Ingenieria Web
Facultad de ingeniería

Elaborado Por:
Hayder Duvan Carreño Ramos
Juan Esteban Bustos Rojas
Miguel Eduardo Angulo Cantor
Juan Felipe Rojas Caceres

Diana Marcela Tojica Rodrigues
Bogotá D.C. Colombia
24 de Febrero del 2026

Contexto del Proyecto y Planteamiento del Problema

En el club deportivo, donde se ofrecen servicios en diferentes canchas de tenis, los empleados deben cumplir turnos en distintos campos según la demanda de los usuarios.

Actualmente, el registro de horas, pagos y asignación de personal se realiza de forma manual, lo cual genera desorden, retrasos en los pagos y confusión en la distribución de empleados.

Además, no existe un sistema que permita visualizar en tiempo real qué canchas están ocupadas, cuáles están disponibles y cuántos empleados se necesitan en cada una.

La asignación del personal se hace mediante sorteos y listas manuales, lo que dificulta su actualización cuando ingresan nuevos turnos o cuando un empleado finaliza su actividad.

Por otra parte, en el club de tenis ubicado en la ciudad, diariamente se presentan múltiples clientes, trabajadores (caddies) y administradores, pero no existe un sistema organizado para gestionar las reservas de canchas ni la asignación del personal.

Los clientes deben preguntar manualmente qué canchas están disponibles, lo que genera confusiones sobre horarios, ocupación y disponibilidad. Esto ocasiona retrasos, discusiones y pérdida de tiempo tanto para los clientes como para los administradores.

Asimismo, los caddies no tienen claridad sobre en qué cancha deben trabajar, por lo que deben consultar constantemente a los jefes, generando desorganización en los turnos. Al momento de realizar los pagos, el proceso también es lento, debido a que no existe un registro claro de las horas trabajadas ni de los servicios prestados.

Ante esta situación, se plantea el desarrollo de una plataforma web que permita administrar el funcionamiento del club de tenis, facilitando:

- La visualización de canchas disponibles y ocupadas en tiempo real.

- El registro de clientes y asignación de canchas según disponibilidad.

- La asignación de caddies a las canchas correspondientes.

- El control de turnos y horas trabajadas.

- El cálculo y pago de los trabajadores de forma rápida y organizada.

Adicionalmente, la estructura utilizada en el desarrollo del proyecto tiene como propósito organizar la creación de la página web, permitiendo:

- Dividir las tareas entre los integrantes del grupo.

- Mantener los archivos ordenados.

- Facilitar la comprensión del proyecto.

- Evitar errores en el desarrollo.

- Mejorar el trabajo en equipo.

Objetivo General

Desarrollar una plataforma web que permita administrar de manera eficiente los turnos, pagos y la disponibilidad de canchas, optimizando el trabajo del personal y mejorando la experiencia de los usuarios.

📌 Objetivos Específicos

- Registrar las horas trabajadas por los empleados.

- Automatizar el cálculo de pagos.

- Visualizar el estado de las canchas en tiempo real.

- Asignar empleados según la demanda.

- Generar reportes administrativos.

- Reducir errores en la gestión manual.

📌 Alcance del Proyecto

El sistema permitirá:

- Registrar empleados y clientes.

- Gestionar turnos y horarios.

- Controlar la asignación de canchas.

- Calcular y generar pagos.

- Mostrar información actualizada.

- Generar reportes.

No incluirá sistemas externos de pago en línea en esta versión.

Requerimientos del Sistema de Gestión de Turnos y Pagos

-Actores

- Empleado

- Administrador

- Sistema

—

Actor	**Requerimiento Funcional**	**Requerimiento No Funcional**
Empleado	Registrar inicio de turno	El registro debe almacenarse en la base de datos en menos de 2 segundos
Empleado	Registrar fin de turno	El cálculo de horas debe ejecutarse automáticamente con precisión del 99.9%

Empleado	Consultar horas trabajadas	La información debe mostrarse en menos de 2 segundos por solicitud
Empleado	Consultar pagos	Solo podrá visualizar sus propios pagos (control de acceso por rol)
Empleado	Visualizar asignación de cancha	Los datos deben actualizarse automáticamente tras cada modificación
Empleado	Visualizar lista de turnos	La información debe provenir de la base de datos centralizada
Administrador	Registrar empleados	Los datos deben validarse en frontend y backend antes de almacenarse
Administrador	Editar o eliminar empleados	Solo usuarios con rol administrador pueden realizar esta acción
Administrador	Registrar clientes	Los datos deben almacenarse de forma persistente en MongoDB
Administrador	Asignar empleados a canchas	Los cambios deben reflejarse inmediatamente en la base de datos
Administrador	Visualizar estado de canchas	El sistema debe consultar la información en tiempo real desde el servidor
Administrador	Aprobar turnos	La actualización del estado debe realizarse mediante API REST segura

