
This is the **published version** of the bachelor thesis:

Bernáldez Racamonde, David. *Plataforma web para la creación y validación de demostraciones modulares en Lean*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334415>

under the terms of the  license.

This is the **published version** of the bachelor thesis:

Bernáldez Racamonde, David. *Plataforma web para la creación y validación de demostraciones modulares en Lean*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334415>

under the terms of the  license.

Plataforma web para la creación y validación de demostraciones modulares en Lean

David Bernaldez Racamonde

Resumen — Este trabajo presenta el diseño y desarrollo de una plataforma web orientada a la construcción de demostraciones de teoremas matemáticos mediante Lean. El software desarrollado permite la creación de teoremas, la división de la demostración matemática en bloques, la definición de dependencias entre ellos mediante grafos acíclicos dirigidos (DAG) y la validación formal de dichos bloques. La solución implementada integra un frontend web, un backend con gestión de usuarios y persistencia de datos, y un servicio de validación Lean encargado de comprobar el contenido formal de cada bloque. Asimismo, la plataforma incorpora una representación gráfica de las dependencias, con el objetivo de ofrecer una visión más clara de la estructura global de la demostración. Como resultado, se ha obtenido un prototipo funcional en entorno local que demuestra la viabilidad de la propuesta planteada.

Palabras clave — Demostración formal, grafos DAG, Lean, plataforma web, pruebas modulares, teoremas matemáticos, validación de bloques.

Abstract — This work presents the design and development of a web platform aimed at the construction of mathematical theorem proofs using Lean. The developed software allows the creation of theorems, the division of the mathematical proof into blocks, the definition of dependencies between them through directed acyclic graphs (DAGs), and the formal validation of those blocks. The implemented solution integrates a web frontend, a backend with user management and data persistence, and a Lean validation service responsible for checking the formal content of each block. In addition, the platform incorporates a graphical representation of dependencies, with the aim of providing a clearer view of the overall structure of the proof. As a result, a functional prototype has been obtained in a local environment, demonstrating the viability of the proposed approach.

Index Terms — Block validation, DAG graphs, formal proof, Lean, mathematical theorems, modular proofs, web platform.

1 INTRODUCCIÓN - CONTEXTO DEL TRABAJO

Las herramientas de demostración formal permiten expresar definiciones, lemas y teoremas con un nivel de rigor que garantiza que cada paso sea verificado por una máquina. Entre estas herramientas, Lean se ha consolidado como una de las más conocidas, tanto por su base teórica como por la disponibilidad de bibliotecas matemáticas amplias y su adopción en investigación.

A pesar de su alcance, el trabajo con Lean se basa habitualmente en la escritura lineal de la demostración, lo que obliga al usuario a gestionar en un mismo flujo la declaración del resultado y la secuencia completa de pasos necesarios para demostrarlo. Este enfoque resulta adecuado en pruebas sencillas, pero presenta dificultades a medida que la demostración crece en tamaño o depende de varias subpruebas relacionadas. Tareas como comprender la estructura global de la prueba o localizar dependencias entre resultados parciales pueden volverse complejas.

Desde una perspectiva de ingeniería del software, esto sugiere aplicar principios de modularidad y separación de responsabilidades también al ámbito de la demostración formal, del mismo modo que un sistema complejo se descompone en módulos con relaciones explícitas, una demostración puede beneficiarse de una organización en bloques con dependencias bien definidas, lo que además permite que la descomposición inicial de un teorema y la resolución de cada bloque puedan llevarse a cabo por personas distintas.

Con esto en mente, este trabajo propone una plataforma web para la construcción modular de demostraciones mediante Lean. Cada demostración se organiza como un conjunto de bloques relacionados entre sí, de forma que la estructura global queda representada como un grafo acíclico dirigido (DAG). Esta representación permite reflejar explícitamente la relación entre los bloques y ofrece una visión más clara de la demostración. La solución integra una interfaz web para crear teoremas y editar bloques, un backend para la gestión de usuarios y persistencia, y un servicio de validación conectado con Lean que comprueba el contenido de cada bloque.

