

2025

Libro de Visitas

JEAN ANDRADE, DAVID tobar

Contenido

[1. Descripción del Proyecto 2](#_Toc201764717)

[2. Integración y Entrega Continua (CI/CD) 3](#_Toc201764718)

[2.1 Relación con CI/CD 3](#_Toc201764719)

[2.2 Etapas del Proceso de CD 3](#_Toc201764720)

[2.3 Beneficios del CI/CD en este Proyecto 4](#_Toc201764721)

[3. Herramientas Utilizadas 4](#_Toc201764722)

[3.1 GitHub 4](#_Toc201764723)

[3.2 GitHub Actions 4](#_Toc201764724)

[3.3 Node.js 4](#_Toc201764725)

[3.4 MySQL Workbench 5](#_Toc201764726)

[4. Arquitectura del Sistema 5](#_Toc201764727)

[Componentes principales: 5](#_Toc201764728)

[5. Estructura de Base de Datos 6](#_Toc201764729)

[Tabla 1: Usuarios 6](#_Toc201764730)

[Tabla 2: Mensajes 6](#_Toc201764731)

[6. Flujo del Usuario 7](#_Toc201764732)

[7. Pruebas Automatizadas 7](#_Toc201764733)

[8. Capturas de Uso de GitHub Actions 8](#_Toc201764734)

[8](#_Toc201764735)

[8. Conclusión y Proyección Futura 10](#_Toc201764736)

## 1. Descripción del Proyecto

La mini aplicación “Libro de Visitas” es una plataforma web que permite a los usuarios registrarse y dejar mensajes breves, los cuales quedan almacenados para su posterior visualización. Esta aplicación está diseñada con un enfoque centrado en la simplicidad, la interacción y la trazabilidad de los usuarios que visitan el sitio. Cada mensaje ingresado por un visitante se guarda junto con su nombre, correo electrónico, contenido del mensaje, así como la fecha y hora exacta en que fue enviado.

Para una estructura de datos robusta y escalable, la aplicación utiliza un modelo relacional, implementado mediante una base de datos compuesta por dos tablas principales: una de usuarios y otra de mensajes. Esta estructura refleja una relación uno a muchos, en la cual un usuario puede dejar múltiples mensajes, pero cada mensaje está asociado únicamente a un solo usuario.

La lógica del backend permite manejar el registro de nuevos usuarios, la validación del contenido de los mensajes (con un límite de 300 caracteres), y la recuperación de los mensajes existentes para ser mostrados en la interfaz. La interfaz de usuario permite una navegación clara e intuitiva, mostrando un listado actualizado de mensajes enviados por los visitantes registrados.

## 2. Integración y Entrega Continua (CI/CD)

### 2.1 Relación con CI/CD

La implementación de CI/CD (Integración Continua y Entrega Continua) ha sido clave en el desarrollo y despliegue de esta mini aplicación, asegurando calidad, rapidez y automatización en cada paso del ciclo de vida del software.

* **Integración Continua (CI):** Permite integrar automáticamente cada nuevo cambio en el código fuente al repositorio central. Por ejemplo, al modificar la lógica de registro de visitantes o ajustar la validación del contenido del mensaje, CI garantiza que esos cambios sean verificados automáticamente mediante pruebas y validaciones antes de ser fusionados.
* **Entrega Continua (CD):** Una vez verificado el código, CD toma ese código funcional y lo despliega de manera automática en entornos de prueba y finalmente en producción, sin intervención manual. Esto asegura que cualquier mejora o corrección esté disponible en el sitio web en cuestión de minutos.

CI/CD en conjunto proporciona una solución automatizada que mejora la calidad del software, reduce errores humanos, y acelera el ciclo de entrega de nuevas funcionalidades.

### 2.2 Etapas del Proceso de CD

1. **Validación del Código:**  
   Cada vez que se hace un commit o se abre un pull request, el sistema ejecuta pruebas automáticas para asegurar que el código cumple con los estándares definidos y no rompe funcionalidades existentes.
2. **Empaquetado del Proyecto:**  
   Utilizando herramientas como GitHub Actions, el proyecto se construye automáticamente, generando un paquete listo para ser desplegado.
3. **Despliegue a Entorno de Pruebas:**  
   Antes de liberar el código al público, se lanza en un entorno de staging o pruebas donde se simulan interacciones reales, como el registro de usuarios o la publicación de mensajes.
4. **Verificación Funcional:**  
   Se ejecutan pruebas funcionales y de integración para asegurar que los flujos principales de la aplicación (registro, validación, guardado y recuperación de mensajes) funcionan correctamente.
5. **Despliegue a Producción:**  
   Si todo ha pasado correctamente, el sistema se despliega automáticamente al entorno de producción. Esto significa que cualquier persona con acceso al sitio web podrá interactuar con la nueva versión del sistema de forma inmediata.

