**Instituto Tecnológico de Costa Rica** **Curso:** Base de Datos 1  
 **II Semestre 2025** **Título:** Análisis de Resultados: 1 Tarea Programada  
 **Autores:**

Gabriel Barrantes Monge

Manfred Cordero Zuñiga  
 **Profesor:** Franco Quirós Ramírez

# 

# **Contenidos**

[**Capítulo 1: Introducción 3**](#_vsip1gtxcc36)

[**Capítulo 2: Contexto del desarrollo 3**](#_wuinl9jzg4hl)

[Arquitectura de Red 3](#_ek9efsfxiuf9)

[Resumen 4](#_dnixqbcl1rym)

[**Capítulo 3: Análisis de resultados 4**](#_lfharqkrpbp3)

[**Tabla 1. Análisis de los resultados 5**](#_aarjt5dfzizl)

[**Tabla 2. Métricas del proyecto 6**](#_6d112j864bb5)

[**Capítulo 5: Gráficos 8**](#_i17o4afbgka0)

[**Capítulo 6: Conclusión 8**](#_142914lp6bb7)

[**Referencias 9**](#_8fe2ucka9jg1)

# 

# **Capítulo 1: Introducción**

Este documento presenta el desarrollo de un sistema web para la gestión de empleados, elaborado como proyecto final del curso Base de Datos 1. El sistema tiene como propósito principal permitir el registro de empleados junto con su salario y la consulta de la lista de empleados existentes, apoyándose en procedimientos almacenados implementados en SQL Server.

Para la construcción del sistema se trabajó con una arquitectura sencilla: el frontend se desarrolló en HTML, CSS y JavaScript, ofreciendo menús para insertar y listar empleados, mientras que el backend se implementó en Node.js con Express, encargado de comunicarse con la base de datos mediante el paquete mssql. De esta forma, el backend expone servicios que son consumidos por la interfaz web.

El trabajo se realizó en equipo de dos integrantes, aplicando colaboración constante y distribuyendo responsabilidades. Uno de los miembros se enfocó en la parte de la interfaz gráfica, mientras que el otro trabajó principalmente en la configuración del servidor y la conexión a la base de datos, participando ambos en la creación de los procedimientos almacenados. Como apoyo al desarrollo se utilizaron herramientas como GitHub, para el control de versiones del código, y un blog de bitácora, donde se documentaron los avances y las dificultades encontradas durante el proceso.

Uno de los principales retos fue la conexión remota a la base de datos entre ambos integrantes, resuelto mediante el uso de Hamachi, lo que permitió simular una red local para acceder a SQL Server desde diferentes equipos. Otro desafío estuvo en la correcta integración del frontend con el backend, asegurando la comunicación a través de peticiones HTTP y validando los datos antes de enviarlos a la base de datos.

En conclusión, este proyecto representa la aplicación práctica de los conocimientos adquiridos en el curso, integrando conceptos de bases de datos, programación web y trabajo colaborativo en un sistema sencillo pero funcional.

# **Capítulo 2: Contexto del desarrollo**

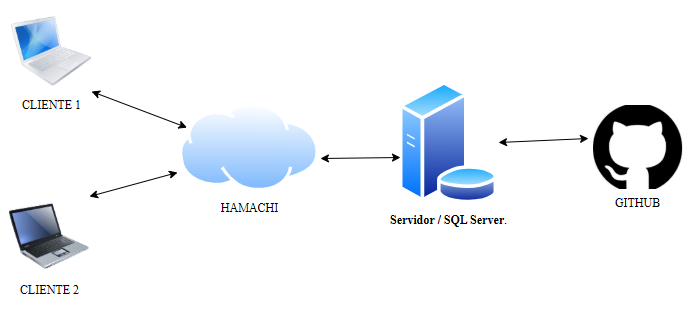
En este capítulo se describe el ambiente de desarrollo del proyecto, las tecnologías utilizadas y la manera en que se integraron los componentes web con la base de datos dentro de un entorno colaborativo.

