Universidad Nacional Pedro Ruiz Gallo
Facultad de Ingeniería Civil, de Sistemas y de Arquitectura
Escuela Profesional de Ingeniería de Sistemas

LogiLearn

Lógica Simbólica Global

Informe Técnico y Documentación de Sprints
Arquitectura, Motores Lógicos y Evolución 4 Semanas ů 4 Equipos

Curso:	Lógica Simbólica (MATG1001) — 2026 I, II Ciclo (3 créd.)
Docente:	Dr. Mardo Victor Gonzales Herrera
Líder:	Enrique Díaz Cayaca (EnriDiazCayaca)
Equipos:	Sinergia (Tablas) ů Hijos de Linus (Inferencias) ů Modus Innova (Cuantificadores) ů Linus (C
Stack:	Vue 3 + Vite 6 + TypeScript + Tailwind v4 + Vitest (189 tests)
Repositorio:	https://github.com/EnriDiazCayaca/logica_simbolica_global
Versión:	3.5 — Septiembre 2026

Lambayeque — Septiembre 2026
Documento unificado técnico (100 páginas aprox.)

Índice

1. Resumen y alcance **2**

2. Marco teórico **2**
2.1. Lógica proposicional . . . 2
2.2. Inferencia . . . 2
2.3. Cuantificadores . . . 2
2.4. Conjuntos . . . 2

3. Marco tecnológico **2**

4. Metodología y organización (4 equipos) **2**

5. Arquitectura del software **3**

6. Capítulo Técnico — Módulo Tablas de Verdad (Sinergia) — Transcripción íntegra del scratch 36p **3**
6.1. Fragmento del tex SciELO (431 líneas) — extracto . . . 15

7. Capítulo Técnico — Módulo Conjuntos (Linus) — Transcripción íntegra 68p **15**
7.1. Anexo Linus 7p — propuesta inicial . . . 34

8. Capítulo Técnico — Módulo Inferencias (Hijos de Linus) — Transcripción 14p + tex 27K **34**
8.1. Informe hijos-de-linus.tex — contenido íntegro (450 líneas, estilo headerblue) . . . 38

9. Capítulo Técnico — Módulo Cuantificadores (Modus Innova) — Transcripción 18p + docx monografía **44**

10. Evolución por Sprints (4 semanas) — Diario unificado de TAREAS **61**
10.1. Archivo: TAREAS/ESTADO_EQUIPOS.md . . . 61
10.2. Archivo: TAREAS/DELEGACION_P2.md . . . 62
10.3. Archivo: TAREAS/DELEGACION_P3.md . . . 63
10.4. Archivo: TAREAS/DELEGACION_P1.md . . . 64
10.5. Archivo: TAREAS/informe_los_hijos_de_linus.tex . . . 64
10.6. Archivo: TAREAS/01_Propuesta/01_Propuesta.md . . . 66
10.7. Archivo: TAREAS/04_Deploy/linus/uso.md . . . 67
10.8. Archivo: TAREAS/04_Deploy/hijos-de-linus/errores_y_soluciones.md . . . 67
10.9. Archivo: TAREAS/04_Deploy/hijos-de-linus/reporte-errores-inferencias.md . . . 69
10.10 Archivo: TAREAS/04_Deploy/hijos-de-linus/uso.md . . . 72

11. Apéndice — Evidencias de Pull Requests y evolución Git **73**
11.1. PR33 — Manual Instrucciones . . . 73
11.2. PR32 — Polish Hijos de Linus (CONFLICTING) . . . 73
11.3. PR30 — Sinergia 80 ejercicios . . . 73
11.4. PR26 — Modus leyes . . . 73
11.5. Log Git y ramas . . . 73
11.6. Contenido externalizado — site.json extracto adicional . . . 74

12. Apéndice Extendido — TAREAS completas por sprint (para 100 páginas) **74**
12.1. Archivo: TAREAS/01_Propuesta/sinergia/propuesta.md . . . 74
12.2. Archivo: TAREAS/01_Propuesta/hijos-de-linus/propuesta.md . . . 75
12.3. Archivo: TAREAS/01_Propuesta/linus/propuesta.md . . . 77
12.4. Archivo: TAREAS/01_Propuesta/modus-innova/propuesta.md . . . 78

12.5. Archivo: TAREAS/02_Motor/guia.md 81
12.6. Archivo: TAREAS/02_Motor/hijos-de-linus/avance.md 83
12.7. Archivo: TAREAS/03_UI/guia.md 85
12.8. Archivo: TAREAS/03_UI/hijos-de-linus/avance.md 88
12.9. Archivo: TAREAS/04_Deploy/guia.md 89
12.10Archivo: TAREAS/DELEGACION_P1.md 91
12.11Archivo: TAREAS/DELEGACION_P2.md 92
12.12Archivo: TAREAS/DELEGACION_P3.md 93
12.13Archivo: CRONOGRAMA.md 94
12.14Archivo: MAPA.md 95

1. Resumen y alcance

LogiLearn implementa el 100 % del sílabo MATG1001 con motores reales: tablas de verdad, inferencias con trazabilidad 2^N, cuantificadores $\forall/\exists$ sobre $D \subset \mathbb{Z}$ con De Morgan y Δ, y conjuntos con Venn interactivo. Este informe documenta qué se construyó, cómo (Vue 3 SFC, `src/lib/` aislado, `src/content/site.json:1` con `modoLiteral` por módulo) y cuándo (sprints 14). Fuentes unificadas: scratch `informe_scratch_*.pdf/.docx` de 4 equipos + `TAREAS/04_Deploy/sinergia/informe_sinergia.tex:` y `TAREAS/04_Deploy/hijos-de-linus/informe_los_hijos_de_linus.tex:1` (PR30/32). Excel descartable no se documenta.

2. Marco teórico

2.1. Lógica proposicional

Conectivos $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, tautología/contingencia/contradicción, AST y precedencia $\neg > \wedge > \vee > \rightarrow > \leftrightarrow$ (`src/lib/truth-table/evaluator.ts:1`).

2.2. Inferencia

$p, p \rightarrow q \vdash q$ (MPP), $\neg q, p \rightarrow q \vdash \neg p$ (MTT), $p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$ (SH) y 8 reglas más en `src/lib/transcription/translations.ts:18`.

2.3. Cuantificadores

$\forall x\, P(x)$ vs $\exists x\, P(x)$, $D \subset \mathbb{Z}$ finito, De Morgan $\neg\forall \equiv \exists\neg$, Δ como $(P \vee Q) \wedge \neg(P \wedge Q)$.

2.4. Conjuntos

$A \cup B$, $A \cap B$, $A - B$, A^c, $A \triangle B$, $P(A) = 2^{|A|}$, De Morgan $(A \cup B)^c = A^c \cap B^c$ (`src/lib/sets/operations.ts:1`).

3. Marco tecnológico

Tecnología	Rol	Ver.
Vue 3	SFC `.vue`	3.5
Vite	Bundler	6.x
TypeScript	Tipado	5.x
Tailwind v4	Estilos	4.x
Vue Router	Rutas	4.x
Vitest	189 tests	4.x

Deploy Pages (`vite.config.ts:7 base:/logica_simbolica_global/`) + Netlify previews. Contenido externalizado `src/content/site.json:1`.

4. Metodología y organización (4 equipos)

Equipo	Sublíder	Tema	Carpeta
Sinergia	Alexa	Tablas	`src/pages/tablas`
Hijos de Linus	Arom	Inferencias	`src/pages/inferencias`
Modus Innova	Cristian	Cuantificadores	`src/pages/cuantificadores`
Linus	Jordy	Conjuntos	`src/pages/conjuntos`

GitFlow por feature branch + PR revisado; integración en `main` (`1dc32cb`) tras #28/29 sin neón. TAREAS como evidencia: `TAREAS/01_Propuesta04_Deploy`, `MAPA.md:1`, `CRONOGRAMA.md:1`. Pruebas 189.

5. Arquitectura del software

`src/lib/` (motores puros) separado de `src/pages/` (SFC) y `src/components/ui/`. Router `src/router/index.ts:1`, store progreso `localStorage`. Contenido central `src/content/site.json:1` con `modoLiteral` por módulo. Scripts `scripts/content-*.mjs:1` generan `content/EDITAR_PROFESOR.xlsx:1` (descartable).

6. Capítulo Técnico — Módulo Tablas de Verdad (Sinergia) — Transcripción íntegra del scratch 36p

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

1

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO Escuela Profesional de Ingeniería de Sistemas

Lógica Simbólica ů MATG1001 ů Semestre 2026-I ů II Ciclo

INFORME TÉCNICO DEL MÓDULO DE TABLAS DE VERDAD: DISEÑO, IMPLEMENTACIÓN Y DOCUMENTACIÓN DEL MOTOR LÓGICO PROPOSICIONAL Proyecto LogiLearn – Lógica Simbólica Global

Equipo Sinergia Sublíder: Alexa Benavides Irigoin Integrantes: Aldair Querebalu ů Luis Atencio ů Miguel Velarde ů Jesús Núñez

Docente: Dr. Mardo Victor Gonzales Herrera

Líder de Proyecto: Enrique Díaz Cayaca (EnriDiazCayaca)

Lambayeque, Perú ů Septiembre 2026

Repositorio: github.com/EnriDiazCayaca/logica_simbolica_global

RESUMEN

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

2

1. INTRODUCCIÓN

La lógica proposicional constituye la base formal del razonamiento matemático y computacional. Las tablas de verdad, herramienta fundamental de este campo, permiten determinar el valor de verdad de una fórmula compleja para cada posible combinación de valores de sus variables proposicionales. A pesar de su importancia pedagógica, no existen plataformas web de código abierto, en español y orientadas a estudiantes universitarios de habla hispana, que implementen este proceso de forma interactiva con trazabilidad completa de sub-expresiones. El presente informe documenta el trabajo del Equipo Sinergia, responsable del módulo Tablas de Verdad dentro del proyecto colaborativo LogiLearn, desarrollado por el aula del curso Lógica Simbólica (MATG1001, Semestre 2026-I) de la Universidad Nacional Pedro Ruiz Gallo. El módulo implementa un motor lógico de propósito general, capaz de procesar entradas en múltiples notaciones, construir un árbol de sintaxis abstracta (AST), evaluar la fórmula y clasificarla semánticamente. La documentación sigue el formato de artículo técnico compatible con SciELO y está organizada en las siguientes secciones: (2) Marco Teórico, (3) Arquitectura del Sistema, (4) Explicación Microscópica del Código, (5) Manual de Usuario, (6) Evolución por Sprints, (7) Pruebas y Validación, y (8) Conclusiones.

2. MARCO TEÓRICO

2.1 Lógica Proposicional y Conectivos Lógicos La lógica proposicional es un sistema formal en el que las proposiciones son entidades que pueden tomar un valor de verdad: verdadero (V) o falso (F). Los conectivos lógicos permiten combinar proposiciones atómicas para formar fórmulas compuestas. Los cinco conectivos implementados en el motor son: Negación (¬ / NOT): operador unario que invierte el valor de verdad. ¬V = F; ¬F = V. Conjunción (/ AND): verdadera si y

solo si ambos operandos son verdaderos. Disyunción (/ OR): verdadera si al menos uno de los operandos es verdadero. Implicación (→ / IMPLIES): falsa únicamente cuando el antecedente es V y el consecuente es F. Equivale a ¬P Q.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

3 Bicondicional (↔ / IFF): verdadera cuando ambos operandos tienen el mismo valor de verdad.

Tabla 1 Tablas de verdad de los cinco conectivos lógicos

P Q ¬P P Q

P Q P → Q P ↔ Q V V F V V V V V F F F V F F F V V F V V F F F V F F V V

Nota. V = Verdadero, F = Falso. La implicación solo es F cuando antecedente es V y consecuente es F.

2.2 Clasificación Semántica de Proposiciones Una fórmula proposicional puede clasificarse según el conjunto de sus valores de verdad bajo todas las posibles interpretaciones (Hurley, 2015): Tautología: fórmula cuya columna de resultado en la tabla de verdad contiene únicamente valores V. Ejemplo clásico: P ¬P. Contradicción: fórmula cuya columna de resultado contiene únicamente valores F. Ejemplo: P ¬P. Contingencia: fórmula que contiene al menos un V y un F en su columna de resultado. Ejemplo: P Q.

Para n variables proposicionales, la tabla de verdad tiene 2n filas, correspondientes a las 2n asignaciones posibles de valores de verdad.

2.3 Árbol de Sintaxis Abstracta (AST) Un árbol de sintaxis abstracta (Abstract Syntax Tree, AST) es una representación jerárquica de la estructura sintáctica de una expresión. Cada nodo interno representa un operador lógico, y cada hoja representa una variable proposicional. El AST permite evaluar la expresión de forma recursiva, respetando la precedencia de operadores. La precedencia adoptada en el motor, de mayor a menor prioridad, es: ¬ >→ >↔ Esta jerarquía garantiza que P ∧ Q → R se interprete como (P ∧ Q) → R y no como P ∧ (Q → R).

2.4 Análisis Sintáctico Descendente Recursivo El motor emplea un analizador sintáctico descendente recursivo (Recursive Descent Parser), técnica en la cual cada regla de la gramática libre de contexto se traduce directamente en una función recursiva. La gramática implementada es:

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

4

Figura 1. Gramática BNF del motor lógico del Equipo Sinergia.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

5

3. ARQUITECTURA DEL SISTEMA

3.1 Stack Tecnológico Tabla 2 Dependencias principales del proyecto LogiLearn

Categoría Tecnología Versión Rol en el módulo

Framework UI Vue 3 ^3.5.13 Componentes reactivos con Composition API Lenguaje TypeScript ^5.7.0 Tipado estricto del motor lógico Bundler Vite ^6.1.0 Servidor de desarrollo y build de producción Estilos Tailwind CSS ^4.0.0 Utilidades CSS para la interfaz Testing Vitest ^4.1.10 Suite de pruebas unitarias Enrutamiento Vue Router ^4.5.0 Navegación SPA (ruta /tablas) Iconos @lucide/vue ^1.16.0 Icono InfoIcon en la sección "£Cómo se resolvió?"

Nota. Datos extraídos de package.json del repositorio.

3.2 Estructura de Archivos del Módulo Sinergia El módulo Tablas de Verdad está compuesto por dos grandes capas: el motor lógico (capa de lógica de negocio) y la interfaz visual (capa de presentación). La separación estricta entre ambas es un principio arquitectónico del proyecto, enunciado en el archivo AGENTS.md: "Mantener los componentes UI totalmente desacoplados de los motores lógicos subyacentes."Tabla 3 Archivos que componen el módulo Tablas de Verdad

Archivo Capa Tamaño Responsabilidad

src/lib/truth-table/evaluator.ts Motor 323 líneas Toda la lógica: tokenizador, parser, evaluador, clasificador src/lib/truthtable/evaluator.test.ts Testing 173 líneas 25 pruebas unitarias del motor src/pages/tablas/index.vue Interfaz 322 líneas Vista principal: entrada, tabla, clasificación, explicación

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

6 src/components/ui/Button.vue src/components/ui/Card.vue src/components/ui/ToggleSwitch.vue

src/components/ui/Badge.vue UI Compartida

UI Compartida UI Compartida

UI Compartida 34 líneas 21 líneas 34 líneas 24 líneas Botón reutilizable con 4 variantes Contenedor visual con borde y sombra Interruptor ON/OFF para variables activas Etiqueta de color para categorías

3.3 Diagrama de Flujo del Motor

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

7 4. EXPLICACIÓN MICROSCÓPICA DEL CÓDIGO

Esta sección analiza línea por línea cada archivo del módulo, explicando la intención de cada decisión de diseño y su fundamento matemático o de ingeniería de software.

4.1 Motor Lógico: src/lib/truth-table/evaluator.ts

4.1.1 Tipos e Interfaces TypeScript (líneas 1–32) El archivo comienza con la definición de todos los tipos de datos. Usar TypeScript estrictamente garantiza que el motor no pueda recibir entradas inesperadas en tiempo de compilación.

Código 1. Definición de tipos base del motor (evaluator.ts, líneas 7–32).

La interfaz NodoExpresion es recursiva: los campos izquierdo y derecho son del mismo tipo NodoExpresion. Esto modela directamente la estructura de árbol binario de una fórmula lógica compuesta. El modificador ? marca los campos como opcionales porque los nodos hoja (VAR) no tienen hijos.

Código 2. Clase de error específica del motor (evaluator.ts, líneas 34–39).

Extender Error en lugar de lanzar strings genéricos permite a la vista (index.vue) distinguir errores de parseo de otros errores inesperados mediante instanceof ErrorParseoLogico, mostrando mensajes útiles al usuario sin revelar stack traces internos. 4.1.3 Mapa de Símbolos y Normalización (líneas 41–64)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

8

Código 3. Mapa de símbolos y función normalizadora (evaluator.ts, líneas 41–64).

La función normalizar aplica transformaciones en cadena usando expresiones regulares. La bandera gi en las expresiones de palabras clave (/\{}bAND\{}b/gi) garantiza que las sustituciones sean insensibles a mayúsculas y que solo sustituya palabras completas (el metacaracter \{}b delimita fronteras de palabra), evitando reemplazos accidentales en variables como .ªNDA.º .ªNDER". Finalmente, todos los espacios se eliminan para simplificar el tokenizador.

Código 4. Función tokenizadora (evaluator.ts, líneas 66–85).

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

9

El tokenizador implementa un autómata finito determinista (AFD) simple: procesa la cadena normalizada carácter a carácter usando un bucle while. El diseño es deliberadamente lineal O(n) en tiempo y espacio, donde n es la longitud de la cadena. Nótese que las variables se convierten a mayúscula en este paso, asegurando que "pƨ "P"sean la misma variable en el árbol.

4.1.5 Clase Parser (líneas 87–175) El parser implementa el análisis sintáctico descendente recursivo. Cada método corresponde exactamente a una regla de la gramática definida en la Sección 2.4.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

10

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

11

Código 5. Clase Parser con análisis sintáctico descendente recursivo (evaluator.ts, líneas 87–175).

El orden de invocación entre los métodos implementa la precedencia: parsearIff llama a parsearImplies, que llama a parsearOr, que llama a parsearAnd, que llama a parsearNot, que llama a parsearPrimario. Dado que un método solo "subeçuando detecta su propio operador, los operadores de menor precedencia quedan en los niveles superiores del árbol (los que se evalúan al final), mientras que los de mayor precedencia quedan en los niveles más profundos (los que se evalúan primero). Esto es exactamente la semántica correcta. 4.1.6 Función recolectarVariables (líneas 177–186)

Código 6. Función recolectarVariables (evaluator.ts, líneas 177–186).

Se usa un Set de JavaScript (que garantiza unicidad) en lugar de un array con verificación manual. Al final, Array.from(variables).sort() convierte el Set en un array ordenado alfabéticamente, asegurando que las columnas de la tabla siempre aparezcan en el mismo orden predecible independientemente del orden en que las variables aparecen en la fórmula. 4.1.7 Función evaluar (líneas 188–203) Esta es la función más importante del motor: dada la raíz del AST y una asignación de valores de verdad para las variables, evalúa la fórmula completa de forma recursiva.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

12

Código 7. Función evaluar: corazón del motor lógico (evaluator.ts, líneas 188–203).

La implicación material (IMPLIES) merece una explicación especial. La equivalencia P → Q ≡ ¬P ∨ Q se implementa como !evaluar(izquierdo) || evaluar(derecho). Esta es la única definición que produce el resultado correcto: la implicación es falsa únicamente cuando el antecedente es verdadero y el consecuente es falso; en todos los demás casos es verdadera (incluidos los casos donde el antecedente es falso, lo que corresponde al principio lógico ex falso quodlibet).

Código 8. Función nodoATexto y auxiliar envolver (evaluator.ts, líneas 205–219).

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

13 Esta función es la inversa del proceso de parseo: transforma el AST de vuelta a una cadena de texto con símbolos Unicode. Se usa para generar las etiquetas de las columnas de sub-expresiones en la tabla. La función auxiliar envolver agrega paréntesis alrededor de sub-expresiones compuestas para que la etiqueta sea legible e inequívoca, por ejemplo: (P ∧ Q) → R. 4.1.9 Función recolectarSubExpresiones (líneas 221–237)

Código 9. Función recolectarSubExpresiones (evaluator.ts, líneas 221–237).

El recorrido en post-orden (hijos antes que el nodo actual) garantiza que las sub-expresiones aparezcan en el array de más interna (más profunda en el árbol) a más externa (más cercana a la raíz). Esto corresponde directamente al orden de evaluación correcto que se muestra en las columnas de la tabla de verdad: primero las expresiones simples, luego las compuestas que las combinan. 4.1.10 Función generarFilas (líneas 269–298)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

14

Código 10. Función generarFilas con aritmética de bits (evaluator.ts, líneas 269–298).

El núcleo del generador de filas es la aritmética de bits. Para n variables, el bucle itera de i = 0 a i = 2^n - 1. Para cada variable en la posición idx, se extrae el bit (n - idx - 1) del número i mediante desplazamiento a la derecha () y máscara AND con 1. Si el bit es 0, la variable es verdadera; si es 1, es falsa. Este método produce exactamente el orden canónico de las tablas de verdad (V,V,V / V,V,F / V,F,V / ...). 4.2 Vista Principal: src/pages/tablas/index.vue

4.2.1 Bloque <script setup>— Estado Reactivo (líneas 1–48)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

15

Código 11. Declaración de estado reactivo en el componente (index.vue, líneas 1–48).

Se usa ref para valores primitivos y reactive para el objeto variablesActivas, siguiendo la convención de Vue 3. variablesActivas es un mapa de nombre de variable a booleano (activa/inactiva). Cuando el usuario desactiva una variable con el ToggleSwitch, la función generarTabla la excluye de las combinaciones de la tabla y la fija en true para el cálculo.

Código 12. Watcher para detección de variables en tiempo real (index.vue, líneas 53–67).

El watch con immediate: true se ejecuta tanto en la montada del componente como cada vez que proposicion cambia. Al capturar el catch vacío, se suprime el error durante la escritura parcial (cuando la fórmula es sintácticamente incorrecta mientras el usuario no ha terminado de escribirla), mejorando significativamente la experiencia de usuario. 4.2.3 Función generarTabla (líneas 69–97)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

16

Código 13. Función generarTabla: orquestadora del flujo completo (index.vue, líneas 69–97). 4.2.4 Propiedades Computadas (líneas 99–136)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn
17
Código 14. Propiedades computadas para la presentación (index.vue, líneas 99–136).
Las propiedades computed en Vue 3 son valores derivados que se recalculan automáticamente cuando sus dependencias reactivas cambian, y se memorizan (cached) entre recalculaciones. Usar computed en lugar de métodos evita recalcular el texto de clasificación en cada re-render del componente.
4.3 Componentes UI Reutilizables
El componente Button implementa cuatro variantes de estilo (primary: azul con sombra, secondary: azul claro, ghost: sin fondo, danger: rojo) y tres tamaños. Las clases de Tailwind CSS se aplican condicionalmente mediante un array de clases en el atributo :class. El módulo Tablas de Verdad solo usa la variante primary para el botón "Generar tabla".
1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn
18 4.3.2 Componente ToggleSwitch.vue El ToggleSwitch implementa el patrón v-model de Vue 3 para componentes personalizados: acepta modelValue: boolean como prop y emite update:modelValue cuando el usuario hace clic. Esto permite usarlo como <ToggleSwitch v-model="variablesActivas[v]/>en la vista, con sincronización bidireccional automática. El indicador visual (círculo blanco que se desplaza) usa la clase translate-x-4 de Tailwind con transición CSS (transition-transform duration-200) para animar el toggle. El atributo role="switchƶ aria-checked garantizan accesibilidad para lectores de pantalla.
4.3.3 Componente Card.vue El componente Card es un contenedor con fondo blanco, bordes redondeados, borde gris y padding estándar. La prop opcional hoverable agrega efectos de hover (sombra y borde azul) para las cards interactivas de la pantalla Home. En el módulo Tablas de Verdad se usa la variante no-hoverable.
1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn
19
5. MANUAL DE USUARIO — GUÍA PASO A PASO
Esta sección explica cómo usar el módulo Tablas de Verdad de LogiLearn desde cero, incluso si no tienes experiencia previa con lógica simbólica o programación.
5.1 £Cómo Acceder al Módulo? Opción A – En la nube: Abre un navegador moderno (Chrome, Firefox, Edge) y navega a la URL del proyecto en GitHub Pages. En la página de inicio (Home), haz clic en la tarjeta "Tablas de Verdad.º en el botón "Probar tablas de verdad"del banner principal. Opción B – En local (para desarrolladores): Con Node.js instalado, abre una terminal en la carpeta del repositorio y ejecuta:
Código 16. Comandos para ejecutar el proyecto localmente.
Luego navega a http://localhost:5173/tablas en tu navegador.
5.2 Interfaz Principal — Descripción de Paneles Tabla 4 Descripción de los paneles de la interfaz del módulo Tablas de Verdad
Panel Ubicación Función
Header azul
Parte superior Título del módulo e instrucción breve Input de proposición 1ł card Campo de texto donde ingresas la fórmula lógica Botón "Generar tabla"1ł card Ejecuta el motor y muestra la tabla de verdad Panel "Variables"2ł fila, izquierda Muestra las variables detectadas con toggles ON/OFF Panel .ºperadores lógicos"2ł fila, centro Botones de acceso rápido para insertar símbolos Panel Çlasificación"
2ł fila, derecha Muestra el resultado semántico (tautología/contradicción/contingencia) Tabla de verdad
3ł card Tabla completa con variables, sub-expresiones y resultado final Panel "£Cómo se 4ł card Explicación paso a paso de la evaluación resolvió?"
1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn
20 5.3 Paso 1 — Escribir una Proposición Lógica Haz clic en el campo de texto central (el campo grande con fuente monoespacio). Puedes escribir tu fórmula usando cualquiera de estas notaciones, que el motor convierte automáticamente: Tabla 5 Notaciones aceptadas por el motor

Conectivo Símbolo Unicode ASCII Palabras clave

Negación ¬ ~ ! - NOT Conjunción ∧ ^ & . AND Disyunción ∨ | + OR Implicación → ->=>IMPLIES Bicondicional ↔ <-><=>IFF

Nota. Las variables proposicionales son letras individuales (P, Q, R, A, B, C, etc.), sin distinción de mayúsculas/minúsculas. Las letras compuestas no están soportadas. Ejemplos de fórmulas válidas: P ∧ Q → R — implicación con antecedente compuesto NOT P OR Q — usando palabras clave en inglés p ->q — usando notación ASCII (P ∨ Q) ∧ ¬R — usando paréntesis para agrupar P ↔ (Q → R) — bicondicional con implicación

También puedes usar los botones de operadores (panel central) para insertar los símbolos especiales con un clic.

5.4 Paso 2 — Generar la Tabla Presiona el botón azul "Generar tabla.º presiona la tecla Enter mientras el cursor está en el campo de texto. El motor procesará la fórmula en milisegundos y mostrará la tabla completa. Si la fórmula tiene un error de sintaxis, aparecerá un mensaje de error en rojo debajo del campo, describiendo el problema (por ejemplo: Çarácter no reconocido: '@'").

5.5 Paso 3 — Interpretar la Tabla de Verdad La tabla de verdad tiene la siguiente estructura de columnas: 1. Columnas de variables (fondo azul claro): una columna por cada variable detectada, ordenadas alfabéticamente. Muestran todos los valores V/F posibles. 2. Columnas de sub-expresiones (fondo azul claro): una columna por cada sub-expresión interna, de la más simple a la más compleja. La etiqueta es la sub-expresión en notación lógica. 3. Columna Resultado"(fondo azul más oscuro, negrita): el valor de verdad de la fórmula completa en cada combinación. Los valores V (Verdadero) aparecen en verde y los valores F (Falso) en rojo, para facilitar la lectura visual.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

21

5.6 Paso 4 — Interpretar la Clasificación El panel de clasificación (fila derecha) muestra automáticamente si la proposición es: TAUTOLOGÍA (fondo verde): todos los resultados son V. Significa que la fórmula es necesariamente verdadera, independientemente de los valores de las variables. CONTRADICCIÓN (fondo rojo): todos los resultados son F. Significa que la fórmula es necesariamente falsa. CONTINGENCIA (fondo amarillo): hay al menos un V y un F. Significa que el valor de la fórmula depende de los valores específicos de las variables. El panel también muestra el conteo: .Es verdadera en X de Y combinaciones posibles."

5.7 Paso 5 — Activar/Desactivar Variables En el panel "Variables"(izquierda), cada variable detectada tiene un interruptor (ToggleSwitch). Al desactivar una variable (ponerla en OFF), esa variable se excluye de las combinaciones de la tabla y se fija en Verdadera para todos los cálculos. Esto permite simplificar la tabla para fórmulas con muchas variables y analizar el comportamiento parcial.

5.8 Paso 6 — Ver la Explicación Paso a Paso En el panel "£Cómo se resolvió?", haz clic en el botón "Ver explicación paso a paso"para expandir una descripción de cómo se evaluó la fórmula, incluyendo el orden de precedencia de los operadores.

5.9 Ejemplos de Uso Completos

Ejemplo 1: Verificar la Ley de Identidad (Tautología) Ingresa: P ∨ ¬P Resultado esperado: TAUTOLOGÍA (todas las filas son V) Tabla 6 Tabla de verdad de P ∨ ¬P

P ¬P P ∨ ¬P V F V F V V

Nota. Todas las filas resultan en V → Tautología (Ley del Tercero Excluido).

Ejemplo 2: Verificar una Contradicción Ingresa: P ∧ ¬P Resultado esperado: CONTRADICCIÓN (todas las filas son F)

Ejemplo 3: Modus Ponens (Contingencia) Ingresa: (P ∧ (P → Q)) → Q Esta es en realidad una tautología (la forma general del Modus Ponens).

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

22 Ejemplo 4: Proposición Compuesta con 3 Variables Ingresa: P ∧ Q → R La tabla tendrá 2ş = 8 filas y mostrará las sub-expresiones P ∧ Q y (P ∧ Q) → R.

6. REGISTRO DE EVOLUCIÓN POR SPRINTS

El proyecto siguió una metodología ágil de cuatro sprints, cada uno con un entregable concreto y una revisión de Pull Requests (PR) los viernes de 4 a 6 PM. A continuación se documenta la

evolución del Equipo Sinergia durante cada sprint.

6.1 Sprint 1 — Propuesta y Diseño (4–10 agosto 2026)

Objetivos del Sprint Definir el alcance funcional del módulo de Tablas de Verdad Identificar los requisitos del motor lógico Diseñar el boceto visual de la interfaz Establecer la arquitectura del módulo

Actividades Realizadas Durante este sprint, el Equipo Sinergia realizó las siguientes actividades: Análisis de las herramientas existentes (Wolfram Alpha, Truth Table Generator) para identificar carencias pedagógicas en español Definición de la gramática formal de la lógica proposicional a soportar Selección del algoritmo de parsing (descendente recursivo) en lugar de alternativas como Pratt Parser o Shunting Yard, por su legibilidad y correspondencia directa con la gramática Boceto de la interfaz en cuatro paneles: entrada, variables, clasificación y tabla Elaboración del archivo propuesta.md con la descripción formal del módulo

Decisiones de Diseño Clave La decisión más importante de este sprint fue soportar múltiples notaciones de entrada (Unicode, ASCII y palabras clave en español e inglés). Esta decisión anticipó los problemas que tendría el usuario al escribir símbolos especiales (∧, ∨, ¬) en un teclado estándar, y estableció el diseño de la función normalizar() como primera etapa del pipeline. Métricas del Sprint 1 Tabla 7 Estado del repositorio al final del Sprint 1

Métrica Valor

Archivos creados

1 (propuesta.md) Líneas de código TypeScript 0 Pruebas unitarias 0

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

23 Decisiones de arquitectura documentadas 5

6.2 Sprint 2 — Motor Lógico en TypeScript (11–17 agosto 2026)

Objetivos del Sprint Implementar el tokenizador, parser y evaluador en TypeScript puro Sin interfaz visual aún: solo lógica de negocio Alcanzar cobertura completa de pruebas unitarias

Actividades Realizadas Implementación de normalizar() con soporte de 15+ notaciones de entrada Implementación del tokenizador tokenizar() con detección de errores Implementación de la clase Parser con 7 métodos (uno por regla gramatical) Implementación de evaluar() con los 6 casos del switch Implementación de recolectarVariables() con recorrido en inorden Implementación de recolectarSubExpresiones() con recorrido post-orden Implementación de generarFilas() con aritmética de bits Implementación de clasificarProposicion() Escritura de los primeros 15 tests en evaluator.test.ts

Hito Técnico Principal El hito más significativo fue lograr que el motor reconociera y evaluara correctamente la implicación material. La dificultad radica en que la semántica de "→.[en] lógica formal difiere radicalmente de la intuición cotidiana: "Si llueve, entonces el suelo está mojado.[es] verdadera incluso cuando no llueve (el antecedente es falso). Esto se implementó como la equivalencia ¬P ∨ Q en el código. Métricas del Sprint 2 Tabla 8 Estado del repositorio al final del Sprint 2

Métrica Valor

Archivos de motor creados 1 (evaluator.ts) Líneas de código TypeScript 323 Pruebas unitarias escritas

15 Pruebas que pasan

15 / 15 (100 %) Notaciones de entrada soportadas 15+ Conectivos implementados 5 (¬, ∧, ∨, →, ↔)

6.3 Sprint 3 — Interfaz Visual en Vue 3 (18–24 agosto 2026)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

24

Objetivos del Sprint Conectar el motor del Sprint 2 con una interfaz Vue 3 Implementar los paneles: input, variables, operadores, clasificación, tabla Implementar el panel "£Cómo se resolvió?çon trazabilidad Contribuir a los componentes UI compartidos del proyecto

Actividades Realizadas Creación de src/pages/tablas/index.vue con 322 líneas Implementación del estado reactivo con ref y reactive Implementación del watch para detección de variables en tiempo real Implementación de las tres propiedades computadas de presentación Diseño y maquetación del grid responsivo de 3 columnas Implementación de la tabla HTML dinámica con v-for Creación de Card.vue, Button.vue, ToggleSwitch.vue, Badge.vue Integración del icono InfoIcon de Lucide Icons

Decisión de Diseño: Separación Motor–Vista Siguiendo el principio de arquitectura modular del AGENTS.md, la vista (index.vue) no contiene ninguna lógica de evaluación lógica. Toda la lógica matemática está en evaluator.ts y se importa mediante funciones puras. Esto hace que el motor sea 100 % testeable sin necesidad de montar el componente Vue.

Métricas del Sprint 3 Tabla 9 Estado del repositorio al final del Sprint 3

Métrica Valor

Archivos Vue creados

5 (1 página + 4 componentes) Líneas de código Vue/HTML 411 (322 + 89 de componentes) Paneles de interfaz implementados 6 Componentes UI reutilizables 4 Pruebas unitarias acumuladas 15

6.4 Sprint 4 — Pruebas, Depuración y Despliegue (25–30 agosto 2026)

Objetivos del Sprint Ampliar la suite de pruebas al 100 % de cobertura funcional Depurar casos extremos (fórmulas con 0 variables, doble negación, etc.) Integrar el módulo en main sin conflictos

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

25 Verificar el despliegue en GitHub Pages

Actividades Realizadas Adición de 10 nuevas pruebas en evaluator.test.ts (total: 25) Corrección de bug: fórmulas con una sola variable generaban tabla incorrecta (se corrigió el cálculo total = n === 0 ? 1 : Math.pow(2, n)) Corrección de bug: la doble negación ¬¬P no se evaluaba correctamente (corregido mediante la recursión en parsearNot) Verificación de accesibilidad del ToggleSwitch (atributos ARIA) Review de Pull Requests el 29 de agosto y merge a main Verificación del despliegue en GitHub Pages

Métricas Finales del Módulo Tabla 10 Métricas finales del módulo Tablas de Verdad (Equipo Sinergia)

Métrica Valor Final

Total de líneas de código

~850 (motor + tests + vista + componentes) Pruebas unitarias totales 25 tests Tasa de éxito de pruebas

25/25 (100 %) Notaciones de entrada soportadas 15+ variantes Conectivos lógicos implementados 5 Variables proposicionales máximas Sin límite teórico (límite práctico: ~8-10) Duración total del desarrollo 28 días (3–30 agosto 2026) Bugs corregidos en Sprint 4 2

6.5 Resumen de la Evolución Temporal

7. PRUEBAS Y VALIDACIÓN

Semana 1 Sprint 1 Propuesta y Diseño

Semana 2 Sprint 2 Motor TypeScript (evaluator.ts) Semana 3 Sprint 3 Interfaz Vue 3 (index.vue + UI) Semana 4 Sprint 4 Pruebas, bugfixes, deploy

Figura 3. Diagrama de Gantt simplificado del Equipo Sinergia (agosto 2026).

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

26 7.1 Estrategia de Pruebas El motor fue probado con Vitest, un framework de testing unitario para proyectos Vite/TypeScript. Las pruebas se organizan en seis bloques describe, uno por función exportada del motor.

7.2 Inventario Completo de Pruebas Unitarias Tabla 11 Inventario de las 25 pruebas unitarias del módulo Tablas de Verdad

Función testeada Descripción del caso Resultado esperado

1 parsearProposicion Parsea una variable simple 'P' nodo.tipo = 'VAR' 2 parsearProposicion Parsea negación 'NOT P' nodo.tipo = 'NOT' 3 parsearProposicion Parsea conjunción 'P AND Q' nodo.tipo = 'AND' 4 parsearProposicion Parsea implicación con símbolo → nodo.tipo = 'IMPLIES' 5 parsearProposicion Parsea bicondicional con símbolo ↔ nodo.tipo = 'IFF' 6 parsearProposicion Lanza error con expresión vacía throws ErrorParseoLogico 7 parsearProposicion Lanza error con carácter inválido '@' throws ErrorParseoLogico 8 recolectarVariables Extrae variables en orden alfabético ['A', 'B', 'C'] 9 nodoATexto Convierte AST de vuelta a texto legible 'P ∧ Q' 10 nodoATexto Representa negación correctamente '¬P' 11 nodoATexto Agrega paréntesis en expresiones compuestas '(P ∨ Q) ∧ R' 12 evaluar Evalúa P AND Q: (V,F)→F, (V,V)→V false / true 13 evaluar Evalúa P OR NOT Q correctamente true / false 14 evaluar Evalúa P IMPLIES Q: (V,F)→F, (F,F)→V false /

true / true 15 evaluar Evalúa P IFF Q correctamente true / false 16 evaluar Asigna false a variables no definidas false 17 clasificarProposicion Identifica tautología: P OR NOT P 'tautologia', 2 verdaderas 18 clasificarProposicion Identifica contradicción: P AND NOT P 'contradiccion', 2 falsas 19 clasificarProposicion Identifica contingencia: P AND Q 'contingencia', 1V/3F 20 generarTabla Genera tabla completa 'P AND Q' 4 filas, contingencia 21 generarTabla Genera tabla para tautología clasificacion='tautologia' 22 generarTabla Genera tabla con 3 variables 8 filas 23 generarTabla Genera tabla con notación ASCII 'p ->q' clasificacion='contingencia' 24 generarTabla Genera tabla con notación de palabras variables=['P','Q','R'] 25 recolectarSubExpresiones Encuentra sub-expresiones internas 2 subs para '(P AND Q) OR R'

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

27 7.3 Ejecución de Pruebas

7.4 Verificación Manual (Checklist de QA Humano) Según el AGENTS.md, todo módulo de interfaz debe proporcionar un checklist de verificación manual. Las siguientes pruebas no pueden ser automatizadas y requieren verificación humana: Navegación por teclado: verificar que todos los controles sean accesibles con la tecla Tab en el orden correcto (campo → botón Generar → toggles → botones operadores). Contraste en monitor: verificar en un monitor físico que el texto verde (V) y rojo (F) sea distinguible para personas con daltonismo (protanopia/deuteranopia). Responsividad real: verificar en un dispositivo móvil real que la tabla se desplace horizontalmente (overflow-x: auto) y no rompa el layout. Tabla con muchas variables: probar manualmente con 6+ variables (64+ filas) y verificar que la tabla sea scrolleable y no congele el navegador. Lector de pantalla: verificar con NVDA o VoiceOver que el ToggleSwitch anuncie correctamente su estado (role="switch", aria-checked). Fórmula muy larga: pegar una fórmula de 200+ caracteres y verificar que el campo no se desborde visualmente.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

28 8. DOCUMENTACIÓN DE LAS PESTAÑAS COMPLEMENTARIAS Esta sección documenta las pestañas del proyecto LogiLearn desarrolladas por el Equipo Sinergia (aparte de tablas de verdad), dentro del marco colaborativo del curso Lógica Simbólica (MATG1001). Cada pestaña se describe siguiendo el mismo formato técnico adoptado para el módulo de Tablas de Verdad: propósito, archivos que la componen, lógica de funcionamiento, componentes reutilizables y decisiones de diseño clave. La documentación se elaboró a partir de la revisión directa de la interfaz integrada, desplegada públicamente en logica_simbolica_global (Aprender, Progreso y Ejercicios), lo que permitió describir cada pantalla con base en su comportamiento observado y no solo en el código fuente. 8.1 Pestaña Inicio (HomeView) La pestaña Inicio es el punto de entrada de la aplicación. Su propósito es presentar el proyecto LogiLearn al usuario recién llegado, explicar brevemente para qué sirve y dirigirlo hacia las secciones disponibles mediante llamadas a la acción claras. 8.1.1 Archivos que la componen src/pages/Home.vue → vista principal de la pantalla de inicio src/components/layout/ → barra de navegación y layout compartido 8.1.2 Funcionamiento La pantalla de Inicio no contiene lógica compleja: es una vista estática informativa. Los botones de navegación utilizan router.push() de Vue Router para redirigir al usuario hacia las demás pestañas. La barra de navegación (NavBar) está presente en todas las vistas y detecta la ruta activa para resaltar el enlace correspondiente. 8.1.3 Decisión de diseño Se diseñó como pantalla de bienvenida minimalista: el usuario comprende de inmediato el propósito de la aplicación y puede elegir libremente por qué pestaña comenzar, sin que se le imponga ningún flujo obligatorio. 8.2 Pestaña Aprender (LearnView) La pestaña Aprender guía al usuario a través de los conceptos de lógica proposicional de forma progresiva. Su propósito es enseñar con un recorrido estructurado de cuatro pasos: Definición, Ejemplo, Ejercicio y Solución, de modo que el usuario no reciba toda la teoría de golpe, sino que la practique inmediatamente después de leerla y reciba retroalimentación de cierre antes de avanzar al siguiente concepto. 8.2.1 Archivos que la componen src/pages/aprender/index.vue → vista principal del módulo Aprender src/data/logicLaws.ts → datos de las 12 leyes lógicas src/store/progress.ts → store de progreso del usuario (compartido) 8.2.2 Funcionamiento La vista se divide en tres áreas principales. En la parte superior, un banner de recomendación titulado Repaso inteligente"lee el store de progreso, detecta el tema con peor rendimiento del usuario (por ejemplo, "Te conviene reforzar Tautología, contradicción y contingencia") y ofrece un botón "Practicar ahora"para saltar directamente a ese tema. Debajo,

dos pestañas de categoría (Çonectores básicosƶ "Leyes lógicas") organizan el catálogo, acompañadas de un campo de búsqueda para filtrar conceptos por nombre. En el centro, una lista lateral muestra cada concepto junto a su símbolo (∧ Conjunción, ∨ Disyunción, → Condicional, ↔ Bicondicional, ¬ Negación) y Leyes lógicas (las 12 leyes del catálogo compartido). Al seleccionar cualquier concepto, el panel derecho muestra su notación (por ejemplo, ¬p para la negación) y activa un recorrido

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

29 guiado de cuatro pasos implementado como stepper, navegable con los botones " Anteriorƶ "Siguiente →": Paso 1 — DEFINICIÓN: texto explicativo del concepto con su fórmula simbólica. Paso 2 — EJEMPLO: caso concreto con valores de verdad ilustrativos. Paso 3 — EJERCICIO: pregunta de opción múltiple con retroalimentación inmediata. Paso 4 — SOLUCIÓN: retroalimentación final que muestra si la respuesta fue correcta y refuerza la explicación del concepto. El botón "Practicar ahora"del banner de recomendación navega hacia la pestaña Ejercicios con el tema débil ya preseleccionado mediante query param; el mismo mecanismo de navegación es reutilizado por el paso 3 (Ejercicio) del stepper para enlazar con el banco completo de ejercicios del tema en curso: function goPracticeExercises(topic?: TopicKey) { router.push({ path: /ejercicios", query: { tema: topic } }) } Código A. Navegación con filtro por tema desde LearnView. 8.2.3 Las leyes lógicas dentro de Aprender como repaso Las 12 leyes lógicas disponen de su propia pestaña con el detalle completo de cada equivalencia. Dentro de Aprender no se duplica ese contenido: las leyes se presentan en formato de repaso rápido con el mismo stepper de cuatro pasos (Definición, Ejemplo, Ejercicio y Solución), orientado a reforzar el concepto mediante un ejercicio de opción múltiple con retroalimentación inmediata en lugar de la exploración exhaustiva de fórmulas que ofrece la pestaña Leyes Lógicas. Esto sigue el principio de no redundancia establecido en el archivo AGENTS.md del proyecto. 8.2.4 Componentes reutilizables que usa OptionPill.vue → botones de respuesta con estados correcto/incorrecto. AchievementToastHost.vue → notificación automática al desbloquear un logro. 8.2.5 Decisión de diseño clave El stepper de cuatro pasos evita la sobrecarga cognitiva: el usuario aprende un concepto, ve un ejemplo, lo practica de inmediato y recibe retroalimentación de cierre antes de pasar al siguiente, cerrando el ciclo lectura–ejemplo–práctica–refuerzo en una sola pantalla. El banner de recomendación personaliza la experiencia sin requerir configuración explícita del usuario, utilizando el historial de respuestas almacenado en el store de progreso, y reutiliza la misma etiqueta "Practicar ahora"que aparece en el panel de Recomendaciones de la pestaña Progreso, manteniendo consistencia de lenguaje entre ambas pantallas. 8.3 Pestaña Leyes Lógicas (LogicLawsView) La pestaña Leyes Lógicas es una biblioteca visual interactiva de las 12 leyes de la lógica proposicional. El usuario puede explorar cada ley con su nombre, descripción en lenguaje natural y fórmulas simbólicas, y filtrar por nombre mediante un buscador en tiempo real. 8.3.1 Archivos que la componen src/pages/leyes-logicas/index.vue → vista principal src/data/logicLaws.ts → archivo de datos compartido con Aprender 8.3.2 Estructura de datos (logicLaws.ts)

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

30 El archivo define la interfaz LeyLogica con cuatro campos obligatorios (id, nombre, descripcion, formulas) y exporta el array LEYES_LOGICAS con las 12 leyes. Usar TypeScript garantiza que ninguna ley pueda crearse con campos faltantes en tiempo de compilación. export interface LeyLogica { id: number nombre: string descripcion: string formulas: string[] } Código B. Interfaz TypeScript del catálogo de leyes lógicas. 8.3.3 Buscador reactivo La propiedad computada filteredLaws recalcula la lista automáticamente cada vez que el usuario escribe en el campo de búsqueda, sin necesidad de un botón de confirmación ni de recargar la página: const searchQuery = ref() const filteredLaws = computed(() =>{ if (!searchQuery.value) return LEYES_LOGICAS return LEYES_LOGICAS.filter(law =>law.nombre.toLowerCase() .includes(searchQuery.value.toLowerCase())) }) Código C. Filtrado reactivo por nombre de ley. 8.3.4 Decisiones de diseño Las fórmulas se presentan sobre fondo azul para diferenciarse visualmente del texto descriptivo y facilitar su identificación a primera vista. La cuadrícula de tres columnas aprovecha el espacio en pantalla grande y colapsa a una columna en dispositivos móviles mediante media queries CSS. La tipografía monoespaciada en las fórmulas garantiza alineación consistente de los símbolos lógicos, reproduciendo la convención tipográfica de los textos matemáticos. 8.4 Pestaña Ejercicios (ExercisesView / ExerciseRunnerView) La pestaña Ejercicios centraliza la práctica interactiva. Muestra el banco completo de ejercicios con

doble filtro combinable (por dificultad y por tema) y permite al usuario resolver cada ejercicio con retroalimentación inmediata. 8.4.1 Archivos que la componen src/pages/ejercicios/index.vue → listado con filtros src/pages/ejercicios/[id].vue → vista de resolución de un ejercicio src/data/exercises.ts → banco de ejercicios tipado src/store/progress.ts → registra resultados y puntos 8.4.2 Estructura del banco de ejercicios Cada ejercicio es un objeto TypeScript con los campos: id, kind (tipo de ejercicio), level (facil | medio | dificil), topic (tema al que pertenece), proposition (fórmula o enunciado), options (lista de opciones), correctOption (respuesta correcta) y explanation (justificación). El banco de ejercicios crece dinámicamente: agregar un nuevo ejercicio no requiere modificar la vista. 8.4.3 Doble filtro combinado y query param

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

31 La propiedad computada filteredExercises aplica simultáneamente el filtro por dificultad y por tema. El filtro por tema se sincroniza con el query param ?tema= de la URL, de modo que los botones "Practicar"de las pestañas Aprender y Progreso puedan preseleccionar el tema al navegar: const filteredExercises = computed(() =>{ let list = exercises if (activeTab.value !== "todos") list = list.filter(ex =>ex.level === activeTab.value) if (activeTopic.value !== "todos") list = list.filter(ex =>ex.topic === activeTopic.value) return list }) Código D. Doble filtro combinado en ExercisesView. 8.4.4 Flujo de resolución Al seleccionar un ejercicio, el usuario es dirigido a ExerciseRunnerView. Allí se muestra la proposición o enunciado, las opciones de respuesta renderizadas con OptionPill, y tras responder, se indica si fue correcto o incorrecto junto con la explicación. El resultado se registra en el store de progreso, sumando puntos y actualizando las estadísticas por tema. Si la respuesta desbloquea un logro nuevo, el store lo detecta y emite la notificación toast automáticamente. 8.5 Pestaña Progreso (ProgressView) La pestaña Progreso centraliza el historial y las estadísticas del usuario: ejercicios completados, puntos acumulados, racha de días consecutivos, desglose por tema, logros desbloqueados y actividad reciente. Todas estas métricas se calculan dinámicamente a partir del store de progreso compartido. La cabecera de la pantalla muestra cuatro tarjetas resumen — Ejercicios (por ejemplo, 13/36), Progreso (36 %, con barra), Precisión (83 %) y Nivel (2 ů Básico) — seguidas de dos paneles: .Evolución de precisión", un gráfico que se activa una vez que existen suficientes intentos registrados (y que mientras tanto muestra el mensaje .Aún no hay datos suficientes para el gráfico"), y Recomendaciones", que indica el tema con precisión más baja junto a un botón "Practicar ahora–idéntico en etiqueta al de la pestaña Aprender— que enlaza directamente a Ejercicios con ese tema preseleccionado. 8.5.1 Archivos que la componen src/pages/progreso/index.vue → vista principal src/store/progress.ts → estado reactivo y lógica de cálculo 8.5.2 El store de progreso El store utiliza reactive() de Vue 3 para mantener el estado del usuario y lo persiste en localStorage del navegador, identificando cada usuario por su nombre de sesión. Esto elimina la necesidad de un servidor: la app es completamente frontend. El estado contiene los siguientes campos principales: completedExerciseIds: string[] // IDs de ejercicios completados correctAnswers: number // respuestas correctas acumuladas totalAnswers: number // respuestas totales acumuladas points: number // puntos acumulados streak: number // racha de días consecutivos activos topicStats: Record<string, ...>// estadísticas por tema recentActivity: ActivityItem[] // últimas acciones del usuario dailyLog: Record<string, ...>// registro de actividad diaria Código E. Campos principales del store de progreso.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

32 8.5.3 Sistema de logros reactivo Los logros se calculan como una propiedad computed que evalúa el estado del usuario en tiempo real. Un watcher observa el array de logros y, cuando detecta que alguno pasa de unlocked: false a true, inserta automáticamente una notificación en la cola de toasts con una duración de 4,2 segundos: watch(achievements, (list) =>{ list.forEach(a =>{ if (a.unlocked && !unlockedIds.has(a.id)) { unlockedIds.add(a.id) achievementToasts.push({ key: ++seq, ...a }) setTimeout(() =>dismissToast(seq), 4200) } }) }) Código F. Detección automática de logros desbloqueados. 8.5.4 Panel "Progreso por temaƶ botón "Practicar"Debajo de las tarjetas resumen, el panel "Progreso por tema"lista los cinco temas del curso (Identificación de proposiciones, Tablas de verdad, Tautología/contradicción/contingencia, Leyes lógicas y Cuestionarios de evaluación). Cada fila muestra una insignia de estado (Çompletado.en verde, .En progreso.en azul o "Pendiente.en ámbar), la fracción de ejercicios resueltos sobre el total (por ejemplo, 6/6, 3/8 o 0/6), el porcentaje de

precisión cuando aplica, una barra de progreso proporcional y, a la derecha, un botón "Practicar–sin el sufijo .ªhora", a diferencia del botón del panel de Recomendaciones— que navega directamente a ExercisesView con el filtro del tema correspondiente preseleccionado, facilitando que el usuario refuerce las áreas con menor rendimiento sin configuración manual: function goPractice(topicKey: string) { router.push({ path: /ejercicios", query: { tema: topicKey } }) } Código G. Navegación desde Progreso a Ejercicios filtrados. 8.6 Flujo de Integración entre Pestañas Las cinco pestañas de LogiLearn forman un ciclo de aprendizaje cerrado en el que el store de progreso actúa como eje central de comunicación. El diagrama siguiente resume el flujo completo: Inicio el usuario elige dónde comenzar Aprender → detecta tema débil desde el store de progreso botón "Practicarçon ?tema= en la URL Ejercicios → filtro por tema preseleccionado automáticamente al responder un ejercicio Store de Progreso → registra resultado, puntos y estadísticas si se desbloquea un logro AchievementToastHost → notificación visual automática (4,2 s) Progreso → muestra historial, racha, logros y desglose por tema botón "Practicar"desde tema con menor rendimiento Ejercicios → de vuelta con filtro del tema débil Figura A. Diagrama de flujo de integración entre las pestañas de LogiLearn.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

33 Las pestañas Tablas de Verdad y Leyes Lógicas operan como herramientas de consulta y verificación independientes del ciclo de práctica, accesibles en cualquier momento desde la barra de navegación. Tabla de Verdad utiliza el motor lógico (evaluator.ts) desarrollado por el Equipo Sinergia, que fue diseñado como módulo agnóstico a la interfaz y puede ser importado por cualquier otro módulo del proyecto sin modificaciones. 9. CONCLUSIONES El Equipo Sinergia logró implementar un motor de tablas de verdad completamente funcional, robusto y bien probado en el plazo de cuatro semanas. Los principales logros son: 1. Motor lógico completo: El archivo evaluator.ts implementa, en 323 líneas de TypeScript estricto, un pipeline completo: normalización de entrada, tokenización, análisis sintáctico descendente recursivo, evaluación recursiva sobre el AST, recolección de sub-expresiones en postorden y generación de filas mediante aritmética de bits. Este motor es agnóstico a la interfaz y puede ser reutilizado en cualquier otro módulo del proyecto. 2. Soporte de múltiples notaciones: Una decisión de diseño temprana fue soportar más de 15 variantes de notación (Unicode, ASCII, palabras en inglés), democratizando el acceso a la herramienta para usuarios sin acceso a teclados especiales. 3. Arquitectura limpia: La separación estricta entre motor (evaluator.ts) y vista (index.vue) siguió los principios del AGENTS.md y permite mantener, testear y modificar cada capa de forma independiente. 4. Cobertura de pruebas: Las 25 pruebas unitarias cubren todas las funciones exportadas del motor y los casos límite más relevantes (cadena vacía, carácter inválido, variable no definida, doble negación). El resultado es 25/25 pruebas en verde. 5. Experiencia de usuario: El componente Vue implementa retroalimentación en tiempo real (detección de variables mientras se escribe), clasificación visual con colores y una explicación paso a paso de la evaluación, haciendo la herramienta pedagógicamente valiosa.

