UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

Introducción a las bases de datos

Ingeniería en ciencias de la computación

RECSIN

Manual técnico

GRUPO: 6CC2

ISAAC GONZÁLEZ AGUILERA 368048

ANGEL ALONSO LOPEZ QUIÑONEZ 355302

LUIS ARTURO HERNANDEZ CASTILLO 367685

Chihuahua, Chihuahua **28 de Mayo del 2025**

Índice

INTRODUCCIÒN

Objetivo del Manual

Este documento proporciona una descripción técnica detallada de RECSIN, un sistema de gestión de servicios técnicos orientado a facilitar el seguimiento de solicitudes, la administración de relaciones con los clientes y la consolidación de una base de conocimientos sobre soluciones aplicadas. RECSIN se compone de:

- Un frontend desarrollado en React
- Un backend implementado con Python (Flask)
- Una base de datos híbrida que utiliza Oracle y SQL Server para el almacenamiento persistente
- El presente manual está dirigido a desarrolladores, administradores del sistema y personal de soporte técnico. Su propósito es proporcionar los lineamientos necesarios para la instalación, configuración, mantenimiento y solución de problemas del sistema.

Audiencia Objetivo

Este manual está dirigido a los siguientes perfiles técnicos:

- **Desarrolladores Backend y Frontend:** Requieren comprender la arquitectura del sistema, la estructura del código y la interacción entre los módulos React (frontend), Flask (API) y las bases de datos (Oracle y SQL Server).
- **Administradores de Bases de Datos (DBA):** Responsables de la gestión, mantenimiento y respaldo de los esquemas utilizados en Oracle y SQL Server, así como de la integridad de las relaciones entre las tablas.
- **Ingenieros DevOps o de Infraestructura:** Encargados de la configuración, despliegue y monitoreo del sistema RECSIN en entornos de desarrollo y producción.
- **Personal de Soporte Técnico de Nivel Avanzado:** Que necesitan diagnosticar y resolver incidencias relacionadas con la funcionalidad del sistema, conexiones a la base de datos o fallos en el servidor.
- **Documentadores Técnicos:** Que participen en la elaboración de guías de usuario, flujos operativos, y referencias de endpoints o estructuras de datos.

DESCRIPCIÓN GENERAL DEL SISTEMA

Áreas Funcionales Principales

1. Gestión de Clientes

Administra la información de los clientes mediante el esquema cliente. Los datos se almacenan de forma estructurada y permiten la recuperación eficiente mediante endpoints dedicados.

Estructura principal:

- id_cliente: CHAR(18), clave primaria
- nombre_completo: VARCHAR2(20)
- numero: INTEGER (teléfono)
- Operaciones clave:
- Crear cliente: POST /agregarCliente
- Consultar todos: GET /allClients
- Integración con servicios mediante claves foráneas
- Fuentes: BaseRECSIN.sql (líneas 1-11), servidor-flask/main.py (líneas 12-14, 21-23)

2. Gestión de Servicios

Registra y gestiona los servicios técnicos prestados por los empleados en distintas sucursales, con información detallada de equipos, descripciones, costos y fechas.

Estructura principal:

- id_servicio: CHAR(18), clave primaria
- equipo, descripcion, costo: VARCHAR2(20)
- fecha_entrega: DATE
- id_empleado, id_sucursal: CHAR(18), claves foráneas

Operaciones clave:

- Registrar nuevo servicio: POST /agregarServicio
- Seguimiento de entregas
- Asociación con empleados y sucursales
- Fuentes: BaseRECSIN.sql (líneas 49-63), servidor-flask/main.py (líneas 16-19)

3. Gestión de la Base de Conocimientos

Permite consolidar el conocimiento institucional sobre soluciones aplicadas mediante operaciones CRUD sobre la tabla servicios_hechos.