### **Arquitectura de Red**

La aplicación se desarrolló bajo un esquema cliente-servidor. La base de datos residió en un servidor Microsoft SQL Server accesible de manera remota gracias a la configuración de Hamachi, lo que permitió a los integrantes del equipo conectarse a la misma instancia desde diferentes equipos. De esta forma se garantizó que ambos pudieran realizar pruebas sobre un único entorno de trabajo compartido.

Para el control de versiones se empleó un repositorio en GitHub, donde se almacenaron los archivos del proyecto, incluyendo la estructura del backend en Node.js, la interfaz web y los scripts SQL correspondientes a la base de datos y a los procedimientos almacenados. Esto permitió llevar un registro ordenado de los cambios, integrar los aportes de cada miembro y mantener la trazabilidad del desarrollo.

**Figura 1 - Arquitectura de red**



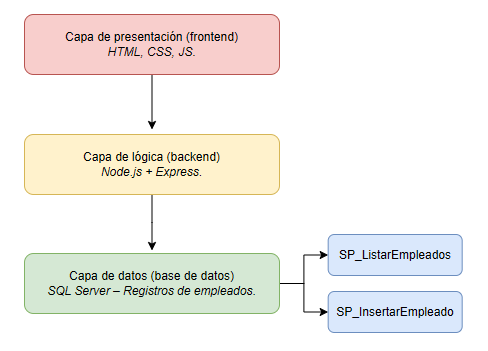
**Arquitectura de la Aplicación**

La solución sigue una arquitectura por capas, diferenciando tres niveles principales:

* Capa de presentación (frontend): construida con HTML, CSS y JavaScript, incluye la interfaz gráfica con menús sencillos para insertar empleados y listar empleados.
* Capa de lógica (backend): implementada en Node.js con Express, se encarga de recibir las solicitudes del frontend y comunicarse con la base de datos. El backend expone rutas API como GET /empleados y POST /empleados, que a su vez invocan a los procedimientos almacenados en SQL Server.
* Capa de datos (base de datos): en SQL Server, donde se almacenan los registros de empleados. La lógica de negocio principal se concentra en dos procedimientos almacenados:  
  + SP\_ListarEmpleados: devuelve todos los empleados registrados con su salario.
  + SP\_InsertarEmpleado: inserta un nuevo empleado, validando los datos recibidos.

La comunicación entre el backend y la base de datos se realizó mediante el paquete mssql de Node.js, que permitió ejecutar los SPs y devolver las respuestas en formato JSON para que el frontend pudiera procesarlas y mostrarlas al usuario.

**Figura 2 - Arquitectura de aplicación**



### **Resumen**

El contexto de desarrollo se caracterizó por el uso de SQL Server, Node.js, Express, HTML, CSS y JavaScript, complementado con herramientas de colaboración como GitHub y Hamachi. Esta integración permitió construir un sistema web simple pero funcional, aplicando en conjunto los conceptos de programación web, bases de datos y trabajo colaborativo.

# **Capítulo 3: Análisis de resultados**

En este capítulo se analizan los resultados obtenidos, verificando el cumplimiento de los requerimientos de la tarea programada. La siguiente tabla resume, por cada ítem evaluable, el estado de implementación, indicando el porcentaje de cumplimiento y un comentario aclaratorio en caso de que algún punto no se haya completado al 100%

# **Tabla 1. Análisis de los resultados**

| **Ítem** | **Resultado (estado y comentario)** |
| --- | --- |
| **Documentación** | 100% – Se entregó la documentación del proyecto en el formato solicitado, incluyendo bitácora y este análisis de resultados. |
| **Base de datos (diseño físico)** | 100% – Se implementó la base de datos en SQL Server con la tabla Empleado, cumpliendo con los requerimientos de la tarea. |
| **Procedimientos almacenados** | 100% – Se desarrollaron los SPs necesarios: SP\_ListarEmpleados y SP\_InsertarEmpleado. El primero devuelve todos los registros, y el segundo valida e inserta un nuevo empleado. |
| **Listado de empleados** | 100% – Se muestra la lista completa de empleados de manera correcta desde la base de datos, cumpliendo con lo solicitado. |
| **Inserción de empleados** | 100% – Funcionalidad operativa. El sistema permite ingresar nuevos empleados desde la interfaz, validando los datos y almacenándolos en la base de datos |
| **Interfaz web (frontend)** | 100% – La aplicación cuenta con un menú sencillo en HTML, CSS y JavaScript, que permite acceder a las funciones de inserción y listado de empleados. |
| **Backend (Node.js + Express)** | 100% – Se implementó la lógica en Node.js para conectar el frontend con la base de datos a través de rutas API (GET /empleados y POST /empleados). La comunicación se realiza correctamente mediante el paquete mssql. |
| **Colaboración y control de versiones** | 100% – Se utilizó un repositorio en GitHub para almacenar el código, permitiendo un control ordenado de los cambios durante el desarrollo. |