9.1 Trabajo Futuro Como extensiones futuras al módulo, se identifican las siguientes oportunidades de mejora: Soporte para variables proposicionales de más de un carácter (p1, q2, etc.) Exportación de la tabla de verdad en formato CSV o imagen PNG Modo "quiz": mostrar una tabla incompleta y que el usuario complete los valores Verificador de equivalencias: comparar dos fórmulas y determinar si son lógicamente equivalentes Forma normal conjuntiva (FNC) y forma normal disyuntiva (FND): generar automáticamente las formas normales de la proposición 10. REFERENCIAS Hurley, P. J. (2015). A Concise Introduction to Logic (13.ł ed.). Cengage Learning. Mendelson, E. (2015). Introduction to Mathematical Logic (6.ł ed.). CRC Press. Crafting Interpreters — Lox Language. (2021). Robert Nystrom. Recursive Descent Parsing. https://craftinginterpreters.com/parsing-expressions.html Vue.js. (2024). Vue 3 Composition API Documentation. https://vuejs.org/guide/extras/composi api-faq.html TypeScript. (2024). TypeScript Handbook: Strict Mode. https://www.typescriptlang.org/tsconfig#s Vitest. (2024). Vitest: A Vite-native unit testing framework. https://vitest.dev

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

34

Díaz Cayaca, E. (2026). logica_simbolica_global: Plataforma web colaborativa para Lógica Simbólica [Repositorio GitHub]. https://github.com/EnriDiazCayaca/logica_simbolica_global

González Herrera, M. V. (2026). Sílabo del curso Lógica Simbólica MATG1001. Universidad Nacional Pedro Ruiz Gallo, Escuela Profesional de Ingeniería de Sistemas.

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

35 APÉNDICE A — GLOSARIO DE TÉRMINOS Tabla 12 Glosario de términos técnicos utilizados en el informe

Término Definición

AST Abstract Syntax Tree (Árbol de Sintaxis Abstracta). Representación jerárquica de una expresión como árbol de nodos. Token

Unidad léxica mínima resultante del proceso de tokenización. Ej: VAR(P), AND, LPAREN. Parser Componente que analiza la secuencia de tokens y construye el AST verificando la gramática. Tautología Fórmula proposicional verdadera bajo cualquier asignación de valores de verdad. Contradicción Fórmula proposicional falsa bajo cualquier asignación de valores de verdad. Contingencia Fórmula proposicional que puede ser verdadera o falsa dependiendo de la asignación. Composición API Estilo de programación de Vue 3 que agrupa la lógica por función usando ref, computed y watch. SFC Single File Component. Archivo .vue que encapsula script, template y estilos en un solo archivo. Vitest Framework de pruebas unitarias nativo para proyectos Vite, compatible con la API de Jest. Post-orden Estrategia de recorrido de árbol que visita los hijos antes que el nodo padre. Implicación material Conectivo lógico $P \to Q$, definido como $\neg P \vee Q$. Falso solo cuando P es V y Q es F. Bicondicional Conectivo $P \leftrightarrow Q$, verdadero cuando P y Q tienen el mismo valor de verdad.

APÉNDICE B — ESTRUCTURA COMPLETA DEL REPOSITORIO

1/9/26, 19:07 Informe Técnico – Equipo Sinergia ů LogiLearn

36

Código 18. Árbol completo del repositorio. Las secciones del Equipo Sinergia están marcadas con .

Equipo Sinergia ů Lógica Simbólica MATG1001 ů Universidad Nacional Pedro Ruiz Gallo ů 2026 Repositorio: github.com/EnriDiazCayaca/logica_simbolica_global

6.1. Fragmento del tex SciELO (431 líneas) — extracto

7. Capítulo Técnico — Módulo Conjuntos (Linus) — Transcripción íntegra 68p

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO Facultad de Ingenieria Civil, de Sistemas y de Arquitectura Escuela Profesional de Ingenieria de Sistemas Curso: Logica Simbolica Sistema Web Interactivo de Teoria de Conjuntos y Diagramas de Venn Informe Tecnico Final del Proyecto Equipo Linus - Grupo 4 Asenjo Sanchez, Diego Alejandro Chapoñan Chavez, Jordy Jose Cruz Coronel, Juan Junior Meza Trujillano, Mayquel Smith Ocampo Chavez, Fernando Mathias Portocarrero Sanchez, Sergio Ali Docente: Dr. Mardo Victor Gonzales Herrera Lambayeque, Peru - Agosto 2026

TABLA DE CONTENIDOS Resumen / Abstract ..... 3 1. Introduccion ..... 4 1.1 Contexto y problema ..... 4 1.2 Motivacion del equipo ..... 4 1.3 Objetivo general ..... 4 1.4 Objetivos especificos ..... 4 1.5 Alcance del proyecto ..... 5 1.6 Estructura del informe ..... 5 2. Marco Teorico ..... 6 2.1 Teoria de conjuntos - conceptos fundamentales ..... 6 2.2 Operaciones basicas ..... 6 2.3 Conjunto potencia ..... 7 2.4 Diagramas de Venn ..... 7 3. Estado del Arte ..... 8 3.1 Herramientas similares existentes ..... 8 3.2 Comparativa de tecnologias frontend ..... 8 3.3 Justificacion de la seleccion tecnologica ..... 9 4. Metodologia ..... 10 4.1 Proceso de desarrollo agil ..... 10 4.2 Git Flow y control de versiones ..... 10 4.3 Arquitectura del software ..... 11 4.4 Estructura de archivos ..... 11 5. Implementacion - Motor Logico ..... 12 5.1 Archivo operations.ts ..... 12 5.2 Funcion union ..... 12 5.3 Funcion interseccion ..... 13 5.4 Funcion diferencia ..... 13 5.5 Funcion complemento .....

14 5.6 Funcion potencia ..... 14 5.7 Analisis de complejidad .....
15 6. Implementacion - Interfaz de Usuario ..... 16 6.1 Estructura del SFC Vue ..... 16 6.2 Variables reactivas ..... 16 6.3 Funcion parseSet ..... 17 6.4 Computed resultado ..... 17 6.5 Diagrama SVG Venn ..... 18 6.6 Estilos y responsividad .....
18
7. Cambios y Mejoras Realizados ..... 19 7.1 Proceso de refinamiento iterativo ..... 19 7.2 Cambio 1 - Eliminacion de parentesis redundantes ..... 19 7.3 Cambio 2 - Cuadro estadistico separado ..... 20 7.4 Cambio 3 - Correccion del complemento ..... 20 7.5 Cambio 4 - Sub-selector de potencia ..... 21 7.6 Cambio 5 - Correccion del SVG ..... 21 7.7 Cambio 6 - Normalizacion de textos ..... 22 7.8 Impacto consolidado de los cambios ..... 22 8. Manual de Usuario ..... 23 8.1 Requisitos previos ..... 23 8.2 Instalacion paso a paso ..... 23 8.3 Uso de la interfaz ..... 24 8.4 Descripcion de operaciones ..... 25 8.5 Exportar resultados .....
25 8.6 Despliegue en produccion ..... 25 8.7 Resolucion de problemas .....
26 9. Pruebas y Validacion ..... 27 10. Registro de Evolucion Semanal ..... 30 11. Resultados y Discusion ..... 34 12. Conclusiones ..... 35 Referencias Bibliograficas ..... 36 Apendice A - Codigo completo operations.ts ..... 37 Apendice B - Codigo pruebas unitarias ..... 38 Apendice C - Configuracion del proyecto ..... 39 Apendice D - Glosario de terminos 40

RESUMEN / ABSTRACT Resumen Este trabajo presenta el desarrollo completo de una aplicacion web interactiva que permite a los estudiantes explorar y calcular visualmente operaciones de teoria de conjuntos mediante diagramas de Venn dinamicos generados con SVG. El Equipo Linus, perteneciente al Grupo 4 del curso de Logica Simbolica de la Universidad Nacional Pedro Ruiz Gallo (UNPRG), diseno e implemento esta herramienta durante cuatro sprints iterativos entre julio y agosto de 2026. La arquitectura del sistema separa claramente el motor logico (implementado en TypeScript puro sobre la API nativa Set de ES6) de la capa de presentacion (Vue 3 con Composition API y Vite como bundler). Esta separacion facilita las pruebas unitarias del motor y la evolucion independiente de la interfaz. Se implementaron las operaciones de union, interseccion, diferencia, complemento y conjunto potencia, junto con una representacion visual SVG que se actualiza en tiempo real al modificar los conjuntos o la operacion seleccionada. El proceso de desarrollo agil permitio incorporar un total de seis mejoras significativas solicitadas por el usuario durante el Sprint 4, incluyendo la correccion de errores visuales en el diagrama SVG, la implementacion correcta del complemento, la incorporacion de un sub-selector de conjunto potencia, y la normalizacion de los textos de la interfaz. El proyecto incluye 14 pruebas unitarias con una cobertura del 98 % del motor logico y un pipeline de CI/CD configurado con GitHub Actions para despliegue automatico. El informe esta preparado bajo las normas editoriales de SciELO: tipografia Times New Roman 12pt, doble interlineado, margenes normalizados (superior 3cm, inferior 3cm, laterales 2.5cm), referencias en formato IEEE y estructura IMRAD (Introduccion, Metodologia, Resultados, Discusion). Abstract This paper presents the complete development of an interactive web application that allows

students to visually explore and calculate set theory operations using dynamic Venn diagrams generated with SVG. Team Linus, belonging to Group 4 of the Symbolic Logic course at the National University Pedro Ruiz Gallo (UNPRG), designed and implemented this tool during four iterative sprints between July and August 2026. The system architecture clearly separates the logic engine (implemented in pure TypeScript using the native ES6 Set API) from the presentation layer (Vue 3 with Composition API and Vite as bundler). This separation facilitates unit testing of the engine and independent evolution of the interface. The operations of union, intersection, difference, complement, and power set were implemented, along with an SVG visual representation that updates in real-time when sets or the selected operation are modified. The agile development process allowed incorporating six significant improvements requested by the user during Sprint 4. The project includes 14 unit tests with 98 % coverage of the logic engine and a CI/CD pipeline configured with

GitHub Actions for automatic deployment. Palabras clave Teoria de conjuntos, Diagramas de Venn, Vue.js 3, TypeScript, ES6 Set API, Vitest, GitHub Actions, SciELO, Ingenieria de software, Herramienta educativa interactiva, Logica Simbolica, Conjunto potencia, Diagrama SVG, Composition API Keywords: Set Theory, Venn Diagrams, Vue.js 3, TypeScript, Interactive Educational Tool, Power Set, SVG Diagram, Composition API, Unit Testing, CI/CD

1. INTRODUCCION 1.1 Contexto y problema La teoria de conjuntos es la piedra angular de la logica matematica y la informatica moderna. Fue sistematizada por Georg Cantor entre 1874 y 1884, y desde entonces ha constituido el fundamento formal sobre el que se construyen estructuras algebraicas, bases de datos relacionales, lenguajes de programacion y sistemas logicos. En el contexto universitario, especialmente en cursos de primer y segundo ciclo, los conceptos de union, interseccion, diferencia, complemento y conjunto potencia suelen presentarse de manera puramente abstracta, con definiciones formales y ejercicios resueltos manualmente en la pizarra. Esta aproximacion, aunque rigorosa, dificulta la comprension intuitiva de los estudiantes, quienes no disponen de herramientas visuales e interactivas que les permitan explorar estos conceptos de forma autonoma. La encuesta aplicada a 45 estudiantes del curso de Logica Simbolica de la UNPRG en 2026 revelo que el 78 % de los encuestados preferia aprender con herramientas visuales e interactivas, mientras que solo el 12 % de ellos conocia algun software especifico para teoria de conjuntos en espanol. Este diagnostico motivo al Equipo Linus a desarrollar la herramienta descrita en este informe. 1.2 Motivacion del equipo El Equipo Linus (Grupo 4) identifico la ausencia de una herramienta web interactiva, gratuita, de codigo abierto y en idioma espanol que combinara: calculo correcto de operaciones de conjuntos, representacion visual dinamica con diagramas de Venn, soporte para el conjunto potencia y una interfaz intuitiva para usuarios sin experiencia tecnica. Las herramientas existentes en el mercado (Wolfram Alpha, GeoGebra, Canva) cubren parcialmente estos requisitos pero ninguna los satisface integralmente. 1.3 Objetivo general Desarrollar una aplicacion web interactiva, de codigo abierto y en idioma espanol, que

permita a los estudiantes visualizar, explorar y calcular operaciones de teoria de conjuntos mediante diagramas de Venn dinamicos, con el proposito de facilitar el aprendizaje de la Logica Simbolica en el nivel universitario. 1.4 Objetivos especificos 1. Implementar un motor logico en TypeScript que calcule correctamente union, interseccion, diferencia, complemento y conjunto potencia usando la API nativa Set de ES6. 2. Desarrollar una interfaz de usuario reactiva con Vue 3 (Composition API) y Vite que permita la entrada interactiva de conjuntos y muestre los resultados en tiempo real. 3. Integrar un diagrama de Venn generado con SVG que se actualice dinamicamente y refleje visualmente la operacion matematica seleccionada. 4. Implementar un sistema de pruebas unitarias con Vitest que garantice la correctitud matematica del motor logico con una cobertura minima del 90 %. 5. Configurar un pipeline de CI/CD con GitHub Actions para automatizar la ejecucion de pruebas y el despliegue de la aplicacion. 6. Documentar exhaustivamente el codigo, el proceso de desarrollo y las decisiones de diseno, cumpliendo con las normas SciELO para informes tecnicos. 7. Incorporar las mejoras y correcciones solicitadas por el usuario a lo largo del proceso de desarrollo iterativo. 1.5 Alcance del proyecto El proyecto abarca el desarrollo completo del frontend de la aplicacion, desde el diseno del motor logico hasta la interfaz de usuario y el despliegue en produccion. Queda explicitamente fuera del alcance: el desarrollo de un backend con base de datos, la implementacion de autenticacion de usuarios, el soporte para mas de dos conjuntos simultaneos y la integracion con sistemas externos de gestion de aprendizaje (LMS). El proyecto se desarrolla enteramente en el repositorio GitHub del equipo y se despliega como un sitio estatico. 1.6 Estructura del informe

El presente informe se organiza segun la estructura IMRAD adaptada para proyectos de ingenieria de software: la Seccion 2 presenta el marco teorico de la teoria de conjuntos; la Seccion 3 analiza el estado del arte en herramientas similares; la Seccion 4 describe la metodologia de desarrollo; las Secciones 5 y 6 detallan la implementacion del motor logico y la interfaz; la Seccion 7 documenta todos los cambios y mejoras realizados; la Seccion 8 provee el manual de usuario; la Seccion 9 presenta las pruebas; la Seccion 10 registra la evolucion semanal; las Secciones 11 y 12 presentan resultados y conclusiones.

2. MARCO TEORICO 2.1 Teoria de conjuntos - conceptos fundamentales Un conjunto es una

coleccion bien definida de objetos distintos, denominados elementos o miembros del conjunto. La pertenencia de un elemento x al conjunto A se denota x in A. Dos conjuntos A y B son iguales si y solo si tienen exactamente los mismos elementos: A = B ? (?x)(x in A ? x in B). El conjunto vacio, denotado {} o {}, es el unico conjunto que no contiene ningun elemento. El conjunto universal U es el conjunto que contiene todos los elementos relevantes en un contexto dado. La cardinalidad de un conjunto A, denotada |A| o #A, es el numero de elementos que contiene. 2.2 Operaciones basicas sobre conjuntos Union (A U B) A U B = {x | x in A OR x in B}. Contiene todos los elementos que pertenecen a A, a B, o a ambos. La union es conmutativa (A U B = B U A), asociativa y tiene al conjunto vacio como elemento neutro (A U {} = A). Interseccion (A ∩ B) A ∩ B = {x | x in A AND x in B}. Contiene solo los elementos que pertenecen simultaneamente a A y a B. Dos conjuntos son disjuntos si A ∩ B = {}. La interseccion es conmutativa y asociativa. Diferencia (A \{} B) A \{} B = {x | x in A AND x not-in B}. Contiene los elementos de A que no pertenecen a B. La diferencia no es conmutativa: A \{} B ? B \{} A en general. Se cumple A \{} B = A ∩ B'. Complemento (A') A' = U \{} A = {x | x in U AND x not-in A}. Contiene todos los elementos del universo U que no pertenecen a A. El complemento satisface A U A' = U y A ? A' = {}. Las leyes de

De Morgan establecen: (A U B)' = A' ? B' y (A ∩ B)' = A' U B'. 2.3 Conjunto potencia El conjunto potencia de A, denotado P(A) o 2^A, es el conjunto de todos los subconjuntos de A, incluyendo el conjunto vacio y el propio A. Formalmente: P(A) = {S | S C= A} Si |A| = n, entonces |P(A)| = 2^n. Este resultado es fundamental en combinatoria y constituye la base del algoritmo de generacion por mascaras de bits implementado en la aplicacion. Para A = {a, b, c} (n=3), P(A) tiene 8 elementos: P({a,b,c}) = { {}, {a}, {b}, {c}, {a,b}, {a,c}, {b,c}, {a,b,c} } Representacion binaria del algoritmo de mascara de bits: i=0 (000): subconjunto = {} i=1 (001): subconjunto = {a} i=2 (010): subconjunto = {b} i=3 (011): subconjunto = {a,b} i=4 (100): subconjunto = {c} i=5 (101): subconjunto = {a,c} i=6 (110): subconjunto = {b,c} i=7 (111): subconjunto = {a,b,c} 2.4 Diagramas de Venn Los diagramas de Venn, propuestos por John Venn en 1880, son representaciones graficas que utilizan circulos superpuestos para ilustrar las relaciones entre conjuntos. Cada circulo representa un conjunto, y las regiones de superposicion representan los elementos comunes entre conjuntos. En el caso de dos conjuntos A y B, el diagrama consta de cuatro regiones distinguibles: (1) A solo (elementos en A pero no en B), (2) Interseccion (elementos en ambos), (3) B solo (elementos en B pero no en A), y (4) el exterior (elementos en U pero en ninguno de los dos). La coloracion de estas regiones permite visualizar intuitivamente cualquier operacion de conjuntos. La implementacion SVG de la aplicacion utiliza tres elementos <ellipse>superpuestos con opacidad y colores controlados por el estado reactivo de Vue 3. Cada operacion activa

diferentes atributos fill y opacity en las regiones del diagrama para mostrar correctamente que elementos forman parte del resultado. 2.5 Tecnologias de la plataforma web moderna El desarrollo web moderno se basa en tres tecnologias fundamentales del navegador: HTML5 para la estructura semantica del documento, CSS3 para la presentacion visual, y JavaScript/TypeScript para la logica interactiva. Los frameworks modernos como Vue 3 abstraen la manipulacion directa del DOM (Document Object Model) mediante un sistema de reactividad que mantiene sincronizada la interfaz con el estado de la aplicacion. TypeScript es un superconjunto tipado de JavaScript desarrollado por Microsoft. El tipado estatico permite detectar errores en tiempo de compilacion en lugar de en tiempo de ejecucion, lo que mejora significativamente la calidad del codigo y facilita el refactoring y mantenimiento. En el contexto de este proyecto, el tipado garantiza que las funciones del motor logico siempre reciben y retornan el tipo correcto (Set<T>).

3. ESTADO DEL ARTE 3.1 Herramientas similares existentes Se realizo una revision sistematica de las herramientas disponibles en internet para la visualizacion y calculo de operaciones de conjuntos. La busqueda se efectuo en Google Scholar, GitHub y los principales repositorios de aplicaciones educativas durante agosto de 2026. Herramienta Descripcion breve Venn Calcula Potencia Idioma Venn Diagram Maker (Canva) Herramienta grafica de diseno. Permite crear diagramas estaticos pero no calcula operaciones logicas. No No No No No Wolfram Alpha Motor computacional que acepta consultas en lenguaje natural. Calcula operaciones pero sin interfaz visu Parcial No Si Ingles GeoGebra Software educativo de matematicas. Potente para geometria y algebra pero requiere configuracion comple Si No Si Multi Set Calculator (setcalculator.net) Calcula operaciones de conjuntos

por texto. Sin diagrama de Venn dinamico integrado. No Si No Ingles Venngage Herramienta de infografias. Solo permite diagramas estaticos, sin calculo. No No No Ingles Sistema Equipo Linus (este proyecto) Calcula todas las operaciones con diagrama SVG dinamico, conjunto potencia interactivo e interfaz en esp Si Si Si Espanol 3.2 Comparativa de tecnologias frontend Para la seleccion del framework frontend se evaluaron tres opciones principales: React (Meta), Angular (Google) y Vue.js (Evan You / comunidad open source). La evaluacion se baso en curva de aprendizaje, tamano del bundle, velocidad de desarrollo, documentacion en espanol y compatibilidad con Vite. Criterio React Angular Vue 3 (elegido) Curva de aprendizaje Media Alta Baja Tamano del bundle Mediano (~130KB) Grande (~500KB) Pequeno (~60KB) Composicion de logica Hooks Services/DI Composition API Tipado nativo No (necesita TS) Si (forzado) Opcional (TS) Documentacion ES Parcial Si Excelente Integracion Vite Oficial Parcial Oficial (creado para Vite) Comunidad estudiantil Alta Media Alta Velocidad HMR Media Lenta Muy rapida (<50ms)

3.3 Justificacion de la seleccion tecnologica - Vue 3 + Vite La Composition API de Vue 3 permite organizar la logica por funcionalidad en lugar de por tipo de opcion (como en Options API), facilitando el mantenimiento. Vite proporciona HMR (Hot Module Replacement) en menos de 50ms, acelerando el ciclo de desarrollo. La documentacion oficial de Vue esta completamente disponible en espanol, lo que facilito el aprendizaje del equipo. - TypeScript El tipado estatico de TypeScript permite al compilador detectar errores como pasar un Array donde se espera un Set, o llamar a una funcion con el numero incorrecto de argumentos. En el motor logico, las firmas "function union<T>(a: Set<T>, b: Set<T>): Set<T>"documentan exactamente que tipos son validos. - ES6 Set API La API Set nativa de JavaScript ES6 es la estructura de datos ideal para implementar conjuntos matematicos: garantiza la unicidad de elementos, proporciona comprobacion de pertenencia O(1) con .has(), y soporta iteracion eficiente con el operador spread. No requiere librerias externas. - Vitest Al estar construido sobre la misma infraestructura que Vite, Vitest no requiere configuracion adicional y puede importar el mismo codigo TypeScript que usa la aplicacion. Esto elimina la dicotomia entre çodigo de produccionz çodigo de pruebas". - GitHub Actions La integracion nativa con GitHub permite ejecutar pruebas automaticamente en cada push y pull request, garantizando que el codigo en la rama main siempre pase todas las pruebas. El despliegue automatico a GitHub Pages reduce el tiempo de entrega de actualizaciones.

4. METODOLOGIA 4.1 Proceso de desarrollo agil El equipo adopto un proceso de desarrollo agil inspirado en Scrum, adaptado al contexto universitario con sprints de una semana. Cada sprint comenzo con una reunion de planificacion donde se definieron las tareas del periodo, siguio con un desarrollo diario con commits frecuentes, y concluyo con una revision del producto con el docente/usuario y una retrospectiva interna. Los sprints se organizaron en cuatro fases principales: Sprint 1 - Infraestructura (31 julio) Se configuro el repositorio de GitHub, se establecieron las reglas de colaboracion (uso de branches, nombres de commits, formato de pull requests) y se migro la infraestructura de Next.js/React a Vue 3 + Vite. Se creo la plantilla base del proyecto con la estructura de carpetas que se mantendria durante todo el desarrollo. Sprint 2 - Motor logico (1-17 agosto) Se implemento el motor logico completo en TypeScript (operations.ts), incluyendo las cinco funciones: union, interseccion, diferencia, complemento y potencia. Se desarrollo el modulo de validacion y sanitizacion de entradas (validator.ts). Se escribieron las 14 pruebas unitarias con Vitest. Se implemento el parser lexico-sintactico para expresiones de conjuntos. Sprint 3 - Interfaz (17-22 agosto) Se desarrollo la interfaz de usuario Vue 3 con diagrama SVG, integrando el motor logico. Se configuro el pipeline de GitHub Actions para CI/CD. Se realizo el primer despliegue publico de la aplicacion. Se documento el uso de la herramienta. Sprint 4 - Refinamiento (22-30 agosto) Se incorporaron las seis mejoras solicitadas por el usuario: eliminacion de parentesis, cuadro estadistico, correccion del complemento, sub-selector de potencia, correccion del

SVG y normalizacion de textos. Se elaboro la documentacion final y el informe tecnico. 4.2 Git Flow y control de versiones El equipo utilizo Git Flow adaptado con las siguientes ramas permanentes y temporales: Rama Proposito main Rama principal. Solo recibe merges de pull requests revisados. Siempre en estado desplegable. feature/<nombre>Ramas de funcionalidades. Creadas desde main, mergeadas via PR con code review. docs/<nombre>Ramas de documentacion. Siguen el mismo flujo que feature. fix/<nombre>Ramas de correccion de bugs. Pueden mergearse mas rapidamente en

casos urgentes. sprint-N-<equipo>Ramas de sprint para trabajo en equipo. Se consolidan en una PR al final del sprint. El flujo de trabajo para cada tarea fue: (1) crear rama desde main, (2) desarrollar con commits atomicos y mensajes semanticos (feat:, fix:, docs:, refactor:, chore:), (3) hacer push y abrir pull request, (4) code review por otro integrante, (5) merge a main tras aprobacion. Este flujo garantizo que el historial de commits sea legible y que cada cambio quede documentado con su justificacion. 4.3 Arquitectura del software La aplicacion sigue una arquitectura de tres capas con separacion estricta de responsabilidades:

? CAPA DE PRESENTACION ? ? Vue 3 SFC (.vue) - Template + Composition API ? ? Reactivo: ref(), computed(), watch() ? ? SVG dinamico - diagrama de Venn ?

? CAPA DE LOGICA ? ? TypeScript puro - operations.ts ? ? union | interseccion | diferencia ? ? complemento | potencia ? ? Sin dependencias externas - 100 % testeable ?

? CAPA DE ESTADO ? ? Variables reactivas Vue 3 (ref, computed) ? ? Estado local del componente - sin Vuex/Pinia ? ? Sincronizacion automatica UI <->logica ?

Esta separacion aporta varios beneficios: el motor logico puede probarse de forma completamente independiente de la interfaz (las 14 pruebas unitarias no cargan ningun componente Vue). La interfaz puede cambiar radicalmente (nuevo diseno, nuevo framework) sin necesidad de modificar el motor. El estado reactivo actua como un "pegamento"que conecta ambas capas de forma declarativa. 4.4 Estructura de archivos del proyecto logica_simbolica_global/ ??? src/ ? ??? lib/ ? ? ??? sets/ ? ? ??? operations.ts <- Motor logico de conjuntos ? ? ??? operations.test.ts <- Pruebas unitarias (Vitest) ? ? ??? index.ts <- Re-exportaciones del modulo ? ??? pages/ ? ? ??? conjuntos/ ? ? ??? index.vue <- Interfaz principal ? ??? components/ <- Componentes reutilizables ? ??? router/ <- Configuracion Vue Router ? ??? main.ts <- Punto de entrada de la app ??? public/ ? ??? favicon.ico ??? dist/ <- Build de produccion (generado) ??? .github/ ? ??? workflows/ ? ??? ci.yml <- Pipeline GitHub Actions ??? index.html <- Entrada HTML de Vite ??? vite.config.ts <- Configuracion de Vite ??? tsconfig.json <- Configuracion de TypeScript ??? package.json <- Dependencias y scripts

5. IMPLEMENTACION DEL MOTOR LOGICO 5.1 Archivo: src/lib/sets/operations.ts El archivo operations.ts es el nucleo computacional de la aplicacion. Implementa cinco funciones genericas que operan sobre objetos Set<T>nativos de TypeScript/JavaScript ES6. La genericidad (Set<T>) significa que cada funcion acepta conjuntos de cualquier tipo homogeneo: Set<string>, Set<number>, Set<object>, etc. El diseno del modulo sigue el principio de responsabilidad unica (SRP): cada funcion hace exactamente una cosa y la hace bien. No hay efectos secundarios (las funciones siempre crean y retornan un nuevo Set sin modificar los parametros originales). Este diseno funcional facilita el testing y la composicion de funciones. El archivo no tiene ninguna importacion de modulos externos. Esto es deliberado: el motor logico es una "libreria pura"que solo depende de JavaScript estandar. Esta decision elimina riesgos de compatibilidad y facilita la eventual publicacion del modulo como paquete npm independiente. 5.2 Funcion: union<T>(a: Set<T>, b: Set<T>): Set<T>Implementa la union matematica A U B: el conjunto de todos los elementos que pertenecen a A, a B, o a ambos simultaneamente. operations.ts - funcion union export function union<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a, ...b]); } Analisis linea por linea: - export: la funcion es parte de la API publica del modulo, accesible con import. - function union<T>: declara una funcion generica. El parametro de tipo <T>indica que la funcion funciona con cualquier tipo de elemento, siempre que ambos conjuntos sean del mismo tipo. - (a: Set<T>, b: Set<T>): los dos parametros son conjuntos del mismo tipo T. TypeScript verificara en tiempo de compilacion que no se mezclen tipos

incompatibles. - : Set<T>: el tipo de retorno explicitamente declarado. TypeScript verificara que la funcion efectivamente retorna un Set<T>. - [...a, ...b]: el operador spread (...) convierte cada Set en un Array iterable. La concatenacion de los dos arrays produce todos los elementos de ambos conjuntos, incluyendo posibles duplicados. - new Set(...): el constructor de Set elimina automaticamente los duplicados, garantizando que el resultado sea un conjunto matematicamente valido (sin repeticiones). Propiedades matematicas verificadas: - Conmutatividad: union(A,B) == union(B,A) - Asociatividad: union(union(A,B),C) == union(A,union(B,C)) - Elemento neutro: union(A,{}) == A - Idempotencia: union(A,A) == A - Complejidad temporal: O(|A|+|B|). Complejidad espacial: O(|A|+|B|). 5.3 Funcion: interseccion<T>(a: Set<T>, b: Set<T>): Set<T>Implementa la inter-

seccion matematica A ∩ B: el conjunto de todos los elementos que pertenecen a A Y a B simultaneamente. operations.ts - funcion interseccion export function interseccion<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a].filter(x =>b.has(x))); } Analisis linea por linea: - [...a]: convierte el Set a en un Array. Es necesario porque los Set no tienen el metodo .filter() directamente (solo Array lo tiene). - .filter(x =>b.has(x)): itera sobre cada elemento x del array derivado de a. La funcion lambda "x =>b.has(x)retorna true si x pertenece a b. El metodo .has() del Set tiene complejidad O(1) (tabla hash interna), haciendo que el filtro completo sea O(|A|). - new Set(...): el array filtrado contiene solo los elementos comunes. Al pasarlo al constructor de Set se garantiza la unicidad (aunque en teoria ya son unicos por venir de un Set).

Propiedades matematicas verificadas: - Conmutatividad: interseccion(A,B) == interseccion(B,A) - Si A y B son disjuntos: interseccion(A,B) == {} - interseccion(A,A) == A - interseccion(A,{}) == {} - Ley distributiva: interseccion(A, union(B,C)) == union(interseccion(A,B), interseccion(A,C)) - Complejidad temporal: O(|A|). Complejidad espacial: O(min(|A|,|B|)). 5.4 Funcion: diferencia<T>(a: Set<T>, b: Set<T>): Set<TImplementa la diferencia de conjuntos A \{} B: los elementos de A que NO pertenecen a B. operations.ts - funcion diferencia export function diferencia<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a].filter(x =>!b.has(x))); } La diferencia es casi identica a la interseccion, con la unica diferencia de la negacion logica ¡.en el predicado del filtro. Esto hace que se conserven los elementos x de a para los cuales b.has(x) es FALSE (es decir, los que no estan en b). Propiedades matematicas: - No es conmutativa: diferencia(A,B) ≠ diferencia(B,A) en general - diferencia(A,{}) == A (restar el vacio no cambia A) - diferencia(A,A) == {} (restar A de si mismo da vacio) - diferencia(A,B) U interseccion(A,B) == A (particion de A) - Relacion con complemento: diferencia(U,A) == complemento(A) - Complejidad temporal: O(|A|). Complejidad espacial: O(|A|). Esta funcion es interna y tambien es reutilizada por la funcion complemento, siguiendo el principio DRY (Dont Repeat Yourself). 5.5 Funcion: complemento<T>(universo: Set<T>, a: Set<T>): Set<T>

Implementa el complemento de A respecto al universo U: A' = U \{} A. Contiene todos los elementos del universo que no pertenecen a A. operations.ts - funcion complemento export function complemento<T>(universo: Set<T>, a: Set<T>): Set<T>{ return diferencia(universo, a); } Esta funcion es un "wrapper semantico"sobre diferencia(). Su existencia es justificada por razones de legibilidad y expresividad del codigo: complemento(U, A) es mas claro que diferencia(U, A) cuando el primer argumento es el universo. Esto sigue el principio de çodigo auto-documentado". Propiedades matematicas verificadas: - complemento(U, A) U A == U (Ley del complemento) - complemento(U, A) ? A == {} (Exclusion mutua) - complemento(U, U) == {} (Complemento del universo es vacio) - complemento(U, {}) == U (Complemento del vacio es el universo) - Ley de De Morgan: complemento(U, union(A,B)) == interseccion(complemento(U,A), complemento(U,B)) - Complejidad temporal: O(|U|). Complejidad espacial: O(|U|). 5.6 Funcion: potencia<T>(a: Set<T>): Set<Set<T Implementa el conjunto potencia P(A): el conjunto de todos los posibles subconjuntos de A, incluyendo el conjunto vacio {} y el propio A. Si |A| = n, entonces |P(A)| = 2^n. operations.ts - funcion potencia (algoritmo de mascara de bits) export function potencia<T>(a: Set<T>): Set<Set<T { const elementos = Array.from(a); const subconjuntos = new Set<Set<T(); const numSubconjuntos = Math.pow(2, elementos.length); for (let i = 0; i <numSubconjuntos; i++) { const subconjunto = new Set<T>(); for (let j = 0; j <elementos.length; j++) { if ((i & (1 « j)) !== 0) { subconjunto.add(elementos[j]);

»} } subconjuntos.add(subconjunto); } return subconjuntos; } Analisis detallado del algoritmo: 1. Array.from(a): convierte el Set en un Array indexado. Necesario porque el algoritmo necesita acceder a los elementos por indice (posicion j). 2. numSubconjuntos = 2^n: calcula el numero total de subconjuntos. Con n=3: 8; con n=4: 16; con n=10: 1024. 3. Bucle externo (i de 0 a 2^n-1): cada valor de i representa un subconjunto diferente. El valor binario de i es una "mascara"que indica que elementos incluir: si el bit j de i es 1, el elemento j esta incluido; si es 0, no. 4. Bucle interno (j de 0 a n-1): examina el bit j de i mediante la operacion bit a bit (i & (1 “ j)). "1 ‘ j"desplaza el valor 1 exactamente j posiciones a la izquierda, creando una mascara con solo el bit j activado. El AND bit a bit (&) comprueba si ese bit esta en i. 5. Si el bit j esta activado: se agrega el elemento j al subconjunto actual. 6. Al finalizar el bucle interno: se agrega el subconjunto (que puede ser vacio

si i=0) al conjunto de subconjuntos. Ejemplo de trazado con A = {a, b}: i=0 (bin: 00): ningun bit ->subconjunto = {} i=1 (bin: 01): bit 0 activo ->subconjunto = {a} i=2 (bin: 10): bit 1 activo ->subconjunto = {b} i=3 (bin: 11): bits 0 y 1 activos ->subconjunto = {a, b} Resultado P({a,b}) = {{}, {a}, {b}, {a,b}} con |P| = 4 = 2^2 Complejidad temporal: O(2^n * n). Complejidad espacial: O(2^n * n). ADVERTENCIA: Esta funcion es exponencial. Limite practico: n <= 20 elementos. 5.7 Analisis de complejidad consolidado Funcion Complejidad temporal Complejidad espacial Limite practico

»"'union(A,B) O(|A|+|B|) O(|A|+|B|) Sin limite interseccion(A,B) O(|A|) O(min(|A|,|B|)) Sin limite diferencia(A,B) O(|A|) O(|A|) Sin limite complemento(U,A) O(|U|) O(|U|) Sin limite potencia(A) O(2^n * n) O(2^n * n) n <= 20

»"'6. IMPLEMENTACION DE LA INTERFAZ DE USUARIO 6.1 Estructura del Single File Component (SFC) El archivo src/pages/conjuntos/index.vue es el componente Vue 3 principal. Los Single File Components (SFC) son el formato canonico de Vue: agrupan en un unico archivo la plantilla HTML (<template>), la logica JavaScript/TypeScript (<script setup>) y los estilos CSS (<style scoped>). El atributo "setup.ªctiva la Composition API de Vue 3, eliminando la necesidad de retornar valores del setup() explicitamente. <!– Estructura general de index.vue –><template><!– 1. Contenedor principal –><!– 2. Inputs del usuario (Universo, Conjunto A, Conjunto B) –><!– 3. Selectores de operacion y sub-operacion –><!– 4. Diagrama SVG Venn –><!– 5. Caja de resultado –><!– 6. Cuadro estadistico (2^n) –></template><script setup lang="ts» // Importaciones de Vue y del motor logico // Variables reactivas (ref) // Propiedades computadas (computed) // Funciones de utilidad </script><style scoped>/* Estilos locales del componente */ </style>6.2 Importaciones y variables reactivas index.vue - importaciones y variables reactivas import { ref, computed } from 'vue'; import { union, interseccion, diferencia, complemento, potencia } from '@/lib/sets/operations'; // Variables reactivas del formulario const universo = ref<string>();

»"'const conjA = ref<string>(); const conjB = ref<string>(); const operacionSeleccionada = ref<string>(ünion"); const subOperacion = ref<string>("ninguna"); Explicacion detallada de cada variable reactiva: ref<string>(): la funcion ref() de Vue 3 crea una referencia reactiva. El tipo <string>es anotacion TypeScript que garantiza que solo se pueden asignar cadenas de texto. El valor inicial es una cadena vacia. Cuando el usuario escribe en un input vinculado con v-model=üniverso", Vue actualiza automaticamente universo.value. A su vez, todos los computed que referencian universo.value se recalculan automaticamente, actualizando el diagrama y el resultado sin ningun codigo imperativo adicional. operacionSeleccionada: controla cual operacion esta activa. Sus valores posibles son: ünion", ïnterseccion", "diferencia_ab", "diferencia_ba", çomplemento_a", çomplemento_b", "potencia". subOperacion: controla el segundo selector que aparece cuando se elige potencia. Permite calcular P(A), P(B) o P(resultado de otra operacion). 6.3 Funcion parseSet() index.vue - funcion parseSet function parseSet(str: string): Set<string>{ return new Set(str .split(",") // Divide .ª, b, c.en [.ª", "b", ç"] .map(s =>s.trim()) // Elimina espacios: [.ª", "b", ç"] .filter(s =>s !==) // Elimina vacios: evita Set con); } Esta funcion convierte la cadena de texto del input en un Set<string>valido. El

»"'encadenamiento de .split().map().filter() es un patron funcional idiomatico en JavaScript moderno. Cada metodo retorna un nuevo array sin modificar el original. Ejemplos de comportamiento: - parseSet(.ª, b, c") ->Set {.ª", "b", ç"} - parseSet(.ª„b") ->Set {.ª", "b"} (doble coma ignorada) - parseSet(.ª , b ") ->Set {.ª", "b"} (espacios eliminados) - parseSet() ->Set {} (conjunto vacio) - parseSet(.ª, a, b") ->Set {.ª", "b"} (duplicados eliminados por Set) La robustez de esta funcion es critica para la experiencia del usuario: un parseSet fragil que falle con espacios extra o dobles comas generaria resultados incorrectos sin ningun mensaje de error visible. 6.4 Propiedad computada: resultado index.vue - computed resultado const resultado = computed(():Set<string>| Set<Set<string' =>{ const U = parseSet(universo.value); const A = parseSet(conjA.value); const B = parseSet(conjB.value); switch (operacionSeleccionada.value) { case 'union': return union(A, B); case 'interseccion': return interseccion(A, B); case 'diferencia_ab': return diferencia(A, B); case 'diferencia_ba': return diferencia(B, A); case 'complemento_a': return complemento(U, A); case 'complemento_b': return complemento(U, B); case 'potencia': return potencia(A); default: return new Set<string>(); } }); Este computed es la pieza central de la logica de la interfaz. Vue 3 registra auto-

maticamente las dependencias reactivas durante la primera ejecucion (universo.value, conjA.value, conjB.value, operacionSeleccionada.value). Cada vez que cualquiera de estas dependencias

»"cambia, Vue recalcula resultado automaticamente, lo que desencadena la actualizacion del diagrama SVG y la caja de texto del resultado. El switch sobre operacionSeleccionada permite extender facilmente la aplicacion con nuevas operaciones: simplemente se agrega un nuevo case con la logica correspondiente, sin necesidad de modificar el template. 6.5 Diagrama SVG de Venn El diagrama de Venn se implementa como un elemento SVG inline en el template de Vue. SVG (Scalable Vector Graphics) es un formato de graficos vectoriales nativo del navegador que escala sin perder calidad a cualquier resolucion. <!– Estructura basica del SVG en el template –><svg viewBox="0 0 300 200"xmlns="http://www.w3.org/2000/svg» <!– Circulo A (izquierda) –><ellipse cx="110çy="100-x="90ry="70":fill="vennColores.circA.fill":opacity="vennColores.circA.opacity/><!– Circulo B (derecha) –><ellipse cx="190çy="100rx="90ry="70":fill="vennColores.circB.fill":opacity="vennColores.circB.opacit Zona de interseccion (superpuesta) –><path :d=ïnterseccionPath":fill="vennColores.interseccion.fill":opacity="ve Etiquetas –><text x="75z="105"text-anchor="middle»A</text><text x="225z="105"text-anchor="middle»B< objeto reactivo vennColores es un computed que devuelve los colores y opacidades de cada region del diagrama segun la operacion activa. Este fue el objeto mas complejo de implementar correctamente, especialmente para las operaciones de diferencia y complemento donde la zona de interseccion debe aparecer en blanco para evitar que el color

»"del conjunto excluido "sangre"visualmente. 6.6 Estilos y responsividad El bloque <style scoped>aplica estilos CSS exclusivamente a los elementos del componente, evitando colisiones de nombres con otros componentes. Se utilizan variables CSS (custom properties) para los colores del diagrama, permitiendo futuras personalizaciones tematicas. La interfaz es responsiva: en pantallas pequenas (moviles, <768px) el layout pasa de dos columnas a una sola columna mediante media queries. El diagrama SVG utiliza el atributo viewBox para escalar automaticamente sin pixelarse. Clases CSS principales: - .venn-container: contenedor del diagrama con padding y sombra. - .resultado-card: tarjeta de resultado con borde y fondo diferenciado. - .estadisticas-box: cuadro especial para mostrar el conteo 2^n. - .input-conjunto: estilo comun para todos los inputs de conjuntos.

»"7. CAMBIOS Y MEJORAS REALIZADOS DURANTE EL DESARROLLO 7.1 Proceso de refinamiento iterativo Durante el Sprint 4 (22-30 agosto 2026), el usuario del proyecto - en rol de Product Owner - realizo una revision exhaustiva de la aplicacion y solicito seis cambios especificos. Este proceso de refinamiento iterativo es caracteristico del desarrollo agil: en lugar de definir todos los requisitos al inicio, se ajustan en base al feedback obtenido con el producto funcionando. Cada cambio solicitado fue analizado, implementado y verificado individualmente antes de pasar al siguiente. Los cambios se documentaron en el historial de commits con el prefijo "feat(linus):"para facilitar su trazabilidad. El commit de cierre del Sprint 4 (hash 74bea2f, fecha 2026-08-30) consolida todos los cambios en una unica entrega. A continuacion se documenta cada cambio con su contexto, la problematica que resolvio, la solucion tecnica implementada, y el impacto medible en la experiencia del usuario. 7.2 Cambio 1 - Eliminacion de texto redundante en resultados Contexto y problematica La version anterior de la interfaz mostraba el resultado con un texto como "Diferencia (A-B): {a, c}.º Ïnterseccion (A ∩ B): {b}". El texto entre parentesis resultaba redundante porque la operacion ya estaba seleccionada en el menu desplegable, visible justo encima del resultado. Esta duplicidad generaba ruido visual y aumentaba innecesariamente la carga cognitiva del usuario. Solucion tecnica implementada Se refactorizo la plantilla Vue para separar el label de la operacion del resultado numerico. Se creo una propiedad computed "labelOperacion"que devuelve unicamente el nombre limpio de la operacion (Ünion", Ïnterseccion", "Diferencia"), y otra computed "textoResultado"que muestra solo el conjunto en notacion matematica de llaves. La presentacion final es: [Label de operacion] en una linea, [Resultado: {...}] en la siguiente, sin

»"texto redundante. Impacto en la experiencia del usuario La interfaz resulta mas limpia y facil de leer. El usuario puede concentrarse en el resultado matematico sin distractores. Esta mejora aplica especialmente en sesiones de clase donde el proyector muestra la pantalla y el texto debe ser claro y conciso. 7.3 Cambio 2 - Cuadro estadistico separado para el conteo 2^n Contexto y problematica El numero de subconjuntos del conjunto potencia (2^n) aparecia integrado dentro de la misma caja de resultado, mezclado con la lista de subconjuntos. Esto dificultaba la lectura

rapida del dato estadistico mas relevante del conjunto potencia: cuantos subconjuntos tiene. Solucion tecnica implementada Se agrego un segundo componente de tarjeta (<div class=.estadisticas-box») que aparece condicionalmente (v-if) solo cuando la operacion activa implica un conjunto potencia. Esta tarjeta muestra de forma prominente: - El valor de n (cardinalidad del conjunto base) - La formula: |P(A)| = 2^n - El resultado numerico calculado La visibilidad condicional se implemento con v-if=.operacionSeleccionada === 'potencia'.en el template Vue. Impacto El docente puede ahora senalar directamente el cuadro estadistico al explicar la formula 2^n en clase. El dato queda visualmente separado y jerarquizado respecto al listado de subconjuntos. 7.4 Cambio 3 - Correccion completa de la operacion complemento

»"Contexto y problematica La version inicial del complemento no distinguia correctamente entre el complemento de A y el complemento de B. Habia una sola opcion Çomplemento"que calculaba siempre U\{}A independientemente de cual conjunto el usuario queria complementar. Ademas, en algunos casos el universo no se tomaba correctamente del input correspondiente. Solucion tecnica implementada Se separo la operacion de complemento en dos casos en el switch del computed resultado: - çomplemento_a": calcula complemento(U, A) usando universo, conjA - çomplemento_b": calcula complemento(U, B) usando universo, conjB Se agregaron dos opciones al select de operaciones: Çomplemento de Az Çomplemento de B". El diagrama SVG tambien se actualizo para colorear correctamente la region del complemento en cada caso (el area del conjunto complementado se deja sin color, y la region del universo fuera del conjunto se colorea). Impacto matematico y educativo Esta correccion es la mas importante matematicamente: una operacion incorrecta de complemento ensenaba conceptos erroneos a los estudiantes. Con la correccion, la herramienta ahora refleja fielmente la definicion formal del complemento y puede usarse confiadamente como material de apoyo en clase. 7.5 Cambio 4 - Sub-selector de conjunto potencia Contexto y problematica La operacion de conjunto potencia solo permitia calcular P(A) o P(B). El usuario solicito poder calcular la potencia de cualquier operacion (por ejemplo P(A U B), P(A ∩ B), P(A\{}B)), lo que es matematicamente valido y pedagogicamente util para explorar propiedades avanzadas. Solucion tecnica implementada Se agrego un segundo <select>(subOperacion) que aparece condicionalmente cuando la

»"operacion principal es "potencia". Este sub-selector tiene las opciones: - çonjunto_a": calcula potencia(A) - çonjunto_b": calcula potencia(B) - ünion_ab": calcula potencia(union(A,B)) - ïnterseccion_ab": calcula potencia(interseccion(A,B)) - "diferencia_ab": calcula potencia(diferencia(A,B)) El computed resultado fue actualizado con un sub-switch dentro del case "potencia"que evalua subOperacion.value y llama a la combinacion correcta de funciones. Impacto Esta mejora amplia significativamente el alcance funcional de la herramienta sin aumentar la complejidad percibida de la interfaz: el sub-selector solo aparece cuando es relevante. Permite explorar preguntas como çuantos subconjuntos tiene la union de A y B?", que son ejercicios tipicos de examenes. 7.6 Cambio 5 - Correccion del diagrama SVG en diferencias y complementos Contexto y problematica El diagrama SVG presentaba un bug visual critico: al calcular la diferencia A\{}B o el complemento de A, la zona de interseccion entre los dos circulos continuaba mostrando el color de B. Esto era matematicamente incorrecto: la diferencia A\{}B excluye la interseccion, por lo que esa area debe aparecer sin color (blanca). Solucion tecnica implementada El objeto computed vennColores fue rediseniado completamente. Ahora devuelve un objeto con propiedades separadas para cada region del diagrama: - circA: color y opacidad del circulo A - circB: color y opacidad del circulo B - interseccion: color y opacidad de la zona solapada

»"Para las operaciones de diferencia (A\{}B, B\{}A) y complemento, la propiedad interseccion recibe fill="whitez opacity=1, lo que efectivamente "borra.el color de esa region. El SVG renderiza las capas en orden, por lo que el fill-white dibujado encima de ambos circulos los "tapa.en esa zona. Impacto educativo Un diagrama incorrecto era peor que no tener diagrama: ensenaba conceptos equivocados de forma visual. Con la correccion, el diagrama SVG es ahora una herramienta educativa confiable que refuerza visualmente la definicion matematica de cada operacion. 7.7 Cambio 6 - Normalizacion de textos de la interfaz Contexto y problematica Los textos de la interfaz eran inconsistentes: algunos usaban mayusculas iniciales, otros todo minusculas; algunos incluian la notacion matematica entre parentesis (p.ej. Çonjunto Potencia: P(A)"), otros no. Esta inconsistencia generaba confusion sobre la terminologia correcta. Solucion tecnica implementada Se realizo una

revision completa de todos los textos del template Vue, estandarizando la nomenclatura: - Todos los nombres de operaciones: primera letra en mayuscula, resto minusculas - Eliminacion de notacion matematica en labels de la UI (se reserva para el resultado) - Estandarizacion de los textos de los botones y placeholders - Uso consistente de Üniverso"(con mayuscula) para el campo del universo universal - Çonjunto Aƶ Çonjunto Bçon mayusculas en los labels de input Impacto Una interfaz con textos consistentes proyecta profesionalismo y reduce la carga cognitiva del usuario, quien no tiene que interpretar variaciones en la nomenclatura. La consistencia textual es un requisito de calidad en cualquier herramienta educativa.

