Instituto Tecnológico de Costa Rica

Área de Ingeniería en Computadores

Lenguajes, Compiladores e Intérpretes (CE 3104)

Tarea II – CallCenterLog

Integrantes:

Meibel Ceciliano Picado, carné 2020023333

Kevin Lobo Juárez, carné 2020087823

Nicol Otárola Porras, carné 2021117282

Profesor:

Marco Rivera Meneses

Fecha de entrega:

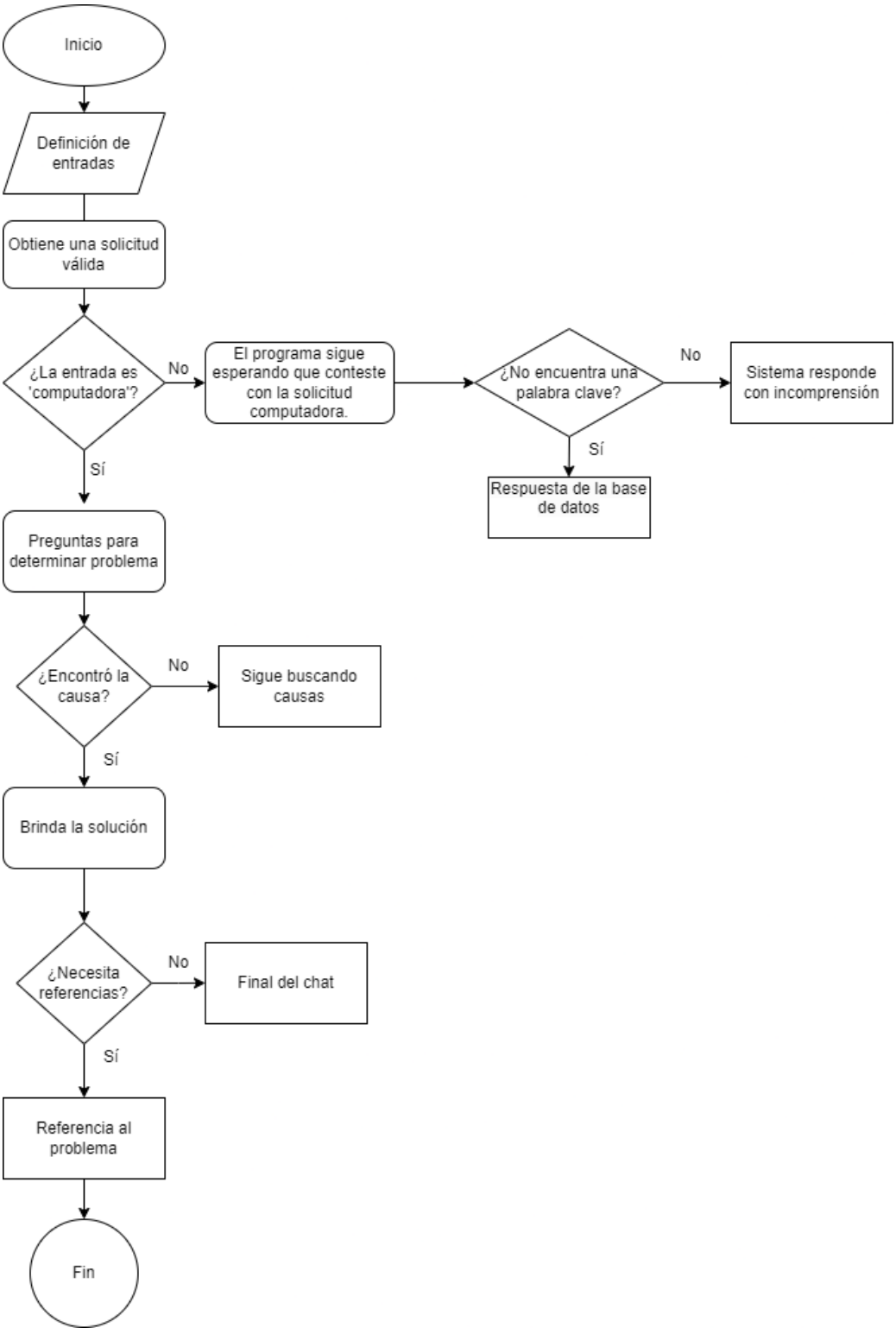
27 de marzo

I Semestre, 2023

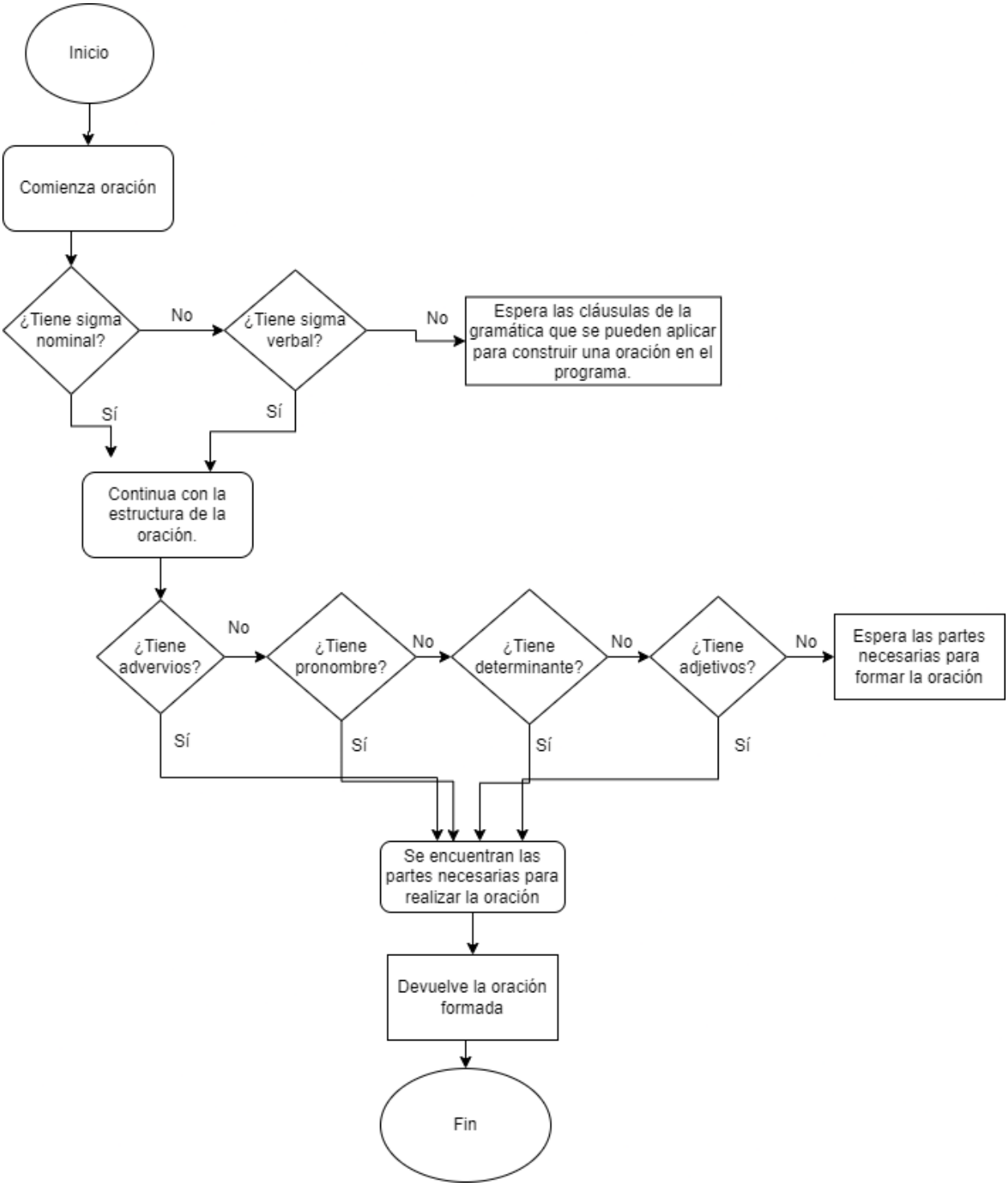
Tabla de Contenidos

Descripción detallada de los algoritmos desarrollados

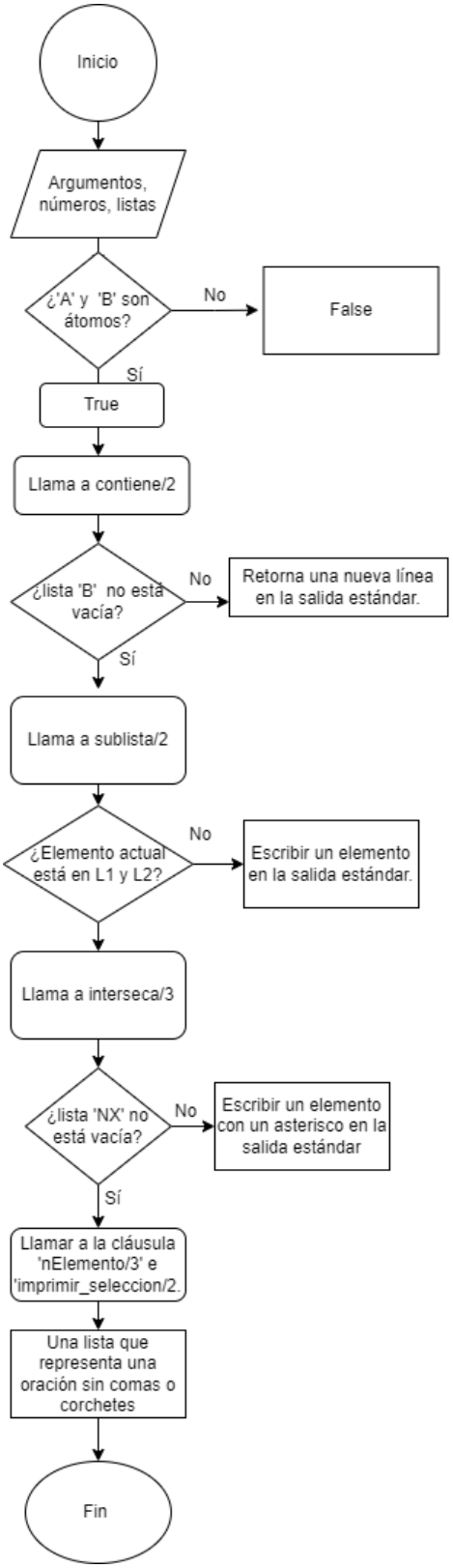
