UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Modelado Visual y Cálculo de Riesgos mediante DSL y Tecnologías Web

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Juan José Amores Arenas

Tutor: Patricio Orlando Letelier Torres

Curso 2025-2026

A mis padres, tutores, pareja y todos mis compañeros que me han acompañado durante estos años.

Agradecimientos

Me gustaría agradecer a Pedro por haberme ofrecido la oportunidad de realizar este Trabajo de Fin de Grado en su empresa durante las prácticas, tanto a él como a todos los compañeros que me han brindado su ayuda siempre que la he necesitado. También quiero agradecer a Patricio, que ha sido un gran apoyo para mi proyecto. Además, quiero agradecerle estos dos últimos años por haberme enseñado tanto sobre la ingeniería del software, con los proyectos hechos durante la carrera, los cuales me ayudaron mucho a desarrollarme como ingeniero. Sobre todo también darle las gracias por haber aceptado la propuesta del proyecto. Por último, quiero dar las gracias a todos mis compañeros que me han acompañado durante estos últimos 4 años de mi vida, quiénes me han ayudado y enseñado mucho a lo largo de este tiempo.

Resum

Aquest TFG presenta una aplicació web transversal per a realitzar modelatge visual i calcular riscos en àmbits crítics com ara missions espacials, banca i assegurances. El nucli d'aquesta aplicació és la definició d'un model de domini que serveix com a base d'un DSL visual, que al seu torn es troba enllaçat a una potent llibreria de diagramació web. Mitjançant una arquitectura que es basa principalment en les tecnologies web modernes, Angular i TypeScript, el sistema tradueix càlculs probabilístics abstractes a diagrames com els que pot utilitzar qualsevol usuari. La solució se sustenta en una arquitectura modular orientada al frontend, basada en Angular i TypeScript, de manera que la interacció, el modelatge visual, la importació/exportació i el càlcul probabilístic s'executen principalment en el navegador. Aquesta organització deixa oberta la possibilitat d'incorporar en el futur un backend especialitzat per a càlculs més pesats, encara que no forma part de l'abast actual de l'aplicació. La idea és fusionar la fredor de les dades estadístiques amb els diagrames, de manera que les empreses puguen entendre models complexos i prendre decisions d'una forma més segura, ràpida i informada.

Paraules clau: aplicació web, modelatge visual, càlcul de riscos, sectors crítics, llenguatge de domini específic, lògica de negoci, diagrames interactius

Resumen

Este TFG presenta una aplicación web transversal para realizar modelado visual y calcular riesgos en ámbitos críticos tales como misiones espaciales, banca y seguros. El núcleo de dicha aplicación es la definición de un modelo de dominio que sirve como base de un DSL visual, que a su vez se encuentra enlazado a una potente librería de diagramación web. Mediante una arquitectura que se basa principalmente en las modernas tecnologías web, Angular y TypeScript, el sistema traduce cálculos probabilísticos abstractos a diagramas como los que son utilizables por cualquier usuario. La solución se soporta en una arquitectura modular orientada al frontend, basada en Angular y TypeScript, de forma que la interacción, el modelado visual, la importación/exportación y el cálculo probabilístico se ejecutan principalmente en el navegador. Esta organización deja abierta la posibilidad de incorporar en el futuro un backend especializado para cálculos más pesados, aunque no forma parte del alcance actual de la aplicación. La idea es fusionar la frialdad de los datos estadísticos con los diagramas de modo que las empresas puedan entender modelos complejos y llevar a cabo decisiones de forma más segura, rápida e informada.

Palabras clave: aplicación web, modelado visual, cálculo de riesgos, sectores críticos, lenguaje de dominio específico, lógica de negocio, diagramas interactivos

Abstract

This Final Degree Project presents a cross-cutting web application for visual modelling and risk calculation in critical domains such as space missions, banking and insurance. The core of this application is the definition of a domain model that serves as the basis for a visual DSL, which is in turn linked to a powerful web diagramming library. Through an architecture mainly based on modern web technologies, Angular and TypeScript, the system translates abstract probabilistic calculations into diagrams that can be used by any user. The solution is supported by a modular frontend-oriented architecture based on Angular and TypeScript, so that interaction, visual modelling, import/export and probabilistic calculation are carried out mainly in the browser. This organisation leaves open the possibility of incorporating a specialised backend for heavier calculations in the future, although this is not part of the current scope of the application. The idea is to merge the coldness of statistical data with diagrams so that companies can understand complex models and make decisions in a safer, faster and more informed way.

Key words: web application, visual modeling, risk calculation, critical sectors, domain-specific language, business logic, interactive diagrams

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este primer capítulo introductorio se presenta el contexto general del TFG. En primer lugar, se expone la motivación del proyecto, la cual se centra en la necesidad de representar y analizar riesgos en situaciones inciertas y con relaciones de dependencia complejas. En segundo lugar, se definen el objetivo general del trabajo y los objetivos específicos que guían el desarrollo de la solución propuesta. Por último, se explica la estructura de la memoria, es decir, el contenido fundamental de cada uno de los capítulos que la componen.

1.1 Motivación

En los últimos años, la gestión del riesgo ha ganado una importancia creciente debido al aumento de la incertidumbre global y a la aparición de crisis cada vez más relacionadas entre sí. Los riesgos actuales no suelen producirse de forma aislada, sino que pueden propagarse entre sectores, organizaciones y sistemas tecnológicos, generando así efectos en cadena difíciles de anticipar. Esta tendencia se refleja en informes recientes como el *Global Risks Report 2025*, donde se describe un contexto internacional marcado por la combinación de riesgos geopolíticos, ambientales, sociales y tecnológicos [33].

Esta evolución también se puede observar en el impacto económico de los eventos extremos. Según Munich Re, las catástrofes naturales provocaron en 2024 pérdidas globales de aproximadamente 320.000 millones de dólares, de las cuales alrededor de 140.000 millones estaban aseguradas [34]. Estos datos muestran que los riesgos tienen consecuencias económicas relevantes para empresas, administraciones y sectores críticos. Bajo estas circunstancias, confiar únicamente en modelos estáticos o en datos históricos puede ser insuficiente, sobre todo cuando surgen eventos poco frecuentes, escenarios sin precedentes o situaciones de alta incertidumbre [1, 8].

En este contexto, las organizaciones necesitan métodos que permitan representar escenarios cambiantes y relaciones complejas, más allá de tablas o análisis aislados. Sin embargo, muchas herramientas utilizadas en el mercado no cubren por completo esta necesidad. Algunas soluciones tradicionales permiten analizar datos, ejecutar simulaciones o generar resultados estadísticos, pero no siempre muestran de forma clara cómo se relacionan las variables ni cómo se propaga la incertidumbre a lo largo del sistema [14, 15, 18, 20].

Frente a ello, los modelos probabilísticos y causales, como las redes bayesianas basadas en grafos dirigidos acíclicos, representan de manera más precisa las dependencias entre variables y actualizan las probabilidades al introducir nueva evidencia [7, 11]. No obstante, para utilizarlas se necesitan habilidades avanzadas de programación, modelado probabilístico o estadística, lo cual restringe su uso a los usuarios menos especializados.

Ante esta situación, surge la posibilidad de crear una herramienta que acerque estas capacidades a personas menos especializadas. La propuesta de este trabajo busca facilitar el uso del modelado probabilístico, al integrar métodos bayesianos rigurosos con una interfaz gráfica e intuitiva. De esta forma, se pretende ofrecer una forma más clara y accesible de representar los riesgos, favoreciendo una mejor comprensión y una toma de decisiones más informada en distintos tipos de organizaciones.

1.2 Objetivos

Con el fin de guiar el desarrollo de la solución propuesta, se han definido un objetivo general y varios objetivos específicos que delimitan el alcance técnico y funcional del proyecto.

1.2.1. Objetivo General

El objetivo general de este TFG es diseñar y desarrollar una aplicación web que permita representar visualmente modelos de riesgo, y calcular probabilidades de forma dinámica para facilitar la toma de decisiones críticas.

1.2.2. Objetivos Específicos

Para materializar el objetivo general, el trabajo se dividirá en los siguientes hitos técnicos:

- **Definir un modelo de dominio específico para riesgos:** Diseñar una representación estructurada que permita expresar variables probabilísticas, relaciones entre eventos, evidencias y reglas básicas de persistencia, sirviendo como base del DSL visual de la aplicación.
- **Integrar el modelo con una herramienta de diagramación:** Conectar la representación interna del sistema con la librería Daga, permitiendo crear, editar y visualizar grafos probabilísticos de forma interactiva.
- **Desarrollar una arquitectura web modular:** Implementar una aplicación frontend basada en Angular y TypeScript, separando la interfaz, el modelado visual, el cálculo probabilístico, la persistencia y las pruebas.

- **Implementar el cálculo probabilístico:** Desarrollar los mecanismos necesarios para trabajar con modelos binomiales y bayesianos, incluyendo la gestión de evidencias, la propagación de probabilidades y técnicas de inferencia.
- **Implementar la persistencia e intercambio de modelos:** Permitir la importación y exportación de datos mediante CSV y un formato propio, de forma que los modelos puedan almacenarse, recuperarse y reutilizarse posteriormente.
- **Validar el sistema desarrollado:** Comprobar el correcto funcionamiento de la aplicación mediante pruebas unitarias, pruebas end-to-end, pruebas de aceptación y ejemplos representativos de uso.

1.3 Estructura de la memoria

La presente memoria se organiza en seis capítulos principales, además de la bibliografía y varios apéndices complementarios.

En el Capítulo 1 se introduce el contexto general, donde se expone la motivación del proyecto y son definidos el objetivo general y los objetivos específicos los cuales servirán de guía en el desarrollo de la solución.

En el Capítulo 2 se presenta el estado del arte. En primer lugar, se analiza el contexto del riesgo, la incertidumbre y la causalidad. Posteriormente, se estudian distintas herramientas de cálculo probabilístico y modelado causal, comparando sus ventajas y limitaciones. Finalmente, se revisan las tecnologías relacionadas con lenguajes de dominio específico, librerías de diagramación web y arquitecturas web modernas, con el objetivo de justificar la necesidad de la solución propuesta.

En el Capítulo 3 se describen las tecnologías utilizadas durante el desarrollo del proyecto. Son detallados: el entorno de desarrollo, los lenguajes de programación, el framework utilizado el cual es Angular, la librería de diagramación Daga, el uso de RxJS, las herramientas de pruebas, el sistema de control de versiones, la integración continua y el despliegue de la aplicación.

En el Capítulo 4 se desarrolla la solución propuesta. Este capítulo recoge la metodología de trabajo empleada a lo largo del desarrollo de la solución, la definición de requisitos funcionales y no funcionales, el diseño de la interfaz de usuario, la arquitectura general del sistema, la implementación de los modos de cálculo probabilístico y las pruebas realizadas para validar el funcionamiento de la aplicación.

En el Capítulo 5 se presenta la cronología del TFG. Se describe la evolución temporal del proyecto, la organización del trabajo, la correspondencia entre casos de uso y unidades de trabajo así como la dedicación temporal realizada durante las distintas fases del desarrollo.

En el Capítulo 6 se exponen las conclusiones del trabajo. Se evalúa el grado de cumplimiento de los objetivos definidos, se describe el estado actual de la aplicación, se reflexiona sobre la relación del proyecto con la formación recibida durante la titulación y se plantean posibles líneas de trabajo a futuro.

Finalmente, se incluye la bibliografía utilizada como apoyo teórico y técnico, así como varios apéndices. Estos recogen información adicional sobre la configuración del sistema, una guía de uso de la aplicación y la relación del proyecto con los Objetivos de Desarrollo Sostenible (ODS).

CAPÍTULO 2
Estado del arte

2.1 Introducción al contexto: riesgo, incertidumbre y causalidad

En los últimos años, el entorno global se ha vuelto poco a poco más incierto y complejo, esto se ve caracterizado por la aparición de crisis globales capaces de propagarse entre múltiples sectores y sistemas. Sucesos como los conflictos políticos, los cambios tecnológicos bruscos, las crisis de energía o los desastres naturales nos han demostrado que los problemas actuales casi nunca vienen solos. Más bien, se conectan entre sí y provocan un efecto dominó que es muy difícil de prever.

En este contexto, los enfoques tradicionales de la gestión del riesgo muestran importantes limitaciones al depender únicamente de los análisis históricos. Muchos modelos clásicos parten de escenarios estables y de relaciones simples entre variables. Esto dificulta representar fenómenos complejos, crisis encadenadas o eventos poco frecuentes. Como consecuencia, la toma de decisiones pueden apoyarse en modelos demasiado simples, los cuales no reflejan bien el comportamiento real de los sistemas analizados [8].

Esta forma de entender el riesgo ha cambiado en los marcos actuales. La norma ISO 31000:2018 [12] define el riesgo como "el efecto de la incertidumbre sobre los objetivos". Esta definición amplía el como era percibido el riesgo, ya que no lo trata como un valor aislado, sino como parte de un conjunto de relaciones entre variables.

Sin embargo, gran parte de las metodologías tradicionales continúan representando el riesgo usando matrices cualitativas o tablas estáticas. Aunque estas herramientas son ampliamente utilizadas por su simplicidad, muchos autores han señalado sus limitaciones matemáticas y analíticas. En algunos casos, la combinación de escalas ordinales mediante operaciones aritméticas puede generar resultados sin un significado estadístico claro, produciendo una falsa sensación de precisión y control sobre el sistema analizado [1].

Ante estas limitaciones, el análisis moderno del riesgo ha evolucionado hacia enfoques basados en modelos probabilísticos y causales. Cuando una variable o evento influye sobre otros elementos del sistema, se vuelve necesario representar dichas dependencias explícitamente para comprender cómo se está propagando la incertidumbre. En este ámbito, los Grafos Dirigidos Acíclicos (Directed Acyclic Graphs, DAG) se han consolidado como una de las estructuras más utilizadas para modelar causalidad y propagación probabilística [7].

En un DAG, cada nodo representa una variable aleatoria y las conexiones dirigidas reflejan relaciones de dependencia entre eventos. Sobre esta estructura es posible aplicar la inferencia bayesiana y la probabilidad condicional para actualizar dinámicamente el estado del sistema cuando aparece nueva evidencia. Este enfoque permite no solo estimar probabilidades, sino también comprender cómo un cambio en una variable puede afectar al resto de elementos conectados [11].

A pesar de sus ventajas, muchas de las herramientas actuales que implementan estos modelos presentan grandes barreras de entrada. Por un lado, las soluciones tradicionales de simulación y análisis estadístico suelen priorizar la capacidad predictiva, pero ofrecen poca transparencia respecto a las relaciones causales del modelo. Por otro lado, las plataformas especializadas en inferencia bayesiana y modelado causal requieren habitualmente conocimientos avanzados de estadística, programación o teoría probabilística.

Como consecuencia de todo esto, existe una necesidad clara de herramientas que combinen rigor probabilístico, representación visual e interacción sencilla. En este sentido, la solución propuesta propone el desarrollo de un sistema orientado al modelado visual de riesgos mediante tecnologías web modernas, donde se integrarán conceptos de inferencia bayesiana, representación causal mediante DAG y lenguajes de dominio específico (DSL) [5,9,10], con el objetivo de facilitar la construcción y comprensión de modelos probabilísticos complejos.

2.2 Estado del arte en herramientas de cálculo probabilístico

El cálculo probabilístico aplicado a la gestión del riesgo ha evolucionado con el tiempo. En sus primeras etapas se apoyaba sobre todo en datos históricos y en modelos estadísticos clásicos, como la prueba T, la regresión lineal o la prueba de Chi-cuadrado. Estos enfoques siguen siendo útiles, pero no siempre permiten entender cómo se relacionan los riesgos dentro de un sistema.

Con los años, han aparecido herramientas capaces de representar dependencias entre variables, cambios en la incertidumbre y relaciones causales. Esta evolución responde a una necesidad clara: no basta con calcular la probabilidad de un evento de forma aislada, si no que también es necesario entender cómo ese evento puede influir en otros elementos del sistema.

En este contexto, el software probabilístico puede dividirse en dos grandes grupos. El primero, incluye las herramientas tradicionales de análisis estadístico y simulación. Estas están orientadas a calcular, predecir o simular resultados a partir de datos y distribuciones previamente definidos. El otro está compuesto por los sistemas centrados en inteligencia artificial probabilística, redes bayesianas y modelos causales. Estas plataformas permiten representar las dependencias entre variables mediante grafos, y actualizar las probabilidades cuando se incorpora nueva evidencia.

Esta diferencia resulta importante para este trabajo, ya que la solución propuesta está situada entre ambos enfoques. Ya que por una parte busca mantener el rigor matemático de los modelos probabilísticos causales, pero por la otra parte trata de reducir la complejidad técnica mediante una interfaz visual y un DSL.

2.2.1. Evolución del modelado probabilístico: de la estadística clásica a la causalidad

Históricamente, el análisis probabilístico se ha apoyado en métodos estadísticos frecuentistas[1] y modelos analíticos clásicos[2]. Desde esta perspectiva, el análisis de variables aleatorias parte de datos observados, distribuciones conocidas y relaciones matemáticas especificadas previamente. Por ejemplo, los eventos discretos suelen modelarse utilizando distribuciones como Bernoulli, Binomial o Poisson. Sin embargo, las variables continuas se representan mediante distribuciones como la Normal, Lognormal o Gamma.

Este enfoque ha sido ampliamente utilizado en ingeniería, economía, fabricación, seguros y gestión de proyectos, ya que permite estimar frecuencias, impactos, desviaciones y escenarios futuros a partir de datos históricos. Sin embargo, su principal limitación aparece cuando las variables del sistema no son independientes entre sí, o cuando el fenómeno estudiado presenta relaciones causales complejas.

Durante años, la forma habitual de analizar las relaciones entre variables consistía en definir a mano ecuaciones o fórmulas que expresaran cómo una variable independiente afectaba a una dependiente. Este método sigue siendo válido en muchos contextos, pero pierde utilidad cuando el sistema incluye varias relaciones cruzadas, dependencias no lineales o propagación de incertidumbre entre eventos.

La aparición de la inteligencia artificial, el aprendizaje automático y la inferencia bayesiana han ampliado significativamente las capacidades del modelaje probabilístico. Frente al análisis estadístico clásico, centrado en explicar relaciones a partir de modelos definidos previamente, las técnicas modernas permiten detectar patrones, aprender dependencias a partir de datos y actualizar dinámicamente las probabilidades del sistema. No obstante, muchas técnicas de predicción avanzadas funcionan como cajas negras, ofreciendo resultados precisos pero difíciles de interpretar.

Por ello, en los últimos años ha ganado importancia el modelado causal mediante Grafos Dirigidos Acíclicos, también conocidos como DAG. En estos modelos, las variables se representan como nodos y las relaciones de dependencia se expresan mediante aristas. Esta estructura permite aplicar inferencia bayesiana y probabilidad condicional para analizar cómo la evidencia sobre una variable modifica la probabilidad de otras variables relacionadas. Este enfoque es especialmente relevante para la gestión del riesgo, donde comprender la propagación del fallo resulta tan importante como estimar numéricamente el riesgo final [7, 11].

[1] Enfoque de inferencia estadística que interpreta la probabilidad como la frecuencia relativa de un evento en un gran número de repeticiones, algunos ejemplos son: Pruebas T, regresión lineal y las pruebas de chi-cuadrado

[2] Estimación de Máxima Verosimilitud (EMV), Intervalos de Confianza, Contrastes de Hipótesis entre otras.

2.2.2. Herramientas tradicionales de análisis estadístico y simulación

Las herramientas tradicionales de cálculo probabilístico nacieron con una orientación principalmente estadística, matemática o de simulación. Su objetivo principal es permitir al usuario introducir datos, definir distribuciones o fórmulas y obtener resultados analíticos, gráficos o simulados. Aunque algunas han incorporado módulos de inteligencia artificial, su núcleo conceptual continúa siendo el análisis estadístico clásico.

Dentro de este grupo destacan IBM SPSS Statistics[3], Minitab[4], @RISK[5], Oracle Crystal Ball[6], MATLAB[7] y Wolfram Mathematica[8].

IBM SPSS Statistics

IBM SPSS Statistics es una de las herramientas más consolidadas en el ámbito del análisis estadístico aplicado. Su principal fortaleza reside en ofrecer una interfaz gráfica accesible que permite realizar análisis descriptivos, regresiones, contrastes de hipótesis, modelos predictivos y análisis multivariante sin necesidad de programar de forma intensiva.

SPSS ha sido ampliamente utilizado en ciencias sociales, investigación de mercados, salud y entornos académicos. Su orientación está claramente centrada en el análisis de tablas de datos: el usuario introduce un conjunto de variables, selecciona procedimientos estadísticos y obtiene tablas o gráficos de resultados. En los últimos años, IBM ha incorporado funcionalidades relacionadas con inteligencia artificial y asistentes de interpretación de resultados, con el objetivo de hacer más comprensibles los análisis generados por el sistema [15].

No obstante, desde el punto de vista del modelado causal de los riesgos, SPSS presenta una limitación importante: aunque permite analizar relaciones entre variables, no está diseñado para construir modelos visuales de propagación causal. La estructura del sistema analizado no se representa como un grafo explícito, sino como tablas, coeficientes y resultados estadísticos.

[3] https://www.ibm.com/es-es/products/spss-statistics
[4] https://www.minitab.com/en-us/products/minitab/
[5] https://goo.su/5HkH4V
[6] https://www.oracle.com/es/applications/crystalball/
[7] https://www.mathworks.com/products/matlab.html
[8] https://www.wolfram.com/mathematica/

Minitab Statistical Software

Minitab es una herramienta estadística especialmente extendida en entornos industriales, fabricación, control de calidad y metodologías Six Sigma[9]. Su propuesta de valor se centra en facilitar el análisis de procesos, la detección de defectos, la evaluación de capacidad y la mejora continua.

Al igual que SPSS, Minitab permite trabajar con variables discretas y continuas, realizar regresiones, análisis de varianza, control estadístico de procesos y modelos predictivos. En versiones recientes también ha incorporado módulos de analítica predictiva basados en técnicas como árboles de decisión, Random Forests o CART, ampliando su utilidad en contextos donde las relaciones entre variables son complejas [20].

Sin embargo, su enfoque sigue siendo principalmente estadístico-industrial. Minitab es muy eficaz para responder preguntas como por ejemplo si un proceso está bajo control o qué factores influyen en una variable de calidad, pero no está orientado a representar gráficamente una red causal de riesgos interdependientes ni a propagar probabilidades mediante inferencia bayesiana.

@RISK y Oracle Crystal Ball

@RISK y Oracle Crystal Ball representan el ecosistema clásico de simulación probabilística sobre hojas de cálculo. Ambas herramientas funcionan como complementos de Microsoft Excel y permiten sustituir valores deterministas por distribuciones de probabilidad. A partir de estas, ejecutan simulaciones de Montecarlo para obtener resultados, percentiles, intervalos de confianza y análisis de sensibilidad.

Su principal ventaja es que se integran en un entorno familiar para muchos usuarios: las hojas de cálculo. Esto facilita su adopción en finanzas, gestión de proyectos, análisis de inversiones, seguros o planificación empresarial. El usuario puede construir un modelo en Excel, definir incertidumbre en determinadas celdas y analizar cómo se comporta el resultado final tras miles de iteraciones [14, 18].

No obstante, esta integración con Excel también constituye una limitación. El modelo queda distribuido entre celdas, fórmulas y hojas, lo que dificulta visualizar la estructura causal completa. Aunque estas herramientas son muy útiles para simular escenarios, no ofrecen de forma nativa una representación gráfica del sistema como red de dependencias. En consecuencia, pueden calcular correctamente un resultado, pero no siempre ayudan a comprender de forma clara cómo se propaga el riesgo entre variables.

[9] La metodología Six Sigma busca reducir la variabilidad de los procesos para alcanzar un nivel de calidad en el que se produzcan menos de 3,4 defectos por millón de oportunidades.

MATLAB y Wolfram Mathematica

MATLAB y Wolfram Mathematica representan un enfoque más técnico y programático del cálculo matemático y probabilístico. Ambas herramientas ofrecen un alto grado de potencia expresiva, permitiendo trabajar con álgebra simbólica, cálculo matricial, simulación, optimización, estadística avanzada y visualización científica.

MATLAB destaca especialmente en ingeniería, procesamiento de señales, control, simulación numérica y análisis matricial. Por su parte, Mathematica ofrece una capacidad muy elevada para cálculo simbólico, manipulación algebraica y resolución analítica de problemas matemáticos complejos [19, 22].

Estas plataformas son extremadamente potentes para usuarios expertos, pero presentan una barrera de entrada significativa. Requieren conocimientos técnicos avanzados y no están diseñadas específicamente para que usuarios no especialistas modelen visualmente riesgos mediante grafos causales. Su orientación es más cercana al cálculo científico general que a la construcción accesible de modelos probabilísticos visuales.

Síntesis del ecosistema tradicional

En conjunto, las herramientas tradicionales destacan por su madurez, robustez y amplia adopción en entornos profesionales. Sin embargo, comparten varias limitaciones relevantes para este trabajo, en la Tabla 2.1 se encuentran las limitaciones principales de los modelos de riesgo tradicionales anteriormente nombrados.

Limitación	Descripción
Modelo poco visual	La estructura del riesgo suele quedar oculta en tablas, fórmulas o scripts.
Causalidad limitada	Permiten analizar correlaciones o simular escenarios, pero no siempre representan causalidad explícita.
Dependencia de expertos	Requieren conocimientos estadísticos, de programación o dominio avanzado de la herramienta.
Caja negra parcial	Algunos módulos predictivos ofrecen resultados difíciles de interpretar causalmente.
Coste y dependencia tecnológica	Muchas soluciones son comerciales, propietarias o dependientes de plataformas concretas.

Tabla 2.1: Limitaciones principales de los modelos de riesgo tradicionales

Estas limitaciones justifican la búsqueda de enfoques que permitan combinar simulación, causalidad, visualización y accesibilidad en un mismo entorno.

2.2.3. Herramientas de inteligencia artificial probabilística y causal

Frente al software estadístico tradicional, las herramientas de inteligencia artificial probabilística y causal han sido diseñadas para trabajar de forma más explícita con incertidumbre, dependencias y relaciones causales. En este grupo destacan BayesiaLab[10], GeNIe Modeler[11] con SMILE[12], PyMC[13] y JASP[14].

Estas plataformas no se limitan a calcular probabilidades aisladas, sino que permiten construir modelos donde las variables están conectadas mediante relaciones de dependencia. En muchos casos, este tipo de relaciones son representadas mediante redes bayesianas, que combinan grafos dirigidos acíclicos con tablas de probabilidad condicional. Este enfoque permite actualizar el sistema cuando se introduce nueva evidencia y observar cómo esta afecta al resto de variables conectadas.

BayesiaLab

BayesiaLab es una de las herramientas comerciales más reconocidas en el ámbito de las redes bayesianas. Su objetivo principal es facilitar la construcción, aprendizaje y análisis de modelos probabilísticos causales. A partir de datos, esta herramienta permite aprender estructuras de dependencia, construir redes bayesianas, realizar inferencia y analizar escenarios.

Una de sus principales fortalezas es la capacidad de representar visualmente relaciones entre variables y actualizar probabilidades cuando se introduce evidencia. Esto la convierte en una herramienta especialmente útil para problemas donde interesa comprender no solo qué puede ocurrir, sino también por qué ocurre y cómo se propaga la información dentro del sistema [2, 17].

Sin embargo, BayesiaLab también presenta barreras importantes. Se trata de un software comercial especializado, con una curva de aprendizaje elevada y orientado a usuarios con conocimientos avanzados en redes bayesianas, análisis de datos y modelado causal. Además, en muchos contextos trabaja preferentemente con variables discretas, lo que puede requerir transformar variables continuas antes del análisis.

[10] https://www.bayesia.com/
[11] https://www.bayesfusion.com/genie/
[12] https://www.bayesfusion.com/smile/
[13] https://www.pymc.io/welcome.html
[14] https://jasp-stats.org/

GeNIe Modeler y SMILE Engine

GeNIe Modeler, junto con el motor SMILE, constituye otro referente dentro del ecosistema de redes bayesianas. Desarrollado originalmente en la Universidad de Pittsburgh y mantenido actualmente por BayesFusion, permite construir redes bayesianas, diagramas de influencia y modelos probabilísticos complejos.

Una de sus principales ventajas es su capacidad para trabajar con redes híbridas, combinando variables discretas y continuas dentro de un mismo modelo. Además, SMILE está desarrollado como un motor eficiente que puede integrarse con distintos lenguajes de programación, lo que lo convierte en una solución muy flexible para investigación, docencia y desarrollo de aplicaciones probabilísticas [11, 13].

