**DOCUMENTACIÓN PROYECTO GESTOR DE TAREAS (TODO APP)**

# 📝 DOCUMENTACIÓN PROYECTO GESTOR DE TAREAS (TODO APP)

## Resumen de la Funcionalidad

Esta aplicación web permite gestionar una lista de tareas personales o laborales mediante las siguientes funcionalidades clave:

- \*\*Listar tareas:\*\* Carga y muestra todas las tareas almacenadas en la base de datos.

- \*\*Añadir tareas:\*\* Permite al usuario crear nuevas tareas con nombre, descripción y estado.

- \*\*Eliminar tareas:\*\* Opción para eliminar tareas existentes.

- \*\*Actualizar estado:\*\* Cambiar el estado de una tarea (por ejemplo, `todo`, `in progress`, `done`).

El sistema sigue una arquitectura cliente-servidor, donde el \*\*frontend\*\* consume una \*\*API REST Java\*\* que gestiona las operaciones CRUD sobre una base de datos \*\*MySQL\*\* local.

--

## ⚙️ Tecnologías Utilizadas

### Backend

- \*\*Java 17+\*\* (probado en JDK 23)

- \*\*`com.sun.net.httpserver.HttpServer`\*\* para servir la API REST

- \*\*Gson\*\* para convertir objetos Java ↔ JSON

- \*\*MySQL (XAMPP/MariaDB)\*\* como base de datos relacional

- \*\*JDBC\*\* para conexión y consultas SQL

- \*\*MySQL Connector/J 9.3.0\*\* incluido en las librerías del proyecto

- \*\*Gestión CORS\*\* implementada manualmente

### Frontend

- \*\*HTML5, CSS3 y JavaScript (Vanilla)\*\*

- \*\*Materialize CSS\*\* para diseño responsivo y componentes UI

- \*\*Fetch API\*\* para comunicación con el backend (`GET`, `POST`, `DELETE`, `OPTIONS`)

---

## 🧰 Requisitos

- XAMPP (MySQL activo en puerto 3306)

- Java JDK 17 o superior

- NetBeans (u otro IDE)

- Conector JDBC (`mysql-connector-j-9.3.0.jar`)

- Gson (`gson-2.10.jar`)

- Navegador con soporte para `fetch()`

---

## Arquitectura y Flujo

1. \*\*Frontend:\*\* Renderiza la interfaz y responde a interacciones del usuario.

2. \*\*Fetch API:\*\* Envía peticiones HTTP al backend.

3. \*\*Backend Java:\*\* Procesa las peticiones, ejecuta consultas SQL y responde en formato JSON.

4. \*\*MySQL (XAMPP):\*\* Almacena las tareas en la base `todo\_app`.

---

## 🗃️ Base de Datos

\*\*Nombre:\*\* `todo\_app`

\*\*Tabla:\*\* `tareas`

```sql

CREATE DATABASE IF NOT EXISTS todo\_app;

USE todo\_app;

CREATE TABLE IF NOT EXISTS tareas (

id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

description VARCHAR(255),

estado VARCHAR(20) DEFAULT 'todo'

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO tareas (nombre, description, estado)

VALUES ('Estudiar Java', 'Recordatorio para 2º', 'todo'),

('Hacer proyecto DAW', 'Entrega final', 'done');

```

---

## 🔌 Conexión a la Base de Datos (JDBC)

\*\*Archivo:\*\* `ConexionBD.java`

```java

private static final String URL =

"jdbc:mysql://localhost:3306/todo\_app?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true";

private static final String USER = "todo\_user";

private static final String PASS = "TU\_CONTRASEÑA";

