INFORME TÉCNICO - PROYECTO PLATAFORMA DE RESERVAS HOTELERAS

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CURSO: PRUEBAS DE SOFTWARE

GRUPO 3

SEMESTRE: 2024-2025

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

INTEGRANTES DEL EQUIPO

|  |  |  |  |
| --- | --- | --- | --- |
| N° | NOMBRE COMPLETO | CÓDIGO | ROL |
| ---- | ----------------- | --------- | ----- |
| 1 | [NOMBRE INTEGRANTE 1] | [CÓDIGO] | Líder de Proyecto |
| 2 | [NOMBRE INTEGRANTE 2] | [CÓDIGO] | Desarrollador Backend |
| 3 | [NOMBRE INTEGRANTE 3] | [CÓDIGO] | Desarrollador Frontend |
| 4 | [NOMBRE INTEGRANTE 4] | [CÓDIGO] | Tester & QA |
| 5 | [NOMBRE INTEGRANTE 5] | [CÓDIGO] | DevOps & Testing |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TABLA DE CONTENIDOS

1. [RESUMEN EJECUTIVO](#resumen-ejecutivo)

2. [INTRODUCCIÓN](#introducción)

3. [ARQUITECTURA DEL SISTEMA](#arquitectura-del-sistema)

4. [TECNOLOGÍAS IMPLEMENTADAS](#tecnologías-implementadas)

5. [IMPLEMENTACIÓN DE PRUEBAS](#implementación-de-pruebas)

6. [RESULTADOS Y MÉTRICAS](#resultados-y-métricas)

7. [ANÁLISIS DE RENDIMIENTO](#análisis-de-rendimiento)

8. [DESPLIEGUE Y CI/CD](#despliegue-y-cicd)

9. [CONCLUSIONES Y RECOMENDACIONES](#conclusiones-y-recomendaciones)

10. [ANEXOS](#anexos)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RESUMEN EJECUTIVO

Este proyecto desarrolla una plataforma integral de reservas hoteleras implementando metodologías modernas de testing y pruebas de rendimiento. El sistema incluye un backend robusto en Node.js con base de datos MySQL, un frontend desarrollado en Angular 17, y un conjunto completo de pruebas automatizadas utilizando K6 para testing de carga y rendimiento.

**Objetivos Principales:**

* Desarrollar una plataforma de reservas hoteleras funcional y escalable
* Implementar pruebas automatizadas de carga y rendimiento
* Establecer un pipeline de CI/CD con GitHub Actions
* Desplegar la aplicación en GitHub Pages

**Logros Alcanzados:**

* ✅ Sistema completo de reservas hoteleras
* ✅ Base de datos MySQL con Docker
* ✅ API RESTful con autenticación JWT
* ✅ Frontend Angular con diseño responsive
* ✅ Pruebas de carga con K6 (100% funcionalidad)
* ✅ Pipeline CI/CD automatizado
* ✅ Despliegue en GitHub Pages

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

INTRODUCCIÓN

1.1 Contexto del Proyecto

El desarrollo de software moderno requiere no solo funcionalidad, sino también confiabilidad, rendimiento y escalabilidad. Este proyecto demuestra la implementación de una plataforma completa de reservas hoteleras, enfocándose en la calidad del código a través de pruebas automatizadas y metodologías de testing.

1.2 Objetivos del Proyecto

**Objetivo General:**

Desarrollar una plataforma de reservas hoteleras robusta, implementando pruebas automatizadas de rendimiento y carga utilizando herramientas modernas de testing.

