Flujos de Interacción de WhatsApp

Route Manager SaaS incorpora un sistema de comunicación automatizada a través de WhatsApp, diseñado para guiar a los clientes a través de diversas opciones, como confirmar, reprogramar o cancelar servicios. Un árbol binario completo es una estructura de datos ideal para modelar estos flujos de decisión ramificados de manera eficiente, permitiendo una interacción estructurada y predecible con el usuario.

Historia de usuario Nro.	1	**Título:**	Navegar Flujo de Decisión de WhatsApp
Descripción	COMO:	Cliente	
	QUIERO:	Interactuar con un sistema automatizado de WhatsApp que me guíe a través de opciones claras (confirmar, reprogramar, cancelar)	
	PARA:	Gestionar mi servicio de mensajería de forma rápida y autónoma sin intervención humana directa.	
Criterios de aceptación	• El sistema debe presentar una pregunta inicial y dos opciones de respuesta (sí/no o A/B). • Cada opción debe llevar a una nueva pregunta, una acción final o un mensaje de confirmación. • La interacción debe seguir una estructura de árbol donde cada nodo interno es una pregunta y cada hoja es una acción o resultado. • El sistema debe poder evaluar la ruta de decisión del usuario y llegar a la conclusión correcta (ej. "Servicio Confirmado", "Servicio Reprogramado"). • Si se ingresa una opción inválida, el sistema debe indicar "OPCION INVALIDA" y repetir la pregunta.		

Historia de Usuario Nro. 1: Gestionar Cola de Servicios Pendientes

Aunque el árbol de decisión actual es relativamente simple, manejando confirmaciones, reprogramaciones y cancelaciones, la evolución futura de Route Manager podría requerir la gestión de escenarios más complejos, como la notificación de incidencias, la consulta del estado de pago o la modificación de direcciones de entrega. Una estructura de árbol binario facilita la expansión sistemática de estas rutas de decisión. La propiedad de "completo" en el árbol asegura un uso eficiente de la memoria y tiempos de recorrido predecibles (O(log N) para la profundidad, O(N) para un recorrido completo), lo cual es fundamental para manejar un alto volumen de interacciones concurrentes con los clientes sin introducir latencia significativa. Esta elección de diseño anticipa el crecimiento futuro de las funcionalidades.

Programa para Árbol de Decisión de WhatsApp

La estructura conceptual del programa en Java implicaría una clase WhatsAppDecisionTree que gestionaría el árbol. Una clase Node representaría cada elemento del árbol, pudiendo ser una pregunta, una opción o una acción final. Los métodos incluirían buildTree() para construir la estructura, traverseTree(String inputSequence) para recorrer el árbol según las entradas del usuario, y evaluateDecision(String inputSequence) para determinar la acción final. El árbol podría definirse de forma preconfigurada o cargarse desde un archivo de configuración.

Entrada	Una única línea que representa una secuencia de elecciones del usuario (por ejemplo, "0 1 0" para izquierda, luego derecha, luego izquierda). Cada elección es '0' (izquierda/No) o '1' (derecha/Sí), separadas por un espacio. La secuencia termina cuando se alcanza una acción final o se agota la entrada.
Salida	● Para cada paso, se debe imprimir la pregunta/opciones actuales. ● Para una entrada inválida, se debe imprimir "OPCION INVALIDA". ● Finalmente, se debe imprimir la acción/resultado determinado (ej. "Servicio Confirmado", "Servicio Reprogramado", "Servicio Cancelado").

Instrucciones para Calificación Automática:

- La clase principal debe llamarse ArbolBinarioCompleto.
- Dentro de la clase ArbolBinarioCompleto, debe existir un método llamado ejecutar.
- Únicamente se deben imprimir los resultados finales y los mensajes de error.

Casos de prueba (visibles)

