Memoria Final – Práctica 4: Servicios y Procesos

Título: Práctica 4 – Servicios y Procesos
Alumno: Rocío Fernández Besoy
Curso: 2º DAM, Programación de Servicios y Procesos
Fecha: 16/12/2025

1. Contexto

Descripción del proyecto:
Este proyecto tiene como objetivo desarrollar una aplicación de gestión de proyectos y tareas, implementando una arquitectura cliente-servidor en Java con comunicación mediante sockets TCP y persistencia en MySQL.
El sistema permite crear, leer, actualizar y eliminar proyectos y tareas, mostrando cómo interactúan cliente y servidor. La comunicación se realiza a través de strings delimitados por ;, y las respuestas se muestran en la consola del cliente.
El cliente es una aplicación de consola, mientras que el servidor centraliza toda la lógica y el acceso a la base de datos. La persistencia se gestiona mediante JDBC y PreparedStatement para evitar inyecciones SQL.

2. Caso práctico

Objetivo:
Permitir la gestión eficiente de proyectos y sus tareas asociadas, con un sistema centralizado en servidor y clientes que puedan realizar operaciones CRUD de forma sencilla.

Funcionalidades principales:
Proyectos: Crear, leer, actualizar y eliminar proyectos.
Tareas: Crear, leer, actualizar y eliminar tareas asociadas a proyectos.

Diseño inicial:

Clases principales:

- Proyecto: representa un proyecto con atributos id(int) y nombre(String).
- Tarea: representa una tarea con atributos: id(int), nombre(String), descripcion(String), estado (String:pendiente, en progreso, completa), urgencia(String: alta, media, baja), fecha_inicio(LocalDate), fecha_fin(LocalDate) e id_proyecto(int).
- ProyectoDAO y TareaDAO: gestionan operaciones CRUD sobre la base de datos.
- Servidor: Escucha conexiones, interpreta los comandos enviados por los clientes y llama a los métodos DAO correspondientes
- Cliente: interfaz de usuario en consola que envía comandos al servidor y muestra las respuestas
- ConexionBD: clase de utilidad que maneja la conexión JDBC con MySQL.

Modelo relacional de la base de datos:

Tabla proyecto
id INT AUTO_INCREMENT PK
nombre VARCHAR(100)

Tabla tarea
id INT PK
nombre VARCHAR(100)
descripción VARCHAR(250)
fecha_inicio DATE
fecha_fin DATE
urgencia ENUM (‘alta’, ‘media’, ‘baja’)
estado ENUM (‘pendiente’, ‘en progreso’ , ‘finalizado’)
proyecto_id INT FK

Relación: Un proyecto puede tener varias tareas (1:N).

3. Arquitectura del sistema

Arquitectura cliente-servidor:

Servidor:

Escucha conexiones entrantes en el puerto 5000.

Procesa comandos CRUD enviados por los clientes.

Conecta con la base de datos mediante JDBC para persistencia.

Atiende a los clientes de forma secuencial, no concurrente.

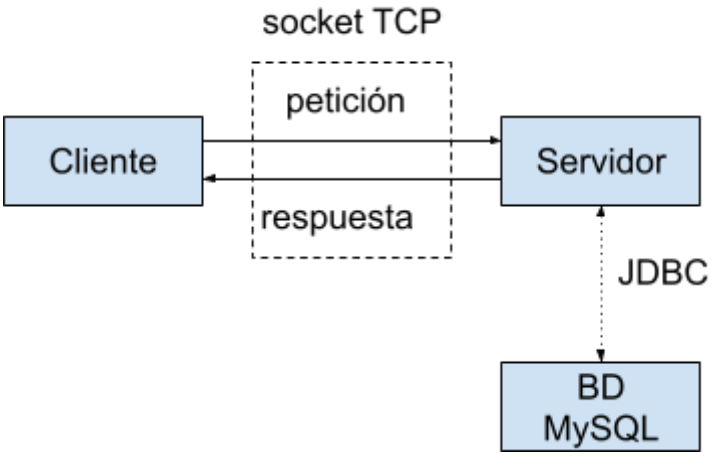
Cliente:

Interfaz de usuario (consola).

Valida los datos introducidos por el usuario.