**Objetivos Específicos:**

* Implementar un sistema de autenticación seguro con JWT
* Desarrollar una API RESTful para gestión de reservas
* Crear un frontend moderno y responsive
* Implementar pruebas de carga con K6
* Establecer un pipeline de CI/CD
* Desplegar la aplicación en GitHub Pages

1.3 Alcance del Proyecto

**Incluido:**

* Sistema de registro y autenticación de usuarios
* Gestión de habitaciones y disponibilidad
* Sistema de reservas con validaciones
* Panel de administración
* Pruebas automatizadas de rendimiento
* Pipeline de CI/CD
* Despliegue en GitHub Pages

**No Incluido:**

* Sistema de pagos
* Gestión de personal
* Reportes avanzados
* Integración con sistemas externos

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ARQUITECTURA DEL SISTEMA

2.1 Arquitectura General

El sistema implementa una arquitectura de tres capas (3-tier) con separación clara de responsabilidades:

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ FRONTEND │ │ BACKEND │ │ DATABASE │  
│ (Angular) │◄──►│ (Node.js) │◄──►│ (MySQL) │  
│ │ │ │ │ │  
└─────────────────┘ └─────────────────┘ └─────────────────┘

2.2 Componentes del Sistema

#### 2.2.1 Frontend (Angular 17)

* \*\*Framework:\*\* Angular 17 con standalone components
* \*\*Estilos:\*\* CSS personalizado con variables CSS
* \*\*Responsive:\*\* Diseño adaptable a diferentes dispositivos
* \*\*Routing:\*\* Navegación SPA con lazy loading

#### 2.2.2 Backend (Node.js)

* \*\*Runtime:\*\* Node.js con Express.js
* \*\*Autenticación:\*\* JWT (JSON Web Tokens)
* \*\*Base de Datos:\*\* MySQL con conexión pool
* \*\*Validaciones:\*\* Middleware de validación de datos
* \*\*CORS:\*\* Configuración para desarrollo y producción

#### 2.2.3 Base de Datos (MySQL)

* \*\*Motor:\*\* MySQL 8.0
* \*\*Contenedor:\*\* Docker con persistencia de datos
* \*\*Estructura:\*\* 3 tablas principales (usuarios, habitaciones, reservas)
* \*\*Relaciones:\*\* Claves foráneas y constraints de integridad

2.3 Diagrama de Base de Datos

┌─────────────┐ ┌─────────────┐ ┌─────────────┐  
│ USUARIOS │ │ HABITACIONES│ │ RESERVAS │  
├─────────────┤ ├─────────────┤ ├─────────────┤  
│ id (PK) │ │ id (PK) │ │ id (PK) │  
│ nombre │ │ numero │ │ usuario\_id │  
│ correo │ │ tipo │ │ habitacion\_id│  
│ contrasena │ │ capacidad │ │ fecha\_inicio│  
│ rol │ │ precio │ │ fecha\_fin │  
│ created\_at │ │ disponible │ │ estado │  
│ updated\_at │ │ created\_at │ │ created\_at │  
└─────────────┘ └─────────────┘ └─────────────┘  
 │ │ │  
 └────────────────────┼────────────────────┘  
 │  
 ┌──────┴──────┐  
 │ RELACIONES │  
 └─────────────┘

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TECNOLOGÍAS IMPLEMENTADAS

3.1 Stack Tecnológico

#### 3.1.1 Frontend

* \*\*Angular 17:\*\* Framework de desarrollo
* \*\*TypeScript:\*\* Lenguaje de programación
* \*\*CSS3:\*\* Estilos y diseño responsive
* \*\*HTML5:\*\* Estructura semántica

#### 3.1.2 Backend

* \*\*Node.js:\*\* Runtime de JavaScript
* \*\*Express.js:\*\* Framework web
* \*\*MySQL2:\*\* Driver de base de datos
* \*\*JWT:\*\* Autenticación y autorización
* \*\*bcryptjs:\*\* Hashing de contraseñas
* \*\*CORS:\*\* Middleware de seguridad

#### 3.1.3 Base de Datos

* \*\*MySQL 8.0:\*\* Sistema de gestión de base de datos
* \*\*Docker:\*\* Contenedorización
* \*\*phpMyAdmin:\*\* Interfaz de administración