Tabla 1 – Análisis de los resultados. La tabla muestra el cumplimiento de los requerimientos planteados. Como se observa, todos los ítems alcanzaron un 100% de implementación, garantizando que el sistema funciona de acuerdo con lo solicitado.

En términos generales, el proyecto cubre de manera exitosa las funcionalidades esenciales definidas en la especificación. Esto incluye la gestión de empleados mediante procedimientos almacenados, la inserción de nuevos registros desde la interfaz web y la visualización de los datos existentes en la base de datos.

El sistema integra correctamente frontend, backend y base de datos en un entorno colaborativo y sencillo, apoyado en GitHub para el control de versiones. El resultado final es una aplicación funcional, práctica y bien estructurada, que demuestra un alto grado de cumplimiento con los objetivos planteados.

**Capítulo 4: Métricas del proyecto**

Las métricas del proyecto permiten cuantificar el esfuerzo de desarrollo y evaluar los resultados alcanzados en términos de dedicación, producción de código y pruebas realizadas. A continuación, se presenta una tabla con los principales indicadores del proyecto, acompañados de comentarios que dan contexto a cada valor.

# **Tabla 2. Métricas del proyecto**

| **Métrica** | **Valor** | **Comentarios** |
| --- | --- | --- |
| **Horas de trabajo - Gabriel** | 15 horas | Tiempo estimado invertido en investigación, creación de la base de datos, desarrollo del backend y documentación. |
| **Horas de trabajo - Manfred** | 15 horas | Tiempo estimado dedicado al desarrollo del frontend, pruebas y coordinación conjunta.. |
| **Horas totales de trabajo** | 30 horas | Esfuerzo global invertido por el equipo en el desarrollo completo del proyecto. |
| **Sesiones de trabajo** | 6 sesiones | Corresponden a las fechas registradas en la bitácora (23/08, 30/08, 02/09, 05/09, 06/09 y 07/09). Cada sesión incluyó reuniones virtuales y trabajo colaborativo. |
| **Entradas en el blog** | 6 entradas | Reflejan los avances de cada jornada de trabajo, documentando actividades, problemas encontrados y soluciones. |
| **Commits en GitHub** | **4**  **commits** | Cantidad aproximada de cambios registrados en el repositorio, evidenciando el uso de control de versiones para integrar el código y la documentación |
| **Líneas de código (total)** | ~400 líneas | Corresponden a la suma de HTML, CSS, JavaScript y SQL del proyecto. |
| **Procedimientos Almacenados** | 2 SPs | Se desarrollaron SP\_ListarEmpleados y SP\_InsertarEmpleado, que encapsulan la lógica de negocio requerida. |
| **Tablas en base de datos** | 1 tablas | Se creó la tabla Empleado, que concentra la información principal del sistema. |
| **Datos de prueba insertados** | 49 registros | Total de empleados registrados en la base de datos durante las pruebas del sistema. |
| **Duración de las pruebas** | 3 horas | Tiempo estimado dedicado a verificar la conexión mediante Hamachi, las inserciones en la base de datos y el correcto funcionamiento del frontend. |
| **Errores detectados (bugs)** | **3 errores** | Incluyen dificultades al conectarse a la base de datos mediante Hamachi, fallos en el diseño de la interfaz y ajustes menores de funcionalidad. |

**Tabla 2 - Métricas del proyecto.** Las métricas muestran un esfuerzo total de **30 horas de trabajo**, con una participación equilibrada de ambos integrantes. Se evidencia una gestión de proyecto organizada, con **6 entradas en la bitácora** que registran el proceso y un total aproximado de **20 commits en GitHub**, lo que refleja el uso constante de herramientas de control de versiones y documentación.

