Rinku – Code Challenge Coppel

M.C. Luis Eduardo Villela Zavala

Repositorio en GitHub: https://github.com/LuisEdo21/Rinku

Descripción del Sistema:

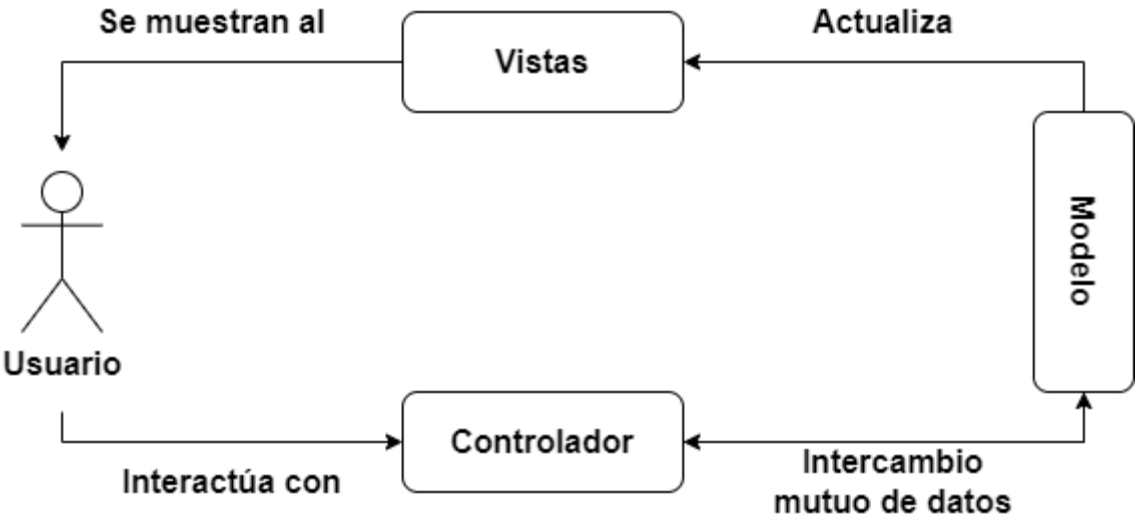
El sistema desarrollado es un pequeño registro de empleados, actividades y sueldos para la empresa cinematográfica Rinku.

Arquitectura del Sistema:

Descripción de la Arquitectura:

La arquitectura empleada para el desarrollo de este sistema es **MVC (Modelo Vista Controlador)**, la cual se emplea en la estructura del sistema, donde se cuenta con los siguientes componentes:

- **Modelo:** Hace referencia a la base de datos del sistema y los procedimientos almacenados con los que cuenta.
- **Vistas:** Archivos EJS con el código HTML que contienen los elementos que el usuario final tendrá visibles desde el navegador.
- **Controlador:** Corresponde al archivo de rutas donde se muestran las funciones que el sistema ejecuta del lado del servidor y que realiza la interacción con el modelo.



Adicional a esto, existe dentro de la vista para añadir nueva actividad un elemento cuyo funcionamiento se basa en una arquitectura de tipo **"Event-Driven"** o **Dirigida por Eventos**; el cual corresponde a la opción Select del número de empleado, que llena en automático los inputs referentes a los datos del empleado; y se realiza un

procedimiento similar para realizar el cálculo de los sueldos una vez que se ha introducido el número de horas trabajadas y el número de entregas realizadas.

Estructura del Sistema:

La estructura con la que cuenta el sistema es la básica que se emplea en cualquier proyecto de **ExpressJS + NodeJS**. Entre los elementos más relevantes con los que cuenta el proyecto se encuentran:

- **Directorios:**
  - **views** → Este directorio contiene todos los archivos EJS empleados para desarrollar las vistas del sistema. Dentro de éste también se encuentra un subdirectorio denominado "partials", en el que se pueden declarar elementos EJS que se pueden colocar en una o varias vistas, lo que evita la creación de código redundante en las vistas.
  - **css** → Contiene los archivos CSS utilizados en el proyecto.
  - **js** → Este directorio no se empleó en el desarrollo del proyecto, de haberse utilizado se hubieran colocado en él los archivos JS con funciones y lógica de programación para poder implementar la funcionalidad.
  - **node_modules** → Todas las librerías de NodeJS instaladas dentro del proyecto, las cuales pueden o no pueden ser utilizadas.
- **Archivos sueltos:**
  - **index.js** → Este archivo cuenta con las configuraciones iniciales para poder ejecutar el sistema, se especifican las librerías principales, el bodyParser, los directorios del proyecto y el motor de vistas EJS; así como la inicialización del sistema.
  - **routes.js** → Contiene todas las rutas necesarias para el funcionamiento del sistema, tanto para cargar las vistas (Por medio de URL's) como la obtención de datos directamente de la base de datos.

Tecnologías y Herramientas Utilizadas:

Para el desarrollo de este sistema se tomó la decisión de realizarlo utilizando tecnologías de desarrollo Web. La justificación para aplicar el desarrollo web a este Code Challenge es:

- El sistema puede adaptarse fácilmente a un sistema mayor, de forma que se puedan reutilizar componentes.
- Es fácil de mantener y de implementar nuevas funcionalidades debido a la estructura que posee.
- Se puede darle el estilo que se desee, ya sea por medio de un framework de Frontend o especificando manualmente las propiedades de diseño utilizando archivos CSS.
- Dominio de las tecnologías empleadas.