Administrador	Autorizar pagos	Solo administradores autenticados pueden cambiar el estado a “pagado”
Administrador	Generar reportes	Los reportes deben generarse en menos de 3 segundos
Administrador	Configurar horarios	Las configuraciones deben almacenarse de forma persistente
Sistema	Autenticar usuarios	Las contraseñas deben almacenarse con hash seguro (bcrypt)
Sistema	Controlar acceso por roles	Implementar autorización basada en roles (RBAC)
Sistema	Calcular pagos automáticamente	El cálculo debe ejecutarse en el servidor (backend)
Sistema	Enviar notificaciones	Las notificaciones deben generarse tras eventos importantes (registro, aprobación, pago)
Sistema	Almacenar información	La base de datos debe contar con respaldo automático diario
Sistema	Proteger la información	La comunicación cliente-servidor debe realizarse mediante HTTPS
Sistema	Disponibilidad del servicio	El sistema debe garantizar una disponibilidad mínima del 99%

Sistema	Soportar múltiples usuarios	Debe soportar al menos 50 usuarios concurrentes sin degradación significativa

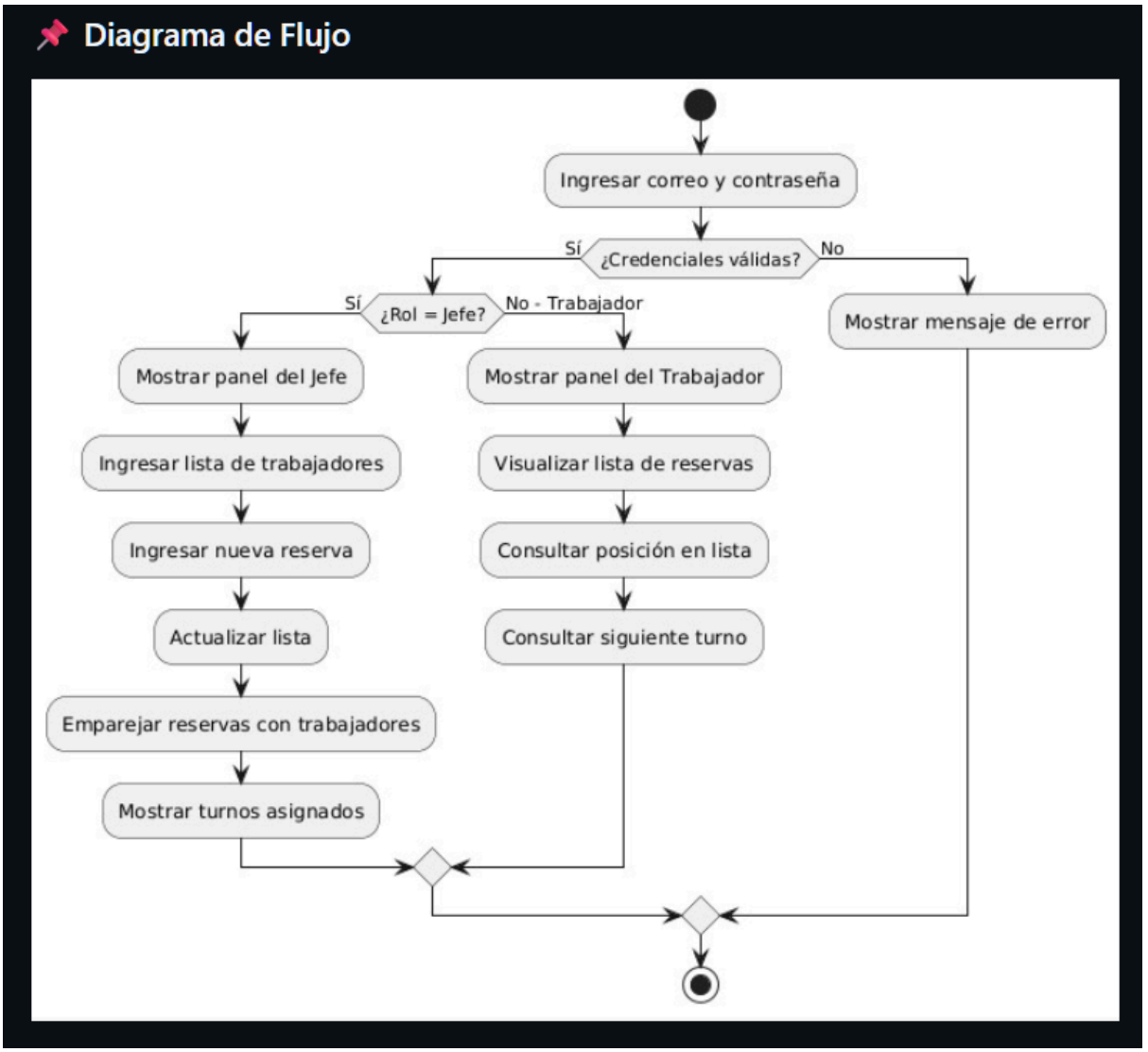


Diagrama de Casos de Uso

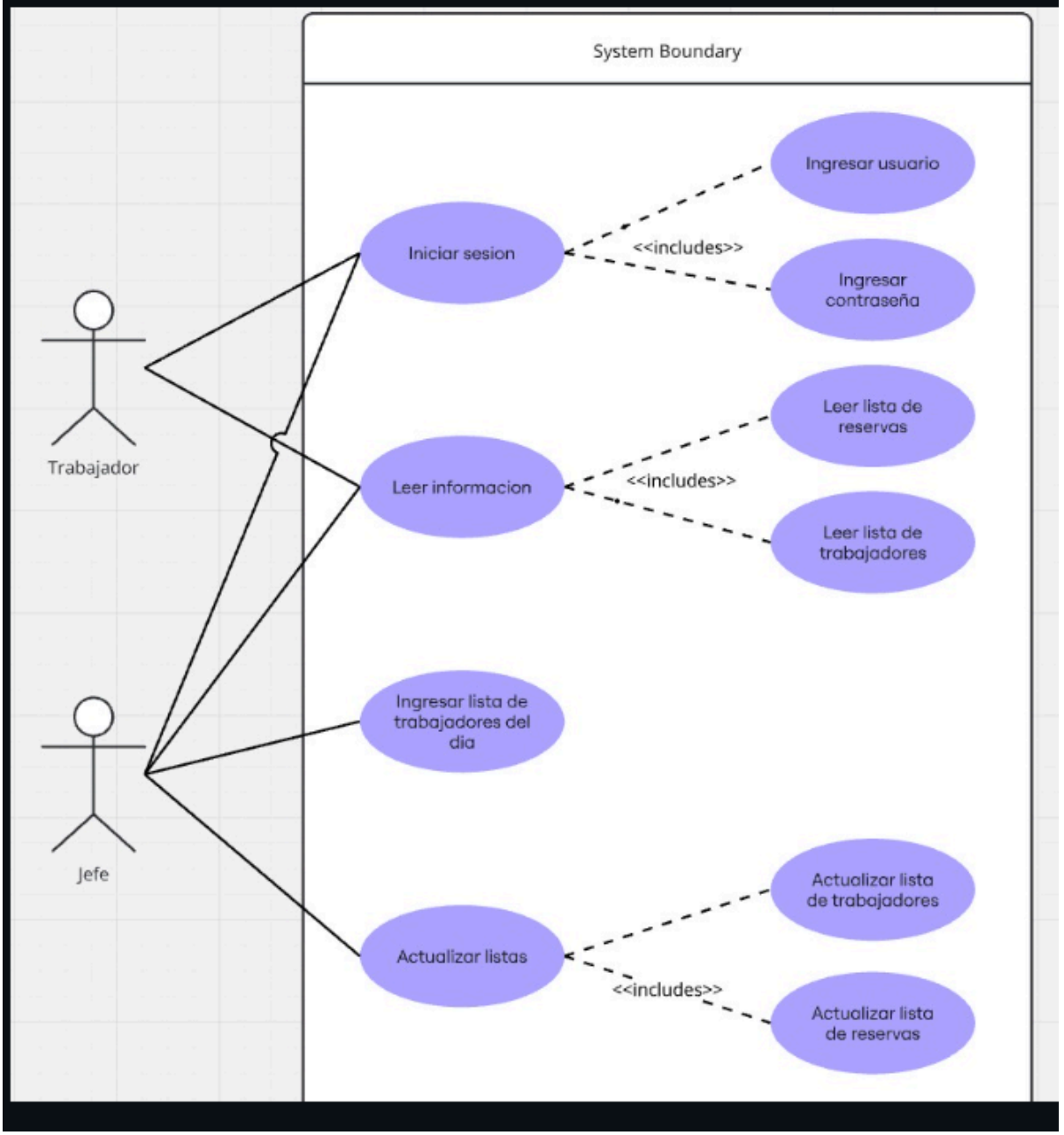
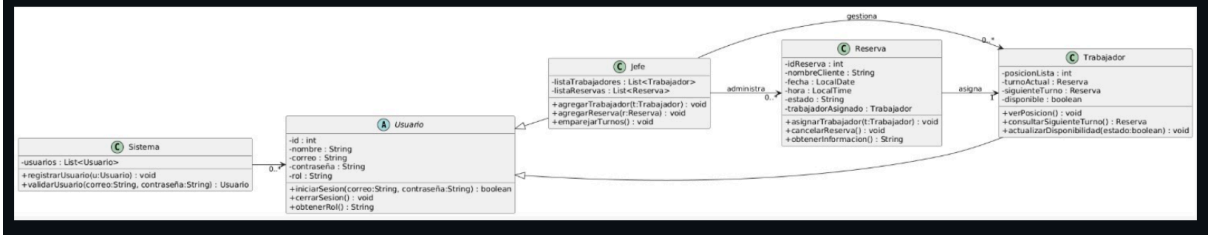