Base de Datos



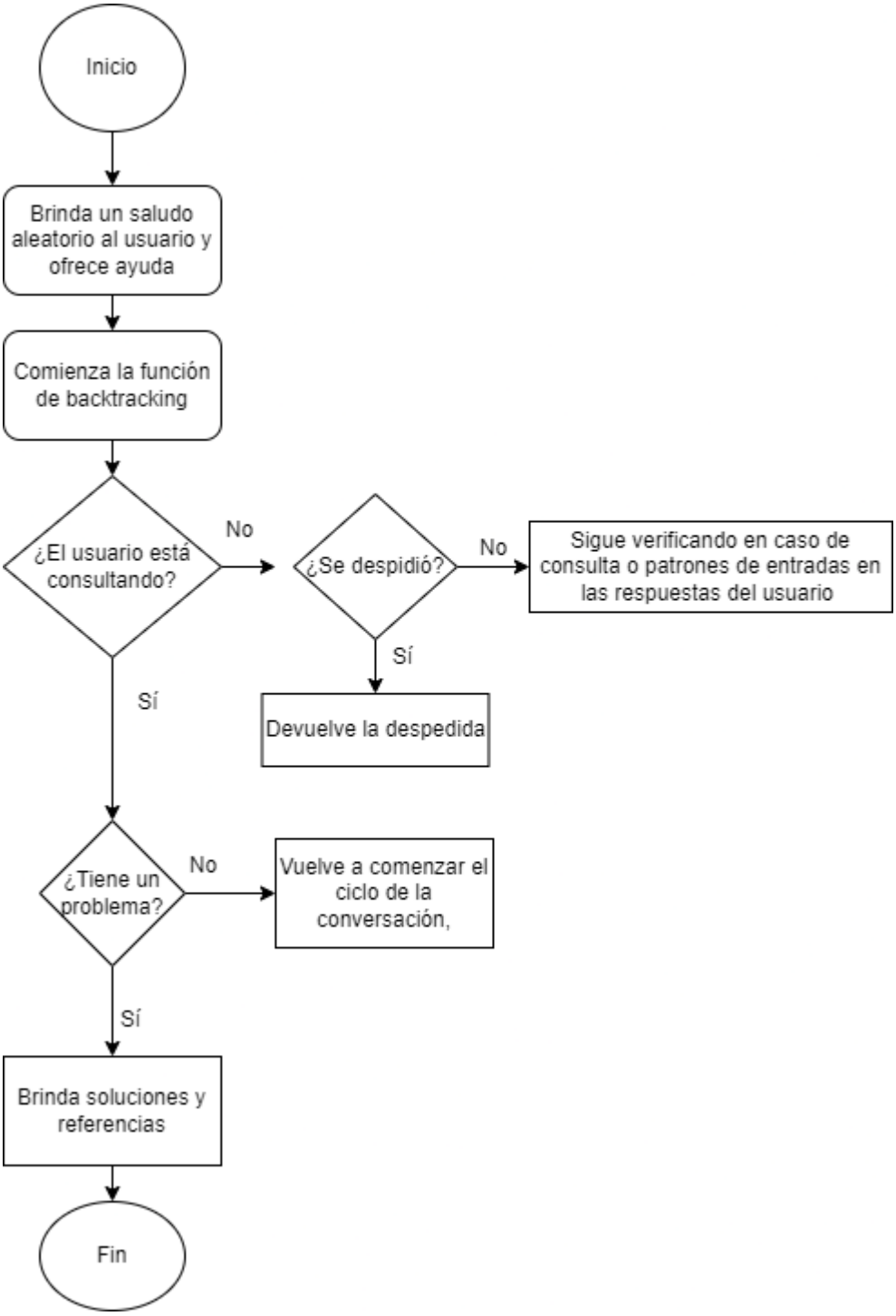
Bnf


Inicio
Comienza oración
¿Tiene sigma nominal?
No
¿Tiene sigma verbal?
No
Espera las cláusulas de la gramática que se pueden aplicar para construir una oración en el programa.
Sí
Sí
Continua con la estructura de la oración.
¿Tiene advervios?
No
¿Tiene pronombre?
No
¿Tiene determinante?
No
¿Tiene adjetivos?
No
Espera las partes necesarias para formar la oración
Sí
Sí
Sí
Sí
Se encuentran las partes necesarias para realizar la oración
Devuelve la oración formada
Fin

