CleanCoders
Diseño del proyecto TheraPose
Versión 2.0

Historial de Revisiones

Fecha	Versión	Descripción	Autor
04/06/2025	1.0	Versión preliminar como propuesta de desarrollo.	CleanCoders
11/06/2025	1.0	Revisión final previo a la presentación del proyecto.	CleanCoders
30/06/2025	2.0	Elaboración del diseño de la base de datos centralizada mediante modelo entidad-relación (MER).	Juan Suárez

1. Arquitectura del sistema

Este documento describe la arquitectura general del sistema de TheraPose, una plataforma terapéutica de gestión de yoga. El sistema está diseñado como una aplicación web cliente-servidor que permite a los instructores registrar usuarios, gestionar sesiones y almacenar datos en una base de datos SQLite.

TheraPose es una aplicación web monolítica contenerizada que utiliza servicios externos como:

- Keycloak para la auntenticación
- PostgreSQL para almacenar los datos de identidad.

Así, la aplicación principal está definida como una sola instancia de FastAPI y tiene un despliegue a través de Docker-compose para los servicios de:

- FastAPI: Un solo contenedor con toda la aplicación monolítica.
- Keycloak: Servicio de autenticación externo (no es parte de la lógica del negocio).
- PostgreSQL: Base de datos para Keycloak únicamente.

El sistema utiliza SQLite para el almacenamiento de datos de aplicaciones y sirve plantillas HTML con estilo Bootstrap.

1.1. Arquitectura de Orquestación de Contenedores con Docker Compose

TheraPose utiliza una arquitectura de contenedores múltiples orquestada a través de Docker Compose. El sistema consta de tres servicios principales que trabajan juntos para proporcionar la plataforma terapéutica completa de yoga.

1.1.1. Especificaciones del contenedor

Servicio PostgreSQL: El contenedor PostgreSQL sirve como backend de base de datos para los datos de usuario y configuración de Keycloak.

Configuration	Value
Image	postgres:15
Container Name	keycloak_postgres
Port Mapping	5432:5432
Database	keycloak
Credentials	keycloak/keycloak
Volume Mount	postgres_data:/var/lib/postgresql/data

Servicio KeyCloak: El contenedor de Keycloak proporciona servicios de autenticación y autorización mediante una imagen personalizada que incluye una configuración predefinida del realm**.**

Configuration	Value
Image	bryanhert/keycloak-yoga:26.1.3
Container Name	keycloak
Port Mapping	8081:8080
Admin Credentials	admin/admin
Startup Command	start-dev --import-realm