Entradas de ejemplo 1	`0`
Salida de ejemplo 1	`Servicio Confirmado`

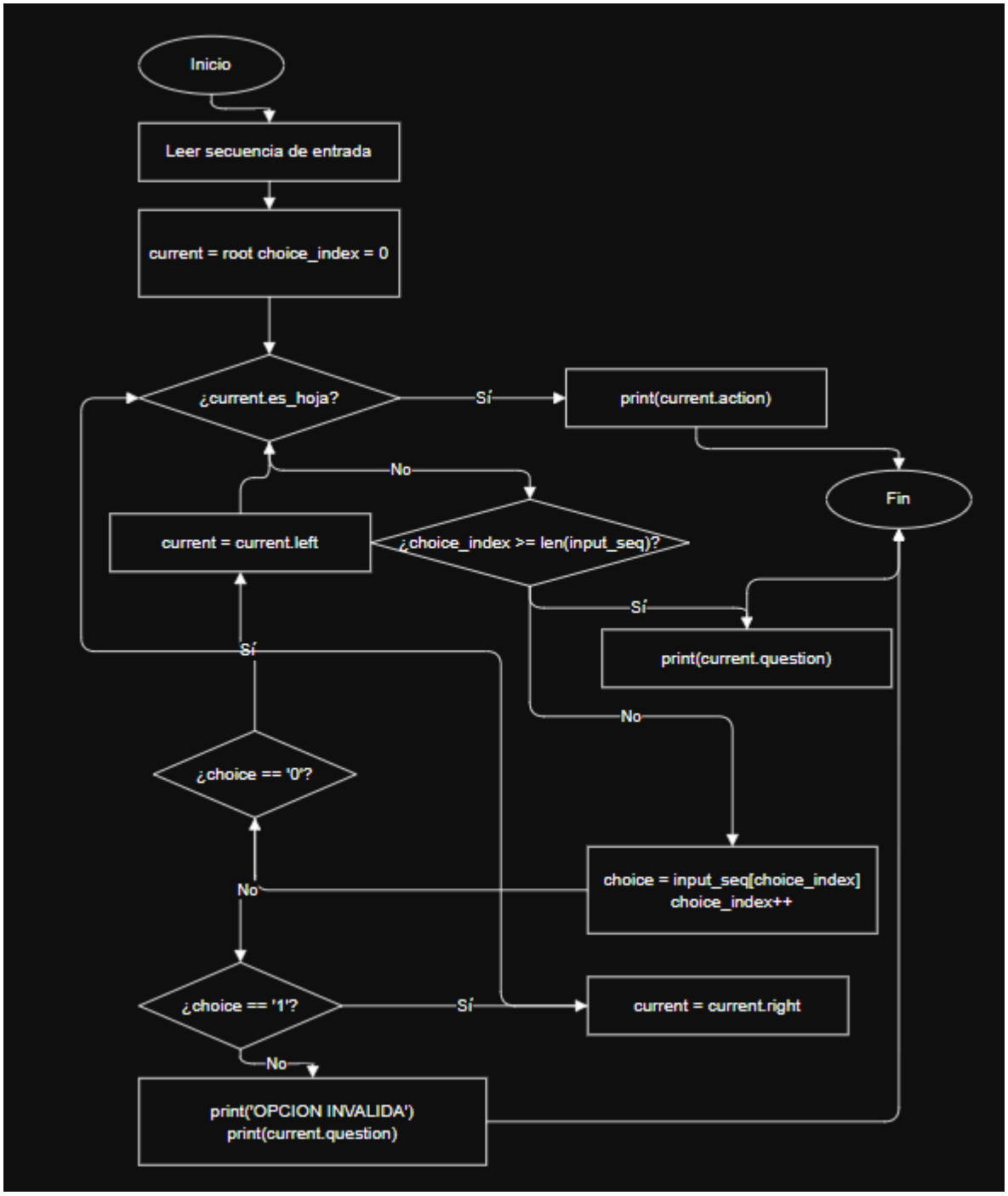
Entradas de ejemplo 2	`1 0`
Salida de ejemplo 2	`Servicio Reprogramado`

Entradas de ejemplo 3	`1 1`
Salida de ejemplo 3	`Servicio Cancelado`

Casos de prueba (ocultos)

Caso prueba	Entrada	Salida esperada
1	2	OPCION INVALIDA ¿Desea confirmar, reprogramar o cancelar su servicio? (0: Confirmar, 1: Reprogramar/Cancelar)
2	1 2	OPCION INVALIDA ¿Desea reprogramar o cancelar su servicio? (0: Reprogramar, 1: Cancelar)
3	1	¿Desea reprogramar o cancelar su servicio? (0: Reprogramar, 1: Cancelar)

Diseño: diagrama de flujo



Desarrollo: Implementación del Código Python

Código Python :

```
# Full trees (Bynary Expressions)

class Node:

    def __init__(self, question=None, action=None):

        self.question = question
```

```
        self.action = action # Solo si es un nodo hoja
        self.left = None
        self.right = None
        self.is_leaf = (action is not None)


class ArbolBinarioCompleto:
    def __init__(self):
        # Define la estructura del árbol de decisión de WhatsApp
        # Nodo raíz: Pregunta inicial
        self.root = Node("¿Desea confirmar, reprogramar o cancelar su servicio?
(0: Confirmar, 1: Reprogramar/Cancelar)")


        # Nivel 1: Opciones principales
        confirm_node = Node(action="Servicio Confirmado")
        reprogram_cancel_node = Node("¿Desea reprogramar o cancelar su servicio?
(0: Reprogramar, 1: Cancelar)")


        self.root.left = confirm_node
        self.root.right = reprogram_cancel_node


        # Nivel 2: Opciones de reprogramar/cancelar
        reprogram_node = Node(action="Servicio Reprogramado")
        cancel_node = Node(action="Servicio Cancelado")


        reprogram_cancel_node.left = reprogram_node
        reprogram_cancel_node.right = cancel_node


    def ejecutar(self):
```

```
        current_node = self.root

        try:
            input_sequence = input().strip().split()
        except EOFError:
            return

        choice_index = 0

        while current_node and not current_node.is_leaf:
            if choice_index >= len(input_sequence):
                # No hay más entradas, pero no se ha llegado a una hoja
                print(current_node.question) # Repetir la pregunta si no hay más entrada
                break

            choice = input_sequence[choice_index]

            if choice == '0':
                current_node = current_node.left
            elif choice == '1':
                current_node = current_node.right
            else:
                print("OPCION INVALIDA")
                print(current_node.question) # Repetir la pregunta
                # No avanzar en el índice de elección para que el usuario pueda reintentar
                # En este ejercicio, la entrada es una secuencia fija, así que esto podría ser un error fatal.
```

```
                # Para el propósito del ejercicio, si la entrada es inválida, se
asume que la secuencia es incorrecta.
                return # Terminar si la opción es inválida en una secuencia
predefinida

            choice_index += 1

        if current_node and current_node.is_leaf:
            print(current_node.action)
        elif current_node:
            # Si se salió del bucle porque no hay más entrada y no es hoja
            pass # Ya se imprimió la pregunta si era necesario
        else:
            # Esto no debería ocurrir si el árbol está bien formado y la entrada
es válida
            pass

if __name__ == "__main__":
    ArbolBinarioCompleto().ejecutar()
```