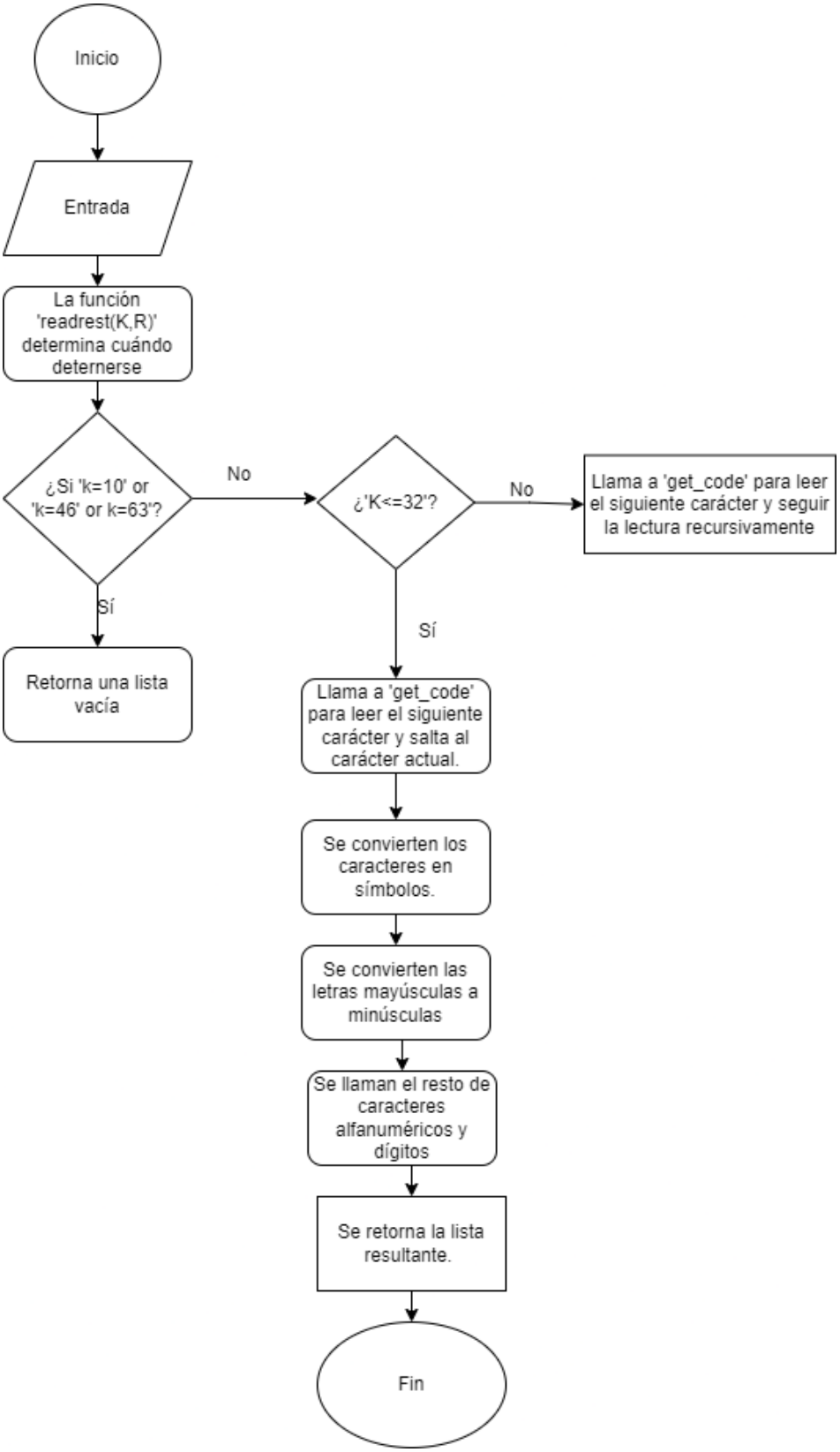
Funciones


Inicio
Argumentos, números, listas
¿'A' y 'B' son átomos?
No
False
Sí
True
Llama a contiene/2
¿lista 'B' no está vacía?
No
Retorna una nueva línea en la salida estándar.
Sí
Llama a sublista/2
¿Elemento actual está en L1 y L2?
No
Escribir un elemento en la salida estándar.
Llama a interseca/3
¿lista 'NX' no está vacía?
No
Escribir un elemento con un asterisco en la salida estándar
Sí
Llamar a la cláusula 'nElemento/3' e 'imprimir_seleccion/2.
Una lista que representa una oración sin comas o corchetes
Fin

