**Sistema QR Dinámico para Gestión y Seguimiento de Choferes en Planta Industrial**

**1. Introducción**

**El control y seguimiento de choferes y vehículos en plantas industriales es una tarea crítica para la seguridad, logística y eficiencia operativa. Tradicionalmente, estos procesos suelen realizarse manualmente, generando inconsistencias, pérdida de información y demoras.**

**Este proyecto presenta un Sistema Web Dinámico basado en códigos QR que permite la gestión eficiente de choferes mediante la asignación temporal y reutilizable de códigos QR, así como el seguimiento en tiempo real de sus movimientos dentro de la planta.**

**2. Objetivos**

**2.1 Objetivo General**

**Desarrollar un sistema digital para la asignación y seguimiento dinámico de códigos QR a choferes, que permita controlar su ingreso, movimientos internos y egreso en la planta, mejorando la trazabilidad y seguridad.**

**2.2 Objetivos Específicos**

* **Implementar un módulo para el registro rápido y sencillo de nuevos choferes y códigos QR asociados.**
* **Registrar de manera automatizada cada estación o punto por donde pasa el chofer dentro de la planta.**
* **Proporcionar una vista clara de los QR activos dentro de la planta con duración de estancia.**
* **Ofrecer un historial completo con capacidad de búsqueda y filtrado para consultas administrativas y auditorías.**
* **Utilizar tecnologías web y bases de datos robustas para garantizar escalabilidad y facilidad de mantenimiento.**

**3. Tecnologías Utilizadas**

| **Tecnología** | **Versión** | **Descripción** |
| --- | --- | --- |
| **Python** | **3.x** | **Lenguaje principal para backend** |
| **Flask** | **Última** | **Framework web ligero y flexible para desarrollo** |
| **MariaDB/MySQL** | **10.4+** | **Sistema de gestión de base de datos relacional** |
| **mysql-connector-python** | **Última** | **Conector para integración entre Python y MariaDB** |
| **HTML5, CSS3** | **N/A** | **Estructura y estilo de las interfaces web** |
| **Jinja2** | **Incluido Flask** | **Motor de plantillas para generar HTML dinámico** |

**4. Arquitectura del Sistema**

**El sistema está dividido en tres capas principales:**

* **Presentación: Interfaces web generadas con HTML y renderizadas por Flask utilizando Jinja2.**
* **Lógica de negocio: Control de flujos de registro, seguimiento, filtrado y cálculo de duraciones.**
* **Persistencia: Base de datos relacional donde se almacenan registros iniciales y seguimiento de estaciones.**

**5. Diseño Funcional**

**5.1 Menú Principal**

**Permite el acceso rápido a las cuatro funcionalidades principales:**

* **Registro de choferes y asignación de QR.**
* **Seguimiento de estaciones.**
* **Visualización de QR activos.**
* **Consulta del historial completo con búsqueda.**

**5.2 Registro**

**Formulario que captura los datos esenciales del chofer, código QR y estación inicial. Se almacena la fecha y hora automáticamente al guardar.**

**5.3 Seguimiento**

**Captura estaciones donde el chofer ha pasado, permitiendo una trazabilidad temporal completa dentro de la planta.**

**5.4 QR Activos**

**Muestra solo los códigos QR que se encuentran dentro de la planta (no han marcado salida o fin), mostrando duración y estaciones recorridas.**

**5.5 Ver Historial Completo**

**Listado completo con opción de buscar por QR, folio, chofer, empresa o estación, facilitando auditorías y consultas rápidas.**

