### 4.6.6 Especificación del Caso de Uso del Sistema: CU011 Generar Hoja de ruta

Objetivo : Determinar un mecanismo eficiente que permita sugerir las rutas adecuadas para la ejecución de visitas técnicas programadas y/o reprogramadas.

Fundamentacion:permitirá recomendar una ruta adecuada para cada uno de los técnicos con la finalidad de que éstos no pierdan demasiado tiempo en desplazarse a cada cliente para la ejecución oportuna de la programación de visitas técnicas diarias considerando la prioridad de atención al cliente.

Problemas que se resuelven : No se obtiene información real y de manera inmediata referente a la programación de visitas técnicas considerando la prioridad de atención deacuerdo a la emergencia dada por el cliente y aprobada por el responsable de planificación.

No se genera información de rutas adecuadas para la ejecución de visitas técnicas programadas ya que no se cuenta con información oportuna y necesaria de los tiempos de duración del servicio y prioridad de atención en cada visita para el cumplimiento de los servicios según lo planificado.

**1. Actores del Sistema**

AS003 - Operador Técnico

Rol encargado de ejecutar las visitas técnicas.

**2. Propósito**

Realizar la generación de la hoja de ruta.

**3. Breve Descripción**

El caso de uso inicia cuando el Operador de Asistencia Técnica ingresa a la opción Generar Hoja de Ruta en la que podrá generar la Hoja de ruta para las visitas técnicas, el sistema procesa la información generando un croquis de la Hoja de ruta de trabajo y finaliza cuando almacena el registro actualizado.

.**4. Flujo de Eventos**

**4.1. Flujo Básico**

4.1.1. El Operador de Asistencia Técnica ingresa a la opción Generar Hoja de ruta. [Figura1]

4.1.2. El sistema muestra el listado de visitas técnicas programadas y/o reprogramadas con los siguientes datos: nro de programación, servicio, técnico, fecha visita, duración, estado, orden, generar HR. Incluye las opciones: Buscar, Limpiar, Generar y Salir. [Figura2]

4.1.3. Si el Operador de Asistencia Técnica selecciona la opción “Buscar”

* + - 1. Ver el Subflujo Buscar en el paso 4.2.1.
    1. Si el Operador de Asistencia Técnica selecciona la opción “Limpiar”
       1. Ver el Subflujo Limpiar en el paso 4.2.2.

4.1.5. Si el Operador de Asistencia Técnica selecciona la opción “Generar Ruta”.

4.1.5.1. El sistema procesa la información ***(Ver 9.2. Algoritmo Dijkstra)***

4.1.5.2. El sistema muestra el listado de visitas técnicas programadas y/o reprogramadas en un **mapa** con los siguientes datos: nro de programación, servicio, técnico, fecha visita, duración, estado, orden, generar HR. Incluye las opciones: Guardar y Salir. [Figura3] ***[RN047]***

4.1.5.3. El Operador de Asistencia Técnica visualiza el listado de visitas técnicas programadas y/o reprogramadas.

4.1.6. El Operador de Asistencia Técnica elige la opción “Guardar”.

* + 1. El Sistema valida, genera y graba los datos ingresados.
    2. El Sistema muestra un mensaje “Se Generó la Ruta para planificación de mantenimiento” con la opción Aceptar.
    3. El Operador de Asistencia Técnica selecciona Aceptar en el mensaje.

4.1.10. El sistema cierra la interfaz “Generar Hoja de ruta” y finaliza el caso de uso.

**4.2. Subflujos**

**4.2.1. Buscar**

4.2.1.1. El Operador de Asistencia Técnica selecciona fecha de inicio y fecha fin.

4.2.1.2. El Operador de Asistencia Técnica seleccion la opción “Buscar”

4.2.1.3. El sistema busca y muestra el listado filtrado de visitas técnicas programadas y/o reprogramadas en un plano con los siguientes datos: : nro de programación, servicio, técnico, fecha visita, duración, estado, orden, generar HR. Incluye las opciones Buscar, Limpiar, Generar y Salir. [Figura2]

4.2.1.4. El subflujo finaliza.

**4.2.2. Limpiar**

4.2.2.1. El Operador de Asistencia Técnica selecciona la opción “Limpiar”.

4.2.2.2. El sistema regresa paso 4.1.2. del flujo básico.

**4.3. Flujos Alternos**

4.3.1 Si en el paso 4.1.1 del flujo básico. No existen visitas técnicas programadas/reprogramadas

4.3.1.1 El sistema muestra el mensaje “No tiene visitas técnicas programadas/reprogramadas”

4.3.1.2 El caso de uso finaliza.

4.3.2 Si en el paso 4.2.1.3 del subflujo Buscar. No existen visitas técnicas programadas/reprogramadas

4.3.2.1 El sistema muestra el mensaje “No tiene visitas técnicas programadas/reprogramadas”

4.3.2.2 El subflujo finaliza.

**5. Precondiciones**

**5.1 Acceso al sistema del Operador** **Técnico**

El usuario Operador Técnico crea una sesión.

**5.2 Visitas técnicas programadas**

Se carga el listado de visitas técnicas programadas y/o reprogramadas.

**6. Poscondiciones**

**6.1 Generación de Hoja de ruta**

Se genera la hoja de ruta de la(s) visita(s) programada(s) y/o reprogramada(s) asignada(s) a un técnico.

