.
- **/utils:** Directorio que contiene ficheros de configuración globales. Contiene un fichero *Colors* para unificar la gama de colores de la interfaz en la plataforma, y un fichero *RemoteServices* para gestionar el acceso unificado a los servicios del componente Backend.

Siendo más específico en la explicación, se procede a explicar detalladamente el contenido del directorio que contiene la práctica totalidad de la funcionalidad, */pages*.

Los ficheros contenidos en este directorio junto con su explicación se muestran a continuación:

- **Signup.js:** Permite el registro de usuarios en el sistema.
- **Login.js:** Permite el acceso de usuario en el sistema basado en la autenticación por usuario y contraseña.
- **ForgotPassword.js:** Servicio de recuperación de contraseña. Se solicita el correo electrónico al usuario para enviarle un correo con un enlace personalizado donde podrá modificar su contraseña.

- **PasswordRecovery.js:** Una vez recibido el correo electrónico, esta será la pantalla donde el usuario deberá proporcional al sistema la nueva contraseña.
- **Home.js:** Es la página primera después de iniciar sesión. Su función es proporcionar acceso a las categorías de ejercicios, a la vez que permite accionar cualquiera de las funcionalidades principales, las cuales son:
    - Perfil de usuario
    - Mostrar categorías de ejercicios
    - Crear una rutina personal
    - Publicar una rutina en una categoría
    - Ver las estadísticas personales
    - Ver el historial de rutinas
    - Gestión de ejercicios físicos
- **CategoryRoutines.js:** Se encarga de permitir el acceso a las funcionalidades de las categorías de ejercicio, las cuales son:
    - Mostrar las rutinas contenidas en la categoría
    - Ejecutar cualquiera de las rutinas
    - Eliminar cualquiera de las rutinas
    - Eliminar todas las rutinas de la categoría
- **RoutineBuilding.js:** Encargado de permitir la ejecución de las funcionalidades relacionadas con la creación de una rutina de carácter personal:
    - Añadir un ejercicio a la rutina
    - Cambiar el tipo de ejercicio
    - Elegir el número de repeticiones del ejercicio
    - Eliminar un ejercicio
    - Modificar el orden de ejercicios en la rutina
    - Configurar tiempo de descanso
    - Ejecutar la rutina
- **PoseRecognition.js:** El fichero más relevante y complejo, se encarga de gestionar la ejecución de las rutinas. En primer lugar, carga el modelo *BlazePose* mediante la API de *TensorFlow*. Seguidamente, ejecuta las siguientes funcionalidades:
    - Inicia el modelo de detección de posturas
    - Detecta la postura del usuario
    - Carga los datos de la rutina configurada
    - Reconoce la señal de inicio e inicia el cronómetro
    - Reconoce la realización de un ejercicio
    - Gestiona tiempos de descanso
    - Determina la terminación de la rutina y gestiona los datos de seguimiento obtenidos
- **PublishRoutine.js:** Se encarga de permitir la publicación de una rutina de ejercicios físicos en una categoría específica:
    - Añadir un ejercicio a la rutina
    - Cambiar el tipo de ejercicio
    - Elegir el número de repeticiones del ejercicio
    - Eliminar un ejercicio
    - Modificar el orden de ejercicios en la rutina
    - Configurar tiempo de descanso
    - Dar una descripción a la rutina
    - Seleccionar la categoría destino

