

Intento 1

Vamos a pedirle a chatpgt que cree un archivo JSON que haga de base de datos para almacenar datos de los usuarios

Entonces, lo que quiero básicamente es u que cree el archivo JSON y lo configure de tal manera que los usuarios pues tengan su hora de entrada, sus pausas, su hora de salida. Así, para empezar, eso. Además, en ese JSON, lo que hay que hacer es jarcotear datos, añadir datos a esa pseudo base de datos de personas aleatorias como Juan, Juanjo, María o algo así, con sus horas de entrada, salidas, pausas y días trabajados. Luego, también hay que decirle que la función de recursos humanos de consultar tiene que ser capaz de leer de ese archivo JSON y que dentro del apartado de usuario, cuando tú estás en tu página principal, donde puedes hacer pausas y demás, el display tiene que actualizarse cuando tú hagas una pausa, mostrarlo, cuando tú empieces jornada, mostrarlo, etcétera.

Listo: dejé un JSON con fichajes de ejemplo y conecté la consulta de RRHH a ese archivo; además añadí un panel de estado en la pantalla de usuario que se actualiza al iniciar/finalizar jornada o pausas.

- `{ } fichajes.json` : nueva pseudo-BD con empleados y registros por día (entrada/salida/pausas). Puedes probar con `juan.perez@empresa.com` , `juanjo.moreno@empresa.com` , `maria.lopez@empresa.com` .
- `<> Consultar_datos_rrhh.html` : ahora carga los fichajes con `fetch` desde `{ } fichajes.json` y usa `empleado.registros` para la tabla, con control de errores.
- `<> usuario.html` : añadí el bloque "Estado actual" y funciones que actualizan horas/estado y la última acción cuando deslizas jornada/pausa.

Intento 2

Seguimos perfilando y añadimos una nueva funcionalidad

ahora hay que hacer que el usuario dentro del apartado resumen.html, manteniendo el estilo visual se sincronice con los datos que vaya poniendo el usuario esto mismo tambien hay que hacerlo con semana.html , ademas en el json hay qe añadiendo lso datos para que desde la funcion de rrhh se puedan consultar los del json

Conecté los fichajes del usuario con el resumen diario y semanal vía `localStorage` , y amplié el JSON para que RRHH tenga más datos consultables.

- `<> usuario.html` : ahora guarda cada acción en `resumen-dia-events` y `fichajes-diaros` y restaura estado/horas si ya hay eventos del día.
- `<> resumen.html` : prioriza `fichajes-diaros` , valida que los eventos sean del día y se refresca si cambia el storage.
- `<> semana.html` : calcula horas por día a partir de `fichajes-diaros` y se actualiza al cambiar el storage.
- `{ } fichajes.json` : añadí más registros y nuevos empleados (`carlos@click.com` , `laura@click.com`) para consultas RRHH.