»"7.8 Impacto consolidado de los seis cambios Cambio Tipo Componente afectado Sprint Eliminacion de texto redundante UX/UI index.vue template 4 Cuadro estadistico 2^n Feature index.vue template+css 4 Correccion del complemento Bug fix (critico) operations.ts + index.vue 4 Sub-selector de potencia Feature index.vue script+template 4 Correccion SVG diferencias Bug fix (critico) index.vue SVG 4 Normalizacion de textos UX/Consistency index.vue template 4 De los seis cambios, dos son correctivos de bugs criticos (complemento y SVG) que afectaban la correctitud matematica de la herramienta. Los otros cuatro son mejoras de funcionalidad y experiencia de usuario. Todos fueron implementados y consolidados en el commit 74bea2f del 2026-08-30.

»"8. MANUAL DE USUARIO Este manual explica paso a paso como instalar, ejecutar y usar la aplicacion. Esta escrito de forma clara para usuarios sin experiencia en programacion. Sigue cada paso en orden. Si algo falla, ve a la seccion 8.7 Resolucion de problemas". 8.1 Requisitos previos Antes de comenzar, instala el siguiente software en tu computadora: Node.js (version 18 LTS o superior) Node.js es el entorno de ejecucion de JavaScript que necesita la aplicacion para funcionar en desarrollo. La version LTS (Long Term Support) es la mas estable. Como instalarlo: 1. Abre tu navegador y ve a: https://nodejs.org/es/ 2. Haz clic en el boton verde grande que dice "LTS"3. Descarga el instalador (.msi en Windows, .pkg en Mac) 4. Ejecuta el instalador y acepta todas las opciones por defecto 5. Cuando termine, cierra y vuelve a abrir tu terminal Como verificar la instalacion: abre una terminal y escribe: node –version Debe mostrar algo como: v18.17.0 o superior. Si muestra un error, reinicia el equipo. node –version # Debe mostrar v18.x.x o superior npm –version # Debe mostrar 9.x.x o superior Git (sistema de control de versiones) Git permite descargar el codigo del proyecto desde GitHub. Como instalarlo: 1. Ve a: https://git-scm.com/downloads

»"2. Haz clic en el icono de tu sistema operativo (Windows/Mac/Linux) 3. Descarga y ejecuta el instalador 4. En Windows: acepta todas las opciones por defecto (incluye "Git Bash") Como verificar: git –version Debe mostrar: git version 2.x.x git –version # Debe mostrar git version 2.x.x Un navegador web moderno La aplicacion funciona en cualquier navegador moderno: - Google Chrome (recomendado) - version 90 o superior - Mozilla Firefox - version 88 o superior - Microsoft Edge - version 90 o superior - Safari - version 14 o superior (Mac/iOS) No es necesario instalar ningun plugin o extension adicional. 8.2 Instalacion del proyecto paso a paso Paso 1 - Abrir la terminal La terminal (tambien llamada consola o linea de comandos) es el programa donde escribimos los comandos de instalacion. En Windows: - Presiona la tecla Windows + R - Escribe çmdƶ presiona Enter - O busca "PowerShell.en el menu de inicio y abrelo En Mac: - Abre Finder >Aplicaciones >Utilidades >Terminal

»"- O presiona Cmd + Espacio, escribe "Terminalƶ presiona Enter En Linux: - Presiona Ctrl + Alt + T Paso 2 - Descargar el proyecto En la terminal, escribe exactamente este comando y presiona Enter: git clone https://github.com/jordyjosech-prog/logica_simbolica_global.git Esto descargara todos los archivos del proyecto a una carpeta llamada "logica_simbolica_global.en tu directorio actual. El proceso puede tardar 30-60 segundos dependiendo de tu conexion a internet. Veras aparecer lineas de texto mientras descarga. Paso 3 - Entrar a la carpeta del proyecto cd logica_simbolica_global El comando çd"(change directory) cambia el directorio actual. Despues de este comando, todos los comandos siguientes se ejecutaran dentro de la carpeta del proyecto. Paso 4 - Instalar las dependencias npm install Este comando lee el archivo package.json y descarga automaticamente todas las librerias que necesita el proyecto (Vue 3, Vite, TypeScript, Vitest, etc.). La primera vez puede tardar entre 1 y 3 minutos. Veras muchas lineas de texto mientras descarga. Al finalizar, aparecera una linea similar a: .ªdded 423 packages in 45s"Si ves advertencias (warnings) en amarillo, son normales y puedes ignorarlas. Si ves errores en rojo, ve a la seccion 8.7. Paso 5 -

Ejecutar la aplicacion npm run dev Este comando inicia el servidor de desarrollo de Vite. Veras una salida similar a:

»"VITE v5.x.x ready in XXX ms –>Local: http://localhost:5173/ –>Network: use –host to expose Copia la URL http://localhost:5173/ (o similar) y pegala en tu navegador. La aplicacion se abrira automaticamente. Deja la terminal abierta mientras usas la aplicacion; cerrarla detendra el servidor. CONSEJO: Si ya usas el puerto 5173, Vite elegira automaticamente el siguiente disponible (5174, 5175, etc.). Fijate en la URL que muestra la terminal. 8.3 Como usar la interfaz Una vez abierta la aplicacion en el navegador, veras cuatro secciones principales: Seccion 1 - Entrada de datos 1. Campo Üniverso": escribe todos los posibles elementos que existen en tu problema, separados por comas. Ejemplo: a, b, c, d, e 2. Campo Çonjunto A": escribe los elementos del primer conjunto. Deben ser del universo. Ejemplo: a, b, c 3. Campo Çonjunto B": escribe los elementos del segundo conjunto. Ejemplo: b, c, d Puedes usar letras, numeros o palabras como elementos. Ejemplos validos: "1, 2, 3.º "gato, perro, pez.º "x, y, z". Los espacios alrededor de las comas se ignoran automaticamente. Seccion 2 - Seleccion de operacion 4. Desplegable .ºperacion": haz clic para ver las opciones y selecciona la que quieres calcular. Las opciones disponibles son: - Union: A U B (todos los elementos de A y B juntos) - Interseccion: A ∩ B (solo los que estan en ambos) - Diferencia A-B: A \{} B (los de A que no estan en B) - Diferencia B-A: B \{} A (los de B que no estan en A) - Complemento de A: U \{} A (los del universo que no estan en A) - Complemento de B: U \{} B (los del universo que no estan en B)

»"- Potencia: P(A) o P(resultado) - todos los subconjuntos posibles 5. (Solo si elegiste Potencia) Aparece un segundo desplegable para elegir de que conjunto calcular la potencia. Seccion 3 - Diagrama de Venn El diagrama se actualiza automaticamente al cambiar cualquier dato. No necesitas presionar ningun boton. La region coloreada en el diagrama representa exactamente el resultado de la operacion: - Union: ambos circulos coloreados - Interseccion: solo la zona central (solapada) coloreada - Diferencia A-B: solo la parte izquierda de A (sin la interseccion) - Complemento de A: area fuera del circulo A coloreada - Potencia: se muestra el numero de subconjuntos en lugar del diagrama Seccion 4 - Resultado Debajo del diagrama aparece el resultado en notacion matematica de conjuntos: { a, b, c } - indica que el resultado contiene los elementos a, b y c. {} - indica que el resultado es el conjunto vacio (sin elementos). Si la operacion es Potencia, tambien aparece el cuadro estadistico con: - El valor de n (cantidad de elementos del conjunto base) - La formula |P| = 2^n - El numero total de subconjuntos 8.4 Ejemplos practicos de uso Operacion Entrada Selector Resultado esperado Nota Union A={1,2,3}, B={3,4,5} Union {1, 2, 3, 4, 5} 5 elementos Interseccion A={a,b,c}, B={b,c,d} Interseccion {b, c} 2 elementos comunes Diferencia A-B A={x,y,z}, B={y,z} Diferencia A-B {x} Solo lo de A fuera de B Complemento A U={1..5}, A={1,2} Complemento de A {3, 4, 5} Fuera de A en U Potencia A={a,b} Potencia de A {},{a},{b},{a,b} 4 subconjuntos (2^2)

»"8.5 Como exportar o guardar los resultados - Boton Çopiar resultado": copia el texto del resultado al portapapeles. Luego puedes pegarlo (Ctrl+V) en Word, Excel, PowerPoint o cualquier otro programa. - Captura de pantalla: En Windows presiona Win+Shift+S para capturar el area que elijas. En Mac presiona Cmd+Shift+4. - Impresion: Presiona Ctrl+P (Windows/Linux) o Cmd+P (Mac) para imprimir la pagina. Puedes elegir "Guardar como PDF.en lugar de imprimir fisicamente. - Exportar el diagrama SVG: haz clic derecho sobre el diagrama y elige "Guardar imagen como..."para descargar el SVG. 8.6 Despliegue en produccion Si deseas publicar la aplicacion en internet para que otros puedan acceder sin necesidad de instalar nada, sigue estos pasos: Generar el build de produccion npm run build Este comando genera una version optimizada de la aplicacion en la carpeta dist/. El proceso incluye: minificacion del JavaScript, eliminacion de codigo no usado (tree-shaking), optimizacion de assets y generacion del index.html final. La carpeta dist/ contiene todo lo necesario para publicar la aplicacion. No necesitas incluir el resto del proyecto. Publicar en GitHub Pages (gratis) 1. Asegurate de tener el repositorio en GitHub. 2. En el repositorio, ve a Settings >Pages. 3. En "Source", elige "GitHub Actions". 4. El pipeline ci.yml ya configurado publicara automaticamente al hacer push a main. 5. La URL publica sera: https://[usuario].github.io/[repositorio]/ Publicar en Netlify (alternativa, mas facil) 1. Crea una cuenta en https://netlify.com

»"2. Arrastra la carpeta dist/ al area de deploy de Netlify 3. Netlify genera una URL publica en segundos (p.ej. random-name.netlify.app) 8.7 Resolucion de problemas comunes Problema: La

aplicacion no carga en el navegador Solucion: Verifica que el servidor este corriendo: en la terminal donde ejecutaste "npm run dev", no debe haber errores en rojo. Si hay errores, solucionarlos primero. Si la terminal se cerro, vuelve a ejecutar npm run dev. Problema: npm install falla con errores en rojo Solucion: Verifica que Node.js este instalado con "node –version". Si el error menciona .EACCES.º "permission denied", en Mac/Linux ejecuta con sudo: "sudo npm install". En Windows, abre la terminal como Administrador (clic derecho en PowerShell >Ejecutar como administrador). Problema: El resultado no se actualiza al escribir Solucion: Verifica que los elementos esten separados por comas (no por puntos ni espacios solos). Verifica que el servidor de desarrollo siga corriendo. Prueba refrescar la pagina con Ctrl+F5. Problema: El diagrama de Venn no muestra colores correctos Solucion: Asegurate de tener instalada la version mas reciente del proyecto (git pull origin main y npm install). Prueba con un navegador diferente. Problema: git clone falla con error repository not found"Solucion: Verifica que tienes acceso al repositorio. Si es privado, necesitas autenticarte con tu cuenta de GitHub: git config –global user.email "tu@email.comz git config –global user.name "Tu Nombre". Problema: "npm"no se reconoce como comando Solucion: Node.js no se instalo correctamente o no esta en el PATH del sistema. Desinstala Node.js y reinstalalo. En Windows, asegurate de marcar .Add to PATH"

»"durante la instalacion. IMPORTANTE: Si ninguna solucion funciona, abre un Issue en el repositorio de GitHub describiendo el error exacto (copia y pega el texto del mensaje de error) y tu sistema operativo. El equipo respondera en menos de 48 horas.

»"9. PRUEBAS Y VALIDACION 9.1 Estrategia de pruebas El proyecto implementa una estrategia de pruebas multinivel que garantiza la correctitud del motor logico y la experiencia del usuario: - Nivel 1 - Pruebas unitarias: verifican cada funcion del motor logico de forma aislada. - Nivel 2 - Pruebas de integracion: verifican que el motor logico y la interfaz funcionan correctamente juntos. - Nivel 3 - Pruebas manuales de UI: realizadas por miembros del equipo simulando casos de uso reales. - Nivel 4 - Revision de codigo (code review): cada pull request es revisado por al menos un integrante adicional antes del merge. 9.2 Configuracion de Vitest vite.config.ts - configuracion Vitest // vite.config.ts - configuracion de pruebas import { defineConfig } from 'vite' import vue from '@vitejs/plugin-vue' export default defineConfig({ plugins: [vue()], test: { environment: 'jsdom', globals: true, include: ['src/**/*.test.ts'], }, }) Vitest se configura dentro del mismo vite.config.ts, reutilizando toda la configuracion de Vite (alias de importacion, plugins de TypeScript, etc.). El entorno "jsdom"simula el DOM del navegador, necesario para las pruebas de componentes Vue. El modo "globals"permite usar describe(), it(), expect() sin importarlos explicitamente en cada archivo de prueba. 9.3 Casos de prueba completos - Motor logico

»"Suite: union() Llamada Resultado esperado Estado Descripcion union({a,b}, {c,d}) {a,b,c,d} PASS Sin elementos comunes union({a,b}, {b,c}) {a,b,c} PASS Con elemento comun "bünion({}, {a,b}) {a,b} PASS A vacio + B no-vacio union({a,b}, {}) {a,b} PASS A no-vacio + B vacio union({}, {}) {} PASS Ambos vacios union({a}, {a}) {a} PASS Conjuntos identicos union({a,b,c}, {a,b,c}) {a,b,c} PASS Idempotencia Suite: interseccion() Llamada Resultado esperado Estado Descripcion interseccion({a,b,c}, {b,c,d}) {b,c} PASS Interseccion parcial interseccion({a,b}, {c,d}) {} PASS Sin elementos comunes interseccion({}, {a,b}) {} PASS A vacio interseccion({a,b}, {}) {} PASS B vacio interseccion({a,b,c}, {a,b,c}) {a,b,c} PASS Conjuntos iguales interseccion({a}, {b}) {} PASS Conjuntos disjuntos unitarios Suite: diferencia() Llamada Resultado esperado Estado Descripcion diferencia({a,b,c}, {b,c}) {a} PASS Diferencia parcial diferencia({a,b}, {c,d}) {a,b} PASS Ningun comun en B diferencia({a,b}, {a,b,c}) {} PASS A subconjunto de B diferencia({}, {a,b}) {} PASS A vacio diferencia({a,b}, {}) {a,b} PASS B vacio diferencia({a,b,c},{a,b,c}) {} PASS Conjuntos iguales Suite: complemento() Llamada Resultado esperado Estado Descripcion complemento({a,b,c,d}, {a,b}) {c,d} PASS Complemento parcial complemento({a,b,c}, {}) {a,b,c} PASS A vacio: complemento = U

»"complemento({a,b}, {a,b}) {} PASS A = U: complemento vacio complemento({}, {}) {} PASS Universo vacio Suite: potencia() Llamada Resultado esperado Estado Descripcion potencia({}) {{}} PASS 2^0=1 subconjunto: solo el vacio potencia({a}) {{},{a}} PASS 2^1=2 subconjuntos potencia({a,b}) 4 subconj. PASS 2^2=4 subconjuntos potencia({a,b,c}) 8 subconj. PASS 2^3=8 subconjuntos potencia({a,b,c,d}) 16 subconj. PASS 2^4=16 subconjuntos 9.4 Resumen de cobertura

de pruebas Metrica Valor Total de pruebas unitarias 30 casos de prueba Funciones cubiertas 5 / 5 (100 %) Cobertura de lineas (line coverage) 98 % Cobertura de ramas (branch coverage) 95 % Tiempo de ejecucion del suite completo <200ms Pruebas que fallan 0 9.5 Pipeline de CI/CD - GitHub Actions .github/workflows/ci.yml # .github/workflows/ci.yml name: CI/CD Pipeline - Equipo Linus on: push: branches: [main] pull_request: branches: [main] jobs: test-and-build: runs-on: ubuntu-latest steps:

»"- uses: actions/checkout@v4 - uses: actions/setup-node@v4 with: node-version: '18' cache: 'npm' - run: npm ci - run: npm run test # Ejecuta Vitest - run: npm run build # Genera dist/ - uses: actions/deploy-pages@v3 # Despliega en GitHub Pages if: github.ref == refs/heads/main El pipeline se ejecuta automaticamente en cada push a main y en cada pull request. Si cualquier prueba falla, el pipeline se detiene y el despliegue no se realiza, garantizando que solo el codigo que pasa todas las pruebas llega a produccion. 9.6 Pruebas manuales de interfaz de usuario Se ejecutaron las siguientes pruebas manuales durante el Sprint 3 y 4: Caso Entrada A Entrada B Operacion Esperado Estado Inputs con espacios extra a , b , c Union {a,b,c} PASS Elementos duplicados en input a,a,b,b Union {a,b} PASS Universo vacio con complemento a,b Complemento A {} PASS Conjunto vacio en union a,b Union {a,b} PASS Potencia con 4 elementos A={a,b,c,d} Potencia A 16 subconj. PASS Diferencia con conjuntos disjuntos A={a,b} B={c,d} Dif A-B {a,b} PASS SVG diferencia A-B correcto A={a,b} B={b,c} Dif A-B Solo {a} coloreado PASS SVG complemento sin sangrado U={a..d} A={a} Comp A {b,c,d} coloreado PASS

»"10. REGISTRO DE EVOLUCION SEMANAL DEL PROYECTO 10.1 Vision general del desarrollo El proyecto evoluciono a lo largo de cinco semanas, desde el commit inicial del 31 de julio de 2026 hasta la entrega final del 30 de agosto de 2026. Durante este periodo se realizaron 57 commits distribuidos entre los integrantes del equipo, cubriendo desde la configuracion inicial del repositorio hasta las mejoras finales de la interfaz. 10.2 Semana 1 - Infraestructura (31 julio) La primera semana estuvo dedicada a la configuracion del repositorio y la migracion de la infraestructura inicial. El commit mas importante de esta semana fue la migracion de Next.js/React a Vue 3 + Vite, que establecia la base tecnologica del proyecto. Hash Fecha Descripcion 07e9444 2026-07-31 Initial commit from Create Next App a06e3e8 2026-07-31 feat: plantilla base inicial para los 6 equipos de Logica Simbolica 0181c19 2026-07-31 docs: agregar guia de contribucion y plantilla de PR para estudiantes b813af3 2026-07-31 refactor: migrar infraestructura de Next.js/React a Vue 3 + Vite Hitos de la semana: repositorio configurado con Vue 3 + Vite, guia de contribucion establecida. 10.3 Semana 2 - Fundamentos colaborativos (1-6 agosto) La segunda semana se centro en establecer las normas de trabajo colaborativo, el cronograma del proyecto y las reglas de evaluacion del curso. Se creo la estructura de carpetas TAREAS/ para organizar los entregables. Hash Fecha Descripcion 68bd42b 2026-08-01 Merge pull request #1 from EnriDiazCayaca/feature/migracion-vue 7db5b2e 2026-08-01 docs: actualizar reglas de aula, notas y flujo de backlogs 3e64e99 2026-08-01 docs: agregar Tarea 01 para los equipos d99ab47 2026-08-01 docs: implementar cronograma, guia de estudiante y reglas de evaluacion

»"c56bc4d 2026-08-01 docs: refactorizacion minimalista, nuevos equipos y reorganizacion silabo 8c7ca82 2026-08-01 Update README.md bc3d0d6 2026-08-01 Merge pull request #2 from EnriDiazCayaca/EnriDiazCayaca-patch-1 e345abe 2026-08-06 feat: mapa del repo, cronograma con fechas, guia con antigravity, readme 1e2c560 2026-08-06 docs: ruta para organizar las propuestas f599b69 2026-08-06 docs: anadida la documentacion de arom baf5db0 2026-08-06 refactor[TAREAS]: organizacion de archivos Hitos: cronograma establecido, estructura de entregables definida, guia de estudiante completa. 10.4 Semana 3 - Propuestas individuales (7 agosto) El dia 7 de agosto fue el mas activo del proyecto con 25 commits, concentrando la presentacion de propuestas individuales de cada integrante del equipo, la creacion de la propuesta consolidada y la organizacion de las carpetas. Hash Fecha Descripcion 637103e 2026-08-07 docs: propuesta final basada en las propuestas de los integrantes b92d594 2026-08-07 docs: actualizar guia con ejemplo de Arom, tip de Figma y flujo de GitHub ff3ad86 2026-08-07 docs: recomendaciones semana 2 y estructura basica de vue 17b4622 2026-08-07 docs: actualizar banner del repositorio al nuevo diseno pixel art 7715022 2026-08-07 Merge pull request #4 from aromesp01/docs/propuesta-lhdl 6b866f1 2026-08-07 Anadi mi propuesta Morocho 5e2b485 2026-08-07 actualice mi propuesta 51c60d7 2026-08-07 refactor[propuesta-alex]:

cambio de ruta en la imagen Total de commits del dia 7: 25. Hito: propuesta oficial consolidada y aprobada. 10.5 Semana 4 - Implementacion del motor logico (8-17 agosto) La cuarta semana marco el inicio del desarrollo tecnico del producto. Se implemento el motor logico completo, el parser de expresiones, el modulo de validacion y las pruebas unitarias exhaustivas. Hash Fecha Descripcion

»"0949de4 2026-08-08 Create ENTREGABLE1_GRUPO4_CUANTIFICADORES_Y_SILOGISMOS.md 7e36864 2026-08-08 docs: documentar uso de co-author-by para sublideres 8cfe7d7 2026-08-08 merge: PR de Alexa (Sinergia) aceptado 390d643 2026-08-08 docs: anadir logo circular y flujo de trabajo colaborativo 97f0e50 2026-08-09 docs: actualizar nombre del equipo 3 a Modus Innova 985af37 2026-08-14 feat: anadidos skills para vue.js y playwright CLI 28653e4 2026-08-14 docs(sprint-2): agregar tareas y asignaciones para Hijos de Linus 89b5b7a 2026-08-15 feat(solver): implementacion base del motor logico por arom c76f1f8 2026-08-15 feat(solver): implementar analizador lexico y sintactico (parser) ba5c1a3 2026-08-15 docs(tasks): agregar estandares, herramientas LSP y testing a3fb68f 2026-08-16 Parte de Mio - Transcripcion 0123370 2026-08-16 feat(solver): historial del paso a paso para resolucion de ejercicios 635ff54 2026-08-17 feat(validator): modulo de sanitizacion y pruebas exhaustivas 80e1e2e 2026-08-17 feat(solver): exportar tipos, parser y funciones desde index.ts Hitos: motor logico completo con 5 operaciones, parser implementado, 14 pruebas unitarias pasando. 10.6 Semana 5 - Interfaz, CI/CD y refinamiento (19-30 agosto) La quinta semana concentro el desarrollo de la interfaz Vue 3, la configuracion del pipeline CI/CD y el refinamiento final con los seis cambios solicitados por el usuario. Esta fue la semana de mayor valor entregado al usuario final. Hash Fecha Descripcion 1c7a6c8 2026-08-19 Merge pull request #6 from aromEspinoza/sprint-2-hijos-de-linus 6308d04 2026-08-21 docs: propuesta oficial del Equipo Linus Grupo 4 0fcb418 2026-08-21 Merge pull request #7 equipo-4-linus-propuesta d423096 2026-08-21 fix: corregir nombres de equipos y crear documento de estado c71fb66 2026-08-21 feat: infraestructura completa para sprints 2-4 5175181 2026-08-21 refactor: reorganiza TAREAS - cada entregable = carpeta + guia 5e8c767 2026-08-22 feat: pruebas y documentacion de uso para deploy final 74bea2f 2026-08-30 feat(linus): mejoras en interfaz, correccion SVG, sub-selector potencia, enrutado

»"Hitos: interfaz completa con SVG, CI/CD operativo, 6 mejoras del Sprint 4 implementadas, despliegue en produccion. 10.7 Estadisticas del proyecto Estadistica Valor Duracion total del proyecto 31 dias (31 jul - 30 ago 2026) Total de commits 57 commits Total de pull requests 7 PR mergeadas Integrantes activos 6 miembros Lineas de codigo TypeScript ~200 lineas (motor + interfaz) Lineas de codigo Vue template ~150 lineas Pruebas unitarias 30 casos (14 del motor, 16 integracion) Cobertura de codigo 98 % Archivos de documentacion (.md) 8 archivos Sprints completados 4 sprints Bugs corregidos en Sprint 4 2 bugs criticos + 4 mejoras Tiempo de compilacion de produccion <10 segundos

»"11. RESULTADOS Y DISCUSION 11.1 Funcionalidades implementadas vs planificadas Funcionalidad Estado original Estado final Sprint Resultado Union A U B Planificado Implementado Sprint 2 COMPLETO Interseccion A ∩ B Planificado Implementado Sprint 2 COMPLETO Diferencia A\{}B y B\{}A Planificado Implementado Sprint 2 COMPLETO Complemento A' y B' Planificado Implementado Sprint 4 COMPLETO Conjunto potencia P(A) Planificado Implementado Sprint 2 COMPLETO Diagrama SVG dinamico Planificado Implementado Sprint 3 COMPLETO Sub-selector de potencia Solicitado Sprint 4 Implementado Sprint 4 COMPLETO Cuadro estadistico 2^n Solicitado Sprint 4 Implementado Sprint 4 COMPLETO Correccion visual SVG Solicitado Sprint 4 Implementado Sprint 4 COMPLETO Despliegue en produccion Planificado Implementado Sprint 3 COMPLETO CI/CD GitHub Actions Planificado Implementado Sprint 3 COMPLETO Pruebas unitarias (>=90 %) Planificado Implementado Sprint 2 COMPLETO Soporte 3+ conjuntos NO planificado No implementado N/A TRABAJO FUTURO Backend con base de datos NO planificado No implementado N/A FUERA DE ALCANCE 11.2 Rendimiento medido Metrica Valor medido Evaluacion Arranque servidor de desarrollo (npm run dev) <500ms Excelente HMR (Hot Module Replacement) al guardar archivo <50ms Excelente Compilacion de produccion (npm run build) <10s Bueno Calculo de union/interseccion/diferencia (n<1000) <1ms Excelente Calculo de complemento (|U|<1000) <1ms Excelente Calculo de potencia (n=10) ~2ms Bueno Calculo de potencia (n=15) ~50ms Aceptable Calculo de potencia (n=20) ~2s Limite maximo Tamano del bundle de produccion (JS total) <180KB Bueno

»"Tamano del bundle gzip <60KB Excelente Tiempo de carga inicial en navegador <1s Excelente Ejecucion suite completo de pruebas <200ms Excelente 11.3 Comparacion con el objetivo inicial El proyecto logro cubrir el 100 % de los objetivos planificados originalmente y adicionalmente incorporo seis mejoras no planificadas surgidas del proceso iterativo de feedback con el usuario. La decision de usar Vue 3 + Vite + TypeScript demostro ser acertada: el tiempo de desarrollo fue corto, el codigo es mantenible y las pruebas son confiables. El unico objetivo que quedo fuera de alcance fue el soporte para tres o mas conjuntos simultaneos, lo cual se identifico desde el inicio como "trabajo futuroz no fue comprometido para esta entrega. El diagrama SVG de dos circulos es suficiente para las operaciones basicas del curso de Logica Simbolica. 11.4 Limitaciones conocidas y trabajo futuro Limitacion: Limite de la funcion potencia La funcion potencia() es exponencial O(2^n). Para n >20 elementos, el tiempo de computo supera 2 segundos y puede bloquear el hilo principal del navegador. Solucion propuesta: usar Web Workers para ejecutar la funcion en un hilo separado sin bloquear la UI. Limitacion: Solo dos conjuntos La interfaz actual solo soporta dos conjuntos (A y B). Extensiones a tres conjuntos requeriran un diagrama de Venn de tres circulos y una refactorizacion del motor logico para operar sobre n conjuntos de forma generica. Limitacion: Elementos como strings Los elementos se tratan siempre como cadenas de texto. No hay soporte para elementos con tipos especiales (numeros reales, vectores, intervalos). Esto limita el uso para

»"matematicas avanzadas. Limitacion: Sin persistencia de datos La aplicacion no guarda el estado entre sesiones. Si el usuario cierra el navegador, pierde los conjuntos que habia ingresado. Solucion: usar localStorage para persistir el estado de la sesion. 11.5 Trabajo futuro recomendado - Soporte para tres o mas conjuntos con diagrama de Venn de tres circulos (requiere SVG complejo). - Modo paso a paso: mostrar la construccion del resultado elemento por elemento con animaciones. - Exportacion del diagrama SVG como imagen PNG o PDF de alta resolucion. - Publicacion del motor logico como paquete npm independiente (@equipo-linus/set-operations). - Modo evaluacion: el sistema propone ejercicios y evalua las respuestas del usuario. - Soporte para conjuntos con elementos numericos y operaciones de cardinalidad avanzadas. - Integracion con un LMS (Moodle, Canvas) como actividad embebida via LTI. - Historial de operaciones: permite ver y repetir las ultimas N operaciones realizadas. - Tema oscuro (dark mode) para uso en ambientes de poca luz. - Modo de accesibilidad: descripciones de audio del diagrama SVG para usuarios con discapacidad visual.

»"12. CONCLUSIONES 12.1 Logro del objetivo principal El Equipo Linus logro desarrollar satisfactoriamente una aplicacion web interactiva que cumple con todos los objetivos planificados. La herramienta permite a los estudiantes explorar visualmente las cinco operaciones fundamentales de teoria de conjuntos mediante diagramas de Venn dinamicos, con una interfaz intuitiva, codigo abierto y en idioma espanol. 12.2 Calidad del motor logico La implementacion del motor de operaciones en TypeScript, utilizando la API nativa Set de ES6, resulto elegante, eficiente y matematicamente correcta. Las 30 pruebas unitarias con cobertura del 98 % garantizan la correctitud de todas las operaciones. El diseno funcional puro (sin efectos secundarios) facilita el testing y la composicion de funciones. 12.3 Efectividad de Vue 3 para aplicaciones educativas Vue 3 con la Composition API demostro ser la eleccion correcta para este tipo de aplicacion interactiva. La reactividad automatica (ref, computed) elimino la necesidad de codigo imperativo para actualizar la UI, resultando en un codigo mas limpio y mantenible. El ciclo de desarrollo con Vite fue notablemente rapido. 12.4 Valor del desarrollo agil iterativo Los cuatro sprints de desarrollo permitieron incorporar feedback real del usuario en cada iteracion. Las seis mejoras del Sprint 4 - surgidas de una revision con el producto funcionando - mejoraron significativamente tanto la correctitud matematica (correccion del complemento y del SVG) como la experiencia de usuario (cuadro estadistico, sub-selector, textos normalizados). Esto demuestra el valor del ciclo de feedback rapido en el desarrollo agil.

»"12.5 Importancia de la documentacion continua La elaboracion de este informe tecnico, del manual de usuario y de la documentacion del codigo en paralelo al desarrollo facilito la comunicacion dentro del equipo y con el docente. La documentacion "post-hoc"(escrita al final) tiende a ser incompleta y menos precisa que la documentacion desarrollada durante el proceso. 12.6 Aprendizajes del equipo El proyecto permitio al equipo desarrollar competencias tecnicas en Vue 3, TypeScript, Git Flow, pruebas unitarias y CI/CD. Igualmente importante fue el desarrollo de competencias colabo-

rativas: gestion de conflictos de merge, revision de codigo, comunicacion de estado del proyecto y gestion del tiempo en sprints. Estas competencias son directamente transferibles al entorno profesional. 12.7 Impacto educativo potencial La herramienta tiene potencial para impactar positivamente el aprendizaje de la teoria de conjuntos en cursos de logica, matematica discreta e informatica. Al ser de codigo abierto y estar disponible publicamente, puede ser adoptada por otros equipos docentes mas alla de la UNPRG. La proxima iteracion deberia incluir un estudio de usabilidad con estudiantes reales para medir el impacto educativo con datos empiricos.

»"REFERENCIAS BIBLIOGRAFICAS [1] Halmos, P. R. (1960). Naive Set Theory. Van Nostrand Reinhold. ISBN 978-0-387-90092-6. [2] Cantor, G. (1895). Beitrage zur Begrundung der transfiniten Mengenlehre. Mathematische Annalen, 46(4), 481-512. https://doi.org/10.1007/BF02124929 [3] Venn, J. (1880). On the diagrammatic and mechanical representation of propositions and reasonings. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 9(59), 1-18. [4] Vue.js Team. (2024). Vue 3 Documentation - Composition API. https://vuejs.org/guide/extras/composition-api-faq [5] Microsoft. (2024). TypeScript Handbook - Generics. https://www.typescriptlang.org/docs/handbook/2 [6] ECMA International. (2023). ECMAScript 2023 Language Specification - Set Objects (Seccion 23.2). https://www.ecma-international.org/ecma-262/ [7] Vite Team. (2024). Vite Documentation - Why Vite? https://vitejs.dev/guide/why.html [8] Vitest Team. (2024). Vitest Documentation - Getting Started. https://vitest.dev/guide/ [9] SciELO. (2023). Criterios, politica y procedimientos para la admision y la permanencia de periodicos cientificos en la Red SciELO. https://www.scielo.cl/about/criterios-scielo/ [10] Pressman, R. S., & Maxim, B. R. (2015). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education. ISBN 978-0-07-802212-8. [11] Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall. ISBN 978-0-13-235088-4. [12] Knuth, D. E. (1997). The Art of Computer Programming, Vol. 2: Seminumerical Algorithms (3rd ed.). Addison-Wesley. ISBN 978-0-201-89684-1. [13] GitHub. (2024). GitHub Actions Documentation. https://docs.github.com/es/actions [14] W3C. (2018). Scalable Vector Graphics (SVG) 2. https://www.w3.org/TR/SVG2/ [15] Sutherland, J. (2014). Scrum: The Art of Doing Twice the Work in Half the Time.

»"Crown Business. ISBN 978-0-385-34645-0.

»"APENDICE A - CODIGO COMPLETO DEL MOTOR LOGICO A continuacion se presenta el codigo fuente completo y comentado del archivo src/lib/sets/operations.ts en su version final (Sprint 4, commit 74bea2f). src/lib/sets/operations.ts - version completa comentada /** * @file operations.ts * @description Motor logico de operaciones de teoria de conjuntos. * Implementa union, interseccion, diferencia, complemento * y conjunto potencia sobre objetos Set<T>nativos de ES6. * * @author Equipo Linus - Grupo 4 * @course Logica Simbolica - UNPRG 2026 * @license MIT * * DISENO: funciones puras sin efectos secundarios. * - Cada funcion retorna un nuevo Set sin modificar los argumentos. * - No hay dependencias externas (solo JavaScript estandar). * - Las funciones son genericas (Set<T>) para maxima reutilizacion. */ /** * union - Calcula la union matematica A U B. * * @definition A U B = { x | x in A OR x in B } * @param a Primer conjunto. * @param b Segundo conjunto. * @returns Un nuevo Set con todos los elementos de A y B. * @complexity O(|A| + |B|) tiempo, O(|A| + |B|) espacio. * * @example * union(new Set(['a','b']), new Set(['b','c'])) // Set {'a','b','c'} */ export function union<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a, ...b]); } /** * interseccion - Calcula la interseccion matematica A ∩ B. * * @definition A ∩ B = { x | x in A AND x in B } * @param a Primer conjunto. * @param b Segundo conjunto.

»"* @returns Un nuevo Set con los elementos comunes a A y B. * @complexity O(|A|) tiempo, O(min(|A|,|B|)) espacio. * * @example * interseccion(new Set(['a','b','c']), new Set(['b','c'])) // Set {'b','c'} */ export function interseccion<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a].filter(x =>b.has(x))); } /** * diferencia - Calcula la diferencia A \{} B. * * @definition A \{} B = { x | x in A AND x not-in B } * @param a Conjunto del que se restan elementos. * @param b Conjunto cuyos elementos se excluyen. * @returns Un nuevo Set con los elementos de A que no estan en B. * @complexity O(|A|) tiempo, O(|A|) espacio. * * @example * diferencia(new Set(['a','b','c']), new Set(['b'])) // Set {'a','c'} */ export function diferencia<T>(a: Set<T>, b: Set<T>): Set<T>{ return new Set([...a].filter(x =>!b.has(x))); } /** * complemento - Calcula el

complemento de A respecto al universo U. * * @definition A' = U \{} A = { x | x in U AND x not-in A } * @param universo El conjunto universal del problema. * @param a El conjunto cuyo complemento se calcula. * @returns Un nuevo Set con los elementos de U que no pertenecen a A. * @complexity O(|U|) tiempo, O(|U|) espacio. * * @example * complemento(new Set(['a','b','c']), new Set(['a'])) // Set {'b','c'} */ export function complemento<T>(universo: Set<T>, a: Set<T>): Set<T>{ return diferencia(universo, a); // Reutiliza diferencia() - DRY principle } /** * potencia - Calcula el conjunto potencia P(A). *

»"* @definition P(A) = { S | S C= A } * |P(A)| = 2^|A| * * ALGORITMO: mascara de bits (bit-mask enumeration). * Para cada entero i de 0 a 2^n-1, el valor binario de i * determina que elementos de A incluir en el subconjunto i-esimo. * Si el bit j de i es 1, el elemento j-esimo de A se incluye. * * @param a El conjunto del que se calcula la potencia. * @returns Un Set<Set<T" con todos los subconjuntos de A. * @complexity O(2^n * n) tiempo y espacio, donde n = |A|. * @warning EXPONENCIAL. Usar solo para |A| <= 20. * * @example * potencia(new Set(['a','b'])) // Set { {}, {'a'}, {'b'}, {'a','b'} } */ export function potencia<T>(a: Set<T>): Set<Set<T» { const elementos = Array.from(a); // Necesario para acceso por indice const subconjuntos = new Set<Set<T(); const numSubconjuntos = Math.pow(2, elementos.length); // 2^n for (let i = 0; i <numSubconjuntos; i++) { const subconjunto = new Set<T>(); for (let j = 0; j <elementos.length; j++) { // Comprueba si el bit j del entero i esta activado // (1 « j) crea una mascara con solo el bit j en 1 // (i & mascara) es distinto de 0 si el bit j de i es 1 if ((i & (1 " j)) !== 0) { subconjunto.add(elementos[j]); } } subconjuntos.add(subconjunto); } return subconjuntos; }

»"APENDICE B - CODIGO DE PRUEBAS UNITARIAS A continuacion se presenta el codigo completo del archivo de pruebas unitarias src/lib/sets/operations.test.ts. src/lib/sets/operations.test.ts - pruebas completas /** * @file operations.test.ts * @description Pruebas unitarias del motor logico de conjuntos. * @framework Vitest (compatible con Vite) * @coverage 98 % de lineas, 95 % de ramas */ import { describe, it, expect } from 'vitest'; import { union, interseccion, diferencia, complemento, potencia } from './operations'; // ?? SUITE: union describe('union()', () =>{ it('une conjuntos sin elementos comunes', () =>{ expect(union(new Set(['a','b']), new Set(['c','d']))) .toEqual(new Set(['a','b','c','d'])); }); it('une conjuntos con elementos comunes', () =>{ expect(union(new Set(['a','b']), new Set(['b','c']))) .toEqual(new Set(['a','b','c'])); }); it('union con A vacio', () =>{ expect(union(new Set(), new Set(['a','b']))) .toEqual(new Set(['a','b'])); }); it('union con B vacio', () =>{ expect(union(new Set(['a','b']), new Set())) .toEqual(new Set(['a','b'])); }); it('union de dos vacios', () =>{ expect(union(new Set(), new Set())).toEqual(new Set()); }); it('union idempotente (A con A)', () =>{ expect(union(new Set(['a','b']), new Set(['a','b']))) .toEqual(new Set(['a','b'])); }); });

»"// ?? SUITE: interseccion describe('interseccion()', () =>{ it('interseccion parcial', () =>{ expect(interseccion(new Set(['a','b','c']), new Set(['b','c','d']))) .toEqual(new Set(['b','c'])); }); it('conjuntos disjuntos ->vacio', () =>{ expect(interseccion(new Set(['a','b']), new Set(['c','d']))) .toEqual(new Set()); }); it('A vacio ->resultado vacio', () =>{ expect(interseccion(new Set(), new Set(['a','b']))) .toEqual(new Set()); }); it('conjuntos identicos ->mismo conjunto', () =>{ expect(interseccion(new Set(['a','b']), new Set(['a','b']))) .toEqual(new Set(['a','b'])); }); it('interseccion unitaria', () =>{ expect(interseccion(new Set(['a','b','c']), new Set(['a']))) .toEqual(new Set(['a'])); }); }); // ?? SUITE: diferencia describe('diferencia()', () =>{ it('diferencia parcial A-B', () =>{ expect(diferencia(new Set(['a','b','c']), new Set(['b','c']))) .toEqual(new Set(['a'])); }); it('B no tiene comunes con A ->retorna A', () =>{ expect(diferencia(new Set(['a','b']), new Set(['c','d']))) .toEqual(new Set(['a','b'])); }); it('A subconjunto de B ->vacio', () =>{ expect(diferencia(new Set(['a']), new Set(['a','b','c']))) .toEqual(new Set()); }); it('A vacio ->vacio', () =>{ expect(diferencia(new Set(), new Set(['a','b']))) .toEqual(new Set()); }); it('B vacio ->retorna A', () =>{ expect(diferencia(new Set(['a','b']), new Set()))

».toEqual(new Set(['a','b'])); }); it('A igual B ->vacio', () =>{ expect(diferencia(new Set(['a','b']), new Set(['a','b']))) .toEqual(new Set()); }); }); // ?? SUITE: complemento describe('complemento()', () =>{ it('complemento parcial', () =>{ const U = new Set(['a','b','c','d']); expect(complemento(U, new Set(['a','b']))) .toEqual(new Set(['c','d'])); }); it('complemento del vacio = U', () =>{ const U = new Set(['a','b']); expect(complemento(U, new Set())).toEqual(U); }); it('complemento de U = vacio', () =>{ const U = new Set(['a','b']); expect(complemento(U, U)).toEqual(new Set()); });

}); // ?? SUITE: potencia describe('potencia()', () =>{ it('potencia del vacio: 1 subconjunto (el vacio)', () =>{ expect(potencia(new Set()).size).toBe(1); }); it('potencia({a}): 2 subconjuntos', () =>{ expect(potencia(new Set(['a'])).size).toBe(2); }); it('potencia({a,b}): 4 subconjuntos', () =>{ expect(potencia(new Set(['a','b'])).size).toBe(4); }); it('potencia({a,b,c}): 8 subconjuntos', () =>{ expect(potencia(new Set(['a','b','c'])).size).toBe(8); }); it('potencia({a,b,c,d}): 16 subconjuntos', () =>{ expect(potencia(new Set(['a','b','c','d'])).size).toBe(16); }); it('potencia verifica formula 2^n', () =>{ for (let n = 0; n <= 10; n++) { const elementos = Array.from({length: n}, (_,i) =>'e{i}'); »"expect(potencia(new Set(elementos)).size).toBe(Math.pow(2, n)); } }); });

»"APENDICE C - CONFIGURACION DEL PROYECTO C.1 package.json package.json { "name": "logica-simbolica-global", "version": "1.0.0", "private": true, "scripts": { "dev": "vite", "build": "vue-tsc && vite build", "preview": "vite preview", "test": "vitest run", "test:ui": "vitest –ui", "test:cov":"vitest run –coverage"}, "dependencies": { "vue": "^3.4.0"}, "devDependencies": { "@vitejs/plugin-vue": "^5.0.0", "@vue/test-utils": "^2.4.0", "jsdom": "^24.0.0", "typescript": "^5.3.0", "vite": "^5.0.0", "vitest": "^1.2.0", "vue-tsc": "^2.0.0"} } C.2 tsconfig.json tsconfig.json { çompilerOptions": { "target": .ES2022", üseDefineForClassFields": true, "module": .ESNext", "moduleResolution": "bundler", "strict": true, "jsx": "preserve",

»"resolveJsonModule":true, ïsolatedModules": true, "noEmit": true, "lib": [.ES2022", "DOM", "DOM.Iterable"], "baseUrl": ".", "paths": { "@/*": ["src/*"] } }, ïnclude": ["src/**/*.ts", "src/**/*.vue"] } C.3 vite.config.ts vite.config.ts import { defineConfig } from 'vite' import vue from '@vitejs/plugin-vue' import path from 'path' export default defineConfig({ plugins: [vue()], resolve: { alias: { '@': path.resolve(__dirname, './src'), }, }, test: { environment: 'jsdom', globals: true, include: ['src/**/*.test.ts'], coverage: { provider: 'v8', include: ['src/lib/**/*.ts'], }, }, })

»"APENDICE D - GLOSARIO DE TERMINOS API (Application Programming Interface) Conjunto de definiciones, protocolos y herramientas que permiten la comunicacion entre diferentes componentes de software. En este proyecto, .APIrefiere tanto a la API Set de ES6 como a la API publica del modulo operations.ts. Bundle El archivo o conjunto de archivos JavaScript/CSS resultante del proceso de compilacion de Vite. Contiene todo el codigo necesario para ejecutar la aplicacion en el navegador, optimizado para minimizar el tamano y maximizar el rendimiento. CI/CD (Continuous Integration / Continuous Delivery) Practica de ingenieria de software que consiste en integrar el codigo frecuentemente (CI) y desplegar automaticamente cada version aprobada (CD). En este proyecto, GitHub Actions ejecuta las pruebas y despliega automaticamente en cada push a main. Composition API API de Vue 3 que permite organizar la logica del componente por funcionalidad (usando ref, computed, watch, etc.) en lugar de por tipo de opcion (como en Options API). Facilita la reutilizacion de logica entre componentes mediante composables. Computed (propiedad computada) En Vue 3, una propiedad computada es una funcion que se recalcula automaticamente cuando cambian sus dependencias reactivas. Equivalente a una "formula.en hojas de calculo que se actualiza sola al cambiar los datos de entrada. DRY (Dont Repeat Yourself) Principio de desarrollo de software que establece que çada pieza de conocimiento debe tener una representacion unica, inequivoca y autorizada dentro del sistema". En operations.ts, complemento() reutiliza diferencia() en lugar de duplicar codigo.

»"ES6 (ECMAScript 2015) Version 6 del estandar ECMAScript (JavaScript). Introdujo caracteristicas clave como: let/const, arrow functions, clases, modulos (import/export), template literals, destructuring y la API Set/Map. Generic / Generica (funcion generica) En TypeScript, una funcion generica es aquella que puede operar sobre diferentes tipos de datos sin perder la seguridad de tipos. Se declaran con <T>y permiten que el mismo codigo funcione con Set<string>, Set<number>, etc. HMR (Hot Module Replacement) Caracteristica de Vite que reemplaza en el navegador solo el modulo modificado al guardar un archivo, sin recargar la pagina completa. Permite ver los cambios instantaneamente (<50ms) durante el desarrollo. SFC (Single File Component) Formato de archivo .vue de Vue que encapsula en un unico archivo la plantilla HTML (<template>), la logica JavaScript/TypeScript (<script>) y los estilos CSS (<style>). Facilita la organizacion del codigo por componente. Set (tipo de dato) Estructura de datos de JavaScript ES6 que representa una coleccion de valores unicos (sin duplicados). Garantiza la unicidad automaticamente y ofrece comprobacion de pertenencia O(1) con el metodo .has(). Spread operator (...) Operador de JavaScript ES6 que

.expandeün iterable (como un Set o Array) en sus elementos individuales. En [...a, ...b], ambos Sets se convierten en sus elementos, que luego se concatenan en un nuevo array. SRP (Single Responsibility Principle) Principio de diseno de software que establece que una clase o funcion debe tener una

»"sola razon para cambiar. En operations.ts, cada funcion tiene una unica responsabilidad: calcular una operacion especifica de conjuntos. SVG (Scalable Vector Graphics) Formato de imagen vectorial basado en XML, nativo del navegador web. Las imagenes SVG escalan sin perdida de calidad a cualquier resolucion. En este proyecto, el diagrama de Venn se implementa como SVG inline en Vue. Tree-shaking Optimizacion del bundler (Vite) que elimina el codigo no utilizado del bundle final. Las exportaciones nombradas de operations.ts permiten el tree-shaking: si la interfaz no usa "potencia", esa funcion se excluye del bundle. TypeScript Superconjunto tipado de JavaScript desarrollado por Microsoft. Agrega anotaciones de tipo opcionales que son verificadas en tiempo de compilacion. El codigo TypeScript se transpila a JavaScript estandar para ejecutarse en el navegador. Vitest Framework de pruebas unitarias moderno, compatible con Vite. Reutiliza la configuracion de Vite (alias, plugins) sin necesidad de configuracion adicional. Es significativamente mas rapido que Jest para proyectos con Vite.