### 2.3 Beneficios del CI/CD en este Proyecto

* Reducción de errores: La validación continua evita que errores críticos lleguen al entorno productivo.
* Automatización del despliegue: Se elimina la necesidad de despliegues manuales, lo cual reduce el tiempo y los riesgos asociados.
* Retroalimentación rápida: Ante cualquier error o fallo, el sistema lo reporta inmediatamente, permitiendo corregirlo a tiempo.
* Colaboración eficiente: Los desarrolladores pueden trabajar en paralelo sin conflictos, gracias al control de versiones y pruebas automatizadas.
* Entrega continua de valor: Las nuevas funcionalidades y mejoras están disponibles de inmediato para los usuarios finales.

## 3. Herramientas Utilizadas

### 3.1 GitHub

Plataforma de control de versiones utilizada para gestionar todo el ciclo de desarrollo. Permite llevar un historial claro de los cambios, revertir errores, trabajar en múltiples ramas y colaborar en equipo. El repositorio centralizado facilita la integración con herramientas de automatización como GitHub Actions.

### 3.2 GitHub Actions

Es la herramienta principal para la automatización del flujo CI/CD. Se configuraron workflows que ejecutan pruebas, validan el código y lo despliegan automáticamente. Cada vez que se actualiza el código, GitHub Actions se encarga de ejecutar las tareas necesarias sin intervención humana.

### 3.3 Node.js

Plataforma usada para desarrollar el backend de la aplicación. Permite ejecutar código JavaScript del lado del servidor y manejar las principales funcionalidades como registro, almacenamiento y recuperación de mensajes. Node.js proporciona rapidez y eficiencia en la ejecución.

### 3.4 MySQL Workbench

Entorno visual que se utilizó para diseñar, modelar y gestionar la base de datos relacional del sistema. Se crearon las tablas de usuarios y mensajes, y se establecieron sus relaciones. Además, permite ejecutar consultas SQL para pruebas y visualización de datos.

## 4. Arquitectura del Sistema

La arquitectura de la miniaplicación “Libro de Visitas” se basa en una estructura cliente-servidor, donde el cliente accede a una interfaz web, mientras que el servidor gestiona la lógica de negocio, acceso a base de datos y validaciones.

### Componentes principales:

* **Frontend (Cliente Web):**
  + Desarrollado con HTML, CSS y JavaScript.
  + Se encarga de la interacción con el usuario: formularios de registro y mensajes, vista de mensajes anteriores.
  + Comunicación con el backend mediante peticiones HTTP (fetch/AJAX).
* **Backend (Servidor Node.js):**
  + Procesa las solicitudes del cliente.
  + Realiza validaciones (campos vacíos, longitud de mensaje).
  + Conecta con la base de datos MySQL para crear, consultar y almacenar datos.
  + Expone rutas como /api/usuarios, /api/mensajes, etc.
* **Base de datos (MySQL):**
  + Maneja dos tablas: usuarios y mensajes.
  + Relación uno a muchos entre usuarios y mensajes.
  + Índices por clave primaria y foráneas para integridad referencial.
* **CI/CD con GitHub Actions:**
  + Automatiza pruebas, empaquetado y despliegue desde GitHub a servidores o plataformas cloud (como Render, Vercel, Railway, etc.).

## 5. Estructura de Base de Datos

### Tabla 1: Usuarios

| **Campo** | **Tipo de dato** | **Descripción** |
| --- | --- | --- |
| id\_usuario | INT (PK) | Identificador único del usuario |
| nombre | VARCHAR(100) | Nombre completo del visitante |
| correo | VARCHAR(150) | Correo electrónico del visitante |
| fecha\_registro | DATETIME | Fecha y hora de registro del usuario |

**Nota:** Esta tabla almacena los datos personales de cada visitante que interactúa con la aplicación.

### Tabla 2: Mensajes

| **Campo** | **Tipo de dato** | **Descripción** |
| --- | --- | --- |
| id\_mensaje | INT (PK) | Identificador único del mensaje |
| id\_usuario | INT (FK) | Clave foránea que referencia al usuario emisor |
| contenido | TEXT | Mensaje del visitante (máximo 300 caracteres) |
| fecha\_envio | DATETIME | Fecha y hora en la que se envió el mensaje |

**Nota:** Esta tabla almacena los mensajes breves enviados por cada usuario registrado.

## 6. Flujo del Usuario

El proceso de interacción de un visitante en el “Libro de Visitas” sigue el siguiente flujo:

1. **Acceso inicial:** El visitante ingresa al sitio y visualiza mensajes anteriores.
2. **Registro:** Si es su primera vez, llena un formulario con su nombre y correo.
3. **Ingreso de mensaje:** Puede escribir un mensaje de máximo 300 caracteres.
4. **Envío del mensaje:** El sistema lo valida y lo guarda en la base de datos.
5. **Visualización:** Su mensaje aparece inmediatamente junto a los demás, con su nombre y hora de publicación.

Este flujo es sencillo pero efectivo, asegurando una experiencia rápida y accesible para el usuario.