Diagrama de Clases



🛠️ Tecnologías Utilizadas

- HTML5

- CSS

- JavaScript

- GitHub

- Visual Studio Code

📂 Estructura del Proyecto

/gestion-turnos

│

├── /imagenes

│ ├── flujo.png

│ ├── casos_uso.png

│ └── canchas.png

│

├── index.html

├── styles.css

├── script.js

└── README.md

📌 Metodología de Desarrollo

El proyecto se desarrolla mediante una metodología incremental, permitiendo construir el sistema por partes, realizar pruebas constantes y aplicar mejoras continuas.

📌 Conclusión

La implementación de este sistema permitirá optimizar la asignación del personal, mejorar el control de pagos y ofrecer una gestión moderna y eficiente del club deportivo. Gracias a esta organización, el proyecto se puede desarrollar de manera más eficiente, clara y funcional, ofreciendo una solución tecnológica que optimice el tiempo, reduzca errores y mejore la experiencia tanto de los clientes como del personal del club.

IMPLEMENTACIÓN DE LA ARQUITECTURA QUE SE USARÁ

Arquitectura del Sistema

Tipo de Arquitectura Implementada

Para el desarrollo del Sistema de Gestión de Turnos y Pagos del Club de Tenis, se implementará una **arquitectura Cliente-Servidor de Tres Capas**, utilizando el patrón de diseño **MVC (Modelo – Vista – Controlador)**.

Esta arquitectura permite separar responsabilidades, mejorar la organización del código y facilitar el mantenimiento, escalabilidad y trabajo en equipo.

Para el desarrollo del Sistema de Gestión de Turnos y Pagos del Club de Tenis, se adopta una **Arquitectura Cliente-Servidor en Tres Capas**, debido a las necesidades funcionales, de escalabilidad y de trabajo colaborativo que presenta el proyecto.

Esta arquitectura es la más adecuada porque permite separar claramente la interfaz de usuario, la lógica del negocio y la gestión de datos, garantizando organización, mantenimiento y crecimiento futuro del sistema.

Arquitectura en Tres Capas

El sistema estará dividido en las siguientes capas:

FRONTEND Y BACKEND

Capa de Presentación (Frontend)

Corresponde a la interfaz gráfica con la que interactuaran los usuarios (Administrador y Empleado).
El club necesita visualizar en tiempo real el estado de las canchas y los turnos. Separar la interfaz del procesamiento interno permite mejorar la experiencia del usuario sin afectar la lógica del sistema.

Además, al trabajar en equipo mediante GitHub, separar el frontend permite que algunos integrantes trabajen en diseño e interfaz mientras otros trabajan en la lógica interna.

Está desarrollada con:

- HTML5
- CSS
- JavaScript

Funciones principales:

- Formularios de inicio de sesión y registro.
- Visualización en tiempo real del estado de las canchas.
- Panel administrativo.
- Registro de turnos.
- Visualización de pagos.
- Generación de reportes.