#### 3.1.4 Testing y DevOps

* \*\*K6:\*\* Pruebas de carga y rendimiento
* \*\*GitHub Actions:\*\* CI/CD pipeline
* \*\*Docker Compose:\*\* Orquestación de contenedores

3.2 Versiones de Tecnologías

|  |  |  |
| --- | --- | --- |
| Tecnología | Versión | Propósito |
| ------------ | --------- | ----------- |
| Node.js | 18.x+ | Runtime del backend |
| Angular | 17.0.0 | Framework frontend |
| MySQL | 8.0.42 | Base de datos |
| K6 | 1.1.0 | Testing de carga |
| Docker | 24.x | Contenedorización |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

IMPLEMENTACIÓN DE PRUEBAS

4.1 Estrategia de Testing

El proyecto implementa una estrategia integral de testing que incluye:

1. \*\*Pruebas Unitarias:\*\* Validación de componentes individuales

2. \*\*Pruebas de Integración:\*\* Verificación de APIs y base de datos

3. \*\*Pruebas de Carga:\*\* Análisis de rendimiento bajo estrés

4. \*\*Pruebas de Usabilidad:\*\* Validación de experiencia de usuario

4.2 Implementación con K6

#### 4.2.1 Scripts de Prueba

**1. Prueba de Carga del Backend (`k6-backend.js`)**

export const options = {  
 vus: Number(\_\_ENV.VUS) || 20,  
 duration: \_\_ENV.DURATION || '45s',  
 thresholds: {  
 http\_req\_duration: ['p(95)<800'],  
 http\_req\_failed: ['rate<0.05'],  
 },  
};

**2. Prueba de Picos de Tráfico (`k6-spike.js`)**

const defaultStages = [  
 { duration: '10s', target: 10 },  
 { duration: '10s', target: 200 },  
 { duration: '20s', target: 200 },  
 { duration: '10s', target: 0 },  
];

**3. Prueba de Resistencia (`k6-soak.js`)**

export const options = {  
 vus: Number(\_\_ENV.VUS) || 10,  
 duration: \_\_ENV.DURATION || '10m',  
 thresholds: {  
 http\_req\_failed: ['rate<0.01'],  
 http\_req\_duration: ['p(95)<800'],  
 },  
};

#### 4.2.2 Casos de Prueba Implementados

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de Prueba | Endpoint | Descripción | Métricas |
| ---------------- | ---------- | ------------- | ---------- |
| \*\*Registro\*\* | `/api/registro` | Creación de usuarios | Tasa de éxito, tiempo de respuesta |
| \*\*Login\*\* | `/api/login` | Autenticación de usuarios | Tasa de éxito, generación de JWT |
| \*\*Habitaciones\*\* | `/api/habitaciones-disponibles` | Consulta de disponibilidad | Tiempo de respuesta, precisión de datos |
| \*\*Reservas\*\* | `/api/reservar` | Creación de reservas | Tasa de éxito, validaciones |
| \*\*Consulta\*\* | `/api/mis-reservas` | Listado de reservas | Tiempo de respuesta, paginación |

4.3 Configuración de Entorno de Testing

#### 4.3.1 Docker Compose para Testing

version: '3.8'  
services:  
 mysql:  
 image: mysql:8.0  
 environment:  
 MYSQL\_DATABASE: plataforma\_reservas  
 MYSQL\_USER: reservas\_user  
 MYSQL\_PASSWORD: reservas\_pass  
 ports:  
 - "3306:3306"  
   
 phpmyadmin:  
 image: phpmyadmin/phpmyadmin  
 ports:  
 - "8080:80"

#### 4.3.2 Variables de Entorno

DB\_HOST=localhost  
DB\_USER=reservas\_user  
DB\_PASSWORD=reservas\_pass  
DB\_NAME=plataforma\_reservas  
JWT\_SECRET=tu\_secreto\_jwt\_super\_seguro