## 7. Pruebas Automatizadas

Las pruebas automáticas son una parte clave del CI. En este proyecto se incluyen pruebas básicas de:

* **Pruebas de validación:** Aseguran que los mensajes no se guarden si no cumplen con los requisitos.
* **Pruebas de conexión a base de datos:** Verifican que se pueda leer y escribir correctamente.
* **Pruebas de integración:** Simulan el flujo completo: registro de usuario + envío de mensaje.

Estas pruebas se ejecutan automáticamente con cada push o pull request.

## 8. Capturas de Uso de GitHub Actions

## 

Cuando se ejecuta (push/pull\_request a main).

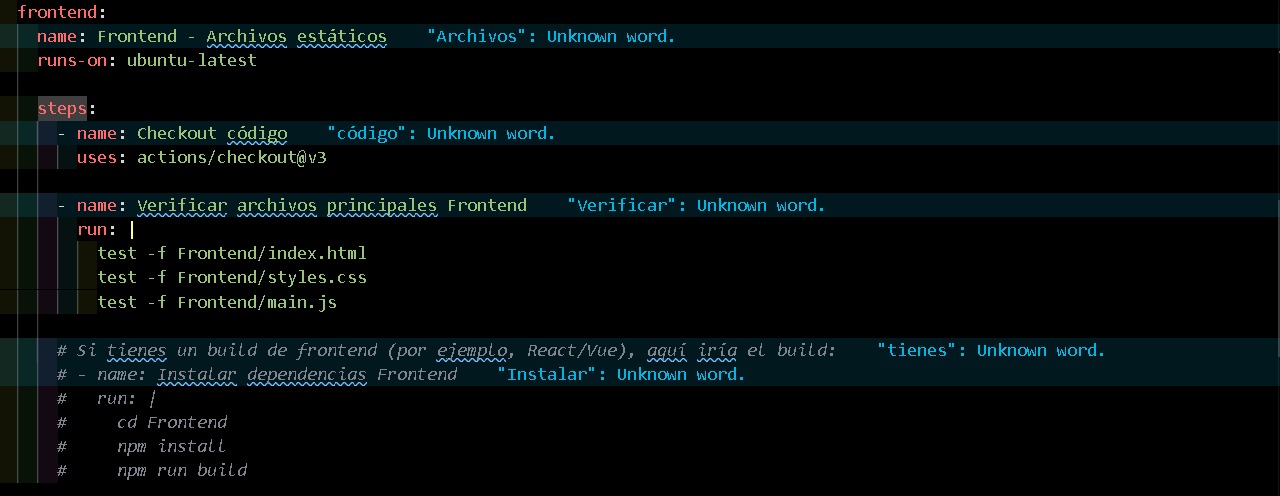
Jobs: Define los trabajos a ejecutar.

backend: Descarga el código.

Instala Node.js.

Instala dependencias del backend.

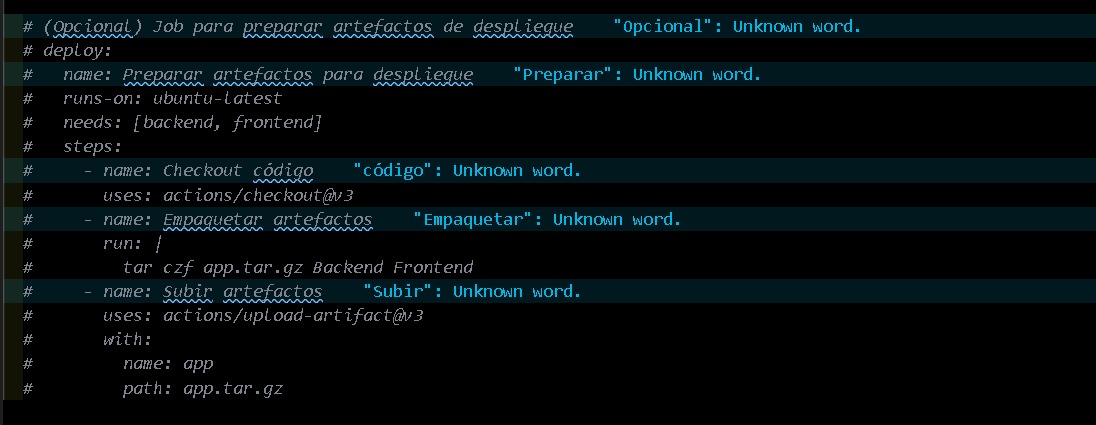
(Opcional) Ejecuta pruebas.Verifica que existan archivos clave.



frontend:Descarga el código.

Verifica que existan los archivos principales del frontend.

(Opcional) Instala dependencias y hace build si usas frameworks modernos.



Deploy: Empaqueta y sube los artefactos para despliegue.

# 8. Conclusión y Proyección Futura

El proyecto “Libro de Visitas” demuestra cómo incluso una aplicación pequeña puede beneficiarse enormemente del uso de CI/CD y buenas prácticas de desarrollo. Se logró:

* Automatizar el ciclo completo de desarrollo.
* Garantizar un sistema funcional, seguro y mantenible.
* Crear una experiencia fluida para el usuario final.