Variables de Entorno Clave para el Contenedor Keycloak

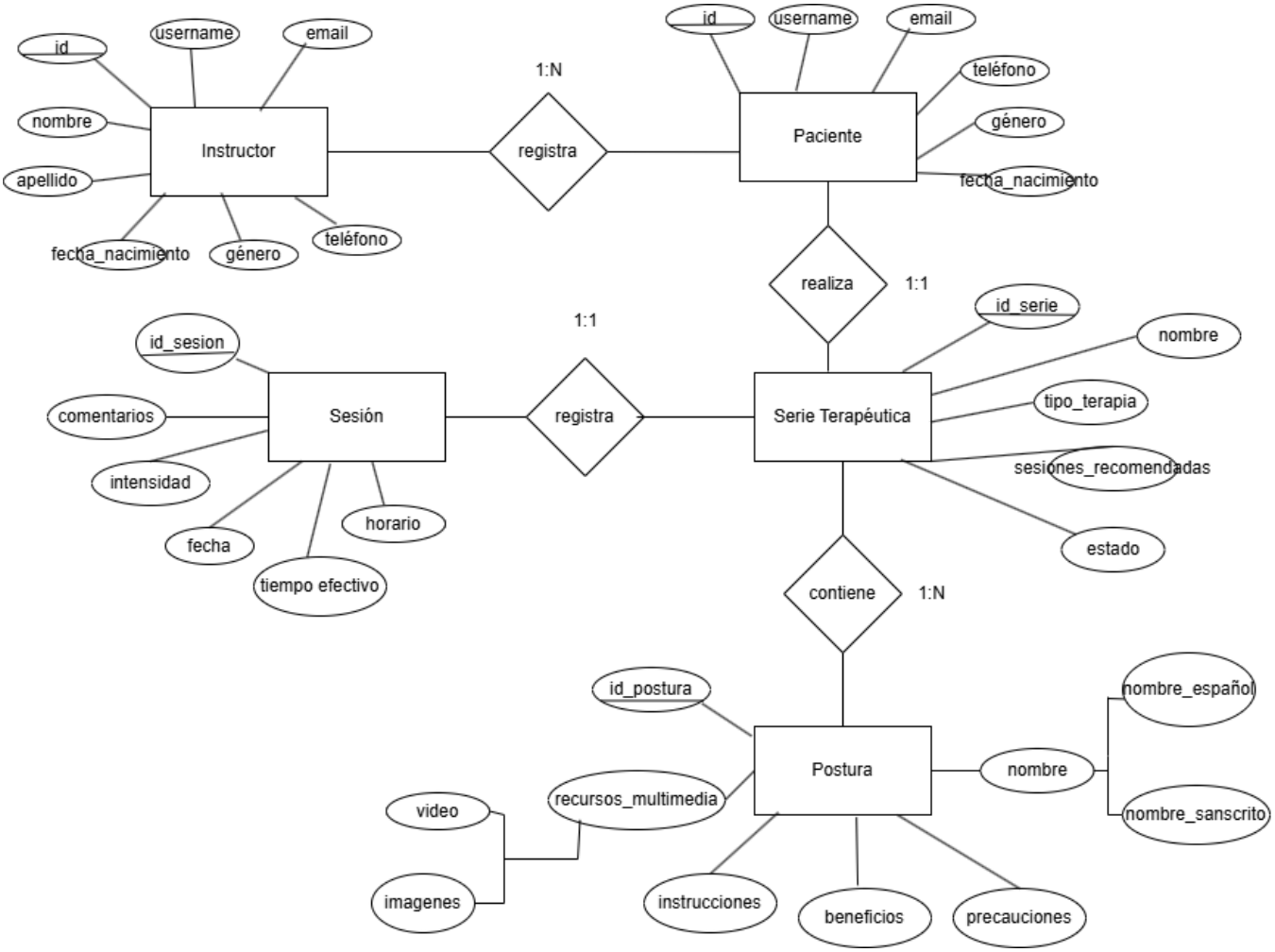
Variable	Valor	Descripción
KC_DB	postgres	Define el tipo de base de datos utilizada por Keycloak (en este caso, PostgreSQL).
KC_DB_URL	jdbc:postgresql://postgres:5432/keycloak	URL JDBC que especifica la conexión a la base de datos PostgreSQL.
KC_HOSTNAME	localhost	Define el nombre de host del servicio Keycloak, usado para generar enlaces en las respuestas del servidor.
KC_HTTP_ENABLED	true	Habilita HTTP sin TLS; útil en entornos de desarrollo (no recomendado para producción).

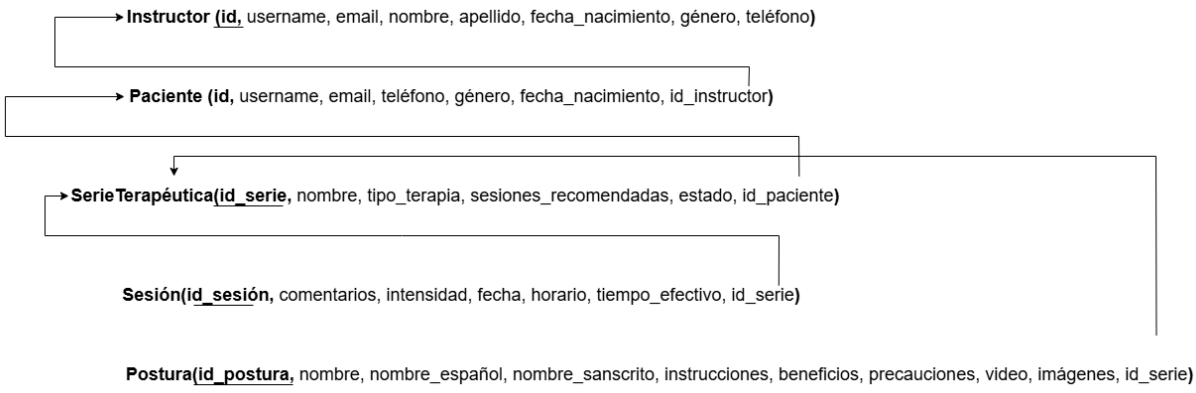
Servicio de FastApi: Ejecuta la aplicación principal del backend, desarrollada en Python y estructurada con el framework FastAPI. Este contenedor se construye a partir de un Dockerfile personalizado que recibe todas las dependencias y configuraciones necesarias para el despliegue de la aplicación

Configuration	Value
Build Context	Current directory (.)
Dockerfile	Dockerfile.fastapi
Container Name	fastapi
Port Mapping	8002:8002
Startup Command	uvicorn proyecto.src.main:app --host 0.0.0.0 --port 8002

1.2. Diseño de la base de datos relacional

TheraPose utiliza SQLite como su base de datos principal para el almacenamiento de datos de la aplicación, separada de la base de datos PostgreSQL utilizada por Keycloak para la autenticación. Así, se utilizará una BD relacional y se diseñará mediante el Modelo Entidad-Relación junto con la notación de Chen.