No obstante, GeNIe también requiere una comprensión sólida de los conceptos de inferencia probabilística, estructura de redes, evidencia y probabilidad condicional. Aunque su interfaz gráfica facilita parte del trabajo, sigue siendo una herramienta especializada, más cercana al usuario experto que al analista generalista.

PyMC

PyMC es una de las bibliotecas de programación probabilística de código abierto más relevantes. A diferencia de BayesiaLab o GeNIe, PyMC no se centra en una interfaz gráfica, sino en la definición programática de modelos probabilísticos mediante Python. Permite construir modelos bayesianos complejos, definir distribuciones a priori, establecer relaciones entre variables y realizar inferencia mediante algoritmos avanzados como Hamiltonian Monte Carlo.

Su principal fortaleza es la expresividad. Esta biblioteca permite modelar problemas muy complejos y personalizados, combinando variables continuas, discretas, jerarquías, modelos latentes y estructuras estadísticas avanzadas. Por este motivo, es una herramienta muy utilizada en ciencia de datos, investigación estadística y modelado probabilístico avanzado [4, 21].

La principal limitación de PyMC, desde la perspectiva de este trabajo, es su elevada complejidad. Requiere conocimientos de programación, estadística bayesiana, definición de modelos y de diagnóstico de inferencia. Además, al no ofrecer una interfaz visual orientada al modelado de riesgos, la estructura del modelo suele quedar implícita en el código, dificultando su comprensión por parte de usuarios no técnicos o especializados.

JASP

JASP es una herramienta estadística de código abierto orientada a facilitar el análisis estadístico clásico y bayesiano mediante una interfaz gráfica sencilla. Su objetivo es ofrecer una alternativa moderna y accesible a herramientas comerciales como SPSS, especialmente en entornos académicos.

Destaca por acercar la estadística bayesiana a usuarios no expertos, permitiendo ejecutar análisis mediante menús y visualizar resultados de forma clara. También ha incorporado módulos relacionados con aprendizaje automático, análisis predictivo y redes de correlación, ampliando progresivamente su alcance [3, 16].

Aun así, esta herramienta no está diseñada como una plataforma de modelado causal visual basada en DAG. Su foco principal sigue siendo el análisis estadístico de tablas de datos, no la construcción de modelos gráficos interactivos donde el usuario defina y observe la propagación del riesgo en tiempo real.

Síntesis del ecosistema causal y bayesiano

Las herramientas causales y bayesianas ofrecen ventajas claras frente al software tradicional, especialmente cuando el objetivo es representar dependencias variables y actualizar probabilidades dinámicamente. Las ventajas principales de los modelos de riesgo causales y bayesianos se muestran en la Tabla 2.2

Ventaja	Descripción
Representación causal	Permiten modelar relaciones explícitas entre variables mediante grafos
Inferencia dinámica	Las probabilidades pueden actualizarse al introducir nueva evidencia
Mayor transparencia	El modelo puede explicar cómo una variable afecta a otra.
Adecuación a sistemas complejos	Son útiles en problemas con múltiples dependencias e incertidumbre estructural
Coste y dependencia tecnológica	Muchas soluciones son comerciales, propietarias o dependientes de plataformas concretas.

Tabla 2.2: Ventajas principales de los modelos de riesgo causales y bayesianos

Sin embargo, en la Tabla 2.3 se presentan algunas de las limitaciones más importantes de estos modelos.

Limitación	Descripción
Curva de aprendizaje elevada	Requieren conocimientos avanzados de estadística, probabilidad o programación.
Complejidad técnica	La configuración de modelos puede resultar difícil para usuarios no expertos.
Barreras de adopción	Algunas herramientas son comerciales, especializadas o poco intuitivas.
Sobrecarga computacional	Determinados modelos pueden requerir recursos elevados.
Menor accesibilidad	No siempre están pensadas para uso transversal en organizaciones.

Tabla 2.3: Barreras y limitaciones técnicas en la implementación de modelos de riesgo causales y bayesianos

Por lo tanto, aunque estas herramientas resuelven mejor el problema de la causalidad, no eliminan por completo la barrera de acceso para usuarios que necesitan modelar riesgos de forma visual, rápida y comprensible.

2.2.4. Comparativa de las herramientas existentes

En la Tabla 2.4 se muestra un resumen de las principales herramientas analizadas, comparando su paradigma de trabajo, nivel de visualización, soporte causal, licencia y adecuación al problema abordado en este trabajo.

Herramienta	Paradigma principal	Visualización	Soporte causal	Licencia	Limitación principal
IBM SPSS Statistics	Estadística clásica y analítica predictiva	Baja: resultados en tablas y gráficos	Limitado	Comercial	No modela grafos causales de forma nativa
Minitab	Control de calidad y estadística industrial	Baja-media	Limitado	Comercial	Orientado a procesos y tablas de datos
@RISK / Crystal Ball	Simulación Montecarlo sobre Excel	Baja: fórmulas en hojas de cálculo	Muy limitado	Comercial	El modelo queda oculto en celdas
MATLAB / Mathematica	Cálculo matemático y científico	Variable, dependiente del usuario	No específico	Comercial	Alta complejidad técnica
BayesiaLab	Redes bayesianas y causalidad	Alta	Alto	Comercial	Curva de aprendizaje elevada
GeNIe / SMILE	Redes bayesianas e híbridas	Alta	Alto	Gratuito académico / comercial	Herramienta especializada
PyMC	Programación probabilística	Baja: modelo en código	Alto, si se modela explícitamente	Open Source	Requiere programación y estadística avanzada
JASP	Estadística clásica y bayesiana accesible	Media	Limitado	Open Source	No está orientado a DAG interactivos

Tabla 2.4: Comparativa de herramientas y paradigmas de modelado

De esta comparativa se puede concluir que no existe una solución que combine simultáneamente accesibilidad, visualización causal, inferencia probabilística y bajo coste de entrada. Por un lado, las herramientas tradicionales son accesibles y robustas, pero no representan adecuadamente la causalidad. Por otro lado, las herramientas bayesianas y causales son mucho más potentes desde el punto de vista matemático, pero resultan difíciles de adoptar para usuarios no expertos.

2.2.5. Evaluación crítica del software actual

El estudio de la información presentada sobre las herramientas existentes, permite reconocer un gran dilema dentro del contexto del cálculo de probabilidades aplicado a los riesgos, se trata de conseguir un equilibrio entre una baja complejidad de uso y una buena precisión lógica.

Las herramientas tradicionales, como SPSS, Minitab, @RISK o Crystal Ball, destacan por su madurez, buena documentación, integración con flujos de trabajo ya definidos así como su relativa facilidad de uso. Estas herramientas son especialmente útiles en el caso de que el usuario necesite ejecutar análisis estadísticos, simular escenarios o generar informes de forma rápida. Sin embargo, su enfoque suele estar basado en tablas de datos, fórmulas o distribuciones definidas manualmente. Ciertamente, esto limita su capacidad para mostrar de manera explícita cómo se relacionan los riesgos entre sí.

En este sentido, es cierto que las herramientas tradicionales pueden ayudar a responder preguntas del tipo "qué resultado es probable que ocurra", pero no a responder de forma clara "por qué ocurre" o "cómo se propaga el riesgo dentro del sistema". Esta limitación resulta ser altamente relevante en entornos donde los riesgos dependen unos de otros y donde una decisión puede provocar efectos en cadena.

Por otra parte, herramientas como BayesiaLab, GeNIe o PyMC sí permiten representar relaciones causales, realizar inferencia bayesiana y actualizar probabilidades cuando hay nueva evidencia. Estas plataformas proporcionan una base más adecuada para modelar riesgos complejos, ya que proporcionan un modo de organizar conocimiento en forma de red y permiten estudiar cómo se propaga la incertidumbre.

Pero estas herramientas, en cambio, tienen una barrera de entrada mayor. En muchos casos el usuario debe entender conceptos avanzados como probabilidad condicional, distribuciones, tablas de probabilidad condicional, algoritmos de inferencia o aprendizaje estructural. Dicha dificultad limita su uso en el ámbito empresarial o multidisciplinar, donde los usuarios tienen que entender el modelo sin tener que construirlo con código o herramientas estadísticas avanzadas.

En consecuencia, el software probabilístico actual deja una brecha clara. Existen herramientas accesibles pero con poca capacidad para representar la causalidad, y por otro lado, existen herramientas causales pero accesibles en menor medida. Esta situación justifica la búsqueda de herramientas híbridas que combinen visualización, rigor probabilístico e interacción sencilla.

2.2.6. Posicionamiento de la solución propuesta

La aplicación desarrollada en este trabajo se posiciona frente a esta brecha como una herramienta de modelado visual de riesgos basada en grafos. A diferencia de las hojas de cálculo o de las herramientas estadísticas tradicionales, el modelo no queda oculto en fórmulas, celdas o scripts, sino que se representa directamente como un diagrama editable. De este modo, la estructura del riesgo pasa a ser el elemento central del sistema.

La propuesta se diferencia de las herramientas tradicionales mediante la inclusión de la noción de dependencia entre variables. En lugar de considerar los distintos riesgos como un elementos aislados, permite representar relaciones dirigidas entre los eventos, lo cual favorece el análisis de la propagación. Esta forma de proceder es coherente con los modelos basados en DAG y redes bayesianas, donde la estructura del grafo es la base utilizada para el cálculo probabilístico.

Al mismo tiempo, la solución presentada busca reducir la barrera de entrada que caracteriza a herramientas como BayesiaLab, GeNIe o PyMC. Para ello, se apoya en una interfaz visual y en un DSL que permite expresar reglas de riesgo y probabilidad de forma más cercana al dominio del usuario. El objetivo no es el de sustituir a las plataformas avanzadas de programación probabilística, sino el de proponer una herramienta más accesible para la construcción, visualización y comprensión de modelos de riesgo en entornos donde la transparencia es muy importante.

En cuanto al planteamiento de la solución, esta evita algunas limitaciones habituales del software de escritorio tradicional. Apoyándose en el uso de tecnologías web modernas, se facilita la portabilidad, la integración de la solución con otros sistemas y la capacidad de extender y/o modificar la herramienta. Esta decisión técnica es coherente con el objetivo general del proyecto: desarrollar una plataforma transversal, visual y escalable que permita la representación y la evaluación dinámica de los riesgos.

En este sentido, la aportación principal del sistema presentado no consiste únicamente en calcular probabilidades, sino en hacer visible la estructura del razonamiento probabilístico. Esta característica permite mejorar la comprensión del modelo, favorecer la trazabilidad de las decisiones y que permita acercar el modelado causal a personas que no tienen conocimientos avanzados de estadística o programación.

Comparación de las herramientas actuales frente a la solución propuesta

A modo de resumen, en la Tabla 2.5 se encuentra una comparación entre las herramientas tradicionales de la industria, las soluciones causales avanzadas y el enfoque presentado en este trabajo. Como puede observarse, la combinación de un DSL más la interfaz gráfica proporciona un equilibrio óptimo entre la transparencia del modelo y la facilidad de uso.

Dimensión	Herramientas tradicionales	Herramientas causales avanzadas	Solución propuesta
Ejemplos	SPSS, Minitab, @RISK, Crystal Ball	BayesiaLab, GeNIe, PyMC	DSL + Daga
Modelo principal	Tablas, fórmulas, simulaciones	Redes bayesianas, modelos probabilísticos	Grafo visual interactivo
Transparencia causal	Baja	Alta	Alta
Facilidad de uso	Media-alta	Baja-media	Alta
Curva de aprendizaje	Moderada	Elevada	Reducida
Programación requerida	Baja o media	Media o alta	Baja
Visualización estructural	Limitada	Alta en algunas herramientas	Nativa
Coste de adopción	Frecuentemente alto	Variable	Bajo
Objetivo principal	Analizar o simular resultados	Modelar incertidumbre causal	Modelar riesgo visualmente

Tabla 2.5: Comparativa dimensional entre enfoques tradicionales, avanzados y la solución propuesta

2.3 Estado del arte tecnológico

El desarrollo de una herramienta que permita realizar tanto el modelado visual como el cálculo probabilístico de riesgos no solo depende de las bases matemáticas descritas en las secciones anteriores, sino que también requiere de las tecnologías adecuadas para expresar las reglas del dominio, representar estructuras gráficas complejas e integrar el cálculo probabilístico en una arquitectura de software mantenible.

Desde la perspectiva de la ingeniería del software, la realización de este proyecto podría haber sido situada en el cruce de tres áreas tecnológicas: los lenguajes de dominio específico, las librerías de diagramación web y las arquitecturas web modulares. La integración de estas tres áreas permite crear una solución en la que el usuario no trabaja directamente sobre fórmulas aisladas o sobre scripts estadísticos, sino que trabaja con un modelo visual interactivo que representa el sistema de riesgos y permite operar sobre el mismo.

2.3.1. Lenguajes de Dominio Específico

Un Lenguaje de Dominio Específico (DSL), está diseñado para resolver problemas dentro de un dominio concreto, en lugar de ofrecer una solución general aplicable a cualquier tipo de problema. A diferencia de los lenguajes que tienen un propósito general, como Java, Python o C#, un DSL restringe su expresividad a un área determinada, pero a cambio, permite representar conceptos del dominio de forma más directa, compacta y comprensible para sus usuarios [5, 9, 10].

En la ingeniería del software, los DSLs ayudan a acercar el problema real a su implementación técnica. En lugar de expresar las reglas del dominio con estructuras de programación generales, un DSL permite trabajar con conceptos más próximos al lenguaje del propio dominio. Algunos ejemplos conocidos son: SQL, que se usa para consultar bases de datos, HTML y CSS, que permiten describir documentos web y estilos, o las expresiones regulares, que sirven para definir patrones de texto.

La literatura suele distinguir entre dos tipos principales de DSLs: los externos y los internos. Un DSL externo es el que cuenta con una sintaxis propia y necesita mecanismos específicos para analizar, interpretar o compilar sus instrucciones. En cambio, un DSL interno es el que se construye sobre un lenguaje existente. De esta forma, se aprovecha de su sintaxis y ecosistema, pero ofrece una forma más clara de expresar operaciones de un dominio concreto [9]. En ambos casos, el objetivo es el mismo: permitir que las operaciones importantes del dominio puedan expresarse de una manera más clara que con código generalista.

En el contexto de este trabajo, el dominio específico es el modelado probabilístico de riesgos. Por tanto, el DSL debe permitir expresar conceptos como eventos, variables aleatorias, relaciones de dependencia, pesos, evidencia y reglas de propagación. Esta especialización resulta especialmente útil porque el usuario final no necesita manipular directamente estructuras matemáticas abstractas, sino definir elementos propios del análisis de riesgo.

Además, el uso de un DSL aporta ventajas importantes desde el punto de vista de la mantenibilidad y la evolución del sistema. Al separar la lógica del dominio de los detalles de la implementación, resulta más sencillo modificar reglas, extender los operadores o añadir nuevos tipos de modelos sin alterar toda la arquitectura de la aplicación. Esta separación también favorece a la trazabilidad, ya que las reglas expresadas en el DSL pueden vincularse directamente con los elementos visuales del diagrama y con los cálculos ejecutados por el motor probabilístico.

No obstante, el desarrollo de un DSL también implica ciertos riesgos. Diseñar un lenguaje específico requiere comprender tanto el dominio de la aplicación como las técnicas de diseño de lenguajes. Una mala definición puede generar ambigüedad, dificultad de aprendizaje o problemas de mantenimiento. Por ello, autores como Mernik, Heering y Sloane señalan que los DSLs resultan adecuados cuando existe un dominio suficientemente estable, un conjunto de conceptos repetidos y una ganancia clara frente al uso de lenguajes generalistas [10].

En este proyecto, estas condiciones se cumplen razonablemente. El cálculo y visualización de riesgos trabajan con conceptos recurrentes: nodos, conexiones, probabilidades, dependencias y evidencia. Además, la alternativa tradicional[15] introduce una complejidad innecesaria para muchos usuarios. Por ello, un DSL específico para riesgos permite encapsular dicha complejidad y ofrecer una interfaz más próxima al razonamiento del analista.

2.3.2. Librerías web de diagramación

La segunda base tecnológica del proyecto es la representación visual de modelos mediante diagramas interactivos. En el caso de los riesgos probabilísticos, la visualización no es un elemento meramente estético, sino una de las partes centrales del modelo. Si los riesgos se representan como grafos dirigidos, la herramienta debe permitir crear, editar, conectar y actualizar nodos de forma dinámica.

En los últimos años han surgido múltiples librerías JavaScript orientadas a la construcción de editores gráficos, visualización de grafos, diagramas de flujo y modelos interactivos en navegador. Entre las opciones más relevantes se encuentran GoJS[16], JointJS[17], React Flow[18], Mermaid[19], Cytoscape.js[20], D3.js[21] y Daga[22]. Cada una de ellas responde a necesidades distintas y presenta ventajas además de limitaciones en función del tipo de aplicación que se desea construir.

[15] Que trata de formular estos modelos mediante hojas de cálculo, scripts o herramientas estadísticas generales.

[16] https://gojs.net/latest/

[17] https://www.jointjs.com/

[18] https://reactflow.dev/

[19] https://mermaid.js.org/

[20] https://js.cytoscape.org/

[21] https://d3js.org/

[22] https://metadev.pro/products/daga/

GoJS es una librería comercial para crear diagramas interactivos en JavaScript y TypeScript. Está orientada a aplicaciones donde se necesitan diagramas complejos, edición directa, plantillas de nodos, enlaces, agrupaciones y layouts avanzados. Su principal ventaja es su madurez y riqueza funcional, aunque su licencia comercial puede ser una barrera en proyectos académicos o de bajo presupuesto [23]. La documentación oficial la presenta como un framework moderno para visualizaciones interactivas y edición de sistemas mediante diagramas.

JointJS es otra librería JavaScript orientada a la creación de editores visuales, herramientas low-code, diagramas de procesos y aplicaciones gráficas complejas. Destaca por su flexibilidad y por estar basada en SVG, lo que permite construir interfaces visuales ricas dentro del navegador [24]. Sin embargo, al igual que otras soluciones generalistas, requiere una adaptación considerable para integrarse con un dominio específico como el cálculo probabilístico de riesgos. La propia documentación de JointJS la define como una librería para construir herramientas de diagramación interactivas en navegadores modernos.

React Flow se ha consolidado como una de las soluciones más populares para construir interfaces basadas en nodos dentro del ecosistema React. Su punto fuerte es la facilidad para crear editores visuales, flujos de trabajo y diagramas interactivos mediante componentes reutilizables [25]. No obstante, su integración natural está orientada a React, mientras que este proyecto se desarrolla sobre Angular. Esto reduce su idoneidad directa, ya que incorporarla implicaría mezclar paradigmas de framework o introducir una capa de integración adicional. Su documentación oficial la describe como un componente personalizable para construir editores basados en nodos y diagramas interactivos.

Mermaid adopta un enfoque distinto: permite generar diagramas a partir de una sintaxis textual inspirada en Markdown. Es especialmente útil para documentación técnica, diagramas simples, flujos, secuencias o modelos versionables como texto [26]. Sin embargo, su foco principal no es la edición visual interactiva ni la construcción de aplicaciones donde el usuario manipula directamente nodos, propiedades y conexiones. Por tanto, aunque resulta muy adecuada para documentación, no cubre completamente las necesidades de una herramienta de modelado visual en tiempo real. La documentación oficial de Mermaid destaca precisamente su orientación a la definición textual de diagramas.

Cytoscape.js está orientada a la visualización y análisis de grafos y redes. Es una opción especialmente potente en dominios donde se necesita explorar redes complejas, como biología, redes sociales o análisis de sistemas conectados [6]. Su fortaleza reside en la visualización y análisis de grafos, pero no está pensada principalmente como un editor de modelos de dominio con paletas, propiedades semánticas y restricciones específicas. La documentación oficial la define como una librería JavaScript de teoría de grafos para análisis y visualización de redes.

D3.js, por su parte, es una librería generalista de visualización de datos. Su flexibilidad permite construir visualizaciones altamente personalizadas sobre SVG, Canvas y HTML, incluyendo grafos y representaciones dinámicas [27]. Sin embargo, esa misma flexibilidad implica una mayor carga de desarrollo: D3 proporciona primitivas de visualización, pero no ofrece de forma nativa un editor de modelos con semántica de dominio, gestión de conexiones, reglas de edición o propiedades de nodos. D3 es muy adecuada para visualización a medida, pero menos directo para construir una herramienta de modelado completamente interactivo. La documentación oficial lo describe como una librería JavaScript para visualizaciones de datos personalizadas y dinámicas.

Frente a estas alternativas, **Daga** se presenta como una librería de diagramación orientada específicamente a la construcción de entornos de modelado web. Según la documentación de Metadev, Daga está construida sobre HTML5, SVG y JavaScript, y está pensada para actuar como frontend visual de modelos complejos en navegadores modernos [28]. Además, Metadev la describe como una librería para construir editores visuales de modelos y DSLs, con funcionalidades como paleta de componentes, restricciones semánticas, explorador de propiedades, zoom, panning, undo/redo, autolayout y soporte para escritorio y dispositivos táctiles [29]. Esta orientación al modelado de DSLs y no solo a la visualización genérica justifica su elección en el contexto de este trabajo.

En la Tabla 2.6 se resumen las principales diferencias entre las librerías analizadas.

Librería	Enfoque principal	Ventaja destacada	Limitación para este TFG
GoJS	Diagramas interactivos comerciales	Madurez, edición avanzada y layouts	Licencia comercial y enfoque generalista
JointJS	Editores visuales y aplicaciones low-code	Flexibilidad y uso de SVG	Requiere adaptación específica al dominio
React Flow	Editores basados en nodos en React	Simplicidad y gran ecosistema React	Integración no nativa con Angular
Mermaid	Diagramas como texto	Ideal para documentación versionable	No está orientado a edición visual interactiva
Cytoscape.js	Visualización y análisis de grafos	Potencia para redes complejas	Menos orientado a editores de modelos de dominio
D3.js	Visualización de datos personalizada	Máxima flexibilidad gráfica	Exige desarrollar manualmente la lógica de edición
Daga	Modelado visual de DSLs y modelos web	Integración con modelos, propiedades y restricciones semánticas	Librería más específica y dependiente del ecosistema Metadev

Tabla 2.6: Comparativa de librerías web de diagramación

A partir de esta comparación, Daga resulta especialmente adecuada para el proyecto porque no se limita a dibujar grafos, sino que ha sido creada para formar entornos de modelado. Esta diferencia es importante: el objetivo del sistema no es únicamente mostrar una red, sino permitir que el usuario construya y modifique un modelo probabilístico con significado propio. Cada nodo, conexión y propiedad visual debe corresponderse con una entidad del dominio de riesgos.

Por tanto, el uso de Daga permite alinear la representación visual con el DSL del sistema. El diagrama no actúa como una imagen estática ni como una simple visualización auxiliar, sino como la interfaz principal para crear, editar y comprender el modelo. Esta decisión tecnológica refuerza una de las ideas centrales del trabajo: en una herramienta de riesgos basada en grafos, el modelo debe ser visible, manipulable y comprensible directamente por el usuario.

2.3.3. Arquitecturas web modulares y frameworks reactivos

La tercera base tecnológica del proyecto es la arquitectura web modular. Una aplicación de modelado visual de riesgos debe ofrecer una interfaz interactiva, una representación gráfica compleja, cálculo probabilístico y mecanismos de importación, exportación y persistencia. Para evitar que todas estas responsabilidades queden acopladas en un único bloque, resulta necesario organizar el sistema en módulos diferenciados.

Las aplicaciones web modernas suelen organizarse siguiendo arquitecturas cliente-servidor, donde el frontend se encarga de la interacción con el usuario y el backend centraliza operaciones de negocio, persistencia o cálculo intensivo. En el caso de este proyecto, el frontend adquiere un papel especialmente relevante, ya que el usuario construye el modelo directamente sobre el navegador mediante un lienzo interactivo.

Angular constituye una opción adecuada para este tipo de sistemas por su orientación a aplicaciones escalables, su arquitectura basada en componentes, su integración con TypeScript y su ecosistema de herramientas. La documentación oficial lo define como un framework para construir aplicaciones web rápidas, fiables y escalables, mantenido por Google y acompañado de APIs, librerías y herramientas de desarrollo [30]. Además, Angular permite estructurar la interfaz en componentes reutilizables, servicios y módulos funcionales, lo que facilita separar responsabilidades como edición del diagrama, cálculo probabilístico, gestión de estado, importación/exportación y visualización de resultados. La documentación oficial destaca Angular como framework para aplicaciones escalables y con soporte moderno para características como componentes, servicios, enrutado, formularios, herramientas de desarrollo y organización modular.

El uso de TypeScript refuerza esta arquitectura al introducir tipado estático sobre JavaScript. Esto resulta especialmente relevante en una aplicación donde existen estructuras de datos complejas, como nodos, conexiones, tablas de probabilidad condicional, evidencias y archivos de persistencia. El tipado permite documentar explícitamente estas entidades, detectar errores en tiempo de desarrollo y mejorar la mantenibilidad del sistema [31]. Según la documentación oficial, TypeScript añade tipos a JavaScript y mejora el soporte de herramientas a escala.

Desde el punto de vista arquitectónico, la solución puede entenderse como una aplicación modular compuesta por los bloques que se especifican en la Tabla 2.7.

Bloque	Tecnologías asociadas	Responsabilidad principal
Frontend interactivo	Angular y TypeScript	Gestión de la interfaz, componentes visuales, interacción del usuario y coordinación del estado de la aplicación.
Motor de diagramación	Daga	Representación y edición visual de nodos, conexiones y propiedades del modelo.
DSL de riesgos	TypeScript / estructuras propias del dominio	Expresión formal de reglas, dependencias, probabilidades y evidencia.
Motor probabilístico	TypeScript, Node.js	Cálculo de probabilidades, propagación, inferencia y validación de modelos.
Persistencia e intercambio	JSON, RiskFile y CSV	Importación/exportación de modelos y datos externos.
Despliegue	Scripts de publicación y AWS S3	Generación y publicación de la aplicación web como artefacto estático accesible desde navegador.

Tabla 2.7: Bloques tecnológicos de la arquitectura propuesta

Esta organización modular aporta varias ventajas. En primer lugar, permite separar la representación visual del cálculo probabilístico, evitando que la lógica matemática quede mezclada con el código de interfaz. En segundo lugar, facilita la evolución del sistema: el motor de inferencia, el formato de persistencia o la interfaz gráfica pueden modificarse sin afectar necesariamente al resto de bloques. En tercer lugar, mejora la testabilidad, ya que las funciones de cálculo y validación pueden probarse de forma independiente respecto al lienzo visual.

Además, esta arquitectura favorece la escalabilidad funcional del sistema. En esta versión, la lógica principal se ejecuta en el navegador, lo que permite mantener una experiencia interactiva inmediata y reducir la dependencia de servicios externos. La separación entre componentes, servicios y utilidades facilita que futuras ampliaciones puedan incorporarse sin modificar la estructura básica de la aplicación.

Sin embargo, si en el futuro se incorporan modelos más complejos, redes de mayor tamaño o algoritmos computacionalmente costosos, estos podrían desplazarse parcial o totalmente a un backend especializado. De esta forma, el sistema mantendría una arquitectura preparada para crecer sin renunciar a la simplicidad inicial.

2.3.4. Justificación tecnológica de la solución propuesta

El análisis de las tecnologías anteriores permite justificar la elección de una solución basada en DSL, Daga y arquitectura web modular. Cada una de estas decisiones responde a una necesidad concreta del problema.

En primer lugar, el uso de un DSL permite representar reglas de riesgo y probabilidad de forma específica, evitando que el usuario tenga que trabajar directamente con código estadístico generalista. Esto reduce la complejidad conceptual y favorece que el modelo sea más comprensible, mantenible y alineado con el dominio del problema.

En segundo lugar, la integración con Daga permite convertir ese modelo en un artefacto visual editable. Frente a herramientas que solo generan gráficos o visualizaciones, Daga se orienta a la construcción de editores de modelos y DSLs, lo que encaja directamente con la naturaleza del proyecto [28,29]. El objetivo no es únicamente mostrar resultados, sino permitir que el usuario construya el propio modelo de riesgo mediante nodos y relaciones.

En tercer lugar, Angular y TypeScript proporcionan una base sólida para desarrollar una aplicación web interactiva, mantenible y escalable. La arquitectura basada en componentes permite aislar responsabilidades, mientras que el tipado estático de TypeScript ayuda a gestionar con mayor seguridad las estructuras internas del modelo [30,31].

Por último, la organización modular de la aplicación permite separar la interfaz, el modelo de dominio, la lógica probabilística y los mecanismos de persistencia. Esta separación mejora la mantenibilidad del sistema y facilita su evolución futura sin introducir complejidad innecesaria en la versión actual.