Estructura principal:

- id_cliente, id_servicio: CHAR(18), clave compuesta
- fecha_salida: DATE
- solucion: VARCHAR2(20)

Operaciones clave:

- Crear: POST /agregarKnowle
- Leer: GET /verKnowledge
- Actualizar: PUT /actualizarKnowledge
- Eliminar: DELETE /borrarKnowledge
- Fuentes: BaseRECSIN.sql (líneas 67-78), servidor-flask/main.py (líneas 25-39)

Relaciones del Esquema de Base de Datos

El diseño del esquema de RECSIN refuerza la integridad referencial mediante claves externas entre tablas como cliente, encargos, servicios, empleados, sucursal y servicios_hechos, garantizando la consistencia a lo largo de todo el flujo operativo.

Ejemplos de relaciones clave:

- cliente ↔ encargos mediante id_cliente
- servicios ↔ empleados, sucursal mediante id_empleado y id_sucursal
- servicios_hechos combina referencias a cliente y servicios

ARQUITECTURA DEL SISTEMA

Propósito y alcance

Aquí se detalla los tres niveles del sistema RECSIN y como el frontend de React, el servidor API de backend de flask y la base de datos Oracle. Como trabajan conjuntamente para proporcionar la funcionalidad de gestión de servicios. Esto incluye los patrones de comunicación entre niveles, el flujo de datos y los componentes arquitectónicos clave.

RECSIN implementa una arquitectura clásica de tres niveles: una capa de presentación (frontend de React), una capa de aplicación (backend de Flask) y una capa de datos (base de datos de Oracle).