    - Publicar la rutina
- **Profile.js:** Permite gestionar la información del usuario:
    - Mostrar la información del usuario
    - Actualizar el peso del usuario
    - Cambiar la contraseña
    - Cerrar sesión
    - Eliminar la cuenta del usuario
- **Statistics.js:** Se encarga de gestionar el acceso a los datos de seguimiento del usuario. Las funcionalidades que se permiten son las siguientes:
    - Mostrar las estadísticas personales
    - Reiniciar las estadísticas
- **RoutineHistory.js:** Encargado de permitir el acceso a la funcionalidad relacionada con el historial de rutinas:
    - Mostrar las rutinas realizadas por el usuario
    - Ejecutar cualquiera de las rutinas
    - Eliminar todas las rutinas del historial de rutinas
- **ExerciseManagement.js:** Permite la gestión de los ejercicios presentes en el sistema. La funcionalidad relacionada es:
    - Mostrar los ejercicios presentes en el sistema
    - Acceso a la adición de ejercicios
    - Acceso a la edición de cualquiera de los ejercicios
    - Eliminar cualquiera de los ejercicios
    - Acceso a la información de ayuda sobre la gestión de ejercicios
- **EditExerciseManagement.js:** Permite la edición de la mayoría de los parámetros contenidos en un ejercicio.
- **AddExerciseManagement.js:** Permite la edición de ejercicios a partir de la introducción de los parámetros necesarios
- **ExerciseManagementHelp.js:** Proporciona una ayuda para la gestión y manejo de la información presentes en los ejercicios físicos.

2.2. SUBSISTEMA BACKEND

Para el funcionamiento de la plataforma, el subsistema Frontend necesita gestionar cierta información, ya sea para mostrarla, insertarla, modificarla o actualizarla. Entre el Frontend y la información se encuentra el subsistema Backend. Se encarga de gestionar las peticiones del Frontend de forma segura y solicitar al subsistema Datos realizar las operaciones con los datos.

La tecnología utilizada es Java junto con la herramienta Spring Boot 4 para facilitar la creación de microservicios web.

Dentro del directorio *src/main/java/com/liveposes/main* se encuentra el código fuente, el cual está distribuido de la siguiente forma:

- **/.:** Contiene la clase que inicia la aplicación Spring Boot.
- **/config:** Contiene la clase que configura el acceso a terceros, las operaciones permitidas y los directorios donde se permiten.

- **/controller:** Conjunto de clases dedicadas a gestionar las peticiones entrantes del subsistema Frontend.
  El controlador de la aplicación Spring boot se divide en 4 clases conformando los conjuntos de funcionalidades presentes en el sistema:
  - **AuthenticationController.java:** Encargado de gestionar las peticiones entrantes relacionadas con la autenticación de los usuarios en el sistema.
  - **ExercisesController.java:** Maneja las peticiones relacionadas con los ejercicios físicos. Es la clase más interesante y con más complejidad.
  - **StatisticsController.java:** Controla las peticiones entrantes que se corresponden con la gestión de las estadísticas de usuario.
  - **UserController.java:** Se encarga de gestionar las peticiones relacionadas con la información del usuario.
- **/model:** Conforma el conjunto de clases dedicadas al almacenamiento temporal de datos, así como su correcto formateo y organización. Las clases presentes en el directorio son las siguientes:
  - **BasicStatistics.java**
  - **CategoryCount.java**
  - **CurrentRoutine.java**
  - **CurrentRoutineExercises.java**
  - **Exercise.java**
  - **PublicRoutine.java**
  - **RoutineHistory.java**
  - **User.java**
- **/services:** Conjunto de clases complementarias a las clases del controlador. Su finalidad es gestionar las solicitudes hacia el subsistema Datos en base a las peticiones del controlador. Las clases contenidas en el directorio son las siguientes:
  - **AuthenticationServices.java**
  - **ExercisesServices.java**
  - **StatisticsServices.java**
  - **UserServices.java**
- **/utils:** Conjunto de funcionalidad que es utilizada en repetidas ocasiones por el resto del código. Las clases presentes son:
  - **EmailService.java:** Clase encargada del envío del correo electrónico a un usuario concreto.
  - **JWTUtil.java:** Clase dedicada a crear y validar los tokens de acceso al sistema. Comprende la seguridad de acceso basado en JWT (Json Web Token) garantizando el legítimo acceso a las funcionalidades del sistema.
  - **PasswordEncryption.java:** Se encarga de encriptar y comprobar la veracidad de una contraseña.
  - **RemoteServices.java:** Comprende un acceso unificado a los servicios remotos.