En conjunto, esta combinación tecnológica permite construir una herramienta coherente con los objetivos del TFG: una aplicación web capaz de representar riesgos como grafos, expresar reglas mediante un DSL, editar modelos visualmente y calcular probabilidades de forma dinámica. La principal aportación no se encuentra solo en el cálculo matemático, sino en la integración de dicho cálculo dentro de una experiencia visual, accesible y orientada al dominio.

2.4 Análisis del problema y conclusión del estado del arte

2.4.1. Brecha detectada en las herramientas actuales

El análisis realizado en las secciones anteriores permite identificar una brecha clara entre las necesidades actuales de la gestión del riesgo y las capacidades ofrecidas por muchas de las herramientas disponibles. En entornos caracterizados por la incertidumbre, la interdependencia y la propagación de efectos entre variables, una estimación de riesgos no puede limitarse a una tabla de datos, una matriz estática o una simulación aislada. Los eventos del mundo real deben traducirse a variables modeladas de forma estructurada, de manera que sea posible visualizar cómo un fallo, una perturbación o una nueva evidencia puede propagarse a través del sistema.

Desde una perspectiva conceptual, el riesgo ya no debe entenderse únicamente como una probabilidad individual asociada a un evento, sino como el resultado de un conjunto de relaciones entre variables que son inciertas. Esta visión resulta coherente con la definición moderna de riesgo como efecto de la incertidumbre sobre los objetivos, así como con la crítica a los enfoques excesivamente cualitativos basados en matrices de impacto y probabilidad [1, 12]. En este sentido, las representaciones estáticas resultan insuficientes para analizar sistemas donde los riesgos están conectados y donde la ocurrencia de un evento puede modificar la probabilidad de otros.

La revisión del software existente muestra que las herramientas tradicionales de análisis estadístico y simulación, como SPSS, Minitab, @RISK o Crystal Ball, son útiles para analizar datos, ejecutar simulaciones o generar informes cuantitativos. Sin embargo, su modelo de trabajo suele estar basado en tablas, hojas de cálculo, fórmulas o procedimientos estadísticos, lo que dificulta representar de forma explícita las relaciones causales entre variables. Estas herramientas pueden responder adecuadamente a la pregunta de "qué resultado es probable que ocurra", pero no siempre permiten comprender cómo se propaga la incertidumbre dentro del sistema ni por qué determinadas variables afectan a otras [14,15,18,20].

Por otro lado, las herramientas especializadas en redes bayesianas, inteligencia artificial probabilística y programación probabilística, como BayesiaLab, GeNIe o PyMC, sí permiten representar dependencias, incorporar evidencia y actualizar probabilidades de forma dinámica. Estas soluciones se apoyan en estructuras como los Grafos Dirigidos Acíclicos y en técnicas de inferencia bayesiana, lo que las convierte en alternativas mucho más adecuadas para modelar relaciones causales complejas [4, 7, 11, 13, 17]. No obstante, su adopción puede verse limitada por su complejidad técnica, su curva de aprendizaje o la necesidad de conocimientos avanzados en estadística, programación o modelado probabilístico.

Por lo tanto, el problema principal no es la falta de herramientas probabilísticas, sino la ausencia de una solución que reúna rigor matemático, representación visual, facilidad de uso y orientación al dominio del riesgo. En la práctica, existe una separación clara entre las herramientas que son accesibles pero tienen poca capacidad para representar relaciones causales y las herramientas que tienen una mejor capacidad para representar las relaciones causales, pero que son mucho menos accesibles por su dificultad o coste. Esta separación dificulta que usuarios no expertos puedan construir modelos probabilísticos comprensibles, trazables y fáciles de modificar sin recurrir a hojas de cálculo complejas, código especializado o software comercial avanzado.

2.4.2. Requisitos derivados del análisis

A partir de esta brecha, surge una serie de necesidades funcionales y técnicas para una herramienta moderna de modelado de riesgos. En primer lugar, el sistema debe permitir representar los riesgos como una red de elementos relacionados, y no como una colección aislada de registros. En segundo lugar, debe ofrecer una interfaz visual que facilite la comprensión de la estructura del modelo. En tercer lugar, debe permitir expresar relaciones probabilísticas y reglas de dependencia de forma clara. Por último, debe apoyarse en una arquitectura software modular que permita que el sistema evolucione, incorpore nuevos métodos de cálculo y mantenga separadas las responsabilidades de visualización, lógica de dominio e inferencia.

En la Tabla 2.8 se resumen los principales problemas detectados y los requisitos derivaados para la solución propuesta.

Problema detectado	**Requisito derivado**
Representación tabular o estática del riesgo	Representar los riesgos mediante grafos visuales formados por nodos y conexiones.
Dificultad para comprender relaciones causales	Mostrar explícitamente las dependencias entre variables y eventos.
Modelos ocultos en fórmulas, celdas o código	Convertir el modelo en un artefacto visual editable y comprensible.
Complejidad de las herramientas bayesianas avanzadas	Reducir la barrera de entrada mediante una interfaz intuitiva y un lenguaje adaptado al dominio.
Escasa trazabilidad del cálculo probabilístico	Vincular cada elemento visual con su significado probabilístico y sus reglas de cálculo.
Dificultad para evolucionar el sistema	Diseñar una arquitectura modular separando interfaz, diagramación, DSL y motor probabilístico.

Tabla 2.8: Problemas detectados y requisitos derivados para la solución propuesta

2.4.3. Solución propuesta

En respuesta a estas necesidades, este TFG propone el desarrollo de una aplicación web para el modelado visual y el cálculo de riesgos a través de una herramienta integrada por un DSL visual y una librería de diagramación. La aplicación permitirá al usuario construir modelos probabilísticos sobre un lienzo visual donde se puedan definir nodos, conexiones, pesos, probabilidades y evidencias sin la necesidad de trabajar con fórmulas dispersas o scripts complejos.

El uso de un DSL permite representar las reglas que son propias del dominio del riesgo de manera más directa que con un lenguaje general. Esta elección resulta coherente con la literatura sobre lenguajes de dominio específico, que destaca su utilidad cuando existe un conjunto estable de conceptos, reglas y operaciones propias de un área concreta [5, 9, 10]. En este caso, conceptos como evento, probabilidad, dependencia, evidencia o propagación forman parte del dominio y pueden ser encapsulados dentro de un lenguaje especializado.

La integración del modelo con Daga refuerza esta propuesta al permitir que el modelo se represente como un diagrama interactivo. En lugar de solamente mostrar el diagrama, la aplicación permite utilizarlo para editar el modelo. Cada elemento visual, como los nodos y conexiones, no actúa únicamente como un recurso gráfico, sino también como un elemento semántico asociado al cálculo probabilístico. Esto permite que el modelo sea visible, editable y comprensible. Así, se reduce la distancia entre la representación del concepto de riesgo y su implementación técnica [28, 29].

Desde el punto de vista arquitectónico, la solución se apoya en tecnologías web modernas, en concreto Angular y TypeScript, para desarrollar una aplicación modular, interactiva y mantenible. Angular permite organizar la interfaz mediante componentes y servicios, mientras que TypeScript ofrece tipado estático para representar de forma más segura las estructuras del dominio, como por ejemplo nodos, conexiones, tablas de probabilidad o archivos de persistencia [30, 31]. Esta base tecnológica permite que la herramienta pueda ejecutarse en navegador y evolucionar hacia una plataforma escalable.

2.4.4. Conclusión del estado del arte

La solución propuesta no intenta reemplazar herramientas avanzadas como PyMC, BayesiaLab o GeNIe cuando se trata de investigación estadística profunda o análisis causal a gran escala. En cambio, se enfoca en ofrecer una herramienta accesible, visual y orientada al dominio para construir modelos de riesgo comprensibles. En este sentido, el sistema busca acercar parte del rigor de los modelos bayesianos y causales a usuarios que necesitan analizar interdependencias, pero que no necesariamente poseen conocimientos avanzados de estadística o programación.

Como conclusión del estado del arte, se puede afirmar que existe una oportunidad clara para desarrollar herramientas que integren el modelado probabilístico, la visualización interactiva y las abstracciones específicas del dominio. Las soluciones tradicionales ofrecen facilidad de uso, pero limitan la representación causal. Las soluciones bayesianas avanzadas ofrecen rigor, pero elevan la barrera de entrada. La propuesta de este TFG se plantea como una respuesta intermedia a esa brecha: una aplicación web basada en grafos, DSL y tecnologías modernas que permita modelar, visualizar y calcular riesgos de forma más transparente y accesible.

Esta conclusión sirve de base para el diseño e implementación descritos en los capítulos posteriores. Una vez identificado el problema y justificada la necesidad tecnológica, el siguiente paso consiste en definir la arquitectura de la solución, los componentes principales del sistema y el modo en que el DSL, el motor probabilístico y la librería de diagramación serán integrados para construir una herramienta funcional de modelado visual de riesgos.

CAPÍTULO 3
Tecnología utilizada

En este capítulo se describen las principales tecnologías utilizadas para el desarrollo de la aplicación, justificando su elección en función de los objetivos del proyecto. La solución propuesta requiere una arquitectura web interactiva, capaz de representar modelos probabilísticos mediante grafos, actualizar los cálculos en tiempo real y mantener una estructura de código modular, mantenible y verificable.

3.1 Entorno de desarrollo

El proyecto se ha desarrollado como una aplicación web utilizando Angular[1] como framework principal. Esta elección responde a la necesidad de construir una interfaz rica e interactiva, donde el usuario pueda crear nodos, conectarlos, editar propiedades y visualizar los resultados probabilísticos directamente sobre el diagrama.

La gestión del proyecto se realiza mediante NPM (Node Package Manager)[2], que centraliza la instalación de dependencias y la ejecución de scripts de desarrollo, pruebas y construcción. Como herramienta de apoyo se utiliza Angular CLI, que facilita la creación de componentes, la ejecución del servidor local y la generación de versiones de producción.

En la Tabla 3.1 se muestran las versiones utilizadas en cada una de las tecnologías a la hora de desarrollar el proyecto, además de su propósito.

Durante el desarrollo, la aplicación se ejecuta localmente mediante el comando `npm start`, quedando disponible en `http://localhost:4200`. En la Figura 3.1 se muestra cómo se ve la aplicación al iniciarla.

[1] `https://angular.dev/`

[2] NPM es el gestor de paquetes predeterminado para el entorno de ejecución JavaScript Node.js

Tecnología	Versión	Uso en el proyecto
npm	11.11.0	Gestión de dependencias y scripts
Angular CLI	^21.2.1	Compilación, servidor de desarrollo y scaffolding
TypeScript	~5.9.2	Lenguaje principal
Angular	^21.2.0	Framework principal de la aplicación

Tabla 3.1: Tecnologías base del proyecto, versiones y su propósito

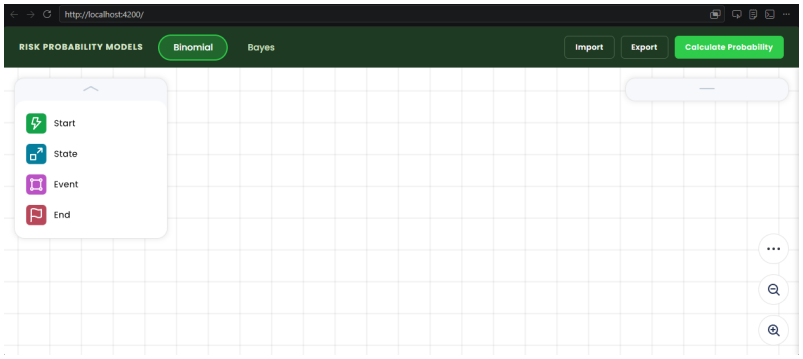

Figura 3.1: Ejemplo visual de la aplicación al iniciarla

3.2 Lenguajes de programación

El lenguaje principal utilizado es TypeScript, una extensión tipada de JavaScript. Su elección se justifica por su integración nativa con Angular y por las ventajas que ofrece el tipado estático en un proyecto donde se manejan estructuras complejas, como nodos, conexiones, probabilidades, evidencias y tablas de probabilidad condicional.

El uso de TypeScript permite definir modelos de datos más seguros y reducir errores durante el desarrollo. Además, facilita la mantenibilidad del código, especialmente en las partes relacionadas con la lógica probabilística y la comunicación entre componentes, las cuales son muy importantes en la aplicación.

También se emplean HTML y CSS/SCSS para definir la estructura visual de los componentes y los estilos de la interfaz.

3.3 Framework principal: Angular

Angular se utiliza como base para construir la interfaz de usuario de la aplicación. Su arquitectura basada en componentes permite dividir el sistema en piezas reutilizables y mantener separadas las distintas responsabilidades de la aplicación.

En este proyecto, Angular se emplea para gestionar la interfaz, los modos de modelado, los paneles de configuración y la integración con el lienzo de diagramación. Además, se utilizan componentes standalone[3], lo que simplifica la organización interna y reduce la dependencia de módulos tradicionales.

Otra razón importante para su elección es su integración con RxJS, que permite gestionar flujos de eventos de forma reactiva. Esto resulta especialmente útil en una herramienta visual, ya que cada cambio realizado por el usuario sobre el diagrama debe reflejarse inmediatamente en el estado del modelo y en los cálculos mostrados.

3.4 Librería de diagramación: Daga

Una de las tecnologías más relevantes del proyecto es Daga, desarrollada por Metadev. Se utilizan los paquetes "@metadev/dagaz "@metadev/daga-angular", que permiten integrar el motor de diagramación dentro de la aplicación de Angular.

Daga se ha elegido porque el objetivo principal del proyecto es representar el modelo de riesgo como un grafo visual editable. A diferencia de las hojas de cálculo o de los scripts estadísticos, en esta aplicación el modelo no queda oculto en fórmulas, sino que se muestra directamente como un conjunto de nodos y conexiones.

En la Tabla 3.2 se muestra la versión utilizada de Daga.

Biblioteca	**Versión**	**Uso**
@metadev/daga	^5.0.5	Motor de diagramas
@metadev/daga-angular	^5.0.5	Integración con Angular

Tabla 3.2: Librerías de modelado visual y diagramación

Sobre este lienzo visual se implementan los dos modos principales de trabajo visibles en la Tabla 3.3

Gracias a esta integración, el usuario puede construir modelos de riesgo de forma visual, modificar sus parámetros y observar cómo cambian los resultados sobre el propio diagrama.

[3]Funcionalidad moderna de Angular que permite crear componentes que no necesitan ser declarados en ningún módulo (NgModule) para funcionar.

Modo	Descripción
Binomial	Permite representar árboles de probabilidad mediante conexiones ponderadas
Bayes	Permite modelar redes bayesianas con tablas de probabilidad condicional y evidencias

Tabla 3.3: Modos de modelado probabilístico soportados

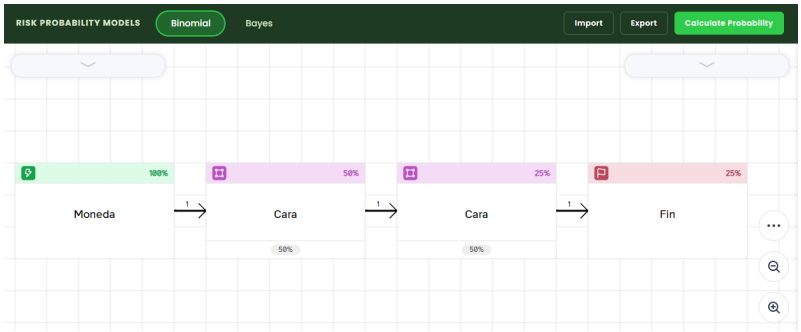

Figura 3.2: Ejemplo de un diagrama en la aplicación

En la Figura 3.2 se muestra un ejemplo de un diagrama en la aplicación.

3.5 Programación reactiva con RxJS

La biblioteca RxJS se utiliza para gestionar eventos y flujos de datos dentro de la aplicación. En una herramienta basada en interacción visual, los cambios del usuario son constantes: creación de nodos, modificación de conexiones, edición de probabilidades o activación de evidencias.

Mediante observables, la aplicación puede reaccionar ante estos cambios y actualizar el modelo interno de forma ordenada. Esto permite mantener sincronizada la representación visual con la lógica de cálculo, evitando inconsistencias entre lo que el usuario ve en pantalla y los datos que utiliza el motor probabilístico.

En la Tabla 3.4 se muestran las versiones de la biblioteca RxJs y de la librería auxiliar para TypeScript.

Biblioteca	Versión	Uso
rxjs	~7.8.0	Gestión reactiva de eventos y cambios
tslib	^2.3.0	Funciones auxiliares de ejecución para TypeScript

Tabla 3.4: Librerías auxiliares y de gestión reactiva

3.6 Motor de cálculo probabilístico

La lógica del cálculo probabilístico se implementa en TypeScript dentro de la propia aplicación. Esto permite que los cálculos se realicen directamente en el navegador, ofreciendo una experiencia interactiva sin depender constantemente de un servidor externo.

En el modo Binomial, el sistema trabaja con árboles de probabilidad ponderados. Cada conexión posee un peso asociado y la aplicación recalcula automáticamente los valores afectados cuando el usuario modifica el modelo.

En el modo de Bayes, la aplicación permite construir redes bayesianas binarias. Cada nodo puede disponer de una tabla de probabilidad condicional, y el usuario puede fijar evidencias para observar cómo se actualizan las probabilidades marginales del resto de variables.

En la Tabla 3.5 se muestran las dos estrategias principales utilizadas para el motor de inferencia en el desarrollo.

Técnica	**Uso**
Inferencia exacta por enumeración	Utilizada en redes pequeñas, con límite de 20 nodos
Monte Carlo por likelihood weighting	Utilizado en redes grandes donde la inferencia exacta resulta costosa

Tabla 3.5: Técnicas de inferencia probabilística utilizadas

Además, el sistema incorpora mecanismos de aprendizaje de parámetros mediante MLE[4] y EM[5], que permiten estimar probabilidades a partir de datos importados en formato CSV. De este modo, la herramienta no solo permite crear modelos manualmente, sino también apoyarse en datos externos para configurarlos.

[4]Método que busca los valores de los parámetros que hacen que los datos observados sean lo más probables posible.

[5]Algoritmo iterativo diseñado para estimar parámetros cuando los datos están incompletos o tienen variables ocultas.

3.7 Persistencia e importación/exportación

Para guardar y compartir modelos se utiliza un formato propio denominado RiskFile, basado en JSON y versionado mediante el campo riskFileVersion".

Este formato almacena en un único archivo la estructura del diagrama y el estado probabilístico del modelo. En el caso del modo Bayes, también permite guardar las tablas de probabilidad condicional y las evidencias configuradas.

La elección de JSON se justifica por ser un formato abierto, legible y fácil de versionar. Esto facilita la trazabilidad del modelo y permite auditar o compartir los archivos generados por la aplicación.

La aplicación también permite importar datos desde CSV, incluyendo la posibilidad de definir relaciones entre variables. Esta funcionalidad facilita la creación de modelos a partir de datos externos y conecta el modelado visual con fuentes de información reales.

3.8 Herramientas de pruebas

Para validar el funcionamiento del sistema se utilizan pruebas unitarias y pruebas de extremo a extremo.

Las pruebas unitarias se ejecutan mediante Vitest, junto con "jsdom"para simular un entorno DOM[6] en Node.js. Estas pruebas permiten comprobar funciones concretas del sistema, especialmente aquellas relacionadas con cálculo, transformación de datos y lógica auxiliar.

En la Tabla 3.6 se muestran las versiones de las herramientas de testing unitario y simulación de entorno.

Herramienta	Versión	Uso
Vitest	~4.0.18	Ejecución de pruebas unitarias
jsdom	^28.0.0	Simulación del DOM en entorno Node.js

Tabla 3.6: Herramientas de testing y simulación de entorno

Para las pruebas end-to-end se utiliza Cypress, que permite comprobar la aplicación desde el punto de vista del usuario. Esta herramienta resulta especialmente adecuada porque el valor principal del proyecto reside en la interacción visual con el diagrama.

En la Tabla 3.7 se muestran las herramientas para pruebas E2E utilizadas durante el desarrollo.

En la Figura 3.3 se muestra un ejemplo de uno de los test E2E de la aplicación.

[6]Son las siglas de Document Object Model (Modelo de Objetos del Documento).

Herramienta	Versión	Uso
Cypress	15.14.1	Pruebas end-to-end
eslint-plugin-cypress	6.3.1	Reglas específicas para pruebas Cypress

Tabla 3.7: Herramientas para pruebas End-to-End (E2E)

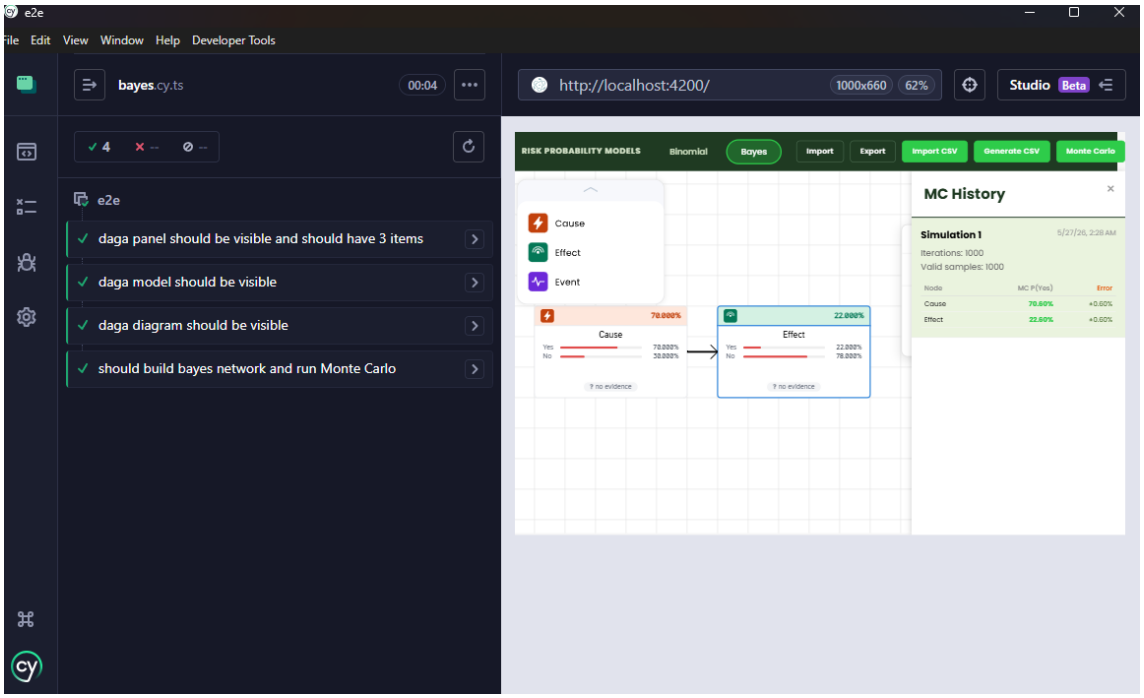

Figura 3.3: Ejemplo de Test E2E

3.9 Calidad de código: ESLint y Prettier

Para mantener un estilo de código homogéneo y detectar errores de forma temprana se utilizan ESLint[7] y Prettier[8].

ESLint se encarga del análisis estático del código, incorporando reglas específicas para Angular y TypeScript. Prettier se utiliza para el formateo automático, evitando diferencias de estilo entre archivos y facilitando la lectura del código.

En la Tabla 3.8 se especifican las versiones y el uso de las herramientas de análisis estático y formateo de código.

[7] https://eslint.org/
[8] https://prettier.io/

Herramienta	Versión	Uso
ESLint	^10.0.3	Análisis estático del código
angular-eslint	21.3.1	Reglas específicas de Angular
typescript-eslint	8.56.1	Reglas específicas de TypeScript
Prettier	^3.8.1	Formateo automático
eslint-config-prettier	^10.1.8	Compatibilidad ESLint/Prettier
eslint-plugin-prettier	^5.5.5	Integración de Prettier con ESLint

Tabla 3.8: Herramientas de análisis estático y formateo de código

3.10 Control de versiones e integración continua

El control de versiones se realiza mediante Git, utilizando GitHub como repositorio remoto. Esto permite registrar la evolución del proyecto, revisar cambios y mantener una trazabilidad clara durante el desarrollo.

Además, el proyecto incorpora integración continua mediante GitHub Actions[9], definida en un archivo en ".github/workflows". Este flujo se ejecuta automáticamente en cada push o pull request sobre la rama main.

En la Tabla 3.9 se explican los jobs definidos en el flujo de integración continua.

Job	Descripción
build	Instala dependencias, comprueba formato, compila el proyecto y ejecuta pruebas unitarias
e2e	Ejecuta pruebas end-to-end con Cypress en Electron

Tabla 3.9: Jobs definidos en el flujo de integración continua

En caso de fallo en las pruebas end-to-end, se almacenan artefactos como capturas de pantalla o vídeos, lo que facilita la detección y corrección de errores.

[9]Plataforma de automatización integrada directamente en GitHub que permite crear flujos de trabajo personalizados para compilar, probar y desplegar código directamente desde el repositorio.

3.11 Despliegue

El proyecto incluye scripts de despliegue mediante "scripts/publish-pre.sh" y "scripts/publish-pro.sh". Estos scripts generan la versión de producción de la aplicación y sincronizan los archivos resultantes con AWS S3.

Esta estrategia de despliegue es adecuada para una aplicación web como dsl-riesgos, ya que permite distribuirla de forma sencilla y acceder a ella desde un navegador sin requerir instalación local por parte del usuario final.

3.12 Síntesis de la elección tecnológica

La selección tecnológica realizada permite construir una aplicación web visual, modular y mantenible. Angular y TypeScript proporcionan una base sólida para la interfaz y la lógica de la aplicación, mientras que Daga permite representar los modelos probabilísticos como grafos editables. RxJS facilita la actualización reactiva del sistema, y las herramientas de pruebas, control de versiones e integración continua contribuyen a mejorar la fiabilidad del desarrollo.

En conjunto, estas tecnologías permiten materializar el objetivo del proyecto: ofrecer una herramienta accesible para modelar riesgos, calcular probabilidades y visualizar la propagación de la incertidumbre en sistemas complejos.

CAPÍTULO 4
Desarrollo de la solución

En este capítulo se describe el proceso seguido para desarrollar la aplicación, desde la organización metodológica del trabajo hasta la definición de requisitos y el diseño de la interfaz de usuario. La solución desarrollada tiene como objetivo proporcionar una herramienta web para construir modelos visuales de riesgo, trabajar con grafos de probabilidad y mostrar los resultados directamente sobre el diagrama. Esta idea encaja con el planteamiento general del proyecto: unir la lógica matemática y la representación visual mediante Angular, Daga y un lenguaje específico para el dominio del riesgo.

4.1 Metodología

Para el desarrollo del proyecto se siguió una adaptación ágil de la metodología utilizada en la asignatura Proyecto de Ingeniería de Software (PIN). Esta metodología combina el enfoque del método *Lean Startup*, basado en la construcción y validación progresiva de un producto mínimo viable, con prácticas propias de metodologías ágiles, como la organización del trabajo en Backlog, unidades de trabajo, Sprints, revisiones periódicas y pruebas de aceptación.

En su planteamiento original, la metodología propone definir una línea de trabajo, preparar un Backlog inicial y organizar el desarrollo mediante un Sprint 0 de puesta en marcha y varios Sprints de implementación. Al final de cada Sprint, se revisa el trabajo realizado, se valida el incremento desarrollado y se ajustan las siguientes tareas en función de los resultados obtenidos.

En este trabajo fue necesario adaptar dicha metodología al contexto de un proyecto individual. La línea de trabajo definida fue la aplicación *Risk*, entendida como un MVP orientado a validar la viabilidad técnica de una herramienta web para el modelado visual y cálculo probabilístico de riesgos. El rol de equipo de desarrollo fue asumido por el autor del trabajo, mientras que la validación del alcance y de las decisiones funcionales se realizó con el apoyo de los tutores académico y de empresa.

La metodología original contempla roles como Product Owner, Scrum Master, Early Adopters y colaboradores. En este proyecto, estos roles se adaptaron de la forma que se muestra en la Tabla 4.1.

Rol metodológico	Adaptación en el TFG
Product Owner	Responsabilidad compartida entre el autor y los tutores, especialmente en la definición del alcance funcional
Equipo de desarrollo	Autor del TFG
Scrum Master / seguimiento	Seguimiento periódico mediante reuniones con los tutores
Early Adopters	Tutores y personas de Metadev que revisaron la evolución del producto
Colaboradores externos	Apoyo puntual en diseño de interfaz por parte de Paco Soria, diseñador de Metadev

Tabla 4.1: Adaptación de los roles metodológicos en el contexto del TFG

El desarrollo del TFG se llevó a cabo en el marco de unas prácticas de empresa en Metadev, realizadas desde inicios de marzo hasta mediados de mayo. Durante este periodo se trabajó de forma continuada en el análisis, diseño, implementación y validación de la aplicación, combinando las tareas propias del desarrollo del producto con la elaboración progresiva de la memoria del trabajo.