Diagrama de arquitectura de tres niveles

Diagrama de React

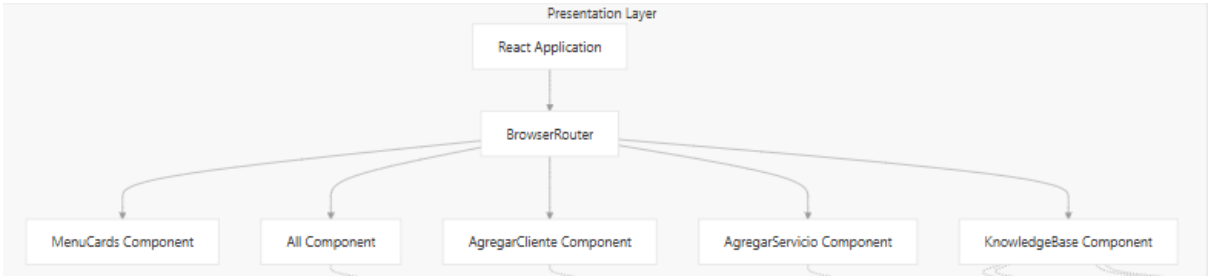


Diagrama de aplicación

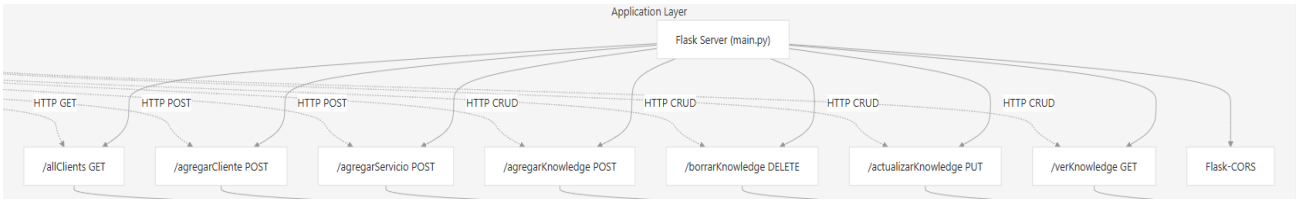
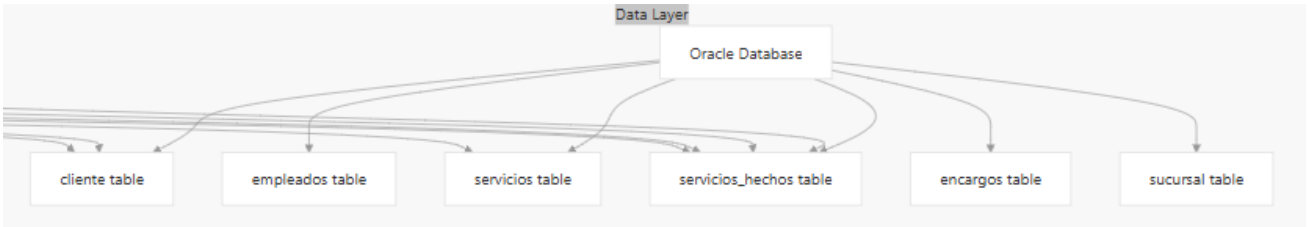
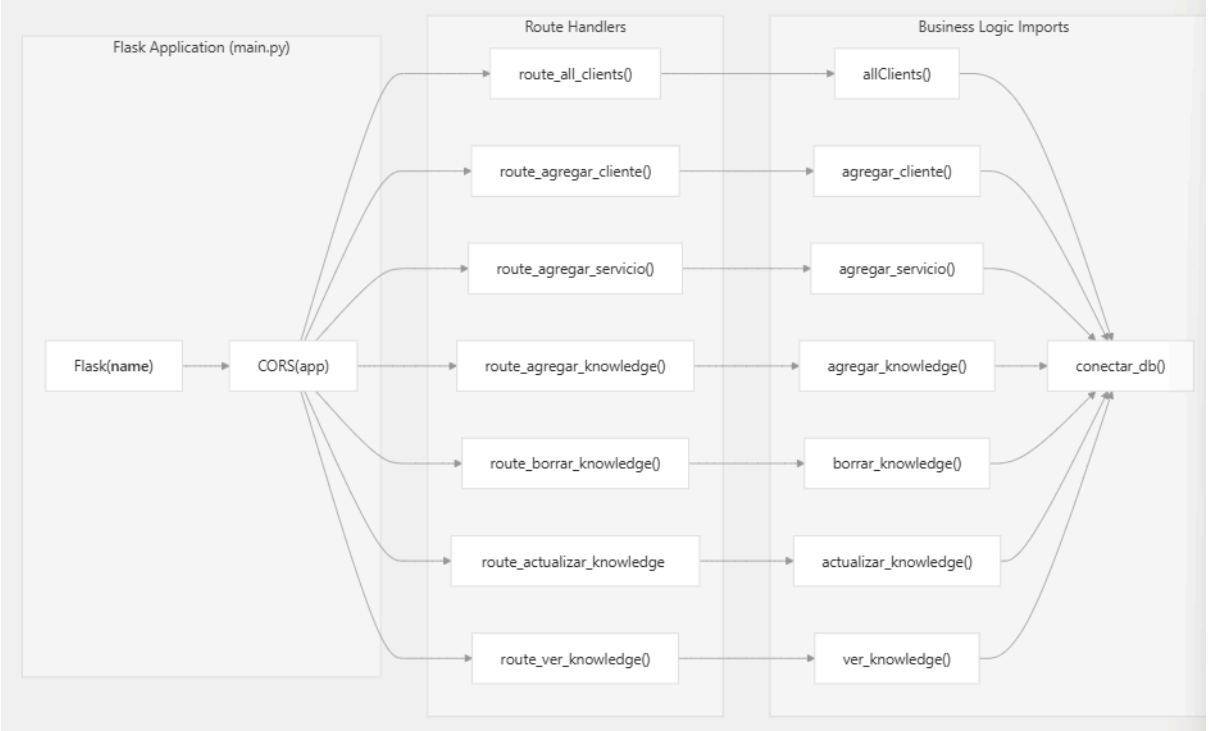


Diagrama de datos



Capa de presentación

El frontend implementa una aplicación de página con React y enrutamiento del lado del cliente mediante BrowserRouter. La estructura de la aplicación se centra en un componente de navegación principal que enruta a componentes funcionales especializados.



Capa de datos

La capa de datos consta de una base de datos en Oracle con un esquema relacional normalizado que respalda el dominio de gestión de servicios. El esquema incluye seis tablas principales con relaciones de clave externa establecidas.

