# API Online

Afondamento nas Competencias Profesionais

Cristian Fernández

23 de enero de 2026

# Índice

# 1. Introducción

El objetivo de esta tarea es detallar el proceso de configuración y despliegue automatizado de un proyecto (contenedor *Docker API Express.js + MariaDB*) alojado en un repositorio remoto (*GitHub*), garantizando que cualquier cambio en el código fuente se refleje de manera eficiente en el entorno de ejecución.

También buscaremos la Integración Continua mediante la conexión directa entre el repositorio remoto y *Coolify*. Este enfoque permitirá al servidor detectar automáticamente nuevas actualizaciones en la rama principal, iniciar la construcción de las imágenes de *Docker* y desplegar los contenedores sin intervención manual, mejorando sustancialmente la calidad de vida del trabajo consiguiendo, entre otra cosas, reducir errores humanos y optimizar los tiempos de entrega.

Para ello seguiremos los siguiente pasos:

- Crearemos una máquina virtual Ubuntu (versión 24.04.2 LTS) que actuará como nodo principal de despliegue.
- Acceso Remoto y Exposición: Debido a las restricciones de red del entorno local, se emplea Ngrok para crear un túnel seguro que expone la instancia de Coolify y los servicios desplegados a la red pública.
- Orquestación con Coolify: Actúa como el centro de control PaaS (Platform as a Service) que gestiona el ciclo de vida de la aplicación, desde la lectura del repositorio hasta el monitoreo de los contenedores.

# 2. Requisitos funcionales

A continuación se listan los requisitos funcionales del sistema:

**RF-1**: Solicitar al usuario que introduzca su nombre antes de entrar en el chat.

**RF-2**: Pulsar un botón que inicie la ejecución del programa en una nueva ventana.

**RF-3**: Permitir la inserción de mensajes de forma ágil únicamente usando el teclado.

**RF-4**: Mostrar correctamente y en tiempo real todos los mensajes de todos los participantes en el chat.

**RF-5**: Terminar los hilos en ejecución cuando se cierre la aplicación Cliente.

# 3. Requisitos no funcionales

A continuación se listan los requisitos NO funcionales del sistema:

**RNF-1**: El tiempo de respuesta al pulsar el botón *Entrar Chat* no debe superar los 2 segundos.

**RNF-2**: La intefaz gráfica de usuario ha de ser fácilmente entendible.

**RNF-3**: El tiempo de respuesta al mandar un mensaje pulsando Enter no debe superar el segundo.

**RNF-4**: El usuario debe distinguir fácilmente sus mensajes del resto de mensajes de los demás usuarios.

# 4. Casos de uso

A continuación se muestran los casos de uso mediante la siguiente tabla:

| **Caso de uso** | **Descripción** |
|---|---|
| Introducir nombre | El usuario debe introducir su nombre antes de entrar en el chat. |
| Iniciar aplicación | El usuario puede iniciar la ventana de chat pulsando un botón. |
| Escribir mensaje | El usuario puede enviar un mensaje escribiendo y pulsando Enter. |
| Cerrar aplicación | El usuario puede cerrar la aplicación por completo pulsando el botón de cerrar ventana. |

# 5. Historias de usuario

A continuación se muestran las historias de usuario mediante la siguiente tabla:

| **ID** | **Como...** | **Quiero...** | **Para...** |
|---|---|---|---|
| HU-01 | Usuario | Insertar mi nombre | Que me identifiquen en el chat |
| HU-02 | Usuario | Pulsar un botón | Iniciar la aplicación |
| HU-03 | Usuario | Manejar el chat únicamente con el teclado | Disfrutar de una buena usabilidad |
| HU-04 | Usuario | Ver los nombres de los participantes | Saber en todo momento con quién hablo |

# 6. Herramientas de desarrollo

**IntelliJ Community Edition** es un IDE gratuito y de código abierto, ideal para el desarrollo en lenguajes JVM y Android. Sus características principales incluyen un potente editor de código con autocompletado inteligente y refactorización, soporte para control de versiones, un depurador integrado, y herramientas de pruebas unitarias. Está disponible para todas las plataformas.

Características principales:

- **Editor de código potente:** Ofrece resaltado de sintaxis, análisis en tiempo real, sugerencias de código, e inspecciones y correcciones rápidas.
- **Soporte para múltiples lenguajes:** Es compatible con Java, Kotlin, Groovy, Scala, y otros lenguajes JVM.
- **Integración con control de versiones:** Permite la integración con sistemas como Git, SVN, y Mercurial.

# 7. Lenguajes de programación

**JavaFX** se caracteriza por su capacidad de crear aplicaciones enriquecidas visualmente, integrando multimedia (audio, video) y gráficos vectoriales. Permite desarrollar aplicaciones multiplataforma (escritorio, web, móvil, smart TV). Otras características clave son la integración fluida con el ecosistema Java y herramientas como FXML y Scene Builder para un desarrollo más eficiente y colaborativo.

Características principales:

- **Interfaces gráficas de usuario (GUI) enriquecidas:** Permite crear interfaces modernas y atractivas con animaciones, gráficos vectoriales, y elementos personalizables.
- **Integración multimedia:** Facilita la inclusión de contenido de audio, video y otros medios dentro de las aplicaciones.
- **Desarrollo declarativo (FXML):** Utiliza FXML, un lenguaje basado en XML, para describir la interfaz de usuario, lo que facilita el trabajo colaborativo entre diseñadores y desarrolladores.

# 8. Interfaz de la aplicación

A continuación unas capturas de las diferentes vistas de la interfaz.