»"7.1. Anexo Linus 7p — propuesta inicial

»"8. Capítulo Técnico — Módulo Inferencias (Hijos de Linus) — Transcripción 14p + tex 27K

»"FACULTAD DE INGENIERIA CIVIL, SISTEMAS Y ARQUITECTURA INGENIERIA DE SISTEMAS UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO INFORME ACADÉMICO Pagina Web interactiva de inferencias lógicas y trazabilidad formal Curso: Lógica Simbólica Ciclo II Equipo - Los Hijos de Linus Altamirano Astupiñan Renato André Centurión Llatas Alex Daniel Espinoza Orrego Piero Arom Mio Caballero Arnold André Morocho Lumbre Juan David Docente: Dr. Mardo Victor Gonzales Herrera 30 de agosto de 2026

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto Tabla de Contenidos 1. Introducción y Propuesta del Proyecto (Sprint 1) 2. Arquitectura del Software y Flujo de Trabajo (Git Flow) 3. Capa Lógica: El Motor de Inferencia y Demostración (Sprint 2) 4. Capa Visual: Interfaz Reactiva y Didáctica (Sprint 3 y 3.5) 5. Pruebas de Calidad del Código (Sprint 4) 6. Participación y Aportes del Equipo 7. Trabajo en Equipo 8. Conclusiones Resumen Ejecutivo El presente informe documenta la investigación, diseño arquitectónico, implementación algorítmica y aseguramiento de calidad del módulo de Inferencias Lógicas, Trazabilidad Pedagógica y Diagnóstico con Contraejemplos desarrollado por el equipo Los Hijos de Linus (Grupo 2) en el marco de la asignatura de Lógica Simbólica de la Escuela Profesional de Ingeniería de Sistemas (UNPRG). El software combina un motor de deducción formal automatizado (Forward Chaining) con un evaluador semántico exhaustivo de 2N combinaciones de verdad, una interfaz moderna y accesible construida en Vue 3 (Composition API) con TypeScript en modo estricto, soporte para 11 reglas de deducción natural, teclado simbólico interactivo, exportación en un clic a LATEX / Markdown, y un sistema de traducción a lenguaje natural con persistencia local. 1

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto 1. Introducción y Propuesta del Proyecto (Sprint 1) El razonamiento deductivo y la deducción formal en lógica proposicional constituyen el núcleo formativo del pensamiento formal en ciencias de la computación. No obstante, el aprendizaje tradicional de las inferencias lógicas presenta barreras de aprendizaje críticas: a) Dificultad en la estructuración deductiva: A los estudiantes universitarios les resulta complejo planificar y encadenar reglas de inferencia para transitar desde un conjunto de premisas iniciales hasta la conclusión deseada de forma sistemática. b) Ausencia de retroalimentación inmediata y explicativa: Cuando un argumento es inválido o incurre en una falacia formal (como la afirmación del consecuente o negación del antecedente), las herramientas convencionales se limitan a indicar un fallo genérico, sin explicar qué regla se vulneró ni proveer la asignación de verdad que refuta el razonamiento. c) Desconexión entre notación matemática y razonamiento humano: Existe una mar-

cada brecha cognitiva al traducir argumentos cotidianos a fórmulas proposicionales (P →Q, P Q) y viceversa. Para resolver estas problemáticas de forma integral, el Equipo Los Hijos de Linus diseñó una herramienta web asistida y estructurada en dos vertientes interactivas: Módulo de Resolución Automática y Demostración Formal: Permite al usuario ingresar premisas libres y una conclusión objetivo para obtener una derivación formal paso a paso o la refutación exacta con su respectivo contraejemplo matemático. Módulo de Trazabilidad y Traducción Pedagógica: Desglosa cada paso demos- trativo con justificaciones semánticas en lenguaje natural, rol deductivo de las líneas antecedentes y mapeo dinámico a enunciados contextuales cotidianos. Figura 1: Boceto preliminar (Arnold Mio) del Módulo 1: Resolución y Demostración Automática. 2

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto Figura 2: Boceto preliminar (David Morocho) del Módulo 2: Entorno de Práctica Interactiva y Validación. 2. Arquitectura del Software y Flujo de Trabajo El proyecto fue desarrollado bajo una arquitectura modular y desacoplada, asegurando que la lógica computacional del motor de inferencias no posea dependencia alguna respecto al framework de presentación visual. 2.1. Estructura de Directorios del Módulo La organización física del código fuente se divide claramente entre la capa matemática pura y la capa de componentes reactivos: Capa Lógica y Motor Matemático (src/lib/): • src/lib/solver/types.ts: Definición formal de los tipos del Árbol de Sintaxis Abstracta (AST), operadores lógicos, reglas y modelos de contraejemplo. • src/lib/solver/parser.ts: Tokenizador léxico y analizador sintáctico por descenso recursivo con precedencia de operadores. 3

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto Figura 3: (Alex Centurion) Diseño y maquetación de la propuesta visual integrada para el sistema de inferencias. • src/lib/solver/solver.ts: Algoritmo de Forward Chaining, evaluador semántico de verdad (2N), detector de falacias y aplicación de 11 reglas de deducción natural. • src/lib/validator/validator.ts: Sanitizador de cadenas, normalización de no- taciones heterogéneas y validador reactivo sin excepciones. • src/lib/trazabilidad/historial.ts: Gestor de historial de deducción, numera- ción de pasos y resolución de referencias cruzadas. • src/lib/transcription/descriptionGenerator.ts y naturalTranslator.ts: Generación de explicaciones semánticas en español y traducción de variables a argu- mentos continuos. Capa de Interfaz de Usuario (src/pages/inferencias/ y src/components/inferencias/): • index.vue: Orquestador reactivo principal con teclado virtual y pestañas de modo formal vs. lenguaje natural. • FormularioInferencia.vue: Captura dinámica de premisas, teclado asistido y validación visual. • IndicadorResultado.vue: Semáforo visual de 3 estados con diagnóstico explícito de falacias. • PanelTrazabilidad.vue: Tabla formal numerada (1), (2), ů ů ů ∴(N), acordeón ex- plicativo y exportadores a LATEX y Markdown. • TraductorLenguajeNatural.vue: Asignación contextual de proposiciones cotidia- nas. Documentación y Tareas de Sprint (TAREAS/): • TAREAS/01_Propuesta/hijos-de-linus/propuesta.md: Documento de propuesta inicial y bocetos. • TAREAS/02_Motor/hijos-de-linus/avance.md y casos_de_prueba.md: Avances del motor y matriz de 25 casos límite. 4

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto • TAREAS/03_UI/hijos-de-linus/avance.md y glosario-diseno.md: Avance de com- ponentes y guía de diseño. • TAREAS/04_Deploy/hijos-de-linus/uso.md y errores_y_soluciones.md: Manual de uso y reporte de control de calidad. 2.2. Flujo de Git por Ramas Para garantizar la estabilidad del repositorio compartido con los demás equipos, se utilizó un flujo estricto de integración continua basado en ramas por sprint y Pull Requests atómicos: sprint-1-propuesta: Maquetación, definición de requerimientos y casos de uso iniciales. sprint-2-hijos-de-linus: Implementación del AST, Parser, Solver, Validador, Traza- bilidad y pruebas unitarias de motor. sprint-3-hijos-de-linus: Construcción de los 4 componentes Vue 3, teclado simbólico e integración de la vista. sprint-3.5-fase-inicial: Incorporación de diagnósticos con contraejemplos, acordeón didáctico y exportación LATEX. deploy / main: Fusión final validada con 113 pruebas automáticas y 0 errores de tipado en TypeScript. 5

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto 3. Capa Lógica: El Motor de Inferencia y Demostración (Sprint 2) El motor computacional de inferencias lógicas opera mediante un pipeline determinista de cuatro fases: Fase 1: Sanitización y Normalización Léxica: Unifica las múltiples notaciones matemáticas y lógicas ingresadas por el estudiante (->, , , ¬, , &, |, , ,) convirtiéndolas en identificadores estándar (ENTONCES, Y, O, NO, O_EXCLUSIVA, SI_Y_SOLO_SI), eliminando caracteres inválidos y previniendo inyecciones de código. Fase 2: Analizador Sintáctico

(Parser AST por Descenso Recursivo): Convierte la secuencia lineal de tokens en un Árbol de Sintaxis Abstracta estructurado, respetando la jerarquía y precedencia matemática rigurosa: Negación (¬) >Conjunción (∧) >Disyunciones (∨,) >Condicional (→) >Bicondicional (↔) Fase 3: Motor Deductivo (Forward Chaining y Pattern Matching): Aplica de manera iterativa (hasta 12 niveles de derivación) las 11 reglas clásicas de deducción natural: Modus Ponendo Ponens (MPP): A →B, A B Modus Tollendo Tollens (MTT): A →B, ¬B ¬A Silogismo Disyuntivo (SD): A ∨B, ¬A B / A ∨B, ¬B A Silogismo Disyuntivo Exclusivo (SDE): AB, A ¬B / AB, B ¬A Silogismo Hipotético (SH): A →B, B →C A →C Simplificación y Conjunción: A ∧B A, B / A, B A ∧B Dilema Constructivo (DC): (A →B) ∧(C →D), A ∨C B ∨D Bicondicionales: Modus Ponens Bicondicional y Eliminación Bicondicional (A ↔ B A →B, B →A). Fase 4: Evaluador Semántico SAT (2N combinaciones de verdad): Garantiza la clasificación estricta del razonamiento en 3 estados de validez: Inferencia Válida (Demostrada): Se completó la deducción paso a paso mediante reglas directas. Inferencia Válida (Método Indirecto Requerido): El argumento es lógicamente válido (0 contraejemplos semánticos), pero requiere Reducción al Absurdo o Prueba Condicional. Inferencia Inválida (Refutada por Contraejemplo): Detecta la asignación con- creta donde todas las premisas son verdaderas y la conclusión es falsa, diagnosticando falacias como el Dilema Inverso o Afirmación del Consecuente. 3.1. Código Fuente del Motor Lógico A continuación, se detalla el evaluador semántico y la detección de modelos implementados en src/lib/solver/solver.ts: 1 // src/lib/solver/solver.ts 2 export function evaluarNodo(3 nodo: NodoExpresion , 4 asignacion: Record <string , boolean >6

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto 5): boolean { 6 if (nodo.tipo === 'variable ') { 7 return Boolean(asignacion[nodo.nombre.toUpperCase ()]); 8 } 9 if (nodo.tipo === 'operacion ') { 10 if (nodo.operador === 'NO' && nodo.derecho) { 11 return !evaluarNodo(nodo.derecho , asignacion); 12 } 13 if (nodo.izquierdo && nodo.derecho) { 14 const izq = evaluarNodo(nodo.izquierdo , asignacion); 15 const der = evaluarNodo(nodo.derecho , asignacion); 16 switch (nodo.operador) { 17 case 'Y': return izq && der; 18 case 'O': return izq || der; 19 case 'O_EXCLUSIVA ': return izq !== der; // Disyuncion fuerte 20 case 'ENTONCES ': return !izq || der; 21 case 'SI_Y_SOLO_SI ': return izq === der; 22 case 'NI': return !(izq || der); 23 case 'INCOMPATIBLE ': return !(izq && der); 24 default: return false; 25 } 26 } 27 } 28 return false; 29 } Listing 1: Evaluador semántico AST y detección de contraejemplos (solver.ts). 7

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto 4. Capa Visual: Interfaz Reactiva y Didáctica (Sprint 3 y 3.5) La interfaz gráfica (src/pages/inferencias/index.vue) fue construida bajo la guía de diseño del proyecto, priorizando claridad visual, respuesta reactiva instantánea y accesibilidad universal. 4.1. Reactividad en Vue 3 y Composition API Flujo Reactivo Unidireccional: Empleo estricto de ref, computed y tipado mediante interfaces compartidas (src/types/inferencias.ts), evitando estados globales super- fluos. Teclado Simbólico en Pantalla: Permite insertar conectivos lógicos (¬, ∧, ∨, , →, ↔, (,)) con un solo clic o toque en dispositivos móviles, manteniendo la posición del cursor en el input activo. Pestañas de Visualización: Permite conmutar instantáneamente entre el formato de Simbología Formal y el de Lenguaje Natural, sincronizando en tiempo real las variables proposicionales. 4.2. Trazabilidad Pedagógica y Herramientas Académicas Desglose Particionado con Acordeón Colapsable: Cada paso de la demostración formal incluye un botón interactivo (£Por qué esta regla?) que despliega: • Las líneas de premisa base utilizadas y su rol deductivo (ej. Condicional base, Antecedente afirmado). • La regla formal aplicada y su justificación semántica en español. • La conclusión parcial derivada. Exportación en 1 Clic a LATEX y Markdown: Formatea automáticamente la demos- tración completa en código compatible con artículos científicos: (1) P →Q (Premisa) (2) P (Premisa) ∴(3) Q (Modus Ponendo Ponens de 1, 2) Historial Persistente en Memoria Local (localStorage): Almacena las últimas demostraciones realizadas para permitir su restauración inmediata. 4.3. Visualización Interactiva del Árbol de Sintaxis Abstracta (AST) Como complemento pedagógico al panel de trazabilidad formal, el sistema incorpora un módulo de visualización gráfica que representa la demostración completa como un árbol jerárquico, permitiendo al estudiante observar la descomposición estructural de cada premisa y de la conclusión en sus respectivos conectivos y variables proposicionales. Sobre un ejercicio de inferencia ya resuelto y clasificado como válido mediante deducción directa, el usuario puede acceder a esta representación a través de una pestaña dedicada dentro de la misma interfaz de resolución,

sin necesidad de abandonar el contexto del problema planteado. Esta pestaña, denominada Árbol de Nodos, se ubica junto a las vistas de Simbología Formal y Lenguaje Natural, evidenciando la integración de las tres modalidades de representación dentro de un mismo flujo de trabajo. 8

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto Figura 4: Panel de resolución de inferencias con las premisas y la conclusión demostrada, mostrando la pestaña Árbol de Nodos disponible como modalidad adicional de visualización. Al ingresar a dicha pestaña, la interfaz presenta una introducción explicativa orientada a familiarizar al usuario con la lectura del diagrama antes de exponer la estructura completa del árbol. Esta sección preliminar define la simbología empleada para cada tipo de conectivo lógico (conjunción, disyunción, disyunción exclusiva, negación, implicación, bicondicional, entre otros) y describe las convenciones de lectura del diagrama, indicando que el nodo raíz representa la demostración completa, que cada rama corresponde a una premisa o a la conclusión, y que la jerarquía se interpreta de manera descendente desde lo general hacia el detalle. Asimismo, se despliegan métricas cuantitativas del ejercicio en curso, tales como el número de proposiciones, la cantidad total de nodos, la profundidad máxima del árbol y el número de conectivos empleados. Figura 5: Introducción explicativa del módulo Árbol de Nodos, con la simbología de conectivos, las convenciones de lectura del diagrama y las métricas estructurales del ejercicio. Una vez comprendida la simbología, el estudiante puede examinar el diagrama jerárquico propiamente dicho, en el cual cada premisa se descompone visualmente en sus componentes lógicos hasta llegar a las variables proposicionales atómicas. Dado que la profundidad y el ancho del árbol varían según la complejidad del ejercicio, el componente fue diseñado con soporte de 9

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto desplazamiento horizontal y vertical (scroll), lo que permite recorrer la totalidad de la estructura sin perder legibilidad, incluso en ejercicios con múltiples niveles de anidación de conectivos. Figura 6: Fragmento del diagrama jerárquico del AST con desplazamiento horizontal y vertical, mostrando la descomposición de una premisa en sus conectivos de implicación, disyunción y negación. Finalmente, la interfaz ofrece una modalidad de visualización ampliada del diagrama en pantalla completa, orientada a facilitar el análisis de ejercicios con un mayor número de premisas. En esta vista expandida se aprecia de forma simultánea la descomposición de las cuatro premisas del ejercicio, evidenciando de manera clara la jerarquía completa de la demostración desde el nodo raíz hasta las variables proposicionales hoja. Figura 7: Vista ampliada en pantalla completa del diagrama de árbol, mostrando de manera simultánea la descomposición de las cuatro premisas del ejercicio en sus respectivos conectivos y variables. 5. Pruebas de Calidad del Código (Sprint 4) Para certificar la confiabilidad matemática del software, se implementó una suite de pruebas automatizadas con Vitest: Cobertura Exhaustiva: 13 archivos de prueba y 113 tests unitarios e integrados ejecutados con 100 % de éxito. Chequeo Estricto de Tipos: Verificación continua con vue-tsc –noEmit (0 errores de TypeScript). Pruebas de Estrés y Seguridad (Fuzzer): Evaluación de expresiones con hasta 10 niveles de anidación, caracteres corruptos, emojis y mitigación de inyecciones de código malicioso. Validación de Casos Límite: Pruebas de derivación encadenada de hasta 6 pasos, disyunción exclusiva fuerte y contraejemplos matemáticos precisos. 10

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto Resultados de la Suite de Pruebas (Vitest) src/lib/solver/solver.test.ts (15 tests) src/lib/solver/parser.test.ts (12 tests) src/lib/validator/validator.test.ts (27 tests) src/lib/validator/validator_stress.test.ts (15 tests) src/lib/trazabilida (9 tests) src/lib/integracion_motor.test.ts (8 tests) src/components/inferencias/___tests___/* (27 tests) Total: 13 archivos pasados (13) | 113 tests pasados (113) — 100 % Éxito 11

»"Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto 6. Participación y Aportes del Equipo La asignación de responsabilidades entre los integrantes del Equipo Los Hijos de Linus se resume en la Tabla 1: Integrante Rol Principal Contribución Clave Altamirano Astupiñan Renatto André Ingeniero de Pruebas y Calidad (QA) Implementación del validador y sanitizador (validator.ts), diseño de matriz de 25 casos límite, tests de estrés (fuzzer). Centurión Llatas Alex Da- niel Desarrollador Frontend y UX Implementación de IndicadorResultado.vue, semáforo de 3 estados, diseño del acordeón didáctico de trazabilidad, feedback visual de falacias y configuración de vitest. Espinoza Orrego Piero Arom Arquitecto de Software y Motor Lógico Modelado del AST, analiza-

dor léxico y sintáctico por descenso recursivo (parser.ts), motor de Forward Chaining (solver.ts) e integración global de la vista. Mio Caballero Arnold An- dré Ingeniero de Transcripción y Lenguaje Natural Desarrollo del generador de explicaciones didácticas (translations.ts, descriptionGenerator.ts), renderizador de AST y componente TraductorLenguajeNatural.vue. Morocho Lumbre Juan David Analista de Lógica y Trazabilidad Construcción del contenedor de historial y trazabilidad (historial.ts), algoritmo de diagnóstico de errores lógicos (pattern matching) y desarrollo de FormularioInferencia.vue. Tabla 1: Distribución de roles y contribuciones técnicas del equipo. 7. Trabajo en Equipo El desarrollo del proyecto se articuló mediante sesiones de trabajo colaborativo en la plataforma Google Meet, las cuales sirvieron como espacio centralizado para la coordinación técnica, revisión de código en vivo y planificación iterativa de cada sprint. La metodología colaborativa contempló: Pair Programming y Code Review: Validación de componentes y funciones críticas antes de fusionar Pull Requests. Sincronización Continua con Git Flow: Manejo disciplinado de ramas por sprint, asegurando commits atómicos y evitando conflictos con los otros equipos del repositorio. Cultura de Calidad: Comprobación sistemática de tipado (npm run type-check) y ejecución de pruebas unitarias antes de cada entrega. 8. Conclusiones 1. Innovación Pedagógica en la Enseñanza de la Lógica: Se construyó una plataforma web capaz de convertir la demostración formal de inferencias en un proceso didáctico 12

»”Equipo Los Hijos de Linus — Lógica Simbólica Informe de Proyecto transparente, combinando derivaciones formales con justificaciones en lenguaje natural y contraejemplos matemáticos claros. 2. Arquitectura Limpia, Modular y Escalable: La separación estricta entre la biblio- teca matemática en TypeScript puro (src/lib/) y la capa de presentación en Vue 3 (src/components/) garantizó alta cohesión, bajo acoplamiento y facilitó un flujo de trabajo paralelo sin interferencias. 3. Rigor Matemático y Verificación Exhaustiva: La combinación del motor deductivo de Forward Chaining con el evaluador semántico SAT de 2N combinaciones permitió resolver con precisión absoluta los 3 estados de validez del argumento, alcanzando 100 % de aprobación en la suite de 113 pruebas automatizadas. 13

»”8.1. Informe hijos-de-linus.tex — contenido íntegro (450 líneas, estilo header-blue)

»”\{}documentclass[11pt,a4paper]{article}

»” % — PAQUETES DE IDIOMA Y CODIFICACIÓN — \{}usepackage[utf8]{inputenc} \{}usepackage[T1]{fontenc} \{}usepackage[spanish,es-tabla]{babel}

»” % — TIPOGRAFÍA Y GEOMETRÍA — \{}usepackage{lmodern} \{}usepackage[top=2.5cm, bottom=2.5cm, left=2.5cm, right=2.5cm]{geometry} \{}usepackage{microtype}

»” % — COLORES Y ESTILO — \{}usepackage[table,xcdraw]{xcolor} \{}definecolor{headerblue}{RGB}{24 43, 73} \{}definecolor{accentblue}{RGB}{30, 80, 150} \{}definecolor{lightbg}{RGB}{248, 249, 251} \{}definecolor{bordergray}{RGB}{220, 224, 230} \{}definecolor{darktext}{RGB}{33, 37, 41} \{}definecolor{codebg}{RGB}{245, 247, 250} \{}definecolor{commentgreen}{RGB}{40, 140, 70} \{}definecolor{keywordpurple}{RGB}{120, 40, 140} \{}definecolor{stringblue}{RGB}{20, 90, 160}

»” % — MATEMÁTICAS Y SÍMBOLOS — \{}usepackage{amsmath} \{}usepackage{amssymb} \{}usepackage{amsfonts}

»” % — IMÁGENES Y RUTAS — \{}usepackage{graphicx} \{}graphicspath{{assets/}{01_Propuesta/hijos-de-linus/assets/}{../../01_Propuesta/hijos-de-linus/assets/}{TAREAS/assets/}{TAREAS/01_Propuesta/hijos-de-linus/assets/}}

»” % — CÓDIGO FUENTE (LISTINGS) — \{}usepackage{listings} \{}lstdefinelanguage{TypeScript}{ keywords={abstract, any, as, boolean, break, case, catch, class, console, const, continue, debugger, declare, default, delete, do, else, enum, export, extends, false, finally, for, from, function, get, if, implements, import, in, instanceof, interface, let, module, new, null, number, of, package, private, protected, public, readonly, return, set, static, string, super, switch, symbol, this, throw, true, try, type, typeof, var, void, while, with, yield}, keywordstyle=\{}color{keywordpurple}\{}bfseries, ndkeywords={class, export, boolean, throw, implements, import, this}, ndkeywordstyle=\{}color{accentblue}\{}bfserie identifierstyle=\{}color{darktext}, sensitive=true, comment=[l]{//}, morecomment=[s]{/*}{*/},

commentstyle=\{}color{commentgreen}\{}itshape, stringstyle=\{}color{stringblue}, morestring=[b]', morestring=[b]"}

»"\{}lstset{ language=TypeScript, backgroundcolor=\{}color{codebg}, basicstyle=\{}ttfamily\{}footnotesi breaklines=true, captionpos=b, frame=single, rulecolor=\{}color{bordergray}, numbers=left, numberstyle=\{}tiny\{}color{gray}, numbersep=8pt, showstringspaces=false, tabsize=2, extendedchars=true, literate={á}{{\{}'a}}1 {é}{{\{}'e}}1 {í}{{\{}'i}}1 {ó}{{\{}'o}}1 {ú}{{\{}'u}}1 {ñ}{{\{}~n}}1 {Á}{{\{}'A}}1 {É}{{\{}'E}}1 {Í}{{\{}'I}}1 {Ó}{{\{}'O}}1 {Ú}{{\{}'U}}1 {Ñ}{{\{}~N}}1 }

»" % — TABLAS Y ELEMENTOS GRÁFICOS — \{}usepackage{tabularx} \{}usepackage{booktabs} \{}usepackage{array} \{}usepackage{enumitem} \{}usepackage{tcolorbox} \{}usepackage{caption}

»" % — ENCABEZADOS Y PIES DE PÁGINA — \{}usepackage{fancyhdr} \{}pagestyle{fancy} \{}fancyhf{}

»"\{}renewcommand{\{}headrulewidth}{0pt} \{}renewcommand{\{}footrulewidth}{0.6pt}

»"\{}fancyhead[L]{\{}fontsize{9}{11}\{}selectfont\{}color{darktext}\{}textbf{Equipo Los Hijos de Linus} — Lógica Simbólica} \{}fancyhead[R]{\{}fontsize{9}{11}\{}selectfont\{}color{darktext}Informe de Proyecto} \{}fancyfoot[C]{\{}fontsize{10}{12}\{}selectfont\{}thepage}

»" % — HIPERVÍNCULOS — \{}usepackage{hyperref} \{}hypersetup{ colorlinks=true, linkcolor=accentblue, filecolor=accentblue, urlcolor=accentblue, pdftitle={Sistema Web Interactivo de Inferencias Lógicas - Informe Técnico}, pdfauthor={Equipo Los Hijos de Linus}, }

»"\{}begin{document}

»" % ========== PORTADA % ========== \{}begin{titlepage} \{}begin{tcolorbox}[colback=headerblue,arc=0mm,boxrule=0pt,height=1.2cm,valign=cente \{}centering \{}textcolor{white}{\{}small\{}textbf{Equipo Los Hijos de Linus — Lógica Simbólica \{}hfill Informe de Proyecto}} \{}end{tcolorbox}

»"\{}vspace{2.2cm} \{}begin{center} {\{}Large\{}textbf{Universidad Nacional Pedro Ruiz Gallo}}\{}\{}[0.3cm] {\{}large\{}textbf{Facultad de Ingeniería Civil, de Sistemas y de Arquitectura}}\{}\{}[0.2cm] {\{}normalsize\{}textbf{Escuela Profesional de Ingeniería de Sistemas}}\{}\{}[1.2cm]

»"\{}textcolor{headerblue}{\{}rule{0.85\{}textwidth}{1.5pt}}\{}\{}[0.8cm]

»"{\{}LARGE\{}textbf{\{}textcolor{headerblue}{Sistema Web Interactivo de Inferencias}}}\{}\{}[0.3cm] {\{}LARGE\{}textbf{\{}textcolor{headerblue}{Lógicas y Trazabilidad Formal}}}\{}\{}[0.5cm] {\{}large\{}textbf{Informe Técnico Final del Proyecto}}\{}\{}[0.8cm]

»"\{}textcolor{headerblue}{\{}rule{0.85\{}textwidth}{1.5pt}}\{}\{}[1.8cm]

»"\{}begin{minipage}{0.85\{}textwidth} \{}centering \{}textbf{\{}large Integrantes — Equipo Los Hijos de Linus (Grupo 2):}\{}\{}[0.4cm] {\{}normalsize Altamirano Astupiñan, Renatto André\{}\{} Centurión Gonzales, Alex\{}\{} Espinoza Orrego, Piero Arom\{}\{} Mio Caballero, Arnold André\{}\{} Morocho Lumbre, Juan David\{}\{}[1.2cm] } \{}textbf{\{}large Docente:}\{}\{} Dr. Mardo Victor Gonzales Herrera \{}end{minipage}

»"\{}vfill {\{}small Lambayeque, 2026} \{}end{center} \{}end{titlepage}

»" % ========== TABLA DE CONTENIDOS % ========== \{}newpage \{}section*{Tabla de Contenidos} \{}vspace{0.5cm} \{}begin{enumerate}[label=\{}textbf{\{}arab \{}setlength{\{}itemsep}{0.35cm} \{}item \{}textbf{Introducción y Propuesta del Proyecto (Sprint 1)} \{}item \{}textbf{Arquitectura del Software y Flujo de Trabajo (Git Flow)} \{}item \{}textbf{Capa Lógica: El Motor de Inferencia y Demostración (Sprint 2)} \{}item \{}textbf{Capa Visual: Interfaz Reactiva y Didáctica (Sprint 3 y 3.5)} \{}item \{}textbf{Pruebas de Calidad del Código (Sprint 4)} \{}item \{}textbf{Participación y Aportes del Equipo} \{}item \{}textbf{Trabajo en Equipo} \{}item \{}textbf{Conclusiones} \{}end{enumerate}

»"\{}vspace{1.5cm} \{}begin{tcolorbox}[colback=lightbg,colframe=bordergray,arc=2mm,title=\{}textbf{R Ejecutivo}] El presente informe documenta la investigación, diseño arquitectónico, implementación algorítmica y aseguramiento de calidad del módulo de \{}textbf{Inferencias Lógicas, Trazabilidad Pedagógica y Diagnóstico con Contraejemplos} desarrollado por el equipo \{}textbf{Los Hijos de Linus} (Grupo 2) en el marco de la asignatura de Lógica Simbólica de la Escuela Profesional de Ingeniería de Sistemas (UNPRG).

»"El software combina un motor de deducción formal automatizado (\{}textit{Forward Chaining}) con un evaluador semántico exhaustivo de 2^N combinaciones de verdad, una interfaz moderna y accesible construida en \{}textbf{Vue 3 (Composition API)} con \{}textbf{TypeScript en modo estricto}, soporte para 11 reglas de deducción natural, teclado simbólico interactivo, exportación en un clic a \{}textbf{\{}LaTeX} / \{}textbf{Markdown}, y un sistema de traducción a lenguaje natural con persistencia local. \{}end{tcolorbox}

»"% ==========
SECCIÓN 1: INTRODUCCIÓN Y PROPUESTA (SPRINT 1) % ==========
\{}newpage \{}section{Introducción y Propuesta del Proyecto (Sprint 1)}

»"El razonamiento deductivo y la deducción formal en lógica proposicional constituyen el núcleo formativo del pensamiento formal en ciencias de la computación. No obstante, el aprendizaje tradicional de las inferencias lógicas presenta barreras de aprendizaje críticas: \{}begin{enumerate}[label=\{}alph*)] \{}item \{}textbf{Dificultad en la estructuración deductiva:} A los estudiantes universitarios les resulta complejo planificar y encadenar reglas de inferencia para transitar desde un conjunto de premisas iniciales hasta la conclusión deseada de forma sistemática. \{}item \{}textbf{Ausencia de retroalimentación inmediata y explicativa:} Cuando un argumento es inválido o incurre en una falacia formal (como la afirmación del consecuente o negación del antecedente), las herramientas convencionales se limitan a indicar un fallo genérico, sin explicar qué regla se vulneró ni proveer la asignación de verdad que refuta el razonamiento. \{}item \{}textbf{Desconexión entre notación matemática y razonamiento humano:} Existe una marcada brecha cognitiva al traducir argumentos cotidianos a fórmulas proposicionales ($P \to Q, P \vdash Q$) y viceversa. \{}end{enumerate}

»"Para resolver estas problemáticas de forma integral, el \{}textbf{Equipo Los Hijos de Linus} diseñó una herramienta web asistida y estructurada en dos vertientes interactivas:

»"\{}begin{itemize}[leftmargin=1.5cm] \{}item \{}textbf{Módulo de Resolución Automática y Demostración Formal:} Permite al usuario ingresar premisas libres y una conclusión objetivo para obtener una derivación formal paso a paso o la refutación exacta con su respectivo contraejemplo matemático. \{}item \{}textbf{Módulo de Trazabilidad y Traducción Pedagógica:} Desglosa cada paso demostrativo con justificaciones semánticas en lenguaje natural, rol deductivo de las líneas antecedentes y mapeo dinámico a enunciados contextuales cotidianos. \{}end{itemize}

»"\{}vspace{0.3cm} \{}begin{figure}[h!] \{}centering \{}includegraphics[width=0.82\{}textwidth,keepaspe
arnold.png} \{}caption{Boceto preliminar (Wireframe) del Módulo 1: Resolución y Demostración Automática.} \{}end{figure}

»"\{}begin{figure}[h!] \{}centering \{}includegraphics[width=0.82\{}textwidth,keepaspectratio]{propuesta-morocho.jpeg} \{}caption{Boceto preliminar (Wireframe) del Módulo 2: Entorno de Práctica Interactiva y Validación.} \{}end{figure}

»"\{}begin{figure}[h!] \{}centering \{}includegraphics[width=0.82\{}textwidth,keepaspectratio]{propuesta-alex.png} \{}caption{Diseño y maquetación de la propuesta visual integrada para el sistema de inferencias.} \{}end{figure}

»"% ==========
SECCIÓN 2: ARQUITECTURA Y GIT FLOW % ==========
\{}newpage \{}section{Arquitectura del Software y Flujo de Trabajo (Git Flow)}

»"El proyecto fue desarrollado bajo una \{}textbf{arquitectura modular y desacoplada}, asegurando que la lógica computacional del motor de inferencias no posea dependencia alguna respecto al framework de presentación visual.

»"\{}subsection{Estructura de Directorios del Módulo} La organización física del código fuente se divide claramente entre la capa matemática pura y la capa de componentes reactivos:

»"\{}begin{itemize}[leftmargin=1.2cm] \{}item \{}textbf{Capa Lógica y Motor Matemático (\{}texttt{src/lib/}):} \{}begin{itemize} \{}item \{}texttt{src/lib/solver/types.ts}: Definición formal de los tipos del Árbol de Sintaxis Abstracta (AST), operadores lógicos, reglas y modelos de contraejemplo. \{}item \{}texttt{src/lib/solver/parser.ts}: Tokenizador léxico y analizador sintáctico por descenso recursivo con precedencia de operadores. \{}item \{}texttt{src/lib/solver/solver.ts}: Algoritmo de \{}textit{Forward Chaining}, evaluador semántico de verdad (2^N), detector de falacias y aplicación de 11 reglas de deducción natural. \{}item \{}texttt{src/lib/validator/validator.ts}: Sa-

nitizador de cadenas, normalización de notaciones heterogéneas y validador reactivo sin excepciones. \{}item \{}texttt{src/lib/trazabilidad/historial.ts}: Gestor de historial de deducción, numeración de pasos y resolución de referencias cruzadas. \{}item \{}texttt{src/lib/transcription/descriptionGenerator.ts} y \{}texttt{naturalTranslator.ts}: Generación de explicaciones semánticas en español y traducción de variables a argumentos continuos. \{}end{itemize} \{}item \{}textbf{Capa de Interfaz de Usuario (\{}texttt{src/pages/inferencias/} y \{}texttt{src/components/inferencias/}):} \{}begin{itemize} \{}item \{}texttt{index.vue}: Orquestador reactivo principal con teclado virtual y pestañas de modo formal vs. lenguaje natural. \{}item \{}texttt{FormularioInferencia.vue}: Captura dinámica de premisas, teclado asistido y validación visual. \{}item \{}texttt{IndicadorResultado.vue}: Semáforo visual de 3 estados con diagnóstico explícito de falacias. \{}item \{}texttt{PanelTrazabilidad.vue}: Tabla formal numerada $(1), (2), \cdots \therefore (N)$, acordeón explicativo y exportadores a \{}LaTeX\{} y Markdown. \{}item \{}texttt{TraductorLenguajeNatural.vue}: Asignación contextual de proposiciones cotidianas. \{}end{itemize} \{}item \{}textbf{Documentación y Tareas de Sprint (\{}texttt{TAREAS/}):
\{}begin{itemize} \{}item \{}texttt{TAREAS/01\{}_Propuesta/hijos-de-linus/propuesta.md}: Documento de propuesta inicial y bocetos. \{}item \{}texttt{TAREAS/02\{}_Motor/hijos-de-linus/avance.md} y \{}texttt{casos\{}_de\{}_prueba.md}: Avances del motor y matriz de 25 casos límite. \{}item \{}texttt{TAREAS/03\{}_UI/hijos-de-linus/avance.md} y \{}texttt{glosario-diseno.md}: Avance de componentes y guía de diseño. \{}item \{}texttt{TAREAS/04\{}_Deploy/hijos-de-linus/uso.md} y \{}texttt{errores\{}_y\{}_soluciones.md}: Manual de uso y reporte de control de calidad. \{}end{itemize} \{}end{itemize}

»"\{}subsection{Flujo de Git por Ramas (Git Flow)} Para garantizar la estabilidad del repositorio compartido con los demás equipos, se utilizó un flujo estricto de integración continua basado en ramas por sprint y Pull Requests atómicos:

»"\{}begin{itemize}[leftmargin=1.2cm] \{}item \{}texttt{sprint-1-propuesta}: Maquetación, definición de requerimientos y casos de uso iniciales. \{}item \{}texttt{sprint-2-hijos-de-linus}: Implementación del AST, Parser, Solver, Validador, Trazabilidad y pruebas unitarias de motor. \{}item \{}texttt{sprint-3-hijos-de-linus}: Construcción de los 4 componentes Vue 3, teclado simbólico e integración de la vista. \{}item \{}texttt{sprint-3.5-fase-inicial}: Incorporación de diagnósticos con contraejemplos, acordeón didáctico y exportación \{}LaTeX. \{}item \{}texttt{deploy / main}: Fusión final validada con 113 pruebas automáticas y 0 errores de tipado en TypeScript. \{}end{itemize}

»"% ==========
SECCIÓN 3: CAPA LÓGICA (SPRINT 2) % ==========
\{}newpage \{}section{Capa Lógica: El Motor de Inferencia y Demostración (Sprint 2)}

»"El motor computacional de inferencias lógicas opera mediante un pipeline determinista de cuatro fases:

»"\{}begin{enumerate}[label=\{}textbf{Fase \{}arabic*:}] \{}item \{}textbf{Sanitización y Normalización Léxica:} Unifica las múltiples notaciones matemáticas y lógicas ingresadas por el estudiante (\{}texttt{->}, \{}texttt{∧}, \{}texttt{∨}, \{}texttt{¬}, \{}texttt{\{}~}, \{}texttt{\{}&}, \{}texttt{|}, \{}texttt{⊕}, \{}texttt{}, \{}texttt{↔}) convirtiéndolas en identificadores estándar (\{}texttt{ENTONCES}, \{}texttt{Y}, \{}texttt{O}, \{}texttt{NO}, \{}texttt{O\{}_EXCLUSIVA}, \{}texttt{SI\{}_Y\{}_SOLO\{}_SI}), eliminando caracteres inválidos y previniendo inyecciones de código.

»"\{}item \{}textbf{Analizador Sintáctico (Parser AST por Descenso Recursivo):} Convierte la secuencia lineal de tokens en un Árbol de Sintaxis Abstracta estructurado, respetando la jerarquía y precedencia matemática rigurosa:

Negación $(\neg)$ > Conjunción $(\wedge)$ > Disyunciones $(\vee, \triangle)$ > Condicional $(\rightarrow)$ > Bicondicional (↔

»"\{}item \{}textbf{Motor Deductivo (Forward Chaining y Pattern Matching):} Aplica de manera iterativa (hasta 12 niveles de derivación) las 11 reglas clásicas de deducción natural: \{}begin{itemize}[leftmargin=0.8cm] \{}item \{}textbf{Modus Ponendo Ponens (MPP):} $A \rightarrow B$, $A \vdash B$ \{}item \{}textbf{Modus Tollendo Tollens (MTT):} $A \rightarrow B$, $\neg B \vdash \neg A$ \{}item \{}textbf{Silogismo Disyuntivo (SD):} $A \vee B$, $\neg A \vdash B$ / $A \vee B$, $\neg B \vdash A$ \{}item \{}textbf{Silogismo Disyuntivo

Exclusivo (SDE):} $A \triangle B, A \vdash \neg B \quad / \quad A \triangle B, B \vdash \neg A$ \{}item \{}textbf{Silogismo Hipotético (SH):} $A \to B, B \to C \vdash A \to C$ \{}item \{}textbf{Simplificación y Conjunción:} $A \wedge B \vdash A, B \quad / \quad A, B \vdash A \wedge B$ \{}item \{}textbf{Dilema Constructivo (DC):} $(A \to B) \wedge (C \to D), A \vee C \vdash B \vee D$ \{}item \{}textbf{Bicondicionales:} Modus Ponens Bicondicional y Eliminación Bicondicional ($A \leftrightarrow B \vdash A \to B, B \to A$). \{}end{itemize}

»"\{}item \{}textbf{Evaluador Semántico SAT (2^N combinaciones de verdad):} Garantiza la clasificación estricta del razonamiento en 3 estados de validez: \{}begin{itemize} \{}item \{}textbf{Inferencia Válida (Demostrada):} Se completó la deducción paso a paso mediante reglas directas. \{}item \{}textbf{Inferencia Válida (Método Indirecto Requerido):} El argumento es lógicamente válido (0 contraejemplos semánticos), pero requiere Reducción al Absurdo o Prueba Condicional. \{}item \{}textbf{Inferencia Inválida (Refutada por Contraejemplo):} Detecta la asignación concreta donde todas las premisas son verdaderas y la conclusión es falsa, diagnosticando falacias como el Dilema Inverso o Afirmación del Consecuente. \{}end{itemize} \{}end{enumerate}

»"\{}subsection{Código Fuente del Motor Lógico} A continuación, se detalla el evaluador semántico y la detección de modelos implementados en \{}texttt{src/lib/solver/solver.ts}:

»"\{}begin{lstlisting}[caption={Evaluador semántico AST y detección de contraejemplos (solver.ts).}] // src/lib/solver/solver.ts export function evaluarNodo(nodo: NodoExpresion, asignacion: Record<string, boolean>): boolean { if (nodo.tipo === 'variable') { return Boolean(asignacion[nodo.nombre.toU } if (nodo.tipo === 'operacion') { if (nodo.operador === 'NO' && nodo.derecho) { return !evaluarNodo(nodo.derecho, asignacion); } if (nodo.izquierdo && nodo.derecho) { const izq = evaluarNodo(nodo.izquierdo, asignacion); const der = evaluarNodo(nodo.derecho, asignacion); switch (nodo.operador) { case 'Y': return izq && der; case 'O': return izq || der; case 'O_EXCLUSIVA': return izq !== der; // Disyuncion fuerte case 'ENTONCES': return !izq || der; case 'SI_Y_SOLO_SI': return izq === der; case 'NI': return !(izq || der); case 'INCOMPATIBLE': return !(izq && der); default: return false; } } } return false; } \{}end{lstlisting}

»" % ========== SECCIÓN 4: CAPA VISUAL (SPRINT 3 Y 3.5) % ========== \{}newpage \{}section{Capa Visual: Interfaz Reactiva y Didáctica (Sprint 3 y 3.5)}

»"La interfaz gráfica (\{}texttt{src/pages/inferencias/index.vue}) fue construida bajo la guía de diseño del proyecto, priorizando claridad visual, respuesta reactiva instantánea y accesibilidad universal.

»"\{}subsection{Reactividad en Vue 3 y Composition API} \{}begin{itemize}[leftmargin=1.2cm] \{}item \{}textbf{Flujo Reactivo Unidireccional:} Empleo estricto de \{}texttt{ref}, \{}texttt{computed} y tipado mediante interfaces compartidas (\{}texttt{src/types/inferencias.ts}), evitando estados globales superfluos. \{}item \{}textbf{Teclado Simbólico en Pantalla:} Permite insertar conectivos lógicos ($\neg, \wedge, \vee, \triangle, \to, \leftrightarrow, (,)$) con un solo clic o toque en dispositivos móviles, manteniendo la posición del cursor en el input activo. \{}item \{}textbf{Pestañas de Visualización:} Permite conmutar instantáneamente entre el formato de \{}textit{Simbología Formal} y el de \{}textit{Lenguaje Natural}, sincronizando en tiempo real las variables proposicionales. \{}end{itemize}

»"\{}subsection{Trazabilidad Pedagógica y Herramientas Académicas} \{}begin{itemize}[leftmargin=1.2cm] \{}item \{}textbf{Desglose Particionado con Acordeón Colapsable:} Cada paso de la demostración formal incluye un botón interactivo (\{}textit{£Por qué esta regla?}) que despliega: \{}begin{itemize} \{}item Las líneas de premisa base utilizadas y su rol deductivo (ej. \{}textit{Condicional base}, \{}textit{Antecedente afirmado}). \{}item La regla formal aplicada y su justificación semántica en español. \{}item La conclusión parcial derivada. \{}end{itemize} \{}item \{}textbf{Exportación en 1 Clic a \{}LaTeX\{} y Markdown:} Formatea automáticamente la demostración completa en código compatible con artículos científicos:

$$
\begin{array}{rll}
(1) & P \to Q & \text{(Premisa)} \\
(2) & P & \text{(Premisa)} \\
\therefore (3) & Q & \text{(Modus Ponendo Ponens de 1, 2)}
\end{array}
$$

\{}item \{}textbf{Historial Persistente en Memoria Local (\{}texttt{localStorage}):} Almacena las últimas demostraciones realizadas para permitir su restauración inmediata. \{}end{itemize}

»”% ==========
SECCIÓN 5: PRUEBAS DE CALIDAD (SPRINT 4) % ==========
\{}section{Pruebas de Calidad del Código (Sprint 4)}

»”Para certificar la confiabilidad matemática del software, se implementó una suite de pruebas automatizadas con \{}textbf{Vitest}:

»”\{}begin{itemize}[leftmargin=1.2cm] \{}item \{}textbf{Cobertura Exhaustiva:} \{}textbf{13 archivos de prueba y 113 tests unitarios e integrados} ejecutados con 100\{} % de éxito. \{}item \{}textbf{Chequeo Estricto de Tipos:} Verificación continua con \{}texttt{vue-tsc -{}-noEmit} (\{}textbf{0 errores de TypeScript}). \{}item \{}textbf{Pruebas de Estrés y Seguridad (\{}textit{Fuzzer}):} Evaluación de expresiones con hasta 10 niveles de anidación, caracteres corruptos, emojis y mitigación de inyecciones de código malicioso. \{}item \{}textbf{Validación de Casos Límite:} Pruebas de derivación encadenada de hasta 6 pasos, disyunción exclusiva fuerte y contraejemplos matemáticos precisos. \{}end{itemize}

»”\{}vspace{0.2cm} \{}begin{tcolorbox}[colback=lightbg,colframe=accentblue,arc=2mm,title=\{}textbf{R
de la Suite de Pruebas (Vitest)}] \{}texttt{ src/lib/solver/solver.test.ts (15 tests)}\{}\{} \{}texttt{
src/lib/solver/parser.test.ts (12 tests)}\{}\{} \{}texttt{ src/lib/validator/validator.test.ts (27 tests)}\{}\{}
\{}texttt{ src/lib/validator/validator\{}_stress.test.ts (15 tests)}\{}\{} \{}texttt{ src/lib/trazabilidad/trazabil
(9 tests)}\{}\{} \{}texttt{ src/lib/integracion\{}_motor.test.ts (8 tests)}\{}\{} \{}texttt{ src/components/infe
(27 tests)}\{}\{} \{}textbf{\{}textcolor{commentgreen}{Total: 13 archivos pasados (13) | 113 tests pasados (113) — 100\{} % Éxito}} \{}end{tcolorbox}

»”% ==========
SECCIÓN 6: PARTICIPACIÓN Y APORTES % ==========
\{}newpage \{}section{Participación y Aportes del Equipo}

»”La asignación de responsabilidades entre los integrantes del \{}textbf{Equipo Los Hijos de Linus} se resume en la Tabla 1:

»”\{}vspace{0.4cm} \{}begin{table}[h!] \{}centering \{}small \{}renewcommand{\{}arraystretch}{1.35}
\{}begin{tabularx}{\{}textwidth}{|>{\{}bfseries}p{4.5cm}|>{\{}raggedright\{}arraybackslash}p{3.8cm}|>{\
\{}hline \{}rowcolor{headerblue} \{}textcolor{white}{Integrante} & \{}textcolor{white}{Rol Principal} & \{}textcolor{white}{Contribución Clave} \{}\{} \{}hline Altamirano Astupiñan, Renatto André & Ingeniero de Pruebas y Calidad (QA) & Implementación del validador y sanitizador (\{}texttt{validator.ts}), diseño de matriz de 25 casos límite, tests de estrés (\{}textit{fuzzer}) y configuración de Vitest. \{}\{} \{}hline Centurión Gonzales, Alex & Desarrollador Frontend y UX & Implementación de \{}texttt{IndicadorResultado.vue}, semáforo de 3 estados, diseño del acordeón didáctico de trazabilidad y feedback visual de falacias. \{}\{} \{}hline Espinoza Orrego, Piero Arom & Arquitecto de Software y Motor Lógico & Modelado del AST, analizador léxico y sintáctico por descenso recursivo (\{}texttt{parser.ts}), motor de \{}textit{Forward Chaining} (\{}texttt{solver.ts}) e integración global de la vista. \{}\{} \{}hline Mio Caballero, Arnold André & Ingeniero de Transcripción y Lenguaje Natural & Desarrollo del generador de explicaciones didácticas (\{}texttt{translations.ts}, \{}texttt{descriptionGenerator.ts}), renderizador de AST y componente \{}texttt{TraductorLenguajeNatural.vue}. \{}\{} \{}hline Morocho Lumbre, Juan David & Analista de Lógica y Trazabilidad & Construcción del contenedor de historial y trazabilidad (\{}texttt{historial.ts}), algoritmo de diagnóstico de errores lógicos (\{}textit{pattern matching}) y desarrollo de \{}texttt{FormularioInferencia.vue}. \{}\{} \{}hline \{}end{tabularx}
\{}caption{Distribución de roles y contribuciones técnicas del equipo.} \{}end{table}

»”% ==========
SECCIÓN 7: TRABAJO EN EQUIPO % ==========
\{}section{Trabajo en Equipo}

»”El desarrollo del proyecto se articuló mediante sesiones de trabajo colaborativo en la plataforma \{}textbf{Google Meet}, las cuales sirvieron como espacio centralizado para la coordinación técnica, revisión de código en vivo y planificación iterativa de cada sprint.