La dedicación al proyecto fue de aproximadamente unas 440 horas. Este tiempo incluye tareas de análisis del problema, formación técnica, diseño de la solución, programación, pruebas, corrección de errores, despliegue y documentación.

Además, dentro de los Sprints se reservó tiempo específico para la redacción de la memoria del TFG. En los primeros Sprints se dedicaron aproximadamente 15 horas por Sprint a tareas de documentación, mientras que en el último Sprint esta dedicación aumentó hasta unas 30 horas, debido a la necesidad de cerrar la redacción, preparar ejemplos de uso, revisar el contenido técnico y consolidar la versión final del proyecto.

4.1.1. Adaptación del workflow

La metodología de referencia define un workflow de desarrollo compuesto por estados como Registrar, Esperar Prioridad, Especificar Requisitos, Diseñar y Estimar, Esperar Sprint, Programar, Aplicar Pruebas de Aceptación y Terminado. Este workflow permite controlar el avance de cada unidad de trabajo y visualizar el estado del desarrollo mediante un tablero Kanban.

En el contexto de este TFG se utilizó Trello como herramienta de seguimiento. Esta adaptación no modificó la metodología aplicada, sino únicamente la herramienta utilizada para representar el flujo de trabajo. Trello permitió crear un tablero Kanban con columnas equivalentes a los estados principales del workflow.

La correspondencia utilizada se muestra en la Tabla 4.2.

Elemento metodológico	**Adaptación en Trello**
Backlog	Lista de tarjetas pendientes
Unidad de Trabajo	Tarjeta de Trello
Workflow	Columnas del tablero Kanban
Sprint	Agrupación temporal de tarjetas planificadas
Pruebas de aceptación	Checklist o descripción dentro de cada tarjeta
Seguimiento del avance	Revisión periódica del tablero

Tabla 4.2: Correspondencia de elementos metodológicos con la herramienta Trello

Las tarjetas de Trello se utilizaron para registrar nuevas funcionalidades, mejoras, correcciones de fallos y tareas técnicas. Cada tarjeta incluía, cuando correspondía, una descripción breve, criterios de aceptación, comentarios de avance y anotaciones sobre errores detectados o decisiones de implementación.

Un ejemplo de cómo se trabajaba en Trello lo podemos ver en la Figura 4.1.

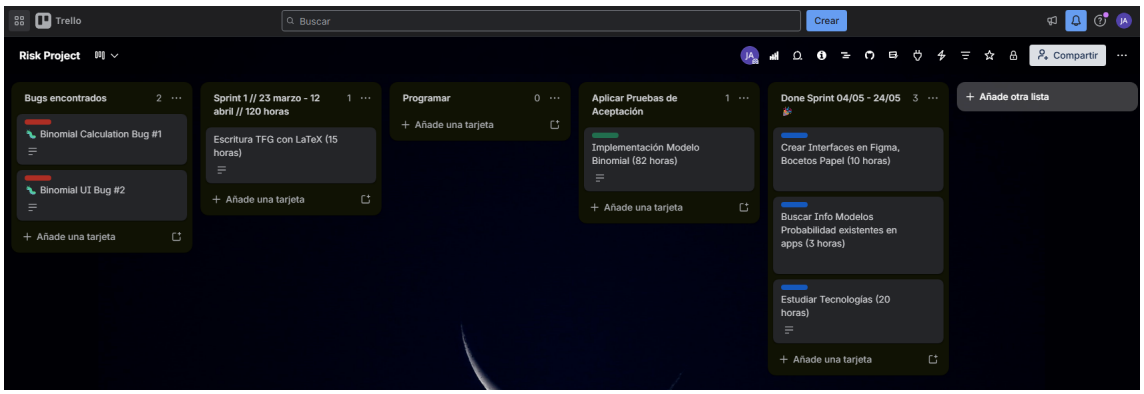

Figura 4.1: Estado del tablero de Trello durante el del Sprint 1

4.1.2. Organización temporal de los sprints

El desarrollo se dividió en un Sprint 0 de puesta en marcha y tres Sprints de implementación. Esta organización permitió comenzar con una fase inicial de estudio y preparación, seguida de iteraciones orientadas a construir progresivamente la aplicación. Además del trabajo técnico, cada Sprint incluyó una dedicación específica a la documentación de la memoria, con mayor intensidad en la fase final del proyecto.

Sprint	Fechas	Objetivo principal	Trabajo realizado
Sprint 0	9 marzo – 22 marzo	Puesta en marcha	Estudio del dominio, instalación del entorno, análisis inicial de Daga, definición del Backlog y aprendizaje de tecnologías
Sprint 1	23 marzo – 12 abril	Base visual y modo binomial	Integración inicial de Angular con Daga, tutorial de Metadev, creación de nodos y conexiones, estudio de propiedades de Daga, primeros bocetos de interfaz e implementación inicial del modo binomial
Sprint 2	13 abril – 3 mayo	Cálculo probabilístico y modo Bayes	Propagación de probabilidades, modo Bayes, CPTs, evidencias, pruebas unitarias, pruebas E2E con Cypress, incorporación de linter, auto-layout para Bayes y corrección de errores del modo binomial
Sprint 3	4 mayo – 24 mayo	Consolidación y entrega	Rediseño completo de la interfaz con apoyo de Paco Soria, importación/exportación, ampliación de pruebas, refactorización, CI/CD, despliegue, escritura de la memoria, corrección de fallos de Bayes y elaboración de ejemplos de uso

Tabla 4.3: Planificación y resumen del trabajo realizado por Sprint

Esta planificación permitió compatibilizar el desarrollo técnico de la aplicación con la elaboración progresiva de la memoria. No obstante, la carga documental se concentró especialmente en el último Sprint, al coincidir con la programación final de la aplicación, la preparación de ejemplos demostrativos y la redacción de una versión completa de la memoria. La revisión final con el tutor,

los ajustes formales y la preparación de la defensa quedaron como tareas posteriores de cierre académico.

4.1.3. Sprint 0: puesta en marcha

El Sprint 0 tuvo como objetivo preparar las condiciones necesarias para iniciar el desarrollo. Durante esta fase se estudió el dominio del problema, centrado en el modelado visual de riesgos y el cálculo de probabilidades. También se instaló el entorno de desarrollo y se realizó un primer análisis de la librería Daga, que sería la base visual de la aplicación.

Además, se definió el Backlog inicial del proyecto, identificando las funcionalidades principales que debía incorporar la primera versión: creación de nodos, creación de conexiones, edición de probabilidades, modo binomial, modo bayesiano, importación/exportación y pruebas. Esta fase también incluyó un periodo de formación técnica, ya que era la primera vez que se trabajaba con varias de las tecnologías utilizadas.

4.1.4. Sprint 1: integración inicial y modo binomial

Durante el Sprint 1 se realizó la primera integración funcional entre Angular y Daga. Para ello, se siguió un tutorial proporcionado por Metadev en la documentación de Daga, lo que permitió comprender el funcionamiento básico del lienzo, la creación de nodos, la creación de conexiones y la gestión de propiedades.

En esta iteración se implementó el primer modo de trabajo de la aplicación: el modo binomial. Este modo permite representar árboles o estructuras de probabilidad mediante conexiones ponderadas. También se realizaron bocetos iniciales en papel para definir la idea general de la interfaz y del flujo de interacción.

De forma paralela, se continuó estudiando Angular, TypeScript y las tecnologías asociadas al desarrollo web, con el objetivo de mejorar la calidad de la implementación y sentar una base sólida para los siguientes Sprints.

4.1.5. Sprint 2: propagación de probabilidades y modo Bayes

El Sprint 2 se centró en la lógica probabilística de la aplicación. En primer lugar, se implementó la propagación de probabilidades dentro del modo binomial, de forma que los cambios introducidos por el usuario en el diagrama se reflejasen automáticamente en los resultados.

Posteriormente, se implementó el modo Bayes, orientado al modelado mediante redes bayesianas binarias. Este modo incorporó tablas de probabilidad condicional, conocidas como CPTs, y la posibilidad de fijar evidencias sobre determinados nodos. De este modo, la aplicación comenzó a permitir no solo la edición visual del grafo, sino también la actualización dinámica de probabilidades marginales.

En esta fase también se incorporaron pruebas unitarias y pruebas end-to-end mediante Cypress. Además, se configuró el linter del proyecto, lo que permitió detectar diversos problemas de estilo y calidad del código. Algunos de estos avisos evidenciaron errores o malas prácticas que fueron corregidas durante el propio Sprint.

Otro trabajo relevante fue la creación de un sistema de auto-layout para el modo Bayes. Esta funcionalidad fue necesaria porque, al importar datos desde CSV, los nodos se generaban correctamente, pero aparecían separados y sin una distribución visual adecuada. El auto-layout permitió organizar los nodos de forma más comprensible dentro del lienzo.

Finalmente, se corrigieron varios errores detectados en el modo binomial durante el uso y prueba de la aplicación.

4.1.6. Sprint 3: consolidación, rediseño y despliegue

El Sprint 3 se dedicó a consolidar la aplicación y preparar una versión más completa y presentable. Una de las tareas principales fue el rediseño de la interfaz de usuario. Para ello, se contó con los consejos de Paco Soria, diseñador de Metadev, quien ayudó a reestructurar visualmente la aplicación y mejorar su apariencia general.

También se implementaron las funcionalidades de importación y exportación de modelos, permitiendo guardar y recuperar el estado de la aplicación. En paralelo, se actualizaron y ampliaron las pruebas unitarias y end-to-end para cubrir las nuevas funcionalidades.

Durante esta fase se realizó una refactorización del código, separando responsabilidades y mejorando la organización interna del proyecto. Asimismo, se configuró el flujo de integración continua y despliegue, lo que permitió automatizar parte del proceso de validación y publicación.

Por último, se corrigieron varios errores detectados en el modo Bayes y se redactaron ejemplos contextualizados para mostrar el funcionamiento de la aplicación tanto en el modo binomial como en el modo bayesiano.

4.2 Requisitos

La definición de requisitos se realizó de forma incremental a partir del Backlog inicial y de las unidades de trabajo planificadas durante los Sprints. Aunque la especificación mediante casos de uso no forma parte estricta de una metodología ágil, en este apartado se incluye un diagrama de casos de uso para representar de forma global las características principales de la aplicación. En este contexto, los casos de uso se interpretan como una vista de alto nivel de las épicas funcionales del sistema.

Risk está orientada a un usuario que desea construir y analizar modelos visuales de riesgo. El usuario trabaja sobre un lienzo, crea nodos y conexiones, selecciona el modo de cálculo y observa los resultados probabilísticos directamente sobre el diagrama. El flujo general de uso contempla que el usuario elija entre modo Binomial o Bayes, dibuje nodos, los conecte, configure probabilidades o evidencias, y pueda importar o exportar modelos mediante CSV o RiskFile.

4.2.1. Actores del sistema

Los actores identificados son los que se pueden ver en la Tabla 4.4

Actor	Descripción
Usuario / Analista	Persona que crea, edita y analiza modelos de riesgo dentro de la aplicación
Sistema de archivos	Medio externo desde el que se importan datos o modelos y hacia el que se exportan archivos

Tabla 4.4: Actores que interactúan con el sistema

El actor principal es el Usuario / Analista, ya que todas las funcionalidades están orientadas a permitir que construya, modifique y comprenda modelos probabilísticos de forma visual.

4.2.2. Diagrama de casos de uso

El diagrama que se muestra en la Figura 4.2 representa los principales casos de uso de la aplicación.

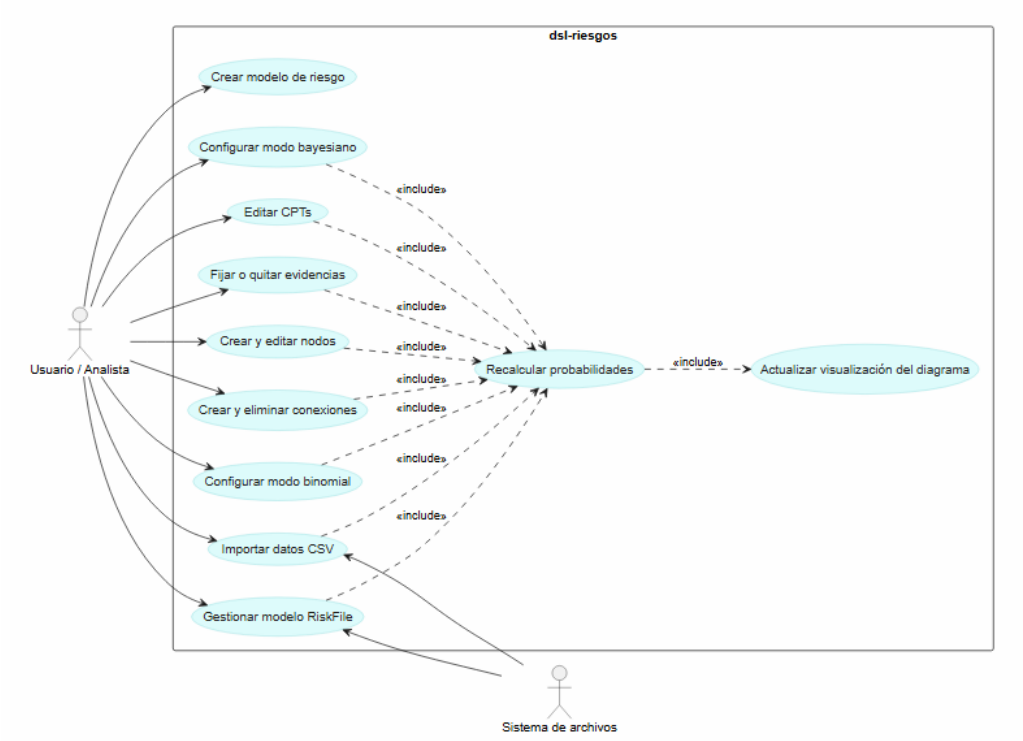

Figura 4.2: Diagrama de los Casos de Uso

4.2.3. Descripción de los casos de uso

En la Tabla 4.5 y en la Tabla 4.6 se describen los casos de uso del sistema.

Código	Caso de uso	Descripción
CU-01	Crear modelo de riesgo	El usuario inicia un nuevo modelo visual sobre el lienzo de trabajo.
CU-02	Crear y editar nodos	El usuario añade, modifica o elimina nodos del diagrama, definiendo los elementos principales del modelo de riesgo.
CU-03	Crear y eliminar conexiones	El usuario establece o elimina relaciones dirigidas entre nodos para representar dependencias o caminos probabilísticos.
CU-04	Configurar modo binomial	El usuario asigna pesos o probabilidades a las conexiones para representar escenarios de ramificación y caminos probabilísticos.
CU-05	Configurar modo bayesiano	El usuario trabaja con una red bayesiana formada por nodos, relaciones dirigidas, CPTs y evidencias configurables.
CU-06	Editar CPTs	El usuario define o modifica las tablas de probabilidad condicional asociadas a los nodos del modelo bayesiano.

Tabla 4.5: Casos de uso del sistema

Código	Caso de uso	Descripción
CU-07	Fijar o quitar evidencias	El usuario marca un nodo como evidencia positiva o negativa, o elimina una evidencia previamente fijada.
CU-08	Recalcular probabilidades	El sistema actualiza automáticamente los valores probabilísticos cuando se modifican nodos, conexiones, pesos, CPTs, evidencias o datos importados.
CU-09	Actualizar visualización del diagrama	El sistema repinta sobre el diagrama las probabilidades teóricas, marginales bayesianos, pesos de conexión y evidencias fijadas.
CU-10	Importar datos CSV	El usuario importa datos externos para alimentar el modelo, definir estructuras mediante relaciones y aprender parámetros mediante MLE o EM cuando procede.
CU-11	Gestionar modelo RiskFile	El usuario importa o exporta modelos mediante el formato propio RiskFile, conservando la estructura visual, las probabilidades, las CPTs, las evidencias y la información necesaria para reconstruir el modelo posteriormente.

Tabla 4.6: Casos de uso del sistema, segunda parte

4.2.4. Requisitos funcionales detallados

Los requisitos funcionales describen las funcionalidades que debe ofrecer el sistema. Estos requisitos son descritos en las tablas: Tabla 4.7 y Tabla 4.8.

Código	CU Relacionados	Requisito funcional
RF-01	CU-01	El sistema debe permitir crear un nuevo modelo visual de riesgo sobre un lienzo de trabajo.
RF-02	CU-02	El sistema debe permitir crear, editar, mover y eliminar nodos en el lienzo.
RF-03	CU-03	El sistema debe permitir crear y eliminar conexiones dirigidas entre nodos.
RF-04	CU-04, CU-05	El sistema debe permitir trabajar con modelos en modo Binomial y en modo Bayesiano.
RF-05	CU-04	El sistema debe permitir asignar pesos o probabilidades a conexiones en modo Binomial.
RF-06	CU-04, CU-08	El sistema debe recalcular automáticamente las probabilidades del modelo binomial cuando se modifiquen nodos, conexiones o pesos.
RF-07	CU-05	El sistema debe permitir crear redes bayesianas binarias en modo Bayesiano.
RF-08	CU-06	El sistema debe permitir editar tablas de probabilidad condicional asociadas a nodos bayesianos.
RF-09	CU-07	El sistema debe permitir fijar y eliminar evidencias positivas o negativas en nodos del modelo bayesiano.
RF-10	CU-06, CU-07, CU-08	El sistema debe recalcular automáticamente las probabilidades marginales al modificar CPTs, evidencias, nodos o conexiones.
RF-11	CU-09	El sistema debe mostrar probabilidades, marginales, pesos de conexión y evidencias directamente sobre el diagrama.

Tabla 4.7: Requisitos funcionales detallados del sistema

Código	CU Relacionados	Requisito funcional
RF-12	CU-10	El sistema debe permitir importar datos desde archivos CSV.
RF-13	CU-10	El sistema debe permitir definir o inferir relaciones entre variables a partir de datos importados, incluyendo directivas de estructura cuando estén disponibles.
RF-14	CU-10	El sistema debe permitir aprender parámetros probabilísticos desde CSV mediante técnicas como MLE o EM cuando proceda.
RF-15	CU-10, CU-09	El sistema debe permitir organizar automáticamente nodos importados mediante un auto-layout o layout causal.
RF-16	CU-11	El sistema debe permitir exportar modelos en formato RiskFile.
RF-17	CU-11	El sistema debe permitir importar modelos previamente guardados en formato RiskFile.
RF-18	CU-11	El sistema debe guardar en el RiskFile el estado visual y probabilístico del modelo, incluyendo diagrama, pesos, CPTs y evidencias.
RF-19	CU-04, CU-06	El sistema debe validar y normalizar los valores de probabilidad introducidos por el usuario.
RF-20	CU-10	El sistema debe normalizar los valores importados desde CSV, aceptando variantes equivalentes como sí/no, yes/no, 1/0 o true/false.
RF-21	CU-05, CU-08	El sistema debe aplicar inferencia exacta en redes bayesianas pequeñas y recurrir a inferencia aproximada mediante Monte Carlo cuando el tamaño de la red lo requiera.
RF-22	CU-05, CU-08	El sistema debe permitir generar o utilizar datos sintéticos para validar modelos y comprobar el funcionamiento del sistema.

Tabla 4.8: Requisitos funcionales detallados del sistema, segunda parte

4.2.5. Requisitos no funcionales

Los requisitos no funcionales recogen restricciones de calidad, usabilidad, mantenimiento y ejecución.

Estos requisitos son descritos en las tablas: Tabla 4.9 y Tabla 4.10.

Código	Requisito no funcional	Verificación
RNF-01	La aplicación debe ejecutarse en un navegador web moderno, sin requerir instalación local de software adicional por parte del usuario final.	Ejecución local con `npm start` y acceso desde navegador.
RNF-02	El proyecto debe poder construirse en modo producción sin errores de compilación.	Ejecución del comando de construcción del proyecto.
RNF-03	La interfaz debe presentar de forma visible los elementos principales del modelo, incluyendo nodos, conexiones, probabilidades, pesos y evidencias cuando correspondan.	Pruebas de aceptación y revisión visual de los casos de uso principales.
RNF-04	Las modificaciones habituales del modelo, como crear nodos, editar conexiones, cambiar probabilidades o fijar evidencias, deben reflejarse en el diagrama sin necesidad de recargar la aplicación.	Pruebas end-to-end y pruebas manuales sobre modelos de ejemplo.
RNF-05	El sistema debe informar al usuario cuando se introduzcan valores probabilísticos no válidos o inconsistentes.	Pruebas unitarias y pruebas manuales de validación de formularios.
RNF-06	El código debe separar la interfaz de usuario, la lógica de diagramación, la lógica probabilística y las utilidades de importación/exportación en componentes, servicios o módulos diferenciados.	Revisión de la estructura del código y de la organización del proyecto.
RNF-07	Las funciones principales de cálculo, normalización, importación y exportación deben estar organizadas de forma que puedan probarse de manera independiente.	Ejecución de pruebas unitarias sobre módulos y utilidades del sistema.

Tabla 4.9: Requisitos no funcionales aplicables al sistema

Código	Requisito no funcional	Verificación
RNF-08	El sistema debe validar y normalizar los valores de probabilidad introducidos por el usuario, evitando valores fuera del rango permitido.	Pruebas unitarias sobre la lógica de validación y normalización.
RNF-09	El sistema debe normalizar los valores importados desde CSV, aceptando variantes equivalentes como sí/no, yes/no, 1/0 o true/false cuando proceda.	Pruebas unitarias y pruebas de importación con archivos CSV de ejemplo.
RNF-10	Los modelos exportados en formato RiskFile deben almacenarse en un formato JSON legible y versionado.	Inspección del archivo exportado y pruebas de exportación.
RNF-11	El formato RiskFile debe permitir restaurar un modelo previamente guardado, conservando su estructura visual y su estado probabilístico principal.	Pruebas de exportación e importación con modelos de ejemplo.
RNF-12	El proyecto debe incluir pruebas unitarias y pruebas end-to-end ejecutables desde los scripts definidos en el entorno de desarrollo.	Ejecución de los comandos de pruebas definidos en el proyecto.
RNF-13	El proyecto debe integrarse con herramientas de análisis estático, linting y formateo automático para mantener la coherencia del código.	Ejecución de ESLint, Prettier o los scripts equivalentes configurados.
RNF-14	El proyecto debe poder validarse mediante un flujo de integración continua que ejecute comprobaciones automáticas de construcción, formato y pruebas.	Revisión y ejecución del flujo de integración continua configurado.

Tabla 4.10: Requisitos no funcionales aplicables al sistema, segunda parte

4.3 Diseño de la interfaz de usuario

El diseño de la interfaz de usuario fue un aspecto relevante del proyecto, ya que la aportación principal de la aplicación no consiste únicamente en calcular probabilidades, sino en permitir que el usuario construya y comprenda modelos de riesgo mediante una representación visual clara. En una herramienta de este tipo, el diagrama no actúa como una simple visualización auxiliar, sino como el elemento principal de interacción. Esta decisión está alineada con la idea central del proyecto: el modelo debe ser visible, manipulable y comprensible directamente por el usuario.

4.3.1. Criterios generales de diseño

Desde el inicio del proyecto se establecieron varios criterios básicos para la interfaz, los cuáles se muestran en la Tabla 4.11.

Criterio	Aplicación en la interfaz
Centralidad del diagrama	El lienzo ocupa la zona principal de la aplicación
Interacción directa	El usuario crea, conecta y modifica nodos visualmente
Separación de modos	La aplicación distingue entre modo Binomial y modo Bayes
Feedback inmediato	Los resultados se actualizan y muestran sobre el propio diagrama
Simplicidad visual	La interfaz evita sobrecargar al usuario con paneles innecesarios
Trazabilidad	Cada resultado se muestra asociado al nodo o conexión que lo produce

Tabla 4.11: Criterios de diseño y su aplicación en la interfaz de usuario

La interfaz se diseñó alrededor de un lienzo principal, gestionado mediante Daga, sobre el cual, el usuario puede construir el modelo. Los controles auxiliares permiten cambiar el modo de trabajo, importar o exportar modelos y modificar parámetros. Esta organización, busca que la atención del usuario permanezca centrada en la estructura del riesgo.

4.3.2. Evolución del diseño durante los Sprints

Durante el Sprint 1 se realizaron los primeros bocetos de interfaz en papel. Estos bocetos no tenían como objetivo definir el diseño visual definitivo, sino aclarar el funcionamiento básico de la aplicación: dónde se ubicaría el lienzo, cómo se seleccionarían los modos y cómo se mostrarían los valores probabilísticos.

En esta primera versión, el diseño era funcional y estaba orientado principalmente a comprobar que la integración con Daga funcionaba correctamente. La prioridad era validar la creación de nodos, conexiones y propiedades, no conseguir una interfaz visualmente final.

Durante el Sprint 2, la interfaz evolucionó para incorporar las nuevas necesidades del modo Bayes. Fue necesario mostrar tablas de probabilidad condicional, evidencias y probabilidades marginales. Esto aumentó la complejidad de la pantalla y obligó a revisar la forma en la que se representaban los datos sobre los nodos.

Finalmente, en el Sprint 3 se realizó una reestructuración completa de la interfaz con el apoyo de Paco Soria, diseñador de Metadev. Esta revisión permitió mejorar la organización visual de la aplicación, simplificar la presentación de controles y dar una apariencia más clara y coherente al producto final.

4.3.3. Estructura de la interfaz

La interfaz final se organiza en torno a tres zonas principales las cuales se especifican en la Tabla 4.12.

Zona	Descripción
Lienzo de diagramación	Área principal donde se crean, conectan y editan los nodos del modelo
Controles de modo y acciones	Zona desde la que se selecciona el modo de trabajo y se accede a acciones como importar o exportar
Información visual sobre el modelo	Elementos integrados en nodos y conexiones que muestran probabilidades, evidencias y resultados

Tabla 4.12: Zonas principales de la interfaz de usuario

El lienzo es el núcleo de la aplicación. En él, cada nodo representa una variable o evento del modelo, y cada conexión representa una relación entre elementos. Esta decisión permite que la estructura del riesgo sea visible en todo momento.

En el modo Binomial, la interfaz muestra conexiones ponderadas y probabilidades asociadas a los caminos del modelo. El objetivo es facilitar la construcción de escenarios ramificados y permitir que el usuario observe cómo se propagan las probabilidades.

En el modo Bayes, los nodos incorporan información adicional, como tablas de probabilidad condicional, evidencias y marginales. El usuario puede fijar o quitar evidencias y observar el efecto que esto produce sobre el resto de la red.

4.3.4. Representación visual de resultados

Una de las decisiones más importantes de diseño fue mostrar los resultados probabilísticos directamente sobre el diagrama. En lugar de presentar únicamente una tabla externa de resultados, la aplicación asocia cada valor calculado al nodo o conexión correspondiente.

Esta decisión aporta varias ventajas, las cuales se muestran en la Tabla 4.13.

Ventaja	Explicación
Comprensión inmediata	El usuario ve el resultado en el mismo lugar donde se encuentra el elemento del modelo
Menor carga cognitiva	No es necesario relacionar manualmente tablas externas con nodos del diagrama
Mayor trazabilidad	Cada probabilidad queda vinculada visualmente a la parte del modelo que la genera
Mejor detección de errores	Los valores incoherentes o inesperados son más fáciles de localizar

Tabla 4.13: Ventajas de la visualización integrada de resultados probabilísticos

En el modo Bayes, esta representación es especialmente útil, ya que las evidencias y las probabilidades marginales pueden cambiar dinámicamente. Por ello, la interfaz muestra de forma visual el estado de cada nodo y permite comprender mejor la propagación de la incertidumbre dentro de la red.

4.3.5. Auto-layout para modelos importados

Durante el desarrollo se detectó una necesidad específica relacionada con la importación de datos desde CSV. Aunque los nodos y relaciones podían generarse a partir de los datos, inicialmente aparecían distribuidos de forma poco clara o separados visualmente. Esto dificultaba la comprensión del modelo importado.

Para resolver este problema se diseñó un sistema de auto-layout para el modo Bayes. Su objetivo es organizar automáticamente los nodos en el lienzo cuando se genera un modelo a partir de datos importados. De esta forma, el usuario obtiene una representación inicial más ordenada y puede comenzar a analizar el modelo sin tener que reorganizar manualmente todos los elementos.