El objetivo no es únicamente disponer de una aplicación funcional, sino estudiar la viabilidad de una forma

-
- E-mail de contacte: 1633445@uab.cat
 - Menció realitzada: Enginyeria del Software
 - Treball tutoritzat per: Antonio Lozano Bagen (Computació)
 - Curs 2025/26

alternativa de interactuar con demostraciones formales, más visual, estructurada y cercana a los principios propios de la ingeniería del software.

2 ESTADO DEL ARTE

2.1 Asistentes de demostración formal y ecosistema actual

La demostración formal se apoya en asistentes de demostración o *proof assistants*, es decir, sistemas capaces de verificar mecánicamente definiciones, teoremas y pruebas. Entre las herramientas más representativas actualmente se encuentran Lean, Rocq e Isabelle. Lean se presenta oficialmente como un lenguaje de programación y asistente de demostración de código abierto orientado a construir código correcto y verificado. Rocq es un popular asistente de pruebas interactivo y de código abierto que proporciona un lenguaje formal para escribir definiciones matemáticas, algoritmos ejecutables y teoremas, permitiendo el desarrollo de pruebas matemáticas verificadas por computadora. Por último, Isabelle dispone de un ecosistema consolidado y se apoya, entre otros recursos, en el Archive of Formal Proofs, un repositorio de bibliotecas, ejemplos y desarrollos científicos verificados mecánicamente.

Lean resulta especialmente relevante para este trabajo porque combina su uso como asistente de demostración y su integración con un ecosistema de desarrollo relativamente moderno. La comunidad de Lean mantiene Mathlib, una biblioteca matemática de gran tamaño y de desarrollo comunitario, que reúne resultados formalizados y definiciones reutilizables tanto para matemáticas como para programación. Además, la documentación oficial de Lean sitúa a Lake como la herramienta estándar de construcción y gestión de dependencias del ecosistema, responsable de compilar código Lean, obtener dependencias externas y dar soporte al trabajo por paquetes y bibliotecas.

2.2 Herramientas de trabajo en Lean e interacción con las pruebas

El flujo de trabajo habitual en Lean sigue estando centrado en la edición textual. La instalación recomendada por el manual oficial incluye la extensión oficial para Visual Studio Code. El manual de la extensión describe un conjunto amplio de funciones para trabajar con archivos Lean desde el editor, lo que refuerza un modelo de uso basado en la escritura secuencial del contenido formal.

A este entorno de escritorio se añaden herramientas de acceso web como Lean 4 Web, presentado oficialmente como un editor interactivo en navegador para probar código Lean. Este tipo de soluciones reduce la barrera de entrada, facilita la experimentación y permite validar expresiones o pequeñas pruebas sin necesidad de una instalación completa local. Sin embargo, incluso en estos casos, la interacción sigue girando en torno a la edición

textual del *script* y a la comprobación de su resultado dentro del propio entorno.

En conjunto, tanto el entorno de escritorio como las herramientas web disponibles comparten un modelo de interacción centrado en el archivo o el *script* como unidad principal de trabajo. Ninguna de estas herramientas está orientada a gestionar una demostración como una estructura de partes relacionadas, con dependencias explícitas y validación independiente por fragmentos.

2.3 Limitaciones detectadas y oportunidad del proyecto

A partir de la investigación realizada, se ha observado que el ecosistema actual de Lean ofrece herramientas para escribir, compilar, explorar y verificar contenido formal. Existen bibliotecas extensas, herramientas estándar de construcción y entornos de edición bien integrados, tanto en escritorio como en navegador. No obstante, las herramientas encontradas están principalmente orientadas a trabajar con pruebas en formato textual, organizadas por archivos más que a gestionar una demostración como una estructura visual compuesta por partes relacionadas.