2.3. SUBSISTEMA DATOS

Para gestionar las operaciones en la información presente en el sistema, el subsistema Backend se comunica con el subsistema Datos. Esta estructura comprende una separación de responsabilidades, lo que favorece la modularidad en el sistema, aumentando la escalabilidad, reutilización, mantenibilidad y seguridad.

La tecnología utilizada es la misma que la del subsistema Backend. El lenguaje de programación es Java junto con la herramienta Spring Boot 4 para facilitar la creación de microservicios web.

Dentro del directorio *src/main/java/com/liveposes/main* se encuentra el código fuente, el cual está distribuido de la siguiente forma:

- **/.:** Contiene la clase principal que inicia la aplicación Spring Boot. Esta clase es el punto de entrada del sistema y configura el contexto inicial de la aplicación.
- **/config:** Incluye la clase responsable de la configuración general del sistema. Esta clase define el acceso a servicios de terceros, las operaciones permitidas y las rutas o directorios autorizados.
- **/controller:** Este paquete agrupa las clases encargadas de gestionar las solicitudes entrantes desde el subsistema Frontend. Las clases de control están organizadas en función de las principales funcionalidades del sistema, de la siguiente manera:
  - **AuthenticationController.java:** Gestiona las solicitudes relacionadas con la autenticación de los usuarios, incluyendo el inicio de sesión y la validación de credenciales.
  - **ExercisesController.java:** Administra las peticiones relacionadas con los ejercicios físicos. Es la clase más compleja y central para la funcionalidad del sistema.
  - **StatisticsController.java:** Maneja las solicitudes correspondientes a la gestión y consulta de estadísticas de usuario.
  - **UserController.java:** Se encarga de gestionar las peticiones relacionadas con la información del usuario, como su perfil y datos personales.
- **/model:** Este paquete incluye las clases destinadas al almacenamiento temporal de datos y su correcta estructuración. Estas clases actúan como representaciones de datos en el sistema, permitiendo su manipulación y transferencia. Las principales clases del modelo son:
  - **BasicStatistics.java**
  - **CategoryCount.java**
  - **CurrentRoutine.java**
  - **CurrentRoutineExercises.java**
  - **Exercise.java**
  - **PublicRoutine.java**
  - **RoutineHistory.java**
  - **User.java**
- **/services:** El paquete de servicios contiene clases que complementan a los controladores, proporcionando la lógica necesaria para gestionar las operaciones solicitadas en la información contenida en la base de datos. Las clases presentes son:
  - **AuthenticationServices.java**

    - **ExercisesServices.java**
    - **StatisticsServices.java**
    - **UserServices.java**
- **/utils:** Incluye clases utilitarias que proporcionan funcionalidades reutilizables en diferentes partes del sistema. Estas clases simplifican tareas recurrentes, mejorando la eficiencia del desarrollo. Los componentes destacados son:
    - **DBConnection.java:** Se encarga de gestionar la comunicación con la base de datos, así como el tipo de operación a realizar sobre la información.

3. DOCUMENTACIÓN TÉCNICA

Se han utilizado herramientas de generación de documentación de códigos de programación para generar una documentación técnica más adaptada a cada aspecto del código. Se detalla a continuación.

3.1. SUBSISTEMA FRONTEND

Se ha utilizado la herramienta JSDoc para generar la documentación técnica del subsistema Frontend. Se ha generado de forma independiente para los paquetes *components*, *libraries* y *pages* explicados anteriormente. En la Figura 1 se muestra una parte de la misma.