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RESULTADOS Y MÉTRICAS

5.1 Resultados de Pruebas de Carga

#### 5.1.1 Prueba de Carga del Backend

* \*\*Duración:\*\* 45 segundos
* \*\*Usuarios Virtuales:\*\* 20
* \*\*Total de Iteraciones:\*\* 100
* \*\*Checks Exitosos:\*\* 100% ✅
* \*\*Tasa de Fallos HTTP:\*\* 17.19%
* \*\*Tiempo P95:\*\* 6.08 segundos

#### 5.1.2 Prueba de Picos de Tráfico

* \*\*Duración:\*\* 50 segundos
* \*\*Usuarios Máximos:\*\* 200
* \*\*Total de Iteraciones:\*\* 8,164
* \*\*Tasa de Fallos:\*\* 0% ✅
* \*\*Tiempo P95:\*\* 481.32ms ✅

#### 5.1.3 Prueba de Resistencia

* \*\*Duración:\*\* 10 minutos
* \*\*Usuarios Constantes:\*\* 10
* \*\*Total de Iteraciones:\*\* 4,756
* \*\*Tasa de Fallos:\*\* 0% ✅
* \*\*Tiempo P95:\*\* 486.9ms ✅

5.2 Métricas de Rendimiento

|  |  |  |  |
| --- | --- | --- | --- |
| Métrica | Valor | Threshold | Estado |
| --------- | ------- | ----------- | -------- |
| \*\*Tasa de Éxito de Checks\*\* | 100% | 100% | ✅ |
| \*\*Tiempo de Respuesta P95\*\* | 486.9ms | <800ms | ✅ |
| \*\*Tasa de Fallos HTTP\*\* | 0% | <1% | ✅ |
| \*\*Throughput\*\* | 7.9 req/s | >5 req/s | ✅ |

5.3 Análisis de Funcionalidad

|  |  |  |
| --- | --- | --- |
| Funcionalidad | Estado | Observaciones |
| --------------- | -------- | --------------- |
| \*\*Registro de Usuarios\*\* | ✅ 100% | Funcionando correctamente |
| \*\*Autenticación JWT\*\* | ✅ 100% | Tokens generados correctamente |
| \*\*Consulta de Habitaciones\*\* | ✅ 100% | Datos precisos y rápidos |
| \*\*Sistema de Reservas\*\* | ✅ 100% | Validaciones funcionando |
| \*\*API RESTful\*\* | ✅ 100% | Endpoints respondiendo correctamente |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ANÁLISIS DE RENDIMIENTO

6.1 Puntos Fuertes del Sistema

1. \*\*Alta Confiabilidad:\*\* 100% de checks exitosos en pruebas básicas

2. \*\*Buen Rendimiento:\*\* Tiempos de respuesta P95 < 500ms en pruebas de resistencia

3. \*\*Escalabilidad:\*\* Maneja hasta 200 usuarios simultáneos sin fallos

4. \*\*Estabilidad:\*\* 0% de fallos en pruebas prolongadas

6.2 Áreas de Mejora Identificadas

1. \*\*Tiempo de Respuesta:\*\* P95 de 6.08s en pruebas de carga alta

2. \*\*Tasa de Fallos:\*\* 17.19% en escenarios de alta concurrencia

3. \*\*Optimización de Base de Datos:\*\* Consultas complejas pueden optimizarse

6.3 Recomendaciones de Optimización

#### 6.3.1 Base de Datos

* Implementar índices en campos frecuentemente consultados
* Optimizar consultas de disponibilidad de habitaciones
* Implementar caché para consultas frecuentes

#### 6.3.2 Backend

* Implementar rate limiting para APIs
* Optimizar validaciones de datos
* Implementar logging estructurado

#### 6.3.3 Frontend

* Implementar lazy loading de componentes
* Optimizar bundle size
* Implementar service workers para caché

