A black background with blue letters

Description automatically generated

**Instituto Tecnológico de Costa Rica**

***Sede Central de Cartago***

Bases de Datos I - Grupo 03

Primera Tarea Programada

**Estudiantes:**

Ben Farzamipour Alfaro

Joshua Obando Castro

**Profesor:**

Franco Quirós

**Fecha de entrega:** 8 de setiembre, 2025

Cartago, Costa Rica

Indice

[Indice de Figuras 2](#_Toc208224543)

[Introducción 3](#_Toc208224544)

[Descripción del Ambiente de Desarrollo 3](#_Toc208224545)

[Tecnologías Utilizadas: 3](#_Toc208224546)

[Arquitectura de la Aplicación: 4](#_Toc208224547)

[Arquitectura de Red: 5](#_Toc208224548)

[Tabla Análisis de Resultados: 6](#_Toc208224549)

[Tabla 2: Métricas Cuantitativas del Proyecto 8](#_Toc208224550)

[Análisis de las Métricas: 8](#_Toc208224551)

[Conclusión 9](#_Toc208224552)

[Links de importancia: 9](#_Toc208224553)

# Indice de Figuras

[Figure 1 5](#_Toc208219952)

# Introducción

Este documento es el análisis formal de los resultados obtenidos para el desarrollo del sistema de gestión de empleados, correspondiente a la primera tarea programada del curso. El objetivo principal de este proyecto fue implementar una aplicación web funcional que interactúe con una base de datos MySQL, cumpliendo con requisitos específicos como la creación de tablas, la implementación de procedimientos almacenados para listar e insertar registros, y el desarrollo de una capa lógica en entorno Flask para su funcionamiento.

El presente análisis junto con la bitácora y el repositorio, sirven como evidencia del estado final del producto software, detallando el ambiente técnico donde se implementó la solución, el grado de cumplimiento de cada uno de los requerimientos funcionales según la rúbrica de evaluación y las métricas cuantitativas derivadas del proceso de desarrollo. Este reporte tiene como finalidad demostrar el funcionamiento integral del sistema y las decisiones técnicas tomadas por el equipo de trabajo.

# Descripción del Ambiente de Desarrollo

## Tecnologías Utilizadas:

* **Backend y Capa Lógica:** FLASK conecta el js con bd, incluyendo la librería que conecta MySQL 8.0
* **Frontend y Capa de Presentación:** HTML5 para la estructura, CSS3 para los estilos y JavaScript vanilla para la lógica de presentación y la comunicación con la API. No se utilizaron frameworks frontend adicionales, con el propósito de mantener una arquitectura simple y enfocada en la lógica de base de datos.
* **Base de Datos:** Motor de base de datos MySQL 8.0 La manipulación de datos se realizó únicamente mediante la ejecución de procedimientos almacenados, garantizando así un nivel alto de seguridad y eficiencia.
* **Control de Versiones:** La plataforma GitHub se utilizó para el hosting del repositorio Git, facilitando el trabajo colaborativo y el control de cambios throughout del ciclo de vida del proyecto.

## Arquitectura de la Aplicación:

Se implementó una arquitectura de tres capas clásica, separando claramente las responsabilidades de cada componente para promover la escalabilidad y el mantenimiento del código.

1. **Capa de Presentación (Frontend):** Compuesta por los archivos estáticos que incluye el index.html, styles.css, scripts.js. Que se ejecutan en el navegador del cliente. Esta capa se comunica con la capa lógica mediante llamadas asíncronas (HTTP requests) utilizando la API fetch de JavaScript para consumir los servicios web.
2. **Capa de Lógica de Negocio (Backend):** Implementada con en FLASK contiene los controladores en archivos Python llamados app.py(inicia la app), BD\_model.py(conecta con la bd) que reciben las peticiones HTTP, procesan la lógica de aplicación y coordinan el acceso a los datos. Esta capa es responsable de invocar los procedimientos almacenados en la base de datos.
3. **Capa de Acceso a Datos:** Aunque no está aislada en un proyecto separado, la lógica de acceso a datos se encapsula dentro de los controladores del backend, utilizando mysql.connector para ejecutar directamente los procedimientos almacenados (AgregarEmpleado, MostrarEmpleados) residentes en el servidor.