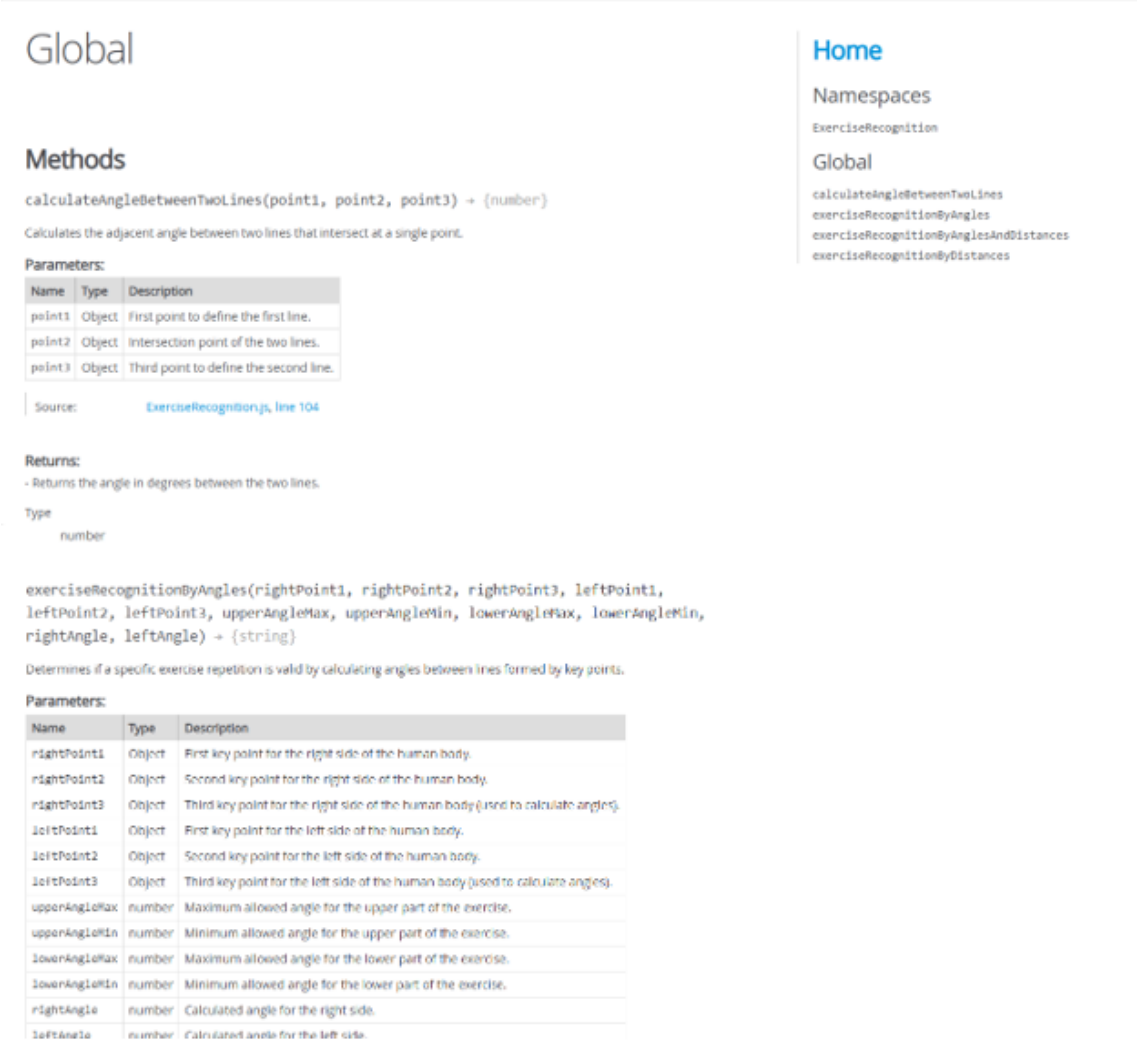

Figura 1: Ejemplo de documentación técnica del subsistema Frontend

3.2. SUBSISTEMAS BACKEND Y DATOS

En los subsistemas Backend y Datos se ha utilizado la herramienta Swagger para generar la documentación. Se ha documentado el acceso a los servicios, ejemplos de los parámetros necesarios en la petición y sus funcionalidades. En la Figura 2 se muestra una parte de la documentación generada.