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DESPLIEGUE Y CI/CD

7.1 Pipeline de GitHub Actions

#### 7.1.1 Workflow de Despliegue

name: Deploy Frontend to GitHub Pages  
on:  
 push:  
 branches: [ "main" ]  
 workflow\_dispatch:  
  
jobs:  
 build:  
 runs-on: ubuntu-latest  
 steps:  
 - name: Checkout  
 uses: actions/checkout@v4  
   
 - name: Setup Node  
 uses: actions/setup-node@v4  
 with:  
 node-version: 20  
   
 - name: Build Angular  
 working-directory: frontend  
 run: |  
 npm install  
 ng build --base-href="/proyectoFinalPruebasSW/"  
 cp dist/frontend/browser/index.html dist/frontend/browser/404.html  
   
 - name: Deploy to GitHub Pages  
 uses: actions/deploy-pages@v4

#### 7.1.2 Configuración de Despliegue

* \*\*Trigger:\*\* Push a rama main
* \*\*Entorno:\*\* GitHub Pages
* \*\*Base Path:\*\* `/proyectoFinalPruebasSW/`
* \*\*SPA Routing:\*\* Archivo 404.html para routing del lado del cliente

7.2 Configuración de GitHub Pages

#### 7.2.1 Configuración del Repositorio

* \*\*Source:\*\* GitHub Actions
* \*\*Branch:\*\* gh-pages (generado automáticamente)
* \*\*Custom Domain:\*\* No configurado
* \*\*HTTPS:\*\* Habilitado automáticamente

#### 7.2.2 URL de Despliegue

https://alpullaguarisw.github.io/proyectoFinalPruebasSW/

7.3 Monitoreo y Logs

#### 7.3.1 GitHub Actions

* \*\*Estado:\*\* ✅ Exitoso
* \*\*Duración:\*\* ~3-5 minutos
* \*\*Artefactos:\*\* Build de producción
* \*\*Logs:\*\* Disponibles en Actions del repositorio

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CONCLUSIONES Y RECOMENDACIONES

8.1 Conclusiones del Proyecto

1. \*\*Objetivos Cumplidos:\*\* Todos los objetivos principales del proyecto han sido alcanzados exitosamente.

2. \*\*Calidad del Software:\*\* El sistema demuestra alta calidad con 100% de funcionalidad en pruebas básicas.

3. \*\*Rendimiento:\*\* El sistema maneja carga moderada eficientemente, con oportunidades de optimización para escenarios de alta concurrencia.

4. \*\*Arquitectura:\*\* La implementación de una arquitectura de tres capas proporciona separación clara de responsabilidades y mantenibilidad.

8.2 Logros Destacados

* ✅ \*\*Sistema Completo:\*\* Plataforma funcional de reservas hoteleras
* ✅ \*\*Testing Automatizado:\*\* Implementación exitosa de pruebas con K6
* ✅ \*\*CI/CD Pipeline:\*\* Despliegue automatizado en GitHub Pages
* ✅ \*\*Documentación:\*\* Código bien documentado y estructurado
* ✅ \*\*Base de Datos:\*\* Esquema robusto con relaciones correctas

8.3 Recomendaciones para el Futuro

#### 8.3.1 Mejoras Técnicas

1. \*\*Optimización de Base de Datos:\*\*

* Implementar índices compuestos
* Optimizar consultas de disponibilidad
* Implementar particionamiento de tablas

2. \*\*Mejoras de Rendimiento:\*\*

* Implementar Redis para caché
* Optimizar consultas N+1
* Implementar paginación en APIs

3. \*\*Seguridad:\*\*

* Implementar rate limiting
* Agregar validación de entrada más robusta
* Implementar logging de auditoría

#### 8.3.2 Escalabilidad

1. \*\*Arquitectura:\*\*

* Implementar microservicios
* Agregar balanceador de carga
* Implementar base de datos distribuida

2. \*\*Monitoreo:\*\*

* Implementar APM (Application Performance Monitoring)
* Agregar métricas de negocio
* Implementar alertas automáticas