No se ha identificado una herramienta cuyo objetivo principal sea descomponer una demostración en bloques independientes, donde sus dependencias puedan visualizarse claramente. Ante este escenario surge la oportunidad de este TFG: construir una capa software adicional que permita crear teoremas, dividir sus demostraciones en bloques, establecer relaciones explícitas entre ellos y validar formalmente cada parte de manera modular. Este proyecto se sitúa, por tanto, como una extensión sobre el ecosistema existente, no como un sustituto de Lean ni de sus herramientas oficiales.

3 OBJETIVOS

3.1 Objetivo general

El objetivo principal de este trabajo es diseñar y desarrollar una plataforma web que permita construir y validar demostraciones de teoremas matemáticos mediante Lean de forma modular, facilitando la organización de la prueba en bloques independientes, la definición explícita de dependencias entre dichos bloques y su validación formal dentro de un entorno software unificado.

3.2 Objetivos específicos

Para alcanzar el objetivo general planteado, el trabajo se centra en los siguientes objetivos específicos:

- Definir un modelo de demostración basado en bloques verificables de forma independiente.
- Desarrollar una aplicación web para la creación, edición y gestión de teoremas y bloques.

- Implementar un sistema básico de usuarios y roles con control de acceso diferenciado.
- Integrar un servicio de validación Lean, capaz de comprobar el contenido de cada bloque y devolver un estado asociado a su validez.
- Incorporar una visualización gráfica de las dependencias entre bloques.
- Evaluar el correcto funcionamiento del prototipo, comprobando si la solución implementada resulta técnicamente viable y si cumple los requisitos funcionales definidos para la gestión, estructuración y validación de demostraciones.

4 METODOLOGÍA Y PROCESO DE DESARROLLO

4.1 Enfoque de desarrollo

Para el desarrollo del proyecto se ha seguido un enfoque incremental e iterativo. Esta decisión se tomó porque la plataforma se divide en distintos componentes, como la interfaz web, el backend, la base de datos y el servicio de validación Lean, cuya funcionalidad podía comprobarse progresivamente y por separado. Este tipo de enfoque resulta especialmente adecuado en proyectos con múltiples componentes interdependientes, ya que permite validar partes del trabajo a medida que se desarrollan, detectar errores en fases tempranas y corregir decisiones antes de llegar a una fase final.

El trabajo no se ha planteado como una implementación cerrada desde el inicio, sino como una evolución continua a partir de un conjunto de objetivos definidos previamente. En una primera etapa se analizó el problema de la construcción lineal de demostraciones en Lean y se identificaron los elementos funcionales mínimos que debía cubrir la aplicación. A continuación, se diseñó una arquitectura inicial capaz de dar soporte a la gestión de teoremas, bloques y dependencias. Sobre esta base, el sistema se fue ampliando por iteraciones, incorporando nuevas capacidades y mejorando las ya existentes.

Este enfoque permitió dividir el desarrollo en pequeños objetivos manejables. En primer lugar, se implementó la estructura básica de datos y la persistencia. Posteriormente, se desarrollaron la autenticación de usuarios, la gestión de teoremas y la creación de bloques. En iteraciones posteriores se incorporó la definición de dependencias entre bloques, su representación visual como grafo y la validación formal mediante Lean. Por último, se realizaron ajustes de integración y pruebas funcionales en el entorno local. De este modo, cada fase producía un resultado verificable y servía de base para la siguiente.

Desde el punto de vista metodológico, esta estrategia ha resultado adecuada porque el proyecto exigía combinar aspectos de análisis, diseño, implementación e integración en un número limitado de horas. Además, el desarrollo incremental ha facilitado adaptar ciertas decisiones técnicas a medida que aparecían problemas reales de integración, especialmente en la comunicación entre el

backend y el servicio de validación Lean.

