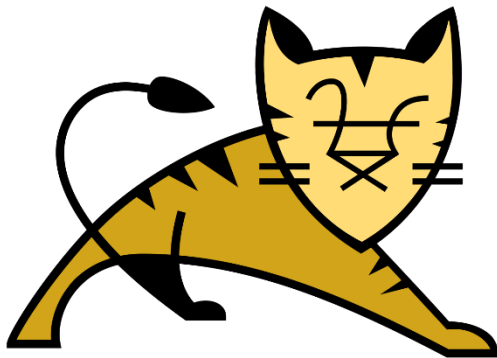


EMV – Entorno de desarrollo Java

Cristina Xu



Le pido a la IA que me cree el prompt para que la IA me cree los archivos como el archivo .yml, el Dockerfile, la estructura del proyecto, etc.

“Hazme una solicitud pidiendo a una IA un contenedor en docker para crear un entorno de desarrollo java incluyendo openjdk, servicio de aplicaciones ligera como tomcat y base de datos mysql.”

Solicitud De Contenedor Docker Para Entorno De Desarrollo Java

Instrucciones para IA: Creación de entorno de desarrollo Java con Docker

Contexto

Actúa como una **IA experta en Docker, Java y entornos de desarrollo**. Tu objetivo es **diseñar y generar un proyecto completo** que permita levantar un entorno de desarrollo Java totalmente funcional mediante contenedores Docker.

Este entorno estará destinado a **desarrollo y pruebas locales**, priorizando simplicidad, buenas prácticas y uso de tecnologías estándar.

Objetivo del proyecto

Crear un entorno de desarrollo Java basado en Docker que incluya:

- Java con OpenJDK
- Un servidor de aplicaciones ligero (Apache Tomcat)
- Una base de datos MySQL

El resultado debe permitir levantar todo el entorno con un solo comando y facilitar el desarrollo de aplicaciones Java web.

Requisitos funcionales

1. Contenedor de Aplicación Java

- Lenguaje: Java
- Distribución: OpenJDK
- Versión recomendada: OpenJDK 17 LTS
- Uso: Compilación y ejecución de aplicaciones Java

- Debe permitir desplegar aplicaciones web en Tomcat (WAR)

2. Servidor de Aplicaciones

- Tecnología: Apache Tomcat
- Versión recomendada: Tomcat 10.x
- Función:
 - Despliegue de aplicaciones Java web
 - Puerto HTTP expuesto (por defecto 8080)

3. Base de Datos

- Motor: MySQL
- Versión recomendada: MySQL 8.x
- Requisitos:
 - Persistencia de datos mediante volúmenes Docker
 - Configuración mediante variables de entorno:
 - Usuario
 - Contraseña
 - Nombre de base de datos
 - Puerto expuesto para conexión desde el contenedor Java y herramientas externas

Requisitos técnicos

- Uso exclusivo de **imágenes oficiales** de Docker Hub.
- Orquestación de servicios mediante **Docker Compose**.
- Definición clara de:
 - Dockerfile (si es necesario para la aplicación Java)
 - docker-compose.yml
- Comunicación entre contenedores mediante red interna Docker.
- Posibilidad de levantar y detener el entorno con:
 - `docker-compose up -d`
 - `docker-compose down`

Estructura esperada del proyecto

La IA debe generar una estructura similar a la siguiente:

```
proyecto-docker-java/  
├── docker-compose.yml  
├── README.md  
├── app/  
│   ├── Dockerfile  
│   └── (código Java de ejemplo o estructura base)  
└── db/  
    └── (scripts SQL opcionales)
```

Documentación requerida

La IA debe proporcionar:

- Instrucciones claras para:
 - Construir las imágenes
 - Levantar el entorno
 - Acceder a Tomcat y MySQL
- Explicación breve de cada servicio y su configuración
- Variables de entorno utilizadas

Buenas prácticas

- Mantener el entorno lo más ligero posible
- Usar versiones LTS
- Evitar configuraciones complejas innecesarias
- Separar responsabilidades por contenedor

Resultado esperado

Un proyecto Docker completamente funcional que permita a cualquier desarrollador:

1. Clonar el proyecto
2. Ejecutar un único comando
3. Tener disponible:
 - Un servidor Tomcat corriendo
 - Java listo para desarrollo
 - Una base de datos MySQL operativa

Fin de las instrucciones para la IA

=====

Primero he creado la estructura del proyecto que me creo la IA (ver README.md).

El archivo “mysql-connector-j-8.1.0.jar” lo he descargado en <https://dev.mysql.com/downloads/connector/j/> para conectar la base de datos con la web.

