CE1106 - TransLog
Paradigmas de Programación
Escuela de Ingeniería en Computadores
Estudiantes:
Eduardo José Canessa Quesada
Luis Felipe Chaves Mena
Deiler Morera Valverde
Fecha: Octubre 2025

A continuación se documentan aspectos del proyecto `CE1106 - TranLog`, una aplicación que se encarga de traducir de inglés a español y de español a inglés, esto gracias al paradigma lógica que se emplea por medio del lenguaje de programación **Prolog**.

1. Descripción de Reglas y hechos.

En esta sección se analizarán los hechos y las reglas empleadas para el desarrollo adecuado del proyecto.

1.1. Hechos, su relación con la base de datos y su estructura.

La base de datos del traductor se construye a partir de **hechos**, los cuales representan de forma explícita la información lingüística necesaria para traducir adecuadamente entre el español y el inglés. Cada hecho describe una categoría gramatical distinta, junto con los atributos que permiten identificar sus propiedades morfológicas y su equivalente en el otro idioma. A continuación, se detallan los principales tipos de hechos utilizados.

1.1.1. Hechos de interjección

Las interjecciones son expresiones que transmiten emociones, saludos o reacciones espontáneas. Su hecho contiene únicamente la forma en español y su equivalente en inglés, ya que no presentan variaciones morfológicas. Este tipo de hecho es utilizado por el traductor para reconocer expresiones completas que no se alteran según número, persona o género.

Estructura general:

```
interjeccion(Interjeccion_español, Interjeccion_inglés).
```

1.1.2. Hechos de sustantivo

Los sustantivos representan entidades, personas, lugares o ideas. El hecho asociado incluye tanto la forma española como su traducción, además de tres atributos gramaticales: *persona*, *número* y *género*. De esta manera, el sistema puede identificar si un sustantivo se refiere a la primera, segunda o tercera persona; si es singular o plural; y si pertenece al género masculino, femenino o neutro. Esta información es esencial para mantener la concordancia entre sustantivos, artículos y adjetivos durante la traducción.

Estructura general:

```
sustantivo(Sustantivo_español, Sustantivo_inglés, Persona, Número, Género).
```

1.1.3. Hechos de verbo

Los verbos son uno de los elementos más complejos de la base de datos, ya que incluyen las distintas formas conjugadas tanto en español como en inglés. El hecho correspondiente asocia un infinitivo en ambos idiomas y dos listas: una con las formas verbales en español y otra con sus equivalentes en inglés. Cada posición de la lista representa una persona gramatical, permitiendo al sistema determinar la forma verbal correcta según el sujeto que acompaña la oración.

Estructura general:

```
verbo(Infinitivo_español, Infinitivo_inglés, [Formas_español], [Formas_inglés]).
```

1.1.4. Hechos de adjetivo

Los adjetivos describen cualidades o características de los sustantivos. El hecho que los define contiene el adjetivo en español, su traducción al inglés y los atributos *género* y *número*. Estos parámetros permiten al traductor mantener la concordancia gramatical con el sustantivo que acompaña, asegurando una traducción coherente y estructuralmente correcta.

Estructura general:

```
adjetivo(Adjetivo_español, Adjetivo_inglés, Género, Número).
```

1.1.5. Hechos de determinante

Los determinantes son palabras que acompañan al sustantivo para precisar su referencia. El hecho incluye la forma española e inglesa, junto con su número, género y tipo (por ejemplo, artículo definido, posesivo o demostrativo). Esta clasificación permite que el traductor distinga las funciones específicas de cada determinante y seleccione el artículo o adjetivo posesivo apropiado en el idioma de destino.

Estructura general:

```
determinante(Determinante_español, Determinante_inglés, Número, Género, Tipo).
```

1.1.6. Hechos de pronombre

Los pronombres sustituyen a los sustantivos y, por lo tanto, requieren información sobre la persona y el número gramatical. El hecho registra la forma del pronombre en ambos idiomas, además de los atributos que permiten determinar su función en la oración. De este modo, el sistema puede identificar si un pronombre actúa como sujeto, objeto o posesivo, y ajustar la traducción de acuerdo con el contexto gramatical.