**A white rectangular sign with black text

AI-generated content may be incorrect.**

Figure 1: Nueva descripción

## Arquitectura de Red:

Para el desarrollo y las pruebas, la aplicación web se alojaba en un **servidor local** sirviendo y la app se ejecutaba desde un servidor web local como La base de datos estaba alojada en una instancia local de MySQL, a la que el backend se conectaba mediante autenticación de MySQL usando una cadena de conexión segura. Con el hamachi se conecta con el IpV4 , en vez de localhost

# Tabla Análisis de Resultados:

|  |  |  |
| --- | --- | --- |
| Elemento Evaluado | Implementacion % | Comentario |
| **Base de Datos y Estructura** | 100% | La base de datos fue creada exitosamente, incluyendo la tabla **empleados** con sus respectivas estructuras. El script de población carga datos de prueba consistentes para su validación. |
| **Procedimientos Almacenados** | 100% | Se implementaron los dos procedimientos solicitados: • MostrarEmpleados: Obtiene la lista completa de empleados. • AgregarEmpleado: Inserta un nuevo empleado validando los parámetros de entrada. Ambos SP’s son funcionales y se ejecutan correctamente. |
| **Capa Lógica de la Aplicación (Backend Flask)** | 100% | La API Web funciona correctamente: • El endpoint GET **/obtener-empleados**, ejecuta el procedimiento para listar empleados y devuelve los datos en formato JSON. • El endpoint POST **/agregar-empleado** recibe los datos, mapea los parámetros y ejecuta el procedimiento para insertar un nuevo empleado. |
| **Funcionamiento de las Páginas Web (Frontend)** | 100% | Las páginas web consumen la API y funcionan como se espera: • **Lista de Empleados (**index.html**)** Muestra dinámicamente los datos obtenidos del backend. • **Insertar Empleado:** Contiene un formulario funcional que envía los datos al backend para su registro en la base de datos. |
| **Documentación** | 100% | **Este análisis de resultados** se presenta de forma formal y profesional, cumpliendo con todos los elementos solicitados. |
| **Uso de GitHub** | 100% | El uso del repositorio de GitHub es correcto. El historial de commits, la estructura de carpetas y el código alojado sirven como evidencia del trabajo realizado y el control de versiones. |

# Tabla 2: Métricas del Proyecto

|  |  |  |  |
| --- | --- | --- | --- |
| Categoría | Métrica | Valor | Fuente / Observación |
| **Base de Datos** | Cantidad de Tablas | 2 | Empleado, Usuario |
|  | Cantidad de SP’s | 2 | AgregarEmpleado, MostrarEmpleados |
|  | Registros de Prueba en Tabla Empleado | 21 |  |
|  |  |  |  |
| **Código Fuente** | Lenguajes de Programación Utilizados | 4 | HTML, CSS, JS, Python |
|  | Archivos de Código Fuente | 6 | Contando MySQL, HTML, JS, CSS y Python. |
|  |  |  |  |
| **Control de Versiones** | Total de Commits en GitHub | 5 | Historial del repositorio. |
|  |  |  |  |
| **Funcionalidad** | Endpoints de API Implementados | 2 | GET /obtener-empleados,  POST/agregar- empleado |
|  | Páginas Web Principales | 1 | index.html (Listar), ventana emergente del Formulario |
|  |  |  |  |
| **Esfuerzo** | Horas Totales de Trabajo (Estimadas) | 20 | Estimación basada en el tiempo dedicado al desarrollo, pruebas y documentación. |

## Análisis de las Métricas:

Las métricas reflejan un proyecto compacto y bien enfocado, donde se priorizó la calidad funcional sobre la cantidad de artefactos. Se hicieron pocos commits porque se trabajó en bloques grandes y bien definidos antes de realizar confirmaciones en el repositorio. Las 20 horas de trabajo estimadas están alineadas con la complejidad de los requerimientos, demostrando una inversión de tiempo adecuada para el desarrollo, las pruebas y la documentación del sistema.

Conclusión

Este proyecto fue mucho más que la entrega de un sistema funcional; se convirtió en un proceso de aprendizaje profundamente enriquecedor que nos permitió comprender la importancia de la planificación meticulosa y la organización del trabajo. Aprendimos que dividir un gran objetivo en tareas más pequeñas y manejables es la clave para avanzar de manera constante y sin abrumarnos, reforzando el valor de la persistencia y la atención al detalle en cada etapa del desarrollo.

Además, la experiencia nos dejó una lección fundamental sobre la capacidad de resolver problemas por nosotros mismos. Investigar, probar diferentes soluciones y superar los obstáculos que surgieron no solo fortaleció nuestra confianza, sino que también nos enseñó a aprender de los errores y a verlos como oportunidades de crecimiento. Este proyecto no solo cumplió con su objetivo académico, sino que nos equipó con una mayor madurez y preparación para enfrentar nuevos desafíos en el futuro.

# Links de importancia:

Link del repositorio de GitHub: [xHellish/Tarea1-BD1: Tarea Programada 1 Bases de Datos - Proyecto Web / SQL](https://github.com/xHellish/Tarea1-BD1/tree/master)  
  
Link de la bitácora Blogger: [[Bitácora Tarea#1 - Bases de Datos - Semestre II 2025](https://tarea1bdg3.blogspot.com/)](https://tarea1bdg3.blogspot.com/2025/08/avance-2-integracion-de-mysql.html)