Instructor (id, username, email, nombre, apellido, fecha_nacimiento, género, teléfono)
Paciente (id, username, email, teléfono, género, fecha_nacimiento, id_instructor)
SerieTerapéutica(id_serie, nombre, tipo_terapia, sesiones_recomendadas, estado, id_paciente)
Sesión(id_sesión, comentarios, intensidad, fecha, horario, tiempo_efectivo, id_serie)
Postura(id_postura, nombre, nombre_español, nombre_sanscrito, instrucciones, beneficios, precauciones, video, imágenes, id_serie)

1.3. Diagramas de secuencia

Un diagrama de secuencia muestra como los objetos o componentes del sistema interactúan entre en el tiempo.

1.3.1 Flujo de registro de instructor.

Este diagrama describe el proceso secuencial para registrar un nuevo instructor en el sistema.

- Interacción entre el cliente, el formulario, el servidor (Keycloak), y la base de datos (SQLite).
- Los caminos de éxitos o error
- El flujo de datos y las respuetas.

2. CI/CD e Integración continua

2.1. Jenkins Pipeline

TheraPose implementa una integración continúa basada en Jenkins, que automatiza el proceso de despliegue en el entorno de desarrollo y pruebas. La pipeline definida en el archivo Jenkinsfile realiza las siguientes tareas:

1. Detención de contenedores existentes:

   docker-compose down

2. Reconstrucción e inicio de servicios:

   docker-compose up d --build

3. Anexos

Script SQL

```
CREATE TABLE [instructors] (
  [instructor_id] text PRIMARY KEY,
  [username] text,
  [email] text,
  [first_name] text,
  [last_name] text,
  [fecha_nac] text,
  [genero] text,
  [celular] text,
  [created_at] timestamp
)

CREATE TABLE [instructor_patients] (
  [id] integer PRIMARY KEY,
  [instructor_id] text,
  [patient_id] text,
  [created_at] timestamp
)

CREATE TABLE [patients] (
  [patient_id] text PRIMARY KEY,
  [username] text,
  [email] text,
  [first_name] text,
  [last_name] text,
  [fecha_nac] text,
```

```
  [genero] text,
  [celular] text,
  [created_at] timestamp
)



CREATE TABLE [serie_terapeutica] (
  [id_serie] integer PRIMARY KEY,
  [nombre] text,
  [tipo_terapia] text,
  [patient_id] text,
  [activa] boolean,
  [sesiones_recomendadas] integer,
  [patiend_id] text
)

CREATE TABLE [postura_en_serie] (
  [id_serie] integer,
  [id_postura] integer,
  [orden] intenger,
  [duracion_min] integer,
  PRIMARY KEY ([id_serie], [id_postura])
)

CREATE TABLE [postura] (
  [id_postura] integer PRIMARY KEY,
  [nombre_es] text,
  [nombre_sans] text,
  [instrucciones] text,
  [beneficios] text,
  [precauciones] text,
  [video] text,
  [fotografia] text
)

CREATE TABLE [sesion] (
  [id_sesion] integer PRIMARY KEY,
  [id_serie] integer,
  [fecha] date,
  [hora_inicio] time,
  [hora_fin] time,
  [intensidad_inicio] integer,
  [intensidad_final] integer,
  [comentario] text,
  [tiempo_efectivo] real
)
```

ALTER TABLE [instructor_patients] ADD FOREIGN KEY ([instructor_id]) REFERENCES [instructors] ([instructor_id])

ALTER TABLE [instructor_patients] ADD FOREIGN KEY ([patient_id]) REFERENCES [patients] ([patient_id])

ALTER TABLE [serie_terapeutica] ADD FOREIGN KEY ([patient_id]) REFERENCES [patients] ([patient_id])

ALTER TABLE [postura_en_serie] ADD FOREIGN KEY ([id_serie]) REFERENCES [serie_terapeutica] ([id_serie])

ALTER TABLE [postura_en_serie] ADD FOREIGN KEY ([id_postura]) REFERENCES [postura] ([id_postura])

ALTER TABLE [sesion] ADD FOREIGN KEY ([id_serie]) REFERENCES [serie_terapeutica] ([id_serie])

ALTER TABLE [serie_terapeutica] ADD FOREIGN KEY ([patiend_id]) REFERENCES [patients] ([patient_id])