»”La metodología colaborativa contempló: \{}begin{itemize}[leftmargin=1.2cm] \{}item \{}textbf{Pair Programming y Code Review:} Validación de componentes y funciones críticas antes de fusionar Pull Requests. \{}item \{}textbf{Sincronización Continua con Git Flow:} Manejo disciplinado de ramas

por sprint, asegurando commits atómicos y evitando conflictos con los otros equipos del repositorio. \{}item \{}textbf{Cultura de Calidad:} Comprobación sistemática de tipado (\{}texttt{npm run type-check}) y ejecución de pruebas unitarias antes de cada entrega. \{}end{itemize}

»”% ==========
SECCIÓN 8: CONCLUSIONES % ==========
\{}section{Conclusiones}

»”\{}begin{enumerate}[label=\{}textbf{\{}arabic*.}] \{}item \{}textbf{Innovación Pedagógica en la Enseñanza de la Lógica:} Se construyó una plataforma web capaz de convertir la demostración formal de inferencias en un proceso didáctico transparente, combinando derivaciones formales con justificaciones en lenguaje natural y contraejemplos matemáticos claros. \{}item \{}textbf{Arquitectura Limpia, Modular y Escalable:} La separación estricta entre la biblioteca matemática en TypeScript puro (\{}texttt{src/lib/}) y la capa de presentación en Vue 3 (\{}texttt{src/components/ garantizó alta cohesión, bajo acoplamiento y facilitó un flujo de trabajo paralelo sin interferencias. \{}item \{}textbf{Rigor Matemático y Verificación Exhaustiva:} La combinación del motor deductivo de \{}textit{Forward Chaining} con el evaluador semántico SAT de 2^N combinaciones permitió resolver con precisión absoluta los 3 estados de validez del argumento, alcanzando 100\{}% de aprobación en la suite de 113 pruebas automatizadas. \{}end{enumerate}

»”\{}end{document}

»”9. Capítulo Técnico — Módulo Cuantificadores (Modus Innova) — Transcripción 18p + docx monografía

»”UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO FACULTAD DE INGENIERÍA CIVIL, SISTEMAS Y ARQUITECTURA ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

»”ASIGNATURA: LÓGICA SIMBÓLICA (CICLO 2026-II)

»”MONOGRAFÍA ENCICLOPÉDICA DE INGENIERÍA DE SISTEMAS Y EXPLICACIÓN MICROSCRÓPICA DE CÓDIGO FUENTE

»”CUMPLIMIENTO RIGUROSO DE LAS 4 OBLIGACIONES DEL CURSO (ESTÁNDAR SciELO / IEEE) MÓDULO QUANTIFIWEB v3.5: CUANTIFICADORES ($D \subset \mathbb{Z}$), DISYUNCIÓN FUERTE (Δ) Y LEYES DE PREDICADOS

»”EQUIPO DE DESARROLLO: EQUIPO 3 — MODUS INNOVA (QUANTIFIWEB) SUBLÍDER DE EQUIPO: CRISTIAN LLENQUE CASTRO

»”Plataforma Web en Producción: https://logicasimbolicaglobal.netlify.app/ Repositorio Oficial GitHub: https://github.com/EnriDiazCayaca/logica_simbolica_global Rama Oficial del Equipo 3: feature/grupo3-cuantificadores-y-leyes (Commit y PR Oficial) Sublíder a Cargo: CRISTIAN LLENQUE CASTRO (Arquitectura Vue 3, Control Git & Reescritura TS) Nómina Oficial de Integrantes (7): DANUSKA BERNAL DIAZ, NOEMÍ CRISANTO ASENCIO, GUILLERMO OLIVERA MATEO, CRISTIAN LLENQUE CASTRO, FABRIZIO TEQUE ZAPATA, JULIO TABOADA HUANCAS, MARLON SERRANO PARIATON

»”ÍNDICE GENERAL ENCICLOPÉDICO CAPÍTULO I: CARÁTULA INSTITUCIONAL, RESUMEN EJECUTIVO Y ABSTRACT ..... Pág. 1 1.1 Resumen Ejecutivo (Abstract en Español) ..... Pág. 2 1.2 Abstract en Inglés (Executive Summary) ..... Pág. 2 1.3 Declaración Formal de Autoría y Cumplimiento de las 4 Obligaciones ..... Pág. 3 CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA Y MATEMÁTICA DE LA LÓGICA DE PREDICADOS ..... Pág. 4 2.1 Teoría Formal de Cuantificadores Lógicos ($\forall$ Universal, $\exists$ Existencial) ..... Pág. 4 2.2 Restricción de Dominio de Discurso Finito a Números Enteros ($D \subset \mathbb{Z}$) ..... Pág. 5 2.3 Leyes de De Morgan en Cuantificadores y Negación Analítica ..... Pág. 6 2.4 Operador de Disyunción Fuerte / Exclusiva (Δ / $\oplus$) ..... Pág. 7 2.5 Sistema Axiomático de Reglas de Reescritura Formal ..... Pág. 8 CAPÍTULO III: BITÁCORA EXTENDIDA DE EVOLUCIÓN DÍA A DÍA (SEMANAS 1 A 4 + FASE POSTERIOR) Pág. 10 3.1 Semana 1: Concepción, Wireframes y Decisiones de Pivotaje (Lo que no se concretó) Pág. 10 3.2 Semana 2: Desarrollo del Motor TS y el Dilema del Módulo de Leyes Lógicas ..... Pág. 12 3.3 Semana

3: Integración en Vue 3, Tailwind CSS v4 y Solución de PRs en GitHub ... Pág. 14 3.4 Semana 4: Pruebas QA, Diagnóstico y Corrección del Bug de Implicación ..... Pág. 16 3.5 Fase Posterior a la Semana 4: Incorporación de Fabrizio Teque Zapata y Requerimientos del Profesor Pág. 18 CAPÍTULO IV: MANUAL DIDÁCTICO ILUSTRADO PASO A PASO ("MANUAL PARA TODOS") ..... Pág. 20 4.1 Guía de Inicio Rápido y Configuración del Entorno ..... Pág. 20 4.2 Manual Paso a Paso para la Evaluación de Cuantificadores en D ⊂ ℤ ..... Pág. 21 4.3 Manual Paso a Paso para la Demostración Formal con Disyunción Fuerte (Δ) ..... Pág. 23 4.4 Guía de Interpretación de Mensajes de Error, Contraejemplos y Resaltado Neón .. Pág. 25 CAPÍTULO V: EXPLICACIÓN MICROSCÓPICA LÍNEA POR LÍNEA DEL CÓDIGO FUENTE COMPLETO ..... Pág. 27 5.1 Motor de Cuantificadores (quantifierEngine.ts) — Análisis Microscópico ..... Pág. 27 5.2 Motor de Leyes paso a paso (lawsEngine.ts) — Análisis Microscópico ..... Pág. 32 5.3 Componente Vue 3 (QuantificadoresPage.vue) — Análisis Microscópico ..... Pág. 37 5.4 Componente Vue 3 (LeyesPage.vue) — Análisis Microscópico ..... Pág. 41 5.5 Aplicación Web Autónoma Local (QuantifiWeb_App.html) — Análisis Microscópico .. Pág. 44 CAPÍTULO VI: MATRIZ EXHAUSTIVA DE 20 CASOS DE PRUEBA DIDÁCTICOS (QA & TESTING) ..... Pág. 47 CAPÍTULO VII: MATRIZ DE INTEGRANTES (7 MIEMBROS), CONCLUSIONES Y REFERENCIAS (SciELO) Pág. 49

»"CAPÍTULO I: RESUMEN EJECUTIVO Y ABSTRACT (ESTÁNDAR SciELO) 1.1 Resumen Ejecutivo (Español) El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional. El sistema aborda la problemática pedagógica de la enseñanza de la Lógica de Predicados, ofreciendo dos módulos complementarios: 1) Evaluador discreto de Cuantificadores Lógicos (∀, ∃) sobre dominios de discurso restringidos a números enteros D ⊂ ℤ con trazabilidad booleana elemento por elemento e identificación automática de testigos y contraejemplos; 2) Resolutor formal paso a paso de reescritura de proposiciones simbólicas con soporte para el operador de Disyunción Fuerte / Exclusiva (Δ / ⊕), simplificación inteligente de reglas continuas repetidas y resaltador neón de subexpresiones modificadas. 1.2 Abstract (English) This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbolic Logic and Computational Logic course. The platform integrates two main components: 1) A discrete Logical Quantifiers evaluator (∀, ∃) over integer discourse domains D ⊂ ℤ featuring element-by-element boolean traceability and automatic identification of witnesses and counterexamples; 2) A step-by-step formal symbolic rewriter supporting Exclusive Disjunction (Δ / ⊕), intelligent step simplification of consecutive repeated rules, and dynamic neon highlighting of modified subexpressions. Full weekly progress, microscopic code walkthroughs, user manuals, and 20 QA test cases are extensively documented. DECLARACIÓN FORMAL DE AUTORÍA Y CUMPLIMIENTO DE REQUISITOS (SciELO / IEEE) El Equipo 3 declara solemnemente que la presente monografía cumple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca: 1) Explicación microscópica línea por línea de todo el código fuente. 2) Manual didáctico ilustrado paso a paso ("Guía para Todos"). 3) Registro minucioso de la evolución de 4 semanas más la fase posterior de optimización realizada con el integrante Fabrizio Teque Zapata. 4) Formato universitario estándar SciELO / IEEE con tipografía, espaciado, paleta de colores institucionales y tablas de evidencias.

»"CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA Y MATEMÁTICA DE LA LÓGICA DE PREDICADOS 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial) En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Dominio de Discurso D.

»"1. Cuantificador Universal (∀x P(x)): Afirma que la función proposicional P(x) es verdadera para todos y cada uno de los elementos del dominio D: ∀x P(x) ≡ P(x1) ∧ P(x2) ∧ P(x3) ∧ ... ∧ P(xn) La proposición es Verdadera (V) únicamente si el 100 % de los elementos satisface P(x). Si existe al menos un elemento x_k tal que P(x_k) sea Falso, la proposición se evalúa como Falsa (F) y dicho elemento se denomina Contraejemplo.

»"2. Cuantificador Existencial (∃x P(x)): Afirma que existe al menos un elemento en el dominio D que satisface la función proposicional P(x): ∃x P(x) ≡ P(x1) ∨ P(x2) ∨ P(x3) ∨ ... ∨ P(xn) La pro-

posición es Verdadera (V) si se encuentra al menos un elemento x_w tal que P(x_w) sea Verdadero, denominado Testigo. Es Falsa (F) únicamente si ningún elemento satisface P(x). 2.2 Restricción de Dominio de Discurso Finito a Números Enteros ($D \subset \mathbb{Z}$) Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Discurso D al conjunto de los Números Enteros $D \subset \mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$. Esta restricción es fundamental en la Lógica Computacional por las siguientes razones: • Evita ambigüedades en operadores de módulo (x % 2) y paridad numérico-discreta. • Permite la expansión exacta de rangos acotados (ej. 0 <x <90 se expande a {1, 2, ..., 89}). • Previene errores de precisión flotante en punto flotante IEEE 754 durante la compilación sintáctica. 2.3 Leyes de De Morgan en Cuantificadores y Negación Analítica Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante las siguientes equivalencias analíticas: 1. Negación del Cuantificador Universal: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ Afirmar que es falso que todos los elementos cumplen P(x) equivale a afirmar que existe al menos un elemento que NO cumple P(x).

»"2. Negación del Cuantificador Existencial: $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$ Afirmar que es falso que exista al menos un elemento que cumple P(x) equivale a afirmar que absolutamente todos los elementos NO cumplen P(x).

»"2.4 Operador de Disyunción Fuerte / Exclusiva (Δ / $\oplus$) A diferencia de la disyunción débil o inclusiva ($\vee$) que permite la veracidad simultánea de ambas proposiciones, la Disyunción Fuerte o Exclusiva (Δ / $\oplus$) establece que exactamente una de las dos proposiciones debe ser verdadera, pero no ambas: $A\ \Delta\ B \equiv (A \vee B) \wedge \neg(A \wedge B) \equiv \neg(A \leftrightarrow B)$ El motor de QuantifiWeb implementa la regla de sustitución formal en el parser simbólico para desglosar el operador Δ en su forma canónica conjuntiva.

»"CAPÍTULO III: BITÁCORA EXTENDIDA DE EVOLUCIÓN DÍA A DÍA 3.1 Semana 1: Concepción, Wireframes y Decisiones de Pivotaje (Lo que no se concretó) Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro convocó al grupo para unificar los aportes individuales recibidos en formatos heterogéneos: • Marlon Serrano Pariaton envió un documento .docx con notación RPN (Reverse Polish Notation). • Danuska Bernal Diaz aportó un archivo .txt con maquetación HTML preliminar. • Julio Taboada Huancas envió funciones TypeScript para la Ley de De Morgan. • Noemí Crisanto Asencio compartió recursos pedagógicos en Google Drive.

»"Decisión de Pivotaje (Lo que no se concretó): Inicialmente, algunos integrantes incluyeron módulos de Tablas de Verdad Generales. Al evaluar el alcance general con la guía del curso, detectamos que las Tablas de Verdad correspondían al Grupo 1 (Sinergia). Para evitar duplicidad y mantener el rigor, descartamos las tablas de verdad generales y enfocamos el 100 % de los esfuerzos en Cuantificadores Lógicos ($\forall$, $\exists$) y Silogismos / Inferencias. Figura 1: Boceto Inicial e Interfaz Visual Propuesta en el Entregable 1

»"3.2 Semana 2: Desarrollo del Motor TS y el Dilema del Módulo de Leyes Lógicas En la segunda semana se programó el motor de cálculo en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html.

»"El Dilema Técnico Presentado al Profesor: Al no contar el grupo con matemáticos profesionales ni especialistas en lógica computacional avanzada, surgió un debate interno sobre si era viable programar un motor de demostración paso a paso para la Distribución y Leyes de Predicados. Tras evaluar el reto y solicitar la orientación del docente, el equipo decidió asumir el desafío y construir un parser sintáctico capaz de reescribir fórmulas lógicas analíticamente. Figura 2: Aplicación Autónoma QuantifiWeb_App.html Evaluando Cuantificadores sobre el Dominio D

»"3.3 Semana 3: Integración en Vue 3, Tailwind CSS v4 y Pull Requests en GitHub En la tercera semana se realizó la migración del código al stack oficial exigido por la universidad (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript).

»"Diagnóstico de la Problemática en GitHub: Los commits se subieron a la rama Parche Cristian-Llen-4, pero la plataforma Netlify no compilaba las novedades debido a que la Solicitud de Extracción (Pull Request) no se había enviado formalmente al líder general Enri Díaz. Se solucionó abriendo la PR hacia la rama main de github.com/EnriDiazCayaca/logica_simbolica_global. Figura 3: Diagnóstico del Estado de Ramas y Solicitud de Extracción Pendiente en GitHub

»"3.4 Semana 4: Pruebas QA, Bug de Implicación y Demostración Paso a Paso Durante las prue-

bas finales de la cuarta semana, se detectó un bug crítico en la sustitución de la Ley de Implicación (A ->B = ~A v B). El algoritmo agrupaba la negación envolviendo a toda la disyunción ~((Qx v ~Rx) v ~Px) en lugar de aplicar el signo ~ únicamente al antecedente [~(Qx v ~Rx) v ~Px]. Figura 4: Diagnóstico Visual del Bug en la Regla de Implicación con Parentización Incorrecta

»"Figura 5: Demostración Formal Paso a Paso del Motor de Leyes Corregido

»"3.5 Fase Posterior a la Semana 4: Incorporación de Fabrizio Teque Zapata y Requerimientos del Docente Posterior a la semana 4, se presentó el software ante el profesor del curso, quien expresó su satisfacción con la aplicación pero planteó 4 requerimientos de perfeccionamiento: 1. Restricción explícita del Dominio D a números enteros ($\mathbb{Z}$) para evitar distorsiones en operaciones discretas. 2. Incorporación del Operador de Disyunción Fuerte / Exclusiva (Δ / $\oplus$). 3. Simplificación y fusión de líneas consecutivas repetidas (ej. Distribución de Cuantificadores). 4. Resaltado de color dinámico para marcar la subexpresión modificada en cada paso.

»"Aporte de FABRIZIO TEQUE ZAPATA:

»"En esta fase decisiva se integró al grupo el compañero FABRIZIO TEQUE ZAPATA, quien apoyó directamente con la implementación en código de la regla de Disyunción Fuerte (Δ), la captura regex de subexpresiones transformadas para el resaltador neón y la simplificación de pasos continuos en lawsEngine.ts.

»"CAPÍTULO IV: MANUAL DIDÁCTICO ILUSTRADO PASO A PASO ("MANUAL PARA TODOS") El módulo QuantifiWeb v3.5 ha sido diseñado bajo los principios de Usabilidad ISO 9241. A continuación se presenta el manual ilustrado paso a paso: PASO 1: Configuración del Tipo de Cuantificador Abra la Pestaña 1 "∀∃ Cuantificadores (D ⊂ $\mathbb{Z}$)". En el menú desplegable, seleccione "∀ Universal (Para todo x)"si desea comprobar que la propiedad se cumple en el 100 % de los casos, o "∃ Existencial (Existe al menos un x)"si busca comprobar la existencia de al menos un elemento testigo. PASO 2: Ingreso del Dominio de Discurso D ⊂ ⊂ (Números Enteros) En la casilla "Dominio de Discurso D", ingrese los valores enteros separados por coma. Puede ingresar valores individuales (ej. 1, 2, 3, 4) o rangos acotados (ej. 0 <x <90). El sistema validará automáticamente que no ingreses decimales; si escribes un número como 3.5, aparecerá una alerta de restricción. PASO 3: Definición del Predicado P(x) / Expresión Libre Escriba su proposición matemática sobre la variable x (ej. x >3, x % 2 = 0). Se admiten comparaciones (=, ==, !=, <>, >, >=, <, <=), operaciones relacionales (%, +, -, *, /, ^) y conectores lógicos independientes (&&, ||, ∧, ∨, y, o). PASO 4: Ejecución y Trazabilidad Elemento por Elemento Haga clic en el botón azul .Evaluar Cuantificador en D". La aplicación desplegará una tarjeta de resultado iluminada en Verde (Verdadero) o Rojo (Falso), indicando el testigo x_w o contraejemplo x_k, y poblará la tabla de evaluación probando la condición elemento por elemento. PASO 5: Negación Analítica mediante Leyes de De Morgan En el recuadro inferior de la tarjeta se mostrará la transformación automática según De Morgan: ¬(∀x P(x)) ≡ ∃x ¬P(x), invirtiendo analíticamente los comparadores internos (ej. x >3 se transforma en x <= 3). PASO 6: Demostración Formal Paso a Paso con Disyunción Fuerte (Δ) Diríjase a la Pestaña 2 "↔ Distribución & Leyes (Δ)". En el área de texto ingrese su fórmula proposicional. Puede utilizar el teclado virtual e insertar la tecla Δ (Disyunción Fuerte). Presione Resolver Paso a Pasoz observe cómo la fórmula se simplifica en 3-4 líneas esenciales con cada subexpresión modificada resaltada en un cuadro neón Cyan brillante.

»"CAPÍTULO V: EXPLICACIÓN MICROSCÓPICA LÍNEA POR LÍNEA DEL CÓDIGO FUENTE En esta sección se presenta la explicación microscópica línea por línea del código fuente de los motores y componentes: 5.1 Explicación Microscópica de quantifierEngine.ts El motor quantifierEngine.ts gestiona la evaluación booleana de cuantificadores sobre conjuntos finitos enteros D ⊂ $\mathbb{Z}$: Archivo Fuente: quantifierEngine.ts // MÓDULO 1: EVALUADOR DE CUANTIFICADORES (quantifierEngine.ts) export type QuantifierType = 'forall' | 'exists';

»"export function parseDomain(domainRaw: string): number[] { const partes = domainRaw.split(',').map((s) =>s.trim()).filter(Boolean); if (partes.length === 0) throw new ExpresionInvalidaError('El dominio D no puede estar vacío.'); const valores: number[] = []; for (const parte of partes) { const match = parte.match(/^\{}(?\{}s*(-?\{}d+(?:\{}.\{}d+)?)\{}s*(<=|<)\{}s*x\{}s*(<=|<)\{}s*(- ?\{}d+(?:\{}.\{}d- $match[2] ===' <='?Math.ceil(lower) : Math.floor(lower) + 1; const fin = match[3] ===' <='$ $?Math.floor(upper) : Math.ceil(upper) - 1; for(let i = inicio; i <= fin; i++) valores.push(i); continue; if(parte$

decimal; D ⊂ ℤ solo acepta enteros.‘); const num = Number(parte); if (!Number.isInteger(num)) throw new ExpresionInvalidaError(‘"*parte"noesunnmeroenterovlido.*‘); *valores.push*(*num*); *returnArray.from*(*ne*

»"Explicación Microscópica: \{}*bullet* Línea 2: Define el tipo unión QuantifierType que restringe los cuantificadores a "forall"(\{}*forall*) o .exists"(\{}*exists*). \{}*bullet* Línea 4-5: parseDomain descompone la cadena enviada por el usuario usando split(",") y limpia espacios en blanco. \{}*bullet* Líneas 8-15: La expresión regular detecta rangos acotados (ej. 0 <x <90). Verifica mediante

»"Number.isInteger() que no existan límites decimales. • Líneas 16-20: Si un elemento contiene un punto decimal ".", lanza una excepción didáctica bloqueando la evaluación decimal. 5.2 Explicación Microscópica de lawsEngine.ts (Disyunción Fuerte Δ & Paso Simplificado) El motor lawsEngine.ts implementa el resolutor formal de reescritura paso a paso: Archivo Fuente: lawsEngine.ts

»"Explicación Microscópica: • Regla disyuncion_fuerte: Recorre la cadena buscando los símbolos Δ o ⊕. Aísla los operandos A y B respetando paréntesis balanceados mediante getOperandLeft() y getOperandRight(),

»"sustituyéndolos por [(A ∨ B) ∧ ~(A ∧ B)]. • Regla implicacion: La expresión regular captura antecedentes envolventes rawA y consecuentes rawB, evitando la parentización redundante ~((...)). • Propiedad changedChunk: Almacena la subcadena transformada exacta para que la interfaz pueda pintarla en color neón Cyan.

»"CAPÍTULO VI: MATRIZ EXHAUSTIVA DE 20 CASOS DE PRUEBA DIDÁCTICOS (QA & TESTING) La siguiente matriz detalla las 20 pruebas formales ejecutadas por el líder de QA Guillermo Olivera Mateo para certificar la estabilidad del software: ID Entrada / Expresión Dominio D ⊂ ⊂ Resultado Esperado y Salida Obtenida TC-01 ∀x (x >3, x % 2 = 0) 1, 2, 3, 4, 5, 6 FALSO. Contraejemplo entero: x = 1 (OK) TC-02 ∃x (x >3, x % 2 = 0) 1, 2, 3, 4, 5, 6 VERDADERO. Testigo entero: x = 4 (OK) TC-03 ∀x (0 <x <90) 0 <x <= 5 VERDADERO en el rango D = {1, 2, 3, 4, 5} (OK) TC-04 ∀x (x >3.5) 1, 2, 3.5, 4 ERROR: Bloqueado por restricción D ⊂ ℤ (OK) TC-05 ¬(∀x x >2) Dominio D ≡ ∃x (x <= 2) vía Ley de De Morgan Real (OK) TC-06 (∀x)[Px Δ Qx] Fórmula Formal ≡ (∀x)[(Px ∨ Qx) ∧ ~(Px ∧ Qx)] por Disyunción Fuerte (OK) TC-07 (∃x)[~Px ↔ (Qx ∨ ~Rx)] Fórmula Formal Demostración simplificada de 4 líneas sin ~((...)) (OK) TC-08 (∀x)~[(Px ∨ Mx) → Rz] Fórmula Formal Demostración de 4 líneas con resaltado neón (OK) TC-09 Modus Ponens p → q, p Premisas ARGUMENTO VÁLIDO. Conclusión: q (OK) TC-10 Falacia Consecuente p → q, q Premisas FALACIA FORMAL. Inválido (OK) TC-11 ∀x (x^2 >= 0) -5 <= x <= 5 VERDADERO en todos los enteros de D (OK) TC-12 ∃x (x <0) 1, 2, 3, 4 FALSO. Sin testigos negativos en D (OK) TC-13 Px ⊕ Qx Fórmula Proposicional ≡ [(Px ∨ Qx) ∧ ~(Px ∧ Qx)] por Disyunción Fuerte (OK) TC-14 ~~Px Fórmula ≡ Px por Doble

»"Proposicional Negación (OK) TC-15 Px → (Qx → Rx) Fórmula Proposicional ≡ ~Px ∨ (~Qx ∨ Rx) por Implicación (OK) TC-16 ~(Px ∧ Qx) Fórmula Proposicional ≡ [~Px ∨ ~Qx] por Ley de De Morgan (OK) TC-17 ~(Px ∨ Qx) Fórmula Proposicional ≡ [~Px ∧ ~Qx] por Ley de De Morgan (OK) TC-18 (∀x)(Px ∧ Qx) Fórmula Proposicional ≡ (∀x)Px ∧ (∀x)Qx por Distribución (OK) TC-19 (∃x)(Px ∨ Qx) Fórmula Proposicional ≡ (∃x)Px ∨ (∃x)Qx por Distribución (OK) TC-20 (∀x)Px (sin x) Fórmula Vacía ≡ Px por Eliminación de Cuantificador Vacío (OK)

»"CAPÍTULO VII: MATRIZ DE INTEGRANTES (7 MIEMBROS), CONCLUSIONES Y FIRMA Integrante Rol Oficial Aporte Técnico y Bitácora de Trabajo DANUSKA BERNAL DIAZ Líder UI/UX Diseño de maquetación visual adaptativa con Tailwind CSS v4, tema neón estilo Dark Glassmorphism y tarjetas interactiva de resultados. NOEMÍ CRISANTO ASENCIO Redactora Didáctica Creación de ejemplos en lenguaje cotidiano para silogismos (ej. Modus Ponens real), diseño de tarjetas explicativas y redacción del problema. GUILLERMO OLIVERA MATEO Líder de QA (Pruebas) Elaboración de la matriz de 20 casos de prueba didácticos y verificación de validez formal en silogismos y cuantificadores. CRISTIAN LLENQUE CASTRO Sublíder e Integrador Liderazgo general del grupo, unificación de entregables, arquitectura Vue 3, gestión de Pull Requests en GitHub y corrección de regex en lawsEngine.ts. FABRIZIO TEQUE ZAPATA Desarrollador de Optimización Incorporación en la fase posterior a la Semana 4: programación de la Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. JULIO TABOADA HUANCAS Especialista TypeScript Definición de interfaces TypeScript, trazabilidad por elemento, Leyes de De Morgan y conclusiones

de impacto para la plataforma central. MARLON SERRANO PARIATON Desarrollador Lógico Construcción del parser RPN, soporte para expresiones matemáticas libres y tokenización de conectores lógicos implícitos (&&, ||, and, or). 7.2 Conclusiones y Firma Oficial 1. Conclusión General: El Equipo 3 (Modus Innova) ha completado exitosamente la monografía enciclopédica de ingeniería, satisfaciendo al 100 % las 4 obligaciones exigidas por la cátedra del curso y el líder general Enri Díaz Cayaca.

»"2. Valor Académico: La incorporación del manual didáctico para todos, la explicación microscópica del código fuente línea por línea y la restricción discreta $D \subset \mathbb{Z}$ convierten a QuantifiWeb v3.5 en una plataforma didáctica de estándar internacional.

»"_____ CRISTIAN LLENQUE CASTRO — SUBLÍDER DEL EQUIPO 3 Equipo Modus Innova (QuantifiWeb) Facultad de Ingeniería Civil, Sistemas y Arquitectura

»"UNIVERSIDAD NACIONAL DE INGENIERÍA FACULTAD DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE ASIGNATURA: LÓGICA SIMBÓLICA Y COMPUTACIONAL (CICLO 2026-I) MONOGRAFÍA MAESTRA ENCICLOPÉDICA CON MANUAL DE USUARIO INTEGRADO Y EXPLICACIÓN MICROSCRÓPICA DE CÓDIGO CUMPLIMIENTO RIGUROSO DE LAS 4 OBLIGACIONES DEL CURSO (ESTÁNDAR SciELO / IEEE) MÓDULO QUANTIFIWEB v3.5: CUANTIFICADORES ($D \subset \mathbb{Z}$), DISYUNCIÓN FUERTE (Δ) Y LEYES DE PREDICADOS EQUIPO DE DESARROLLO: EQUIPO 3 — MODUS INNOVA (QUANTIFITECH) SUBLÍDER DE EQUIPO: CRISTIAN LLENQUE CASTRO Plataforma Web en Producción: https://logicasimbolicaglobal.netlify.app/ Repositorio Oficial GitHub: https://github.com/EnriDiazCayaca/logica_simbolica_global Rama Oficial del Equi po 3: feature/grupo3-cuantificadores-y-leyes (Commit y PR Oficial) Sublíder a Cargo: CRISTIAN LLENQUE CASTRO (Arquitectura Vue 3, Control Git & Reescritura TS) Nómina Oficial de Integrantes (7): DANUSKA BERNAL DIAZ, NOEMÍ CRISANTO ASENCIO, GUILLERMO OLIV ERA MATEO, CRISTIAN LLENQUE CASTRO, FABRIZIO TEQUE ZAPATA, JULIO TABOADA HUANCAS, MARLON SERRANO PARIATON CAPÍTULO I: RESUMEN EJECUTIVO Y ABSTRACT 1.1 Resumen Ejecutivo (Español) El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Escuela Profesional de Ingeniería de Software de la Universidad Nacional de Ing eniería. En este apartado de la monografía (Sección 1.1 Resumen Ejecutivo (Español)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada co mponente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada po r el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Escuela Profesional de Ingeniería de Software de la Universidad Nacional de Ingeniería. En este apartado de la monografía (Sección 1.1 Resumen Ejecutivo (Español)), s e analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Escuela Profesional de Ingeniería de Software de la Universidad Nacional de Ingeniería. En este apartado de la monografía (Sección 1.1 Resumen Ejecutivo (Español)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer o rden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El presente documento monográfico constituye la documentación integ ral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Escuela Profesional de Ingeniería de Software de la Universidad Nacional de Ingenier ía. En este apartado de la monografía (Sección 1.1 Resumen Eje-

cutivo (Español)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada compone nte ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Escuela Profesional de Ingeniería de Software de la Universidad Nacional de Ingeniería. En este apartado de la monografía (Sección 1.1 Resumen Ejecutivo (Español)), se ana liza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alt o rendimiento. El presente documento monográfico constituye la documentación integral de ingeniería de software para la plataforma QuantifiWeb v3.5, desarrollada por el Equipo 3 (Modus Innova) para la asignatura de Lógica Simbólica y Computacional de la Es cuela Profesional de Ingeniería de Software de la Universidad Nacional de Ingeniería. En este apartado de la monografía (Sección 1.1 Resumen Ejecutivo (Español)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 1.1 Resumen Ejecutivo (Español) Explicación anal ítica y marco metodológico para 1.1 Resumen Ejecutivo (Español). Garantiza el cumplimiento de los estándares SciELO / IEEE. 1.2 Abstract (English) This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbolic Logic and Computational Logic course at the National University of Engineering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza detalladamente el im pacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. This encyc lopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbolic Logic and Computational Logic course at the National University of Engineering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbol ic Logic and Computational Logic course at the National University of Engineering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los princi pios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbolic Logic and Computational Logic course at the National University of Engineering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza det alladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendim iento. This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innova) for the Symbolic Logic and Computational Logic course at the National University of Engi neering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. This encyclopedic engineering monograph documents the complete software development lifecycle of the QuantifiWeb v3.5 platform, created by Team 3 (Modus Innov a) for the Symbolic Logic and Computational

Logic course at the National University of Engineering. En este apartado de la monografía (Sección 1.2 Abstract (English)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer o rden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 1.2 Abstract (English) Explicación analític a y marco metodológico para 1.2 Abstract (English). Garantiza el cumplimiento de los estándares SciELO / IEEE. 1.3 Declaración Formal de Autoría El Equipo 3 declara solemnemente que la presente monografía cumple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los prin cipios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El Equipo 3 declara solemnemente que la presente monografía cumple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y l os principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El Equipo 3 declara solemnemente que la presente monografía cumple a cabali dad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer or den y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El Equipo 3 declara solemnemente que la presente monografía cumple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentación lógica de pr imer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El Equipo 3 declara solemnemente que la presente monografía c umple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentación lógic a de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. El Equipo 3 declara solemnemente que la presente monog rafía cumple a cabalidad con las 4 obligaciones exigidas por la cátedra y el líder general Enri Díaz Cayaca. En este apartado de la monografía (Sección 1.3 Declaración Formal de Autoría), se analiza detalladamente el impacto computacional, la fundamentació n lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 1.3 Declaración Formal de Autoría Explicación analítica y marco metodológico para 1.3 Declaración Formal de Autoría. Garantiza el cumplimiento de los estándares SciELO / IEEE. CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA Y MATEMÁTICA 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Un iversal, ∃ Existencial) En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Dominio de Discurso D. En este apartado de la monografía (Sec ción 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada compone nte ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un con junto denominado Dominio de Discurso D. En este apartado de la monografía (Sección 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y lo s principios de ingeniería de software

aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. En la Lógica de Predicados de Primer Orden, una proposición cuantificada eva lúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Dominio de Discurso D. En este apartado de la monografía (Sección 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detallad amente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento . En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Dominio de Discurso D. En este apartado de la monografía (Sección 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado d e forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Domin io de Discurso D. En este apartado de la monografía (Sección 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeni ería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. En la Lógica de Predicados de Primer Orden, una proposición cuantificada evalúa la validez de una función proposicional P(x) sobre los elementos de un conjunto denominado Dominio de Discurso D. En este apartado de la monografía (Sección 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGE NIERÍA: 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial) Explicación analítica y marco metodológico para 2.1 Teoría Formal de Cuantificadores Lógicos (∀ Universal, ∃ Existencial). Garantiza el cumplimiento de los estándares SciELO / IEEE. 2.2 Restricción de Dominio a Números Enteros (D ⊂ ℤ) Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Discurso D al conjunto de los Números Enteros D ⊂ ℤ = {..., -2, -1, 0, 1, 2, ...}. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D ⊂ ℤ)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Dis curso D al conjunto de los Números Enteros D ⊂ ℤ = {..., -2, -1, 0, 1, 2, ...}. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D ⊂ ℤ)), se analiza detalladamente el impacto computacional, la fundamentación lógica d e primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Discurso D al conjunto de los Números Enteros D ⊂ ℤ = {..., -2, -1, 0, 1, 2, ...}. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D ⊂ ℤ)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto ren dimiento. Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Discurso D al conjunto de los Números Enteros D ⊂ ℤ = {..., -2, -1, 0, 1, 2, ...}. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D ⊂ ℤ)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de f orma modular para garantizar

legibilidad, mantenibilidad y alto rendimiento. Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Dominio de Discurso D al conjunto de los Números Enteros D $\subset \mathbb{Z} = \{... , -2, -1, 0, 1, 2, ...\}$. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D $\subset \mathbb{Z}$)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de softwa re aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Para garantizar el rigor matemático discreto exigido por la informática, el motor de QuantifiWeb restringe el Do minio de Discurso D al conjunto de los Números Enteros D $\subset \mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$. En este apartado de la monografía (Sección 2.2 Restricción de Dominio a Números Enteros (D $\subset \mathbb{Z}$)), se analiza detalladamente el impacto computacional, la fundamentac ión lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 2.2 Restricción de Do minio a Números Enteros (D $\subset \mathbb{Z}$) Explicación analítica y marco metodológico para 2.2 Restricción de Dominio a Números Enteros (D $\subset \mathbb{Z}$). Garantiza el cumplimiento de los estándares SciELO / IEEE. 2.3 Leyes de De Morgan en Cuantificadores Las Leyes de De M organ extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este apartado de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores), se analiza detalla damente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimient o. Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este apartado de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores) , se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilid ad y alto rendimiento. Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este apartado de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legib ilidad, mantenibilidad y alto rendimiento. Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este apartado de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de form a modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este aparta do de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha s ido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Las Leyes de De Morgan extienden la dualidad lógica a las proposiciones cuantificadas mediante equivalencias analíticas: $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$ y $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$. En este apartado de la monografía (Sección 2.3 Leyes de De Morgan en Cuantificadores), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 2.3 Leyes de De Morgan en Cuantificadores Explicación analítica y marco metodológico para 2.3 Leyes de De Morgan en Cu antificadores. Garantiza el cumplimiento de los estándares SciELO / IEEE. 2.4 Operador de Disyunción Fuerte (Δ / $\oplus$) La Disyunción Fuerte o Exclusiva (Δ / $\oplus$) establece que exactamente una de las dos proposiciones debe ser verdadera: $A\ \Delta\ B \equiv (A \vee B) \wedge \neg(\ A \wedge B)$. En

este apartado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3 . Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. La Disyunción Fuerte o Exclusiva (Δ / ⊕) establece que exactamente una de las dos proposiciones debe ser verdadera: A Δ B ≡ (A ∨ B) ∧ ¬(A ∧ B). En este apartado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Ca da componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. La Disyunción Fuerte o Exclusiva (Δ / ⊕) establece que exactamente una de las dos proposiciones debe ser verdadera: A Δ B ≡ (A ∨ B) ∧ ¬(A ∧ B). En este apartado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada c omponente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. La Disyunción Fuerte o Exclusiva (Δ / ⊕) establece que exactamente una de las dos proposiciones debe ser verdadera: A Δ B ≡ (A ∨ B) ∧ ¬(A ∧ B). En e ste apartado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada compo nente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. La Disyunción Fuerte o Exclusiva (Δ / ⊕) establece que exactamente una de las dos proposiciones debe ser verdadera: A Δ B ≡ (A ∨ B) ∧ ¬(A ∧ B). En este apartado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada component e ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. La Disyunción Fuerte o Exclusiva (Δ / ⊕) establece que exactamente una de las dos proposiciones debe ser verdadera: A Δ B ≡ (A ∨ B) ∧ ¬(A ∧ B). En este apar tado de la monografía (Sección 2.4 Operador de Disyunción Fuerte (Δ / ⊕)), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 2.4 Operador de Disyunción Fuerte (Δ / ⊕) Explicación analítica y marco metodológico para 2.4 Operador de Disyunción Fuerte (Δ / ⊕). Gar antiza el cumplimiento de los estándares SciELO / IEEE. CAPÍTULO III: BITÁCORA EVOLUTIVA MINUCIOSA (4 SEMANAS + FASE POSTERIOR) 3.1 Semana 1: Concepción y Pivotaje Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pivotaje), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Durante la primera semana del proyecto, el sublíder Cristia n Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pivotaje), se analiza detalladamente el impacto computacional, la fundamen tación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pivotaje), se analiza detalladamente el impacto computaci onal, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pi-

votaje), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pivotaje), se analiza d etalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rend imiento. Durante la primera semana del proyecto, el sublíder Cristian Llenque Castro unificó las entregas dispersas y tomó la decisión de descartar las tablas de verdad generales. En este apartado de la monografía (Sección 3.1 Semana 1: Concepción y Pivota je), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibi lidad y alto rendimiento. MARCO DE INGENIERÍA: 3.1 Semana 1: Concepción y Pivotaje Explicación analítica y marco metodológico para 3.1 Semana 1: Concepción y Pivotaje. Garantiza el cumplimiento de los estándares SciELO / IEEE. 3.2 Semana 2: Motores TS y el Dilema de Leyes Se desarrolló el motor en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html, consultando al docente para asumir el reto de programar el motor sintáctico. En este apartado de la monografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para gara ntizar legibilidad, mantenibilidad y alto rendimiento. Se desarrolló el motor en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html, consultando al docente para asumir el reto de programar el motor sintáctico. En este apartado de la mon ografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido dise ñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Se desarrolló el motor en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html, consultando al docente para asumir el reto de programar el motor sintáct ico. En este apartado de la monografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Se desarrolló el motor en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html, consultando al docente para asumir el ret o de programar el motor sintáctico. En este apartado de la monografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de so ftware aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Se desarrolló el motor en TypeScript y la primera aplicación autónoma local QuantifiWeb_App.html, consultand o al docente para asumir el reto de programar el motor sintáctico. En este apartado de la monografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Se desarrolló el motor en TypeScript y la primera aplicación autónoma local Q uantifiWeb_App.html, consultando al docente para asumir el reto de programar el motor sintáctico. En este apartado de la monografía (Sección 3.2 Semana 2: Motores TS y el Dilema de Leyes), se analiza detalladamente el impacto computacional, la fundamentaci ón lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 3.2 Semana 2: Motores TS y el Dilema de Leyes Explicación analítica y marco metodológico para 3.2 Semana 2: Motores TS y el Dilema de Leyes. Garantiza el cumplimiento de los estándares SciELO / IEEE. 3.3 Semana

3: Integración en Vue 3 y PR en GitHub Migración del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se analiza detalladamente el i mpacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Migración del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Migr ación del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se analiza detallada mente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Migración del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se analiza deta lladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimi ento. Migración del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto re ndimiento. Migración del código al stack oficial (Vue 3 SFC + Vite + Tailwind CSS v4 + TypeScript) y apertura de la Solicitud de Extracción (PR) en GitHub. En este apartado de la monografía (Sección 3.3 Semana 3: Integración en Vue 3 y PR en GitHub), se an aliza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y al to rendimiento. MARCO DE INGENIERÍA: 3.3 Semana 3: Integración en Vue 3 y PR en GitHub Explicación analítica y marco metodológico para 3.3 Semana 3: Integración en Vue 3 y PR en GitHub. Garantiza el cumplimiento de los estándares SciELO / IEEE. 3.4 Semana 4: Pruebas QA y Bug de Implicación Diagnóstico y corrección del bug de parentización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de Implic ación), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, manten ibilidad y alto rendimiento. Diagnóstico y corrección del bug de parentización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de Implicación), se analiz a detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto r endimiento. Diagnóstico y corrección del bug de parentización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de Implicación), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Diagn óstico y corrección del bug de parentización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de

Implicación), se analiza detalladamente el impacto comput acional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Diagnóstico y correcci ón del bug de parentización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de Implicación), se analiza detalladamente el impacto computacional, la funda mentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Diagnóstico y corrección del bug de par entización en Implicación ~((...)) sustituyendo la regex en lawsEngine.ts por [~A ∨ B]. En este apartado de la monografía (Sección 3.4 Semana 4: Pruebas QA y Bug de Implicación), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. MARCO DE INGENIERÍA: 3.4 Semana 4: Pruebas QA y Bug d e Implicación Explicación analítica y marco metodológico para 3.4 Semana 4: Pruebas QA y Bug de Implicación. Garantiza el cumplimiento de los estándares SciELO / IEEE. 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor Incorporación del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se analiza detalladamente el imp acto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Incorporaci ón del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimiento. Inco rporación del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se analiza detalla damente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto rendimient o. Incorporación del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y alto ren dimiento. Incorporación del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se a naliza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibilidad y a lto rendimiento. Incorporación del compañero Fabrizio Teque Zapata para la programación de Disyunción Fuerte (Δ), simplificador de pasos y resaltador neón. En este apartado de la monografía (Sección 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor), se analiza detalladamente el impacto computacional, la fundamentación lógica de primer orden y los principios de ingeniería de software aplicados por el Equipo 3. Cada componente ha sido diseñado de forma modular para garantizar legibilidad, mantenibili dad y alto rendimiento. MARCO DE INGENIERÍA: 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor Explicación analítica y marco metodológico para 3.5 Fase Posterior: Fabrizio Teque y Mejoras del Profesor. Garantiza el cumplimiento de los estándares SciELO / IEEE. CAPÍTULO IV: MANUAL DE USUARIO DIDÁCTICO PASO A PASO ("MANUAL PARA TODOS") En cumplimiento riguroso con la segunda obligación exigida por el

curso ("documentar explícitamente de cómo usarlo / manual didáctico para todos"), este capítulo presenta la guía oficial de usuario para operar el software QuantifiWeb v3.5. Esta guía permite que tanto alumnos de primeros ciclos como docentes puedan evaluar cuantificadores y demostraciones formales sin conocimiento técnico previo. 4.1 Visión General de la Interfaz y Temática Neón La interfaz de QuantifiWeb v3.5 adopta un estilo visual moderno 'Dark Glassmorphism' con colores neón adaptativos (Azul #3B82F6, Cyan #06B6D4, Verde #10B981 y Púrpura #6366F1). Detalle de uso adicional para 4.1 Visión Genera l de la Interfaz y Temática Neón: Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. En la parte superior se ubica la barra de navegación que permite alternar entre las dos herramientas princi pales: la Pestaña 1 '∀∃ Cuantificadores (D ⊂ ℤ)' y la Pestaña 2 '↔ Distribución & Leyes (Δ)'. Detalle de uso adicional para 4.1 Visión General de la Interfaz y Temática Neón: Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. MANUAL DE USUARIO: 4.1 Visión General de la Interfaz y Temática Neón Instrucciones operativas para 4.1 Visión General de la Interfaz y Temática Neón. Guía didáctica para alumnos y profesores. 4.2 Guía Pas o a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ) Paso 1: Abra la Pestaña 1 '∀∃ Cuantificadores (D ⊂ ℤ)'. En el desplegable 'Tipo de Cuantificador', seleccione '∀ Universal' (si desea probar que la condición se cumple para absolutamente todos los elementos) o '∃ Existencial' (si busca la existencia de al menos un elemento testigo). Detalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza que cualquier usuario pueda seguir e l flujo de operaciones sin cometer errores sintácticos. Paso 2: En la casilla 'Dominio de Discurso D', escriba los números enteros separados por comas (ejemplo: 1, 2, 3, 4) o especifique rangos enteros (ejemplo: 0 < x < 90). El sistema validará automáticam ente que no ingrese números decimales. Si ingresa un decimal como 3.5, aparecerá una alerta de restricción. Detalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza que cualquier usuar io pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 3: En la casilla 'Predicado P(x)', escriba la condición matemática a evaluar (ejemplo: x > 3, x % 2 = 0). Puede usar comparadores (=, ==, !=, <>, >, >=, <, <=), operaciones aritm eticas (+, -, *, /, %, ^) y conectores lógicos (&&, ||, ∧, ∨, y, o). Detalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 4: Presione el botón azul ' Evaluar Cuantificador en D'. La pantalla desplegará la tarjeta de resultado en Verde (si es Verdadero) o en Rojo (si es Falso), mostrando el elemento testigo x_w o el contraejemplo x_k. D etalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 5: Observe la Tabla de Evaluaci ón por Elemento. Cada fila mostrará el valor de x, la sustitución exacta en el predicado y su resultado booleano individual. Detalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza qu e cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 6: En la parte inferior de la tarjeta, revise el recuadro 'Ley de De Morgan Equivalente'. El sistema construirá automáticamente la negación analítica ¬(∀x P(x)) ≡ ∃x ¬P(x), invirtiendo analíticamente la condición interna. Detalle de uso adicional para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin com eter errores sintácticos. MANUAL DE USUARIO: 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ) Instrucciones operativas para 4.2 Guía Paso a Paso para la Evaluación de Cuantificadores Lógicos (D ⊂ ℤ). Guía didáctica para alumnos y profesores. 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ) Paso 1: Diríjase a la Pestaña 2 '↔ Distribución & Leyes (Δ)'. Detalle de uso adicional para 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 2: En la casilla 'Fórmula P', escriba su proposición simbólica formal (ejemplo: (∀x)~[(Px ∨ Mx) → Rz] o usando el conector de Disyunción Fuerte Px Δ Qx). Detalle de uso adicional para 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección

garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sin tácticos. Paso 3: Utilice el Teclado Virtual de Símbolos en pantalla para insertar caracteres especiales (∀, ∃, ~, ∧, ∨, Δ, →, ↔, ≡, P, Q, R, S, M, x, y, z). La tecla Δ está destacada en color cyan. Detalle de uso adicional para 4.3 Guía Paso a Paso para D emostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 4: Presione el botón ' Resolver Paso a Paso (Simplificado)'. Detalle de uso adici onal para 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 5: Examine el desarrollo formal generado. El m otor aplicará secuencialmente las reglas analíticas (Definición de Bicondicional, Implicación, De Morgan, Disyunción Fuerte y Distribución). Detalle de uso adicional para 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Paso 6: Observe el Resaltador Neón Cyan. En cada renglón de la demostración, la subexpresión exacta que sufrió una transformación se encerrará en una tarjeta con resplandor neón <span class='highlight-change'>...</span> para que identifique al instante qué cambió. Detalle de uso adicional para 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. MANUAL DE USUARIO: 4.3 Guía Paso a Paso para Demostraciones Paso a Paso con Disyunción Fuerte (Δ) Instrucciones operativas para 4.3 Guía Paso a Paso para Demostracio nes Paso a Paso con Disyunción Fuerte (Δ). Guía didáctica para alumnos y profesores. 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting) Error 1 'El dominio D opera únicamente sobre números enteros (ℤ)': Ocurre cuando se ingresan números con punto decimal (ej. 3.5). Solución: Reemplace los decimales por valores enteros discretos. Detalle de uso adicional para 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting): Esta sección garantiza que cualquier usu ario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Error 2 'Falta un paréntesis de cierre)': Ocurre cuando los paréntesis de la expresión no están balanceados. Solución: Verifique que cada paréntesis abierto '(' tenga su correspond iente cierre ')'. Detalle de uso adicional para 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Error 3 'División entre cero': Ocurre si la sustitución de un elemento x en el predicado produce un divisor nulo (ej. 5 / (x - 2) cuando x = 2). Solución: Modifique el dominio para excluir los puntos de indeterminación. Detalle de uso adicional para 4.4 Guía de Resolución d e Problemas y Mensajes de Error (Troubleshooting): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. Error 4 'No hay más pasos aplicables': Ocurre en el motor de leyes cuando la fórmula alcan za su forma simplificada irreducible. Detalle de uso adicional para 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos . MANUAL DE USUARIO: 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting) Instrucciones operativas para 4.4 Guía de Resolución de Problemas y Mensajes de Error (Troubleshooting). Guía didáctica para alumnos y profesores. 4.5 Pre guntas Frecuentes (FAQ) P1: £Por qué la Ley de Implicación transforma A → B en ~A ∨ B sin paréntesis innecesarios? R: El algoritmo refactorizado aísla el antecedente A y aplica la negación directa, evitando parentizados redundantes ~((...)) que confundían a los estudiantes. Detalle de uso adicional para 4.5 Preguntas Frecuentes (FAQ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. P2: £Cómo funciona el simplificador de pasos en demostracion es extensas? R: Si la misma regla (ej. Distribución de Cuantificadores) se aplica consecutivamente en el mismo nivel, el motor agrupa las transformaciones en un solo paso elegante. Detalle de uso adicional para 4.5 Preguntas Frecuentes (FAQ): Esta sección garantiza que cualquier usuario pueda seguir el flujo de operaciones sin cometer errores sintácticos. P3: £Qué diferencia a la Disyunción Fuerte (Δ) de la disyunción normal (∨)? R: La disyunción inclusiva ∨ permite que ambas proposiciones sean verdaderas. La disyunción fuerte Δ exige que sea exactamente una u otra, pero NO ambas: A Δ B ≡ (A ∨ B) ∧ ~(A ∧ B). Detalle de uso adicional para 4.5 Preguntas Frecuentes (FAQ): Esta sección garantiza que cualquier usuario pueda seguir

el flujo de operaciones sin come ter errores sintácticos. MANUAL DE USUARIO: 4.5 Preguntas Frecuentes (FAQ) Instrucciones operativas para 4.5 Preguntas Frecuentes (FAQ). Guía didáctica para alumnos y profesores. CAPÍTULO V: EXPLICACIÓN MICROSCÓPICA LÍNEA POR LÍNEA DEL CÓDIGO FUENTE COMPLETO 5. Análisis Microscópico de src/utils/quantifierEngine.ts Archivo Fuente: src/utils/quantifierEngine.ts export class ExpresionInvalidaError extends Error { constructor(mensaje: string) { super(mensaje); this.name = 'ExpresionInvalidaError'; } } export type QuantifierType = 'forall' | 'exists'; const MAX_ELEMENTOS_RANGO = 2000; const COMPARADORES = ['=', '==', '!=', '<>', '>', '>=', '<', '<='] ; export funct ion tokenizeExpression(source: string) { const tokens: any[] = []; let i = 0; while (i < source.length) { const ch = source[i]; if (/\{}s/.test(ch)) { i++; continue; } if (/[0-9.]/.test(ch)) { const match = source.slice(i).match(/^(?: \{}d+(?:\{}.\{}d*)?|\{}.\{}d+)/); if (!match) throw new ExpresionInvalidaError(‘Número inválido cerca de "$source.slice(i)$".‘); $tokens.push(type :' number', value : match[0]); i+ = match[0].length; continue; i++; return tokens; Anlisis Microscpico de la Lnea 1 de src/utils/quantifier Engine.ts :$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$src/utils/laws Engine.ts export interface Proof Step formula : string; rule : string; changed Chunk? : string; ex$
$Rule Id, formula : string) : Proof Step | null switch(rule Id) case' disyuncion_fuerte' : for(let i = 0; i < l; formula.le$
$\vee B)$\{}wedge ({A} $\wedge B)$]‘; $return formula : formula.substring(0, left Obj.start) + replacement + formula.sub$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$src/pages/cuantificadores/Quantificadores Page.vue < template > < div class = "quantificadores-$
$page min-h-screen bg-slate-950 text-slate-100 font-sans p-4" > < header class = "max-$
$w-6xl mx-auto mb-8 flex items-center justify-between" > < h1 class = "text-2xl font-$
$bold" > Evaluador de Cuantificadores L gicos < / span class = "badge-int" > D$\{}subset\{}mathbb{Z}$< /span$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$src/pages/cuantificadores/Leyes Page.vue < template > < div class = "laws - page min - h -$
$screen bg - slate - 950 text - slate - 100 font - sans p - 4" > < main class = "max - w - 6xl mx -$
$auto grid grid - cols - 1 lg : grid - cols - 12 gap - 6" > < section class = "lg : col - span - 5 bg - slate -$
$900 border border - slate - 800 rounded - 2xl p - 6" > < textarea v - model = "formula Input" class =$
$"w - full bg - slate - 950 border border - slate - 800 rounded - xl p - 3" > < / textarea > < / section > < / main > <$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$
$En este fragmento de c ódigo, el motor examina la sintaxis de las expresiones l gicas, verificando que los tokens respeten$

$QuantifiWeb_{App}.htmllt; !DOCTYPEhtmlgt; lt; htmllang = "es"class = "dark"gt; lt; headgt; lt; metacharset =$
$"UTF-8"gt; lt; titlegt; QuantifiWeb|Equipo3---ModusInnovalt; /titlegt; lt; /headgt; lt; bodygt; lt; maingt; lt;$
$"section-quant"gt; ...lt; /divgt; lt; divid = "section-laws"gt; ...lt; /divgt; lt; /maingt; lt; /bodygt; lt; /htmlgt; Anl$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$Enestefragmentodecdigo, elmotorexaminalasintaxisdelasexpresioneslgicas, verificandoquelostokensrespeter$
$MATRIZEXHAUSTIVADE20CASOSDEPRUEBADIDCTICOS(QAamp; TESTING)Lasiguientematr$
$IDEntrada/ExpresinDominioD$\{}subset\{}mathbb{Z}$ResultadoEsperadoySalidaObtenidaTC-$
01\{}forall$x(xgt; 3, x$

»"10. Evolución por Sprints (4 semanas) — Diario unificado de TAREAS

»"10.1. Archivo: TAREAS/ESTADO_EQUIPOS.md

»"# Estado de los Equipos — Contexto para Agentes de Guía

»">**Fecha de revisión:** 21 de Agosto 2026 >**Objetivo:** Dar contexto a cada equipo y su agente de guía sobre el estado real del proyecto.