Esta capa se ejecuta en el navegador del cliente y se comunica con el servidor mediante peticiones HTTP.

Capa de Lógica de Negocio (Backend)

Se encarga de procesar la información enviada desde el frontend y aplicar las reglas del sistema.

Tecnologías utilizadas:

- Node.js
- Express.js

Funciones principales:

- Autenticación y validación de usuarios.
- Registro y actualización de empleados y clientes.
- Asignación de canchas.
- Control de turnos.
- Cálculo automático de pagos.
- Generación de reportes administrativos.
- Validación de datos antes de almacenarlos.

El backend actúa como intermediario entre el frontend y la base de datos, garantizando seguridad y organización en el manejo de la información.

Capa de Datos (Base de Datos)

Se utilizará una base de datos para almacenar la información de forma persistente.

Base de datos propuesta:

- FireBase

Datos almacenados:

- Usuarios (Administrador y Empleados).
- Clientes.
- Canchas.
- Turnos.
- Pagos.
- Reportes históricos.

Esta capa permite que la información esté disponible desde cualquier dispositivo y no dependa del almacenamiento local del navegador.

Patrón de Diseño: MVC (Modelo – Vista – Controlador)

El sistema implementará el patrón MVC para organizar mejor el código:

- **Modelo (Model):** Representa la estructura de los datos (usuarios, canchas, turnos, pagos).

- **Vista (View):** Corresponde a la interfaz gráfica desarrollada en HTML, CSS y JavaScript.

- **Controlador (Controller):** Gestiona las peticiones del usuario, procesa la lógica del negocio y comunica la vista con el modelo.

Este patrón permite:

- Separar responsabilidades.
- Facilitar el mantenimiento.
- Mejorar la escalabilidad.
- Optimizar el trabajo en equipo.

Flujo de Funcionamiento del Sistema

1. El usuario accede desde el navegador.
2. El frontend envía una solicitud al servidor (backend).

3. El backend procesa la solicitud y consulta la base de datos.
4. La base de datos responde con la información solicitada.
5. El backend envía la respuesta al frontend.
6. El frontend actualiza la interfaz en tiempo real.

Este flujo garantiza una gestión organizada, segura y eficiente.

Justificación de la Arquitectura

Inicialmente, el sistema utilizaba almacenamiento local (localStorage), lo que limitaba el funcionamiento a un solo dispositivo.

La implementación de una arquitectura Cliente-Servidor con base de datos permite:

- Acceso multiusuario.
- Persistencia real de la información.
- Mayor seguridad.
- Escalabilidad.
- Mejor control administrativo.
- Implementación profesional acorde a los estándares de Ingeniería Web.

Beneficios de la Arquitectura Propuesta

- Organización estructurada del código.
- Mejor rendimiento y escalabilidad.
- Trabajo colaborativo eficiente.
- Reducción de errores.
- Separación clara entre interfaz y lógica.
- Sistema preparado para futuras mejoras (como pagos en línea).