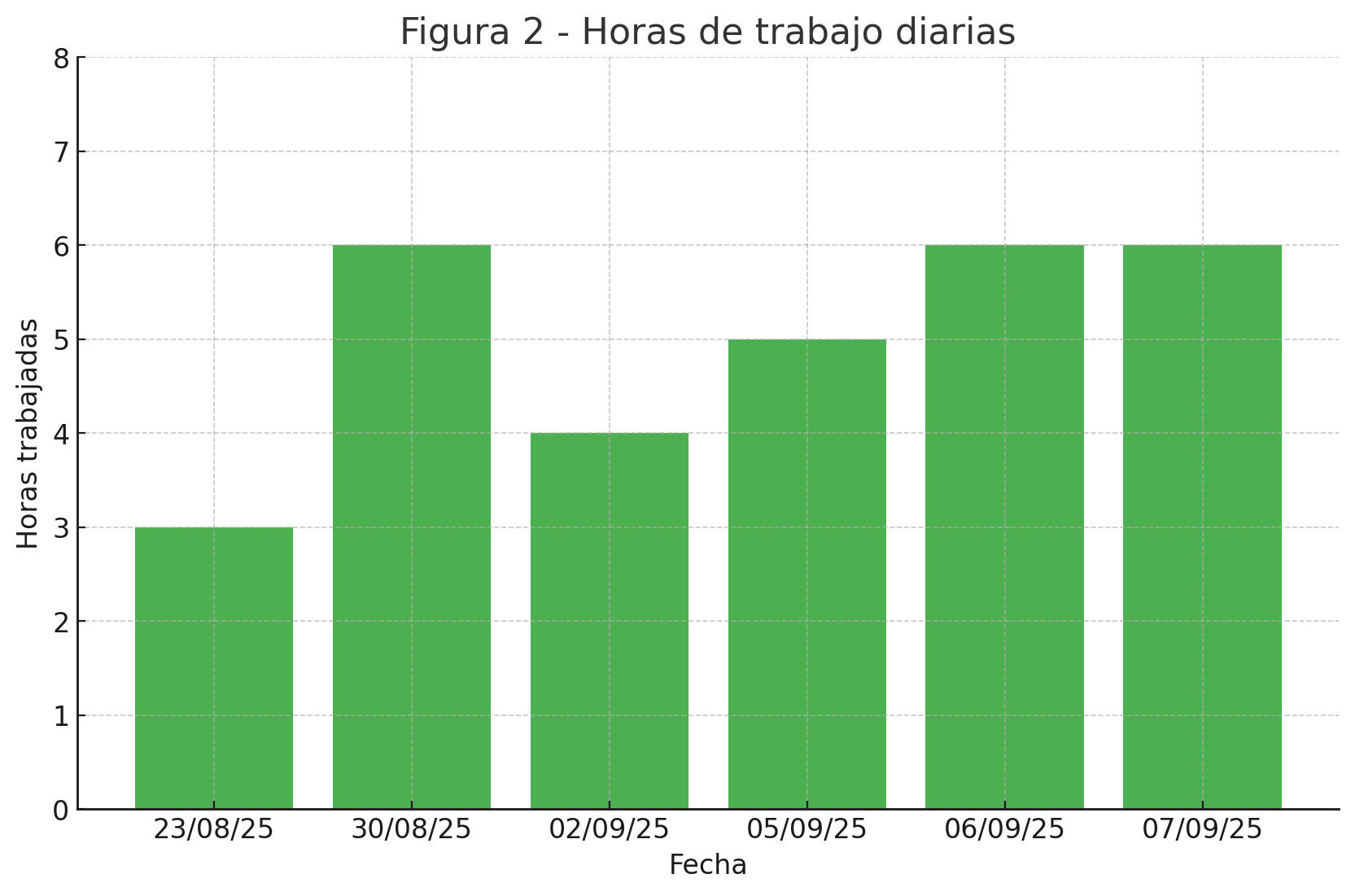
La escala del proyecto es **compacta pero funcional**, reflejada en alrededor de **400 líneas de código**, **2 procedimientos almacenados** y **1 tabla principal en la base de datos**. El sistema fue probado con **49 empleados registrados**, validando el correcto funcionamiento de la lógica implementada.