Estructura general:

```
pronombre(Pronombre_español, Pronombre_inglés, Persona, Número).
```

1.1.7. Hechos de preposición

Las preposiciones son invariables y expresan relaciones de lugar, tiempo o propósito entre elementos de la oración. El hecho correspondiente solo contiene las versiones en español e inglés, ya que no poseen variaciones de género, número ni persona. Estos hechos permiten que el sistema reconozca las conexiones sintácticas y preserve la estructura relacional en la traducción.

Estructura general:

```
preposicion(Preposicion_español, Preposicion_inglés).
```

1.1.8. Hechos de adverbio

Los adverbios modifican verbos, adjetivos o incluso otras expresiones adverbiales. El hecho que los representa almacena la forma en español y su traducción directa al inglés. Dado que los adverbios son invariables, no requieren información adicional de número o género. Su presencia en la base de datos permite al sistema mantener el significado y la intensidad de las acciones o cualidades descritas.

Estructura general:

```
adverbio(Adverbio_español, Adverbio_inglés).
```

1.1.9. Hechos de conjunción

Las conjunciones son palabras que enlazan oraciones o términos con igual función gramatical. El hecho correspondiente asocia la conjunción española con su equivalente inglesa. Su papel en la base de datos es garantizar la correcta unión de las proposiciones en la traducción, manteniendo la coherencia lógica y sintáctica del texto.

Estructura general:

```
conjuncion(Conjuncion_español, Conjuncion_inglés).
```

1.2. Reglas implementadas

El archivo `BNF.pl` constituye la capa de interfaz principal del sistema de traducción. En él se definen las reglas encargadas de gestionar la interacción con el usuario, el control del flujo de ejecución y la comunicación con los módulos lógicos del traductor. Estas reglas no realizan directamente la traducción, sino que coordinan los procesos de entrada, tokenización y salida de resultados.

1.2.1. Reglas de inicialización

Las reglas `tranlog_ei/0` y `tranlog_ie/0` actúan como puntos de entrada al sistema. Cada una configura el modo de traducción (español → inglés o inglés → español) e inicia un ciclo de interacción con el usuario. Estas reglas también presentan los mensajes iniciales y las instrucciones en pantalla, asegurando una interfaz textual intuitiva.

```
tranlog_ei/0   % Activa el modo Español -> Inglés
tranlog_ie/0   % Activa el modo Inglés -> Español
```

1.2.2. Reglas de control de flujo

El ciclo de ejecución está gestionado por la regla `iniciar_modo_traduccion/1`, la cual mantiene el sistema en funcionamiento continuo hasta que el usuario introduzca una línea vacía. Esta regla se encarga de mostrar el indicador de idioma correspondiente, leer la entrada del usuario y enviarla al módulo de procesamiento de traducción.

El flujo general del sistema puede describirse del siguiente modo:

1. Mostrar el prompt correspondiente (`mostrar_prompt/1`).
2. Leer la línea ingresada (`leer_entrada/1`).
3. Procesar la línea recibida según el modo activo (`procesar_entrada_traduccion/2`).

1.2.3. Reglas de procesamiento

La regla `procesar_entrada_traduccion/2` representa el núcleo operativo de este módulo. Recibe el texto introducido, lo convierte en una lista de tokens y delega la traducción a los predicados definidos en el módulo lógico (`Logic.pl`). Posteriormente, las reglas de salida se encargan de reconstruir la oración traducida y mostrarla al usuario.

Estas reglas se apoyan en predicados auxiliares tales como:

- `tokenizar_entrada/2`: divide el texto en unidades léxicas.
- `dividir_por_delimitadores/2`: separa el texto en oraciones según signos de puntuación.
- `traducir_lista_oraciones/3`: aplica la lógica de traducción a cada oración individual.

Cada uno de estos predicados está definido o extendido en el archivo `Logic.pl`, el cual contiene las reglas gramaticales y semánticas del sistema.

1.2.4. Reglas de salida y reconstrucción

Finalmente, las reglas `traduccion_completa/2` y `unir_oraciones_traducidas/2` se encargan de reconstruir las oraciones traducidas en un texto continuo, preservando la puntuación y el orden original. Estas reglas aseguran que la salida mostrada al usuario sea natural y legible, evitando errores comunes en la unión de tokens o en la disposición de los signos.

```
traduccion_completa(OracionesTraducidas, Modo)
unir_oraciones_traducidas(OracionesTraducidas, TextoCompleto)
```

Relación con el módulo lógico

Este módulo depende directamente de las reglas definidas en `Logic.pl`, donde se encuentra la lógica de traducción propiamente dicha: búsqueda en la base de datos, correspondencia entre categorías gramaticales, y generación de equivalencias lingüísticas. En conjunto, ambos módulos conforman una arquitectura de dos capas:

- **Capa de control (BNF.pl):** maneja entrada, salida y flujo de ejecución.
- **Capa lógica (Logic.pl):** contiene las reglas lingüísticas y el acceso a la base de datos léxica.