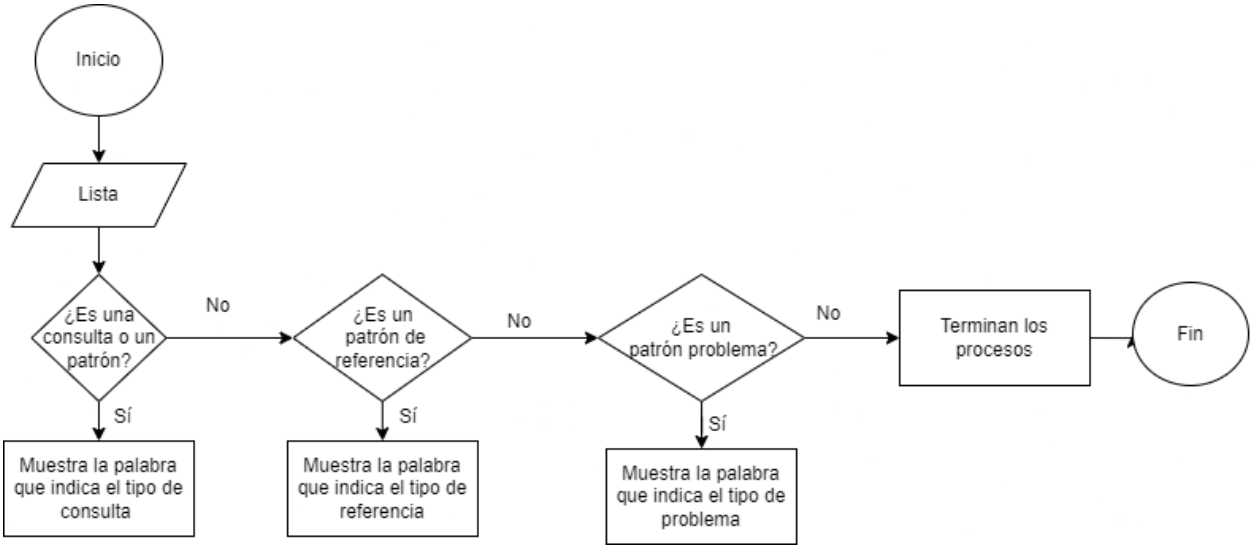
Main


Inicio
Brinda un saludo aleatorio al usuario y ofrece ayuda
Comienza la función de backtracking
¿El usuario está consultando?
No
¿Se despidió?
No
Sigue verificando en caso de consulta o patrones de entradas en las respuestas del usuario
Sí
Devuelve la despedida
Sí
¿Tiene un problema?
No
Vuelve a comenzar el ciclo de la conversación,
Sí
Brinda soluciones y referencias
Fin

Reader


Inicio
Entrada
La función 'readrest(K,R)' determina cuándo deternerse
¿Si 'k=10' or 'k=46' or k=63'?
No
¿'K<=32'?
No
Llama a 'get_code' para leer el siguiente carácter y seguir la lectura recursivamente
Sí
Retorna una lista vacía
Sí
Llama a 'get_code' para leer el siguiente carácter y salta al carácter actual.
Se convierten los caracteres en símbolos.
Se convierten las letras mayúsculas a minúsculas
Se llaman el resto de caracteres alfanuméricos y dígitos
Se retorna la lista resultante.
Fin

Sistema experto



Descripción de los hechos y reglas implementadas.

Base de datos

respuestas: almacenan respuestas para diferentes tipos de preguntas que el usuario puede hacer al asistente, tales como saludos, despedidas, agradecimientos, problemas que se pueden resolver, entre otras. También se almacenan respuestas para casos específicos como la conexión de una computadora o un dispositivo.

causas: almacena una lista de posibles causas para los problemas relacionados con las computadoras y se utiliza para proporcionar información adicional al usuario.

preguntas: almacena una lista de preguntas que el asistente puede hacer al usuario para recopilar más información sobre el problema que tiene el usuario. Cada pregunta se enumera con un identificador único que se utiliza para identificar la causa del problema en la base de datos "causas_db".

referencias: almacena una lista de enlaces web que pueden proporcionar información adicional sobre la solución a un problema específico. Cada referencia se enumera con un identificador único que se utiliza para identificar la solución en la base de datos "respuestas".

BNF

oracion: se encarga de definir las diferentes formas en que una oración puede ser construida. Hay varias reglas definidas que combinan sintagmas nominales y verbales, así como otros elementos como pronombres, adverbios y determinantes.

sintagma_nominal/ sintagma_verbal: son subrutinas que ayudan a construir los sintagmas nominales y verbales respectivamente. Estas funciones reciben como entrada ciertos parámetros, como el número, género y persona del sintagma, así como una lista de palabras que representan el sintagma en sí.

adjetivo: define una lista de adjetivos que pueden ser utilizados para describir sustantivos en las oraciones construidas por el sistema de Call Center.

Funciones

contiene(String, SubString): Esta función tiene como función verificar si un SubString se encuentra dentro de un String. Primero verifica si ambos argumentos son átomos, si es así, convierte los átomos en listas de caracteres utilizando name/2, y luego aplica la misma función con la lista. Si A es un átomo y B es una lista, convierte A a una lista de caracteres y aplica la función. Por último, si B es una sublista de A, devuelve true.

mostrar_lista/1: Esta función tiene como función imprimir una lista en la salida estándar. Imprime cada elemento de la lista seguido de un espacio. Si la lista está vacía, imprime una nueva línea.

mostrar_listacero/2: Esta función tiene como función imprimir los elementos de una lista como una lista no ordenada. Imprime cada elemento de la lista usando imprimir_lista/1 y lo precede con el carácter * y un tabulador. Imprime tantos elementos como se indique en el segundo argumento.

mostrarr_seleccion/2: Esta función tiene como función imprimir una selección de elementos de una lista. Imprime los elementos de la lista cuyos índices se encuentran en el segundo argumento.

Para cada índice, utiliza nElemento/3 para obtener el elemento correspondiente y lo imprime utilizando imprimir_lista/1.

intersecar(conjunto 1, conjunto2, sub_conjunto): Esta función tiene como función encontrar la intersección entre dos conjuntos (Set1 y Set2) y devolver los elementos que se encuentran en ambos conjuntos en SubSet. Recorre Set1 y verifica si cada elemento está en Set2 utilizando member/2, si es así, lo agrega a SubSet.

n_elemento(list, N, elemento): Esta función tiene como función devolver el enésimo elemento de una lista. Verifica si el número N es igual a 1 y devuelve el primer elemento de la lista. Si N es mayor que 1, se llama recursivamente a nElemento/3 con la cola de la lista y N-1, hasta que N es igual a 1.

sub_lista(sub_lista, lista): Esta función tiene como función verificar si una lista es una sublista de otra. Utiliza append/3 para encontrar dos listas que al ser concatenadas forman Lista, y comprueba si SubLista es la primera de ellas.

sub_conjunto(sub_conjunto, conjunto): Esta función tiene como función verificar si un conjunto es un subconjunto de otro. Utiliza member/2 para verificar si cada elemento de SubSet está en Set. Si todos los elementos de SubSet están en Set, devuelve true. Si SubSet está vacío, devuelve true.

