Stack y Setup del desarrollo

[1] Juan José Mendoza Marquez, Juan Pablo Peña Bernal[1], Santiago Bautista Velasquez[1], Santiago Alvarez Serrano[1], Santiago Martinez Cuellar[1]

[1] *Departamento Ingeniería de Sistemas, Pontificia Universidad Javeriana, Bogotá, 110231, Colombia.

Contributing authors: mendoza_@javeriana.edu.co; jp.penab@javeriana.edu.co; bautista_s@javeriana.edu.co; alvarezs.s@javeriana.edu.co; martinezcs@javeriana.edu.co;

Abstract

This document describes the technological architecture and future execution plan of a mobile application designed to support the tracking of physical training activities. The system aims to allow users to monitor their progress, access structured workout routines, and manage their training data in an organized and accessible manner. The solution is based on a client–server architecture composed of a Flutter mobile application, a backend developed in Java using Spring Boot, and a PostgreSQL database for data persistence. This architecture ensures a clear separation of responsibilities, scalability, and maintainability of the system, while enabling secure communication between components through a RESTful API.

Keywords: Mobile Application, Client–Server Architecture, REST API, Spring Boot, Flutter, PostgreSQL, Software Architecture, Training Tracking System, Data Persistence.

1 Introducción

En el presente documento se describe la arquitectura tecnológica definida para el proyecto y la manera en que este será ejecutado en fases futuras. El proyecto consiste en el desarrollo de una aplicación móvil que permitirá el seguimiento del progreso en actividades deportivas, permitiendo a los usuarios registrar su desempeño y acceder a rutinas de entrenamiento organizadas en un mismo espacio.

La solución se planteará bajo una arquitectura cliente–servidor, con el objetivo de separar claramente la interfaz de usuario, la lógica del sistema y la persistencia de los datos. Para esto, se utilizarán tecnologías que permitan que el sistema sea escalable, mantenible y preparado para futuras modificaciones o mejoras.

2 Arquitectura General

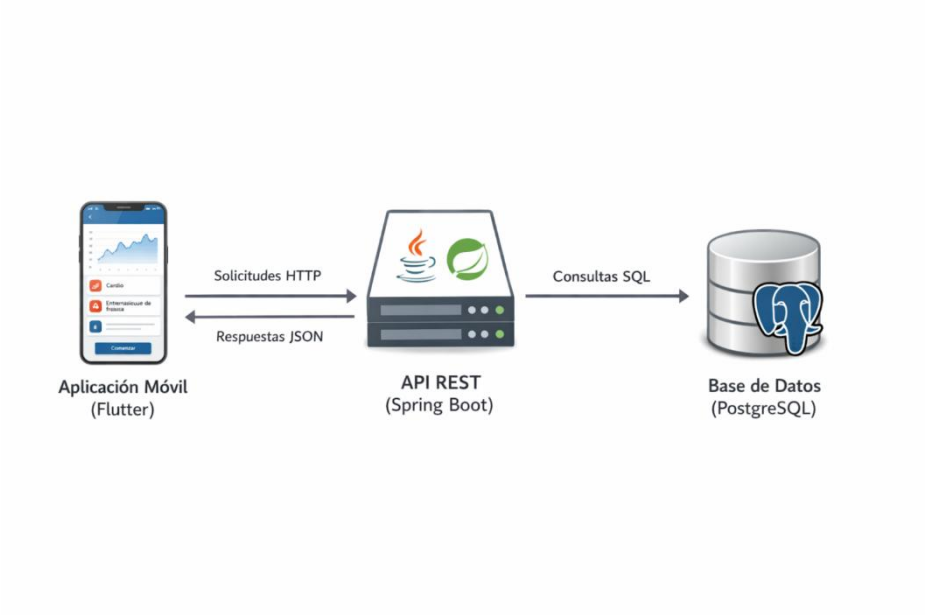
El proyecto se implementará basado en la arquitectura cliente-servidor, la cual esta compuesta por tres componentes principales, la aplicación móvil (cliente), un backend (servidor de aplicación) una base de datos (persistencia). Esta separación permite que cada parte del sistema cumpla con un rol especifico y facilite el mantenimiento y la escalabilidad del proyecto.

La aplicación móvil, desarrollada en Flutter, será la encargada de la interacción con el usuario. En la interacción del usuario con el sistema, el usuario podrá consultar rutinas, ejercicios, registrar sesiones de entrenamiento y almacenar su progreso. A pesar de esto, la aplicación no realizará operaciones directamente sobre la base de datos; en u lugar se comunicará con el backend mediante solicitudes HTTP.

El backend, desarrollado en Java con Spring Boot, actuará como intermediario entre la aplicación móvil y la base de datos. Acá se implementará la lógica de negocio, validar la información recibida desde el cliente y se expondrá una API REST que permita enviar y recibir datos en formato JSON. De esta manera, se garantiza que las reglas del sistema se apliquen de forma centralizada y constante.

Finalmente, la base de datos se desarrollará en PostgreSQL, donde almacenaremos toda la información del sistema, incluyendo usuarios, ejercicios, rutinas y registros de progreso. El acceso a la base de datos se realizará únicamente a través del backend, lo que permite controlar la seguridad, mantener integridad de la información y facilitar cambios futuros sin afectar directamente a la aplicación.