Envía comandos al servidor y muestra las respuestas.

Diagrama simplificado de comunicación:



El cliente envía un comando delimitado por ; (ej. "INSERT_PROYECTO;NombreProyecto")

El servidor interpreta el comando, ejecuta la operación en la base de datos y devuelve un resultado.

El cliente muestra la respuesta al usuario.

4. Funcionamiento CRUD y comunicación

CRUD de Proyectos:

Crear proyecto: "INSERT_PROYECTO;Nombre"

Crea un nuevo proyecto con el nombre indicado

Leer proyectos: "LIST_PROYECTOS"

Devuelve la lista de todos los proyectos

Actualizar nombre: "UPDATE_PROYECTO;id;nuevoNombre"

Actualiza el nombre de un proyecto

Borrar proyecto: "DELETE_PROYECTO;id"

Elimina un proyecto por su id

CRUD de Tareas:

Crear tarea:

"INSERT_TAREA;nombre;descripcion;estado;urgencia;fecha_inicio;fecha_fin;idProyecto"

Crea una nueva tarea asociada a un proyecto

Leer tareas por proyecto:

"LIST_TAREAS;idProyecto"

Devuelve la lista de tareas de un proyecto

Actualizar tarea:

"UPDATE_TAREA;id;campo;valor"

Modifica un campo de una tarea (fecha_fin, estado, urgencia)

Borrar tarea:

"DELETE_TAREA;id"

Elimina una tarea por su ID

Comunicación:

Uso de sockets TCP para enviar y recibir comandos.

Servidor centraliza toda la lógica y acceso a la base de datos.

Cada operación CRUD genera un mensaje de respuesta que el cliente muestra al usuario.
Ejemplo de flujo:
Cliente envía: "INSERT_PROYECTO;Proyecto1"
Servidor inserta en proyecto y responde: "OK Proyecto creado"
Cliente muestra: Servidor: OK Proyecto creado

5. Dificultades y conclusiones

Dificultades encontradas:
Durante la realización de esta práctica me encontré con varios retos, principalmente relacionados con mi nivel inicial de conocimientos en Java y en conceptos de desarrollo de aplicaciones cliente-servidor:

- Organización del código y diseño de clases: Al principio me resultaba complicado comprender cómo distribuir correctamente las responsabilidades entre las diferentes clases y paquetes. No estaba familiarizada con la idea de separar la lógica de negocio, la comunicación con el servidor y la persistencia de datos en capas distintas, y tuve que aprender cómo estructurar la aplicación de forma clara y coherente.
- Comprender y utilizar JDBC: Antes de esta práctica no había trabajado con JDBC, por lo que fue un reto entender cómo se realiza una conexión a la base de datos, cómo se ejecutan consultas SQL desde Java y cómo se gestionan los resultados. Aprender a utilizar Connection, PreparedStatement y ResultSet fue clave para poder implementar las operaciones CRUD de forma segura.
- Concepto y uso de clases DAO: No estaba familiarizada con la idea de crear clases DAO (Data Access Object) para encapsular el acceso a la base de datos. Con ejemplos y guía logré entender cómo delegar correctamente la gestión de datos a estas clases.
- Validación de datos y comunicación cliente-servidor: Otro reto fue garantizar que los datos introducidos por el usuario fueran correctos antes de enviarlos al servidor, así como entender cómo se implementa la comunicación mediante sockets TCP de manera secuencial y confiable.

Conclusiones:

El proyecto demuestra una correcta implementación de una arquitectura cliente-servidor en Java. Las operaciones CRUD sobre proyectos y tareas funcionan correctamente, con persistencia en MySQL mediante JDBC. La comunicación entre cliente y servidor mediante sockets TCP es estable y fiable. La práctica me ha permitido adquirir experiencia práctica en

el uso de DAO, JDBC y la organización de clases según responsabilidades, consolidando conocimientos sobre arquitectura cliente-servidor y gestión de datos. Si bien la aplicación podría mejorarse para soportar concurrencia y validaciones más estrictas, cumple con los objetivos de la práctica y constituye un caso práctico útil para comprender cómo se integra la lógica de negocio, la comunicación y la persistencia en un sistema distribuido.