Esta separación permite mantener el sistema modular, facilitando la ampliación del vocabulario o la modificación del motor lógico sin afectar la interfaz de usuario.

2. Estructuras Definidas en parte lógica

El archivo `Logic.pl` constituye el núcleo lógico del traductor, responsable de definir las estructuras de datos necesarias para representar la información lingüística y morfosintáctica del texto procesado. Estas estructuras sirven de puente entre la base de datos léxica (`BD.pl`) y los procesos de análisis y traducción, permitiendo una organización clara, jerárquica y semánticamente coherente de la información.

2.1. Representación de Tokens

El elemento básico de toda oración es el *token*. Cada token representa una unidad mínima del texto, como una palabra, un signo de puntuación o un símbolo. El sistema maneja los tokens dentro de listas ordenadas, conservando la secuencia original del texto ingresado por el usuario.

- **Ejemplo de lista de tokens:**

$$Tokens = [\texttt{'la'}, \texttt{'niña'}, \texttt{'es'}, \texttt{'grande'}, \texttt{'.'}]$$

- **Tipo de dato:** Lista de átomos o cadenas.
- **Propósito:** Servir como punto de partida para la identificación gramatical de cada elemento y su posterior clasificación.

2.2. Estructura de Oración

Una oración se define como una secuencia ordenada de tokens que posee sentido gramatical completo. En el sistema, cada oración se representa internamente mediante una lista de tokens. A su vez, múltiples oraciones pueden agruparse dentro de una lista mayor cuando el texto de entrada contiene varios enunciados.

$$Oracion = [Token_1, Token_2, \ldots, Token_n]$$

$$Oraciones = [Oracion_1, Oracion_2, \ldots, Oracion_m]$$

Esta estructura permite segmentar el texto en unidades lingüísticas manejables y facilita el análisis individual de cada oración.

2.3. Estructura Léxica Interna

Cada palabra reconocida en el texto es vinculada con su correspondiente definición en la base de datos léxica (`BD.pl`). Estas definiciones se representan mediante hechos de la forma:

$$tipo(Palabra, Traduccion, Atributo_1, Atributo_2, \ldots)$$

Dependiendo de la categoría gramatical, cada tipo de palabra presenta una estructura particular. Esto se explica en la seccion de hechos que es donde se definene las estructuras generales.

Estas estructuras proporcionan una representación semántica rica, permitiendo identificar las propiedades morfológicas y sintácticas necesarias para establecer correspondencias entre idiomas.

2.4. Estructura de Concordancia

Además de las estructuras léxicas, el archivo `Logic.pl` define estructuras lógicas destinadas a verificar la *concordancia gramatical* entre los elementos de una oración. Estas estructuras adoptan la forma de predicados relacionales que asocian sujetos, verbos, complementos y sus atributos comunes.

$$concordancia(Sujeto, Verbo, Atributos)$$

$$relacion(Sujeto, Predicado, Complemento)$$

El propósito de estas estructuras es garantizar que los elementos coincidan en género, número y persona, manteniendo la corrección gramatical tanto en el idioma original como en el traducido.

2.5. Estructura de Oraciones Traducidas

Una vez realizada la traducción de cada oración, el sistema almacena temporalmente los resultados en una lista estructurada de oraciones traducidas:

$$OracionesTraducidas = [OracionTraducida_1, OracionTraducida_2, \ldots, OracionTraducida_m]$$

Posteriormente, todas las oraciones traducidas son concatenadas mediante un procedimiento de unión, generando una salida textual coherente:

$$TraduccionCompleta = \texttt{unir\_oraciones\_traducidas(OracionesTraducidas, TextoFinal)}$$

Esta representación facilita la presentación de resultados en un formato natural y legible, respetando el orden lógico y gramatical del texto.

Cada nivel de esta jerarquía mantiene una relación directa con el siguiente, lo que asegura una representación coherente del flujo de información desde el texto original hasta la traducción final. Estas estructuras constituyen, por tanto, la base sobre la cual se construyen las reglas de análisis y los mecanismos de traducción del sistema.