```

\*\*Librerías añadidas al proyecto:\*\*

- `mysql-connector-j-9.3.0.jar`

- `gson-2.10.jar`

---

## 🌐 API REST

\*\*Base URL:\*\* `http://localhost:8000/tareas`

| Método | Endpoint | Descripción |

|--------|-----------|-------------|

| `GET` | `/tareas` | Devuelve todas las tareas |

| `POST` | `/tareas` | Inserta una nueva tarea |

| `DELETE` | `/tareas/{id}` | Elimina una tarea |

| `OPTIONS` | `/tareas` | Maneja preflight CORS |

Todas las respuestas incluyen los encabezados CORS:

```java

exchange.getResponseHeaders().add("Access-Control-Allow-Origin", "\*");

exchange.getResponseHeaders().add("Access-Control-Allow-Methods", "GET, POST, DELETE, OPTIONS");

exchange.getResponseHeaders().add("Access-Control-Allow-Headers", "Content-Type");

``

---

## 🧩 Clases Principales

- \*\*`Main.java`\*\* → Inicia el servidor en `http://localhost:8000`

- \*\*`Tarea.java`\*\* → Modelo de datos

- \*\*`ConexionBD.java`\*\* → Manejo de conexión JDBC

- \*\*`TareaDAO.java`\*\* → Operaciones CRUD (MySQL)

- \*\*`TareasHandler.java`\*\* → Controlador HTTP (GET, POST, DELETE)

---

## 💻 Frontend

\*\*Estructura:\*\*

```

frontend/

│

├── index.html

├── style.css

└── app.js

```

\*\*Ejemplo de carga de tareas:\*\*

```js

async function cargarTareas() {

const res = await fetch("http://localhost:8000/tareas");

const tareas = await res.json();

// Renderiza en HTML

}

```

---

## PASO A PASO

1. \*\*Crea tu repositorio Git:\*\* Utilizo Git Bash.

2. \*\*Inicia MySQL en XAMPP\*\* (`db/schema.sql`). Arranca MySQL desde el panel de control.

- Base de datos: `todo\_app`

- Tabla: `tareas`

3. \*\*Ejecuta el backend\*\*

- En NetBeans, ejecuta `Main.java`.

- Verifica en la consola que el servidor HTTP se haya iniciado (debería mostrar que la API REST está escuchando):

`Servidor iniciado en http://localhost:8000`

4. \*\*Abre el frontend\*\*

- Usa Live Server (VSCode) o abre `index.html`.

- Verifica en la consola del navegador que carga las tareas.

- - Para evitar problemas de CORS, lo ideal es:

- Servir tu frontend desde un servidor local (ej. Live Server en VSCode), o

- Abrir el navegador con configuración que permita CORS en modo desarrollo, o

- Abrir el frontend también con un servidor local (`netbeans`, `python -m http.server`, etc).

- Si abres directo el archivo HTML con `file://`, algunas funciones fetch pueden fallar.

5. \*\*Testea por separado:\*\*

- Primero: prueba solo el frontend con JS (sin backend).

- Luego: ejecuta el backend en consola y verifica inserciones en MySQL.

- Al abrir el frontend, debería ejecutarse el método `cargarTareas()` que hace una petición GET a `http://localhost:8000/tareas`.

- Si el backend está bien y la DB tiene datos, deberías ver listadas las tareas.

6. \*\*Probar añadir una nueva tarea\*\*

- Escribe una tarea en el formulario y presiona el botón "Añadir tarea".

- El fetch hace un POST a la API REST enviando la nueva tarea.

- Si todo funciona, la tarea se insertará en la base de datos y la lista se actualizará automáticamente.

7. \*\*Confirmar que la tarea fue añadida\*\*

- Puedes volver a cargar la página para asegurarte que la tarea sigue ahí (la lee de la base de datos).

- O usa un cliente MySQL para consultar directamente la tabla tareas:

```sql

SELECT \* FROM tareas;

```

8. \*\*Depurar posibles errores\*\*

- Si no ves las tareas cargando o no se añade la tarea, abre la consola del navegador (F12 > Consola) para ver si hay errores JavaScript o de red (fetch).

- En el backend, revisa la consola donde ejecutas Java para ver si aparece algún error o excepción.

---

## ERRORES DETECTADOS

### Cómo habilitar CORS en tu backend Java simple

Como estás usando un servidor básico hecho en Java (probablemente con `HttpServer` o similar), debes agregar los headers CORS en las respuestas para que el navegador permita las peticiones.

Ejemplo básico para agregar CORS en tu manejador de `/tareas` (en el método `handle(HttpExchange exchange)` agrega estas líneas justo antes de enviar la respuesta):