Estructura del esquema de base de datos

Tabla	Clave Principal	Objetivo
cliente	**id_cliente**	Almacenamiento de información del cliente
empleados	**id_empleados**	registros de empleados
servicios	**id_servicio**	Definiciones de servicios
servicios_hechos	**id_cliente, id_servicio**	Registros de servicio completados
encargos	**id_encargo**	Ordenes de servicio
sucursal	**id_sucursal**	Datos de ubicación de sucursales

Relaciones de tablas

La base de datos implementa integridad referencial a través de restricciones de clave externa:

- encargos.id_cliente -> cliente.id_cliente
- encargos.id_empleado -> empleados.id_empleado
- servicios.id_empleado -> empleados.id_empleado
- servicios.id_sucursal -> sucursal.id_sucursal
- servicios_hechos.id_cliete -> cliente.id_cliente
- servicios_hechos.id_servicio -> servicios.id_servicio

Patrones de comunicación

Comunicación frontend-backend

El frontend de React se comunica con el backend de Flask mediante solicitudes HTTP a **localhost:5000.** El **All** componente demuestra el patrón de comunicación estándar.

const response = await fetch (“http://localhost:5000/allClients”);

Este patrón se replica en todos los componentes frontend que requieren interacción backend, con métodos HTTP apropiados (GET, POST, PUT, DELETE) según el tipo de operación.

Comunicación entre el backend y la base de datos

El backend de Flash se conecta a la base de datos Oracle mediante la conectar_db() función importada del functions.db módulo. todas las funciones de lógica de negocio utilizan esta conexión a la base de datos para realizar operaciones CRUD en las tablas correspondientes.

Puntos de integración del sistema

El sistema mantiene un acoplamiento flexible entre niveles a través de interfaces bien definidas:

1. **Interfaz frontend-backend:** API HTTP RESTful con intercambio de datos JSON
2. **Interfaz de bases de datos backend:** Controlador de base de datos Oracle con operaciones SQL.
3. **Interfaz de componentes:** React Router para la navegación interna del frontend.

Esta arquitectura permite el desarrollo y la implementación independientes de cada nivel manteniendo la cohesión del sistema a través de protocolos de comunicación estandarizados.