3. Descripcion detallada de algoritmos

En esta seccion se describe de forma detallada la logica y el flujo de nuestro traductor bidireccional Espanol-Ingles / Ingles-Espanol. Se hace especial enfasis en los procesos de tokenizacion, analisis de sintagmas, conjugacion verbal y traduccion contextual.

3.1. display_results/3

Este predicado se encarga de mostrar al usuario la entrada original, los tokens obtenidos y la traduccion final segun el modo de traduccion seleccionado (Español a Ingles o Ingles a Español).

```
% Caso Espa ol a Ingles
display_results(Input, Tokens, ei) :-
    % Muestra la entrada original
    format('Entrada: ~w~n', [Input]),
    % Muestra los tokens generados por la tokenizacion
    format('Tokens: ~w~n', [Tokens]),
    % Llama a la funcion que traduce los tokens Espa ol -> Ingles
    translate_tokens_ei(Tokens, Translation),
    % Convierte la lista de palabras traducidas en un unico atomo separado por espacios
    atomic_list_concat(Translation, ' ', Text),
    % Muestra la traduccion final
    format('Traduccion: ~w~n', [Text]).

% Caso Ingles a Espa ol
display_results(Input, Tokens, ie) :-
    format('Input: ~w~n', [Input]),
    format('Tokens: ~w~n', [Tokens]),
    translate_tokens_ie(Tokens, Translation),
    atomic_list_concat(Translation, ' ', Text),
    format('Translation: ~w~n', [Text]).
```

Explicacion: Este predicado es la interfaz principal de salida. Primero muestra lo que el usuario ingreso, luego los tokens que se generaron al separar palabras y signos de puntuacion. Posteriormente llama a la funcion de traduccion segun el modo y finalmente concatena las palabras traducidas en una sola cadena.

3.2. translate_tokens_ei/2 y translate_tokens_ie/2

Estos predicados se encargan de traducir listas de tokens, aplicando varias estrategias:

- Se prioriza la traduccion de frases de dos palabras (por ejemplo interjecciones o expresiones idiomaticas)
- Si no se encuentra la frase, se intenta traducir palabra por palabra
- Si una palabra no tiene traduccion, se deja tal cual

```
% Traduccion Espa ol -> Ingles
translate_tokens_ei([], []).  % Caso base, lista vacia
translate_tokens_ei([H1,H2|T], [Eng|Rest]) :-
    % Intenta concatenar dos palabras y buscar traduccion de frase
    atomic_list_concat([H1,H2], ' ', Phrase),
    interjeccion(Phrase, Eng),  % busca la traduccion de la frase
    !,  % evita backtracking innecesario
    translate_tokens_ei(T, Rest).

translate_tokens_ei([H|T], [Eng|Rest]) :-
    % Traduccion palabra por palabra
    interjeccion(H, Eng), !,
    translate_tokens_ei(T, Rest).

translate_tokens_ei([H|T], [H|Rest]) :-
    % Si no hay traduccion, se conserva la palabra original
    translate_tokens_ei(T, Rest).

% Traduccion Ingles -> Espa ol
translate_tokens_ie([], []).
translate_tokens_ie([H1,H2|T], [Esp|Rest]) :-
    atomic_list_concat([H1,H2], ' ', Phrase),
    interjeccion(Esp, Phrase), !,
    translate_tokens_ie(T, Rest).

translate_tokens_ie([H|T], [Esp|Rest]) :-
    interjeccion(Esp, H), !,
    translate_tokens_ie(T, Rest).

translate_tokens_ie([H|T], [H|Rest]) :-
    translate_tokens_ie(T, Rest).
```

Notas: El algoritmo primero intenta encontrar traducciones de frases completas (dos palabras) antes de ir palabra por palabra. Esto es esencial para interjecciones o modismos. El operador ! asegura que una vez que se encuentra una traduccion valida, no se sigan evaluando reglas innecesarias.