Main

dynamic dispositivo/1, solicitud/1: Establece que dispositivo y solicitud son variables dinámicas que pueden cambiar en tiempo de ejecución.

consultar/0: Inicia una conversación con el bot.

saludar/0: Saluda al usuario con una frase de saludo aleatoria.

conversacion/0: Ciclo de backtracking principal. Se repite hasta que el usuario se despide. Lee las entradas del usuario, genera respuestas y las imprime en la consola.

generar_respuesta/2: Es la función principal que procesa las entradas del usuario y genera respuestas apropiadas.

buscar_saludo/1, buscar_gracias/1, salir/1: Estas son funciones auxiliares que buscan frases específicas en las entradas del usuario y generan respuestas apropiadas.

patronConsulta/2, patronReferencia/2, patronProblema/2: Son funciones auxiliares que se utilizan para identificar patrones en las entradas del usuario y determinar qué tipo de consulta está haciendo el usuario.

verificar_dispositivo/1, obtener_dispositivo/0: Son funciones auxiliares que se utilizan para determinar el dispositivo sobre el que el usuario está haciendo la consulta.

resolver_consulta/0: Es una función auxiliar que se utiliza para procesar la consulta del usuario y generar una respuesta apropiada.

es_causa/2, es_caso_especial/2: Son funciones auxiliares que se utilizan para determinar si el problema mencionado por el usuario tiene una solución específica en la base de datos del chatbot.

brindar_solucion/1: Es una función auxiliar que se utiliza para buscar soluciones en la base de datos del chatbot y generar una respuesta apropiada.

brindar_referencia/1, brindar_referencias/0: Son funciones auxiliares que se utilizan para buscar referencias en la base de datos del chatbot y generar una respuesta apropiada.

Reading

readin(S): se encarga de recibir una oración ingresada por el usuario y retornar una lista de palabras y caracteres especiales.

read_in(P): que utiliza otras funciones para leer cada caracter de la oración ingresada por el usuario y convertirlos en una lista de palabras.

initread(L): se encarga de leer cada caracter y convertirlo según su valor ASCII.

readrest(K,R): es una función recursiva que lee de manera recursiva el resto de la oración hasta encontrar un simbolo que indique el final de la oración .

words(P,L,[]): es llamada por read_in(P) para convertir la lista de caracteres obtenida por initread(L) en una lista de palabras.

Sistema experto

modeloConsulta/2: busca patrones que indican que el usuario está realizando una consulta, por ejemplo, si la lista de palabras comienza con "tengo una consulta", "necesito consultar", "realizar una consulta", "hacer una consulta", "necesito conocer" o "consultar", devuelve la palabra siguiente (X) que se encuentra en la lista.

modeloReferencia/2: busca patrones que indican que el usuario está solicitando una referencia, por ejemplo, si la lista de palabras comienza con "tienes una referencia", "tendrás referencias", "puedo conseguir información", "tienes información acerca de" o "necesito conocer", devuelve la palabra siguiente (X) que se encuentra en la lista.

modeloProblema/2: busca patrones que indican que el usuario tiene un problema, por ejemplo, si la lista de palabras contiene "tengo un problema", "presentan un problema", "presenta un problema", "tiene un problema", "tienen un problema", "no funciona", "mal funcionamiento", "dañado" o "no sirve", devuelve la palabra siguiente (X) que se encuentra en la lista

Descripción de la ejemplificación de las estructuras de datos desarrolladas.

Listas

En el lenguaje de programación Prolog es posible hacer uso de listas para la representación de datos, gran parte de la estructura realiza en CallCenterLog necesitó de estas estructuras. Mediante las listas se manipularon colecciones de datos, por ejemplo la lista de elementos detectados en una oración, la cual seguía un sintagma nominal y verbal definido. Todos los elementos de manera individual eran colocados en listas de sustantivos, determinantes, verbos, entre otros. Estos valores se iban concatenando para formar frases u oraciones.

Las listas en Prolog son flexibles y pueden contener cualquier tipo de dato, en algunos casos fue utilizados listas de strings y en otros como enteros. Además se puede analizar una lista por partes analizando la cabeza y resto de ella. Con ayuda de las listas fue posible implementar el uso de patrones/modelos en el sistema experto que detectara palabras claves en las consultas del usuario, de esta manera se consultaba si coincidía con la sintaxis del bnf y reader, o en el bnf.

Finalmente, se usaron funciones predefinidas que operan en listas como append, member y length, mejorando la eficiencia del código. Por ejemplo en el main.pl podrá ver la utilización de estas.

Backtracking

El backtracking es utilizado en el ciclo principal conversacion/0 para permitir que el usuario pueda interactuar con el bot múltiples veces. El ciclo se repite usando el predicado repeat/0, lo que significa que seguirá ejecutando el código en su interior hasta que se satisfaga una condición de salida explícita.

Cada vez que se inicia una conversación, el bot saluda al usuario y luego entra en un ciclo de backtracking. El primer paso del ciclo es leer la entrada del usuario mediante readin/1 y procesarla mediante el predicado generar_respuesta/2. Este predicado verifica la entrada del usuario y determina qué respuesta debe ofrecer el bot. Si la entrada no coincide con ninguno de los patrones conocidos, se brinda una respuesta predeterminada.

Si la entrada del usuario coincide con un patrón conocido, se ejecutará el código correspondiente y se volverá al ciclo de backtracking. En cada iteración, el bot espera a que el usuario ingrese una nueva entrada.

La condición de salida se encuentra en el predicado salir/1, que verifica si el usuario se despide del bot. Si es así, se ejecuta la respuesta de despedida correspondiente y se finaliza el ciclo de backtracking.

Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

- Al momento de solicitar el usuario más de una referencia, CallCenterLog va a mandar las referencias repetidas n cantidad de veces.
- Al momento de ejecutar el archivo main.pl, a veces no se carga el chat a la consola por lo que se necesita de volver a consultar el archivo main.pl.

Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada

- Al momento de llamar varios archivos en una lista, se encontró que los archivos no pueden contener mayúsculas al inicio, ya que Prolog lee los archivos como una variable, por lo que se cambió el nombre en minúscula y con guion bajo, para que puedan ser leídos por Prolog.
- Las tildes, la letra ‘ñ’ y el signo ‘¿’, no se pueden incluir en la base de datos, ya que de lo contrario al momento de realizar las consultas estas no generaban ninguna consulta.
- El orden de los problemas que en algunas pruebas al preguntar por la causa de un problema el programa le brindaba la solución pero de otro distinto a ese, y nos dimos cuenta que la numeración utilizada era incorrecta.
- Que en las referencias si se incluye una palabra no contenida en el bnf o el sistema experto el programa no va a detectar su pregunta porque es en otro formato distinto al del sintagma.

Conclusiones

- Para el desarrollo de esta tarea, se hizo uso de una base de datos, que resulto ser una herramienta muy útil para organizar y almacenar información de manera estructurada. En el código presentado, se utilizan varias bases de datos para almacenar información, como las respuestas a diferentes tipos de consultas, las causas de problemas en dispositivos y las preguntas a hacer para diagnosticarlos, y las referencias para obtener más información al respecto. Esto permite al programa ofrecer soluciones más precisas y útiles a los usuarios que solicitan asistencia técnica.
- El BNF es una herramienta que se utilizó en el trabajo para establecer las estructuras o sintagmas para la creación de oraciones para el Call Center Log, ya que la finalidad de implementar el BNF lo que permite es definir de manera precisa y formal la sintaxis de un lenguaje, lo que a su vez mejora la eficiencia y la confiabilidad del código.
- Las listas permiten almacenar y manipular conjuntos de datos de forma eficiente y estructurada, en este caso, se realizaron operaciones sobre listas, tales como imprimir listas eliminando caracteres especiales, imprimir elementos de una lista como una lista no ordenada, imprimir elementos seleccionados de una lista, determinar si un elemento es una sublista de otra, encontrar el enésimo elemento de una lista, y determinar si una lista es un subconjunto de otra. Estas funciones pueden ser útiles para realizar tareas específicas en Prolog y manipular listas de manera efectiva.