```java

// Permitir cualquier origen (para desarrollo)

exchange.getResponseHeaders().add("Access-Control-Allow-Origin", "\*");

exchange.getResponseHeaders().add("Access-Control-Allow-Methods", "GET, POST, OPTIONS");

exchange.getResponseHeaders().add("Access-Control-Allow-Headers", "Content-Type");

## SOLUCIÓN DE PROBLEMAS

| Error | Causa | Solución |

|-------|--------|-----------|

| `Communications link failure` | MySQL apagado o puerto incorrecto | Inicia MySQL en XAMPP (3306) |

| `Access denied for user` | Usuario/clave incorrectos | Verifica `todo\_user` y contraseña |

| `No suitable driver` | Falta el conector JDBC | Añade `mysql-connector-j-9.3.0.jar` a Libraries |

| `CORS` bloqueado | Falta cabeceras en Java | Agrega headers `Access-Control-\*` en todas las respuestas |

---

## CRÉDITOS

Proyecto educativo tipo Trello (ToDo App) con Java, MySQL y HTML/JS.

Estructura simple y portable, pensada para aprendizaje y despliegue local.

**ERRORES DETECTADOS:**

## Cómo habilitar CORS en tu backend Java simple

Como estás usando un servidor básico hecho en Java (probablemente con HttpServer o similar), debes agregar los headers CORS en las respuestas para que el navegador permita las peticiones.

### Ejemplo básico para agregar CORS en tu manejador de /tareas

Si usas com.sun.net.httpserver.HttpServer, en el handle(HttpExchange exchange) agrega estas líneas justo antes de enviar la respuesta:

// Permitir cualquier origen (para desarrollo)

exchange.getResponseHeaders().add("Access-Control-Allow-Origin", "\*");

exchange.getResponseHeaders().add("Access-Control-Allow-Methods", "GET, POST, OPTIONS");

exchange.getResponseHeaders().add("Access-Control-Allow-Headers", "Content-Type");

**ERROR java.sql.SQLException: Field 'nombre' doesn't have a default value**

java.sql.SQLException: Field 'nombre' doesn't have a default value

**¿Qué significa?**  
Que tu tabla en MySQL tiene una columna llamada nombre que está definida como NOT NULL y no tiene valor por defecto, y al hacer el INSERT no le estás pasando ningún valor para esa columna.

### ¿Cómo arreglarlo?

Tienes dos opciones:

### 1. Modificar la tabla en la base de datos para que nombre pueda ser NULL o tenga un valor por defecto

Ejemplo SQL para permitir NULL en la columna nombre:

ALTER TABLE tareas MODIFY nombre VARCHAR(255) NULL;

O para poner un valor por defecto (ejemplo vacío):

ALTER TABLE tareas MODIFY nombre VARCHAR(255) NOT NULL DEFAULT '';

### 2. Ajustar tu código Java para que al insertar tareas también proporcione el campo nombre

Por lo que muestras, tu clase Tarea y el JSON que envías solo tienen descripcion. Si la tabla espera nombre, entonces:

* Debes incluir nombre en tu objeto Tarea y en el JSON que envías desde el frontend.
* Modificar el método insert de TareaDAO para que también inserte el valor nombre.

### Cómo identificar el problema con más detalle

Revisa la estructura actual de tu tabla en MySQL con:

DESCRIBE tareas;

Ahí verás qué columnas hay, si nombre es obligatoria, etc.

### Para probar rápido:

Si tu tabla tiene nombre como NOT NULL y no tiene valor por defecto, y no quieres complicarte, cambia la tabla para que esa columna pueda ser NULL o tenga un default. Luego prueba tu app de nuevo y debería funcionar.

**MEJORAS**

* 1. Ocultar datos de usuario y contraseña en ConexiónBD.java utilizando un archivo externo db.properties
  2. Mejora de apariencia visual de tarjetas añadiendo datos en styles.css
  3. Añadir funcionalidad Drag & Drop con SortableJS en scripts.js para que las tarjetas puedan arrastrarse y añadirse a diferentes estados
  4. Nuevo estado IN PROGRESS añadido. Se elimina lógica antigua del campo Completada, y se añade nueva lógica creando campo Estado en BBDD.
  5. Nueva funcionalidad editar tarea