3.3. Tokenizacion y segmentacion de oraciones

- `tokenizar_entrada/2`: Convierte la entrada a minusculas, separa palabras y signos de puntuacion, eliminando elementos vacios
- `dividir_por_delimitadores/2`: Divide la lista de tokens en oraciones completas usando signos de puntuacion o conjunciones como delimitadores

3.4. Analisis de sintagmas

Sintagmas nominales (SN): Manejan estructuras de tipo "determinante + adjetivo + sustantivo + complementos". Se utilizan para mantener la concordancia de genero y numero.

```
% SN completo
sn(Tokens, SN, Resto) :- sn_completo(Tokens, SN, Resto), !.
% SN con adjetivos
sn(Tokens, SN, Resto) :- sn_con_adjetivos(Tokens, SN, Resto), !.
% SN con determinante
sn(Tokens, SN, Resto) :- sn_con_det(Tokens, SN, Resto), !.
% SN simple
sn(Tokens, SN, Resto) :- sn_simple(Tokens, SN, Resto), !.
```

Sintagmas preposicionales (SP): Se forman con preposicion + SN. Permiten traducir construcciones como "of the house" o "en la casa".

```
sp([Prep|RestoTokens], sp(Prep, SN), Resto) :-
    es_preposicion(Prep),
    sn(RestoTokens, SN, Resto), !.
```

Sintagmas verbales (SV): Manejan la conjugacion de verbos segun sujeto y modo de traduccion, incluyendo complementos directos e indirectos, adverbios y adjetivos predicativos.

```
sv(Tokens, Sujeto, Modo, SV, Resto) :-
    sv_con_cd_ci(Tokens, Sujeto, Modo, SV, Resto), !.
sv(Tokens, Sujeto, Modo, SV, Resto) :-
    sv_con_adverbio(Tokens, Sujeto, Modo, SV, Resto), !.
sv(Tokens, Sujeto, Modo, SV, Resto) :-
    sv_con_adjetivo(Tokens, Sujeto, Modo, SV, Resto), !.
sv(Tokens, Sujeto, Modo, SV, Resto) :-
    sv_con_complemento(Tokens, Sujeto, Modo, SV, Resto), !.
sv(Tokens, Sujeto, Modo, SV, Resto) :-
    sv_simple(Tokens, Sujeto, Modo, SV, Resto), !.
```

3.5. Conjugacion verbal y concordancia

- `conjugar_verbo/5` se utiliza para conjugar verbos segun persona, numero y modo
- La concordancia de genero y numero se aplica a sustantivos, adjetivos y determinantes
- Pronombres compuestos o sujetos multiples se consideran para determinar la conjugacion correcta

3.6. Traduccion contextual y por sintagmas

- Se prioriza la traduccion de sintagmas completos antes que palabra por palabra
- Se traducen SN, SV y SP considerando concordancia y modo
- Se soporta traduccion de listas de oraciones y se concatenan en texto final
- Determinantes y adjetivos se traducen segun genero y numero del sustantivo
- Se mantiene coherencia en conjunciones y puntuacion

3.7. Resumen de tipos de traduccion

1. **Traduccion de frases de dos palabras**: expresiones idiomaticas o interjecciones
2. **Traduccion palabra por palabra**: para palabras que no forman parte de sintagmas
3. **Traduccion de sintagmas nominales**: SN completos, con determinantes y adjetivos
4. **Traduccion de sintagmas verbales**: SV con complementos, adjetivos y adverbios
5. **Traduccion de sintagmas preposicionales**: SP que dependen de SN
6. **Traduccion contextual**: mantiene concordancia, modo y coherencia textual

4. Problemas sin solución

Durante el desarrollo del proyecto `TranLog`, se identificaron ciertos aspectos que permanecen abiertos o sin una solución definitiva. Entre los más relevantes se encuentran:

- **Traducción de expresiones idiomáticas complejas:** algunas expresiones compuestas, refranes o modismos que dependen fuertemente del contexto cultural no se traducen correctamente de manera automática, ya que requieren interpretación semántica avanzada.
- **Concordancia en oraciones extensas:** oraciones muy largas con múltiples cláusulas subordinadas presentan dificultades para mantener la concordancia de género, número y persona a lo largo de toda la oración traducida.
- **Ambigüedad lexical:** palabras con múltiples significados o polisemia pueden generar traducciones incorrectas si no se considera el contexto completo de la oración.
- **Pronombres relativos y referencias:** la resolución de referencias a sustantivos anteriores (como "el que", "la cual") aún no se maneja completamente, lo que puede afectar la coherencia de la traducción.