4.2 Planificación del trabajo

La planificación del trabajo se estructuró en varias fases consecutivas, aunque parcialmente solapadas, siguiendo la lógica del desarrollo incremental adoptado. En la fase inicial se realizó la definición detallada del problema, la concreción de los objetivos y la revisión de herramientas. Esta etapa permitió delimitar el alcance real del proyecto y establecer una solución viable dentro del tiempo disponible para su desarrollo.

A continuación, se abordó una fase de diseño, centrada en la definición de la arquitectura general del sistema y en el modelado de las entidades principales de la aplicación. En esta etapa se decidió separar claramente la solución en varios componentes: frontend, backend, base de datos y servicio de validación Lean. También se definió la representación de las dependencias entre bloques como un grafo acíclico dirigido (DAG), al considerarse esta opción la más adecuada para expresar relaciones entre bloques sin introducir ciclos lógicos.

La fase de implementación se desarrolló de forma progresiva. En primer lugar, se construyó la base del backend y la persistencia de datos. Después, se implementó la interfaz web y la gestión de usuarios. Una vez establecida la funcionalidad básica, se añadieron las operaciones específicas sobre teoremas y bloques, así como la edición de dependencias y la visualización gráfica de estas. Finalmente, se integró el servicio de validación Lean, que constituye el componente más específico del proyecto.

La fase final se dedicó a pruebas funcionales, corrección de errores y validación del comportamiento global de la plataforma. Durante esta etapa también se intentó el despliegue en un entorno distinto al local, con el objetivo de comprobar la portabilidad de la solución. Sin embargo, las dificultades encontradas en la ejecución del validador Lean fuera del entorno de desarrollo habitual llevaron a priorizar un prototipo estable y funcional en local. Esta decisión no cambia el objetivo principal del proyecto, que consiste en demostrar la viabilidad técnica de la propuesta, sino que delimita con mayor precisión su alcance final.

4.3 Herramientas utilizadas

El desarrollo del proyecto ha requerido el uso combinado de varias herramientas software, seleccionadas en función de su adecuación técnica, su compatibilidad entre sí y su utilidad dentro de un flujo de trabajo centrado en aplicaciones web con validación formal.

El frontend se implementó con React y TypeScript sobre Vite, utilizando una organización basada en componentes y servicios. Esta parte del sistema se diseñó para gestionar la interacción con teoremas, bloques y dependencias, así como para representar visualmente la estructura del grafo

e interactuar con la API del backend.

El backend se desarrolló con FastAPI y actúa como API central del sistema. Sus responsabilidades principales son la autenticación y gestión de usuarios, la persistencia de teoremas y bloques, el mantenimiento de las dependencias entre ellos y la comunicación con el servicio de validación Lean. Para la persistencia se utilizó PostgreSQL, ejecutado en entorno local mediante Docker, al tratarse de una solución adecuada para representar entidades bien definidas y relaciones estructuradas.

La validación formal se separó en un servicio independiente conectado con Lean, diseñado para desacoplar la lógica principal de la aplicación de la ejecución del asistente de demostración. Para la gestión del proyecto Lean y de sus dependencias se empleó Lake, herramienta estándar del ecosistema. Esta separación permitió mantener una arquitectura modular y aislar mejor los problemas técnicos asociados a la validación formal.

4.4 Criterios de validación del proyecto

La validación del proyecto se definió en términos funcionales y técnicos. Entre los criterios funcionales considerados se incluyeron el registro e inicio de sesión de usuarios, la creación de teoremas, la creación y edición de bloques, la definición de dependencias, la persistencia de la información y la visualización estructurada del grafo. A ello se añadió la validación formal, entendida como la capacidad de enviar un bloque al servicio Lean, recibir una respuesta coherente y reflejar el estado correspondiente dentro de la aplicación.

En segundo lugar, se consideraron criterios de integración. Dado que la plataforma depende de la interacción coordinada de varios subsistemas, no bastaba con comprobar cada componente de manera aislada. Era necesario verificar que la comunicación entre frontend, backend, base de datos y servicio de validación se producía correctamente, que los datos persistidos podían recuperarse sin inconsistencias y que los cambios en la interfaz tenían un reflejo correcto en la lógica del software y en el almacenamiento.