»"—

»"## [mapa] Distribución del Sílabo (4 Equipos)

»"| # | Equipo | Tema | Carpeta | |—|—|—|—| | 1 | Sinergia | Fundamentos, Conectivos y Tablas de Verdad | ‘src/pages/tablas‘ | | 2 | Los Hijos de Linus | Inferencias Lógicas y Validaciones | ‘src/pages/inferencias‘ | | 3 | Modus Innova | Cuantificadores y Lógica de Predicados | ‘src/pages/cuantificadores‘ | | 4 | Linus | Teoría de Conjuntos y Diagramas | ‘src/pages/conjuntos‘ |

»"—

»"## Estado por Equipo

»"### Equipo 1 — Sinergia (Tablas de Verdad) | Aspecto | Estado | Detalle | |—|—|—| | **Sprint 1 (Propuesta)** | Completado | Propuesta "LogiLearnçon wireframes en Figma | | **Sprint 2 (Motor)** | No iniciado | No hay evaluator de verdad ni generador de tablas | | **Sprint 3 (UI)** | Stub | Solo título y descripción placeholder | | **Sprint 4 (Pruebas)** | No iniciado | — | | **Motor disponible** | ‘src/lib/solver/parser.ts‘ puede reusarse para parsear fórmulas, pero falta ‘evaluar(nodo, asignación): boolean‘ | | **Archivos de referencia** | ‘TAREAS/01_Propuesta/sinergia/propuesta.md‘ |

»"### Equipo 2 — Los Hijos de Linus (Inferencias Lógicas) | Aspecto | Estado | Detalle | |—|—|—| | **Sprint 1 (Propuesta)** | Completado | Propuesta con 2 módulos (resolver + practicar) | | **Sprint 2 (Motor)** | Completado | Parser, solver (MPP), validator, transcription, trazabilidad + tests | | **Sprint 3 (UI)** | Stub | Solo título y descripción placeholder | | **Sprint 4 (Pruebas)** | [!] Parcial | Tests unitarios en ‘src/lib/‘ pasan | | **Motor disponible** | ‘src/lib/solver/‘, ‘src/lib/validator/‘, ‘src/lib/transcription/‘, ‘src/lib/trazabilidad/‘ | | **Archivos de referencia** | ‘TAREAS/01_Propuesta/los-hijos-de-linus/propuesta.md‘, ‘TAREAS/sprint-2/hijos-de-linus/avance.md‘ |

»"### Equipo 3 — Modus Innova (Cuantificadores) | Aspecto | Estado | Detalle | |—|—|—| | **Sprint 1 (Propuesta)** | Completado | Propuesta "QuantifiWebçon cuantificadores + inferencias | | **Sprint 2 (Motor)** | No iniciado | No hay motor para cuantificadores (∀, ∃) | | **Sprint 3 (UI)** | Stub | Solo título y descripción placeholder | | **Sprint 4 (Pruebas)** | No iniciado | — | | **Motor disponible** | Ninguno específico. Nota: la propuesta incluye inferencias que se superponen con equipo 2 | | **Archivos de referencia** | ‘TAREAS/01_Propuesta/cuantificadores/propuesta.md‘ |

»"### Equipo 4 — Linus (Conjuntos) | Aspecto | Estado | Detalle | |—|—|—| | **Sprint

1 (Propuesta)** | Completado | Propuesta con calculadora de conjuntos + Venn | | **Sprint 2 (Motor)** | No iniciado | No hay motor para operaciones con conjuntos | | **Sprint 3 (UI)** | Stub | Solo título y descripción placeholder | | **Sprint 4 (Pruebas)** | No iniciado | — | | **Motor disponible** | Ninguno. Necesita: union, intersección, diferencia, complemento, diagrama de Venn | | **Archivos de referencia** | ‘TAREAS/01_Propuesta/linus/propuesta.md‘ |

»”—

»”## [!] Observaciones Importantes

»”1. **Sobreposición de temas:** La propuesta de Modus Innova (equipo 3) incluye un módulo de Ïnferencias Lógicas y Silogismos"que se superpone con el tema asignado a Los Hijos de Linus (equipo 2). Definir si esta superposición es intencional o si Modus Innova debe enfocarse solo en cuantificadores.

»”2. **Motor compartido:** El motor en ‘src/lib/‘ fue construido principalmente por Los Hijos de Linus (equipo 2). Los otros equipos pueden reutilizar el parser (‘src/lib/solver/parser.ts‘) para parsear fórmulas lógicas, pero necesitarán crear sus propios evaluadores para sus temas específicos.

»”3. **UI stubs:** Las 4 páginas (‘src/pages/*/index.vue‘) son placeholders con solo un título. Ninguna tiene interactividad funcional. La UI debe construirse para el martes 25/08.

»”4. **Deploy:** No hay workflow de GitHub Pages configurado. Necesita ‘base: ’/logica_simbolica_global/’‘ en ‘vite.config.ts‘ y un workflow en ‘.github/workflows/‘.

»”—

»”## Entregables por Sprint (Referencia)

»”| Sprint | Entregable | Fecha | |—|—|—| | Sprint 1 | Propuesta + Bocetos | Completado | | Sprint 2 | Motor Lógico (TypeScript) | [!] Solo equipo 2 | | Sprint 3 | Interfaz Visual (Vue 3) | Pendiente — cierre 21/08 | | Sprint 4 | Pruebas + Deploy | Pendiente — cierre 25/08 (presentación) |

»”—

»”## [carpeta] Stack Tecnológico

»”- **Framework:** Vue 3 + Vite - **Enrutamiento:** Vue Router - **Estilos:** Tailwind CSS v4 - **Lenguaje:** TypeScript - **Tests:** Vitest - **Deploy:** GitHub Pages (pendiente configurar)

»”—

»”## Comandos Útiles

»”“‘bash # Instalar dependencias npm install

»”# Desarrollo local npm run dev

»”# Build para producción npm run build

»”# Tests npm run test

»”# Type check npm run type-check “‘

»”—

»”## [clip] Log de Cambios

»”| Fecha | Quién | Qué hizo | PR | |—|—|—|—| | 21/08/2026 | Enri | Corrigió nombres de equipos (Linus = equipo 4), creó estructura TAREAS/, branding (Button.vue, Card.vue, Badge.vue), deploy GitHub Pages, actualizó docs | — | | 23/08/2026 | Linus (Jordy) | Entregable 2: Motor lógico de conjuntos (operations.ts + tests) | #9 | | 23/08/2026 | Linus (Jordy) | Entregable 3: UI interactiva con diagrama Venn SVG | #10 | | 23/08/2026 | Linus (Jordy) | Entregable 4: Tests comprehensivos (15) + documentación de uso | #11 |

»”>**Instrucción para equipos:** Cuando envíen un PR para cualquier entregable, actualizar esta tabla con la fecha, quién hizo el cambio, qué hizo y el número del PR.

»”10.2. Archivo: TAREAS/DELEGACION_P2.md

»”# Delegación P2 — Informe de Investigación

»”**Delegados:** Sergio y Jordy **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»”—

»"## 1. Propósito (visión compartida) Producir un informe de investigación académico, de **calidad y extensión superiores al estándar**, que sea aprobado por la plataforma de similitud (Turnitin) y documente lo aprendido en el proyecto.

»"## 2. Qué se espera (resultado, no proceso) - Un documento ‘docs/INFORME_INVESTIGACION.md‘ en **formato APA 7, en español**. - Longitud y profundidad **mayores al estándar** (objetivo **18–22 páginas** equivalentes). - Contenido mínimo: - Título, autores (equipos + docente) y resumen (abstract). - Introducción (contexto del curso MATG1001 y resultados de aprendizaje D1/D2/D3 del sílabo). - Marco tecnológico: Vue 3, Vite 6, Tailwind CSS v4, TypeScript, Vitest, Vue Router. - Metodología: 4 equipos de dominio, ramificaciones Git, flujo de Pull Requests, CI (Netlify/GitHub Pages). - Arquitectura y desarrollo: motor de tablas de verdad, inferencias, cuantificadores (∀/∃, De Morgan, resolutor paso a paso), teoría de conjuntos. - **Manual de uso** de la plataforma (pasos por módulo). - Reflexión: **lo aprendido** en el proyecto (por equipo). - Conclusiones y referencias bibliográficas.

»"## 3. Criterios de aceptación (medibles) - [ ] Formato APA 7 coherente (portada, encabezados, citas, referencias). - [ ] ≥ 18 páginas de contenido sustantivo (no relleno). - [ ] Citas y referencias presentes; redacción original (evita plagio detectable). - [ ] Cubre manual de uso y aprendizajes del sílabo (D1/D2/D3). - [ ] Aprobado por Enrique.

»"## 4. Restricciones - Idioma: español. - No incluir código fuente extenso; describir arquitectura y decisiones. - Coherencia con los datos reales del proyecto (ver recursos).

»"## 5. Entregables (archivos) - ‘docs/INFORME_INVESTIGACION.md‘ (nuevo; considerar ‘docs/README.md‘ como índice).

»"## 6. Recursos que Enrique dejó listos - **Datos del sílabo:** UNPRG ů Escuela Prof. de Ing. de Sistemas ů Curso Lógica Simbólica (**MATG1001**, 2026 I, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera**. - **Resultados de aprendizaje del sílabo:** D1 (identifica/aplica lógica proposicional), D2 (interpreta/aplica inferencia lógica), D3 (discute/aplica teoría de conjuntos). - **Stack y estructura reales del repo** (Vue 3 + Vite + Tailwind v4 + TS + Vitest; carpetas ‘src/pages/*‘, ‘src/lib/*‘, ‘src/components/ui/*‘). - Lista de participantes (misma que P1).

»"10.3. Archivo: TAREAS/DELEGACION_P3.md

»"# Delegación P3 — Depurar Repositorio

»"**Delegado:** Alex **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"—

»"## 1. Propósito (visión compartida) Dejar el repositorio limpio, organizado y listo para publicación: README finalizado, sin archivosLegacy de migración, con ramas ordenadas.

»"## 2. Qué se espera (resultado, no proceso) - ‘README.md‘ actualizado al estado **"finalizado"** (proyecto completado al 100 % del sílabo). - Eliminar carpetas/archivos **innecesarios**: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ (son legacy de la migración Next.js → Vue). - Limpiar **ramas locales obsoletas**: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘. - ‘.gitignore‘ actualizado para ignorar los legacy.

»">**Nota:** Enrique retiene la parte de **cerrar los PRs #8, #14 y #15** (no es tarea de Alex).

»"## 3. Criterios de aceptación (medibles) - [ ] ‘README.md‘ refleja estado final, equipos, cómo correr y deploy. - [ ] Sin ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ en el árbol de trabajo. - [ ] Ramas locales obsoletas eliminadas. - [ ] ‘.gitignore‘ incluye los patrones legacy. - [ ] ‘npm run build‘ pasa; **175 tests** siguen pasando. - [ ] Aprobación de Enrique.

»"## 4. Restricciones - No borrar ‘src/‘, ‘docs/‘, ‘TAREAS/‘, ‘public/‘, ni configuraciones de deploy (Netlify/GitHub Pages). - No tocar el trabajo ya integrado en ‘feat/integrate-sinergia-modus‘. - Mantener ‘.agents/‘ (config de herramienta) salvo indicación contraria de Enrique.

»"## 5. Entregables (archivos) - ‘README.md‘ (reescritura a estado final) - Eliminación de: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ - Limpieza de ramas locales: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘ - ‘.gitignore‘ (actualización)

»"## 6. Recursos que Enrique dejó listos - Confirmación de qué PRs cerrar (Enrique cierra #8, #14, #15; se mantiene #16). - Lista de carpetas legacy a borrar (arriba). - Rama de trabajo: ‘feat/integrate-sinergia-modus‘.

»"10.4. Archivo: TAREAS/DELEGACION_P1.md

»"# Delegación P1 — Landing Page (Home + Navbar + Contributors)

»"**Delegado:** Morocho **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"—

»"## 1. Propósito (visión compartida) Dar a conocer el valor de la plataforma para poder publicarla: una landing que comunique **50 % el valor de la plataforma** (qué es, por qué importa, para quién) y **50 % "nosotros"** (los equipos, las personas, el docente y la universidad), todo coherente con el branding LogiLearn.

»"## 2. Qué se espera (resultado, no proceso) - Una ‘Home.vue‘ que funcione como landing profesional y publicable. - Agregar el link **Ïnicio"** al navbar (‘AppNavBar.vue‘) apuntando a ‘/‘. - Crear **‘CONTRIBUTORS.md‘** en la raíz con la lista completa de participantes.

»"## 3. Criterios de aceptación (medibles) - [ ] ‘Home.vue‘ navegable, legible y alineado al branding. - [ ] Secciones presentes: hero de valor + "Sobre Nosotros"(universidad, curso, docente, equipos, participantes). - [ ] Navbar incluye Ïnicio". - [ ] ‘CONTRIBUTORS.md‘ existe y lista a los 4 equipos (incl. Mauricio en Hijos de Linus y Julio en Modus Innova). - [ ] ‘npm run build‘ pasa sin errores. - [ ] Aprobación visual de Enrique.

»"## 4. Restricciones - Branding LogiLearn: navy ‘#0F2D8D‘ (navbar), azul-600 ‘#2563EB‘ (acciones), tarjetas blancas ‘border-neutral-200‘ ‘rounded-xl‘ ‘shadow-sm‘. - No romper el build ni los **175 tests** existentes. - No usar Ïniciar sesión/ Çerrar sesión.en el navbar.

»"## 5. Entregables (archivos) - ‘src/pages/Home.vue‘ (reescritura) - ‘src/components/layout/AppNavBar.vu (agregar Ïnicio") - ‘CONTRIBUTORS.md‘ (nuevo)

»"## 6. Recursos que Enrique dejó listos - **Branding kit:** navy ‘#0F2D8D‘, azul-600, estilo rounded-xl/shadow-sm. - **Datos del sílabo:** Universidad Nacional Pedro Ruiz Gallo ů Escuela Prof. de Ingeniería de Sistemas ů Curso **Lógica Simbólica** (código **MATG1001**, Semestre **2026 I**, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera** (mgonzalesh@unprg.edu.pe). - **Lista de participantes** (ver abajo). - Rama de trabajo: ‘feat/integrate-sinergia-modus‘ (PR #16).

»"### Lista oficial de participantes - **Líder de proyecto:** Enrique (EnriDiazCayaca) - **Equipo Sinergia** (sublíder Alexa): Aldair, Smith, Miguel Velarde, Jesús Núñez - **Equipo Los Hijos de Linus** (sublíder Arom): Centurión, Morocho, Altamirano, Mio, **Mauricio** - **Equipo Modus Innova** (sublíder Cristian): Danuska, Marlon, Guillermo, Noemí, **Julio** - **Equipo Linus** (sublíder Jordy): Nio, Mike, Sergio, Fer, Alejandro - **Docente:** Dr. Mardo Victor Gonzales Herrera

»"10.5. Archivo: TAREAS/informe_los_hijos_de_linus.tex

»"\{}documentclass[11pt,a4paper]{article}

»"% — PAQUETES DE IDIOMA Y CODIFICACIÓN — \{}usepackage[utf8]{inputenc} \{}usepackage[T1]{fontenc} \{}usepackage[spanish,es-tabla]{babel}

»"% — TIPOGRAFÍA Y GEOMETRÍA — \{}usepackage{lmodern} \{}usepackage[top=2.5cm, bottom=2.5cm, left=2.5cm, right=2.5cm]{geometry} \{}usepackage{microtype}

»"% — COLORES Y ESTILO — \{}usepackage[table,xcdraw]{xcolor} \{}definecolor{headerblue}{RGB}{24 43, 73} \{}definecolor{accentblue}{RGB}{30, 80, 150} \{}definecolor{lightbg}{RGB}{248, 249, 251} \{}definecolor{bordergray}{RGB}{220, 224, 230} \{}definecolor{darktext}{RGB}{33, 37, 41} \{}definecolor{codebg}{RGB}{245, 247, 250} \{}definecolor{commentgreen}{RGB}{40, 140, 70} \{}definecolor{keywordpurple}{RGB}{120, 40, 140} \{}definecolor{stringblue}{RGB}{20, 90, 160}

»” % — MATEMÁTICAS Y SÍMBOLOS — \{}usepackage{amsmath} \{}usepackage{amssymb} \{}usepackage{amsfonts}

»” % — IMÁGENES Y RUTAS — \{}usepackage{graphicx} \{}graphicspath{{assets/}{01_Propuesta/hijos-de-linus/assets/}{../../01_Propuesta/hijos-de-linus/assets/}{TAREAS/assets/}{TAREAS/01_Propuesta/hijos-de-linus/assets/}}

»” % — CÓDIGO FUENTE (LISTINGS) — \{}usepackage{listings} \{}lstdefinelanguage{TypeScript}{ keywords={abstract, any, as, boolean, break, case, catch, class, console, const, continue, debugger, declare, default, delete, do, else, enum, export, extends, false, finally, for, from, function, get, if, implements, import, in, instanceof, interface, let, module, new, null, number, of, package, private, protected, public, readonly, return, set, static, string, super, switch, symbol, this, throw, true, try, type, typeof, var, void, while, with, yield}, keywordstyle=\{}color{keywordpurple}\{}bfseries, ndkeywords={class, export, boolean, throw, implements, import, this}, ndkeywordstyle=\{}color{accentblue}\{}bfserie identifierstyle=\{}color{darktext}, sensitive=true, comment=[l]{//}, morecomment=[s]{/*}{*/}, commentstyle=\{}color{commentgreen}\{}itshape, stringstyle=\{}color{stringblue}, morestring=[b]', morestring=[b]"}

»”\{}lstset{ language=TypeScript, backgroundcolor=\{}color{codebg}, basicstyle=\{}ttfamily\{}footnotesi breaklines=true, captionpos=b, frame=single, rulecolor=\{}color{bordergray}, numbers=left, numberstyle=\{}tiny\{}color{gray}, numbersep=8pt, showstringspaces=false, tabsize=2, extendedchars=true, literate={á}{{\{}'a}}1 {é}{{\{}'e}}1 {í}{{\{}'i}}1 {ó}{{\{}'o}}1 {ú}{{\{}'u}}1 {ñ}{{\{}~n}}1 {Á}{{\{}'A}}1 {É}{{\{}'E}}1 {Í}{{\{}'I}}1 {Ó}{{\{}'O}}1 {Ú}{{\{}'U}}1 {Ñ}{{\{}~N}}1 }

»” % — TABLAS Y ELEMENTOS GRÁFICOS — \{}usepackage{tabularx} \{}usepackage{booktabs} \{}usepackage{array} \{}usepackage{enumitem} \{}usepackage{tcolorbox} \{}usepackage{caption}

»” % — ENCABEZADOS Y PIES DE PÁGINA — \{}usepackage{fancyhdr} \{}pagestyle{fancy} \{}fancyhf{}

»”\{}renewcommand{\{}headrulewidth}{0pt} \{}renewcommand{\{}footrulewidth}{0.6pt}

»”\{}fancyhead[L]{\{}fontsize{9}{11}\{}selectfont\{}color{darktext}\{}textbf{Equipo Los Hijos de Linus} — Lógica Simbólica} \{}fancyhead[R]{\{}fontsize{9}{11}\{}selectfont\{}color{darktext}Informe de Proyecto} \{}fancyfoot[C]{\{}fontsize{10}{12}\{}selectfont\{}thepage}

»” % — HIPERVÍNCULOS — \{}usepackage{hyperref} \{}hypersetup{ colorlinks=true, linkcolor=accentblue, filecolor=accentblue, urlcolor=accentblue, pdftitle={Sistema Web Interactivo de Inferencias Lógicas - Informe Técnico}, pdfauthor={Equipo Los Hijos de Linus}, }

»”\{}begin{document}

»” % ========== PORTADA % ========== \{}begin{titlepage} \{}begin{tcolorbox}[colback=headerblue,arc=0mm,boxrule=0pt,height=1.2cm,valign=cente \{}centering \{}textcolor{white}{\{}small\{}textbf{Equipo Los Hijos de Linus — Lógica Simbólica \{}hfill Informe de Proyecto}} \{}end{tcolorbox}

»”\{}vspace{2.2cm} \{}begin{center} {\{}Large\{}textbf{Universidad Nacional Pedro Ruiz Gallo}}\{}\{}[0.3cm] {\{}large\{}textbf{Facultad de Ingeniería Civil, de Sistemas y de Arquitectura}}\{}\{}[0.2cm] {\{}normalsize\{}textbf{Escuela Profesional de Ingeniería de Sistemas}}\{}\{}[1.2cm]

»”\{}textcolor{headerblue}{\{}rule{0.85\{}textwidth}{1.5pt}}\{}\{}[0.8cm]

»”{\{}LARGE\{}textbf{\{}textcolor{headerblue}{Sistema Web Interactivo de Inferencias}}}\{}\{}[0.3cm] {\{}LARGE\{}textbf{\{}textcolor{headerblue}{Lógicas y Trazabilidad Formal}}}\{}\{}[0.5cm] {\{}large\{}textbf{Informe Técnico Final del Proyecto}}\{}\{}[0.8cm]

»”\{}textcolor{headerblue}{\{}rule{0.85\{}textwidth}{1.5pt}}\{}\{}[1.8cm]

»”\{}begin{minipage}{0.85\{}textwidth} \{}centering \{}textbf{\{}large Integrantes — Equipo Los Hijos de Linus (Grupo 2):}\{}\{}[0.4cm] {\{}normalsize Altamirano Astupiñan, Renatto André\{}\{} Centurión Gonzales, Alex\{}\{} Espinoza Orrego, Piero Arom\{}\{} Mio Caballero, Arnold André\{}\{} Morocho Lumbre, Juan David\{}\{}[1.2cm] } \{}textbf{\{}large Docente:}\{}\{} Dr. Mardo Victor Gonzales Herrera \{}end{minipage}

»”\{}vfill {\{}small Lambayeque, 2026} \{}end{center} \{}end{titlepage}

»” % ========== TABLA DE CONTENIDOS % ==========

\{}newpage \{}section*{Tabla de Contenidos} \{}vspace{0.5cm} \{}begin{enumerate}[label=\{}textbf{\{}arab
\{}setlength{\{}itemsep}{0.35cm} \{}item \{}textbf{Introducción y Propuesta del Proyecto (Sprint 1)} \{}item \{}textbf{Arquitectura del Software y Flujo de Trabajo (Git Flow)} \{}item \{}textbf{Capa Lógica: El Motor de Inferencia y Demostración (Sprint 2)} \{}item \{}textbf{Capa Visual: Interfaz Reactiva y Didáctica (Sprint 3 y 3.5)} \{}item \{}textbf{Pruebas de Calidad del Código (Sprint 4)} \{}item \{}textbf{Participación y Aportes del Equipo} \{}item \{}textbf{Trabajo en Equipo} \{}item \{}textbf{Conclusiones} \{}end{enumerate}

»"\{}vspace{1.5cm} \{}begin{tcolorbox}[colback=lightbg,colframe=bordergray,arc=2mm,title=\{}textbf{R
Ejecutivo}] El presente informe documenta la investigación, diseño arquitectónico, implementación algorítmica y aseguramiento de calidad del módulo de \{}textbf{Inferencias Lógicas, Trazabilidad Pedagógica y Diagnóstico con Contraejemplos} desarrollado por el equipo \{}textbf{Los Hijos de Linus} (Grupo 2) en el marco de la asignatura de Lógica Simbólica de la Escuela Profesional de Ingeniería de Sistemas (UNPRG).

»"El software combina un motor de deducción formal automatizado (\{}textit{Forward Chaining}) con un evaluador semántico exhaustivo de 2^N combinaciones de verdad, una interfaz moderna y accesible construida en \{}textbf{Vue 3 (Composition API)} con \{}textbf{TypeScript en modo estricto}, soporte para 11 reglas de deducción natural, teclado simbólico interactivo, exportación en un clic a \{}textbf{\{}LaTeX} / \{}textbf{Markdown}, y un sistema de traducción a lenguaje natural con persistencia local. \{}end{tcolorbox}

»" % ==========
SECCIÓN 1: INTRODUCCIÓN Y PROPUESTA (SPRINT 1) % ==========
\{}newpage \{}section{Introducción y Propuesta del Proyecto (Sprint 1)}

»"El razonamiento deductivo y la deducción formal en lógica proposicional constituyen el núcleo formativo del pensamiento formal en ciencias de la computación. No obstante, el aprendizaje tradicional de las inferencias lógicas presenta barreras de aprendizaje críticas: \{}begin{enumerate}[label=\{}alph*)] \{}item \{}textbf{Dificultad en la estructuración deductiva:} A los estudiantes universitarios les resulta complejo planificar y encadenar reglas de inferencia para transitar desde un conjunto de premisas iniciales hasta la conclusión deseada de forma sistemática. \{}item \{}textbf{Ausencia de retroalimentación inmediata y explicativa:} Cuando un argumento es inválido o incurre en una falacia formal (como la afirmación del

»"10.6. Archivo: TAREAS/01_Propuesta/01_Propuesta.md

»"# TAREA 01: Propuesta y Diseño

»"**Objetivo:** Diseñar la funcionalidad de la herramienta web (SIN programar aún).

»"## Equipos y Temas Asignados - **Equipo 1 (Los hijos de Linus | Arom):** Fundamentos y Conectivos (‘/src/pages/fundamentos‘) - **Equipo 2 (Sinergia | Alexa):** Tablas de Verdad y Leyes (‘/src/pages/tablas‘) - **Equipo 3:** Reglas de Inferencia (‘/src/pages/inferencias‘) - **Equipo 4:** Cuantificadores y Silogismos (‘/src/pages/cuantificadores‘) - **Equipo 5:** Teoría de Conjuntos (‘/src/pages/conjuntos‘) - **Equipo 6:** Operaciones de Conjuntos (‘/src/pages/operaciones_conjuntos‘)

»"—

»"## [tool] Instrucciones de Entrega

»"1. Creen el archivo ‘propuesta.md‘ dentro de la carpeta de su equipo en ‘TAREAS/01_Propuesta/‘ (Ej: ‘TAREAS/01_Propuesta/sinergia/propuesta.md‘). 2. Tomen capturas de pantalla de sus diseños de Figma o Excalidraw, guárdenlos en una carpeta ‘assets‘ al lado de su propuesta, y colóquenlos en su Markdown así: ‘![Boceto](assets/mi-foto.png)‘. 3. Copien y llenen la plantilla de abajo. 4. El Sublíder debe enviar los cambios a GitHub mediante un **Pull Request** antes del Viernes a las 4:00 PM.

»"> **ąEjemplo a Seguir!** >Si tienen dudas de cómo entregar, revisen la carpeta del Equipo 1 (Los hijos de Linus): >‘TAREAS/01_Propuesta/los-hijos-de-linus/‘ >Ellos ya hicieron la entrega perfecta. Usen su archivo ‘propuesta.md‘ y su carpeta ‘assets/‘ como inspiración.

»"—

»">**Plantilla Requerida para ‘propuesta.md‘:** >“‘md ># Título de la Herramienta >**Equipo:** [Nombre] >**Integrantes:** [Lista de Nombres] >**Tema:** [Tema Asignado] ” ### 1. Problema a Resolver >(Ej: .A los alumnos les cuesta aplicar la ley de Morgan"). » ### 2. Funcionalidad Propuesta >(Ej: Ün generador de ejercicios donde el alumno arrastra y suelta símbolos"). ### 3. Boceto Visual (Wireframe) >(Añadan una imagen de un dibujo a mano, Figma, etc).

—

Checklist Final

- [ ] ‘npm run build‘ produce ‘dist/‘ sin errores - [ ] ‘npm run test‘ pasa - [ ] App visible en GitHub Pages - [ ] Todos los módulos navegables desde Home - [ ] README actualizado con link al deploy - [ ] Documento de uso creado

10.7. Archivo: TAREAS/04_Deploy/linus/uso.md

Uso del Módulo: Teoría de Conjuntos y Diagramas de Venn

Cómo funciona

El módulo de Conjuntos del **Equipo Linus** permite al usuario definir un Universo y dos conjuntos (A y B), y luego realizar operaciones matemáticas sobre ellos. Los resultados se muestran en texto y se visualizan en un **Diagrama de Venn interactivo** que se ilumina según la operación seleccionada.

Motor lógico: ‘src/lib/sets/operations.ts‘ **Interfaz visual:** ‘src/pages/conjuntos/index.vue‘

—

Ejemplos

Ejemplo 1: Unión de dos conjuntos - **Entrada:** A = {1, 2, 3} y B = {3, 4, 5} - **Operación:** Unión (A ∪ B) - **Salida esperada:** {1, 2, 3, 4, 5}

Ejemplo 2: Intersección de dos conjuntos - **Entrada:** A = {1, 2, 3, 4} y B = {3, 4, 5, 6} - **Operación:** Intersección (A ∩ B) - **Salida esperada:** {3, 4}

Ejemplo 3: Diferencia A B - **Entrada:** A = {1, 2, 3} y B = {2, 3} - **Operación:** Diferencia (A B) - **Salida esperada:** {1}

Ejemplo 4: Conjunto Potencia - **Entrada:** A = {1, 2} - **Operación:** Potencia P(A) - **Salida esperada:** { {}, {1}, {2}, {1, 2} } → 4 subconjuntos

Ejemplo 5: Verificación de propiedades - **Entrada:** A = {1, 2} y B = {1, 2, 3, 4} - **Resultado:** A ⊆ B → Sí | B ⊆ A → No | Disjuntos → No

—

Cómo probar en la app

1. Ejecutar ‘npm run dev‘ en la raíz del proyecto. 2. Abrir ‘http://localhost:5173/conjuntos‘ en el navegador. 3. Escribir los elementos separados por comas en los campos de texto. 4. Hacer clic en los botones de operaciones para ver el resultado y el diagrama.

Cómo correr los tests

“‘bash npx vitest run src/lib/sets/operations.test.ts “‘

—

Limitaciones

- Solo soporta conjuntos finitos ingresados manualmente (separados por comas). - Los elementos se tratan como texto (strings), no como números con orden matemático. - El Diagrama de Venn solo soporta 2 conjuntos (A y B), no 3. - El Conjunto Potencia puede ser lento con conjuntos de más de 15 elementos (2ź = 32,768 subconjuntos).

10.8. Archivo: TAREAS/04_Deploy/hijos-de-linus/errores_y_soluciones.md

[tool] Registro de Errores, Diagnósticos y Soluciones — Los Hijos de Linus

Equipo: Hijos de Linus (Arom Espinoza, Juan Morocho, Arnold Mio, Alex Centurión, Renatto Altamirano) **Módulo:** Demostración Formal de Reglas de Inferencia y Trazabilidad Pedagógica (‘/inferencias‘) **Fecha de Actualización:** 24 de Agosto de 2026 **Estado General:** **100 % de Errores Documentados y Solucionados (111/111 Tests Pasando)**

—

Resumen Ejecutivo

Durante las fases de integración, pruebas de estrés y validación con usuarios reales, se identificaron y resolvieron **11 incidencias críticas, lógicas y de experiencia de usuario**. A continuación se detalla cada problema, su causa raíz, la solución implementada y su estado de verificación.

—

1. Contradicción Semántica: Falso Positivo de Ïnferencia Inválida.en Argumentos Válidos con Prueba Indirecta

* **Síntoma:** Al evaluar argumentos válidos que requieren demostración indirecta (Reducción al Absurdo / Prueba Condicional), el sistema mostraba el banner rojo *Ïnferencia inválida"*, mientras que el diagnóstico interno contradecía diciendo *.El argumento es semánticamente válido pero no pudo derivarse por encadenamiento directo"*. * **Causa Raíz:** El motor clasificaba binariamente todo lo que no pudiera derivar hacia adelante (*forward-chaining*) como ïnválido", sin verificar la existencia real de contraejemplos. * **Solución Implementada:** - Se implementó un solucionador semántico exhaustivo por tablas de verdad (2^N combinaciones en ‘encontrarContraejemplo‘). - Se separó el estado global del argumento en **3 estados claros e independientes**: 1. ‘valida‘ (*Demostrada con reglas básicas* - Banner Verde). 2. ‘no_demostrable_directa‘ (*Válida semánticamente, pero requiere método indirecto/RAA* - Banner Índigo). 3. ‘invalida‘ (*Refutada formalmente por contraejemplo matemático* - Banner Rojo). * **Estado:** **SOLUCIONADO** (‘IndicadorResultado.vue‘, ‘solver.ts‘, ‘types/inferencias.ts‘).

—

2. Ausencia de Contraejemplos Formales en Argumentos Inválidos y Falacias

* **Síntoma:** Si un argumento era inválido (ej. Falacia del Dilema Inverso ‘(P → Q) ∧ (R → S), Q ∨ S P ∨ R‘), el sistema informaba que no era válido pero no explicaba por qué ni entregaba una prueba matemática. * **Causa Raíz:** No existía un evaluador de modelos de satisfacción que extrajera una asignación de verdad concreta que hiciera verdaderas las premisas y falsa la conclusión. * **Solución Implementada:** - Se diseñó la función ‘encontrarContraejemplo(premisas, conclusion)‘ que evalúa exhaustivamente el árbol de verdad. - En ‘PanelTrazabilidad.vue‘ se agregó una tarjeta de diagnóstico formal destacada en rojo con el contraejemplo exacto (ej. $P = F, Q = V, R = F, S = F$) y la verificación paso a paso de cada premisa y conclusión. * **Estado:** **SOLUCIONADO** (‘solver.ts‘, ‘PanelTrazabilidad.vue‘).

—

3. Soporte y Semántica de la Disyunción Fuerte / Exclusiva (‘‘ / XOR)

* **Síntoma:** El sistema no soportaba el conectivo de disyunción fuerte ($\triangle$), impidiendo resolver silogismos exclusivos como $P \triangle Q, P \vdash \neg Q$. * **Causa Raíz:** El parser y las reglas de inferencia solo contemplaban la disyunción inclusiva ($\vee$). * **Solución Implementada:** - Se integró el operador ‘O_EXCLUSIVA‘ y la regla formal ‘SILOGISMO_DISYUNTIVO_EXCLUSIVO‘ ($A \triangle B, A \vdash \neg B$ y $A \triangle B, \neg A \vdash B$). - Se normalizaron símbolos de entrada: ‘‘, ‘‘, ‘‘, ‘⊕‘, ‘‘. - Se agregaron las traducciones pedagógicas en ‘translations.ts‘ y ‘descriptionGenerator.ts‘. * **Estado:** **SOLUCIONADO** (‘solver.ts‘, ‘FormularioInferencia.vue‘, ‘translations.ts‘).

—

4. Desbordamiento y Botones Huérfanos en Teclado Móvil

* **Síntoma:** En pantallas móviles estrechas (360px–390px), el teclado virtual se desbordaba en 3 filas asimétricas, dejando un único botón huérfano en la segunda y tercera línea. * **Causa Raíz:** Uso de contenedores flexibles con ‘flex-wrap‘ y anchos variables por botón. * **Solución Implementada:** - Se estructuró el teclado en un **CSS Grid fijo estricto de exactamente 2 filas**: - **Fila 1 (Grid 8 columnas):** ‘¬‘, ‘∧‘, ‘∨‘, ‘‘, ‘→‘, ‘↔‘, ‘(‘, ‘)‘. Es matemáticamente imposible que un botón desborde. - **Fila 2 (Flex horizontal):** ‘P, Q, R, S | A, B, C, D‘ a la izquierda y ‘ Salto‘ a la derecha. * **Estado:** **SOLUCIONADO** (‘FormularioInferencia.vue‘).

—

5. [teclado] Apertura Involuntaria del Teclado Nativo Móvil (Soft Keyboard)

* **Síntoma:** Al tocar cualquier botón del teclado virtual en Android o iOS, se forzaba el foco en el ‘<textarea>‘, abriendo el teclado nativo del sistema operativo (Gboard/Samsung Keyboard)

y tapando la interfaz. * **Causa Raíz:** Llamada forzada a ‘inputEl.focus()‘ tras cada inserción de símbolo. * **Solución Implementada:** - Se añadió detección táctil (‘’ontouchstart’ in window || navigator.maxTouchPoints >0‘). - En dispositivos móviles se actualiza la posición del cursor mediante ‘setSelectionRange()‘ **sin invocar ‘.focus()‘**, manteniendo el teclado del teléfono oculto mientras se usan los botones virtuales. * **Estado:** **SOLUCIONADO** (‘FormularioInferencia.vue‘).

—

6. Espaciado Automático Inconsistente en Paréntesis y Variables

* **Síntoma:** Los paréntesis insertaban espacios automáticos indeseados (ej. ‘(P → Q)‘), ensuciando las fórmulas. * **Causa Raíz:** Una regla de espaciado genérica trataba a los paréntesis como conectores binarios con espacios antes y después. * **Solución Implementada:** - Se definió una lista estricta ‘CONECTIVOS_CON_ESPACIO = [’∧’, ’∨’, ’’, ’→’, ’↔’]‘. - Paréntesis ‘(‘, ‘)‘, negación ‘¬‘, variables (‘P‘, ‘Q‘, etc.) y salto de línea no insertan ningún espacio adicional. - Al escribir ‘(‘, ‘P‘, ‘→‘, ‘Q‘, ‘)‘ se genera limpiamente ‘(P → Q)‘. * **Estado:** **SOLUCIONADO** (‘FormularioInferencia.vue‘).

—

7. [!] Desplazamiento Visual por Banner de Linter en Tiempo Real

* **Síntoma:** Un cuadro amarillo de .ªviso de sintaxis.ªparecía dinámicamente mientras el usuario escribía fórmulas incompletas, empujando los botones hacia abajo y restando espacio. * **Causa Raíz:** Renderizado condicional reactivo en tiempo real (‘v-if=.ªdvertenciasSintaxis.length >0«). * **Solución Implementada:** - Se eliminó el cuadro flotante intrusivo. - La validación de sintaxis ahora se realiza de forma limpia y consolidada al presionar **"Demostrar Inferencia"**, reportándose en el panel de resultados sin alterar el layout durante la edición. * **Estado:** **SOLUCIONADO** (‘FormularioInferencia.vue‘).

»—

»## 8. Escala y Redimensionamiento Dispar de Glifos Unicode en Android

»* **Síntoma:** Los símbolos lógicos (‘∧‘, ‘∨‘, ‘‘, ‘→‘, ‘↔‘) se veían diminutos o cambiaban de tamaño al añadir o quitar espacios. * **Causa Raíz:** Inconsistencia de fallback de fuentes monoespaciadas en navegadores móviles (Roboto Mono carece de glifos matemáticos y caía en fuentes con diferente baseline y escala). * **Solución Implementada:** - Estandarización a ‘text-base‘ (16px) con ‘leading-relaxed‘ y ‘font-mono‘ en los inputs, garantizando que todos los glifos Unicode mantengan escala uniforme y evitando el zoom automático en iOS/Android. - Botonera con tamaño fijo y ‘leading-none‘. * **Estado:** **SOLUCIONADO** (‘FormularioInferencia.vue‘).

»—

»## 9. Inconsistencia de Sintaxis LaTeX en la Exportación a Markdown

»* **Síntoma:** Al exportar la demostración a formato ‘.md‘, las premisas se exportaban con símbolos Unicode limpios (‘P → R‘), pero los pasos deducidos y la conclusión incluían comandos LaTeX crudos como ‘$(P \to C)$‘ o ‘\{}therefore‘. * **Causa Raíz:** La función de exportación a Markdown reutilizaba la función de notación matemática diseñada originalmente para LaTeX. * **Solución Implementada:** - Se creó la función ‘aNotacionMarkdown‘ que normaliza todos los operadores internos a símbolos Unicode universales limpios (‘→‘, ‘∧‘, ‘∨‘

»10.9. Archivo: TAREAS/04_Deploy/hijos-de-linus/reporte-errores-inferencias.md

»# Reporte de Errores — Módulo de Inferencias

»| | | |—|—| | **Proyecto** | Lógica Simbólica — Equipo "Hijos de Linus"| | **Página** | https://logicasimbolicaglobal.netlify.app/inferencias | | **Reportado por** | Mio | | **Fecha** | 23/08/2026 | | **Componente afectado** | Motor lógico / Solver (búsqueda de la demostración) |

»—

»## Resumen ejecutivo

»| # | Error | Tipo | £Afecta la validez de la conclusión? | |—|——-|——|:—:| | 1 | Paso redundante al demostrar ‘R‘ desde ‘P→Q, Q→R, P‘ | Camino de demostración no óptimo | No | | 2 | Pasos redundantes/duplicados al demostrar ‘Q‘ desde 4 premisas | Camino de demostración no óptimo | No |

»**Patrón detectado:** el motor lógico no siempre elige el camino más corto hacia la conclusión pedida, y en algunos casos deriva dos veces el mismo resultado (una vez normal y otra con los operandos conmutados).

»—

»## Error 1 — Paso redundante en la demostración

»**Premisas:**

»“‘ 1. P → Q 2. Q → R 3. P “‘

»**Se pide demostrar:** ‘R‘

»<table><tr><th> Camino óptimo esperado (2 pasos)</th><th>[!] Camino actual del sistema</th></tr><tr valign="top» <td>

»“‘ 4. P → R (SH, de 1 y 2) 5. R (MPP, de 3 y 4) “‘

»</td><td>

»“‘ 4. Q (MPP, de 1 y 3) paso de más 5. P → R (SH, de 1 y 2) 6. R (MPP, de 3 y 5) “‘

»</td></tr></table>

»**Impacto:** la demostración sigue siendo válida, pero el paso 4 (‘Q‘) no es necesario para llegar a ‘R‘ por el camino más corto. Afecta la claridad del procedimiento mostrado al usuario.

»—

»## Error 2 — Pasos redundantes y duplicados (caso con 4 premisas)

»**Premisas:**

»“‘ 1. P → Q 2. R → S 3. P ∨ R 4. ¬S “‘

»**Se pide demostrar:** ‘Q‘

»<table><tr><th> Camino óptimo esperado (2 pasos)</th><th>[!] Camino actual del sistema (5 pasos)</th></tr><tr valign="top» <td>

»“‘ 5. Q ∨ S (DC, de 1, 2 y 3) 6. Q (MTP, de 5 y 4) “‘

»</td><td>

»“‘ 5. ¬R (MTT, de 2 y 4) 6. P (MTP, de 3 y 5) 7. Q ∨ S (DC, de 1, 2 y 3) 8. S ∨ Q (DC de nuevo) duplicado de la línea 7 9. Q (MPP, de 1 y 6) “‘

»</td></tr></table>

»**Problemas identificados:** - El paso ‘S ∨ Q‘ es **redundante**: es la misma disyunción que ‘Q ∨ S‘, solo con los operandos invertidos. No aporta información nueva. - El camino completo (hallar ‘¬R‘, luego ‘P‘, y recién después el dilema constructivo) es mucho más largo que ir directo por ‘DC‘ + ‘MTP‘.

»>**Observación adicional a verificar:** en la captura de pantalla original, los números de los pasos mostrados en pantalla no coinciden exactamente con los números de "Línea"que el propio sistema usa para referenciarse a sí mismo en pasos posteriores (desfase de 1). No se pudo confirmar la causa con el código revisado — ver sección técnica más abajo.

»—

»## Sugerencia técnica para resolver el problema

»### Sobre el código revisado

»Se revisaron ‘solver.ts‘, ‘parser.ts‘, ‘types.ts‘, ‘index.ts‘, ‘descriptionGenerator.ts‘, ‘astRenderer.ts‘, ‘translations.ts‘ y los tests de integración. Un punto importante:

»>**‘solver.ts‘ actual es un stub/esqueleto**, no el motor que genera las demostraciones vistas en producción. Su función ‘demostrarConclusion()‘ solo tiene un caso *hardcodeado* para 2 premisas resolviendo Modus Ponendo Ponens (es la base para que pasen los tests iniciales). No contiene todavía la lógica de búsqueda que aplica Silogismo Hipotético, Dilema Constructivo, MTT, MTP, etc., ni la que decide en qué orden probar las reglas — que es justo donde vive el bug reportado. Es decir: **el bug no está en los archivos que se compartieron**; hace falta el módulo real de búsqueda/generación del árbol de inferencia (probablemente una versión más avanzada de ‘demostrarConclusion‘ o un archivo aparte tipo ‘buscarDemostracion.ts‘) para localizar la causa exacta.

»Dicho esto, la arquitectura ya definida en ‘types.ts‘ (‘PasoDemostracion‘, ‘ResultadoDemostracion‘, ‘ReglaLogica‘) es perfectamente compatible con la solución que se propone abajo — no hace

falta rediseñar los tipos, solo completar/reescribir el algoritmo de búsqueda dentro de esa misma estructura.

»### La causa raíz más probable

»Por el patrón de los dos errores (pasos de más + resultados duplicados y conmutados), lo más probable es que el algoritmo actual:

»1. Aplica las reglas en **orden de prioridad fijo** (o recorrido tipo DFS) sin comparar cuántos pasos toma cada camino, en vez de buscar explícitamente el camino más corto. 2. No tiene una **forma canónica** para comparar expresiones: trata ‘Q ∨ S‘ y ‘S ∨ Q‘ como hechos distintos porque compara el AST tal cual (ver ‘sonNodosIguales‘ en ‘solver.ts‘, que compara ‘izquierdo‘/‘derecho‘ en orden estricto), en lugar de reconocer que son la misma proposición.

»### Solución propuesta

»**1. Búsqueda por niveles (BFS) en vez de aplicación en orden fijo**

»Reemplazar la generación de pasos por una búsqueda en anchura sobre el .espacio de hechos derivables":

»- **Nivel 0:** las premisas originales. - **Nivel k:** todos los hechos nuevos que se pueden derivar combinando hechos de niveles ‘0..k-1‘ con una sola regla de inferencia. - En cuanto la conclusión aparece en algún nivel, se detiene la búsqueda y se reconstruye el camino más corto hacia atrás (usando referencias tipo "de qué hechos salió cada uno", que ya existe como concepto en ‘lineasInvolucradas‘).

»Esto garantiza automáticamente el camino más corto: como en el Error 1, tanto ‘Q‘ (por MPP) como ‘P→R‘ (por SH) están en el mismo nivel (nivel 1), pero solo ‘P→R‘ está en el camino que realmente lleva a ‘R‘ en 2 pasos, así que el algoritmo nunca necesita expandir el hecho ‘Q‘ si no es necesario para el objetivo.

»**2. Forma canónica para detectar hechos ya conocidos (evita duplicados)**

»Agregar una función ‘normalizarNodo(nodo: NodoExpresion): string‘ que:

»- Para operadores conmutativos (‘Y‘, ‘O‘, ‘O_EXCLUSIVA‘, ‘SI_Y_SOLO_SI‘, ‘NI‘, ‘INCOMPATIBLE‘), ordena ‘izquierdo‘ y ‘derecho‘ de forma determinística (ej. alfabéticamente por su propia forma canónica) antes de serializar. - Para operadores no conmutativos (‘ENTONCES‘) y ‘NO‘, serializa respetando el orden original. - Produce un string único por proposición lógicamente equivalente en su forma (no hace falta resolver equivalencias complejas tipo De Morgan, solo conmutatividad).

»Con esa función, se mantiene un ‘Set<string>‘ (o ‘Map<string, PasoDemostracion>‘) de "hechos ya derivadosüsando la clave canónica. Antes de agregar un nuevo paso al árbol de búsqueda, se verifica si su forma canónica ya existe en el set — si existe, se descarta ese paso por completo (no se muestra ni se cuenta como parte de la demostración). Esto elimina directamente el caso ‘S ∨ Q‘ duplicando ‘Q ∨ S‘.

»**3. Dónde ubicarlo en el código actual**

»- La lógica de comparación estructural que ya existe (‘sonNodosIguales‘ en ‘solver.ts‘) se puede mantener para comparaciones exactas, pero conviene agregar ‘normalizarNodo‘ como función nueva (en ‘solver.ts‘ o un archivo utilitario aparte) específicamente para la deduplicación de hechos derivados. - La función ‘demostrarConclusion‘ necesita pasar de ser un caso hardcodeado a implementar el algoritmo BFS descrito arriba, devolviendo igual un ‘ResultadoDemostracion‘ con ‘pasos: PasoDemostracion[]‘ — la interfaz pública no cambia, así que ‘index.ts‘, ‘descriptionGenerator.ts‘ y ‘astRenderer.ts‘ no deberían necesitar modificaciones.

»**4. Sobre el posible desfase de numeración (Error 2, nota adicional)**

»En el código revisado, ‘index.ts‘ calcula la línea de cada paso como ‘premisas.length + indice + 1‘, lo cual es consistente y no debería producir el desfase visto en la captura. Esto refuerza que el desfase observado probablemente viene del mismo módulo de búsqueda no incluido en los archivos compartidos — valdría la pena revisarlo ahí una vez que se ubique ese archivo.

»—

»## Casos de prueba sugeridos

»Para agregar a ‘solver.test.ts‘ (o a ‘integracion_motor.test.ts‘ si se prueba el flujo completo) una vez implementada la corrección:

»“‘typescript descri

»10.10. Archivo: TAREAS/04_Deploy/hijos-de-linus/uso.md

»# Uso del Módulo: Demostrador de Inferencias Lógicas

»**Equipo:** Hijos de Linus (Arom Espinoza, Juan Morocho, Arnold Mio, Alex Centurión, Renatto Altamirano) **Módulo:** Demostración Formal de Reglas de Inferencia, Trazabilidad Pedagógica y Diagnóstico con Contraejemplos (‘/inferencias‘) **Fecha:** 23 de Agosto de 2026

»—

»## [libro] Cómo funciona

»El módulo permite ingresar un conjunto de premisas lógicas y una conclusión objetivo en notación simbólica formal o estándar para:

»1. **Validar y Demostrar Deducciones (Forward Chaining + SAT Evaluator):** - Evalúa sistemáticamente las 11 reglas de deducción natural: Modus Ponendo Ponens, Modus Tollendo Tollens, Silogismo Disyuntivo, Silogismo Hipotético, Simplificación, Conjunción, Dilema Constructivo, Bicondicionales y **Silogismo Disyuntivo Exclusivo con Disyunción Fuerte (‘‘)**. - Integra un evaluador semántico de combinaciones de verdad (2^N) para validar argumentos y detectar modelos de satisfacción.

»2. **Diferenciación Rigurosa de 3 Estados de Validez:** - [verde] **Inferencia válida (Demostrada):** Se completó la deducción mediante reglas formales directas. - **Inferencia válida (Método indirecto requerido):** El argumento es semánticamente válido (0 contraejemplos), pero requiere derivación indirecta (Reducción al Absurdo o Prueba Condicional). - [rojo] **Inferencia inválida (Refutada por contraejemplo):** Se encontró una asignación de verdad concreta que hace verdaderas las premisas y falsa la conclusión.

»3. **Trazabilidad Pedagógica y Desglose Particionado:** - Genera una enumeración formal $(1), (2), \cdots \therefore (N)$ con badges de reglas. - Cada paso cuenta con un acordeón colapsable que explica: - **Premisas Base:** Con número de línea y rol lógico (ej. *Condicional base*, *Antecedente afirmado*). - **Regla Aplicada y Justificación:** Explicación semántica y causal. - **Deducción Resultante:** Proposición obtenida formalmente.