4.3.6. Imágenes de la evolución del proyecto

Para complementar la descripción del diseño de la interfaz, se incluyen a continuación varias capturas representativas de la evolución y del resultado final de la aplicación. Estas figuras no pretenden documentar exhaustivamente todas las pantallas del sistema, sino mostrar los elementos más relevantes del diseño: la organización general de la interfaz, la centralidad del lienzo de diagramación, la diferenciación entre modos de trabajo y la visualización directa de resultados probabilísticos sobre el modelo.

En primer lugar, en la Figura 4.3 se muestran algunos de los bocetos iniciales elaborados durante las primeras fases del proyecto. Estos bocetos sirvieron para definir la disposición general de la aplicación y validar la idea de que el usuario debía trabajar principalmente sobre un lienzo visual.

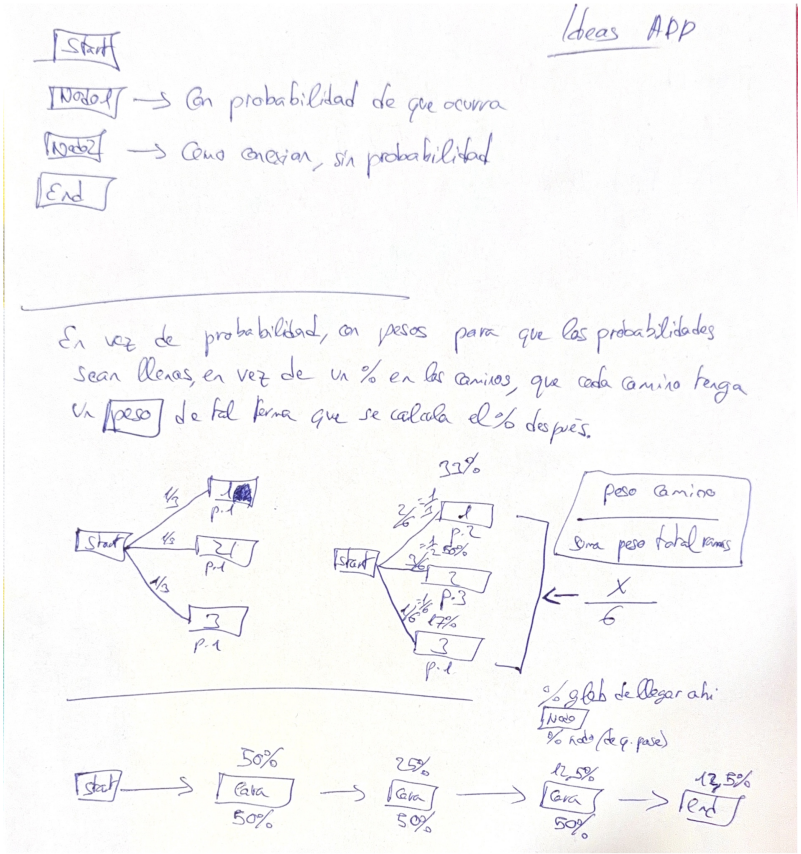

Figura 4.3: Primeros bocetos de la interfaz hechos a mano

A continuación, se incluye una captura del modo Binomial en la Figura 4.4, donde se aprecia cómo el usuario puede construir un modelo basado en nodos y conexiones ponderadas. Esta vista permite comprobar cómo los valores probabilísticos se integran directamente en el diagrama, evitando que el usuario tenga que consultar resultados en tablas externas.

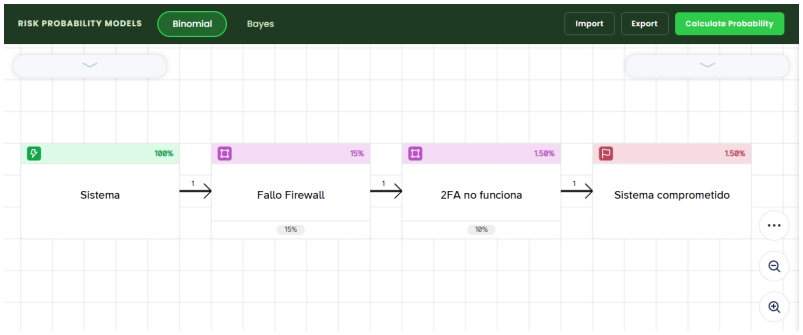

Figura 4.4: Interfaz del modelo binomial final con un ejemplo

En la Figura 4.5, se muestra una captura del modo Bayes. En este caso, la interfaz incorpora información adicional asociada a redes bayesianas, como tablas de probabilidad condicional, evidencias y probabilidades marginales. Esta representación permite visualizar cómo la introducción de evidencia en un nodo afecta al resto del modelo.

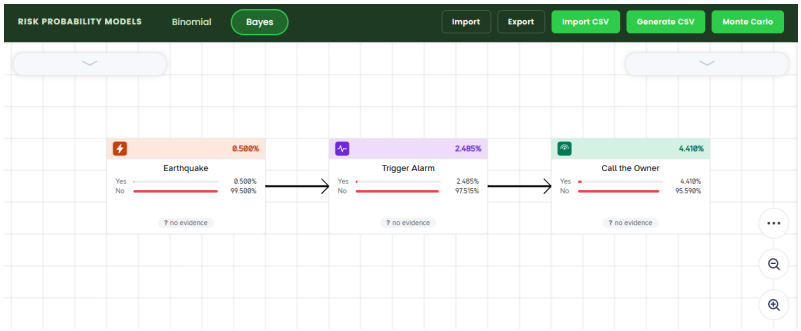

Figura 4.5: Interfaz del modelo bayes final con un ejemplo

Finalmente, en la Figura 4.6 se incluye una captura de un modelo generado a partir de datos importados. Esta figura permite mostrar la utilidad del sistema de auto-layout, incorporado para organizar automáticamente los nodos y mejorar la comprensión visual del grafo tras la importación.

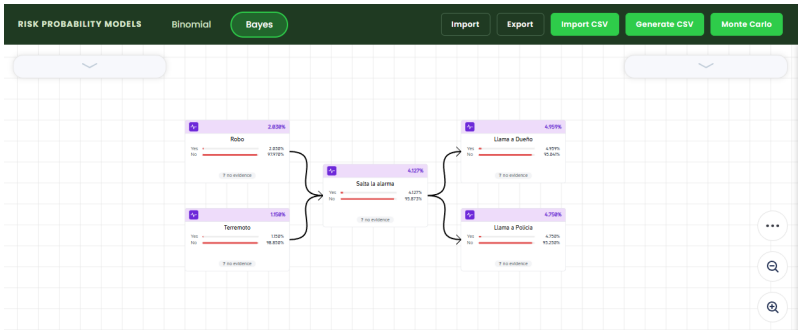

Figura 4.6: Interfaz del modelo bayes al hacer auto-layout tras importar un CSV

Estas capturas reflejan la evolución de la interfaz desde una primera aproximación funcional hasta una versión más estructurada y visualmente clara. Además, permiten evidenciar que el diseño de la IU no se abordó como un elemento secundario, sino como una parte esencial del proyecto, ya que la comprensión del modelo probabilístico depende en gran medida de su representación visual.

4.4 Diseño de la solución

El diseño de la solución se planteó con el objetivo de construir una aplicación web interactiva, modular y extensible, capaz de integrar un motor de diagramación visual con lógica probabilística. La arquitectura resultante de esta solución se centra principalmente en el cliente, ya que la mayor parte de la interacción con el usuario y de los cálculos se ejecutan directamente en el navegador.

Esta decisión responde a la naturaleza del proyecto: el usuario debe poder crear nodos, establecer conexiones, modificar probabilidades y observar los resultados en tiempo real sobre el propio diagrama. Por ello, se buscó reducir la dependencia de peticiones constantes a un servidor y mantener una experiencia fluida durante la edición del modelo.

4.4.1. Arquitectura global

La arquitectura global de la aplicación se organiza alrededor de una aplicación Angular que integra la librería Daga como motor de diagramación. Sobre este lienzo visual se apoyan los distintos módulos de cálculo, importación, exportación y actualización de resultados.

A nivel general, la solución se compone de los elementos visibles en la Tabla 4.14

Elemento	**Responsabilidad**
Navegador del usuario	Ejecuta la interfaz, el lienzo visual y la lógica principal de interacción
Aplicación Angular	Estructura la interfaz, los componentes y los modos de trabajo
Daga	Proporciona el motor de diagramación para nodos y conexiones
Motor binomial	Calcula probabilidades en árboles o grafos ponderados
Motor bayesiano	Gestiona CPTs, evidencias e inferencia probabilística
Módulo CSV	Permite importar datos y generar modelos a partir de archivos externos
Módulo RiskFile	Permite guardar y recuperar modelos en formato JSON
AWS S3	Permite publicar la aplicación en un entorno accesible desde navegador.

Tabla 4.14: Elementos principales del sistema y sus responsabilidades

La persistencia de modelos no se implementó mediante una base de datos tradicional, sino mediante archivos exportables. Esta decisión simplifica la arquitectura y resulta coherente con el alcance del TFG, ya que el objetivo principal era validar el modelado visual y el cálculo probabilístico, no construir una plataforma multiusuario completa.

En la Figura 4.7 se puede visualizar el diagrama que puede utilizarse como referencia para representar la arquitectura global.

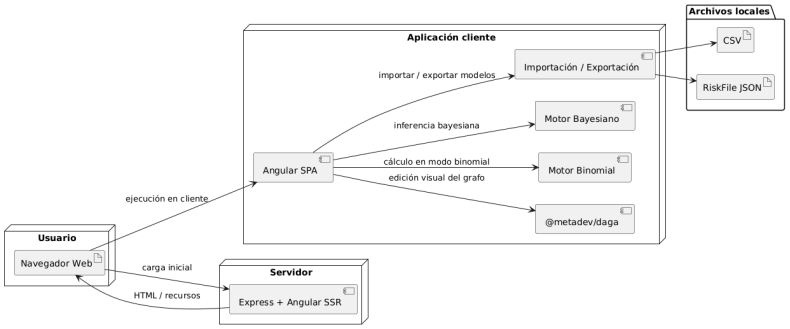

Figura 4.7: Diagrama de la arquitectura global de la aplicación

En esta arquitectura, el DSL no se materializa como un lenguaje textual independiente con un parser propio, sino como un modelo de dominio interno construido sobre las estructuras de datos de la aplicación. Dicho modelo permite representar conceptos propios del dominio del riesgo, como nodos probabilísticos, relaciones dirigidas, evidencias, tablas de probabilidad condicional, pesos, modos de cálculo y reglas de persistencia.

De este modo, el DSL actúa como una capa conceptual entre la interfaz visual y el motor de cálculo. El usuario no necesita escribir instrucciones formales, sino que construye el modelo mediante acciones gráficas sobre el lienzo. Internamente, estas acciones se traducen a una representación estructurada que puede ser interpretada por los módulos de cálculo binomial, inferencia bayesiana, importación/exportación y persistencia. Esta decisión reduce la barrera de entrada para usuarios no técnicos, manteniendo al mismo tiempo una separación clara entre la representación visual y la lógica probabilística del sistema.

4.4.2. Diseño del frontend

El frontend constituye el núcleo principal de la solución. Se desarrolló con Angular y TypeScript, empleando una organización modular basada en componentes y utilidades independientes.

El componente principal de la aplicación actúa como contenedor general y permite seleccionar el modo de trabajo. A partir de esta estructura se reutiliza una base común para la integración con Daga, evitando duplicar lógica entre el modo Binomial y el modo Bayes.

Los elementos principales del diseño frontend se describen en la Tabla 4.15

Elemento	Responsabilidad
SimpleComponent	Componente principal de la aplicación y punto de entrada de la interfaz
DagaBaseComponent	Componente base que centraliza la interacción con el lienzo Daga
Modo Binomial	Gestión de árboles de probabilidad y conexiones ponderadas
Modo Bayes	Gestión de redes bayesianas, CPTs, evidencias y marginales
probability.utils.ts	Normalización y conversión de probabilidades
connectionCalculate.utils.ts	Cálculo y reequilibrado de conexiones
bayesInference.utils.ts	Inferencia bayesiana exacta
montecarlo.utils.ts	Simulación aproximada mediante Monte Carlo
csv.utils.ts	Importación y tratamiento de datos CSV
importExport.utils.ts	Exportación e importación de modelos RiskFile

Tabla 4.15: Componentes y utilidades principales del frontend

Esta separación permite distinguir entre la lógica visual, la probabilística y las operaciones de entrada/salida. El uso de utilidades independientes facilita además la creación de pruebas unitarias, ya que gran parte del comportamiento puede validarse sin depender directamente de la interfaz gráfica.

En la Figura 4.8 se encuentra el diagrama que puede representar la estructura lógica del frontend.

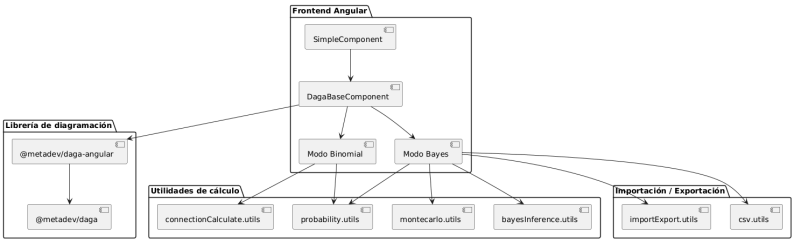

Figura 4.8: Diagrama de los componentes del frontend

4.4.3. Diseño del modo Binomial

El modo Binomial fue el primer modo funcional implementado. Su objetivo es permitir al usuario construir modelos de ramificación donde las conexiones tienen pesos o probabilidades asociadas.

En este modo, cada nodo representa un evento o estado, mientras que cada conexión representa una transición ponderada. Cuando el usuario modifica el valor de una conexión, el sistema recalcula automáticamente las probabilidades afectadas y mantiene la coherencia con el resto de conexiones relacionadas.

Además del cálculo determinista de probabilidades, el modo Binomial incorpora la posibilidad de realizar una simulación mediante Monte Carlo. Esta funcionalidad permite ejecutar múltiples iteraciones sobre el modelo definido por el usuario y comparar los resultados simulados con las probabilidades teóricas calculadas por el sistema. De este modo, el usuario puede validar de forma experimental el comportamiento del árbol de probabilidad y observar posibles desviaciones entre el cálculo esperado y la simulación.

La incorporación de Monte Carlo aporta valor al modo Binomial porque permite comprobar la estabilidad del modelo y ofrecer una aproximación más cercana a escenarios reales, donde los resultados se obtienen a partir de repeticiones sucesivas de eventos probabilísticos. Esta funcionalidad también resulta útil desde un punto de vista didáctico, ya que ayuda a comprender cómo las probabilidades teóricas se reflejan en resultados observados tras un número elevado de simulaciones.

Este modo sirvió como primera aproximación al problema porque permitió validar la integración entre Daga y Angular antes de abordar el modelo bayesiano, más complejo. Además, parte de la lógica desarrollada para normalizar probabilidades, gestionar conexiones y ejecutar simulaciones fue posteriormente reutilizada o adaptada en el modo Bayes.

4.4.4. Diseño del modo Bayes

El modo Bayes permite trabajar con redes bayesianas binarias. En este modo, los nodos representan variables aleatorias con dos posibles estados, y las conexiones representan dependencias entre variables.

Cada nodo puede disponer de una tabla de probabilidad condicional, o CPT, que define su comportamiento en función de sus nodos padre. Además, el usuario puede fijar evidencias en determinados nodos para observar cómo cambian las probabilidades marginales del resto de la red.

El diseño del módulo bayesiano incluye los elementos descritos en la Tabla 4.16.

Elemento	Función
Nodo bayesiano	Representa una variable binaria del modelo
Conexión dirigida	Representa una relación de dependencia
CPT	Define las probabilidades condicionales del nodo
Evidencia	Fija el valor observado de una variable
Inferencia exacta	Calcula marginales mediante enumeración
Monte Carlo	Aproxima resultados cuando la red crece
Auto-layout	Organiza visualmente redes importadas

Tabla 4.16: Elementos principales del modelo bayesiano y sus funciones

En la Figura 4.9 se muestra el diagrama de secuencia que puede utilizarse para representar el flujo de actualización al fijar una evidencia.

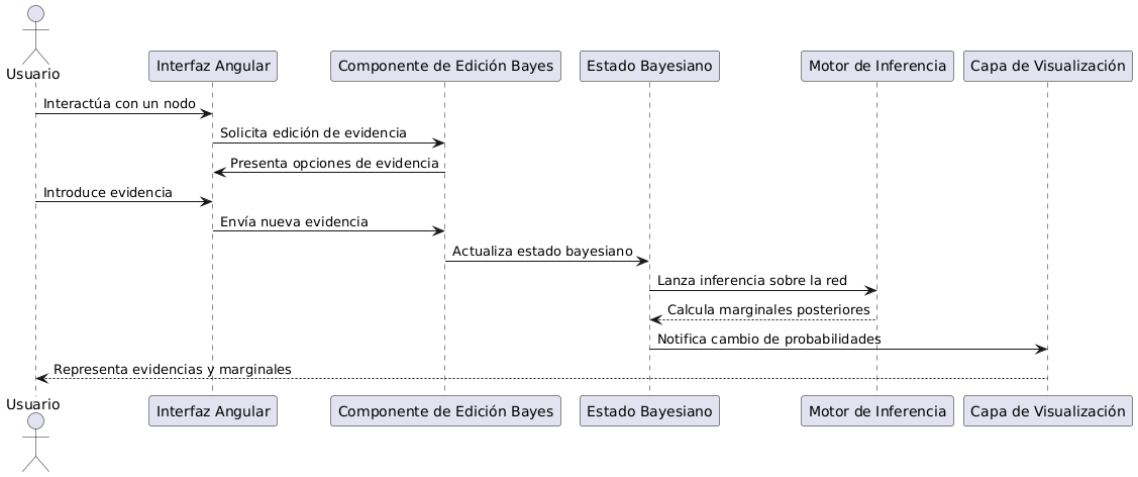

Figura 4.9: Diagrama de secuencia para la actualización de evidencias en modo Bayes

4.4.5. Diseño de la persistencia

La persistencia de modelos se diseñó mediante un formato propio denominado RiskFile, basado en JSON. Este archivo permite guardar en un único documento tanto la estructura visual del diagrama como el estado probabilístico asociado.

En el caso del modo Bayes, el archivo también conserva las CPTs y evidencias configuradas. Esta decisión permite que un modelo pueda exportarse, compartirse, versionarse y volver a importarse posteriormente sin perder información.

El uso de JSON se justifica por tres razones principales, las cuales se muestran en la Tabla 4.17.

Razón	**Justificación**
Legibilidad	El archivo puede inspeccionarse sin herramientas propietarias
Portabilidad	Puede compartirse o almacenarse fácilmente
Versionado	Es compatible con herramientas de control de versiones como Git

Tabla 4.17: Justificación de los beneficios del formato de archivo seleccionado

Además del formato RiskFile, la aplicación permite importar datos desde CSV. Esta funcionalidad resultó especialmente relevante para el modo Bayes, ya que permite generar nodos, relaciones y tablas de probabilidad condicional a partir de información externa.

4.4.6. Alcance arquitectónico de la versión final

Durante el desarrollo del TFG se decidió acotar el alcance de la solución y centrar la arquitectura en una aplicación frontend. La lógica principal de interacción, cálculo probabilístico, importación y exportación se ejecuta en el navegador, de forma que la herramienta puede funcionar como una aplicación web interactiva centrada en el modelado visual y el cálculo probabilístico.

Esta decisión permite concentrar el esfuerzo en el objetivo principal del proyecto: construir una plataforma visual para representar modelos de riesgo y calcular probabilidades de forma dinámica. Además, la generación de una versión de producción permite publicar la aplicación como un conjunto de archivos estáticos accesibles desde navegador.

4.5 Programación

El desarrollo de la aplicación se realizó de forma incremental durante los Sprints descritos anteriormente. La programación no consistió únicamente en implementar funcionalidades nuevas, sino también en estudiar tecnologías desconocidas, integrar una librería de diagramación específica, corregir errores, refactorizar código y mejorar progresivamente la experiencia de usuario.

4.5.1. Integración inicial con Daga

Una de las primeras dificultades del proyecto fue la integración de la librería Daga dentro de una aplicación Angular. Durante el Sprint 1 se realizó un estudio específico de la herramienta, ya que era la primera vez que se trabajaba con ella.

Para facilitar este proceso se contó con una pequeña sesión introductoria impartida por un trabajador de Metadev, que permitió comprender mejor la filosofía de la librería y su funcionamiento general. Además, se completó el tutorial disponible en la página oficial de Daga, lo que ayudó a entender cómo se creaban los nodos, conexiones, propiedades y eventos dentro del lienzo.

Este aprendizaje fue fundamental para poder transformar Daga de una librería genérica de diagramación en la base visual de un modelo probabilístico. Cada elemento del diagrama debía tener significado dentro del dominio: un nodo representaba una variable o evento, una conexión representaba una dependencia o transición, y las propiedades visuales debían estar sincronizadas con los cálculos internos.

4.5.2. Implementación del modo Binomial

El modo Binomial fue la primera funcionalidad completa implementada. Su desarrollo permitió validar el flujo básico de la aplicación: crear nodos, conectarlos, asignar pesos y recalcular probabilidades.

Durante esta fase se trabajó especialmente en la gestión de conexiones ponderadas. Uno de los retos fue mantener la coherencia entre conexiones hermanas cuando el usuario modificaba un valor. Para ello se implementó una lógica de reequilibrado automático, evitando que el usuario tuviera que ajustar manualmente todos los valores relacionados.

Este modo también sirvió como base técnica para funcionalidades posteriores. Algunas utilidades creadas inicialmente para el modo Binomial, especialmente las relacionadas con normalización de probabilidades, fueron reutilizadas después en el modo Bayes.

En la Figura 4.10 se muestra una parte de la implementación encargada de calcular las probabilidades teóricas de los nodos en el modo Binomial. Este cálculo parte de los nodos iniciales del modelo y realiza un recorrido en profundidad por las conexiones salientes del grafo.

Durante el recorrido, la probabilidad acumulada se multiplica por la probabilidad efectiva del nodo actual. A continuación, dicha probabilidad se reparte entre sus conexiones salientes en función del peso asignado a cada rama. Para ello, los pesos de las conexiones se normalizan respecto al peso total de las ramas hermanas, de forma que cada camino recibe una proporción de la probabilidad acumulada.

Este fragmento también contempla dos aspectos importantes para la robustez del cálculo. Por un lado, las contribuciones de probabilidad se acumulan, ya que un mismo nodo puede ser alcanzado desde varios caminos distintos. Por otro lado, se utiliza un conjunto de nodos visitados para evitar recorridos infinitos en caso de que existan ciclos no deseados en el diagrama.

```
function walkTheoreticalProbabilities(currentNode, accumulatedProbability, outgoingConnections,
                                      nodeMap, results, visited, startNodeId, ...): void {
  const factor = currentId === startNodeId ? maxProbability
               : getEffectiveNodeProbability(currentNode, probabilityKey, maxProbability);
  const nodeProbability = accumulatedProbability * (factor / maxProbability);
  addProbabilityContribution(results, currentId, nodeProbability);   // acumula (un nodo puede recibir de varios caminos)

  const outgoing = outgoingConnections.get(currentId) ?? [];
  const branchWeights = outgoing.map((c) => resolveConnectionWeight(c, branchValueKey));
  const totalWeight = branchWeights.reduce((a, w) => a + w, 0);
  if (totalWeight <= 0) return;

  outgoing.forEach((connection, i) => {
    const next = getNextNodeFromConnection(connection, nodeMap);
    if (!next || visited.has(nextId)) return;   // evita ciclos
    visited.add(nextId);
    walkTheoreticalProbabilities(next, nodeProbability * (branchWeights[i] / totalWeight), ...);
    visited.delete(nextId);
  });
}

export function calculateTheoreticalNodeProbabilities(model, probabilityKey, branchValueKey, maxProbability): Map<NodeId,
number> {
  const startNodes = findStartNodes(nodes, connections);
  for (const startNode of startNodes)
    walkTheoreticalProbabilities(startNode, maxProbability, ..., new Set([startNodeId]), startNodeId, ...);
  return results;
}
```

Figura 4.10: Fragmento de código para el cálculo de probabilidades teóricas en el modo Binomial

4.5.3. Desarrollo del modo Bayes

El Sprint 2 se centró en la implementación del modo Bayes, que fue una de las partes más complejas del proyecto. La primera versión funcional permitió crear redes de nodos, definir relaciones causales, asociar tablas de probabilidad condicional y fijar evidencias.

Sin embargo, durante las primeras pruebas se detectó un problema importante: la lógica inicial de inferencia contemplaba correctamente la propagación de información de causa a efecto, pero no gestionaba correctamente los casos en los que la evidencia se introducía en nodos hijos. Esto provocaba que los nodos padre no actualizasen sus probabilidades de forma coherente.

Para resolverlo fue necesario refactorizar la lógica de inferencia, de forma que la red pudiera recalcular probabilidades teniendo en cuenta la evidencia introducida en cualquier punto del grafo. Esta corrección supuso un avance importante, ya que permitió que el modo Bayes se comportara de forma más coherente con el funcionamiento esperado de una red bayesiana.

En la Figura 4.11 se muestra una parte del motor de inferencia exacta utilizado en el modo Bayesiano. Este fragmento calcula las probabilidades marginales de los nodos de la red a partir de las tablas de probabilidad condicional y de las evidencias fijadas por el usuario.

El cálculo se basa en la enumeración de los posibles estados de la red. Para ello, la función `enumerateWorlds` recorre los nodos en orden topológico y construye de forma recursiva las combinaciones posibles de valores *sí* y *no*. Durante este proceso se descartan las combinaciones incompatibles con la evidencia introducida, y a cada mundo posible se le asigna un peso obtenido a partir de las probabilidades definidas en las CPTs.

Una vez generados los mundos compatibles con la red, el sistema calcula la probabilidad marginal de cada nodo sumando los pesos de aquellos mundos en los que dicho nodo toma el valor *sí* y dividiendo el resultado entre el peso total. En el caso de los nodos observados, la marginal se fija directamente de acuerdo con la evidencia indicada por el usuario.

La implementación también incluye controles para evitar situaciones no válidas o costosas. Por un lado, si se detecta un ciclo en el grafo, el cálculo exacto se interrumpe, ya que la red dejaría de ser un DAG válido. Por otro lado, se establece un límite máximo de nodos para la inferencia exacta mediante la constante `MAX EXACT INFERENCE NODES`. Cuando la red supera dicho tamaño, el sistema puede recurrir a un enfoque aproximado, como la simulación Monte Carlo, para evitar un coste computacional excesivo.

```ts
export const MAX_EXACT_INFERENCE_NODES = 20;

function enumerateWorlds(graph, order, evidence, idx, currentWorld): WorldState[] {
  if (idx === order.length) return [{ states: { ...currentWorld }, weight: 1 }];

  const node = graph.get(order[idx])!;
  const worlds: WorldState[] = [];
  for (const state of ['si', 'no']) {
    if (evidence[order[idx]] && evidence[order[idx]] !== state) continue;
    const prob = node.parents.length === 0
      ? node.cpt['prior']?.[state]
      : node.cpt[node.parents.map((p) => `${p}_${currentWorld[p]}`).join('|')]?.[state];
    if (!prob) continue;
    const sub = enumerateWorlds(graph, order, evidence, idx + 1, { ...currentWorld, [order[idx]]: state });
    for (const sw of sub) sw.weight *= prob;
    worlds.push(...sub);
  }
  return worlds;
}

export function recalcAllMarginals(graph: BayesGraph): BayesGraph {
  if (hasCycle(graph)) { ... /* abortar: marginales 0.5 */ }
  if (graph.size > MAX_EXACT_INFERENCE_NODES) { ... /* saltar: usar Monte Carlo */ }

  const order = topologicalSort(graph);
  const evidence = ...; // recoge node.evidence de cada nodo
  const worlds = enumerateWorlds(graph, order, evidence, 0, {});
  const totalWeight = worlds.reduce((s, w) => s + w.weight, 0);

  for (const [nodeId, node] of graph) {
    if (node.evidence) { node.marginals = ...; continue; }   // nodo observado
    const weightSi = worlds.filter((w) => w.states[nodeId] === 'si')
                           .reduce((s, w) => s + w.weight, 0);
    node.marginals = { si: weightSi / totalWeight, no: 1 - weightSi / totalWeight };
  }
  return graph;
}
```

Figura 4.11: Fragmento de código para la inferencia exacta en el modo Bayesiano

4.5.4. Normalización y precisión de probabilidades

Otro problema relevante detectado durante la implementación estuvo relacionado con la representación numérica de probabilidades. En las primeras versiones existía cierta mezcla entre valores expresados como porcentajes y valores normalizados entre 0 y 1.