**6. Modelo de Base de Datos**

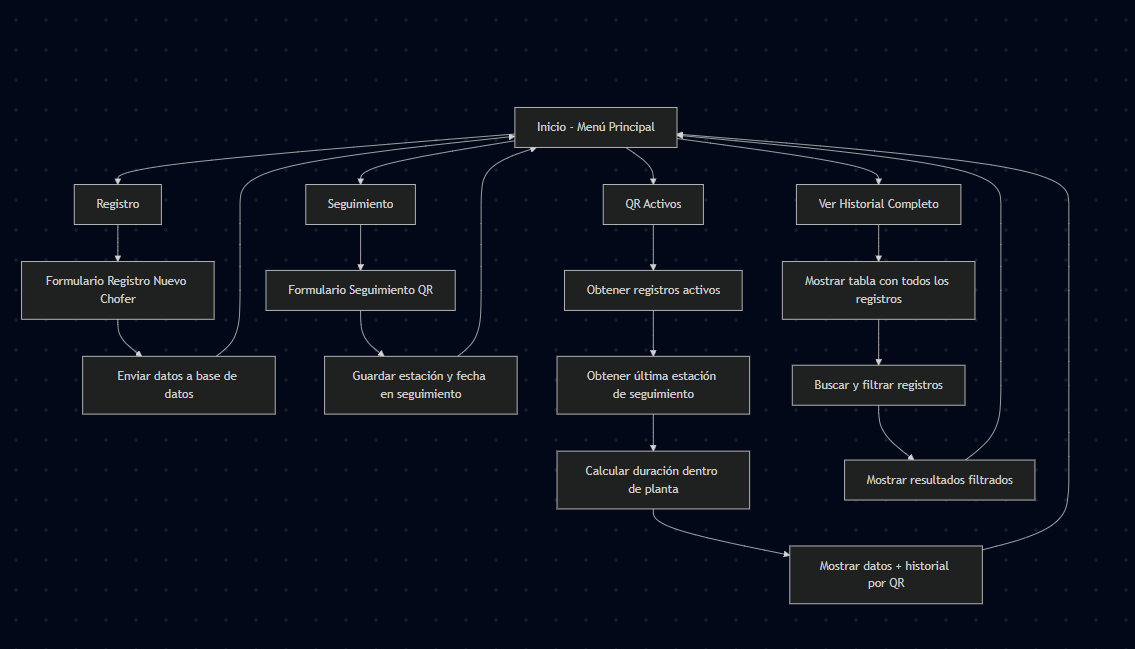
**6.1 Tablas principales**

| **Tabla** | **Campos Principales** | **Descripción** |
| --- | --- | --- |
| **registros** | **id, qr\_id, folio, chofer, licencia, empresa, tipo\_carga, estacion, fecha, activo** | **Datos iniciales del chofer y QR asignado** |
| **seguimiento** | **id, qr\_id, estacion, fecha** | **Historial de estaciones y tiempos** |

**6.2 Relaciones**

* **La tabla seguimiento está relacionada con registros a través del campo qr\_id.**
* **El campo activo en registros indica si el QR está actualmente activo (1) o ha salido (0).**

**7. Diagrama de Flujo**

****

**8. Detalles Técnicos**

**8.1 Registro y seguimiento**

* **Uso de consultas parametrizadas para evitar inyección SQL.**
* **Fechas registradas con función NOW() para consistencia temporal.**
* **Consultas optimizadas con subconsultas para obtener última estación.**

**8.2 Visualización y búsqueda**

* **Uso de plantillas Jinja2 para renderizar resultados dinámicos.**
* **Filtros aplicados con cláusulas SQL LIKE para búsquedas flexibles.**
* **Presentación en tablas estilizadas para mejor experiencia usuario.**

**9. Pantallas y Experiencia de Usuario**

* **Menú principal con botones claros y accesibles.**
* **Formularios sencillos, con validación básica (campos requeridos).**
* **Tablas con paginación y estilos para facilitar lectura.**
* **Opción de búsqueda rápida en historial para eficiencia.**

**10. Resultados y Beneficios**

* **Reducción en tiempos de registro y seguimiento manual.**
* **Mayor control sobre el movimiento de choferes dentro de planta.**
* **Acceso rápido a información histórica para auditorías.**
* **Facilita la gestión logística y aumenta la seguridad operativa.**

**11. Recomendaciones y Próximos Pasos**

* **Implementar autenticación para control de acceso.**
* **Integrar escaneo automático de QR con dispositivos móviles.**
* **Añadir reportes gráficos y estadísticas en tiempo real.**
* **Considerar migración a frameworks frontend para interfaz más moderna.**

**12. Conclusión**

**El sistema QR Dinámico desarrollado satisface las necesidades planteadas para el control logístico en planta, siendo una base escalable y adaptable para futuras mejoras y ampliaciones.**

**Anexos**

* **Código fuente principal (app.py).**
* **Diagramas Mermaid.**
* **Estructura y script de base de datos.**
* **Capturas de pantalla del sistema funcionand**