Levanto el contenedor abriendo un terminal en el directorio de mi carpeta y ejecuto “docker compose up -d”.

```
C:\docker\Java>docker compose up -d
times="2025-12-14T21:21:36+01:00" level=warning msg="C:\\docker\\Java\\docker-compose.yml: the attribute `version` is obsolete, it will
be ignored, please remove it to avoid potential confusion"
[+] Building 5.6s (11/11) FINISHED
=> [internal] load local bake definitions 0.0s
=> => reading from stdin 478B 0.0s
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 335B 0.0s
=> [internal] load metadata for docker.io/library/tomcat:10.1-jdk17-temurin 2.3s
=> [auth] library/tomcat:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/3] FROM docker.io/library/tomcat:10.1-jdk17-temurin@sha256:1c3896808980e90cd874809df16a1e086fa4da79c93bce3cc6d0effca6e6a 1.7s
=> => resolve docker.io/library/tomcat:10.1-jdk17-temurin@sha256:1c3896808980e90cd874809df16a1e086fa4da79c93bce3cc6d0effca6e6a 0.0s
=> => sha256:c5d4b6d4e39f76f9fcc6299595c28f97c238e684ed71b5b2eb07a03cc545254 14.51MB / 14.51MB 1.0s
=> => sha256:d4af6080fef22d7f7d8af392966f596a02c8c8e51d885ef47624e787b60c9b30 140B / 140B 0.5s
=> => extracting sha256:d4af6080fef22d7f7d8af392966f596a02c8c8e51d885ef47624e787b60c9b30 0.1s
=> => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0c5577484a6d75e68dc38e8acc1 0.0s
=> => extracting sha256:c5d4b6d4e39f76f9fcc6299595c28f97c238e684ed71b5b2eb07a03cc545254 0.4s
=> [internal] load build context 0.1s
=> => transferring context: 401B 0.0s
=> [2/3] RUN rm -rf /usr/local/tomcat/webapps/* 0.7s
=> [3/3] COPY src/main/webapp /usr/local/tomcat/webapps/ROOT 0.0s
=> exporting to image 0.3s
=> => exporting layers 0.1s
=> => exporting manifest sha256:cf2339476433cb3ea7661c54549da67c3381519c6bb2231a37ce8ac0a9cd9b35 0.0s
=> => exporting config sha256:122656a89d01501bb3e3ec5704aeabb1b47f8d4e280e92a90a098292945e64d 0.0s
=> => exporting attestation manifest sha256:1bce3109346d16e9b3bc1f870ac29cbcd0e0b912bf3c19d061acde68f11c118a 0.0s
=> => exporting manifest list sha256:c4112169680b0b383354d1c406c079be2bd03b47a410aec3f24c2d4f8947ebfc 0.0s
=> => naming to docker.io/library/java-app:latest 0.0s
=> => unpacking to docker.io/library/java-app:latest 0.0s
=> => resolving provenance for metadata file 0.0s
[+] Running 4/4
✔ java-app Built 0.0s
✔ Network java_java-net Created 0.1s
✔ Volume java_mysql_data Created 0.0s
✗ Container mysql-db Error response from daemon: Conflict. The container name "/mysql-db" is already... 0.1s
Error response from daemon: Conflict. The container name "/mysql-db" is already in use by container "72f648d049148ab305e0802a7b5601f1e
46d159e92d11c2e8f0b97b8e01f84e". You have to remove (or rename) that container to be able to reuse that name.
C:\docker\Java>docker compose down
```

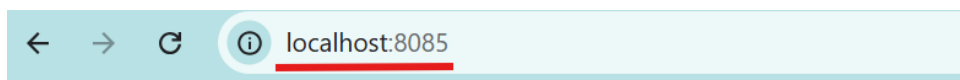
Me da error cuando el contenedor de la base de datos trata de crearse, porque el puerto esta ocupado, por lo tanto cambio 3306 a 3309.

Detengo y elimino los contenedores, y luego los levanto de nuevo para que se aplique el cambio que he realizado.

```
C:\docker\Java>docker compose down
times="2025-12-14T21:24:18+01:00" level=warning msg="C:\\docker\\Java\\docker-compose.yml: the attribute `version` is obsolete, it will
be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
✔ Container java-tomcat Removed 0.0s
✔ Container bd-mysql Removed 0.0s
✔ Network java_java-net Removed 0.4s

C:\docker\Java>docker compose up -d
times="2025-12-14T21:24:34+01:00" level=warning msg="C:\\docker\\Java\\docker-compose.yml: the attribute `version` is obsolete, it will
be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
✔ Network java_java-net Created 0.1s
✔ Container bd-mysql Started 0.6s
✔ Container java-tomcat Started 0.7s
```

Compruebo si funciona la web accediendo a “localhost:8085” desde un navegador.