Esta situación podía provocar errores en los cálculos y dificultades para introducir valores pequeños. Para solucionarlo, se centralizó la normalización de probabilidades en una utilidad específica, evitando que cada parte del sistema gestionase la conversión por separado.

En la Figura 4.12 se muestra una parte de las utilidades empleadas para normalizar probabilidades e identificadores internos del modelo. Estas funciones son transversales al sistema, ya que se utilizan antes de almacenar o procesar valores que afectan al cálculo probabilístico y a la correspondencia entre el modelo visual y el lógico.

La función de normalización de probabilidades convierte la entrada recibida a un valor numérico válido, permite trabajar con distintos formatos decimales y descarta valores no permitidos. Además, limita el resultado al rango definido por la aplicación y aplica un redondeo controlado. De esta forma, el sistema mantiene una escala coherente de probabilidad entre los distintos módulos.

Por otra parte, la normalización de identificadores permite eliminar sufijos añadidos por la librería de diagramación en los puertos de conexión. Esto es necesario porque, durante el cálculo, cada nodo debe tener una única identidad lógica. Si no se realizara esta normalización, un mismo nodo podría aparecer con identificadores distintos en los mapas internos, provocando errores al recorrer el grafo o al actualizar sus probabilidades.

```ts
export const MAX_PROBABILITY = 100;

export function normalizeProbability(rawValue, maxProbability = MAX_PROBABILITY, decimals = 6): number | null {
  const numericValue = parseProbabilityNumber(rawValue); // admite coma/punto y locales
  if (numericValue == null || numericValue < 0) return null;
  const clamped = Math.max(0, Math.min(maxProbability, numericValue)); // recorte al rango válido
  return decimals == null ? clamped : Number(clamped.toFixed(decimals));
}

export function normalizeNodeId(rawId: string): string {
  return rawId.replace(/_port_\d+$/i, ''); // el motor daga sufija ids con _port_N
}
```

Figura 4.12: Fragmento de código para la normalización de probabilidades e identificadores

Además, se revisó la precisión decimal utilizada en los cálculos. Inicialmente algunos valores se limitaban a dos decimales, lo que generaba errores acumulativos en determinadas inferencias. Posteriormente se amplió la precisión y se ajustaron los controles de entrada para permitir incrementos más pequeños. Esto mejoró tanto la exactitud del cálculo como la experiencia de uso.

También se implementó un mecanismo de auto-normalización en determinados valores complementarios. Por ejemplo, si el usuario introducía una probabilidad, el sistema podía completar automáticamente su complementaria. Esta funcionalidad redujo la carga manual y disminuyó la posibilidad de errores de entrada.

En la Figura 4.13 se muestra el mecanismo utilizado para rebalancear las probabilidades de las conexiones salientes de un nodo. Esta parte de la implementación es relevante en el modo Binomial, ya que garantiza que la suma de las probabilidades de las ramas que parten de un mismo nodo se mantenga dentro de la escala definida por el sistema.

El método contempla dos situaciones principales. Cuando el usuario modifica manualmente el valor de una conexión, dicha probabilidad se conserva y el valor restante se distribuye entre las demás ramas salientes. En cambio, cuando el cambio procede de una modificación estructural, como añadir o eliminar una conexión, el sistema reparte la probabilidad de forma equilibrada entre todas las ramas disponibles.

Además, el cálculo se realiza utilizando unidades enteras en lugar de operar directamente con valores decimales. Esta decisión reduce los problemas derivados de la aritmética en coma flotante y permite que la suma final de las ramas sea consistente. De este modo, se evita que pequeños errores de redondeo generen probabilidades acumuladas incorrectas o resultados visuales incoherentes para el usuario.

```
function rebalanceOutgoingConnections(canvas, sourceNodeId, probabilityKey, maxProbability,
                                      manualConnectionId?, manualValue?): void {
  const outgoing = canvas.model.connections.all().filter((c) => getSourceNodeId(c) === sourceNodeId);
  if (outgoing.length === 0) return;
  const totalUnits = toProbabilityUnits(maxProbability, maxProbability); // trabaja con enteros para evitar errores de coma flotante

  if (manualConnectionId && manualValue !== undefined) {
    // El usuario fijó una rama: esa conserva su valor y el resto se reparte el remanente.
    const others = outgoing.filter((c) => c.id !== manualConnectionId);
    const manualUnits = toProbabilityUnits(manualValue, maxProbability);
    manualConn.valueSet.overwriteValues({ [probabilityKey]: fromProbabilityUnits(manualUnits) });
    applyDistributedProbabilities(others, distributeProbabilityUnits(others.length, totalUnits - manualUnits), probabilityKey);
  } else {
    // Reparto equitativo entre todas las ramas.
    applyDistributedProbabilities(outgoing, distributeProbabilityUnits(outgoing.length, totalUnits), probabilityKey);
  }
}
```

Figura 4.13: Fragmento de código para el rebalanceo de probabilidades en ramas salientes

Por último, además de calcular la inferencia exacta en redes pequeñas, se integró inferencia aproximada mediante simulaciones de Monte Carlo. En este proyecto, el uso de Monte Carlo permite mejorar la escalabilidad del modo bayesiano y contemplar redes de mayor tamaño o complejidad estructural.

Lejos de sustituir a la inferencia exacta, esta técnica la complementa. El sistema mantiene el cálculo exacto como método principal para los modelos manejables, mientras que la simulación permite ampliar el abanico de casos de uso contemplados por la aplicación.

4.5.5. Importación de CSV y generación automática de grafos

La importación de archivos CSV se implementó como una funcionalidad orientada a facilitar la creación automática de modelos a partir de tablas de datos. El sistema interpreta las columnas del archivo como variables del modelo y las filas como observaciones. Cuando el archivo incluye información estructural adicional, como una directiva de relaciones entre variables, la aplicación puede generar también las conexiones dirigidas entre nodos.

Esta funcionalidad está vinculada al modo Bayesiano, ya que permite construir redes a partir de conjuntos de datos. En los casos en los que los datos están completos, el sistema puede estimar parámetros mediante conteo de frecuencias y estimación de máxima verosimilitud (MLE). Cuando existen valores faltantes, el uso de EM se contempla como un mecanismo de estimación aplicable siempre que la estructura del modelo y la información disponible lo permitan.

Durante el desarrollo, esta funcionalidad se planteó de forma incremental. En primer lugar, se abordó la creación automática de nodos y relaciones a partir del archivo importado. Posteriormente, se integró con el sistema de cálculo y con el mecanismo de auto-layout, de forma que los modelos generados no aparecieran desordenados en el lienzo, sino distribuidos de manera más comprensible para el usuario.

4.5.6. Auto-layout en modo Bayes

Para resolver el problema de disposición visual de redes importadas, durante el Sprint 2 y su consolidación posterior se incorporó un sistema de auto-layout orientado al modo Bayes.

El problema surgía especialmente al importar modelos desde CSV: los nodos y relaciones se creaban, pero aparecían separados o distribuidos de forma poco clara. Esto obligaba al usuario a recolocar manualmente los elementos, lo que perjudicaba la experiencia de uso.

La incorporación del auto-layout permitió organizar automáticamente los nodos según sus relaciones causales. De este modo, al importar una red, el usuario obtiene una representación inicial más ordenada y comprensible. Esta mejora fue especialmente importante porque reforzó la idea central del proyecto: no basta con calcular correctamente, sino que el modelo debe ser visualmente interpretable.

4.5.7. Uso de herramientas de inteligencia artificial como apoyo

Durante el desarrollo se utilizaron herramientas de inteligencia artificial como apoyo en tareas de generación inicial, revisión de código, planteamiento de soluciones y refactorización. Estas herramientas permitieron acelerar ciertas fases del desarrollo, especialmente cuando se exploraban alternativas o se buscaban ejemplos de implementación.

No obstante, su uso requirió siempre supervisión y adaptación manual. En varias ocasiones, las propuestas generadas no se ajustaban completamente al contexto del proyecto o contenían errores conceptuales. Por ello, la inteligencia artificial se utilizó como una herramienta de apoyo, no como sustituto del criterio técnico.

Esta experiencia permitió comprobar que estas herramientas pueden aumentar la productividad, pero también hacen necesario reforzar la validación mediante pruebas, revisión manual y herramientas de calidad de código.

4.5.8. Refactorización y mejora de calidad

A medida que aumentaba el tamaño del proyecto, fue necesario refactorizar varias partes del código. Una decisión relevante fue separar la lógica común de integración con Daga en un componente base, reutilizable tanto por el modo Binomial como por el modo Bayes.

También se extrajo lógica de cálculo a utilidades independientes, como normalización de probabilidades, inferencia bayesiana, Monte Carlo, importación CSV y exportación RiskFile. Esta separación mejoró la mantenibilidad y facilitó la creación de pruebas unitarias.

Durante el Sprint 3 se intensificó este trabajo de limpieza y organización. La incorporación de ESLint y Prettier permitió detectar problemas de estilo, inconsistencias y posibles malas prácticas. La resolución de estos avisos contribuyó a obtener un código más homogéneo y preparado para futuras ampliaciones.

4.5.9. Rediseño final de la interfaz

En el Sprint 3 se llevó a cabo un rediseño completo de la interfaz con la ayuda de Paco Soria, diseñador de Metadev. Esta fase permitió mejorar la apariencia general de la aplicación, reorganizar controles y hacer que la herramienta fuese más clara para usuarios externos.

El rediseño supuso una transición desde una interfaz inicialmente funcional, centrada en validar la tecnología, hacia una interfaz más orientada a producto. Este cambio fue importante porque el proyecto no solo debía demostrar que los cálculos funcionaban, sino también que podían presentarse de forma comprensible y usable.

4.6 Pruebas

La validación y verificación del sistema se abordó mediante distintos niveles de prueba: pruebas de aceptación, pruebas unitarias, pruebas de integración y pruebas end-to-end. Además, se aplicó una estrategia de regresión basada en repetir pruebas de estos tipos cuando se incorporaban nuevas funcionalidades, se realizaban refactorizaciones o se preparaba una integración en la rama principal del proyecto.

Esta combinación permitió comprobar tanto la lógica interna del sistema como los flujos principales desde el punto de vista del usuario. Las pruebas automatizadas se integraron en el flujo de integración continua del repositorio, de forma que, al crear una pull request hacia la rama principal, se ejecutaban comprobaciones de construcción, formato, pruebas unitarias y pruebas end-to-end.

La metodología utilizada en PIN propone definir pruebas de aceptación durante la especificación de las unidades de trabajo y aplicarlas antes de considerar terminada una funcionalidad. En este proyecto se adaptó esa idea mediante criterios de aceptación asociados a las tarjetas de Trello y validaciones funcionales al cierre de cada Sprint.

4.6.1. Pruebas de aceptación

Las pruebas de aceptación se utilizaron para validar que las funcionalidades principales cumplían el comportamiento esperado desde el punto de vista del usuario.

Algunas pruebas de aceptación representativas fueron las que se encuentran en la Tabla 4.18

Estas pruebas se aplicaron de forma manual durante el desarrollo y especialmente al cierre de los Sprints, comprobando que las nuevas funcionalidades no rompían los flujos ya implementados.

4.6.2. Pruebas unitarias

Las pruebas unitarias se implementaron con Vitest. Se centraron principalmente en funciones puras y módulos de lógica interna, ya que estas partes podían validarse de forma automática sin depender de la interacción visual con el navegador.

Los módulos más relevantes para las pruebas unitarias fueron los que se encuentran en la Tabla 4.19.

Estas pruebas fueron especialmente útiles tras la incorporación del linter y las refactorizaciones del Sprint 3, ya que permitieron comprobar que los cambios estructurales no alteraban el comportamiento esperado.

Se definieron un total de 44 pruebas unitarias repartidas en 4 archivos test, todas ellas ejecutadas automáticamente mediante Vitest.

Código	Prueba de aceptación	Resultado esperado
PA-01	Crear un nodo en el lienzo	El nodo aparece correctamente y puede ser editado
PA-02	Crear una conexión entre dos nodos	La conexión se dibuja y queda asociada a ambos nodos
PA-03	Modificar un peso en modo Binomial	Se recalculan las probabilidades afectadas
PA-04	Crear una red en modo Bayes	Los nodos y conexiones representan dependencias válidas
PA-05	Editar una CPT	El nodo conserva los valores introducidos y actualiza sus cálculos
PA-06	Fijar evidencia en un nodo	Se actualizan las probabilidades marginales del resto de la red
PA-07	Importar un CSV	Se generan nodos, conexiones y valores según los datos importados
PA-08	Aplicar auto-layout	Los nodos importados se distribuyen de forma más legible
PA-09	Exportar un RiskFile	Se genera un archivo JSON con el estado del modelo
PA-10	Importar un RiskFile	Se restaura correctamente el modelo guardado

Tabla 4.18: Pruebas de aceptación definidas para la validación del sistema

Módulo	Aspectos validados
probability.utils.ts	Conversión, normalización y validación de probabilidades
connectionCalculate.utils.ts	Reequilibrado y cálculo de conexiones
bayesInference.utils.ts	Cálculo de marginales en redes bayesianas
montecarlo.utils.ts	Comportamiento de simulaciones aproximadas
csv.utils.ts	Lectura, interpretación y normalización de datos CSV
importExport.utils.ts	Serialización y restauración de modelos RiskFile

Tabla 4.19: Módulos técnicos y aspectos críticos validados en las pruebas unitarias

Estas pruebas también formaron parte del flujo de integración continua del proyecto, descrito posteriormente en el apartado de despliegue, donde se ejecutaban automáticamente antes de integrar cambios en la rama principal.

4.6.3. Pruebas de integración

Las pruebas de integración se centraron en validar la colaboración entre la interfaz Angular, el lienzo Daga y los módulos de cálculo. En una aplicación de estas características, no basta con comprobar funciones aisladas, ya que gran parte del comportamiento depende de que los cambios visuales se traduzcan correctamente en cambios del modelo interno.

Algunas pruebas de integración relevantes fueron las que se encuentran en la Tabla 4.20

Prueba	Objetivo
Integración Daga + modo Binomial	Comprobar que un cambio en una conexión actualiza el cálculo
Integración Daga + modo Bayes	Comprobar que una evidencia modifica los marginales
Integración CSV + grafo	Verificar que un archivo importado genera nodos y relaciones
Integración auto-layout + CSV	Verificar que la red importada se organiza visualmente
Integración RiskFile + lienzo	Comprobar que un modelo exportado puede restaurarse

Tabla 4.20: Pruebas de integración y sus objetivos principales

Estas pruebas permitieron detectar problemas que no aparecían en las pruebas unitarias, especialmente relacionados con sincronización entre el estado interno y la representación visual.

4.6.4. Pruebas end-to-end

Las pruebas end-to-end se realizaron mediante Cypress. Su objetivo fue validar flujos completos desde la perspectiva del usuario, ejecutando la aplicación en un navegador y comprobando que las interacciones principales funcionaban correctamente.

Los flujos E2E más representativos fueron los descritos en la Tabla 4.21.

Código	Flujo probado
E2E-01	Abrir la aplicación y comprobar la carga inicial
E2E-02	Crear nodos y conexiones en el lienzo
E2E-03	Trabajar en modo Binomial y modificar valores
E2E-04	Trabajar en modo Bayes y fijar evidencias

Tabla 4.21: Flujos principales validados mediante pruebas End-to-End

Cypress resultó especialmente adecuado porque el valor principal de la aplicación está en la interacción visual. Estas pruebas ayudaron a comprobar que los flujos más importantes podían ejecutarse de forma completa y repetible.

Se definieron 11 pruebas end-to-end, repartidas en los 2 modos de la aplicación y ejecutadas de forma automática mediante Cypress.

Estas pruebas se incorporaron también al flujo de integración continua, de modo que los recorridos principales de la aplicación se comprobaban automáticamente antes de aceptar cambios en la rama principal.

4.6.5. Linter y control de calidad

Además de las pruebas funcionales, se incorporaron herramientas de análisis estático y formato automático. ESLint permitió detectar errores de estilo, posibles malas prácticas y problemas de consistencia en el código. Prettier se utilizó para normalizar el formato de los archivos.

Durante su incorporación se detectaron varios fallos y avisos que fueron corregidos antes de consolidar la versión final. Este proceso fue útil no solo para mejorar la apariencia del código, sino también para localizar comportamientos potencialmente problemáticos y reducir deuda técnica.

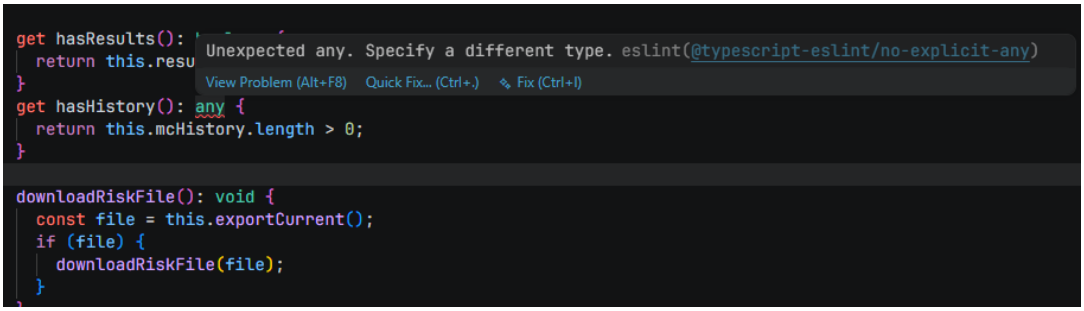

Figura 4.14: Ejemplo de cómo detecta malas prácticas ESLint

4.6.6. Estrategia de regresión

La estrategia de regresión combinó dos enfoques. Por un lado, las pruebas automatizadas se ejecutaron de forma recurrente mediante los scripts del proyecto y el flujo de integración continua. Por otro lado, al final del último Sprint se aplicó una revisión funcional de los principales casos de uso, siguiendo la adaptación de la metodología PIN, que propone aplicar pruebas de regresión sobre las pruebas de aceptación al cierre del proyecto.

Esta regresión se centró en verificar que las nuevas funcionalidades del Sprint 3, como importación/exportación, rediseño de interfaz, refactorizaciones y despliegue, no hubieran afectado negativamente a funcionalidades ya implementadas en Sprints anteriores.

Los casos revisados fueron principalmente los que se muestran en la Tabla 4.22.

Área	Funcionalidades revisadas
Modo Binomial	Creación de nodos, conexiones, edición de pesos y propagación
Modo Bayes	CPTs, evidencias, marginales e inferencia
CSV	Importación de datos y generación de redes
RiskFile	Exportación e importación de modelos
Interfaz	Visualización correcta tras el rediseño
Auto-layout	Distribución automática de nodos importados

Tabla 4.22: Áreas del sistema y funcionalidades revisadas

Durante estas pruebas se detectaron varios errores, especialmente en el modo Bayes y en la representación visual de modelos importados. Dichos fallos fueron corregidos antes de la versión final.

4.6.7. Resultados de las pruebas

El proceso de pruebas permitió detectar y corregir errores significativos a lo largo del desarrollo. Entre los más relevantes destacan los descritos en la Tabla 4.23.

Problema detectado	Solución aplicada
Inferencia bayesiana solo de causa a efecto	Refactorización de la lógica de inferencia
Mezcla entre porcentajes y valores normalizados	Centralización de la normalización de probabilidades
Pérdida de precisión por pocos decimales	Ampliación de precisión y ajuste de controles
Nodos importados desde CSV mal distribuidos	Incorporación de auto-layout
Errores de estilo y consistencia	Corrección mediante ESLint y Prettier
Bugs detectados en validación externa	Corrección durante la fase final del proyecto

Tabla 4.23: Problemas detectados y soluciones aplicadas durante el desarrollo

4.7 Despliegue

El despliegue de la aplicación se abordó durante el Sprint 3, una vez consolidadas las funcionalidades principales y tras la incorporación de pruebas, refactorización y mejoras de interfaz.

La aplicación se preparó para ser publicada como aplicación web accesible desde navegador. Para ello se utilizaron scripts específicos de despliegue, encargados de generar la versión de producción y sincronizar los archivos resultantes con AWS S3.

Los scripts principales utilizados fueron los mostrados en la Tabla 4.24.

Script	Función
scripts/publish-pre.sh	Publicación en entorno de preproducción
scripts/publish-pro.sh	Publicación en entorno de producción

Tabla 4.24: Scripts utilizados para la publicación en los distintos entornos

El proceso general de despliegue fue el siguiente:

1. Instalación de dependencias del proyecto.
2. Ejecución de comprobaciones de formato y pruebas.
3. Generación de la build de producción.
4. Sincronización de los artefactos generados con AWS S3.
5. Acceso a la aplicación desde navegador.

Durante el desarrollo se publicó una versión preliminar o pre-release en un entorno accesible públicamente. Esta publicación permitió que otras personas del entorno de Metadev pudieran probar la aplicación y detectar errores que no habían aparecido durante las pruebas internas. Esta validación externa fue útil para localizar problemas de consola, errores de interacción y posibles mejoras de usabilidad.

El despliegue en AWS S3 resulta adecuado para este tipo de aplicación, ya que permite distribuir el frontend de forma sencilla sin requerir instalación local por parte del usuario final. Además, encaja con la naturaleza web del proyecto y facilita futuras fases de validación con usuarios externos.

La aplicación también incorporó integración continua mediante GitHub Actions. El workflow definido ejecuta automáticamente tareas de instalación, comprobación de formato, construcción y pruebas cuando se realizan cambios en la rama principal o se prepara una pull request. En caso de fallo en pruebas end-to-end, se generan artefactos como capturas o vídeos para facilitar el diagnóstico.

En la Figura 4.15 se muestra un ejemplo de ejecución correcta del job de construcción. Este job instala las dependencias, comprueba el formato del código, genera la build del proyecto y ejecuta las pruebas unitarias automatizadas.

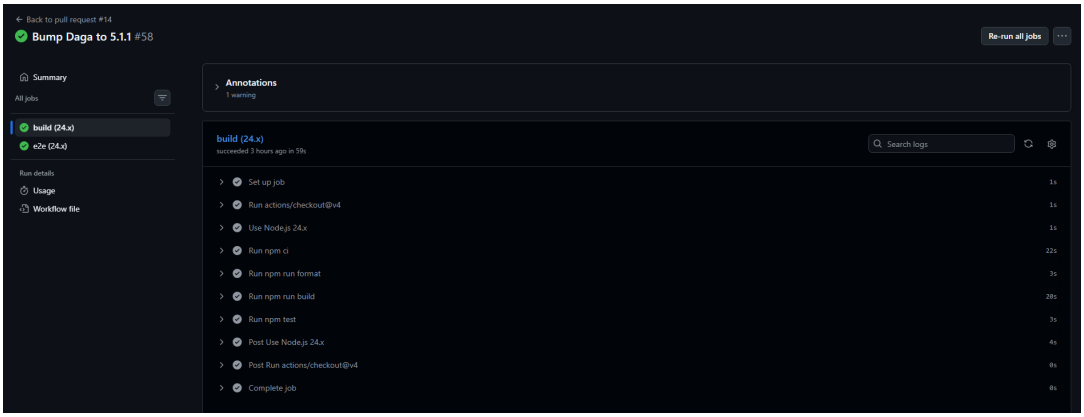

Figura 4.15: Ejecución automática del job de construcción y pruebas unitarias en GitHub Actions.

En la Figura 4.16 se muestra el job encargado de ejecutar las pruebas end-to-end mediante Cypress. Este proceso permite validar de forma automática los principales recorridos de la aplicación en un entorno de navegador antes de integrar los cambios.

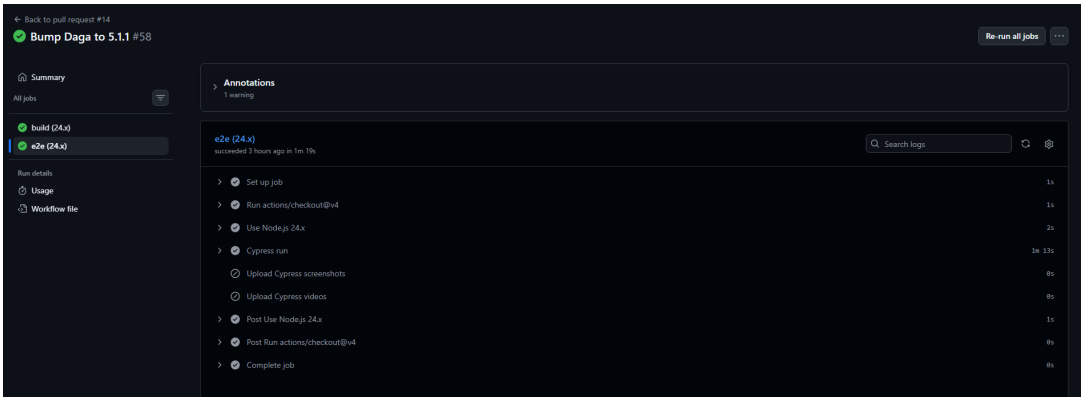

Figura 4.16: Ejecución automática del job de pruebas end-to-end con Cypress en GitHub Actions.

En conjunto, el despliegue y la integración continua permitieron pasar de un prototipo ejecutado localmente a una versión más cercana a un producto real, accesible desde navegador y validable por usuarios externos.

CAPÍTULO 5
Cronología del TFG

En este capítulo se presenta la cronología seguida durante el desarrollo del trabajo. Para ello, se relacionan los casos de uso principales con las unidades de trabajo creadas en el Backlog, se explica la estrategia de priorización seguida, se describe la evolución temporal del proyecto y se resume la dedicación aproximada empleada en cada fase.

El seguimiento del trabajo se realizó mediante un tablero Kanban en Trello, utilizado como adaptación práctica del workflow ágil descrito en el apartado metodológico. La metodología de referencia propone trabajar con unidades de trabajo, Backlog, Sprints y registro del esfuerzo invertido, contrastando las horas estimadas con las horas reales dedicadas.

5.1 Correspondencia entre casos de uso y unidades de trabajo

Los casos de uso definidos en el apartado de requisitos representan las principales funcionalidades de la aplicación desde el punto de vista del usuario. Sin embargo, durante el desarrollo el seguimiento no se realizó directamente sobre casos de uso, sino mediante unidades de trabajo registradas en Trello.

Estas unidades de trabajo incluyeron funcionalidades detalladas, mejoras de interfaz, correcciones de errores, refactorizaciones y tareas técnicas o documentales. Por ello, en la Tabla 5.1 relaciona las características principales del sistema con los casos de uso asociados y con las unidades de trabajo que permitieron desarrollarlas.

Característica	CU	UT
Base visual del modelo	CU-01, CU-02, CU-03	Creación del lienzo de trabajo, integración inicial con Daga, creación y edición de nodos, creación y eliminación de conexiones, sincronización entre el diagrama y el modelo interno, y corrección de errores de interacción visual.
Modo Binomial	CU-04, CU-08, CU-09	Configuración de pesos y probabilidades en conexiones, propagación de probabilidades, rebalanceo de ramas, normalización de valores, actualización visual de resultados y corrección de errores de precisión.
Modo Bayesiano	CU-05, CU-06, CU-07, CU-08, CU-09	Creación de redes bayesianas binarias, edición de CPTs, fijación y eliminación de evidencias, cálculo de marginales, inferencia exacta, simulación Monte Carlo y actualización de probabilidades en el diagrama.
Importación de datos CSV	CU-10, CU-08, CU-09	Lectura de archivos CSV, normalización de valores importados, generación automática de nodos y conexiones, aprendizaje de parámetros cuando procede, aplicación de auto-layout y corrección de errores en modelos importados.
Persistencia mediante RiskFile	CU-11	Exportación de modelos en formato JSON versionado, importación de modelos guardados, restauración del estado visual y probabilístico, conservación de pesos, CPTs, evidencias y estructura del grafo.

Tabla 5.1: Correspondencia entre características, casos de uso y unidades de trabajo

Además de las unidades de trabajo asociadas directamente a funcionalidades visibles por el usuario, el Backlog incluyó tareas transversales necesarias para el desarrollo del proyecto. Estas tareas no se corresponden con un único caso de uso, pero resultaron esenciales para garantizar la calidad, mantenibilidad y correcta finalización del TFG.

En la Tabla 5.2 se muestra una lista con las unidades de trabajo transversales, su tipo y finalidad.