Diagrama de Estructura Arquitectónica del Proyecto

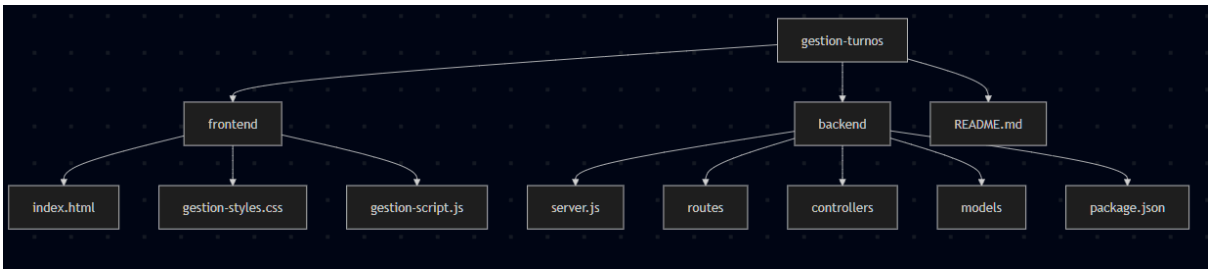


Diagrama de Flujo de Funcionamiento del Sistema

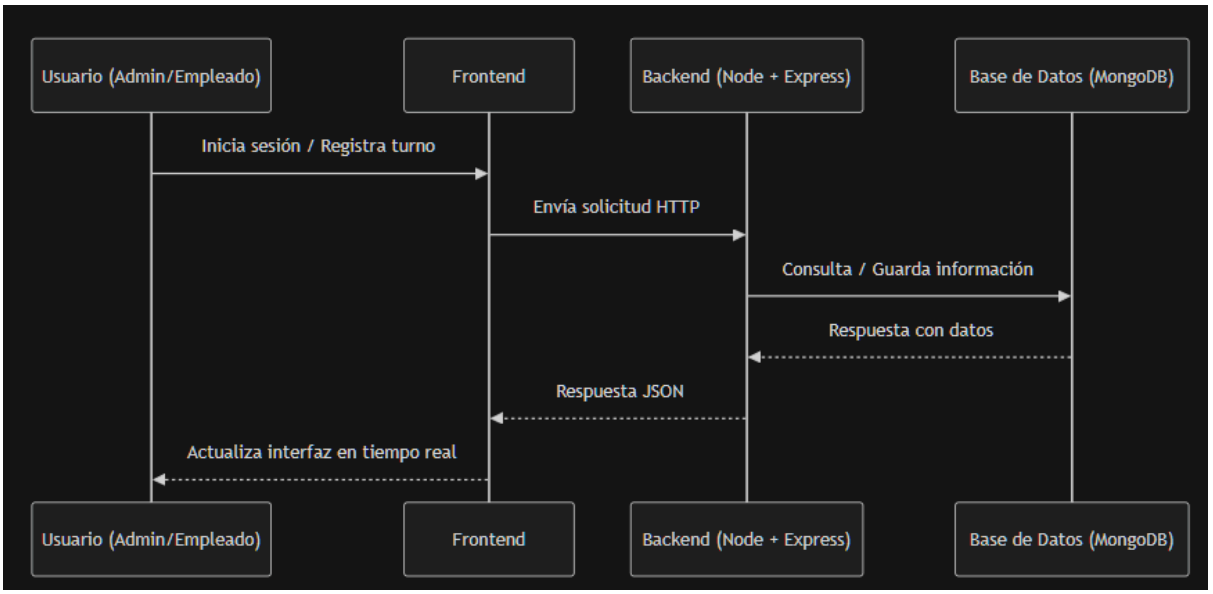


Diagrama de Arquitectura Cliente-Servidor (3 Capas)

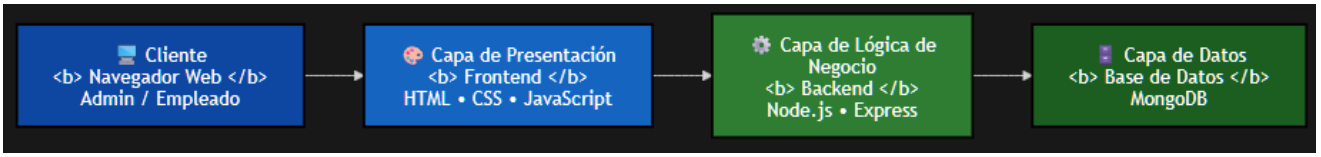


Diagrama MVC (Modelo - Vista - Controlador)

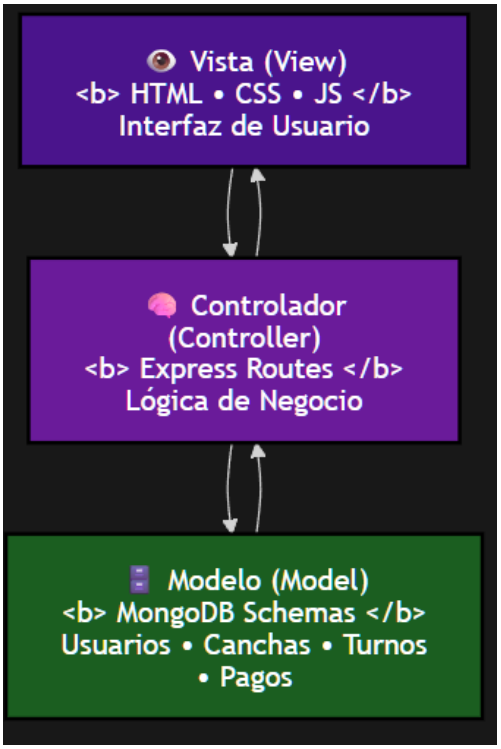


Diagrama de Componentes