- Para el sistema experto, se utiliza la técnica de backtracking para manejar las diferentes situaciones que pueden surgir en la conversación con el usuario, a partir del backtracking es que se permite el retroceso, que es una técnica algorítmica de resolución de problemas que implica un enfoque de prueba y error para encontrar una solución.
- Prolog es un lenguaje de programación declarativo que se utiliza principalmente para resolver problemas de inteligencia artificial y lógica, por lo que permitió trabajar de manera efectiva en el desarrollo del proyecto.
- Una de las principales ventajas que encontramos de utilizar Prolog es que permite al programador expresar la lógica de un problema de manera clara y concisa, lo que puede ahorrar mucho tiempo y esfuerzo en el proceso de resolución de problemas.

Recomendaciones

- Para el desarrollo de programas de asistencia técnica como el Call Center Log, el uso de bases de datos estructuradas es una herramienta muy útil para organizar y almacenar información de manera eficiente. Además, es importante tomar en cuenta que la base de datos sea fácil de actualizar y mantener, para que la información sea siempre precisa y esté actualizada. También es recomendable tener en cuenta la privacidad y seguridad de los datos almacenados, especialmente si se manejan datos sensibles de los usuarios.
- Estudiar el funcionamiento del BNF, ya que este requiere de ciertos conocimientos técnicos. Por otro lado, si se trabaja en programas de Call Center o Servicio al Cliente, se recomienda que sea implementado ya que genera un mejor rendimiento del proyecto, ya que puede facilitar la comprensión y la colaboración entre los miembros del equipo, ya que permite una definición clara y precisa de los elementos del lenguaje.
- Para el desarrollo de proyectos de programación o análisis de datos una buena práctica es considerar la implementación de listas para almacenar y manipular conjuntos de datos de manera eficiente y estructurada.
- Dedicar tiempo a aprender a trabajar con listas y asegurarse de utilizar las herramientas adecuadas para la tarea en cuestión, ya que la eficiencia en la manipulación de listas depende en gran medida de la implementación correcta de algoritmos y estructuras de datos.

- Hay que considerar que la implementación de backtracking puede tener un impacto significativo en el rendimiento del sistema, especialmente cuando se trata de situaciones complejas con múltiples opciones y soluciones. Por lo tanto, es importante evaluar cuidadosamente el impacto de la técnica en el rendimiento del sistema y buscar formas de optimizar su implementación.
- Implementar Prolog puede ser una opción valiosa para aquellos que trabajan en proyectos de inteligencia artificial y lógica, ya que permite expresar la lógica de un problema de manera clara y concisa, ahorrando tiempo y esfuerzo en el proceso de resolución de problemas. Sin embargo, es importante evaluar cuidadosamente si Prolog es la herramienta adecuada para el problema específico que se está tratando de resolver ya que también puede tener limitaciones en la resolución de otros tipos de problemas.

Bibliografía

Toledo, F., Pacheco, J., & Escrig, T. (2001). El Lenguaje de Programación PROLOG.

Plan de Actividades realizadas por estudiante:

Esta tarea debe realizarse antes de empezar con el proyecto y lo que busca es que se realice un plan, este debe contener al menos: la lista de actividades, responsable y fecha de entrega de la actividad.

CRONOGRAMA			
ACTIVIDAD	**TIEMPO ESTIMADO DE COMPLETITUD**	**RESPONSABLE**	**FECHA DE ENTREGA**
Reunión inicial para definir roles y tareas Se establece el tema en el cual consistirá la base de datos y se acuerda: que Nicol	1 hora	Todos	12/3/2023

trabajará en la Base de Datos, Meibel en BNF y Kevin en el Sistema Experto			
Creación de repositorio	5 minutos	Nicol	
Se crean los documentos donde se realizará el manual de usuario y documentación técnica, ambos con su debido formato.	10 minutos	Meibel	13/3/2023
Se empieza a trabajar en el sistema experto	3 horas	Kevin	20/3/2023
Reunión #2 para conocer los avances de la Tarea 2 y corregir errores que surjan	1 hora	Todos	22/3/2023
Debe estar lista la Base de Datos	3 horas	Nicol	20/3/2023
Se revisa que el formato usado en el Backus-Naur Form sea correcto	1 hora , 30 min	Meibel	24/3/2023
Entrega del BNF	1 hora	Meibel	24/3/2023

Entrega del Sistema Experto	1 hora	Kevin	24/3/2023
Con la Base de Datos y el BNF listo, se enlazan en el Sistema Experto	2 horas	Todos	25/3/2023
Reunión #3 para revisar el funcionamiento de los 3 archivos de código, BNF, Base de Datos y Sistema Experto	1 hora	Todos	25/3/2023
Detalles finales de CallCenterLog	1 hora	Todos	25/3/2023
Creación de manual de usuario	3 horas	Meibel	26/3/2023
Reunión #4 para distribuir las secciones de la documentación	1 hora	Todos	26/3/2023
Se inicia con la elaboración de la documentación técnica	2 horas	Todos	26/3/2023
Terminar de redactar el trabajo escrito	3 horas	Todos	26/3/2023
Entregar la carpeta con todos los	10 minutos	Todos	27/3/2023

entregables de la tarea 2			
Reunión #5 Ultima reunión para discutir lo que se explicará en la defensa del proyecto	1 hora	Todos	28/3/2023

Bitácoras

en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.