Unidad de trabajo transversal	Tipo	Finalidad
Estudio de tecnologías	Formación	Aprendizaje de Angular, TypeScript, Daga y herramientas asociadas.
Estado del arte	Documentación	Análisis de herramientas probabilísticas, redes bayesianas, DSLs y librerías de diagramación.
Escritura de memoria en LaTeX	Documentación	Redacción progresiva del documento del TFG.
Automatización de pruebas unitarias	Calidad	Ejecución automática de pruebas unitarias mediante Vitest para validar funciones de cálculo y utilidades internas.
Automatización de pruebas end-to-end con Cypress	Calidad	Ejecución automática de flujos completos de usuario en navegador mediante Cypress.
Arreglo de fallos de linter	Calidad	Corrección de problemas de estilo, consistencia y mantenibilidad detectados por ESLint.
Refactorización	Mejora técnica	Reorganización del código y separación de responsabilidades.
Corrección de bugs	Corrección	Resolución de errores detectados durante pruebas, validaciones internas y revisión externa.
Ampliación de pruebas	Calidad	Extensión de los casos de prueba tras incorporar nuevas funcionalidades o corregir errores.
Despliegue de Risk	Despliegue	Publicación de la aplicación en un entorno accesible desde navegador.
Elaboración de ejemplos de uso	Documentación / validación	Creación de escenarios demostrativos para modo Binomial y modo Bayesiano.
Preparación de defensa	Presentación	Preparación de diapositivas, demo y ensayo de la exposición final.

Tabla 5.2: Unidades de trabajo transversales del Backlog

Los requisitos no funcionales se trataron principalmente de forma transversal. Por ejemplo, la mantenibilidad se abordó mediante refactorización y herramientas de calidad de código, la usabilidad mediante el rediseño visual de la interfaz, la fiabilidad mediante pruebas automatizadas y pruebas de aceptación, y la portabilidad mediante el despliegue web y el formato RiskFile basado en JSON.

5.2 Estrategia de priorización del Backlog

La priorización del Backlog se realizó teniendo en cuenta la dependencia técnica entre los casos de uso, el riesgo de implementación y el valor aportado al MVP. En primer lugar, se priorizaron los casos de uso necesarios para disponer de una base visual mínima: creación del modelo, creación y edición de nodos, y creación y eliminación de conexiones (CU-01 a CU-03).

Una vez validada esta base, se abordó el modo Binomial (CU-04), al ser el primer modelo probabilístico implementado y permitir comprobar la relación entre el diagrama visual y el motor de cálculo. Posteriormente se priorizó el modo Bayesiano (CU-05 a CU-07), por ser la parte técnicamente más compleja del proyecto, al incorporar redes bayesianas, CPTs, evidencias e inferencia probabilística.

Tras consolidar ambos modos, se trabajó en el recálculo automático y la actualización visual del diagrama (CU-08 y CU-09), aspectos transversales necesarios para que la aplicación ofreciera feedback inmediato al usuario. Finalmente, se implementaron las funcionalidades de interoperabilidad y persistencia: importación CSV (CU-10) y gestión de modelos RiskFile (CU-11).

La estrategia de priorización puede resumirse de la forma visible en la Tabla 5.3.

Prioridad	Criterio	Unidades de trabajo representativas
1	Reducir incertidumbre técnica	Estudio de tecnologías, pruebas con Daga, instalación del entorno y configuración inicial del proyecto.
2	Construir la base visual	Integración Angular-Daga, creación del lienzo, creación y edición de nodos, y creación de conexiones.
3	Validar el primer modelo probabilístico	Implementación del modo Binomial, configuración de pesos, propagación de probabilidades y actualización visual de resultados.
4	Resolver el núcleo algorítmico complejo	Implementación del modo Bayesiano, edición de CPTs, evidencias, inferencia exacta y simulación Monte Carlo.
5	Mejorar interoperabilidad y persistencia	Importación CSV, generación automática de grafos, auto-layout y gestión de modelos RiskFile.
6	Aumentar calidad y mantenibilidad	Automatización de pruebas unitarias, automatización de pruebas end-to-end con Cypress, linter, refactorización y corrección de bugs.
7	Preparar entrega y validación	Rediseño visual, despliegue, ejemplos de uso, memoria y preparación de la defensa.

Tabla 5.3: Criterios de priorización y unidades de trabajo representativas

Esta priorización permitió reducir incertidumbre técnica en las primeras fases y concentrar el último Sprint en convertir el prototipo funcional en una versión más estable, presentable y validable.

5.3 Línea temporal del proyecto

El desarrollo del TFG se distribuyó entre marzo y junio. La fase principal de implementación se concentró entre el 9 de marzo y el 24 de mayo, organizada en un Sprint 0 y tres Sprints de desarrollo. Posteriormente, durante el mes de junio, se planificó la preparación de la defensa, una vez validada la memoria con el tutor del TFG.

La línea temporal general fue la visible en la Figura 5.1

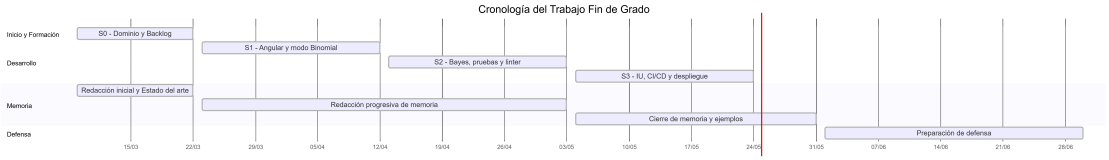

Figura 5.1: Línea temporal del desarrollo del TFG

5.4 Evolución por Sprints

El trabajo se organizó en cuatro iteraciones principales. En cada Sprint se abordó un conjunto de unidades de trabajo y se obtuvo una versión incremental del producto.

Sprint	**Fechas**	**Objetivo**	**Resultado de la revisión**	**Flecos o trabajo pendiente**
Sprint 0	9 marzo – 22 marzo	Puesta en marcha	Entorno instalado, dominio estudiado, Backlog inicial definido y primera aproximación a Daga	Continuar formación técnica y profundizar en Angular, TypeScript y Daga
Sprint 1	23 marzo – 12 abril	Integración visual y modo Binomial	Integración inicial Angular-Daga, creación de nodos y conexiones, primer modo Binomial operativo	Mejorar propagación, corregir errores iniciales y preparar la implementación bayesiana
Sprint 2	13 abril – 3 mayo	Cálculo probabilístico y modo Bayes	Propagación de probabilidades, CPTs, evidencias, pruebas unitarias, Cypress, linter y auto-layout inicial	Mejorar interfaz, corregir bugs del modo Bayes y consolidar importación/exportación
Sprint 3	4 mayo – 24 mayo	Consolidación del producto	Interfaz rediseñada, importación/exportación, pruebas ampliadas, refactorización, CI/CD y despliegue	Cierre de memoria, validación con tutor y preparación de defensa en junio

Tabla 5.4: Resumen de las revisiones y trabajo pendiente por Sprint

5.4.1. Sprint 0: Puesta en marcha

El Sprint 0 tuvo como finalidad preparar el proyecto antes de iniciar el desarrollo funcional. Durante este periodo se estudió el dominio del modelado visual de riesgos y de los modelos probabilísticos, se instaló el entorno de desarrollo y se realizó un primer análisis de la librería Daga.

También se definió el Backlog inicial, identificando las principales funcionalidades que debía incluir el MVP: creación de nodos, conexiones, modo Binomial, modo Bayes, propagación de probabilidades, importación/exportación y pruebas.

La revisión del Sprint permitió confirmar que el proyecto era técnicamente viable, aunque quedaba como fleco principal continuar con la formación en las tecnologías empleadas, especialmente Angular, TypeScript y Daga.

5.4.2. Sprint 1: integración inicial y modo Binomial

El Sprint 1 se centró en construir la primera base funcional de la aplicación. Se realizó la integración inicial entre Angular y Daga, siguiendo el tutorial disponible en la documentación de la librería y apoyándose en una sesión introductoria impartida por personal de Metadev.

Durante este Sprint se implementó la creación de nodos, la creación de conexiones y el estudio de propiedades internas de Daga. También se desarrolló el primer modo de trabajo, el modo Binomial, que permitió validar que el modelo visual podía asociarse a cálculos probabilísticos.

Además, se realizaron bocetos en papel y PowerPoint para explorar la organización de la interfaz. La revisión del Sprint permitió obtener una primera versión funcional, aunque todavía con una interfaz preliminar y con necesidad de mejorar la propagación automática de probabilidades.

5.4.3. Sprint 2: modo Bayes, pruebas y calidad

El Sprint 2 abordó la parte más compleja del proyecto desde el punto de vista algorítmico. En primer lugar, se completó la propagación de probabilidades del modo Binomial. Posteriormente se implementó el modo Bayes, incluyendo tablas de probabilidad condicional, evidencias y recálculo de probabilidades marginales.

Durante este Sprint también se incorporaron pruebas unitarias y pruebas end-to-end con Cypress. Además, se introdujo el linter, que permitió detectar errores de estilo, inconsistencias y algunos problemas de calidad que fueron corregidos durante la iteración.

Otro trabajo relevante fue la creación del auto-layout para el modo Bayes. Esta mejora surgió al detectar que, al importar datos CSV, los nodos podían generarse correctamente, pero quedaban distribuidos de forma poco clara. El auto-layout permitió mejorar la legibilidad inicial de los grafos importados.

La revisión de este Sprint fue positiva, ya que se disponía de una versión funcional de los dos modos principales. Como flecos quedaron la mejora visual de la interfaz, la corrección de bugs del modo Bayes y la consolidación de las funcionalidades de importación y exportación.

5.4.4. Sprint 3: consolidación, rediseño y despliegue

El Sprint 3 se orientó a consolidar la aplicación y convertirla en una versión más presentable y estable. Una de las tareas principales fue el rediseño completo de la interfaz, realizado con el apoyo y consejos de Paco Soria, diseñador de Metadev.

También se implementaron las funcionalidades de importación y exportación de modelos, se ampliaron las pruebas unitarias y end-to-end, y se realizó una refactorización general del código. Esta refactorización permitió separar mejor responsabilidades y mejorar la mantenibilidad del proyecto.

Durante este Sprint se configuró también el flujo de CI/CD y el despliegue, permitiendo publicar una versión accesible de la aplicación. Además, se corrigieron varios errores detectados en el modo Bayes y se redactaron ejemplos contextualizados para mostrar el funcionamiento de la aplicación en modo Binomial y Bayes.

Al finalizar este Sprint, el producto alcanzó una versión suficientemente completa para ser validada y documentada. Como trabajo pendiente quedó el cierre de la memoria con el tutor y la preparación de la defensa durante el mes de junio.

5.5 Capturas del estado del proyecto

Dado que Trello no proporciona una vista histórica exacta del tablero al inicio de cada Sprint, se han incluido dos capturas representativas del seguimiento realizado. La Figura 5.2 muestra las listas correspondientes a los Sprints 0, 1, 2 y 3, permitiendo observar la organización temporal del Backlog. Por otro lado, la Figura 5.3 muestra el estado del Sprint 3, donde se aprecia el uso del workflow mediante columnas como Programar, Aplicar Pruebas de Aceptación, Bugs encontrados y Done. Estas capturas sirven como evidencia visual del método de seguimiento utilizado, mientras que la evolución detallada del proyecto queda recogida en las tablas cronológicas de este capítulo.

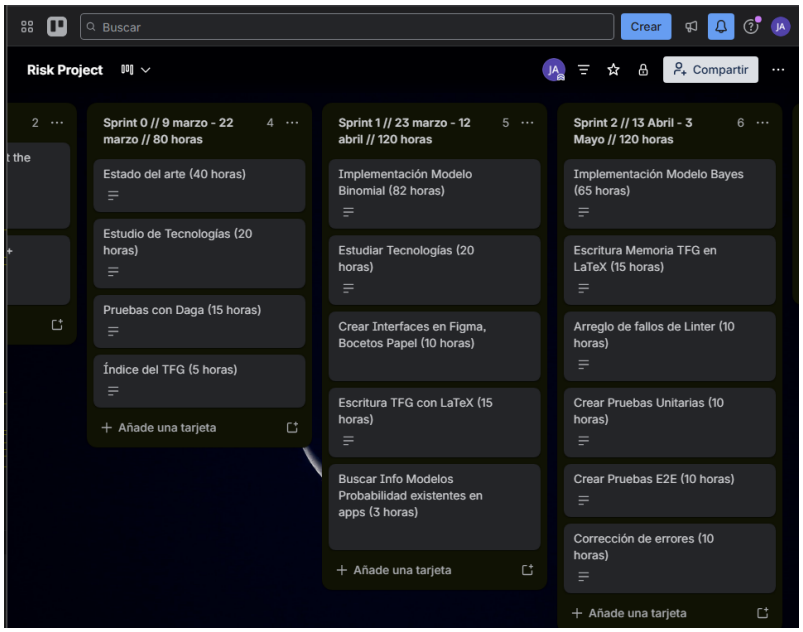

Figura 5.2: Listas del tablero de Trello correspondientes a los Sprints 0, 1 y 2

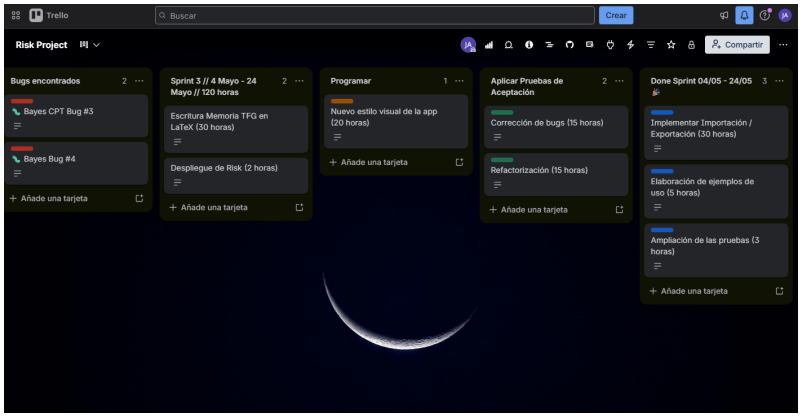

Figura 5.3: Estado del tablero de Trello durante el Sprint 3

5.6 Dedicación temporal

La dedicación durante la fase principal del proyecto fue de aproximadamente 40 horas semanales. Esta dedicación incluyó formación, análisis, programación, pruebas, corrección de errores, diseño de interfaz, documentación y reuniones de seguimiento.

El esfuerzo total estimado de la fase principal de desarrollo asciende a unas 440 horas, sin contar la preparación de la defensa. Esta cifra supera el mínimo de 300 horas asociado al TFG y refleja que el proyecto no se limitó a la implementación de la aplicación, sino que incluyó una fase importante de formación, investigación, diseño, pruebas y documentación.

La memoria se trabajó de forma progresiva durante los Sprints. En los primeros Sprints se dedicó tiempo a la redacción inicial, estado del arte y documentación técnica, mientras que en el último Sprint se intensificó la escritura para cerrar la memoria y preparar ejemplos de uso.

En la Tabla 5.5 se muestra un resumen general de las fases, fechas, dedicación y tareas principales desarrolladas.

Fase	Fechas	Duración	Dedicación	Tareas principales
Sprint 0	9 marzo – 22 marzo	2 semanas	80 h	Estudio del dominio, tecnologías, estado del arte y pruebas con Daga
Sprint 1	23 marzo – 12 abril	3 semanas	120 h	Integración Angular-Daga, modo Binomial, bocetos de interfaz y formación
Sprint 2	13 abril – 3 mayo	3 semanas	120 h	Modo Bayes, CPTs, evidencias, pruebas, linter, auto-layout y bugs
Sprint 3	4 mayo – 24 mayo	3 semanas	120 h	Rediseño de IU, import/export, refactorización, pruebas, CI/CD y despliegue
Preparación de la defensa	Junio	3-4 semanas	25-30 h aprox.	Presentación, demo, ensayos y ajustes tras validación de memoria

Tabla 5.5: Resumen general de las fases, fechas y dedicación del proyecto

CAPÍTULO 6
Conclusiones

El objetivo general de este TFG era diseñar y construir una plataforma web transversal y escalable para la representación gráfica y evaluación dinámica de probabilidades en la gestión de riesgos empresariales. Este objetivo puede considerarse cumplido en gran medida, ya que se ha desarrollado una aplicación web funcional que permite construir modelos de riesgo mediante nodos y conexiones, calcular probabilidades y visualizar los resultados directamente sobre el diagrama.

La solución desarrollada materializa la idea principal del proyecto: transformar cálculos probabilísticos, que tradicionalmente suelen expresarse mediante fórmulas, tablas o código, en una representación visual interactiva. De este modo, el usuario puede comprender mejor la estructura del modelo, modificarla y observar cómo cambian los resultados. Esta idea coincide con la motivación inicial del trabajo, centrada en unir la lógica matemática y la representación visual para facilitar la toma de decisiones en contextos de incertidumbre.

Respecto a los objetivos específicos, el grado de cumplimiento ha sido el que se muestra en la Tabla 6.1.

Objetivo específico	Grado de cumplimiento	Comentario
Diseñar un modelo de dominio específico como base del DSL visual de riesgos	Cumplido	Se ha desarrollado una base funcional orientada al dominio mediante nodos, conexiones, CPTs, evidencias, modos de cálculo y formato RiskFile. La evolución hacia un lenguaje textual independiente con sintaxis propia y parser queda como línea futura.
Integrar la representación gráfica con Daga	Cumplido	La aplicación integra Daga como motor visual para crear, editar y representar grafos de probabilidad de forma interactiva.
Desarrollar una arquitectura web modular	Cumplido	Se ha construido una aplicación Angular/TypeScript organizada en componentes, servicios y utilidades, con separación entre interfaz, cálculo probabilístico, importación/exportación y pruebas. La aplicación puede generarse como versión de producción y publicarse como herramienta web accesible desde navegador.
Implementar cálculo probabilístico	Cumplido	Se han implementado el modo Binomial, el modo Bayesiano, inferencia exacta, Monte Carlo, CPTs, evidencias y propagación de probabilidades.
Permitir la persistencia e intercambio de modelos	Cumplido	La aplicación permite importar datos desde archivos CSV, así como exportar e importar modelos mediante un formato propio llamado RiskFile que almacena la estructura del grafo, sus probabilidades y la información necesaria para reconstruir el modelo posteriormente.
Validar el sistema mediante pruebas y ejemplos	Cumplido	Se han creado pruebas unitarias, pruebas end-to-end automatizadas con Cypress, pruebas de aceptación manuales y ejemplos contextualizados para mostrar el funcionamiento de la herramienta.

Tabla 6.1: Grado de cumplimiento de los objetivos específicos del proyecto

6.1 Estado actual de la aplicación

El resultado actual del proyecto es una aplicación web funcional denominada Risk, desarrollada con Angular y TypeScript e integrada con la librería de diagramación Daga. La aplicación permite trabajar sobre un lienzo visual en dos modos principales: modo Binomial y modo Bayesiano.

El modo Binomial permite crear modelos de ramificación mediante conexiones ponderadas, calcular probabilidades teóricas y realizar simulaciones mediante Monte Carlo. Este modo resulta útil para representar escenarios probabilísticos encadenados o árboles de decisión sencillos.

El modo Bayesiano permite construir redes bayesianas binarias, definir tablas de probabilidad condicional, fijar evidencias y recalcular probabilidades marginales. Además, se han incorporado funcionalidades de importación de datos CSV, aprendizaje de parámetros mediante MLE o EM cuando procede, auto-layout para organizar redes importadas y persistencia mediante archivos RiskFile en formato JSON versionado.

Estas funcionalidades deben interpretarse dentro del alcance de un MVP académico avanzado. Algunas de ellas, como el aprendizaje de parámetros desde CSV o la inferencia aproximada mediante Monte Carlo, se han incorporado como mecanismos funcionales y extensibles, pero no como una plataforma estadística completa comparable a herramientas especializadas. Su objetivo dentro del TFG es demostrar la viabilidad técnica del enfoque y establecer una base coherente para futuras ampliaciones.

La aplicación también cuenta con una interfaz rediseñada durante el último Sprint con apoyo de diseño externo, pruebas automatizadas, herramientas de calidad de código, integración continua y despliegue. En conjunto, el estado actual puede considerarse el de un MVP funcional avanzado, suficiente para demostrar la viabilidad técnica del enfoque y servir como base para futuras ampliaciones.

No obstante, la aplicación todavía presenta limitaciones propias del alcance de un TFG. Actualmente, el modo Bayes trabaja principalmente con variables binarias, la inferencia exacta se limita a redes pequeñas y las redes de mayor tamaño dependen de aproximaciones mediante Monte Carlo. Tampoco se ha implementado una base de datos centralizada, autenticación de usuarios ni un backend de cálculo independiente. Estas limitaciones no invalidan el resultado, pero delimitan claramente el alcance de la versión actual.

6.2 Relación con la formación recibida en la titulación

La realización de este TFG ha permitido aplicar de forma integrada muchos conocimientos adquiridos durante el Grado en Ingeniería Informática. A diferencia de otros trabajos más acotados, este proyecto ha requerido combinar análisis del problema, diseño de software, programación, pruebas, gestión de proyecto, documentación y despliegue.

Una de las asignaturas más relevantes ha sido Proyecto de Ingeniería de Software (PIN). La experiencia previa en PIN resultó especialmente útil porque permitió afrontar el TFG con una mentalidad más cercana a la de un proyecto real: definición de Backlog, organización en Sprints, priorización de tareas, seguimiento mediante tablero Kanban, pruebas de aceptación y revisión incremental del producto. Haber trabajado previamente en un contexto similar facilitó la adaptación de una metodología ágil al desarrollo individual del TFG.

También han sido importantes las asignaturas relacionadas con Ingeniería del Software, especialmente en aspectos como separación de responsabilidades, diseño modular, uso de diagramas, definición de requisitos y mantenimiento del código. El proyecto obligó a tomar decisiones de arquitectura y a refactorizar partes del sistema cuando el tamaño de la aplicación creció, por lo que los conceptos de diseño software fueron especialmente relevantes.

Las asignaturas de programación y estructuras de datos también resultaron fundamentales, ya que la aplicación trabaja internamente con grafos, nodos, conexiones, relaciones dirigidas y estructuras de datos asociadas a probabilidades. Del mismo modo, los conocimientos matemáticos y estadísticos adquiridos durante la titulación ayudaron a comprender mejor la base probabilística del proyecto, especialmente en lo relacionado con inferencia, simulación y modelos bayesianos.

Por último, las asignaturas relacionadas con interfaces de usuario, desarrollo web y bases de datos aportaron una base útil para diseñar una aplicación comprensible, estructurada y preparada para futuras ampliaciones. Aunque la versión actual no utiliza una base de datos tradicional, los conceptos de persistencia, estructuración de información y formatos de intercambio han sido relevantes en el diseño del formato RiskFile.

6.3 Reflexión personal y profesional

Desde un punto de vista personal, este TFG ha supuesto uno de los proyectos más completos y exigentes que he realizado durante la titulación. No se ha tratado únicamente de implementar una aplicación, sino de comprender un dominio complejo, aprender tecnologías nuevas, tomar decisiones de diseño, corregir errores importantes y construir una herramienta que pudiera tener continuidad más allá del ámbito académico.

Uno de los principales aprendizajes ha sido comprobar que una solución técnicamente correcta no siempre es suficiente. En un proyecto como este, donde el objetivo es hacer comprensibles modelos probabilísticos, la interfaz, la claridad visual y la experiencia de usuario son tan importantes como los algoritmos de cálculo. Esta idea se hizo especialmente evidente durante el rediseño de la aplicación y la preparación de ejemplos contextualizados.

También ha sido importante para mí el aprender a trabajar con incertidumbre técnica. Al inicio del proyecto había varias tecnologías desconocidas, especialmente Daga y Angular. Además, el modo Bayesiano planteó retos algorítmicos relevantes, como la correcta propagación de evidencias, la normalización de probabilidades y la precisión numérica. Resolver estos problemas me permitió adquirir más confianza técnica y mejorar la capacidad de análisis y depuración.

A nivel profesional, el proyecto también me ha permitido acercarme a una forma de trabajo similar a la de un entorno real: uso de control de versiones, pruebas automatizadas, linter, integración continua, despliegue, revisión con tutores y feedback de personas de Metadev. También he podido entender mejor la importancia de documentar decisiones, planificar por iteraciones y validar progresivamente el producto.

En conjunto, la realización del TFG ha aportado una mejora significativa para mi perfil tanto en competencias técnicas como en competencias transversales: autonomía, organización, comunicación, capacidad de aprendizaje, gestión del tiempo y adaptación ante problemas no previstos.

6.4 Trabajo futuro

6.5 Trabajo futuro

El resultado obtenido abre varias líneas de continuación. Algunas están relacionadas con la ampliación funcional de la aplicación, mientras que otras se orientan a su aplicación real en distintos ámbitos.

Una primera línea de trabajo futuro sería completar el concepto de DSL. La versión actual se centra en el modelado visual y en un formato JSON de persistencia, pero podría evolucionar hacia un lenguaje textual o híbrido que permitiera definir modelos mediante reglas de dominio. Esto permitiría combinar la potencia de una sintaxis formal con la claridad de la representación visual.

Otra línea importante sería ampliar el modo Bayesiano para soportar variables no binarias, variables categóricas con más de dos estados e incluso variables continuas discretas. Esta mejora acercaría la herramienta a problemas reales más complejos y aumentaría su aplicabilidad en sectores como seguros, banca, ingeniería o salud.

También sería interesante incorporar un backend especializado para cálculo intensivo, persistencia centralizada y gestión de usuarios. Aunque la ejecución en navegador es adecuada para el MVP, un backend permitiría almacenar modelos de forma colaborativa, gestionar versiones, compartir proyectos entre usuarios y ejecutar cálculos más pesados en servidor. Como alternativa o complemento para estas operaciones matemáticas intensivas, C# y ASP.NET Core representan una opción sólida para construir APIs de cálculo, servicios de negocio o módulos independientes de procesamiento. ASP.NET Core es un framework web multiplataforma, de alto rendimiento y código abierto para construir aplicaciones y servicios modernos sobre .NET [32]. Esta opción resultaría especialmente interesante si el sistema necesitara separar el cálculo probabilístico pesado del frontend o integrarse con entornos empresariales basados en tecnologías Microsoft.

Otra línea de trabajo futuro sería realizar una evaluación formal de usabilidad con usuarios. Aunque la interfaz actual se ha diseñado buscando claridad visual y facilidad de uso, sería necesario validar esta percepción mediante pruebas con usuarios reales, cuestionarios de satisfacción, observación de tareas y métricas como tiempo de realización, número de errores o facilidad percibida. Esta evaluación permitiría comprobar hasta qué punto la herramienta resulta realmente intuitiva para perfiles no expertos en estadística o programación.

También sería conveniente llevar a cabo un estudio de rendimiento y escalabilidad de la aplicación. La versión actual permite trabajar con modelos de tamaño reducido o medio dentro del navegador, pero sería necesario medir de forma sistemática el comportamiento del sistema ante grafos más grandes, mayor número de nodos, más evidencias y tablas de probabilidad condicional de mayor tamaño. A partir de estas pruebas podrían definirse límites recomendados de uso, mecanismos de aviso al usuario, estrategias de optimización o cambios automáticos hacia técnicas de inferencia aproximada cuando la inferencia exacta resulte demasiado costosa.

Por último, desde el punto de vista técnico, sería interesante reforzar la extensibilidad de la arquitectura. Aunque el proyecto se ha organizado de forma modular, una evolución futura podría definir interfaces más estables para incorporar nuevos modos de cálculo, nuevos formatos de entrada y salida o nuevas técnicas de inferencia. Esta mejora facilitaría que la aplicación creciera sin modificar de forma significativa los módulos ya existentes.

Desde el punto de vista de uso real, una posible línea de continuidad sería aplicar la herramienta a casos de estudio sectoriales. Por ejemplo, en seguros podría utilizarse para modelar riesgo de churn, probabilidad de siniestro o estimación de primas. En banca e inversiones podría emplearse para representar dependencias entre factores macroeconómicos, riesgo geopolítico y evolución de mercados. En ingeniería aeroespacial podría explorarse su integración con modelos de riesgo de componentes críticos, conectando la herramienta con proyectos relacionados con SysML2 o Apricot, tal como se planteaba en la motivación inicial del proyecto.

Otra línea futura sería mejorar la parte explicativa de la aplicación. Además de mostrar resultados, la herramienta podría explicar por qué cambia una probabilidad, qué evidencias influyen más en un nodo o qué caminos del grafo tienen mayor impacto. Esto reforzaría el objetivo de transparencia y ayudaría a usuarios no expertos a interpretar mejor los modelos.

También sería útil incorporar funcionalidades colaborativas, como comentarios sobre nodos, historial de cambios, comparación entre versiones de un modelo o exportación de informes automáticos. Estas mejoras permitirían utilizar la aplicación no solo como herramienta de cálculo, sino también como soporte para discusión y toma de decisiones en equipos multidisciplinares.