El flujo de información es el siguiente:



Esta arquitectura permite que el sistema pueda evolucionar a futuro, agregando nuevas categorías de entrenamiento, nuevas funcionalidades o clientes, sin la necesidad de rehacer el núcleo del sistema.

3 Diseño del sistema

El diseño del sistema se organiza internamente bajo un esquema por capas que permite mantener una clara separación de responsabilidades dentro del backend.

En el servidor desarrollado con Spring Boot, el sistema se estructurará en tres niveles principales:

- Controladores (Controllers): recibirán las solicitudes HTTP provenientes de la aplicación móvil y definirán los endpoints de la API REST.
- Servicios (Services): implementarán la lógica de negocio del sistema, aplicando reglas y validaciones antes de realizar operaciones.
- Repositorios (Repositories): gestionarán la interacción directa con la base de datos PostgreSQL, permitiendo realizar consultas y operaciones de persistencia.

Esta organización facilita la mantenibilidad del código, reduce el acoplamiento entre componentes y permite futuras ampliaciones sin afectar la estructura general del sistema.

Desde el lado del cliente, la aplicación desarrollada en Flutter mantendrá una separación entre la interfaz de usuario y la lógica de comunicación con el backend, delegando las solicitudes HTTP a servicios internos que gestionen la interacción con la API.

4 Tecnologías definidas

El desarrollo de la aplicación móvil se realizará utilizando Flutter, un framework multiplataforma que permite construir aplicaciones para dispositivos móviles a partir de una única base de código escrita en Dart. Esta tecnología facilitará la creación de una interfaz moderna e intuitiva para el usuario, además de permitir compatibilidad tanto en sistemas Android como iOS. Flutter será el encargado de gestionar la interacción con el usuario, mostrar las rutinas y ejercicios disponibles, registrar las sesiones de entrenamiento y comunicarse con el servidor para enviar y recibir información relacionada con el progreso.

El backend del sistema será desarrollado en Java utilizando el framework Spring Boot. Esta tecnología permitirá estructurar el servidor de manera organizada, permitiendo la implementación de la lógica de negocio, la validación de datos y la gestión de las operaciones del sistema. Por medio de Spring Boot se expondrá una API REST que permitirá la comunicación entre la aplicación móvil y el servidor mediante solicitudes HTTP, intercambiando información en formato JSON. El backend actuará como intermediario

entre el cliente y la base de datos, asegurando que todas las reglas del sistema se apliquen de forma centralizada y consistente.

Para la persistencia de la información se utilizará PostgreSQL como sistema de gestión de base de datos. En esta base de datos se almacenarán los datos correspondientes a usuarios, rutinas, ejercicios y registros de progreso. La conexión a PostgreSQL será gestionada exclusivamente por el backend, lo que garantiza que el acceso a la información esté controlado y que los datos mantengan su integridad y coherencia a lo largo del tiempo. Esta estructura permite que el sistema pueda crecer y adaptarse sin comprometer la estabilidad de la información almacenada.

5 Flujo de Comunicación del sistema

El flujo general de funcionamiento del sistema será el siguiente:

1. La aplicación móvil envía una solicitud HTTP al backend.
2. El backend recibe la solicitud a través de un controlador.
3. El servicio correspondiente procesa la información aplicando la lógica de negocio.
4. Si es necesario, se realiza una consulta o modificación en la base de datos.
5. El backend devuelve una respuesta en formato JSON.
6. La aplicación móvil procesa la respuesta y actualiza la interfaz del usuario.

Este flujo garantiza que la comunicación sea estructurada y controlada entre los diferentes componentes del sistema.

6 Ejecución Futura del Proyecto

Para ejecutar el proyecto en un entorno de desarrollo se requerirá:

- Flutter SDK instalado.
- Java JDK 17 o superior.
- Maven o Gradle.
- PostgreSQL configurado.
- Un entorno de desarrollo compatible.

Ejecución del backend:

1. Configurar la conexión a la base de datos en el archivo de propiedades.
2. Ejecutar el servidor mediante Maven o Gradle.
3. Verificar que el backend esté disponible en el puerto configurado.

Ejecución de la aplicación móvil:

1. Instalar dependencias utilizando el comando correspondiente de Flutter.
2. Ejecutar la aplicación en un emulador o dispositivo físico.
3. Confirmar que la aplicación se comunique correctamente con el backend.

Referencias

1. Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine (2000).
2. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Sebastopol (2007).
3. Walls, C.: Spring Boot in Action. Manning Publications, Shelter Island (2016).
4. Pivotal Software, Inc.: Spring Boot Reference Documentation. In: Spring Official Documentation. VMware, Inc., https://docs.spring.io/spring-boot/docs/current/reference/html/last accessed 2026/02/11.
5. Google Developers: Flutter Documentation. https://docs.flutter.dev, last accessed 2026/02/11.
6. PostgreSQL Global Development Group: PostgreSQL Documentation. https://www.postgresql.org/docs/, last accessed 2026/02/11.
7. Sommerville, I.: Software Engineering. 10th edn. Pearson, Boston (2016).