5. Problemas encontrados

Durante la implementación y pruebas del traductor se identificaron diversos problemas que, aunque solucionables, impactaron en el desarrollo y la funcionalidad final:

- **Limitaciones de la base de datos léxica:** el número de palabras y expresiones almacenadas es limitado, lo que provoca que algunas oraciones no puedan ser traducidas de manera completa.
- **Manejo de mayúsculas y puntuación:** al tokenizar la entrada, ciertas combinaciones de mayúsculas, signos de exclamación o interrogación generaron errores en la traducción, requiriendo reglas adicionales de normalización.
- **Problemas de retroceso (*backtracking*):** algunas reglas Prolog generaban resultados múltiples o no deseados debido a un retroceso excesivo, obligando a incluir cortes (`!`) de manera estratégica para controlar la ejecución.
- **Dificultades con estructuras complejas:** la traducción de oraciones con múltiples sintagmas preposicionales o coordinaciones largas produjo a veces resultados incorrectos en el orden de las palabras.
- **Limitaciones en la traducción contextual:** aunque se prioriza la traducción por sintagmas, existen casos donde la concordancia semántica y el contexto pragmático no son correctamente capturados.

6. Conclusiones y recomendaciones

El desarrollo de `TranLog` ha permitido implementar un sistema de traducción basado en lógica que cumple con los objetivos fundamentales de traducir entre español e inglés de manera bidireccional. Se concluye lo siguiente:

- La utilización de Prolog y una base de datos léxica organizada por categorías gramaticales facilita la identificación de estructuras sintácticas y la generación de traducciones coherentes para oraciones simples y medianamente complejas.
- La separación entre la **capa de control** y la **capa lógica** contribuye a la modularidad del sistema, permitiendo futuras mejoras en la base de datos o en las reglas sin afectar la interfaz de usuario.
- Se recomienda ampliar la base léxica con expresiones idiomáticas, refranes y vocabulario especializado para mejorar la cobertura de traducción.

- Implementar técnicas adicionales de procesamiento semántico o de resolución de ambigüedades podría aumentar la precisión en la traducción de oraciones complejas y la correcta interpretación de pronombres relativos.
- Finalmente, se sugiere integrar mecanismos de evaluación automática mediante corpus bilingües para medir la calidad de las traducciones y guiar futuras optimizaciones del sistema.

7. Bitácora

No.	Actividad	Descripción	Responsable(s)	Tiempo estimado	Fecha de entrega	Estado
1	Análisis de requisitos	Estudio de la necesidad de un traductor bidireccional Español-Inglés / Inglés-Español	Todo el equipo	2 h	18/10/2025	Completo
2	Investigación técnica	Búsqueda de recursos sobre Prolog, sintagmas, conjugación y tokenización	Todo el equipo	3 h	19/10/2025	Completo
3	Diseño de base de datos léxica	Definición de hechos para sustantivos, verbos, adjetivos, determinantes, etc.	Eduardo Canessa	4 h	21/10/2025	Completo
4	Implementación de reglas de traducción	Creación de predicados para SN, SV, SP, concordancia y conjugación	Luis Chaves	6 h	21/10/2025	Completo
5	Tokenización y segmentación de oraciones	Predicados para dividir entrada en tokens y oraciones	Deiler Morera	3 h	21/10/2025	Completo
6	Traducción contextual	Traducción de sintagmas nominales, verbales y preposicionales	Todo el equipo	5 h	21/10/2025	Completo
7	Manejo de interjecciones y modismos	Traducción de frases de dos palabras y expresiones idiomáticas	Eduardo Canessa	2 h	22/10/2025	Completo
8	Conjugación verbal y concordancia	Implementación de predicado conjugar_verbo/5 y concordancia de género/número	Luis Chaves	4 h	22/10/2025	Completo
9	Pruebas unitarias	Verificación individual de predicados y hechos	Todo el equipo	4 h	22/10/2025	Completo
10	Pruebas de integración	Ensamblaje y ejecución del sistema completo	Todo el equipo	3 h	23/10/2025	Completo
11	Documentación	Elaboración de reporte de reglas, hechos, estructuras y algoritmos	Deiler Morera	2 h	23/10/2025	Completo
12	Preparación de presentación	Creación de diapositivas y explicación del flujo del traductor	Todo el equipo	1 h	23/10/2025	Completo
13	Entrega final	Revisión general y entrega de proyecto	Todo el equipo	1 h	24/10/2025	Completo

Cuadro 1: Bitácora de actividades del proyecto Prolog

Referencias

[1] Documentación oficial de SWI-Prolog