Finalmente, una línea especialmente interesante sería conectar la herramienta con fuentes de datos reales. La importación CSV ya permite una primera aproximación, pero en el futuro podrían incorporarse conectores con APIs externas, bases de datos corporativas o sistemas de monitorización. Esto permitiría actualizar modelos de riesgo de forma más dinámica y acercar la aplicación a un entorno de producción real.

En conclusión, este TFG ha permitido desarrollar una base sólida para una herramienta de modelado visual y cálculo probabilístico de riesgos. La aplicación actual demuestra la viabilidad del enfoque y deja abiertas múltiples posibilidades de evolución, tanto desde el punto de vista técnico como desde su aplicación en contextos reales de toma de decisiones.

Referencias

[1] Douglas W. Hubbard. *The Failure of Risk Management: Why It's Broken and How to Fix It*. John Wiley & Sons, 2009.

[2] Yu-Chuan Chen et al. Learning discrete Bayesian networks from continuous data. *Journal of Artificial Intelligence Research*, 59:103–132, 2017.

[3] Jonathon Love, Ravi Selker, Maarten Marsman, Tahira Jamil, Damian Dropmann, Alexander J. Verhagen y Eric-Jan Wagenmakers. JASP: graphical statistical software for common statistical designs. *Journal of Statistical Software*, 88(2):1–17, 2019.

[4] John Salvatier, Thomas V. Wiecki y Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[5] Arie van Deursen, Paul Klint y Joost Visser. Domain-specific languages: an annotated bibliography. *ACM SIGPLAN Notices*, 35(6):26–36, 2000. https://doi.org/10.1145/352029.352035 [Consulta: marzo-2026]

[6] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer y Gary D. Bader. Cytoscape.js 2023 update: a graph theory library for visualization and analysis. *Bioinformatics*, 2023.

[7] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[8] Nassim N. Taleb. *The Black Swan: The Impact of the Highly Improbable*. Random House, 2007.

[9] Martin Fowler. *Domain-Specific Languages*. Addison-Wesley Professional, 2010.

[10] Marjan Mernik, Jan Heering y Anthony M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005. https://doi.org/10.1145/1118890.1118892 [Consulta: marzo-2026]

[11] Marek J. Druzdzel. SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models. En *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, páginas 342–343, 1999.

[12] International Organization for Standardization. *ISO 31000:2018 Risk Management — Guidelines*. ISO, 2018.

[13] BayesFusion, LLC. GeNIe Modeler and SMILE Engine, 2024. https://www.bayesfusion.com/ [Consulta: marzo-2026]

[14] Decisioneering, Inc. *Crystal Ball 7.1*. Decisioneering, Inc., 2005.

[15] IBM Corporation. IBM SPSS Statistics, 2024. https://www.ibm.com/products/spss-statistics [Consulta: marzo-2026]

[16] JASP Team. *JASP (Version 0.95.4)* [software informático], 2025. https://jasp-stats.org/ [Consulta: marzo-2026]

[17] Lionel Jouffe y Paul Munteanu. *BayesiaLab*. Bayesia S.A.S., 2001.

[18] Lumivero. *@RISK*, 2026. https://www.lumivero.com/products/at-risk [Consulta: marzo-2026]

[19] MathWorks. Statistics and Machine Learning Toolbox. https://www.mathworks.com/ [Consulta: marzo-2026]

[20] Minitab, LLC. *Minitab Statistical Software*, 2024. https://www.minitab.com/ [Consulta: marzo-2026]

[21] PyMC Development Team. *PyMC*, 2026. https://www.pymc.io/ [Consulta: marzo-2026]

[22] Wolfram Research. Wolfram Language: Probability and Statistics. https://www.wolfram.com/ [Consulta: marzo-2026]

[23] Northwoods Software. *GoJS: JavaScript Diagramming Library*. https://gojs.net/ [Consulta: marzo-2026]

[24] JointJS. *JointJS: JavaScript Diagramming Library*. https://www.jointjs.com/ [Consulta: marzo-2026]

[25] xyflow. *React Flow: Node-Based UIs in React*. https://reactflow.dev/ [Consulta: marzo-2026]

[26] Mermaid. *Mermaid: Diagramming and Charting Tool*. https://mermaid.js.org/ [Consulta: marzo-2026]

[27] Observable. *D3.js: The JavaScript Library for Bespoke Data Visualization*. https://d3js.org/ [Consulta: marzo-2026]

[28] Metadev. *Daga: Modeling Diagramming Library*. https://metadev.pro/products/daga/ [Consulta: marzo-2026]

[29] Metadev. *Metadev launches Daga*. Metadev, 2024. https://metadev.pro/resources/media/2024-01-29-metadev-launches-daga/ [Consulta: marzo-2026]

[30] Angular Team. *Angular Documentation*. https://angular.dev/ [Consulta: abril-2026]

[31] Microsoft. *TypeScript Documentation*. https://www.typescriptlang.org/ [Consulta: marzo-2026]

[32] Microsoft. *ASP.NET Core Documentation*. https://learn.microsoft.com/en-us/aspnet/core/ [Consulta: abril-2026]

[33] World Economic Forum. *The Global Risks Report 2025*. *World Economic Forum, 2025*. https://www.weforum.org/publications/global-risks-report-2025/ [Consulta: marzo-2026]

[34] Munich Re. *Natural disasters in 2024: Climate change is showing its claws*. *Munich Re, 2025*. https://www.munichre.com/en/company/media-relations/media-information-and-corporate-news/media-information/2025/natural-disaster-figures-2024.html [Consulta: marzo-2026]

APÉNDICE A
Configuración del sistema

En este apéndice se describen los pasos necesarios para preparar, ejecutar y validar la aplicación desarrollada durante el Trabajo de Fin de Grado. El objetivo de esta sección es facilitar la reproducción del entorno de desarrollo y permitir que otro usuario pueda poner en marcha el sistema de forma local.

A.1 Requisitos previos

Para ejecutar la aplicación es necesario disponer de un entorno de desarrollo compatible con tecnologías web modernas. Durante el desarrollo del proyecto se utilizaron principalmente Node.js, npm, Angular, TypeScript y las herramientas asociadas al ecosistema frontend.

Los requisitos principales son los siguientes:

- Tener instalado Node.js.
- Tener instalado npm como gestor de paquetes.
- Disponer de Git para clonar o gestionar el repositorio del proyecto,
- Utilizar un editor de código compatible con TypeScript, como Visual Studio Code.
- Disponer de un navegador web moderno para ejecutar y validar la aplicación.

Además, para las fases de pruebas y despliegue se utilizaron herramientas adicionales como Vitest, Cypress, ESLint, Prettier y scripts específicos de publicación.

A.2 Instalación del proyecto

Una vez obtenido el código fuente del proyecto, el primer paso consiste en instalar las dependencias necesarias. Para ello, desde la carpeta raíz del proyecto se ejecuta el siguiente comando:

```
npm install
```

Este comando descarga las librerías necesarias para el funcionamiento de la aplicación, incluyendo Angular, Daga, RxJS y las herramientas de pruebas y calidad de código utilizadas durante el desarrollo.

A.3 Ejecución en entorno local

Para ejecutar la aplicación en modo desarrollo se utiliza el servidor local proporcionado por Angular. Desde la carpeta raíz del proyecto se ejecuta:

```
npm start
```

Una vez iniciado el servidor, la aplicación queda disponible desde el navegador en la dirección:

```
http://localhost:4200
```

Desde esta interfaz el usuario puede acceder al lienzo principal de la aplicación, crear modelos visuales, seleccionar el modo de cálculo, configurar nodos y relaciones, e importar o exportar modelos.

A.4 Ejecución de pruebas

Durante el desarrollo se incorporaron distintos tipos de pruebas para validar el funcionamiento del sistema. Las pruebas unitarias permiten comprobar la lógica interna de cálculo, normalización de probabilidades, inferencia y utilidades auxiliares.

Para ejecutar las pruebas unitarias se puede utilizar el siguiente comando:

```
npm test
```

En caso de ejecutar pruebas end-to-end, se utiliza la herramienta configurada para validar flujos completos de usuario, como la creación de modelos, la edición de nodos, la importación de datos y la visualización de resultados.

Se tendrá que tener una terminal con el servidor iniciado, y en otra terminal a parte ejecutar el siguiente comando:

```
npm run cy:run
```

o, alternativamente para ver la interfaz de cypress:

```
npm run cy:open
```

Estas pruebas ayudan a comprobar que las funcionalidades principales continúan funcionando correctamente tras los cambios realizados en el código.

A.5 Generación de la versión de producción

Para generar una versión optimizada de la aplicación preparada para despliegue, se ejecuta:

```
npm run build
```

Este proceso compila el proyecto, optimiza los archivos generados y produce una versión lista para ser publicada en un entorno web.

APÉNDICE B
Guía de uso de la aplicación

En este capítulo se presenta una guía básica de uso de la aplicación desarrollada. El objetivo no es documentar de forma exhaustiva todas las opciones internas del sistema, sino mostrar el flujo principal de trabajo y explicar cómo un usuario puede crear, configurar, analizar, importar y exportar modelos de riesgo.

Risk se ha diseñado como una herramienta web visual para el modelado y cálculo de probabilidades mediante grafos. Su funcionamiento se basa en un lienzo interactivo, construido sobre la librería Daga, donde el usuario puede crear nodos, conectarlos y asignar valores probabilísticos. A partir de esta estructura visual, el sistema recalcula y muestra los resultados directamente sobre el diagrama.

La herramienta dispone de dos modos principales de trabajo: modo Binomial y modo Bayesiano. El primero está orientado a árboles o grafos de probabilidad ponderados, mientras que el segundo permite construir redes bayesianas binarias con tablas de probabilidad condicional y evidencias.

B.1 Acceso a la aplicación

La aplicación se ejecuta desde el navegador web. Durante el desarrollo se utilizó un entorno local, accesible mediante la dirección:

`http://localhost:4200`

En la versión desplegada, la aplicación puede accederse desde el entorno web preparado para su validación. Al entrar, el usuario visualiza la interfaz principal, formada por un área central de trabajo y controles auxiliares para seleccionar el modo de modelado y realizar acciones sobre el modelo como se puede ver en la Figura **??**.

B.2 Estructura general de la interfaz

La interfaz de la aplicación se organiza alrededor de un lienzo visual. Este lienzo es el elemento principal de interacción, ya que en él se representan los nodos, las conexiones y los resultados probabilísticos.

De forma general, la interfaz se divide en las siguientes zonas:

Zona de la interfaz	Función
Lienzo de diagramación	Área principal donde el usuario crea, mueve y conecta nodos
Controles de modo	Permiten seleccionar entre modo Binomial y modo Bayesiano
Acciones del modelo	Permiten importar, exportar o limpiar modelos
Decoradores visuales	Muestran probabilidades, pesos, evidencias y marginales sobre el diagrama

Tabla B.1: Zonas principales de la interfaz y sus funciones específicas

El diseño busca que el usuario no tenga que interpretar fórmulas externas ni revisar tablas separadas para comprender el estado del modelo. La información relevante se muestra directamente sobre los elementos visuales del grafo.

B.3 Creación de un modelo visual

El flujo básico de uso comienza con la creación de un modelo de riesgo sobre el lienzo. Para ello, el usuario debe añadir nodos que representen eventos, variables o estados del sistema analizado. Posteriormente, puede establecer conexiones dirigidas entre ellos para representar dependencias, caminos probabilísticos o relaciones causales.

De forma simplificada, el proceso general es el siguiente:

1. Seleccionar el modo de trabajo.
2. Crear los nodos del modelo.
3. Conectar los nodos mediante relaciones dirigidas.
4. Configurar probabilidades, pesos o tablas condicionales.
5. Observar los resultados calculados sobre el diagrama.
6. Guardar o exportar el modelo si se desea conservarlo.
7. **(Opcionalmente)** Ejecutar, si procede, una simulación Monte Carlo.

B.4 Uso del modo Binomial

El modo Binomial permite construir modelos basados en conexiones ponderadas. Este modo resulta útil para representar árboles de decisión, escenarios encadenados o caminos probabilísticos en los que cada transición tiene una probabilidad asociada.

En este modo, cada nodo representa un evento o estado, y cada conexión representa una transición entre dos elementos. El usuario puede asignar pesos o probabilidades a las conexiones, y el sistema recalcula automáticamente las probabilidades afectadas.

El flujo de uso del modo Binomial es el siguiente:

Paso	Acción
1	Seleccionar el modo Binomial
2	Crear los nodos que representan los eventos del modelo
3	Conectar los nodos para formar los caminos probabilísticos
4	Asignar pesos o probabilidades a las conexiones
5	Revisar las probabilidades teóricas calculadas por el sistema
6	Ejecutar, si procede, una simulación Monte Carlo

Tabla B.2: Pasos para la creación y cálculo de un modelo en modo Binomial

Cuando el usuario modifica el peso de una conexión, la aplicación actualiza los valores asociados y mantiene la coherencia del modelo. Esto evita tener que recalcular manualmente cada camino probabilístico.

Además del cálculo teórico, el modo Binomial permite realizar simulaciones mediante Monte Carlo. Esta funcionalidad ejecuta múltiples iteraciones sobre el modelo definido y permite comparar los resultados simulados con los valores esperados. De esta forma, el usuario puede comprobar experimentalmente el comportamiento del modelo.

Un ejemplo del modelo lo podemos encontrar en la Figura 3.2

B.5 Uso del modo Bayesiano

El modo Bayesiano permite construir redes bayesianas binarias. En este modo, los nodos representan variables con dos posibles estados, como por ejemplo “sí/no”, “verdadero/falso” o “ocurre/no ocurre”. Las conexiones dirigidas representan dependencias entre variables.

A diferencia del modo Binomial, en el modo Bayesiano cada nodo tiene asociada una tabla de probabilidad condicional o CPT. Esta tabla define la probabilidad del nodo en función del estado de sus nodos padre.

El flujo básico de uso es el siguiente:

La edición de CPTs permite definir cómo afecta cada combinación de causas al nodo correspondiente. Por ejemplo, si un nodo depende de dos variables padre, su tabla condicional debe indicar la probabilidad del evento para cada combinación posible de esos padres.

Paso	Acción
1	Seleccionar el modo Bayesiano
2	Crear los nodos que representan las variables del modelo
3	Conectar los nodos para definir dependencias causales
4	Editar las tablas de probabilidad condicional
5	Fijar evidencias cuando se disponga de información observada
6	Observar la actualización de probabilidades marginales

Tabla B.3: Pasos para la creación y cálculo de un modelo en modo Bayesiano

Una de las funcionalidades principales del modo Bayesiano es la posibilidad de fijar evidencias. Una evidencia representa información conocida sobre una variable. Al fijar una evidencia positiva o negativa en un nodo, el sistema recalcula las probabilidades marginales del resto de la red.

Este comportamiento permite responder preguntas del tipo: "si este evento ha ocurrido, ¿cómo cambia la probabilidad de los demás eventos?". Por ello, el modo Bayesiano resulta especialmente útil para analizar propagación de incertidumbre, diagnóstico y dependencias causales.

Un ejemplo del modelo lo podemos encontrar en la siguiente imagen:

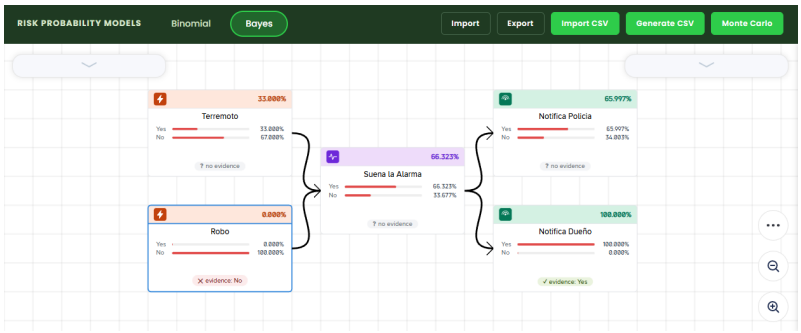

Figura B.1: Ejemplo de uso del modo bayes

B.6 Visualización de resultados

Una característica central de la aplicación es que los resultados se muestran directamente sobre el diagrama. Esto evita separar el modelo visual de los valores calculados.

En el modo Binomial, la aplicación muestra información asociada a nodos y conexiones, como pesos de conexión y probabilidades teóricas. En el modo Bayesiano, se muestran probabilidades marginales, evidencias fijadas y valores asociados a las CPTs.

Esta visualización integrada aporta varias ventajas:

Ventaja	Descripción
Comprensión directa	El usuario interpreta los resultados en el propio modelo
Trazabilidad	Cada valor aparece vinculado al nodo o conexión correspondiente
Menor carga cognitiva	No es necesario relacionar manualmente tablas externas con el diagrama
Detección de errores	Los valores incoherentes son más fáciles de identificar visualmente

Tabla B.4: Ventajas de la representación visual integrada en el diagrama

Cuando el usuario modifica el modelo, la aplicación actualiza los resultados de forma interactiva. Esto permite experimentar con distintos escenarios y observar cómo cambian las probabilidades.

B.7 Importación de datos CSV

La aplicación permite importar datos desde archivos CSV. Esta funcionalidad está especialmente orientada al modo Bayesiano, ya que permite alimentar el modelo con datos externos y aprender parámetros a partir de observaciones.

El archivo CSV puede contener variables en columnas y observaciones en filas. Además, puede incluir una directiva opcional para definir las relaciones entre nodos mediante una línea de estructura. Esta directiva permite indicar las conexiones del grafo sin tener que dibujarlas manualmente.

Un ejemplo conceptual de estructura sería:

edges: RiesgoGeopolitico->Mercado; Mercado->Inversion

A partir de esta información, la aplicación puede crear nodos, establecer conexiones y calcular las probabilidades correspondientes. Cuando existen datos completos, puede aplicarse aprendizaje de parámetros mediante MLE. Si existen valores faltantes, puede emplearse EM para estimar parámetros cuando proceda.

El flujo de importación es el siguiente:

Durante el desarrollo se incorporó un sistema de auto-layout para mejorar la disposición visual de los modelos importados. Esta funcionalidad permite que los nodos no aparezcan desordenados o separados, sino distribuidos de forma más comprensible según sus relaciones.

Paso	Acción
1	Seleccionar la opción de importar CSV
2	Cargar el archivo con los datos
3	Interpretar las variables como nodos
4	Crear relaciones si el archivo contiene estructura
5	Calcular o actualizar probabilidades
6	Organizar el modelo mediante auto-layout

Tabla B.5: Pasos para la importación de un modelo desde un archivo CSV

B.8 Exportación e importación de modelos RiskFile

Además de importar datos CSV, la aplicación permite guardar y recuperar modelos completos mediante el formato RiskFile. Este formato está basado en JSON y contiene tanto la estructura visual del diagrama como el estado probabilístico asociado.

La exportación RiskFile permite guardar:

Elemento guardado	Descripción
Nodos	Elementos visuales del modelo
Conexiones	Relaciones entre nodos
Probabilidades	Valores asociados al modelo
CPTs	Tablas de probabilidad condicional del modo Bayesiano
Evidencias	Estados fijados por el usuario
Versión del archivo	Información necesaria para migraciones futuras

Tabla B.6: Elementos que se almacenan al exportar un modelo en formato RiskFile

El flujo de exportación es sencillo: el usuario crea o modifica un modelo y selecciona la opción de exportar. La aplicación genera un archivo JSON que puede almacenarse, compartirse o versionarse.

La importación RiskFile realiza el proceso inverso. El usuario selecciona un archivo previamente exportado y la aplicación restaura tanto la estructura visual como el estado probabilístico. Esto permite continuar trabajando sobre modelos anteriores sin tener que reconstruirlos desde cero.

Esta funcionalidad resulta importante porque convierte el modelo en un artefacto reutilizable, auditable y portable.

B.9 Ejemplos de uso

Para facilitar la comprensión de la aplicación, durante el desarrollo se prepararon ejemplos contextualizados para los dos modos principales.

En el modo Binomial, un ejemplo típico podría representar una secuencia de decisiones o eventos probabilísticos. Por ejemplo, un escenario de evaluación de riesgo donde un evento inicial puede derivar en varias ramas con distintas probabilidades.

En el modo Bayesiano, un ejemplo puede representar una red de dependencias entre eventos. Por ejemplo, un modelo en el que factores externos influyen sobre una alarma, un fallo, una decisión financiera o una situación de riesgo. Al fijar una evidencia, el usuario puede observar cómo cambia la probabilidad del resto de variables.

Estos ejemplos ayudan a mostrar que la aplicación no está limitada a un único dominio. Aunque el proyecto se plantea en el contexto del cálculo de riesgos, la estructura basada en grafos permite aplicar la herramienta en ámbitos como seguros, banca, inversiones, ingeniería, misiones espaciales o análisis de decisiones.

B.10 Flujo completo de trabajo

A modo de resumen, el flujo completo de uso de la aplicación puede describirse de la siguiente forma:

Seleccionar modo

↓

Crear o importar modelo

↓

Editar nodos, conexiones y probabilidades

↓

Recalcular resultados

↓

Visualizar probabilidades sobre el diagrama

↓

Simular o fijar evidencias

↓

Exportar modelo

Este flujo refleja la filosofía principal de la herramienta: el usuario trabaja directamente sobre el modelo visual y recibe feedback inmediato del sistema. La aplicación busca que el razonamiento probabilístico sea más comprensible, trazable e interactivo.

B.11 Limitaciones de uso actuales

La versión actual de la aplicación cumple los objetivos principales del TFG, pero presenta algunas limitaciones propias de un MVP.

En primer lugar, el modo Bayesiano trabaja con variables binarias, por lo que no se contemplan todavía variables categóricas con más de dos estados ni variables continuas. En segundo lugar, la inferencia exacta está limitada a redes pequeñas, utilizando técnicas aproximadas como Monte Carlo cuando el tamaño del modelo aumenta. Además, la aplicación no incorpora todavía autenticación, almacenamiento en servidor ni colaboración multiusuario.

Estas limitaciones no impiden utilizar la herramienta para validar el enfoque propuesto, pero marcan posibles líneas de evolución para futuras versiones.

APÉNDICE C
Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

C.1 Reflexión sobre la relación del TFG con los ODS y con los ODS más relacionados

Este TFG, se relaciona principalmente con los Objetivos de Desarrollo Sostenible desde una perspectiva tecnológica, educativa y de apoyo a la toma de decisiones. Aunque la aplicación desarrollada no está orientada de forma directa a resolver problemas sociales o ambientales concretos, sí proporciona una base tecnológica que puede aplicarse en contextos donde la comprensión del riesgo, la incertidumbre y la toma de decisiones informada resultan fundamentales.

El ODS con mayor relación con este proyecto es el ODS 9: Industria, innovación e infraestructuras. Este objetivo promueve el desarrollo de soluciones tecnológicas innovadoras, infraestructuras resilientes y procesos de industrialización sostenibles. La aplicación desarrollada encaja especialmente en este ámbito porque propone una herramienta software innovadora para modelar visualmente riesgos mediante grafos, calcular probabilidades y representar dependencias entre eventos. Frente a enfoques tradicionales basados en hojas de cálculo, tablas o fórmulas poco visibles, este proyecto plantea una solución web interactiva en la que el modelo se construye directamente sobre un diagrama. Esta aproximación puede ayudar a mejorar la forma en que organizaciones e industrias analizan escenarios complejos, identifican relaciones causales y evalúan posibles consecuencias antes de tomar decisiones.

Además, el proyecto tiene relación con la innovación porque integra diferentes tecnologías modernas: Angular, TypeScript, la librería de diagramación Daga, inferencia bayesiana, simulaciones Monte Carlo, importación de datos CSV y persistencia mediante archivos JSON versionados. Esta combinación permite construir una herramienta modular, extensible y preparada para futuras ampliaciones. En este sentido, el TFG no solo desarrolla una aplicación concreta, sino que también explora una línea de trabajo más amplia: acercar modelos probabilísticos avanzados a usuarios que no necesariamente son expertos en estadística o programación.

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza				X
ODS 2. Hambre cero				X
ODS 3. Salud y bienestar			X	
ODS 4. Educación de calidad		X		
ODS 5. Igualdad de género				X
ODS 6. Agua limpia y saneamiento				X
ODS 7. Energía asequible y no contaminante			X	
ODS 8. Trabajo decente y crecimiento económico		X		
ODS 9. Industria, innovación e infraestructuras	X			
ODS 10. Reducción de las desigualdades			X	
ODS 11. Ciudades y comunidades sostenibles			X	
ODS 12. Producción y consumo responsables			X	
ODS 13. Acción por el clima			X	
ODS 14. Vida submarina				X
ODS 15. Vida de ecosistemas terrestres				X
ODS 16. Paz, justicia e instituciones sólidas		X		
ODS 17. Alianzas para lograr objetivos	X			

Tabla C.1: Relación del proyecto con los Objetivos de Desarrollo Sostenible (ODS)

También existe una relación significativa con el ODS 4: Educación de calidad. Aunque la aplicación no ha sido desarrollada específicamente como herramienta educativa, su enfoque visual facilita la comprensión de conceptos probabilísticos complejos. Las redes bayesianas, las probabilidades condicionales o las simulaciones Monte Carlo pueden resultar difíciles de entender cuando se presentan únicamente mediante fórmulas o código. Sin embargo, al representarlas como nodos, conexiones, evidencias y resultados visibles sobre un grafo, el usuario puede comprender mejor cómo se propaga la incertidumbre dentro de un sistema. Por ello, la herramienta puede tener utilidad en contextos formativos, tanto para estudiantes como para profesionales que necesiten familiarizarse con el razonamiento probabilístico.

El proyecto también puede relacionarse en un grado medio con el ODS 8: Trabajo decente y crecimiento económico. La gestión adecuada del riesgo es una parte importante de la actividad empresarial, especialmente en sectores como banca, seguros, inversiones, ingeniería o planificación de proyectos. Una herramienta que permita representar riesgos de forma clara y calcular probabilidades de manera más comprensible puede contribuir a mejorar procesos de análisis, reducir incertidumbre y apoyar decisiones más informadas. Esto no implica que la aplicación genere crecimiento económico por sí misma, pero sí que puede formar parte de soluciones orientadas a mejorar la eficiencia, la transparencia y la calidad de la toma de decisiones en organizaciones.

Otro objetivo relacionado es el ODS 17: Alianzas para lograr los objetivos. Este TFG se ha desarrollado en un contexto de colaboración entre la universidad y la empresa Metadev, combinando conocimiento académico, necesidades tecnológicas reales y orientación práctica. La integración con Daga, el apoyo recibido por parte de profesionales de la empresa y la revisión del producto durante el desarrollo reflejan una forma de trabajo colaborativa. Esta relación universidad-empresa es relevante porque permite que un proyecto académico no se limite a un ejercicio teórico, sino que pueda conectarse con tecnologías, metodologías y necesidades más cercanas al entorno profesional.

En menor medida, la aplicación podría relacionarse con otros ODS dependiendo del ámbito en el que se utilice. Por ejemplo, podría tener una relación indirecta con el ODS 3: Salud y bienestar si en el futuro se aplicara al modelado de riesgos médicos, diagnóstico probabilístico o evaluación de escenarios sanitarios. También podría vincularse con el ODS 7: Energía asequible y no contaminante o el ODS 13: Acción por el clima si se utilizara para analizar riesgos energéticos, eventos climáticos extremos o impactos ambientales. De forma similar, podría tener aplicación en el ODS 11: Ciudades y comunidades sostenibles si se empleara para modelar riesgos urbanos, infraestructuras críticas o planes de emergencia. No obstante, en la versión actual del TFG estas aplicaciones son potenciales y no forman parte del alcance implementado, por lo que su relación debe considerarse baja.

También se puede establecer una relación baja con el ODS 12: Producción y consumo responsables, ya que una mejor evaluación del riesgo puede contribuir indirectamente a decisiones más responsables en el uso de recursos. Sin embargo, la aplicación no aborda directamente procesos productivos ni patrones de consumo. Del mismo modo, el ODS 16: Paz, justicia e instituciones sólidas podría rela-

cionarse de forma indirecta con la transparencia en la toma de decisiones, ya que la herramienta permite visualizar relaciones y resultados en lugar de ocultarlos en modelos opacos. Aun así, no se trata de un objetivo central del proyecto.

En conclusión, este TFG se relaciona principalmente con el ODS 9, por su carácter tecnológico e innovador, y de forma relevante con los ODS 4, 8 y 17. La contribución del proyecto no se basa en resolver directamente un problema social concreto, sino en ofrecer una herramienta que puede mejorar la comprensión de sistemas complejos y apoyar decisiones más seguras, trazables e informadas. Su mayor valor en relación con los ODS está en su potencial de aplicación transversal: una vez desarrollada la base tecnológica, la herramienta podría adaptarse a sectores donde el análisis de riesgos tenga impacto social, económico o ambiental.