Los **entornos de desarrollo** que se emplearon para este proyecto son los siguientes:

- **Visual Studio Code** (versión 1.75.1) como IDE de desarrollo, se empleó para escribir todo el código necesario para el sistema.
- **MySQL Workbench 8.0** como motor de base de datos, aquí se creó la base de datos por medio de queries de MySQL, se realizaron los procedimientos almacenados y se probaron los mismos.
- **Postman** v10.12, el cual se empleó para realizar pruebas de los endpoints creados para obtener datos proveniente de la base de datos. Conforme el sistema fue creciendo en funcionalidad ya no fue necesario seguir haciendo pruebas por medio de Postman.
- **Navegador web Brave** versión 1.50.119 basado en Chromium, además de utilizarse para verificar que el sistema estuviera viéndose y funcionando de forma adecuada, también se utilizó el Inspector de Elementos para realizar algunas tareas de debuggeo del código, por medio del cual se encontraron algunos errores que pudieron ser corregidos posteriormente.
- **Git + GitHub**: Sistema de control de versiones, tanto para la parte local (Git) como para el repositorio remoto (GitHub).

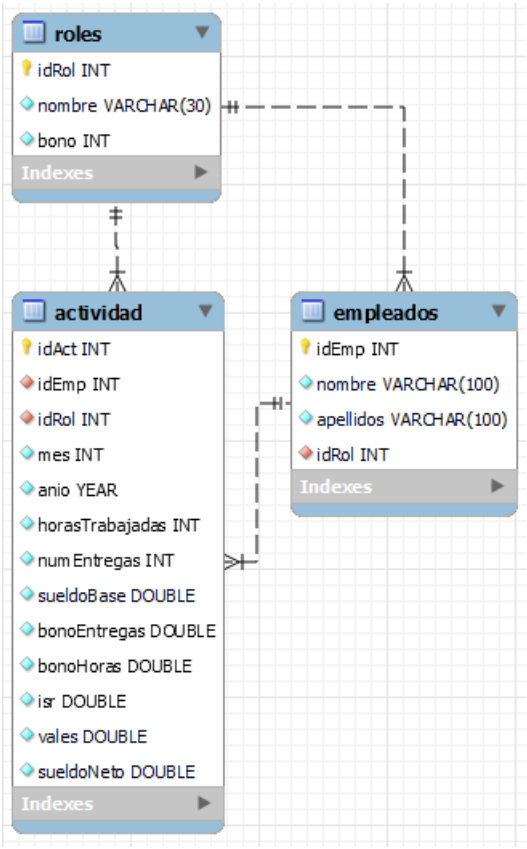
Los lenguajes de programación y frameworks empleados para el desarrollo son:

- **NodeJS + ExpressJS:** Son los frameworks de desarrollo web que fueron utilizados, los cuales se basan en JavaScript. Se utilizó NodeJS para realizar el Backend del sistema y el manejo de las rutas del sistema; y ExpressJS para gestionar el Frontend por medio del sistema de vistas que posee. Se emplearon algunas dependencias para gestionar la conexión a la base de datos MySQL y algunas rutas.
- **MySQL:** Motor de base de datos empleado para crear una pequeña base de datos relacional y los procedimientos almacenados necesarios para su correcto funcionamiento.

Base de Datos:

Para el desarrollo de la base de datos del sistema Rinku se decidió utilizar una base de datos relacional en MySQL, la cual está compuesta por 3 tablas:

- **Roles:** Almacena los 3 roles disponibles que puede tomar un empleado (Chófer, Cargador, Auxiliar).
- **Empleados:** Lleva el registro de los empleados que se den de alta por medio del sistema.
- **Actividad:** Esta tabla llevará el registro de las tareas por empleado (Entregas realizadas) y una relación de horas trabajadas y sueldos por empleado.



Se anexa al documento una imagen del diagrama Entidad – Relación de la base de datos creada para el sistema.

Mejoras y Áreas de Oportunidad:

Se tienen identificadas algunas mejoras y áreas de oportunidad que el proyecto presenta, entre las más destacables se encuentran:

- **Mejorar el apartado visual:** Esto se puede realizar utilizando las propiedades y características que Bootstrap ya incluye, por medio de código CSS o bien, una combinación de ambas.
- **Hacer que los formularios sean más claros y reducidos**, es decir, alinear los componentes para hacerlo más atractivo para el usuario final (Mejorar la experiencia de usuario (UX)).
- **Añadir formatos de datos a los inputs de los formularios** (Identificar valores numéricos, de fecha, a los de texto limitar los caracteres que pueden introducirse de acuerdo con el tamaño máximo permitido en los campos de la base de datos).
- No se incorporaron al sistema, pero se pueden añadir **opciones para modificar y eliminar** tanto **registros** de actividad como empleados. Para hacerlo se deberán crear los procedimientos almacenados respectivos, la ventana para Modificar (Tanto para actividad como para empleados) y los botones para Eliminar.
- Contemplar **mostrar una lista de empleados**, de la misma manera de cómo se hizo con el reporte de actividad.

- Analizar si es posible colocar la **conexión a la base de datos** fuera del archivo de rutas del proyecto.
- Colocar un **módulo de inicio de sesión**, para evitar que cualquier persona pueda ingresar y hacer modificaciones al sistema. Para hacerlo se puede utilizar algún sistema de tokens como **JWT (JSON Web Tokens).**