Por último, se introdujo un criterio de viabilidad del prototipo. Este criterio no exige que la solución alcance el nivel de un producto final completo, pero sí que demuestre que el enfoque planteado es técnicamente realizable y útil para el problema definido. En este sentido, el proyecto puede considerarse validado si permite gestionar demostraciones en bloques, representar sus dependencias de forma explícita y comprobar formalmente esos bloques dentro de un flujo de trabajo integrado. Sobre esta base, las limitaciones detectadas no invalidan la propuesta, sino que delimitan las condiciones en las que el prototipo funciona correctamente y orientan las líneas de mejora futura.

5 DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

5.1 Arquitectura general del sistema

La solución desarrollada se ha diseñado siguiendo una arquitectura modular compuesta por cuatro elementos principales, tal como se muestra en la Fig. 1: una interfaz web para la interacción con el usuario, un backend encargado de la lógica de la app, una base de datos para la persistencia de la información y un servicio de validación formal conectado con Lean. Esta organización responde a la necesidad de separar responsabilidades, facilitar la evolución de cada componente y mantener una estructura coherente con el planteamiento general del proyecto.

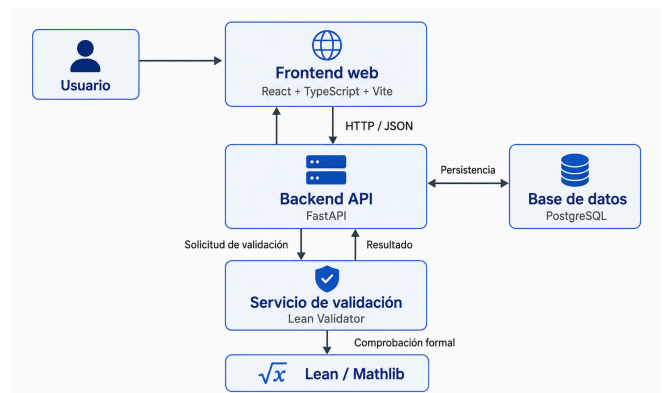


Fig. 1. Arquitectura general del sistema

El frontend actúa como punto de entrada para el usuario y concentra las funcionalidades relacionadas con la gestión de teoremas, la edición de bloques, la definición de dependencias y la visualización de la estructura de la demostración. Su función no es realizar validación formal ni aplicar lógica compleja, sino ofrecer una capa de interacción clara sobre la información almacenada y sobre los estados devueltos por el backend.

El backend centraliza la lógica principal del sistema. Se encarga de autenticar usuarios, aplicar los permisos asociados a cada rol, gestionar la creación y edición de teoremas y bloques, almacenar sus relaciones y coordinar la comunicación con el servicio de validación Lean. De este modo, el backend actúa como intermediario entre la interfaz y el componente especializado de validación, garantizando que la aplicación mantenga una estructura limpia y una comunicación bien definida entre subsistemas.

La persistencia del sistema se resuelve mediante una base de datos relacional, que almacena usuarios, teoremas, bloques y sus dependencias. El modelo relacional resulta adecuado ya que las entidades tienen relaciones explícitas y bien definidas, lo que permite mantener la coherencia estructural de los datos y consultarlos de forma eficiente.

Por último, el servicio de validación Lean constituye el componente más importante del proyecto. Su propósito es recibir el contenido formal de un bloque, comprobarlo

mediante Lean y devolver el estado correspondiente de validación. Esta separación entre aplicación principal y validador formal evita acoplar directamente la lógica web con la ejecución del asistente de demostración y refuerza el carácter modular de la arquitectura.

5.2 Modelo de datos

El modelo de datos de la aplicación se ha definido a partir de las entidades necesarias para representar de manera estructurada el dominio del problema, como se resume en la Fig. 2. Las entidades principales son los usuarios, los teoremas y los bloques de la demostración. A ellas se añaden los atributos y relaciones necesarias para gestionar permisos, persistencia y dependencias entre partes de una misma prueba.