Bitácora Meibel Ceciliano

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
13/3/2023	Se diseña el formato que se va a usar para la documentación técnica, y el manual de usuario	1 hora	Completado
15/3/2023	Se inicia una investigación sobre la implementación de las gramáticas de libre	2 horas	Completada

	contexto en códigos de programación		
16/3/2023	Se empieza a trabajar en el archivo bnf colocando la estructura de las oraciones con sintagma verbal y nominal	2 horas	Completado
18/3/2023	Se corrigen unos errores generados en la parte de sintagmas y se inicia agregando los elementos que conforman la oración, en este caso los interrogativos, determinantes, pronombres y adverbios	2 horas, 30 minutos	Completado
20/3/2023	Se continúa agregando los componentes de la oración, como ya la base de datos fue entregada se pueden colocar los sustantivos, adjetivos y verbos que se incluyen en las preguntas, causas y respuestas.	3 horas	Completado
22/3/2023	Se trabaja en un reader o lector que tendrá como función devolver una lista que tiene de	2 horas	Completado

	elementos las palabras de la oración ingresada por el usuario		
23/3/2023	Se continua con el código del Reader	1 hora 30 minutos	Completada
24/3/2023	Se verifica que la sintaxis y sintagmas funcionen correctamente con las reglas y hechos del BNF para subir el archivo al repo y Kevin pueda utilizarlo en el sistema experto	1 hora 30 minutos	Completada
25/3/2023	Este día se trabaja en conjunto con los otros dos compañeros para intentar solucionar unos errores que está tirando el programa y también para enlazar todos los archivos	4 horas	Completada
26/3/2023	Se comienza con la redacción del manual de usuario	2 horas	Completada
26/3/2023	Colaborar en la realización de la documentación técnica	4 horas	Completada

27/3/2023	Se pulen detalles finales para proceder con la entrega de la Tarea 2	2 horas	Completado
28/3/2023	Ultima reunión para preparar la defensa que se hará el jueves 30 de marzo	1 hora	Pendiente

Bitácora Kevin Lobo

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
20/03/2023	Diseño del sistema experto.	1 hora	Completada
22/3/2023	El sistema experto brinda soluciones.	1 hora	Completada
23/3/2023	El sistema experto brinda las causas del problema.	1 hora	Completada
24/3/2023	El sistema experto detecta las preguntas que realiza el usuario para brinda la solución.	1 hora	Completada
26/3/2023	El sistema experto detecta las causas que	1 hora	Completada

	realiza el usuario para brinda la solución.		
26/3/2023	El sistema experto detecta las referencias que realiza el usuario para brinda la solución.	1 hora	Completada
26/3/2023	Se trabaja en algunos errores al intentar enlazar los archivos del código El sistema experto detecta las referencias que realiza el usuario para brinda la solución.	1 hora	Completada
26/3/2023	Junto con Meibel se hacen unas pruebas del funcionamiento de CallCenterLog para finalizar esa parte y agregar los ejemplos al manual de usuario	6 horas	Completada
27/3/2023	Se añaden a la documentación externa los problemas encontrados durante la elaboración de la tarea, tanto con solución, como aquellos sin solucionar.	40 minutos	Completada

27/3/2023	Se comenta el código fuente que hacía falta, como el main y el sistema experto	50 minutos	Completada
28/3/2023	Reunión final para preparación de la defensa	1 hora	Pendiente

Bitácora Nicol Otárola

Diario de Trabajo			
Fecha	**Actividades realizadas**	**Duración**	**Estado**
13/03/2023	Creación del repositorio.	5 minutos	Completado
14/03/2023	Se comienza con una investigación para determinar la estructura que va a llevar la base de datos.	2 horas	Completado
15/03/2023	Redactar las causas y asociarlas con las preguntas, solución y las referencias.	1 hora	Completado
16/03/2023	Crear la base de datos y establecer las relaciones	3 horas	Completado

	correspondientes, es decir, definir el problema y con base a eso, establecer las preguntas		
17/03/2023	Reestructuración de la base de datos.	2 horas	Completado
18/03/2023	Definir las entradas en la base de datos, además, de tener respuestas de saludos, despedidas, agradecimiento, final la oración y en caso de que el sistema no entienda	2 horas	Completado
26/03/2023	Se trabaja en la parte escrita, específicamente en el diagrama de flujo.	3 horas	Completado
27/09/2023	Establecer las conclusiones y recomendaciones, trabajar en descripción de hechos y la descripción de la estructura desarrollada (backtracking).	4 hora	Completado
28/3/2023	Reunión final para preparación de la defensa	1 hora	Pendiente

Evidencias de reuniones

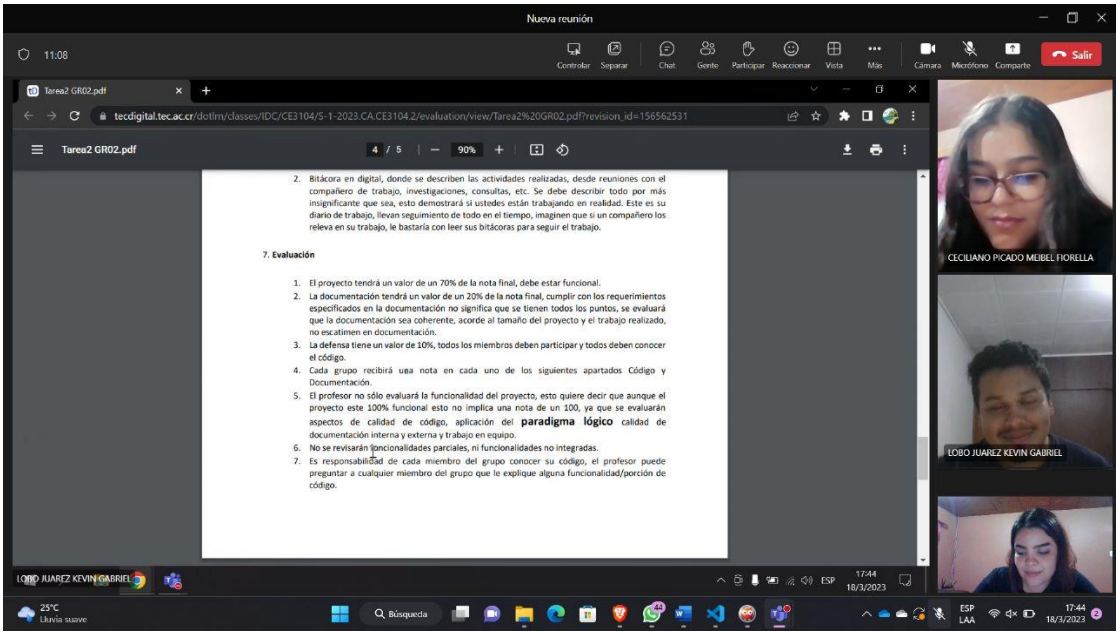


Figura 1: Reunión inicial para definir roles y tareas, además revisar las especificaciones del proyecto