El proceso de pruebas, que abarcó cerca de **3 horas**, permitió identificar y corregir **3 errores relevantes**, principalmente relacionados con la conexión a la base de datos mediante Hamachi y con la interfaz gráfica. Esto garantizó que la versión final del sistema fuera estable y operativa.

En conjunto, estas métricas respaldan la consecución de un proyecto **sencillo pero bien estructurado**, que cumple de manera completa con los requerimientos planteados y demuestra un buen nivel de organización y colaboración entre los integrantes.

**Capítulo 5: Gráficos**

**Figura 3– Horas de trabajo diarias**

****

**Descripción:**

Este gráfico muestra la distribución del esfuerzo diario del equipo durante el desarrollo del proyecto. Se observa un **pico el 30/08**, relacionado con la fase inicial de conexión y creación de la base de datos, y jornadas largas al final (06 y 07 de septiembre) dedicadas a depuración de errores, desarrollo del frontend y documentación. La correlación entre horas de trabajo y commits refleja un esfuerzo planificado y sostenido, que permitió completar el proyecto de forma organizada.

# **Capítulo 6: Conclusión**

El Proyecto Final de Base de Datos 1 se completó de manera exitosa, logrando desarrollar un sistema web funcional para la gestión de empleados y sus salarios. A lo largo del proyecto, el equipo enfrentó diversos retos técnicos y de coordinación, los cuales se superaron satisfactoriamente, consolidando tanto el aprendizaje como la experiencia práctica en el desarrollo de aplicaciones web conectadas a bases de datos.

En primer lugar, se comprobó la importancia de centralizar la lógica de acceso y manipulación de datos en procedimientos almacenados dentro de SQL Server. Esto permitió mantener la integridad de la información y simplificar las operaciones críticas, como la inserción de nuevos empleados y la consulta de la lista de empleados. Esta práctica reforzó buenas prácticas de desarrollo, como el manejo de errores y la organización clara de las consultas a la base de datos.

En segundo lugar, la experiencia destacó la relevancia de mantener una arquitectura modular y separada entre frontend y backend. La estructura del proyecto permitió desarrollar en paralelo la interfaz de usuario en HTML/JS y la lógica de conexión y rutas en Node.js/Express. Esta separación facilitó la detección y corrección de errores, y permitió que el equipo trabajara de manera simultánea sin conflictos, manteniendo el proyecto organizado y eficiente.

Otro aspecto clave fue la colaboración y coordinación del equipo. El trabajo conjunto entre Gabriel y Manfred fue efectivo, con una distribución equilibrada de tareas y apoyo mutuo en la resolución de problemas. La utilización de GitHub para el control de versiones y la bitácora de avances resultó esencial, registrando de manera detallada la evolución del código y asegurando que ambos miembros mantuvieran el proyecto sincronizado.

Durante el desarrollo, se adquirieron aprendizajes técnicos significativos, tales como la conexión remota segura a la base de datos mediante Hamachi, la integración de Node.js con SQL Server y la creación de una interfaz sencilla pero funcional para gestionar empleados. Cada reto superado contribuyó a mejorar la comprensión del ciclo completo de desarrollo de un sistema web basado en bases de datos.

En conclusión, el proyecto final cumplió con los objetivos planteados, entregando un sistema robusto, seguro y fácil de usar. La experiencia demostró la importancia de planificar, trabajar colaborativamente y seguir buenas prácticas de desarrollo. Además, subrayó cómo un enfoque centrado en la base de datos puede simplificar la construcción de aplicaciones confiables y mantenibles, incluso en proyectos con una lógica sencilla. Los conocimientos y habilidades adquiridos serán valiosos para futuros proyectos similares.

# **Referencias**

· **Bitácora del proyecto:** [BITÁCORA](https://tp1basesdedatos.blogspot.com/)

· **Repositorio GitHub (historial de commits):** [REPOSITORIO](https://github.com/gabo17608/TAREA_PROGRAMADA_1)