**7. Puntos de Extensión**

7.1 No Aplica.

**8. Reglas de negocio**

8.1 **RN047- Generación de Rutas**

La visita técnica genera una ruta.

**9. Adicional**

**9.1. Prototipo(s)**

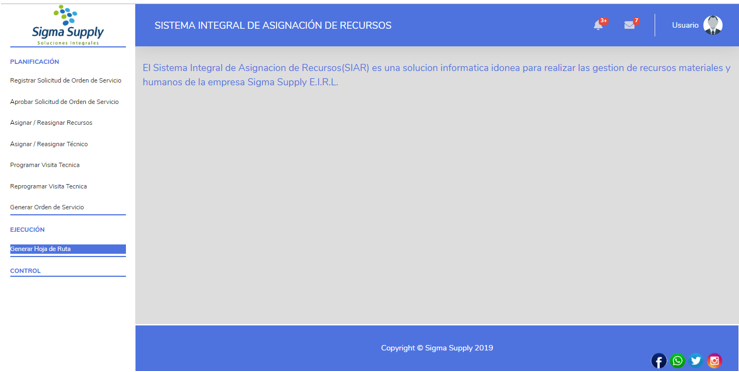


Figura 1. Pantalla de Menú Principal

Fuente: Elaboración Propia

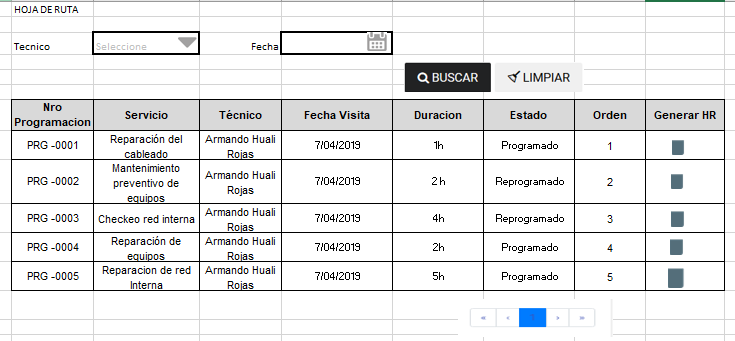


Figura 2. Pantalla de CUS007 Generar Hoja de ruta – Listado de Programaciones

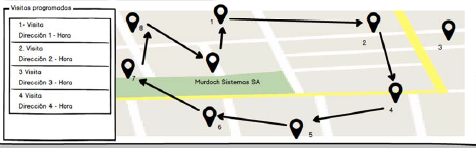
Fuente: Elaboración Propia



Figura 3. Generar Hoja de ruta diaria

Fuente: Elaboración Propia





**9.2. Algoritmo Dijkstra (REVISAR FUNCION DIJKSTRA REALIZADA EN POSTGRESQL BASE DE DATOS) query de consulta: select sistema.dijkstra('SELECT id,source, target,cost FROM sistema.graph',1,null)**

|  |
| --- |
| public class Grafico {  private Set<Nodo> nodos = new HashSet<>();    public void addNodo(Nodo nodo)  {  nodos.add(nodo);  }  // getters and setters  }  public class Nodo {  private String nombre;    private List<Nodo> listaNodo = new LinkedList<>();    private Integer distancia = Integer.MAX\_VALUE;    Map<Nodo, Integer> adyacentes = new HashMap<>();    public void agregarDestino(Nodo destino, int distancia)  {  adyacentes.put(destino, distancia);  }    public Nodo(String nombre)  {  this.nombre = nombre;  }    // getters and setters  }  public static Grafico calcularCaminoCorto(Grafico grafico, Nodo nodo) {  nodo.setDistancia(0);    Set<Nodo> contadoNodos = new HashSet<>();  Set<Nodo> sinContarNodos = new HashSet<>();    sinContarNodos.add(nodo);    while (sinContarNodos.size() != 0) {  Nodo currentNodo = getLowestDistanciaNodo(sinContarNodos);  sinContarNodos.remove(currentNodo);  for (Entry < Nodo, Integer> nodoPareja:  currentNodo.getAdjacentNodos().entrySet()) {  Nodo ayacenteNodo = nodoPareja.getKey();  Integer valor = nodoPareja.getValue();  if (!contadoNodos.contains(ayacenteNodo)) {  calcularMinimaDistancia(ayacenteNodo, valor, currentNodo);  sinContarNodos.add(ayacenteNodo);  }  }  contadoNodos.add(currentNodo);  }  return grafico;  }  private static Nodo obtenerDistanciaCorta(Set<Nodo> sinContarNodos) {  Nodo nodoCercano = null;  int distanciaCorta = Integer.MAX\_VALUE;  for (Nodo nodo: sinContarNodos) {  int distanciaNodo = nodo.getDistance();  if (distanciaNodo < distanciaCorta) {  distanciaCorta = distanciaNodo;  nodoCercano = nodo;  }  }  return nodoCercano;  }  private static void calcularMinimaDistancia(Nodo evaluacionNodo,  Integer valor, Nodo recursoNodo)  {  Integer recursoDistancia = recursoNodo.getDistance();  if (recursoDistance + valor < evaluationNodo.getDistance()) {  evaluationNodo.setDistance(recursoDistancia + valor);  LinkedList<Nodo> rutaCorta = new LinkedList<>(recursoNodo.obtenerDistanciaCorta());  rutaCorta.add(recursoNodo);  evaluationNodo.setShortestPath(rutaCorta);  }  } |