Aplicación Frontend

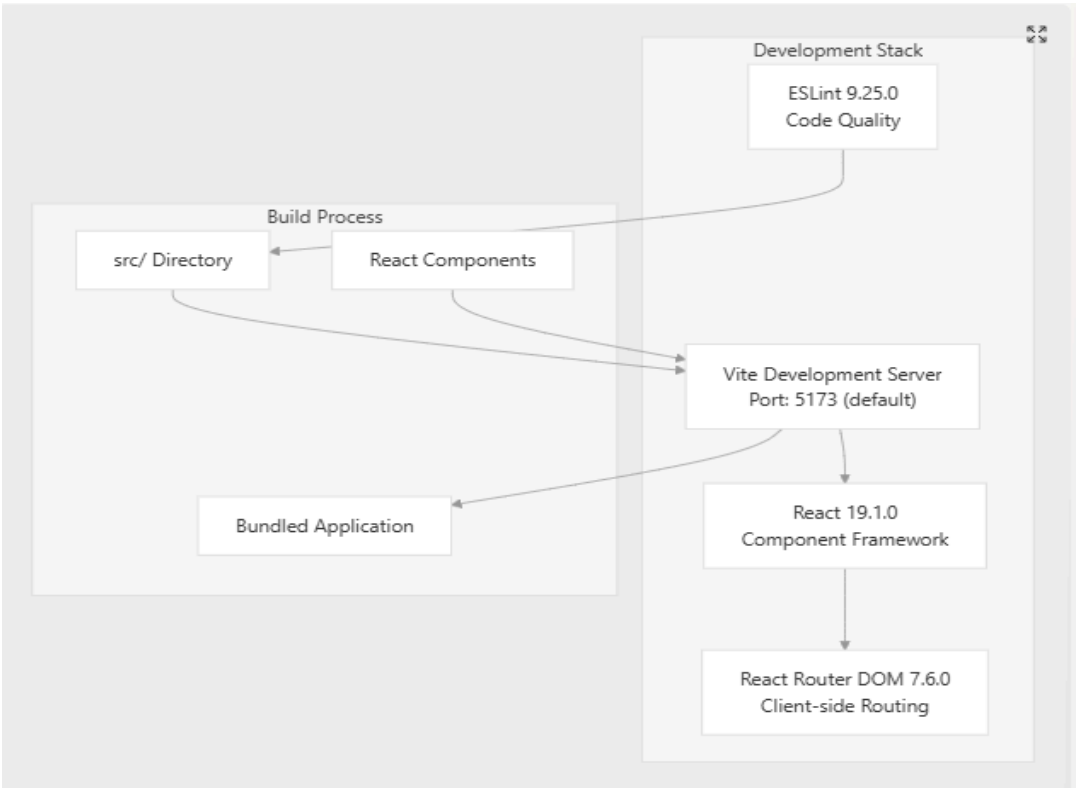
Propósito y alcance

Se describe la aplicación frontend basada en React del sistema de gestión de servicios RECSIN. La interfaz proporciona interfaces de usuario para la gestión de clientes, el seguimiento de servicios y las operaciones de la base de conocimientos mediante una arquitectura de aplicación de página única.

La interfaz de RECSIN está construida utilizando tecnologías web modernas:

TECNOLOGÍA	VERSIÓN	OBJETIVO
React	19.1.0	Marco de interfaz de usuario principal
DOM del enrutador React	7.6.0	Enrutamiento del lado del cliente
Fast	6.3.5	Herramienta de compilación y servidor de desarrollo
ESLint	9.25.0	Calidad del código

Configuración del entorno de desarrollo

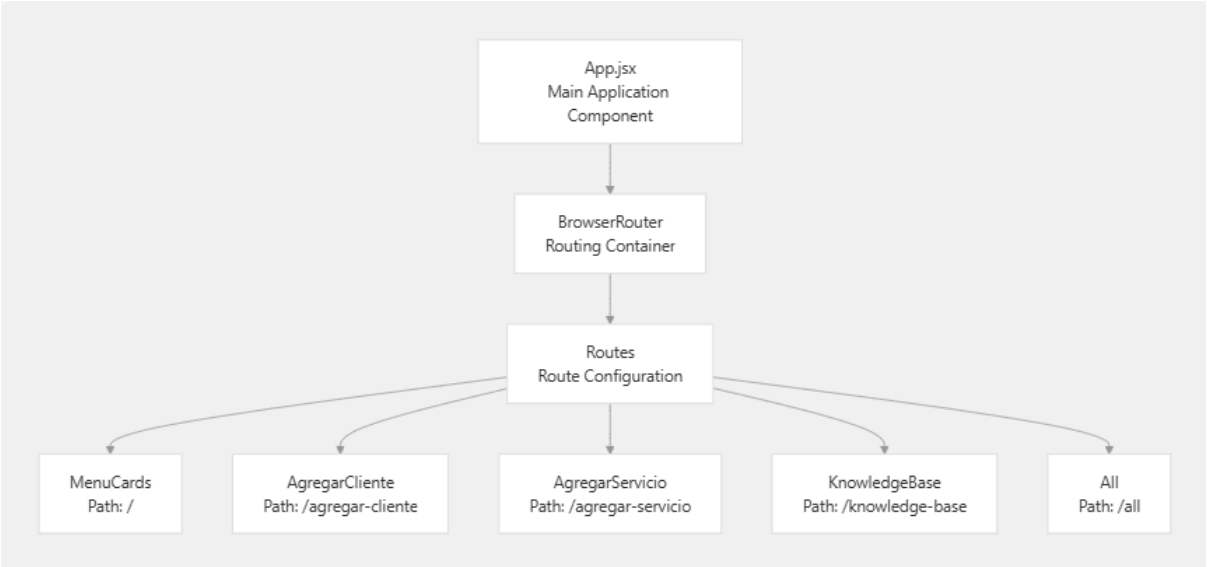


Estructura y enrutamiento de la aplicación

La aplicación frontend está organizada alrededor de un sistema de enrutamiento central que proporciona acceso a diferentes áreas funcionales del sistema RECSIN.