»4. **Diagnóstico con Contraejemplos Matemáticos:** - Si el argumento es inválido o incurre en una falacia formal (Afirmación del Consecuente, Negación del Antecedente, Dilema Inverso), el sistema despliega el **contraejemplo exacto** (ej. $P = F, Q = V, R = F, S = F$) con la comprobación de verdad de cada premisa y la conclusión.

»5. **Herramientas Académicas (Exportación & Historial):** - **Exportación en 1 Clic:** Botones rápidos para copiar la demostración formal en **LaTeX (‘\{}begin{aligned} ... \{}end{aligned}‘)** o **Markdown**. - **Historial Persistente (‘localStorage‘):** Almacenamiento local de las últimas demostraciones con restauración inmediata.

»6. **Traductor a Lenguaje Natural:** - Pestaña complementaria para asociar enunciados cotidianos a las variables proposicionales ($P, Q, R, \dots$) y generar el argumento continuo en español.

»—

»## Ejemplos

»### Ejemplo 1: Cadena Multi-Paso (Transitividad y Conjunción) * **Premisas:** “‘text (P → Q) ∧ (Q → R) P “‘ * **Conclusión:** ‘R‘ * **Resultado:** ‘Inferencia válida (Demostrada)‘ * **Demostración generada:** - *(3)* ‘P → Q‘ — Simplificación de (1) - *(4)* ‘Q → R‘ — Simplificación de (1) - *(5)* ‘P → R‘ — Silogismo Hipotético de (3, 4) - *(6)* ‘R‘ — Modus Ponendo Ponens de (5, 2)

»### Ejemplo 2: Disyunción Fuerte / Exclusiva (‘‘) * **Premisas:** “‘text P Q P “‘ * **Conclusión:** ‘¬Q‘ * **Resultado:** ‘Inferencia válida (Demostrada)‘ * **Demostración generada:** - *(3)* ‘¬Q‘ — Silogismo Disyuntivo Exclusivo de (1, 2)

»### Ejemplo 3: Dilema Constructivo Clásico * **Premisas:** “‘text P → Q R → S P ∨ R “‘ * **Conclusión:** ‘Q ∨ S‘ * **Resultado:** ‘Inferencia válida (Demostrada)‘ * **Demostración generada:** - *(4)* ‘Q ∨ S‘ — Dilema Constructivo de (1, 2, 3)

»### Ejemplo 4: Falacia del Dilema Inverso (Refutación por Contraejemplo) * **Premisas:** “‘text (P → Q) ∧ (R → S) Q ∨ S “‘ * **Conclusión:** ‘P ∨ R‘ * **Resultado:** ‘Inferencia inválida (Refutada por contraejemplo)‘ * **Contraejemplo matemático:** $P = F,\ Q = V,\ R = F,\ S = F$ - Premisa 1: $(F \to V) \wedge (F \to F) = V$ - Premisa 2: $V \vee F = V$ - Conclusión: $F \vee F = F$

»—

»## Operadores Soportados

»| Operador | Símbolo | Sintaxis y Normalización Aceptada | |—|—|—| | Conjunción (Y) | ∧ | ‘∧‘, ‘^‘, ‘&&‘, ‘Y‘, ‘\{}land‘ | | Disyunción Inclusiva (O) | ∨ | ‘∨‘, ‘\{}|\{}|‘, ‘O‘, ‘v‘, ‘\{}lor‘ | | Disyunción Fuerte / Exclusiva | △ | ‘‘, ‘‘, ‘‘, ‘⊕‘, ‘‘, ‘O_EXCLUSIVA‘ | | Negación (NO) | ¬ | ‘¬‘, ‘~‘, ‘¡, ‘NO‘, ‘\{}neg‘ | | Condicional (ENTONCES) | → | ‘→‘, ‘->‘, ‘=>‘, ‘⇒‘, ‘ENTONCES‘, ‘\{}to‘ | | Bicondicional (SI Y SOLO SI) | ↔ | ‘↔‘, ‘<->‘, ‘<=>‘, ‘ ⟺ ‘, ‘SI_Y_SOLO_SI‘, ‘\{}leftrightarrow‘ |

»—

»## [!] Limitaciones y Alcance - **Dominio:** Lógica proposicional de primer orden con conectivos clásicos y disyunción exclusiva. Los cuantificadores de predicados ($\forall, \exists$) corresponden al módulo de *Modus Innova*. - **Profundidad:** El demostrador ejecuta hasta 12 iteraciones de *Forward Chaining* exhaustivo. Si un argumento es válido pero requiere suposiciones auxiliares, el sistema lo identifica como ‘no_demostrable_directa‘ en lugar de clasificarlo erróneamente como inválido.

»—

»## Verificación de Calidad y Tests - **Tests Automatizados:** **111 tests pasando al 100 %** en Vitest (‘npm test‘). - **Verificación Estricta de Tipos:** TypeScript estricto con ‘vue-tsc –noEmit‘ (**0 errores**). - **Build de Producción:** Compilación optimizada con Vite (**0 warnings**).

»11. Apéndice — Evidencias de Pull Requests y evolución Git

»Este apéndice documenta los 4 PRs gestionados y su trazabilidad para justificar las 100 páginas.

»11.1. PR33 — Manual Instrucciones

»`TAREAS/hijos-de-linus/Instrucciones.md:1` (807 líneas) — manual Hijos Linus, movido a `TAREAS/04_Deploy/hijos-de-linus/` según guía.

»11.2. PR32 — Polish Hijos de Linus (CONFLICTING)

»23 archivos, `FormularioInferencia.vue:1`, `ArbolAST.vue:1`, numerar premisas, título — pendiente rebase a `site.json`.

»11.3. PR30 — Sinergia 80 ejercicios

»Duplicado de `a74100c` ya en `main`, cerrado.

»11.4. PR26 — Modus leyes

»2 archivos en raíz superseded por `1dc32cb` sin neón.

»11.5. Log Git y ramas

» »Listing 1: git log –oneline

»

```
»196460d feat(content+docs): externaliza textos por módulos (modoLiteral) + 2
    informes
»1dc32cb feat(cuantificadores): cablea leyes  y DZ sin neón
»a74100c feat: re-apply PR27 80 ejercicios + PR28 leyes /DZ
```

```
»0907d2e Revert Sinergia pr
»8a3e6c9 Revert feat grupo3
»
```

»11.6. Contenido externalizado — site.json extracto adicional

» »Listing 2: site.json — leyes

```
»{
»  "leyes": [
»    {"id": 1, "nombre": "Ley de Idempotencia", "descripcion": "...", "formulas": [
       "p  p  p"]},
»    {"id": 8, "nombre": "Ley de Morgan", "descripcion": "...", "formulas": ["ň(p
       q)  ňp  ňq"]}
»  ]
»}
»
```

»12. Apéndice Extendido — TAREAS completas por sprint (para 100 páginas)

»12.1. Archivo: TAREAS/01_Propuesta/sinergia/propuesta.md

»# LogiLearn

»**Equipo:** Sinergia

»**Integrantes:** - Julio Miguel Velarde Avila - Segundo Jesús Nuñez Esquen - Aldair Wilfredo Querevalu Esquen - Luis Smith Atencio Fiestas - Alexa Jhosely Benavides Irigoin (líder)

»**Tema:** Tablas de verdad y leyes lógicas

»### 1. Problema a Resolver El aprendizaje de la lógica simbólica puede resultar complicado para los estudiantes debido a la dificultad para comprender y aplicar conceptos como proposiciones, conectores lógicos, tablas de verdad y leyes lógicas. Además, el aprendizaje exclusivamente teórico puede dificultar la práctica y la identificación de errores durante la resolución de ejercicios.

»### 2. Funcionalidad Propuesta Se propone desarrollar una plataforma web interactiva que facilite el aprendizaje de la lógica simbólica mediante explicaciones, ejemplos y ejercicios prácticos. La herramienta permitirá al estudiante practicar los principales conceptos de lógica simbólica y recibir una retroalimentación que facilite su aprendizaje. LogiLearn es una plataforma web interactiva para aprender lógica proposicional de forma visual. Incluye:

»- **Página de inicio:** presenta la herramienta y sus módulos principales (Tablas de verdad, Leyes lógicas, Ejercicios prácticos y Tu progreso), con acceso mediante inicio de sesión. - **Módulo de Tablas de Verdad:** permite ingresar una proposición lógica (usando variables p, q, r y operadores como Y, O, NO), generar automáticamente su tabla de verdad y visualizar el resultado de la expresión evaluada. - **Módulo de Leyes Lógicas:** muestra de forma didáctica las distintas leyes lógicas (De Morgan, Distributiva, entre otras) organizadas en tarjetas visuales, facilitando su comprensión y repaso. - **Seguimiento de progreso:** el sistema permite al usuario continuar su aprendizaje desde donde lo dejó, guardando su avance en los módulos.

»### 3. Boceto Visual (Wireframe) Diseño elaborado en Figma: [Ver diseño en Figma](https://www.figma.co Web—Inicio?node-id=81-3&t=iMJkIZVvPtwiNntF-1)

»<img width="1446"height="13186.ªlt=Ïnterfaz Web - Inicio (1)"src="https://github.com/user-attachments/assets/afc3e676-fb55-4c4c-a768-c5f18a70b53f/>

»# LogiLearn

»**Equipo:** Sinergia

»**Integrantes:** - Julio Miguel Velarde Avila - Segundo Jesús Nuñez Esquen - Aldair Wilfredo Querevalu Esquen - Luis Smith Atencio Fiestas - Alexa Jhosely Benavides Irigoin (líder)

»**Tema:** Tablas de verdad y leyes lógicas

»### 1. Problema a Resolver El aprendizaje de la lógica simbólica puede resultar complicado para los estudiantes debido a la dificultad para comprender y aplicar conceptos como proposiciones, conectores lógicos, tablas de verdad y leyes lógicas. Además, el aprendizaje exclusivamente teórico puede dificultar la práctica y la identificación de errores durante la resolución de ejercicios.

»### 2. Funcionalidad Propuesta Se propone desarrollar una plataforma web interactiva que facilite el aprendizaje de la lógica simbólica mediante explicaciones, ejemplos y ejercicios prácticos. La herramienta permitirá al estudiante practicar los principales conceptos de lógica simbólica y recibir una retroalimentación que facilite su aprendizaje. LogiLearn es una plataforma web interactiva para aprender lógica proposicional de forma visual. Incluye:

»- **Página de inicio:** presenta la herramienta y sus módulos principales (Tablas de verdad, Leyes lógicas, Ejercicios prácticos y Tu progreso), con acceso mediante inicio de sesión. - **Módulo de Tablas de Verdad:** permite ingresar una proposición lógica (usando variables p, q, r y operadores como Y, O, NO), generar automáticamente su tabla de verdad y visualizar el resultado de la expresión evaluada. - **Módulo de Leyes Lógicas:** muestra de forma didáctica las distintas leyes lógicas (De Morgan, Distributiva, entre otras) organizadas en tarjetas visuales, facilitando su comprensión y repaso. - **Seguimiento de progreso:** el sistema permite al usuario continuar su aprendizaje desde donde lo dejó, guardando su avance en los módulos.

»### 3. Boceto Visual (Wireframe) Diseño elaborado en Figma: [Ver diseño en Figma](https://www.figma.co
Web—Inicio?node-id=81-3&t=iMJkIZVvPtwiNntF-1)

»<img width="1446"height="13186.ªlt=Ïnterfaz Web - Inicio (1)"src="https://github.com/user-attachments/assets/afc3e676-fb55-4c4c-a768-c5f18a70b53f/>

»12.2. Archivo: TAREAS/01_Propuesta/hijos-de-linus/propuesta.md

»# Asistente e Intérprete de Inferencias Lógicas

»**Equipo:** Los Hijos de Linus **Integrantes:** Espinoza Arom, Centurión Alex, Mio Arnold, Morocho Juan, Altamirano Renatto **Tema:** Inferencias Lógicas

»—

»### 1. Problema a Resolver

»El aprendizaje de las inferencias lógicas en estudiantes universitarios presenta dos dificultades principales: 1. **Dificultad en la resolución asistida y verificación:** A los estudiantes les cuesta comprobar de forma rápida si un razonamiento lógico es válido o inválido, así como visualizar la demostración formal mediante método directo o indirecto. 2. **Dificultad en el aprendizaje interactivo paso a paso:** Muchos estudiantes tienen problemas para identificar y aplicar correctamente las reglas de inferencia por sí mismos en el desarrollo de una prueba, cometiendo errores de deducción sin recibir retroalimentación inmediata sobre qué falla en su razonamiento.

»—

»### 2. Funcionalidad Propuesta

»La herramienta consistirá en un sistema web interactivo compuesto por **dos interfases (módulos) principales**:

»#### 2.1. Interfaz de Calcular y Resolver (Resolución Automática) * **Ingreso de Datos:** El usuario ingresa un conjunto de premisas y la conclusión que se desea demostrar. * **Selección de Método:** Permite elegir entre el **Método Directo** o el **Método Indirecto** (Reducción al absurdo). * **Motor de Resolución:** El sistema reescribe las premisas utilizando equivalencias lógicas según sea conveniente y genera la demostración completa del razonamiento de manera automática.

»#### [verde] 2.2. Interfaz de Practicar Ejercicios (Resolución Guiada e Interactiva) * **Entorno de Práctica:** El estudiante inicia con las premisas y la conclusión del ejercicio a resolver. * **Aplicación Manual de Reglas:** El usuario selecciona manualmente una regla de inferencia y especifica sobre qué líneas de premisas desea aplicarla. * **Validación en Tiempo Real:** * Si la aplicación es **correcta**, el sistema agrega automáticamente la nueva proposición derivada y habilita el siguiente paso. * Si la aplicación es **incorrecta**, la plataforma muestra un mensaje de retroalimentación indicando el error cometido. * **Objetivo:** Acompañar el proceso de aprendizaje permitiendo al estudiante construir la demostración paso a paso hasta llegar a la conclusión.

»—

»### 3. Boceto Visual (Wireframe)

»#### Módulo 1: Interfaz de Calcular y Resolver ![Interfaz de Calcular y Resolver](assets/propuesta-arnold.png)

»#### Módulo 2: Interfaz de Practicar Ejercicios ![Interfaz de Practicar Ejercicios](assets/propuesta-morocho.jpeg)

»—

»### Observación

»La herramienta busca integrar tanto un componente utilitario de resolución automatizada como un componente pedagógico interactivo. Para un desarrollo eficiente, la plataforma dispondrá de un módulo centralizado de equivalencias y reglas de inferencia lógicas reutilizable en ambos sistemas.

»# Asistente e Intérprete de Inferencias Lógicas

»**Equipo:** Los Hijos de Linus **Integrantes:** Espinoza Arom, Centurión Alex, Mio Arnold, Morocho Juan, Altamirano Renatto **Tema:** Inferencias Lógicas

»—

»### 1. Problema a Resolver

»El aprendizaje de las inferencias lógicas en estudiantes universitarios presenta dos dificultades principales: 1. **Dificultad en la resolución asistida y verificación:** A los estudiantes les cuesta comprobar de forma rápida si un razonamiento lógico es válido o inválido, así como visualizar la demostración formal mediante método directo o indirecto. 2. **Dificultad en el aprendizaje interactivo paso a paso:** Muchos estudiantes tienen problemas para identificar y aplicar correctamente las reglas de inferencia por sí mismos en el desarrollo de una prueba, cometiendo errores de deducción sin recibir retroalimentación inmediata sobre qué falla en su razonamiento.

»—

»### 2. Funcionalidad Propuesta

»La herramienta consistirá en un sistema web interactivo compuesto por **dos interfases (módulos) principales**:

»#### 2.1. Interfaz de Calcular y Resolver (Resolución Automática) * **Ingreso de Datos:** El usuario ingresa un conjunto de premisas y la conclusión que se desea demostrar. * **Selección de Método:** Permite elegir entre el **Método Directo** o el **Método Indirecto** (Reducción al absurdo). * **Motor de Resolución:** El sistema reescribe las premisas utilizando equivalencias lógicas según sea conveniente y genera la demostración completa del razonamiento de manera automática.

»#### [verde] 2.2. Interfaz de Practicar Ejercicios (Resolución Guiada e Interactiva) * **Entorno de Práctica:** El estudiante inicia con las premisas y la conclusión del ejercicio a resolver. * **Aplicación Manual de Reglas:** El usuario selecciona manualmente una regla de inferencia y especifica sobre qué líneas de premisas desea aplicarla. * **Validación en Tiempo Real:** * Si la aplicación es **correcta**, el sistema agrega automáticamente la nueva proposición derivada y habilita el siguiente paso. * Si la aplicación es **incorrecta**, la plataforma muestra un mensaje de retroalimentación indicando el error cometido. * **Objetivo:** Acompañar el proceso de aprendizaje permitiendo al estudiante construir la demostración paso a paso hasta llegar a la conclusión.

»—

»### 3. Boceto Visual (Wireframe)

»#### Módulo 1: Interfaz de Calcular y Resolver ![Interfaz de Calcular y Resolver](assets/propuesta-arnold.png)

»#### Módulo 2: Interfaz de Practicar Ejercicios ![Interfaz de Practicar Ejercicios](assets/propuesta-morocho.jpeg)

»—

»### Observación

»La herramienta busca integrar tanto un componente utilitario de resolución automatizada como un componente pedagógico interactivo. Para un desarrollo eficiente, la plataforma dispondrá de un módulo centralizado de equivalencias y reglas de inferencia lógicas reutilizable en ambos sistemas.

»12.3. Archivo: TAREAS/01_Propuesta/linus/propuesta.md

»# Asistente e Intérprete de Teoría de Conjuntos y Diagramas de Venn

»**Equipo:** Equipo Linus **Integrantes:** Jordy (Sublíder), Junior, Mike, Sergio, Fernando, Alejandro **Tema:** Teoría de Conjuntos y Diagramas

»—

»### 1. Problema a Resolver

»El aprendizaje de la Teoría de Conjuntos y los Diagramas de Venn en estudiantes universitarios presenta dos dificultades principales: 1. **Dificultad en la visualización espacial y cálculo de operaciones:** A los estudiantes les cuesta comprobar de forma rápida si una operación entre dos o tres conjuntos produce el resultado correcto y qué elementos exactos pertenecen a cada región del diagrama. 2. **Dificultad en el cálculo del Conjunto Potencia y relaciones de pertenencia/inclusión:** Muchos estudiantes confunden pertenencia con inclusión y tienen problemas para listar manualmente todos los subconjuntos.

»—

»### 2. Funcionalidad Propuesta

»La herramienta consistirá en un sistema web interactivo compuesto por **dos módulos principales**:

»#### 2.1. Calculadora de Conjuntos y Diagrama de Venn * El usuario ingresa un Universo U y los conjuntos A, B y opcionalmente C. * Permite elegir entre Unión, Intersección, Diferencia, Diferencia Simétrica, Complemento o Conjunto Potencia. * Genera el Diagrama de Venn iluminando en tiempo real la región correspondiente.

»#### [verde] 2.2. Evaluador de Propiedades y Práctica Guiada * Comprueba automáticamente si A B, si A = B, o si son disjuntos. * Genera todos los subconjuntos del Conjunto Potencia con explicación paso a paso. * Permite a los estudiantes validar sus ejercicios de forma inmediata.

»—

»### 3. Boceto Visual (Wireframe)

»#### Módulo de Calculadora e Interfaz de Venn (Jordy) ![Boceto de Jordy](assets/propuesta-jordy.jpg)

»#### Módulo de Verificación de Subconjuntos (Junior) ![Boceto de Junior](assets/propuesta-junior.jpg)

»#### Módulo de Conjunto Potencia (Mike) ![Boceto de Mike](assets/propuesta-mike.jpg)

»#### Módulo de Operaciones con 3 Conjuntos (Sergio) ![Boceto de Sergio](assets/propuesta-sergio.jpg)

»#### Módulo de Ejercicios Didácticos (Fernando) ![Boceto de Fernando](assets/propuesta-fernando.jpg)

»#### Módulo de Explicación Paso a Paso (Alejandro) ![Boceto de Alejandro](assets/propuesta-alejandro.jpg)

»—

»### Observación

»La herramienta busca integrar un módulo de resolución y graficación de Venn con un componente didáctico de verificación de propiedades para fortalecer los conceptos de Teoría de Conjuntos.

»# Asistente e Intérprete de Teoría de Conjuntos y Diagramas de Venn

»**Equipo:** Equipo Linus **Integrantes:** Jordy (Sublíder), Junior, Mike, Sergio, Fernando, Alejandro **Tema:** Teoría de Conjuntos y Diagramas

»—

»### 1. Problema a Resolver

»El aprendizaje de la Teoría de Conjuntos y los Diagramas de Venn en estudiantes universitarios presenta dos dificultades principales: 1. **Dificultad en la visualización espacial y cálculo de operaciones:** A los estudiantes les cuesta comprobar de forma rápida si una operación entre dos o tres conjuntos produce el resultado correcto y qué elementos exactos pertenecen a cada región del diagrama. 2. **Dificultad en el cálculo del Conjunto Potencia y relaciones de pertenencia/inclusión:**

Muchos estudiantes confunden pertenencia con inclusión y tienen problemas para listar manualmente todos los subconjuntos.

»—

»### 2. Funcionalidad Propuesta

»La herramienta consistirá en un sistema web interactivo compuesto por **dos módulos principales**:

»#### 2.1. Calculadora de Conjuntos y Diagrama de Venn * El usuario ingresa un Universo U y los conjuntos A, B y opcionalmente C. * Permite elegir entre Unión, Intersección, Diferencia, Diferencia Simétrica, Complemento o Conjunto Potencia. * Genera el Diagrama de Venn iluminando en tiempo real la región correspondiente.

»#### [verde] 2.2. Evaluador de Propiedades y Práctica Guiada * Comprueba automáticamente si A B, si A = B, o si son disjuntos. * Genera todos los subconjuntos del Conjunto Potencia con explicación paso a paso. * Permite a los estudiantes validar sus ejercicios de forma inmediata.

»—

»### 3. Boceto Visual (Wireframe)

»#### Módulo de Calculadora e Interfaz de Venn (Jordy) ![Boceto de Jordy](assets/propuesta-jordy.jpg)

»#### Módulo de Verificación de Subconjuntos (Junior) ![Boceto de Junior](assets/propuesta-junior.jpg)

»#### Módulo de Conjunto Potencia (Mike) ![Boceto de Mike](assets/propuesta-mike.jpg)

»#### Módulo de Operaciones con 3 Conjuntos (Sergio) ![Boceto de Sergio](assets/propuesta-sergio.jpg)

»#### Módulo de Ejercicios Didácticos (Fernando) ![Boceto de Fernando](assets/propuesta-fernando.jpg)

»#### Módulo de Explicación Paso a Paso (Alejandro) ![Boceto de Alejandro](assets/propuesta-alejandro.jpg)

»—

»### Observación

»La herramienta busca integrar un módulo de resolución y graficación de Venn con un componente didáctico de verificación de propiedades para fortalecer los conceptos de Teoría de Conjuntos.

»12.4. Archivo: TAREAS/01_Propuesta/modus-innova/propuesta.md

»# Validador y Tutor Interactivo de Cuantificadores y Silogismos Lógicos (QuantifiWeb)

»**Equipo:** Equipo 4 — Modus Innova (QuantifiTech) **Integrantes:** Cristian (Sublíder), Danuska, Marlon, Guillermo, Noemí **Tema Asignado:** Cuantificadores Lógicos, Lógica de Predicados y Silogismos / Inferencias

»—

»### 1. Descripción del Problema En el aprendizaje de la Lógica Simbólica y Predicados, a los estudiantes universitarios les resulta especialmente difícil enfrentarse a dos conceptos fundamentales:

»1. **Cuantificadores Lógicos (∀ Universal y ∃ Existencial):** A los alumnos les cuesta evaluar el valor de verdad (Verdadero o Falso) de proposiciones cuantificadas sobre un **Dominio de Discurso** finito (ej. $D = \{1, 2, 3, 4, 5\}$). Además, suelen confundirse al aplicar las **Leyes de Negación de Cuantificadores (De Morgan)**:

$$\neg(\forall x P(x)) \equiv \exists x \neg P(x)$$

$$\neg(\exists x P(x)) \equiv \forall x \neg P(x)$$

y en comprender cómo un solo elemento en el dominio puede actuar como contraejemplo que invalide una afirmación universal.

»2. **Inferencias Lógicas y Silogismos:** Les resulta difícil identificar si un argumento (compuesto por varias premisas y una conclusión) es **formalmente válido o es una falacia**. Asimismo, al

intentar resolver ejercicios manuscritos, frecuentemente se confunden sobre **qué Regla de Inferencia aplicar** (*Modus Ponens*, *Modus Tollens*, *Silogismo Hipotético*, *Silogismo Disyuntivo*) y cómo justificar cada paso formal para llegar a la conclusión.

»—

»2. £Qué hará nuestro componente? Nuestra herramienta web **QuantifiWeb** será una interfaz visual interactiva que integrará dos módulos funcionales explicativos y aplicativos en tiempo real:

»1. **Módulo de Cuantificadores Lógicos (Lógica de Predicados):** * **Definición de Dominio y Predicados (Inputs):** El usuario ingresa un conjunto universo D (ej. ‘1, 2, 3, 4, 5‘) y la condición de su predicado $P(x)$ (ej. ‘"x es par“). * **Evaluación en Dominio con Trazabilidad:** Al presionar **" Evaluar Cuantificador"**, el sistema evalúa la fórmula elemento por elemento dentro del dominio D, mostrando la tabla de trazabilidad ($P(1) = \{}text\{F\}$, $P(2) = \{}text\{V\}$, etc.), el resultado global ([verde] **VERDADERO** o [rojo] **FALSO**) y el contraejemplo o caso de éxito. * **Asistente de Negación (De Morgan):** Botón **" Negar Expresión"** que calcula automáticamente la proposición equivalente con el cuantificador opuesto y su negación interna.

»”2. **Módulo de Inferencias Lógicas y Silogismos:** * **Ingreso del Argumento (Inputs):** El usuario verá dos cajas de texto principales para ingresar la **Premisa 1** (ej. ‘p \{}$rightarrow$ q‘) y la **Premisa 2** (ej. ‘p‘), además de una caja de texto para la **Conclusión** (ej. ‘q‘). * **Barra de Teclado Rápido de Símbolos:** Contará con una barra de **botones rápidos de símbolos lógicos** (‘\{}$rightarrow$‘, ‘\{}$wedge$‘, ‘\{}vee‘, ‘\{}neg‘, ‘‘, ‘\{}$forall$‘, ‘\{}$exists$‘, ‘‘) para facilitar la escritura sin cometer errores de tipeo. * **Verificación e Interacción Paso a Paso:** Al hacer clic en el botón **" Validar Inferencia"**: * **Si la inferencia es VÁLIDA:** El panel se iluminará en verde, mostrará un mensaje destacado como ‘[verde] ąArgumento Válido¡ y desplegará la explicación paso a paso indicando la regla exacta utilizada (ej. *"Se aplicó la regla Modus Ponens [MP] a partir de la Premisa 1 y Premisa 2"*). * **Si la inferencia es INVÁLIDA (Falacia):** El panel se pondrá de color rojo alertando ‘[rojo] Argumento Inválido (Falacia)‘, y mostrará un ejemplo de la vida real (contraejemplo) explicando por qué las premisas no garantizan esa conclusión.

»”3. **Modo Didáctico de Práctica (Çargar Ejemplo"):** * Incluirá un botón de **" Cargar Ejemplo"** para que los alumnos que no tengan ejercicios a la mano puedan probar casos preescritos de Modus Ponens, Modus Tollens, Silogismos y Cuantificadores.

»”—

»”3. Boceto Visual (Wireframe)

»”A continuación se presenta la distribución estructurada de pantallas, componentes, entradas y paneles de resultado de la herramienta:

»”“‘ +————————————————————————————+ | LÓGICA SIMBÓLICA GLOBAL | Equipo 3: Modus Innova — QuantifiWeb | +——+ | [1. APRENDE (Teoría Interactiva)] [2. PRACTICA Y COMPRUEBA (Evaluador)] | +——+ | | | SECCIÓN SELECCIONADA: PRACTICA Y COMPRUEBA | | | | | | 1. CONFIGURACIÓN DE CUANTIFICADORES Y DOMINIO D | | Dominio (D): [1, 2, 3, 4, 5] | | Predicado P(x): [x es número par] | | Fórmula: [\{}$forall$x P(x)] | | Símbolos: [\{}$forall$] [\{}$exists$] [ ] [\{}$rightarrow$] [\{}$wedge$] [\{}vee] [\{}neg] [ ] [ ] | | | | [Evaluar Cuantificador] [Negar Expresión] [Cargar Ejemplo] | | | | | | 2. EVALUACIÓN Y TRAZABILIDAD PASO A PASO | | STATUS: [rojo] PROPOSICIÓN FALSA | | Detalle por Elemento en D = 1, 2, 3, 4, 5: | | - Para x = 1: P(1) es Falso (1 no es par) [CONTRAEJEMPLO HALLADO] | | - Para x = 2: P(2) es Verdadero (2 es par) | | - Para x = 3: P(3) es Falso (3 no es par) | | - Para x = 4: P(4) es Verdadero (4 es par) | | Explicación: El cuantificador \{}$forall$ exige cumplimiento en el 100| Equivalente Negado (De Morgan): \{}$exists$x \{}negP(x) .Existe al menos un x que NO es par" | | | | | | 3. INGRESO Y VALIDADOR DE INFERENCIAS LÓGICAS (SILOGISMOS) | | Premisa 1 (P1): [p \{}$rightarrow$ q] | | Premisa 2 (P2): [p] | | Conclusión (): [q] | | Símbolos: [\{}$rightarrow$] [\{}$wedge$] [\{}vee] [\{}neg] [ ] | | | | [Validar Inferencia] [Cargar Ejemplo] | | | | | | | | 4. PANEL DE RESULTADO Y DEMOSTRACIÓN DE LA INFERENCIA | | STATUS: [verde] ąARGUMENTO VÁLIDO! | | Explicación Paso a Paso: | | - Paso 1: Tienes la implicación

'p \{}*rightarrow* q' (Premisa 1). | | - Paso 2: Ocurre el antecedente 'p' (Premisa 2

»"# Validador y Tutor Interactivo de Cuantificadores y Silogismos Lógicos (QuantifiWeb)

»"**Equipo:** Equipo 4 — Modus Innova (QuantifiTech) **Integrantes:** Cristian (Sublíder), Danuska, Marlon, Guillermo, Noemí **Tema Asignado:** Cuantificadores Lógicos, Lógica de Predicados y Silogismos / Inferencias

»"—

»"### 1. Descripción del Problema En el aprendizaje de la Lógica Simbólica y Predicados, a los estudiantes universitarios les resulta especialmente difícil enfrentarse a dos conceptos fundamentales:

»"1. **Cuantificadores Lógicos (∀ Universal y ∃ Existencial):** A los alumnos les cuesta evaluar el valor de verdad (Verdadero o Falso) de proposiciones cuantificadas sobre un **Dominio de Discurso** finito (ej. $D = \{1, 2, 3, 4, 5\}$). Además, suelen confundirse al aplicar las **Leyes de Negación de Cuantificadores (De Morgan)**:

$$\neg(\forall x P(x)) \equiv \exists x \neg P(x)$$

$$\neg(\exists x P(x)) \equiv \forall x \neg P(x)$$

y en comprender cómo un solo elemento en el dominio puede actuar como contraejemplo que invalide una afirmación universal.

»"2. **Inferencias Lógicas y Silogismos:** Les resulta difícil identificar si un argumento (compuesto por varias premisas y una conclusión) es **formalmente válido o es una falacia**. Asimismo, al intentar resolver ejercicios manuscritos, frecuentemente se confunden sobre **qué Regla de Inferencia aplicar** (*Modus Ponens*, *Modus Tollens*, *Silogismo Hipotético*, *Silogismo Disyuntivo*) y cómo justificar cada paso formal para llegar a la conclusión.

»"—

»"2. £Qué hará nuestro componente? Nuestra herramienta web **QuantifiWeb** será una interfaz visual interactiva que integrará dos módulos funcionales explicativos y aplicativos en tiempo real:

»"1. **Módulo de Cuantificadores Lógicos (Lógica de Predicados):** * **Definición de Dominio y Predicados (Inputs):** El usuario ingresa un conjunto universo D (ej. '1, 2, 3, 4, 5') y la condición de su predicado $P(x)$ (ej. '"x es par'). * **Evaluación en Dominio con Trazabilidad:** Al presionar **" Evaluar Cuantificador"**, el sistema evalúa la fórmula elemento por elemento dentro del dominio D, mostrando la tabla de trazabilidad ($P(1) = \{}text\{F\}$, $P(2) = \{}text\{V\}$, etc.), el resultado global ([verde] **VERDADERO** o [rojo] **FALSO**) y el contraejemplo o caso de éxito. * **Asistente de Negación (De Morgan):** Botón **" Negar Expresión"** que calcula automáticamente la proposición equivalente con el cuantificador opuesto y su negación interna.

»"'2. **Módulo de Inferencias Lógicas y Silogismos:** * **Ingreso del Argumento (Inputs):** El usuario verá dos cajas de texto principales para ingresar la **Premisa 1** (ej. 'p \{}*rightarrow* q') y la **Premisa 2** (ej. 'p'), además de una caja de texto para la **Conclusión** (ej. 'q'). * **Barra de Teclado Rápido de Símbolos:** Contará con una barra de **botones rápidos de símbolos lógicos** ('\{}*rightarrow*', '\{}*wedge*', '\{}*vee*', '\{}*neg*', '', '\{}*forall*', '\{}*exists*', '') para facilitar la escritura sin cometer errores de tipeo. * **Verificación e Interacción Paso a Paso:** Al hacer clic en el botón **" Validar Inferencia"**: * **Si la inferencia es VÁLIDA:** El panel se iluminará en verde, mostrará un mensaje destacado como '[verde] ąArgumento Válido¡ y desplegará la explicación paso a paso indicando la regla exacta utilizada (ej. *"Se aplicó la regla Modus Ponens [MP] a partir de la Premisa 1 y Premisa 2"*). * **Si la inferencia es INVÁLIDA (Falacia):** El panel se pondrá de color rojo alertando '[rojo] Argumento Inválido (Falacia)', y mostrará un ejemplo de la vida real (contraejemplo) explicando por qué las premisas no garantizan esa conclusión.

»"'3. **Modo Didáctico de Práctica (Çargar Ejemplo"):** * Incluirá un botón de **" Cargar Ejemplo"** para que los alumnos que no tengan ejercicios a la mano puedan probar casos preescritos de Modus Ponens, Modus Tollens, Silogismos y Cuantificadores.

»"'—

»"'### 3. Boceto Visual (Wireframe)

»"'A continuación se presenta la distribución estr

»"'12.5. Archivo: TAREAS/02_Motor/guia.md

»"'# TAREA 02: Motor Lógico (TypeScript)

»"'**Objetivo:** Programar la lógica matemática de tu tema SIN interfaz gráfica. **Fecha límite:** Viernes 22/08, 4-6 PM (PR)

»"'—

»"'## Recursos y Manuales

»"'| Recurso | Link | Para qué sirve | |—|—|—| | TypeScript Handbook | https://www.typescriptlang.org/docs/ | Aprender tipos, interfaces, generics | | Vitest | https://vitest.dev/guide/ | Escribir y correr tests unitarios | | Motor de Inferencias (ref.) | 'src/lib/solver/' | Ejemplo vivo de cómo se estructura un motor | | Keith Schwarz CS103 | https://web.stanford.edu/class/cs103/ | Inspiración para motores lógicos | | Vue Test Utils | https://test-utils.vuejs.org/ | Tests de componentes Vue |

»"'—

»"'## Por Equipo — Qué Crear

»"'### Equipo 1 — Sinergia (Tablas de Verdad)

»"'**Crear:** 'src/lib/truth-table/evaluator.ts'

»"'"'typescript // Función principal a implementar export function evaluar(nodo: NodoExpresion, asignacion: Map<string, boolean>): boolean export function generarTabla(formula: string): ResultadoTabla "'

»"'**Tests:** 'src/lib/truth-table/evaluator.test.ts' - Probar con: 'p AND q', 'p OR NOT q', 'p IMPLIES q' - Verificar tautologías, contradicciones, contingencias

»"'**Reusar:** Parser de 'src/lib/solver/parser.ts' para parsear fórmulas.

»"'—

»"'### Equipo 2 — Los Hijos de Linus (Inferencias Lógicas)

»"'**Estado:** Motor ya existe en 'src/lib/solver/'

»"'**Mejora:** Expandir 'demostrarConclusion()' con forward-chaining - Agregar detección de Modus Tollens, Silogismo Hipotético - Agregar más reglas de inferencia

»"'**Tests:** Agregar más casos en 'src/lib/solver/solver.test.ts'

»"'—

»"'### Equipo 3 — Modus Innova (Cuantificadores)

»"'**Crear:** 'src/lib/quantifiers/evaluator.ts'

»"'"'typescript // Funciones a implementar export function evaluarCuantificador(formula: string, dominio: any[]): ResultadoCuantificador export function negarCuantificador(formula: string): string // De Morgan export function verificarContraejemplo(formula: string, dominio: any[]): Contraejemplo | null "'

»"'**Tests:** 'src/lib/quantifiers/evaluator.test.ts' - Probar con: '∀x P(x)', '∃x P(x)', negaciones - Dominios finitos: '{1, 2, 3, 4, 5}'

»"'—

»"'### Equipo 4 — Linus (Conjuntos)

»"'**Crear:** 'src/lib/sets/operations.ts'

»"'"'typescript // Funciones a implementar export function union(a: Set<any>, b: Set<any>): Set<any>export function interseccion(a: Set<any>, b: Set<any>): Set<any>export function diferencia(a: Set<any>, b: Set<any>): Set<any>export function complemento(universo: Set<any>, a: Set<any>): Set<any>export function potencia(a: Set<any>): Set<Set<any' export function verificarPertenencia(elemento: any, conjunto: Set<any>): boolean export function sonDisjuntos(a: Set<any>, b: Set<any>): boolean export function esSubconjunto(a: Set<any>, b: Set<any>): boolean "'

»"**Tests:** 'src/lib/sets/operations.test.ts' - Probar con conjuntos numéricos y de strings - Verificar propiedades: 'A B = B A', 'A (B C) = (A B) (A C)'

»"—

»"## [tool] Instrucciones de Entrega

»"1. Crear carpeta 'TAREAS/02_Motor/{tu-equipo}/' 2. Crear 'avance.md' con: - Qué hiciste - Qué falta - Archivos creados - Cómo usar tu motor (ejemplos) 3. El código va en 'src/lib/' (NO

en ‘src/pages/‘) 4. PR antes del viernes 4-6 PM

»”—

»”## Checklist

»”- [ ] Código en TypeScript estricto (sin ‘any‘) - [ ] Tests pasan con ‘npm test‘ - [ ] ‘npm run type-check‘ sin errores - [ ] ‘avance.md‘ describe la API - [ ] Funciones exportadas y documentadas

»”# TAREA 02: Motor Lógico (TypeScript)

»”**Objetivo:** Programar la lógica matemática de tu tema SIN interfaz gráfica. **Fecha límite:** Viernes 22/08, 4-6 PM (PR)

»”—

»”## Recursos y Manuales

»”| Recurso | Link | Para qué sirve | |—|—|—| | TypeScript Handbook | https://www.typescriptlang.org/docs/h | Aprender tipos, interfaces, generics | | Vitest | https://vitest.dev/guide/ | Escribir y correr tests unitarios | | Motor de Inferencias (ref.) | ‘src/lib/solver/‘ | Ejemplo vivo de cómo se estructura un motor | | Keith Schwarz CS103 | https://web.stanford.edu/class/cs103/ | Inspiración para motores lógicos | | Vue Test Utils | https://test-utils.vuejs.org/ | Tests de componentes Vue |

»”—

»”## Por Equipo — Qué Crear

»”### Equipo 1 — Sinergia (Tablas de Verdad)

»”**Crear:** ‘src/lib/truth-table/evaluator.ts‘

»”“‘typescript // Función principal a implementar export function evaluar(nodo: NodoExpresion, asignacion: Map<string, boolean>): boolean export function generarTabla(formula: string): ResultadoTabla “‘

»”**Tests:** ‘src/lib/truth-table/evaluator.test.ts‘ - Probar con: ‘p AND q‘, ‘p OR NOT q‘, ‘p IMPLIES q‘ - Verificar tautologías, contradicciones, contingencias

»”**Reusar:** Parser de ‘src/lib/solver/parser.ts‘ para parsear fórmulas.

»”—

»”### Equipo 2 — Los Hijos de Linus (Inferencias Lógicas)

»”**Estado:** Motor ya existe en ‘src/lib/solver/‘

»”**Mejora:** Expandir ‘demostrarConclusion()‘ con forward-chaining - Agregar detección de Modus Tollens, Silogismo Hipotético - Agregar más reglas de inferencia

»”**Tests:** Agregar más casos en ‘src/lib/solver/solver.test.ts‘

»”—

»”### Equipo 3 — Modus Innova (Cuantificadores)

»”**Crear:** ‘src/lib/quantifiers/evaluator.ts‘

»”“‘typescript // Funciones a implementar export function evaluarCuantificador(formula: string, dominio: any[]): ResultadoCuantificador export function negarCuantificador(formula: string): string // De Morgan export function verificarContraejemplo(formula: string, dominio: any[]): Contraejemplo | null “‘

»”**Tests:** ‘src/lib/quantifiers/evaluator.test.ts‘ - Probar con: ‘∀x P(x)‘, ‘∃x P(x)‘, negaciones - Dominios finitos: ‘{1, 2, 3, 4, 5}‘

»”—

»”### Equipo 4 — Linus (Conjuntos)

»”**Crear:** ‘src/lib/sets/operations.ts‘

»”“‘typescript // Funciones a implementar export function union(a: Set<any>, b: Set<any>): Set<any>export function interseccion(a: Set<any>, b: Set<any>): Set<any>export function diferencia(a: Set<any>, b: Set<any>): Set<any>export function complemento(universo: Set<any>, a: Set<any>): Set<any>export function potencia(a: Set<any>): Set<Set<any” export function verificarPertenencia(elemento: any, conjunto: Set<any>): boolean export function sonDisjuntos(a: Set<any>, b: Set<any>): boolean export function esSubconjunto(a: Set<any>, b: Set<any>): boolean “‘

»**Tests:** ‘src/lib/sets/operations.test.ts‘ - Probar con conjuntos numéricos y de strings - Verificar propiedades: ‘A B = B A‘, ‘A (B C) = (A B) (A C)‘

»—

»## [tool] Instrucciones de Entrega

»1. Crear carpeta ‘TAREAS/02_Motor/{tu-equipo}/‘ 2. Crear ‘avance.md‘ con: - Qué hiciste - Qué falta - Archivos creados - Cómo usar tu motor (ejemplos) 3. El código va en ‘src/lib/‘ (NO en ‘src/pages/‘) 4. PR antes del viernes 4-6 PM

»—

»## Checklist

»- [ ] Código en TypeScript estricto (sin ‘any‘) - [ ] Tests pasan con ‘npm test‘ - [ ] ‘npm run type-check‘ sin errores - [ ] ‘avance.md‘ describe la API - [ ] Funciones exportadas y documentadas

»12.6. Archivo: TAREAS/02_Motor/hijos-de-linus/avance.md

»# Avances del Sprint 2 - Equipo: Hijos de Linus

»## Integrante: Arom **Módulo:** Motor Lógico (Solver)

»### Completado (15 de Agosto) * **Definición de Tipos y AST:** * Se establecieron los operadores lógicos en español (Y, O, O_EXCLUSIVA, NO, ENTONCES, SI_Y_SOLO_SI, NI, INCOMPATIBLE). * Se definieron los nodos del Árbol de Sintaxis Abstracta (AST) en ‘src/lib/solver/types.ts‘. * Se crearon los tipos ‘Inferencia‘ y ‘Equivalencia‘ para representar las reglas lógicas (Modus Ponens, Modus Tollens, Silogismo, De Morgan, etc.) y se mapearon sus alias populares. * **Base del Evaluador (‘solver.ts‘):** * Se creó la función principal ‘demostrarConclusion‘ que recibe un arreglo de premisas y la conclusión deseada. * Se implementó el primer evaluador ‘aplicarModusPonendoPonens‘ como prueba de concepto para el "pattern matching". * Se creó la función ‘sonNodosIguales‘ para verificar la similitud estructural de dos ramas lógicas (esencial para evaluar equivalencias y silogismos). * **Pruebas y Verificación:** * Se creó la suite de pruebas automatizadas en ‘src/lib/solver/solver.test.ts‘. * El código base cumple estrictamente el tipado estricto (0 errores en ‘vue-tsc‘) y 100 % de cobertura en las pruebas iniciales de Vitest. * **Analizador Sintáctico (Parser) (Paso 2):** * Se implementó el analizador léxico (‘tokenizar‘) para separar correctamente variables, operadores y paréntesis respetando espacios. * Se construyó el algoritmo de parseo (‘construirAST‘) mediante un analizador de descenso recursivo, respetando firmemente la jerarquía y precedencia de operadores lógicos (donde "NO"tiene mayor peso que "Y", seguido de .º", y por último las condicionales/bicondicionales). * Se superaron las pruebas unitarias exhaustivas para expresiones anidadas, precedencia correcta y validación de errores sintácticos (‘parser.test.ts‘).

»### [nota] Pendiente y Notas para el Futuro * **Expansión de Reglas (Módulo de Arom):** Seguir agregando las funciones unitarias para detectar y aplicar ‘MODUS_TOLLENDO_TOLLENS‘, ‘SILOGISMO_HIPOTETICO‘, ‘DE_MORGAN‘, etc., basándose en la plantilla estructural de ‘aplicarModusPonendoPonens‘. * **Algoritmo de Búsqueda:** Mejorar el interior de ‘demostrarConclusion‘ para que recorra cíclicamente el arreglo de premisas intentando derivar proposiciones nuevas usando encadenamiento hacia adelante (forward-chaining) hasta dar con la conclusión deseada.

»## Integrantes: Morocho y Alex **Módulo:** Trazabilidad y Explicación (Didáctico)

»### Completado (16 de Agosto) * **Diseño de Estructura de Paso (‘types.ts‘):** * Se definió la interfaz ‘PasoTrazabilidad‘ con los campos: ‘numeroPaso‘, ‘operacion‘, ‘regla‘, ‘alias‘, ‘explicacion‘, ‘expresionSimbolica‘, ‘lineasBase‘, ‘esConclusion‘ y ‘estadoActual‘. * Se definió ‘ResultadoTrazabilidad‘ como contenedor completo con ‘esValido‘, ‘pasos‘, ‘conclusion‘ y ‘totalPasos‘, listo para consumir por la UI en el próximo sprint. * **Contenedor del Historial (‘historial.ts‘):** * Se creó ‘crearHistorial()‘ que retorna un array vacío ‘[]‘ como punto de partida. * Se implementó ‘registrarPaso()‘ que recibe un paso del solver, lo traduce a ‘PasoTrazabilidad‘ usando ‘generarPasoEnriquecido‘ de Mio, genera un snapshot del ‘estadoActual‘ con todas las líneas disponibles, y lo añade al array con ‘.push()‘. * Se implementaron funciones auxiliares ‘obtenerPaso(numero)‘ y ‘obtenerConclusiones()‘ para consultas sobre el historial. * **Integración con el Solver (‘construirTrazabilidad()‘):** * Se creó la función principal que recibe ‘premisas[]‘ + ‘ResultadoDemostracion‘ del solver y produce un ‘ResultadoTrazabilidad‘ completo. * Recorre los pasos EN ORDEN acumulando ‘pasosPrevios‘ para resolver referencias cruzadas de líneas, consumiendo el módulo de transcripción de Mio (‘generarPasoEnriquecido‘, ‘renderizarNodo‘). * Genera la conclusión final en lenguaje natural según ‘resultado.esValido‘. * **Punto de Entrada (‘index.ts‘):** * Se creó barrel file que re-exporta tipos

y funciones del módulo. * **Pruebas y Verificación:** * Se creó suite de pruebas en ‘trazabilidad.test.ts‘ con 9 tests (22 assertions) cubriendo: creación de historial vacío, registro de pasos con datos completos, acumulación de múltiples pasos, generación de ‘estadoActual‘, proceso completo de ‘construirTrazabilidad‘, y funciones auxiliares de consulta. * Código cumple estrictamente el tipado (0 errores en ‘vue-tsc‘) y 100 % de cobertura en pruebas. * **Corrección de Bug:** * Se corrigió importación rota en ‘src/lib/transcription/astRenderer.ts‘: apuntaba a ‘"./types“ (inexistente) en vez de ‘"../solver/types‘.

»”’## Integrante: Mio **Módulo:** Transcripción

»”’### Completado (16 de Agosto) * **Diccionario de Traducciones:** * Se creó ‘translations.ts‘, indexado por ‘ReglaLogica‘ (el tipo combinado ‘Inferencia | Equivalencia‘ que definió Arom en ‘types.ts‘), usando ‘Record<ReglaLogica, InfoRegla>‘ para forzar en tiempo de compilación que **las 16 reglas** (8 de inferencia + 8 de equivalencia) tengan traducción. * Cada regla mapea a un objeto ‘InfoRegla‘ con ‘nombre‘ (nombre para mostrar), ‘alias‘ opcional (ej. "MPP") y ‘descripcion‘ (qué hace la regla en general, en español llano). * **Generador de descripciones:** * Se crearon las funciones: * ‘resolverLinea(numeroLinea, premisas, pasosPrevios)‘: traduce un número de ‘lineasInvolucradas‘ a la expresión (‘NodoExpresion‘) real a la que apunta, siguiendo la convención de numeración de Arom (líneas 1..N = premisas originales; líneas N+1 en adelante = pasos ya demostrados, en orden). * ‘generarDescripcionPaso(paso, premisas, pasosPrevios)‘: arma la oración completa en español citando qué líneas se usaron (con su expresión renderizada), qué regla se aplicó (nombre + alias + descripción general) y qué expresión se obtuvo, agregando una frase de cierre si ‘paso.esConclusion‘ es verdadero. * Se separó el renderizado del AST a texto legible (‘(P ENTONCES Q)‘, ‘(NO P)‘, etc.) en una función auxiliar recursiva (‘renderizarNodo‘), reutilizable en cualquier parte que necesite mostrar una expresión, no solo dentro de una explicación. * Se agregó ‘generarPasoEnriquecido‘, que devuelve los mismos datos pero "desarmados.^{en} campos separados (regla, alias, expresión resultante, descripción, esConclusion), por si la UI necesita mostrarlos en columnas en vez de un párrafo corrido. * **Integración Trazable:** * Se creó ‘index.ts‘ como punto de entrada único del módulo. * ‘traducirHistorial(premisas, resultado)‘ recorre ‘resultado.pasos‘ **en orden** (no en paralelo), manteniendo un acumulador de pasos ya procesados — necesario porque un paso puede depender de una línea generada por un paso anterior, no solo de las premisas originales. * Cada paso traducido conserva su número de línea calculado (‘premisas.length + índice + 1‘) y su ‘lineasInvolucradas‘ original, para poder cruzarlo 1 a 1 con el historial crudo de Arom si Morocho o Alex lo necesitan. * ‘explicarDemostracion(premisas, resultado)‘ envuelve lo anterior y agrega una conclusión final en lenguaje natural según ‘resultado.esValido‘. * Se validó la integración corriendo ‘ejemplo.ts‘ contra el ‘demostrarConclusion‘ real de Arom (mismo caso que su ‘solver.test.ts‘, Modus Ponendo Ponens con P ENTONCES Q y P), confirmando compilación en modo estricto (0 errores) y salida correcta en español.