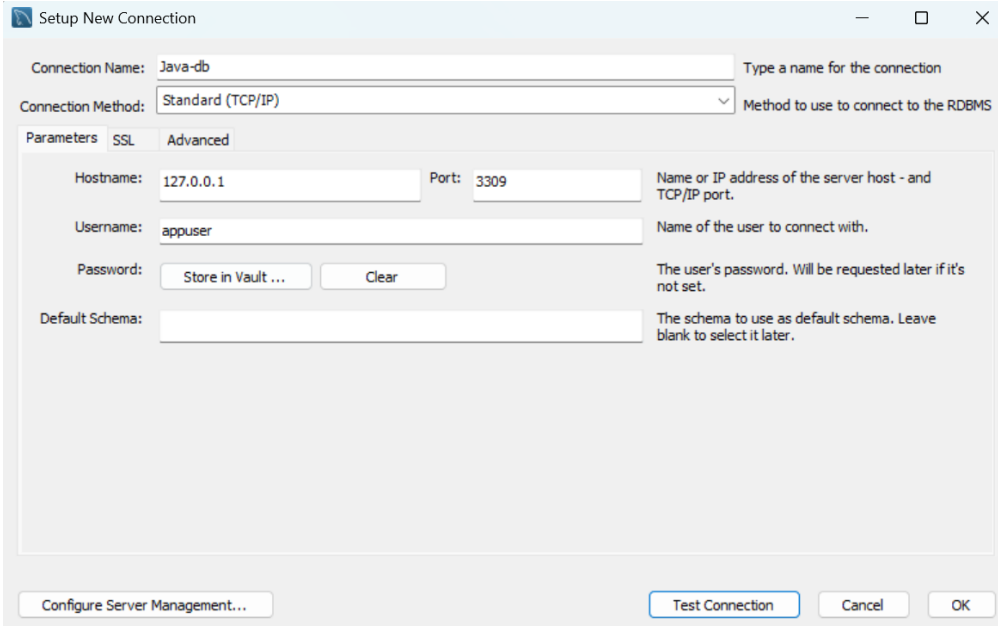
Tomcat funcionando con Docker

Java: OpenJDK 17

Servidor: Apache Tomcat 10

Ahora conecto la base de datos de MySQL. Para ello abro MySQL Workbench y conecto la base de datos añadiendo una nueva conexión y cambiando el puerto por 3309 y el usuario por appuser. Luego realizo un testeo de conexión, me pedirá la contraseña del usuario (password: apppass). Si la conexión se hizo correctamente se mostrara una ventana con un resumen indicando que ha tenido éxito la conexión.

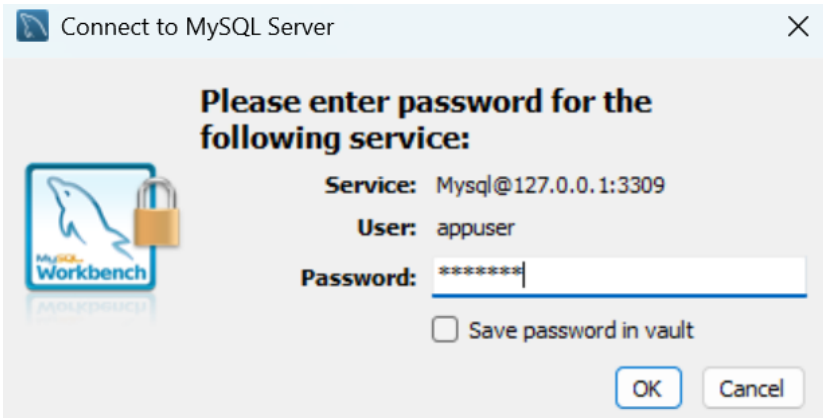
Configuración:



The 'Setup New Connection' dialog box is shown with the following configuration:

- Connection Name:** Java-db
- Connection Method:** Standard (TCP/IP)
- Parameters tab:**
 - Hostname:** 127.0.0.1
 - Port:** 3309
 - Username:** appuser
 - Password:** (masked with dots, with 'Store in Vault ...' and 'Clear' buttons)
 - Default Schema:** (empty)

Buttons at the bottom: 'Configure Server Management...', 'Test Connection', 'Cancel', and 'OK'.