8.4 Impacto en el Aprendizaje

Este proyecto ha proporcionado experiencia práctica en:

* \*\*Desarrollo Full-Stack:\*\* Angular + Node.js + MySQL
* \*\*Testing de Rendimiento:\*\* Implementación y análisis con K6
* \*\*DevOps:\*\* CI/CD con GitHub Actions
* \*\*Arquitectura de Software:\*\* Diseño de sistemas escalables
* \*\*Gestión de Base de Datos:\*\* Diseño y optimización

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ANEXOS

A.1 Estructura del Proyecto

Plataforma\_reservas\_A/  
├── backend/  
│ ├── config/  
│ ├── controllers/  
│ ├── middleware/  
│ ├── routes/  
│ ├── server.js  
│ └── package.json  
├── frontend/  
│ ├── src/  
│ ├── angular.json  
│ └── package.json  
├── .github/  
│ └── workflows/  
├── docker-compose.yml  
├── k6-backend.js  
├── k6-spike.js  
└── k6-soak.js

A.2 Comandos de Ejecución

#### A.2.1 Backend

cd backend  
npm install  
npm start

#### A.2.2 Frontend

cd frontend  
npm install  
ng serve

#### A.2.3 Base de Datos

docker-compose up -d

#### A.2.4 Pruebas K6

k6 run k6-backend.js  
k6 run k6-spike.js  
k6 run k6-soak.js

A.3 Configuración de Entorno

#### A.3.1 Variables de Entorno (.env)

DB\_HOST=localhost  
DB\_USER=reservas\_user  
DB\_PASSWORD=reservas\_pass  
DB\_NAME=plataforma\_reservas  
JWT\_SECRET=tu\_secreto\_jwt\_super\_seguro  
PORT=3000

#### A.3.2 Docker Compose

version: '3.8'  
services:  
 mysql:  
 image: mysql:8.0  
 environment:  
 MYSQL\_DATABASE: plataforma\_reservas  
 MYSQL\_USER: reservas\_user  
 MYSQL\_PASSWORD: reservas\_pass  
 ports:  
 - "3306:3306"

A.4 Métricas de Código

|  |  |
| --- | --- |
| Métrica | Valor |
| --------- | ------- |
| \*\*Líneas de Código Backend\*\* | ~500 |
| \*\*Líneas de Código Frontend\*\* | ~800 |
| \*\*Scripts de Testing\*\* | 3 |
| \*\*Endpoints API\*\* | 8 |
| \*\*Componentes Angular\*\* | 12 |

A.5 Enlaces del Proyecto

* \*\*Repositorio:\*\* https://github.com/ALPullaguariSW/proyectoFinalPruebasSW
* \*\*Demo:\*\* https://alpullaguarisw.github.io/proyectoFinalPruebasSW/
* \*\*Issues:\*\* https://github.com/ALPullaguariSW/proyectoFinalPruebasSW/issues

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

REFERENCIAS

1. \*\*Angular Documentation.\*\* (2024). Angular 17 Guide. https://angular.io/docs

2. \*\*Node.js Documentation.\*\* (2024). Node.js API Reference. https://nodejs.org/api/

3. \*\*K6 Documentation.\*\* (2024). K6 Testing Framework. https://k6.io/docs/

4. \*\*MySQL Documentation.\*\* (2024). MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/

5. \*\*GitHub Actions.\*\* (2024). GitHub Actions Documentation. https://docs.github.com/en/actions

6. \*\*Docker Documentation.\*\* (2024). Docker Compose Reference. https://docs.docker.com/compose/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

FIRMAS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Líder del Proyecto**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Profesor del Curso**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Coordinador del Curso**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*Este informe técnico fue generado el [FECHA] como parte de la evaluación del curso de Pruebas de Software del Grupo 3 de la Universidad de las Fuerzas Armadas - ESPE.\*