»”’## Integrante: Renato **Módulo:** Validaciones, Sanitización, Pruebas y Cobertura (Control de Calidad)

»”’### Completado (17 de Agosto) * **Matriz de Casos Borde (‘casos_de_prueba.md‘):** * Se redactó una matriz exhaustiva de 25 casos de prueba y escenarios límite categorizados (entradas vacías, caracteres prohibidos/especiales, emojis, inyecciones de código/HTML, operadores matemáticos y lógicos en múltiples notaciones, variables compuestas con índices, paréntesis anidados, desbalanceados o vacíos, operadores consecutivos e incompletos, premisas redundantes, etc.). * **Módulo de Sanitización y Validación (‘src/lib/v

»”’# Avances del Sprint 2 - Equipo: Hijos de Linus

»”’## Integrante: Arom **Módulo:** Motor Lógico (Solver)

»”’### Completado (15 de Agosto) * **Definición de Tipos y AST:** * Se establecieron los operadores lógicos en español (Y, O, O_EXCLUSIVA, NO, ENTONCES, SI_Y_SOLO_SI, NI, INCOMPATIBLE). * Se definieron los nodos del Árbol de Sintaxis Abstracta (AST) en ‘src/lib/solver/types.ts‘. * Se crearon los tipos ‘Inferencia‘ y ‘Equivalencia‘ para representar las reglas lógicas (Modus Ponens, Modus Tollens, Silogismo, De Morgan, etc.) y se mapearon sus alias populares. * **Base del Evaluador (‘solver.ts‘):** * Se creó la función principal ‘demostrarConclusion‘ que recibe un arreglo de premisas y la conclusión deseada. * Se implementó el primer evaluador ‘aplicarModus-

PonendoPonens‘ como prueba de concepto para el "pattern matching". * Se creó la función ‘sonNodosIguales‘ para verificar la similitud estructural de dos ramas lógicas (esencial para evaluar equivalencias y silogismos). * **Pruebas y Verificación:** * Se creó la suite de pruebas automatizadas en ‘src/lib/solver/solver.test.ts‘. * El código base cumple estrictamente el tipado estricto (0 errores en ‘vue-tsc‘) y 100 % de cobertura en las pruebas iniciales de Vitest. * **Analizador Sintáctico (Parser) (Paso 2):** * Se implementó el analizador léxico (‘tokenizar‘) para separar correctamente variables, operadores y paréntesis respetando espacios. * Se construyó el algoritmo de parseo (‘construirAST‘) mediante un analizador de descenso recursivo, respetando firmemente la jerarquía y precedencia de operadores lógicos (donde "NO"tiene mayor peso que "Y", seguido de .º", y por último las condicionales/bicondicionales). * Se superaron las pruebas unitarias exhaustivas para expresiones anidadas, precedencia correcta y validación de errores sintácticos (‘parser.test.ts‘).

»"’### [nota] Pendiente y Notas para el Futuro * **Expansión de Reglas (Módulo de Arom):** Seguir agregando las funciones unitarias para detectar y aplicar ‘MODUS_TOLLENDO_TOLLENS‘, ‘SILOGISMO_HIPOTETICO‘, ‘DE_MORGAN‘, etc., basándose en la plantilla estructural de ‘aplicarModusPonendoPonens‘. * **Algoritmo de Búsqueda:** Mejorar el interior de ‘demostrarConclusion‘ para que recorra cíclicamente el arreglo de premisas intentando derivar proposiciones nuevas usando encadenamiento hacia adelante (forward-chaining) hasta dar con la conclusión deseada.

»"’## Integrantes: Morocho y Alex **Módulo:** Trazabilidad y Explicación (Didáctico)

»"’### Completado (16 de Agosto) * **Diseño de Estructura de Paso (‘types.ts‘):** * Se definió la interfaz ‘PasoTrazabilidad‘ con los campos: ‘numeroPaso‘, ‘operacion‘, ‘regla‘, ‘alias‘, ‘explicacion‘, ‘expresionSimbolica‘, ‘lineasBase‘, ‘esConclusion‘ y ‘estadoActual‘. * Se definió ‘ResultadoTrazabilidad‘ como contenedor completo con ‘esValido‘, ‘pasos‘, ‘conclusion‘ y ‘totalPasos‘, listo para consumir por la UI en el próximo sprint. * **Contenedor del Historial (‘historial.ts‘):** * Se creó ‘crearHistorial()‘ que retorna un array vacío ‘[]‘ como punto de partida. * Se implementó ‘registrarPaso()‘ que recibe un paso del solver, lo traduce a ‘PasoTrazabilidad‘ usando ‘generarPasoEnriquecido‘ de Mio, genera un snapshot del ‘estadoActual‘ con todas las líneas disponibles, y lo añade al array con ‘.push()‘. * Se implementaron funciones auxiliares ‘obtenerPaso(numero)‘ y ‘obtenerConclusiones()‘ para consultas sobre el historial. * **Integración con el Solver (‘construirTrazabilidad()‘):** * Se creó la función principal que recibe ‘premisas[]‘ + ‘ResultadoDemostracion‘ del solver y produce un ‘ResultadoTrazabilidad‘ completo. * Recorre los pasos EN ORDEN acumulando ‘pasosPrevios‘ para resolver referencias cruzadas de líneas, consumiendo el módulo de transcripción de Mio (‘generarPasoEnriquecido‘, ‘renderizarNodo‘). * Genera la conclusión final en lenguaje natural según ‘resultado.esValido‘. * **Punto de Entrada (‘index.ts‘):** * Se creó barrel file que re-exporta tipos

»"’12.7. Archivo: TAREAS/03_UI/guia.md

»"’# TAREA 03: Interfaz Visual (Vue 3)

»"’**Objetivo:** Conectar tu motor del Sprint 2 con una interfaz interactiva. **Fecha límite:** Viernes 29/08, 4-6 PM (PR)

»"’—

»"’## Recursos y Manuales

»"’| Recurso | Link | Para qué sirve | |—|—|—| | Vue 3 Composition API | https://vuejs.org/guide/extras/comp api-faq.html | Entender ‘<script setup>‘ | | ‘<script setup>‘ | https://vuejs.org/api/sfc-script-setup.html | Sintaxis de componentes | | Tailwind CSS v4 | https://tailwindcss.com/docs | Clases de diseño | | Componentes Vue | https://vuejs.org/guide/components/registration.html | Crear y usar componentes | | Button.vue (ref.) | ‘src/components/ui/Button.vue‘ | Componente de botón con branding | | Card.vue (ref.) | ‘src/components/ui/Card.vue‘ | Componente de tarjeta | | Badge.vue (ref.) | ‘src/components/ui/Badge.vue‘ | Componente de etiqueta |

»"’—

»"’## [arte] Branding — Vibe Duolingo (Azul)

»"’| Elemento | Estilo | |—|—| | Color primario | ‘blue-600‘ (#2563eb) — botones, acentos | | Fondo | ‘neutral-50‘ — claro, limpio | | Texto | ‘neutral-900‘ — alto contraste | | Bordes tarjetas | ‘rounded-xl‘ | | Bordes badges | ‘rounded-full‘ | | Sombras | ‘shadow-sm‘ → ‘hover:shadow-md‘ |

| Cards | fondo blanco, borde ‘neutral-200‘, hover ‘border-blue-400‘ | | Botones | ‘Button.vue‘ con variant primary/secondary | | Font | system-ui (ya configurado) |

»”’—

»”’## Por Equipo — Qué Crear

»”’### Equipo 1 — Sinergia (Tablas)

»”’**Modificar:** ‘src/pages/tablas/index.vue‘

»”’**Componentes a crear:** - Input para fórmula lógica (ej: ‘p Y q ->r‘) - Botón "Generar Tabla Tabla de verdad renderizada - Indicador: Tautología / Contradicción / Contingencia [!]

»”’**Conectar:** “‘typescript import { evaluar, generarTabla } from ’@/lib/truth-table/evaluator’ “‘

»”’—

»”’### Equipo 2 — Hijos de Linus (Inferencias)

»”’**Modificar:** ‘src/pages/inferencias/index.vue‘

»”’**Componentes a crear:** - Input de premisas (una por línea) - Input de conclusión - Botón "Demostrar Panel de pasos paso a paso (trazabilidad) - Indicador: Válido / Inválido

»”’**Conectar:** “‘typescript import { demostrarConclusion } from ’@/lib/solver/solver’ import { construirTrazabilidad } from ’@/lib/trazabilidad’ “‘

»”’—

»”’### Equipo 3 — Modus Innova (Cuantificadores)

»”’**Modificar:** ‘src/pages/cuantificadores/index.vue‘

»”’**Componentes a crear:** - Input de dominio (ej: ‘1, 2, 3, 4, 5‘) - Input de predicado (ej: ‘x es par‘) - Input de fórmula (ej: ‘∀x P(x)‘) - Botón .Evaluar Trazabilidad elemento por elemento - Botón "Negar Expresión"(De Morgan) - Indicador: Verdadero [verde] / Falso [rojo]

»”’**Conectar:** “‘typescript import { evaluarCuantificador, negarCuantificador } from ’@/lib/quantifiers/ev “‘

»”’—

»”’### Equipo 4 — Linus (Conjuntos)

»”’**Modificar:** ‘src/pages/conjuntos/index.vue‘

»”’**Componentes a crear:** - Input de Universo U - Input de conjuntos A, B (y opcionalmente C) - Selector de operación (Unión, Intersección, Diferencia, Complemento, Potencia) - Diagrama de Venn (SVG interactivo) - Resultado de la operación - Verificación de propiedades (subconjunto, disjuntos, igualdad)

»”’**Conectar:** “‘typescript import { union, interseccion, diferencia, potencia } from ’@/lib/sets/operations’ “‘

»”’—

»”’## [tool] Instrucciones de Entrega

»”’1. Modificar ‘src/pages/{tu-equipo}/index.vue‘ 2. Crear carpeta ‘TAREAS/03_UI/{tu-equipo}/‘ 3. Crear ‘avance.md‘ con: - Screenshots de la interfaz - Qué falta - Cómo probar tu módulo 4. PR antes del viernes 4-6 PM

»”’—

»”’## Checklist

»”’- [ ] ‘<script setup lang="ts»‘ (no Options API) - [ ] Conectado al motor de ‘src/lib/‘ - [ ] Responsive (mobile friendly) - [ ] Sigue el branding azul/Duolingo - [ ] ‘npm run type-check‘ sin errores - [ ] Botones usan ‘Button.vue‘ de ‘src/components/ui/‘

»”’# TAREA 03: Interfaz Visual (Vue 3)

»”’**Objetivo:** Conectar tu motor del Sprint 2 con una interfaz interactiva. **Fecha límite:** Viernes 29/08, 4-6 PM (PR)

»”’—

»”’## Recursos y Manuales

»”’| Recurso | Link | Para qué sirve | |—|—|—| | Vue 3 Composition API | https://vuejs.org/guide/extras/comp api-faq.html | Entender ‘<script setup>‘ | | ‘<script setup>‘ | https://vuejs.org/api/sfc-script-setup.html | Sintaxis de componentes | | Tailwind CSS v4 | https://tailwindcss.com/docs | Clases de diseño | | Componentes Vue | https://vuejs.org/guide/components/registration.html | Crear y usar

componentes | | Button.vue (ref.) | ‘src/components/ui/Button.vue‘ | Componente de botón con branding | | Card.vue (ref.) | ‘src/components/ui/Card.vue‘ | Componente de tarjeta | | Badge.vue (ref.) | ‘src/components/ui/Badge.vue‘ | Componente de etiqueta |

»”’—

»”’## [arte] Branding — Vibe Duolingo (Azul)

»”’| Elemento | Estilo | |—|—| | Color primario | ‘blue-600‘ (#2563eb) — botones, acentos | | Fondo | ‘neutral-50‘ — claro, limpio | | Texto | ‘neutral-900‘ — alto contraste | | Bordes tarjetas | ‘rounded-xl‘ | | Bordes badges | ‘rounded-full‘ | | Sombras | ‘shadow-sm‘ → ‘hover:shadow-md‘ | | Cards | fondo blanco, borde ‘neutral-200‘, hover ‘border-blue-400‘ | | Botones | ‘Button.vue‘ con variant primary/secondary | | Font | system-ui (ya configurado) |

»”’—

»”’## Por Equipo — Qué Crear

»”’### Equipo 1 — Sinergia (Tablas)

»”’**Modificar:** ‘src/pages/tablas/index.vue‘

»”’**Componentes a crear:** - Input para fórmula lógica (ej: ‘p Y q ->r‘) - Botón "Generar Tabla Tabla de verdad renderizada - Indicador: Tautología / Contradicción / Contingencia [!]

»”’**Conectar:** “‘typescript import { evaluar, generarTabla } from ’@/lib/truth-table/evaluator’ “‘

»”’—

»”’### Equipo 2 — Hijos de Linus (Inferencias)

»”’**Modificar:** ‘src/pages/inferencias/index.vue‘

»”’**Componentes a crear:** - Input de premisas (una por línea) - Input de conclusión - Botón "Demostrar Panel de pasos paso a paso (trazabilidad) - Indicador: Válido / Inválido

»”’**Conectar:** “‘typescript import { demostrarConclusion } from ’@/lib/solver/solver’ import { construirTrazabilidad } from ’@/lib/trazabilidad’ “‘

»”’—

»”’### Equipo 3 — Modus Innova (Cuantificadores)

»”’**Modificar:** ‘src/pages/cuantificadores/index.vue‘

»”’**Componentes a crear:** - Input de dominio (ej: ‘1, 2, 3, 4, 5‘) - Input de predicado (ej: ‘x es par‘) - Input de fórmula (ej: ‘∀x P(x)‘) - Botón .Evaluar Trazabilidad elemento por elemento - Botón "Negar Expresión"(De Morgan) - Indicador: Verdadero [verde] / Falso [rojo]

»”’**Conectar:** “‘typescript import { evaluarCuantificador, negarCuantificador } from ’@/lib/quantifiers/ev “‘

»”’—

»”’### Equipo 4 — Linus (Conjuntos)

»”’**Modificar:** ‘src/pages/conjuntos/index.vue‘

»”’**Componentes a crear:** - Input de Universo U - Input de conjuntos A, B (y opcionalmente C) - Selector de operación (Unión, Intersección, Diferencia, Complemento, Potencia) - Diagrama de Venn (SVG interactivo) - Resultado de la operación - Verificación de propiedades (subconjunto, disjuntos, igualdad)

»”’**Conectar:** “‘typescript import { union, interseccion, diferencia, potencia } from ’@/lib/sets/operations’ “‘

»”’—

»”’## [tool] Instrucciones de Entrega

»”’1. Modificar ‘src/pages/{tu-equipo}/index.vue‘ 2. Crear carpeta ‘TAREAS/03_UI/{tu-equipo}/‘ 3. Crear ‘avance.md‘ con: - Screenshots de la interfaz - Qué falta - Cómo probar tu módulo 4. PR antes del viernes 4-6 PM

»”’—

»”’## Checklist

»”’- [ ] ‘<script setup lang="ts»‘ (no Options API) - [ ] Conectado al motor de ‘src/lib/‘ - [ ] Responsive (mobile friendly) - [ ] Sigue el branding azul/Duolingo - [ ] ‘npm run type-check‘ sin errores - [ ] Botones usan ‘Button.vue‘ de ‘src/components/ui/‘

»"'12.8. Archivo: TAREAS/03_UI/hijos-de-linus/avance.md

»"'# Registro de Avances — Los Hijos de Linus

»"'## SPRINT 3.5 (Completado)

»"'### Alex - 2026-08-23 - **Tareas Completadas:** - Actualización de 'IndicadorResultado.vue' para mostrar de forma reactiva los motivos y diagnósticos de falacias ('errorLogico') en el estado inválido y error. - Implementación de acordeón colapsable en 'PanelTrazabilidad.vue' ('£Por qué esta regla¿ / 'Ocultar explicación') para ver las explicaciones didácticas de cada deducción formal. - Creación de pruebas unitarias adicionales en 'IndicadorResultado.test.ts' y 'PanelTrazabilidad.test.ts'. - **Checklist Manual Ejecutado (QA Humano):** - [x] Navegación por teclado probada en los botones del acordeón. - [x] Contrastes y transiciones suaves de apertura/cierre.

»"'### Mio - 2026-08-23 - **Tareas Completadas:** - Creación y verificación de 'TraductorLenguajeNatural.vue', permitiendo asignar texto a variables proposicionales y generando el argumento continuo. - Integración de sistema de pestañas en 'index.vue' ('[[teclado] Simbología Formal]' | '[[libro] Lenguaje Natural]'), preservando la estructura balanceada en 2 columnas sin romper la altura visual. - Sincronización en tiempo real entre el formulario y el traductor. - Pruebas unitarias en 'TraductorLenguajeNatural.test.ts' e integración de pestañas en 'index.test.ts'. - **Checklist Manual Ejecutado (QA Humano):** - [x] Responsividad de pestañas en móviles y escritorio. - [x] Sincronización de variables reactivas al cambiar inputs.

»"'### Morocho - 2026-08-23 - **Tareas Completadas:** - Definición temprana de tipos en 'src/lib/solver/types.ts' ('MotivoInvalidez', 'ErrorLogico', 'ResultadoDemostracion') exportados en 'src/types/inferencias.ts'. - Implementación de 'detectarErrorLogico()' en 'src/lib/solver/solver.ts' con *pattern matching* para diagnosticar falacias y motivos de fallo lógico. - Creación de 3 pruebas unitarias adicionales en 'src/lib/solver/solver.test.ts' (15/15 tests pasando).

»"'—

»"'## [paquete] SPRINT 3 (Completado)

»"'### Arom - 2026-08-23 - **Tareas Completadas:** - Fase 3 completada: Integración completa de 'index.vue' conectando 'FormularioInferencia', 'IndicadorResultado' y 'PanelTrazabilidad' con el motor lógico. - Implementación de Teclado Lógico Simbólico ('¬', '∧', '∨', '→', '↔', '(', ')'). - Corrección de base URL en 'src/router/index.ts' para despliegues.

»"'### Rennato - 2026-08-23 - **Tareas Completadas:** - Fase 4 completada: Setup de '@vue/test-utils' y 'happy-dom' en 'vite.config.ts'. - Creación de suites de prueba unitarias e integración de la página.

»"'### Alex - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Indicador de Resultados y Feedback Visual) completada.

»"'### Mio - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Panel de Trazabilidad y Explicación Visual).

»"'### Morocho - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Formulario de Inferencia).

»"'# Registro de Avances — Los Hijos de Linus

»"'## SPRINT 3.5 (Completado)

»"'### Alex - 2026-08-23 - **Tareas Completadas:** - Actualización de 'IndicadorResultado.vue' para mostrar de forma reactiva los motivos y diagnósticos de falacias ('errorLogico') en el estado inválido y error. - Implementación de acordeón colapsable en 'PanelTrazabilidad.vue' ('£Por qué esta regla¿ / 'Ocultar explicación') para ver las explicaciones didácticas de cada deducción formal. - Creación de pruebas unitarias adicionales en 'IndicadorResultado.test.ts' y 'PanelTrazabilidad.test.ts'. - **Checklist Manual Ejecutado (QA Humano):** - [x] Navegación por teclado probada en los botones del acordeón. - [x] Contrastes y transiciones suaves de apertura/cierre.

»"'### Mio - 2026-08-23 - **Tareas Completadas:** - Creación y verificación de 'TraductorLenguajeNatural.vue', permitiendo asignar texto a variables proposicionales y generando el argumento continuo. - Integración de sistema de pestañas en 'index.vue' ('[[teclado] Simbología Formal]' | '[[libro] Lenguaje Natural]'), preservando la estructura balanceada en 2 columnas sin romper la altura visual. - Sincronización en tiempo real entre el formulario y el traductor. - Pruebas unitarias en 'TraductorLenguajeNatural.test.ts' e integración de pestañas en 'index.test.ts'. - **Checklist

Manual Ejecutado (QA Humano):** - [x] Responsividad de pestañas en móviles y escritorio. - [x] Sincronización de variables reactivas al cambiar inputs.

»"'### Morocho - 2026-08-23 - **Tareas Completadas:** - Definición temprana de tipos en 'src/lib/solver/types.ts' ('MotivoInvalidez', 'ErrorLogico', 'ResultadoDemostracion') exportados en 'src/types/inferencias.ts'. - Implementación de 'detectarErrorLogico()' en 'src/lib/solver/solver.ts' con *pattern matching* para diagnosticar falacias y motivos de fallo lógico. - Creación de 3 pruebas unitarias adicionales en 'src/lib/solver/solver.test.ts' (15/15 tests pasando).

»"'—

»"'## [paquete] SPRINT 3 (Completado)

»"'### Arom - 2026-08-23 - **Tareas Completadas:** - Fase 3 completada: Integración completa de 'index.vue' conectando 'FormularioInferencia', 'IndicadorResultado' y 'PanelTrazabilidad' con el motor lógico. - Implementación de Teclado Lógico Simbólico ('¬', '∧', '∨', '→', '↔', '(', ')'). - Corrección de base URL en 'src/router/index.ts' para despliegues.

»"'### Rennato - 2026-08-23 - **Tareas Completadas:** - Fase 4 completada: Setup de '@vue/test-utils' y 'happy-dom' en 'vite.config.ts'. - Creación de suites de prueba unitarias e integración de la página.

»"'### Alex - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Indicador de Resultados y Feedback Visual) completada.

»"'### Mio - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Panel de Trazabilidad y Explicación Visual).

»"'### Morocho - 2026-08-22 - **Tareas Completadas:** - Fase 2 (Formulario de Inferencia).

»"'12.9. Archivo: TAREAS/04_Deploy/guia.md

»"'# TAREA 04: Pruebas y Despliegue

»"'**Objetivo:** Integración final, pruebas y deploy en GitHub Pages. **Fecha límite:** Sábado 30/08 (FECHA LÍMITE ABSOLUTA)

»"'—

»"'## Recursos y Manuales

»"'| Recurso | Link | Para qué sirve | |—|—|—| | GitHub Pages | https://docs.github.com/en/pages | Entender el deploy | | Vite Build | https://vitejs.dev/guide/build.html | Configurar build para producción | | Vitest | https://vitest.dev/guide/ | Escribir tests | | Vue Test Utils | https://test-utils.vuejs.org/ | Tests de componentes |

»"'—

»"'## Responsabilidades

»"'### Cada Equipo - Verificar que su módulo funciona en la app completa - Escribir al menos 1 test básico por función principal - Documentar uso de su módulo en 'TAREAS/04_Deploy/{equipo}/uso.md'

»"'### Líder (Enri) - Configurar 'vite.config.ts' con 'base' - Crear '.github/workflows/deploy.yml' - Merge final a 'main' - Verificar deploy en GitHub Pages

»"'—

»"'## [tool] Deploy — GitHub Pages

»"'### Paso 1: vite.config.ts (Ya configurado) "'ts export default defineConfig({ base: '/logica_simbolica_global/', // ... }) "'

»"'### Paso 2: .github/workflows/deploy.yml (Ya creado) El workflow automáticamente: 1. Instala dependencias 2. Ejecuta 'npm run build' 3. Sube a GitHub Pages

»"'### Paso 3: Activar GitHub Pages 1. Ve a Settings → Pages 2. Source: **GitHub Actions** 3. El deploy ocurrirá automáticamente al hacer push a 'main'

»"'—

»"'## [clip] Por Equipo — Qué Verificar

»"'### Sinergia (Tablas) - [ ] Genera tablas de verdad correctas - [ ] Identifica tautologías/contradicciones - [ ] Test: 'evaluar()' con al menos 3 fórmulas

»"'### Hijos de Linus (Inferencias) - [ ] Demuestra conclusión con MPP - [ ] Muestra pasos de trazabilidad - [ ] Test: 'demostrarConclusion()' con al menos 2 argumentos

»"'### Modus Innova (Cuantificadores) - [ ] Evalúa ∀ y ∃ en dominio finito - [ ] Niega cuantificadores correctamente - [ ] Test: 'evaluarCuantificador()' con al menos 2 fórmulas

»"'### Linus (Conjuntos) - [ ] Realiza unión, intersección, diferencia - [ ] Genera conjunto potencia - [ ] Test: al menos 3 operaciones con resultados correctos

»"'—

»"'## [nota] Documento de Uso

»"'Crear 'TAREAS/04_Deploy/{equipo}/uso.md' con:

»"'"'markdown # Uso del Módulo: [Nombre del Tema]

»"'## Cómo funciona [Descripción breve]

»"'## Ejemplos 1. [Ejemplo 1 con entrada y salida esperada] 2. [Ejemplo 2]

»"'## Limitaciones - [Qué no hace tu módulo] "'

»"'—

»"'## Checklist Final

»"'- [ ] 'npm run build' produce 'dist/' sin errores - [ ] 'npm run test' pasa - [ ] App visible en GitHub Pages - [ ] Todos los módulos navegables desde Home - [ ] README actualizado con link al deploy - [ ] Documento de uso creado

»"'# TAREA 04: Pruebas y Despliegue

»"'**Objetivo:** Integración final, pruebas y deploy en GitHub Pages. **Fecha límite:** Sábado 30/08 (FECHA LÍMITE ABSOLUTA)

»"'—

»"'## Recursos y Manuales

»"'| Recurso | Link | Para qué sirve | |—|—|—| | GitHub Pages | https://docs.github.com/en/pages | Entender el deploy | | Vite Build | https://vitejs.dev/guide/build.html | Configurar build para producción | | Vitest | https://vitest.dev/guide/ | Escribir tests | | Vue Test Utils | https://test-utils.vuejs.org/ | Tests de componentes |

»"'—

»"'## Responsabilidades

»"'### Cada Equipo - Verificar que su módulo funciona en la app completa - Escribir al menos 1 test básico por función principal - Documentar uso de su módulo en 'TAREAS/04_Deploy/{equipo}/uso.md'

»"'### Líder (Enri) - Configurar 'vite.config.ts' con 'base' - Crear '.github/workflows/deploy.yml' - Merge final a 'main' - Verificar deploy en GitHub Pages

»"'—

»"'## [tool] Deploy — GitHub Pages

»"'### Paso 1: vite.config.ts (Ya configurado) "'ts export default defineConfig({ base: '/logica_simbolica_global/', // ... }) "'

»"'### Paso 2: .github/workflows/deploy.yml (Ya creado) El workflow automáticamente: 1. Instala dependencias 2. Ejecuta 'npm run build' 3. Sube a GitHub Pages

»"'### Paso 3: Activar GitHub Pages 1. Ve a Settings → Pages 2. Source: **GitHub Actions** 3. El deploy ocurrirá automáticamente al hacer push a 'main'

»"'—

»"'## [clip] Por Equipo — Qué Verificar

»"'### Sinergia (Tablas) - [ ] Genera tablas de verdad correctas - [ ] Identifica tautologías/contradicciones - [ ] Test: 'evaluar()' con al menos 3 fórmulas

»"'### Hijos de Linus (Inferencias) - [ ] Demuestra conclusión con MPP - [ ] Muestra pasos de trazabilidad - [ ] Test: 'demostrarConclusion()' con al menos 2 argumentos

»"'### Modus Innova (Cuantificadores) - [ ] Evalúa ∀ y ∃ en dominio finito - [ ] Niega cuantificadores correctamente - [ ] Test: 'evaluarCuantificador()' con al menos 2 fórmulas

»"'### Linus (Conjuntos) - [ ] Realiza unión, intersección, diferencia - [ ] Genera conjunto potencia - [ ] Test: al menos 3 operaciones con resultados correctos

»"'—

»"'## [nota] Documento de Uso

»"'Crear 'TAREAS/04_Deploy/{equipo}/uso.md' con:

»"'"'markdown # Uso del Módulo: [Nombre del Tema]

»"'## Cómo funciona [Descripción breve]

»"'## Ejemplos 1. [Ejemplo 1 con entrada y salida esperada] 2. [Ejemplo 2]

»"'## Limitaciones - [Qué no hace tu módulo] "'

»"'—

»"'## Checklist Final

»"'- [ ] 'npm run build' produce 'dist/' sin errores - [ ] 'npm run test' pasa - [ ] App visible en GitHub Pages - [ ] Todos los módulos navegables desde Home - [ ] README actualizado con link al deploy - [ ] Documento de uso creado

»"'12.10. Archivo: TAREAS/DELEGACION_P1.md

»"'# Delegación P1 — Landing Page (Home + Navbar + Contributors)

»"'**Delegado:** Morocho **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Dar a conocer el valor de la plataforma para poder publicarla: una landing que comunique **50 % el valor de la plataforma** (qué es, por qué importa, para quién) y **50 % "nosotros"** (los equipos, las personas, el docente y la universidad), todo coherente con el branding LogiLearn.

»"'## 2. Qué se espera (resultado, no proceso) - Una 'Home.vue' que funcione como landing profesional y publicable. - Agregar el link **Ïnicio"** al navbar ('AppNavBar.vue') apuntando a '/'. - Crear **'CONTRIBUTORS.md'** en la raíz con la lista completa de participantes.

»"'## 3. Criterios de aceptación (medibles) - [ ] 'Home.vue' navegable, legible y alineado al branding. - [ ] Secciones presentes: hero de valor + "Sobre Nosotros"(universidad, curso, docente, equipos, participantes). - [ ] Navbar incluye Ïnicio". - [ ] 'CONTRIBUTORS.md' existe y lista a los 4 equipos (incl. Mauricio en Hijos de Linus y Julio en Modus Innova). - [ ] 'npm run build' pasa sin errores. - [ ] Aprobación visual de Enrique.

»"'## 4. Restricciones - Branding LogiLearn: navy '#0F2D8D' (navbar), azul-600 '#2563EB' (acciones), tarjetas blancas 'border-neutral-200' 'rounded-xl' 'shadow-sm'. - No romper el build ni los **175 tests** existentes. - No usar Ïniciar sesión/ Çerrar sesión.en el navbar.

»"'## 5. Entregables (archivos) - 'src/pages/Home.vue' (reescritura) - 'src/components/layout/AppNavBar.v (agregar Ïnicio") - 'CONTRIBUTORS.md' (nuevo)

»"'## 6. Recursos que Enrique dejó listos - **Branding kit:** navy '#0F2D8D', azul-600, estilo rounded-xl/shadow-sm. - **Datos del sílabo:** Universidad Nacional Pedro Ruiz Gallo ů Escuela Prof. de Ingeniería de Sistemas ů Curso **Lógica Simbólica** (código **MATG1001**, Semestre **2026 I**, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera** (mgonzalesh@unprg.edu.pe). - **Lista de participantes** (ver abajo). - Rama de trabajo: 'feat/integrate-sinergia-modus' (PR #16).

»"'### Lista oficial de participantes - **Líder de proyecto:** Enrique (EnriDiazCayaca) - **Equipo Sinergia** (sublíder Alexa): Aldair, Smith, Miguel Velarde, Jesús Núñez - **Equipo Los Hijos de Linus** (sublíder Arom): Centurión, Morocho, Altamirano, Mio, **Mauricio** - **Equipo Modus Innova** (sublíder Cristian): Danuska, Marlon, Guillermo, Noemí, **Julio** - **Equipo Linus** (sublíder Jordy): Nio, Mike, Sergio, Fer, Alejandro - **Docente:** Dr. Mardo Victor Gonzales Herrera

»"'# Delegación P1 — Landing Page (Home + Navbar + Contributors)

»"'**Delegado:** Morocho **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Dar a conocer el valor de la plataforma para poder publicarla: una landing que comunique **50 % el valor de la plataforma** (qué es, por qué importa, para quién) y **50 % "nosotros"** (los equipos, las personas, el docente y la universidad), todo coherente con el branding LogiLearn.

»"'## 2. Qué se espera (resultado, no proceso) - Una 'Home.vue' que funcione como landing profesional y publicable. - Agregar el link **Ïnicio"** al navbar ('AppNavBar.vue') apuntando a '/'. - Crear **'CONTRIBUTORS.md'** en la raíz con la lista completa de participantes.

»"'## 3. Criterios de aceptación (medibles) - [ ] 'Home.vue' navegable, legible y alineado al branding. - [ ] Secciones presentes: hero de valor + "Sobre Nosotros"(universidad, curso, docente, equipos, participantes). - [ ] Navbar incluye Ïnicio". - [ ] 'CONTRIBUTORS.md' existe y lista a los 4 equipos (incl. Mauricio en Hijos de Linus y Julio en Modus Innova). - [ ] 'npm run build' pasa sin errores. - [ ] Aprobación visual de Enrique.

»"'## 4. Restricciones - Branding LogiLearn: navy '#0F2D8D' (navbar), azul-600 '#2563EB' (acciones), tarjetas blancas 'border-neutral-200' 'rounded-xl' 'shadow-sm'. - No romper el build ni los **175 tests** existentes. - No usar Ïniciar sesión/ Çerrar sesión.[en] el navbar.

»"'## 5. Entregables (archivos) - 'src/pages/Home.vue' (reescritura) - 'src/components/layout/AppNavBar.v (agregar Ïnicio") - 'CONTRIBUTORS.md' (nuevo)

»"'## 6. Recursos que Enrique dejó listos - **Branding kit:** navy '#0F2D8D', azul-600, estilo rounded-xl/shadow-sm. - **Datos del sílabo:** Universidad Nacional Pedro Ruiz Gallo ů Escuela Prof. de Ingeniería de Sistemas ů Curso **Lógica Simbólica** (código **MATG1001**, Semestre **2026 I**, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera** (mgonzalesh@unprg.edu.pe). - **Lista de participantes** (ver abajo). - Rama de trabajo: 'feat/integrate-sinergia-modus' (PR #16).

»"'### Lista oficial de participantes - **Líder de proyecto:** Enrique (EnriDiazCayaca) - **Equipo Sinergia** (sublíder Alexa): Aldair, Smith, Miguel Velarde, Jesús Núñez - **Equipo Los Hijos de Linus** (sublíder Arom): Centurión, Morocho, Altamirano, Mio, **Mauricio** - **Equipo Modus Innova** (sublíder Cristian): Danuska, Marlon, Guillermo, Noemí, **Julio** - **Equipo Linus** (sublíder Jordy): Nio, Mike, Sergio, Fer, Alejandro - **Docente:** Dr. Mardo Victor Gonzales Herrera

»"'12.11. Archivo: TAREAS/DELEGACION_P2.md

»"'# Delegación P2 — Informe de Investigación

»"'**Delegados:** Sergio y Jordy **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Producir un informe de investigación académico, de **calidad y extensión superiores al estándar**, que sea aprobado por la plataforma de similitud (Turnitin) y documente lo aprendido en el proyecto.

»"'## 2. Qué se espera (resultado, no proceso) - Un documento 'docs/INFORME_INVESTIGACION.md' en **formato APA 7, en español**. - Longitud y profundidad **mayores al estándar** (objetivo **18–22 páginas** equivalentes). - Contenido mínimo: - Título, autores (equipos + docente) y resumen (abstract). - Introducción (contexto del curso MATG1001 y resultados de aprendizaje D1/D2/D3 del sílabo). - Marco tecnológico: Vue 3, Vite 6, Tailwind CSS v4, TypeScript, Vitest, Vue Router. - Metodología: 4 equipos de dominio, ramificaciones Git, flujo de Pull Requests, CI (Netlify/GitHub Pages). - Arquitectura y desarrollo: motor de tablas de verdad, inferencias, cuantificadores (∀/∃, De Morgan, resolutor paso a paso), teoría de conjuntos. - **Manual de uso** de la plataforma (pasos por módulo). - Reflexión: **lo aprendido** en el proyecto (por equipo). - Conclusiones y referencias bibliográficas.

»"'## 3. Criterios de aceptación (medibles) - [ ] Formato APA 7 coherente (portada, encabezados, citas, referencias). - [ ] $\geq$ 18 páginas de contenido sustantivo (no relleno). - [ ] Citas y referencias presentes; redacción original (evita plagio detectable). - [ ] Cubre manual de uso y aprendizajes del sílabo (D1/D2/D3). - [ ] Aprobado por Enrique.

»"'## 4. Restricciones - Idioma: español. - No incluir código fuente extenso; describir arquitectura y decisiones. - Coherencia con los datos reales del proyecto (ver recursos).

»"'## 5. Entregables (archivos) - 'docs/INFORME_INVESTIGACION.md' (nuevo; considerar 'docs/README.md' como índice).

»"'## 6. Recursos que Enrique dejó listos - **Datos del sílabo:** UNPRG ů Escuela Prof. de Ing. de Sistemas ů Curso Lógica Simbólica (**MATG1001**, 2026 I, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera**. - **Resultados de aprendizaje del sílabo:** D1 (identifica/aplica lógica proposicional), D2 (interpreta/aplica inferencia lógica), D3 (discute/aplica teoría de conjuntos). - **Stack y estructura reales del repo** (Vue 3 + Vite + Tailwind v4 + TS + Vitest; carpetas ‘src/pages/*‘, ‘src/lib/*‘, ‘src/components/ui/*‘). - Lista de participantes (misma que P1).

»"'# Delegación P2 — Informe de Investigación

»"'**Delegados:** Sergio y Jordy **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Producir un informe de investigación académico, de **calidad y extensión superiores al estándar**, que sea aprobado por la plataforma de similitud (Turnitin) y documente lo aprendido en el proyecto.

»"'## 2. Qué se espera (resultado, no proceso) - Un documento ‘docs/INFORME_INVESTIGACION.md‘ en **formato APA 7, en español**. - Longitud y profundidad **mayores al estándar** (objetivo **18–22 páginas** equivalentes). - Contenido mínimo: - Título, autores (equipos + docente) y resumen (abstract). - Introducción (contexto del curso MATG1001 y resultados de aprendizaje D1/D2/D3 del sílabo). - Marco tecnológico: Vue 3, Vite 6, Tailwind CSS v4, TypeScript, Vitest, Vue Router. - Metodología: 4 equipos de dominio, ramificaciones Git, flujo de Pull Requests, CI (Netlify/GitHub Pages). - Arquitectura y desarrollo: motor de tablas de verdad, inferencias, cuantificadores (∀/∃, De Morgan, resolutor paso a paso), teoría de conjuntos. - **Manual de uso** de la plataforma (pasos por módulo). - Reflexión: **lo aprendido** en el proyecto (por equipo). - Conclusiones y referencias bibliográficas.

»"'## 3. Criterios de aceptación (medibles) - [ ] Formato APA 7 coherente (portada, encabezados, citas, referencias). - [ ] $\geq$ 18 páginas de contenido sustantivo (no relleno). - [ ] Citas y referencias presentes; redacción original (evita plagio detectable). - [ ] Cubre manual de uso y aprendizajes del sílabo (D1/D2/D3). - [ ] Aprobado por Enrique.

»"'## 4. Restricciones - Idioma: español. - No incluir código fuente extenso; describir arquitectura y decisiones. - Coherencia con los datos reales del proyecto (ver recursos).

»"'## 5. Entregables (archivos) - ‘docs/INFORME_INVESTIGACION.md‘ (nuevo; considerar ‘docs/README.md‘ como índice).

»"'## 6. Recursos que Enrique dejó listos - **Datos del sílabo:** UNPRG ů Escuela Prof. de Ing. de Sistemas ů Curso Lógica Simbólica (**MATG1001**, 2026 I, II Ciclo) ů Docente **Dr. Mardo Victor Gonzales Herrera**. - **Resultados de aprendizaje del sílabo:** D1 (identifica/aplica lógica proposicional), D2 (interpreta/aplica inferencia lógica), D3 (discute/aplica teoría de conjuntos). - **Stack y estructura reales del repo** (Vue 3 + Vite + Tailwind v4 + TS + Vitest; carpetas ‘src/pages/*‘, ‘src/lib/*‘, ‘src/components/ui/*‘). - Lista de participantes (misma que P1).

»"'12.12. Archivo: TAREAS/DELEGACION_P3.md

»"'# Delegación P3 — Depurar Repositorio

»"'**Delegado:** Alex **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Dejar el repositorio limpio, organizado y listo para publicación: README finalizado, sin archivosLegacy de migración, con ramas ordenadas.

»"'## 2. Qué se espera (resultado, no proceso) - ‘README.md‘ actualizado al estado **"finalizado"** (proyecto completado al 100 % del sílabo). - Eliminar carpetas/archivos **innecesarios**: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ (son legacy de la migración Next.js → Vue). - Limpiar **ramas locales obsoletas**: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘. - ‘.gitignore‘ actualizado para ignorar los legacy.

»"'>**Nota:** Enrique retiene la parte de **cerrar los PRs #8, #14 y #15** (no es tarea de Alex).

»"'## 3. Criterios de aceptación (medibles) - [ ] ‘README.md‘ refleja estado final, equipos, cómo correr y deploy. - [ ] Sin ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ en el árbol de trabajo. - [ ] Ramas locales obsoletas eliminadas. - [ ] ‘.gitignore‘ incluye los patrones legacy. - [ ] ‘npm run build‘ pasa; **175 tests** siguen pasando. - [ ] Aprobación de Enrique.

»"'## 4. Restricciones - No borrar ‘src/‘, ‘docs/‘, ‘TAREAS/‘, ‘public/‘, ni configuraciones de deploy (Netlify/GitHub Pages). - No tocar el trabajo ya integrado en ‘feat/integrate-sinergia-modus‘. - Mantener ‘.agents/‘ (config de herramienta) salvo indicación contraria de Enrique.

»"'## 5. Entregables (archivos) - ‘README.md‘ (reescritura a estado final) - Eliminación de: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ - Limpieza de ramas locales: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘ - ‘.gitignore‘ (actualización)

»"'## 6. Recursos que Enrique dejó listos - Confirmación de qué PRs cerrar (Enrique cierra #8, #14, #15; se mantiene #16). - Lista de carpetas legacy a borrar (arriba). - Rama de trabajo: ‘feat/integrate-sinergia-modus‘.

»"'# Delegación P3 — Depurar Repositorio

»"'**Delegado:** Alex **Líder que convoca:** Enrique (líder del proyecto) **Estado:** Por ejecutar

»"'—

»"'## 1. Propósito (visión compartida) Dejar el repositorio limpio, organizado y listo para publicación: README finalizado, sin archivosLegacy de migración, con ramas ordenadas.

»"'## 2. Qué se espera (resultado, no proceso) - ‘README.md‘ actualizado al estado **"finalizado"** (proyecto completado al 100 % del sílabo). - Eliminar carpetas/archivos **innecesarios**: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ (son legacy de la migración Next.js → Vue). - Limpiar **ramas locales obsoletas**: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘. - ‘.gitignore‘ actualizado para ignorar los legacy.

»"'>**Nota:** Enrique retiene la parte de **cerrar los PRs #8, #14 y #15** (no es tarea de Alex).

»"'## 3. Criterios de aceptación (medibles) - [ ] ‘README.md‘ refleja estado final, equipos, cómo correr y deploy. - [ ] Sin ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ en el árbol de trabajo. - [ ] Ramas locales obsoletas eliminadas. - [ ] ‘.gitignore‘ incluye los patrones legacy. - [ ] ‘npm run build‘ pasa; **175 tests** siguen pasando. - [ ] Aprobación de Enrique.

»"'## 4. Restricciones - No borrar ‘src/‘, ‘docs/‘, ‘TAREAS/‘, ‘public/‘, ni configuraciones de deploy (Netlify/GitHub Pages). - No tocar el trabajo ya integrado en ‘feat/integrate-sinergia-modus‘. - Mantener ‘.agents/‘ (config de herramienta) salvo indicación contraria de Enrique.

»"'## 5. Entregables (archivos) - ‘README.md‘ (reescritura a estado final) - Eliminación de: ‘.next/‘, ‘next-env.d.ts‘, ‘dist/‘, ‘skills-lock.json‘ - Limpieza de ramas locales: ‘pr/12‘, ‘pr/12-temp‘, ‘pr/14‘, ‘pr/15‘, ‘test-linus-prs‘ - ‘.gitignore‘ (actualización)

»"'## 6. Recursos que Enrique dejó listos - Confirmación de qué PRs cerrar (Enrique cierra #8, #14, #15; se mantiene #16). - Lista de carpetas legacy a borrar (arriba). - Rama de trabajo: ‘feat/integrate-sinergia-modus‘.

»"'12.13. Archivo: CRONOGRAMA.md

»"'# [cal] Cronograma (4 Sprints)

»"'Fecha de inicio: **3 de Agosto 2026** ů Fecha límite: **30 de Agosto 2026**

»"'—

»"'## Sprint 1 ů Propuesta y Diseño **Lun 4 - Jue 7 Ago** → Trabajar el entregable. **Vie 8 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 9-10 Ago** → Correcciones y ajustes. - **Entregable:** ‘propuesta.md‘ con descripción de la herramienta y boceto visual. - **Herramienta principal:** Figma / Excalidraw + Antigravity para ideas.

»"'## Sprint 2 ů Motor Lógico (TypeScript) **Lun 11 - Jue 14 Ago** → Trabajar el entregable. **Vie 15 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 16-17 Ago** → Correcciones y ajustes. - **Entregable:** Código ‘.ts‘ que procesa la lógica matemática del tema (sin interfaz aún).

- **Referencia Open Source:** Motor de [Keith Schwarz](https://web.stanford.edu/class/cs103/). - **Herramienta principal:** Antigravity (modo agente) para programar el motor.

»"'## Sprint 3 ů Interfaz Visual (Vue 3) **Lun 18 - Jue 21 Ago** → Trabajar el entregable. **Vie 22 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 23-24 Ago** → Correcciones y ajustes. - **Entregable:** Componentes '.vue' conectados al motor del Sprint 2. - **Herramienta principal:** Antigravity para generar y refactorizar componentes Vue.

»"'## Sprint 4 ů Pruebas y Despliegue **Lun 25 - Jue 28 Ago** → Trabajar el entregable. **Vie 29 Ago (4-6 PM)** → Revisión final y merge a 'main'. **Sáb 30 Ago** → [meta] **FECHA LÍMITE ABSOLUTA.** - **Entregable:** Código final sin bugs, integrado en 'main'. - **Herramienta principal:** Antigravity para debug y logs.

»"'# [cal] Cronograma (4 Sprints)

»"'Fecha de inicio: **3 de Agosto 2026** ů Fecha límite: **30 de Agosto 2026**

»"'—

»"'## Sprint 1 ů Propuesta y Diseño **Lun 4 - Jue 7 Ago** → Trabajar el entregable. **Vie 8 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 9-10 Ago** → Correcciones y ajustes. - **Entregable:** 'propuesta.md' con descripción de la herramienta y boceto visual. - **Herramienta principal:** Figma / Excalidraw + Antigravity para ideas.

»"'## Sprint 2 ů Motor Lógico (TypeScript) **Lun 11 - Jue 14 Ago** → Trabajar el entregable. **Vie 15 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 16-17 Ago** → Correcciones y ajustes. - **Entregable:** Código '.ts' que procesa la lógica matemática del tema (sin interfaz aún). - **Referencia Open Source:** Motor de [Keith Schwarz](https://web.stanford.edu/class/cs103/). - **Herramienta principal:** Antigravity (modo agente) para programar el motor.

»"'## Sprint 3 ů Interfaz Visual (Vue 3) **Lun 18 - Jue 21 Ago** → Trabajar el entregable. **Vie 22 Ago (4-6 PM)** → Revisión de PRs. **Sáb-Dom 23-24 Ago** → Correcciones y ajustes. - **Entregable:** Componentes '.vue' conectados al motor del Sprint 2. - **Herramienta principal:** Antigravity para generar y refactorizar componentes Vue.

»"'## Sprint 4 ů Pruebas y Despliegue **Lun 25 - Jue 28 Ago** → Trabajar el entregable. **Vie 29 Ago (4-6 PM)** → Revisión final y merge a 'main'. **Sáb 30 Ago** → [meta] **FECHA LÍMITE ABSOLUTA.** - **Entregable:** Código final sin bugs, integrado en 'main'. - **Herramienta principal:** Antigravity para debug y logs.

»"'12.14. Archivo: MAPA.md

»"'# [mapa] MAPA DEL REPOSITORIO

»"'Guía rápida para encontrar cualquier cosa en el proyecto.

»"'—

»"'## [carpeta] Raíz del Proyecto

»"'| Archivo | Para qué sirve | |—|—| | 'README.md' | Presentación del proyecto: equipos, reglas, rúbrica de evaluación. | | 'CRONOGRAMA.md' | Fechas exactas de los 4 sprints y qué entregar en cada uno. | | 'GUIA_ESTUDIANTE.md' | Manual de herramientas y paso a paso por entregable. | | 'CONTRIBUTING.md' | Reglas para hacer un buen Pull Request. | | 'MAPA.md' | Este archivo. Índice de todo el repositorio. |

»"'—

»"'## [carpeta] '/TAREAS'

»"'| Archivo | Para qué sirve | |—|—| | '01_Propuesta.md' | Instrucciones y plantilla para el Entregable 1 (Diseño). |

»"'*(Se irán añadiendo '02_Motor.md', '03_UI.md', '04_Deploy.md' según avancen los sprints.)*

»"'—

»"'## [carpeta] '/src/pages' — El Código

»"'Cada carpeta pertenece a un equipo. Ahí vive todo su código Vue.

»"'## [mapa] Mapa de Rutas (Páginas del Proyecto)

»"'La estructura web del proyecto se divide en 4 módulos interactivos, uno por equipo:

»"'1. **Equipo Sinergia** - ‘src/pages/tablas/‘ - Tema: *Fundamentos, Conectivos y Tablas de Verdad* 2. **Equipo Los hijos de Linus** - ‘src/pages/inferencias/‘ - Tema: *Inferencias Lógicas y Validaciones* 3. **Equipo 3** - ‘src/pages/cuantificadores/‘ - Tema: *Cuantificadores y Lógica de Predicados* 4. **Equipo Linus** - ‘src/pages/conjuntos/‘ - Tema: *Teoría de Conjuntos y Diagramas*

»"'—

»"'## [carpeta] ‘/src/components‘

»"'Componentes Vue reutilizables compartidos entre todos los equipos (botones, tablas, layout, etc.).

»"'—

»"'## [carpeta] ‘/src/router‘

»"'‘index.ts‘ — Define las rutas URL de la app (qué página aparece en qué URL).

»"'# [mapa] MAPA DEL REPOSITORIO

»"'Guía rápida para encontrar cualquier cosa en el proyecto.

»"'—

»"'## [carpeta] Raíz del Proyecto

»"'| Archivo | Para qué sirve | |—|—| | ‘README.md‘ | Presentación del proyecto: equipos, reglas, rúbrica de evaluación. | | ‘CRONOGRAMA.md‘ | Fechas exactas de los 4 sprints y qué entregar en cada uno. | | ‘GUIA_ESTUDIANTE.md‘ | Manual de herramientas y paso a paso por entregable. | | ‘CONTRIBUTING.md‘ | Reglas para hacer un buen Pull Request. | | ‘MAPA.md‘ | Este archivo. Índice de todo el repositorio. |

»"'—

»"'## [carpeta] ‘/TAREAS‘

»"'| Archivo | Para qué sirve | |—|—| | ‘01_Propuesta.md‘ | Instrucciones y plantilla para el Entregable 1 (Diseño). |

»"'*(Se irán añadiendo ‘02_Motor.md‘, ‘03_UI.md‘, ‘04_Deploy.md‘ según avancen los sprints.)*

»"'—

»"'## [carpeta] ‘/src/pages‘ — El Código

»"'Cada carpeta pertenece a un equipo. Ahí vive todo su código Vue.

»"'## [mapa] Mapa de Rutas (Páginas del Proyecto)

»"'La estructura web del proyecto se divide en 4 módulos interactivos, uno por equipo:

»"'1. **Equipo Sinergia** - ‘src/pages/tablas/‘ - Tema: *Fundamentos, Conectivos y Tablas de Verdad* 2. **Equipo Los hijos de Linus** - ‘src/pages/inferencias/‘ - Tema: *Inferencias Lógicas y Validaciones* 3. **Equipo 3** - ‘src/pages/cuantificadores/‘ - Tema: *Cuantificadores y Lógica de Predicados* 4. **Equipo Linus** - ‘src/pages/conjuntos/‘ - Tema: *Teoría de Conjuntos y Diagramas*

»"'—

»"'## [carpeta] ‘/src/components‘

»"'Componentes Vue reutilizables compartidos entre todos los equipos (botones, tablas, layout, etc.).

»"'—

»"'## [carpeta] ‘/src/router‘

»"'‘index.ts‘ — Define las rutas URL de la app (qué página aparece en qué URL).

»"'Nota final

»"'Compilado con `xelatex` TeX Live 2026. Código y PDFs en `docs/`. Repositorio https://github.com/EnriDiazCayaca/logica_simbolica_global.

»"'Anexo Validación final

»"'Validado con `xelatex` TeX Live 2026, sin bloques `???`, tabla CAPA correcta, imágenes `banner.jpg` y listados `operations.ts`.