The 'Connect to MySQL Server' dialog box prompts for the password for the service:

- Service:** Mysql@127.0.0.1:3309
- User:** appuser
- Password:** (masked with dots)
- ☐ Save password in vault

Buttons: 'OK' and 'Cancel'.

MySQL Workbench



Successfully made the MySQL connection

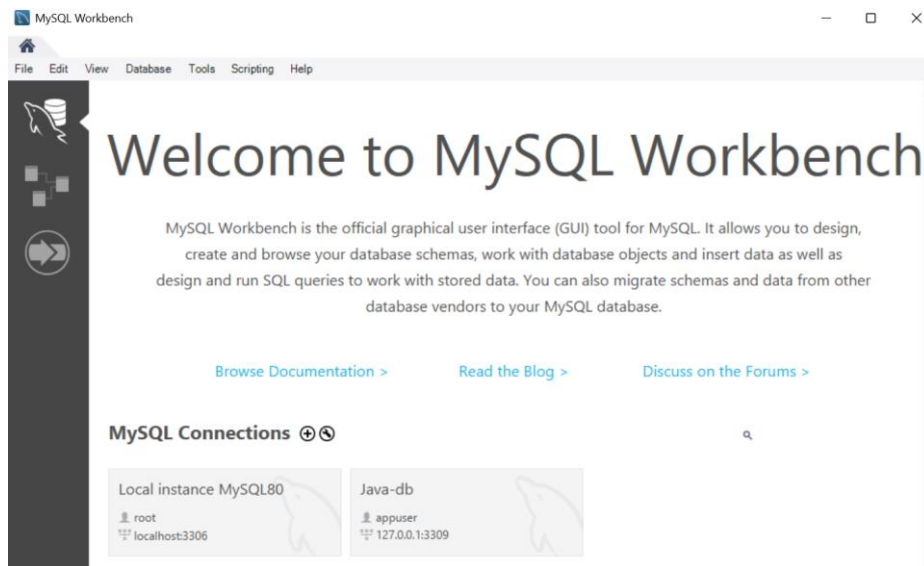
Information related to this connection:

Host: 127.0.0.1
Port: 3309
User: appuser
SSL: enabled with TLS_AES_128_GCM_SHA256

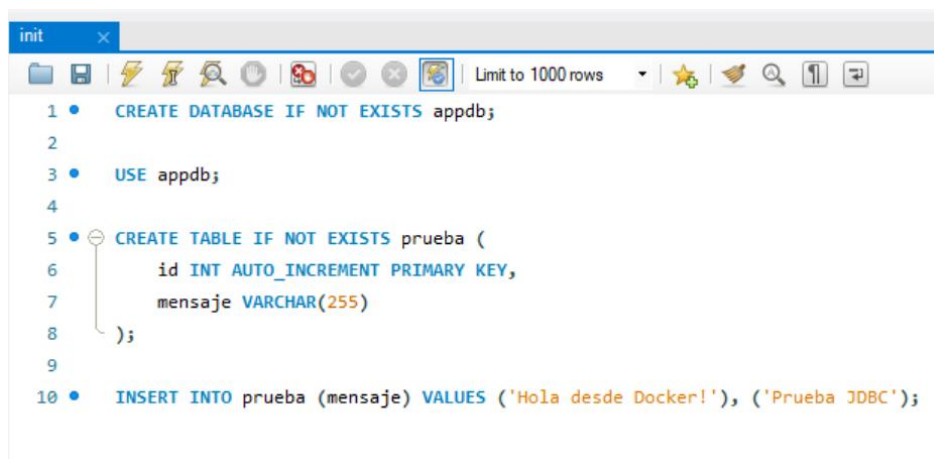
A successful MySQL connection was made with the parameters defined for this connection.

OK

Se ha creado correctamente la base de datos:

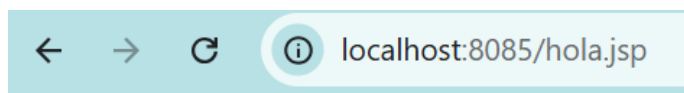


El archivo init.sql, que permitirá comprobar si manda los datos a la web.



Compruebo que la base de datos sí manda los datos accediendo a “localhost:8085/hola.jsp” desde un navegador.

Se puede observar que los mensajes introducidos en la base de datos se muestran.



Hola Mundo JSP c

Datos desde MySQL:

- 1 - Hola desde Docker!
- 2 - Prueba JDBC