Punto de entrada principal de la aplicación

El **App** componente actúa como componente raíz, y define la estructura de enrutamiento de la aplicación utilizando React Router.



Configuración de ruta

La aplicación define cinco rutas principales, cada una de las cuales cumple una función específica:

Ruta	Components	Funcion
/	MenuCards	Panel de navegación principal
/agregar_cliente	AgregarCliente	Formulario de registro de cliente
/agregar_servicio	AgregarServicio	Formulario de creación de servicios
/knowledge_base	KnowledgeBase	Interfaz de gestión del conocimiento
/all	All	Listado y descripción general de clientes

Componente de gestion de datos del cliente

El **All** componente demuestra los patrones de obtención y visualización de datos del frontend utilizados en toda la aplicación.

Gestión del estado de los componentes

El **All** componente utiliza ganchos de React para la gestión del estado:

- **clientes** estado para almacenar los datos del cliente obtenidos
- **error** estado para el manejo de errores
- **use Effect** gancho para la obtencion de datos en el montaje del componente

Integración de backend

Protocolo de comunicación API

El fronted se comunica con el backend de Flask mediante solicitudes HTTP. El patrón estándar utilizado en todos los componentes es el siguiente:

Configuración de URL base

- Servidor backend: http://localhost:5000
- Métodos HTTP estándar: GET, POST, PUT, DELETE
- Formato de datos JSON para solicitud/respuesta

Configuración de desarrollo

Configuración del proyecto

La aplicación utiliza Vite como herramienta de compilación y servidor de desarrollo, configurado a través de scripts npm estándar:

Guión	Dominio	Objetivo
dev	vite	Iniciar el servidor de desarrollo
build	vite build	Construir para producción
lint	eslint .	Ejecutar comprobaciones de calidad del código
preview	vite preview	Vista previa de la versión de producción

INTEGRACIÓN DE BASES DE DATOS

Esta es la capa de integración de bases de datos del sistema de gestión de servicios RECSIN, incluyendo la configuración de la conexión a la base de datos de Oracle, el diseño de esquemas y los patrones de acceso de datos. Esta capa sienta las bases para las operaciones de datos que utilizan las funciones de la API de backend.

Configuración de la conexión a la base de datos

El sistema RECSIN utiliza Oracle Database como su almacén de datos principal y la administración de la conexión se realiza a través de la **conectar_db()** función en el módulo de utilidades de la base de datos.

Funcion de conexion

La **conectar_bd()** función en **servidor-flask/funciones/db.py** . Proporciona una fábrica de conexiones centralizada que devuelve un objeto de conexión de bases de datos Oracle activo mediante la **oracledb** biblioteca en Python.

Descripción general del esquema de bases de datos

La base de datos de RECSIN implementó un esquema integral, de gestión de servicios con seis tablas principales que manejan las relaciones con los clientes, la gestión de los empleados, el seguimiento del servicio y la funcionalidad de la base de datos.

Tablas básicas

Nombre de la tabla	Clave principal	Objetivo
cliente	id_cliente	Información del cliente y datos de contacto
empleados	id_empleado	Registros de empleados e información de contratación
servicios	id_ servicio	Definiciones y especificaciones de servicios
servicios_hechos	id_cliente, id_servicio	Base de knowledge de servicios completados
encargos	id_encargo	Ordenes de trabajo y asignaciones
sucursal	id_sucursal	Información sobre la ubicación de la sucursal

REFERENCIAS BIBLIOGRÁFICAS

- https://deepwiki.com/IsacProg61/ProyectoFinalDB/4.3-database-integration