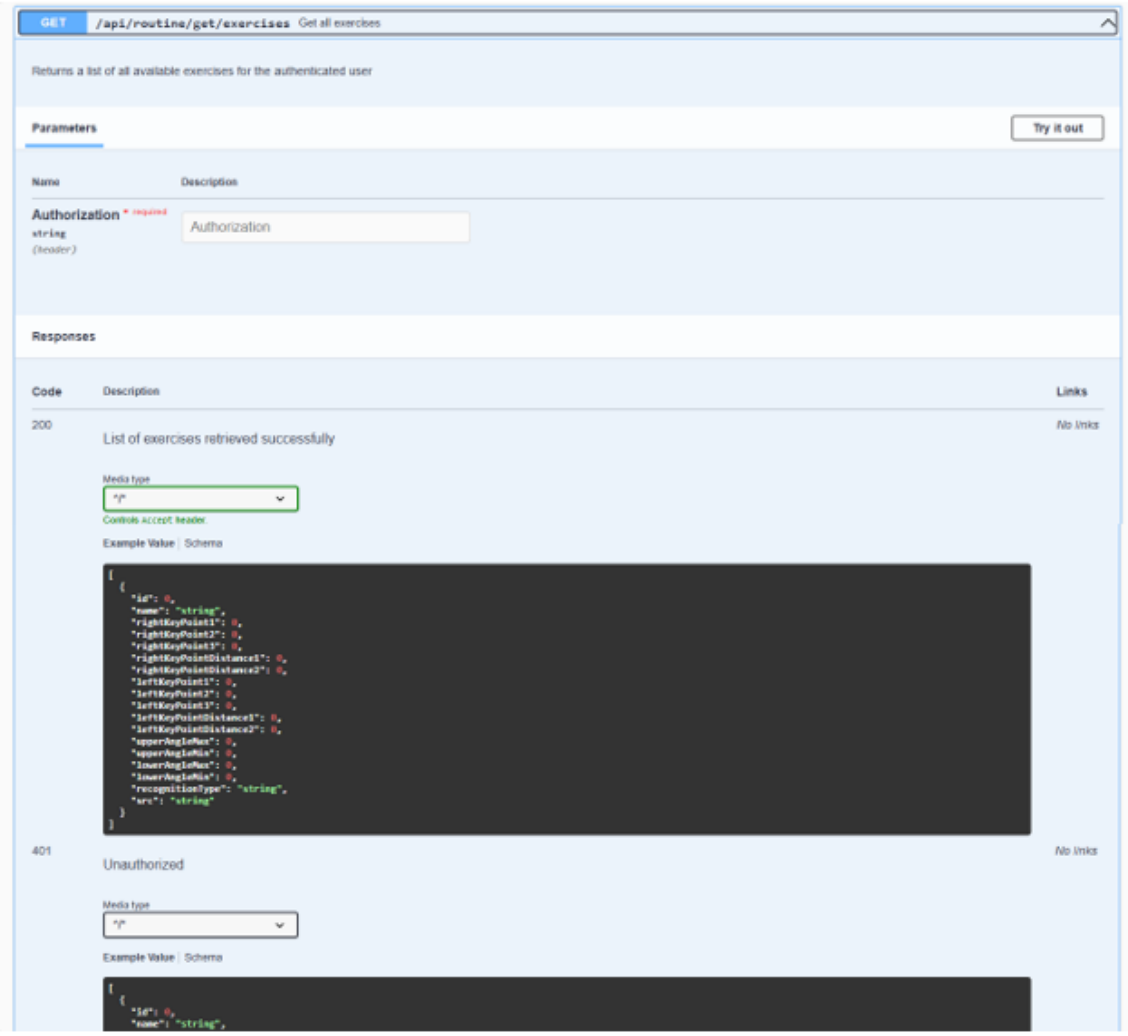

Figura 2: Ejemplo de documentación técnica de los subsistemas Backend y Datos

4. REFERENCIAS

[1] JSDoc. Accedido: 03-01-2025. Disponible en: https://jsdoc.app/

[2] Swagger. Accedido: 03-01-2025. Disponible en: API Documentation & Design Tools for Teams | Swagger