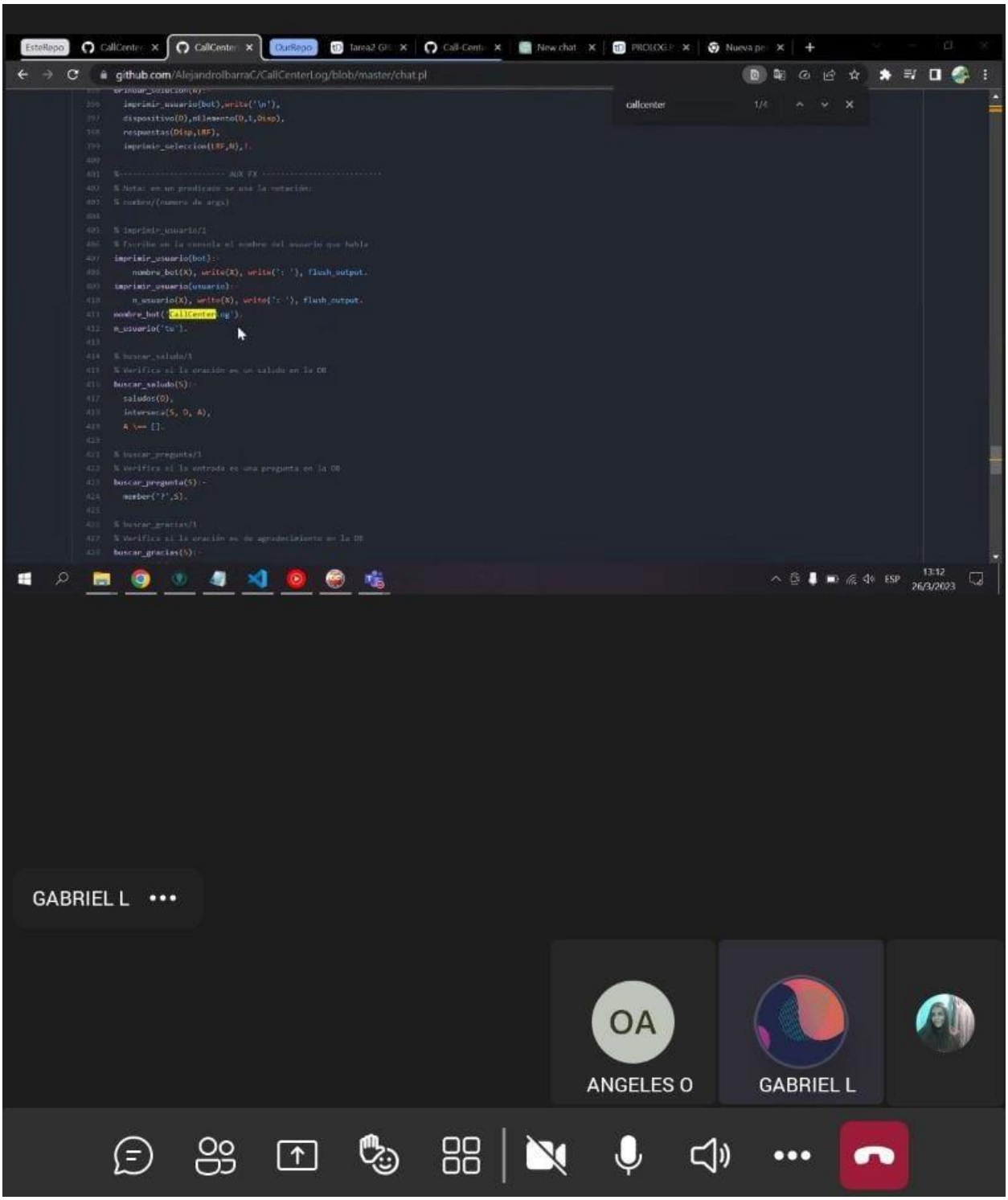


Figura 2: Reunión #2 para conocer los avances de la Tarea 2 y corregir errores que surjan

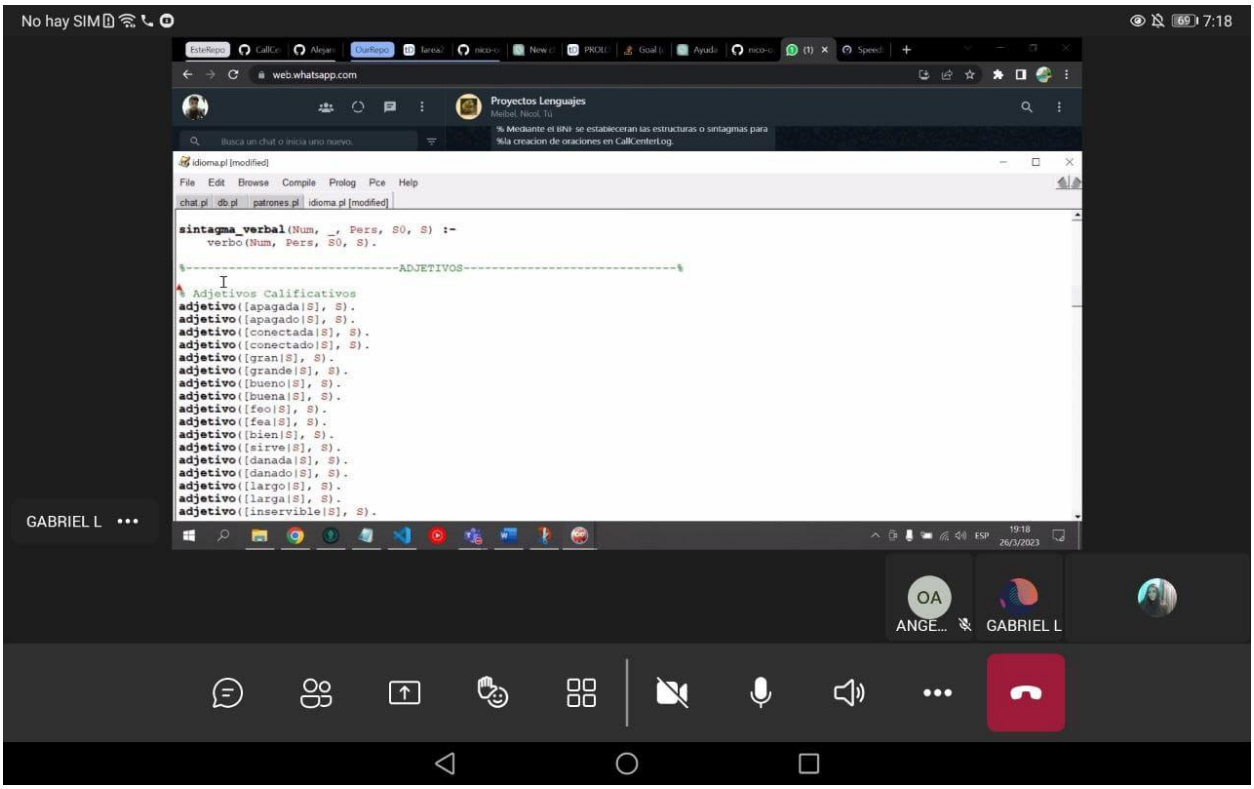


Figura 3: Reunión #3 Se revisa el funcionamiento de todos los archivos de CallCenterLog

Enlace repositorio Github:

Para llevar un trabajo en línea y simultaneo se decide crear un repositorio en Github donde cada integrante subiera sus avances, aquí también se puede visualizar los commits frecuentes demostrando un trabajo constante.

https://github.com/nico-op/Call_Center_Log.git