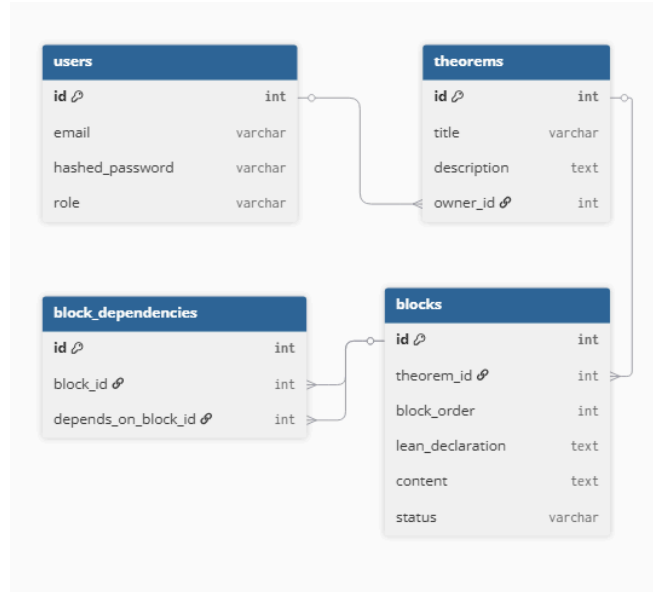


Fig. 2. Modelo de datos de la aplicación

La entidad usuario permite identificar a cada persona que accede al sistema y asociarle unas credenciales y un rol determinado. El modelo incorpora al menos una distinción entre usuarios con capacidad de edición y usuarios con capacidad exclusiva de visualización, en línea con uno de los objetivos específicos del proyecto.

La entidad teorema actúa como contenedor lógico de una demostración. Cada teorema dispone de una identidad propia y de la información necesaria para describirlo dentro de la plataforma. Sobre esta entidad se apoyan las operaciones de creación, consulta y edición que forman parte del flujo habitual de trabajo del sistema.

La entidad bloque representa la unidad básica de la demostración. Cada bloque está asociado a un teorema concreto y contiene la información necesaria para expresar una parte de la prueba. Además, cada bloque incorpora un estado de validación que refleja el resultado devuelto por Lean. Esta decisión permite que la demostración no se trate como un texto único, sino como un conjunto de fragmentos verificables por separado.

Las dependencias entre bloques constituyen uno de los aspectos más importantes del modelo. Estas relaciones permiten expresar que un bloque depende del contenido o del resultado de otros bloques previos, lo que hace posible representar la estructura global de la demostración como un grafo. A nivel de datos, esto implica almacenar no sólo la existencia de los bloques, sino también las relaciones entre ellos, que posteriormente son utilizadas tanto por la lógica del backend como por la visualización del frontend.

5.3 Diseño funcional de la aplicación

Desde el punto de vista funcional, la aplicación se ha diseñado para cubrir un flujo de trabajo simple pero suficiente para validar la propuesta del proyecto. El usuario accede al sistema mediante autenticación, consulta los teoremas existentes o crea uno nuevo y, a partir de ahí, puede construir la demostración asociada mediante bloques. Cada bloque puede editarse de forma independiente y puede relacionarse con otros bloques a través de dependencias explícitas. Finalmente, el usuario puede solicitar la validación formal del contenido.

La funcionalidad principal de la plataforma gira en torno a cuatro acciones básicas: gestionar teoremas, gestionar bloques, definir relaciones entre bloques y validar el contenido de un bloque. A partir de estas operaciones se desarrolla la mayor parte del comportamiento del sistema.

En términos de experiencia de uso, el sistema se ha planteado de manera que la estructura de la demostración sea visible y editable. La aplicación no se limita a almacenar texto, sino que permite trabajar sobre unidades diferenciadas de razonamiento y observar sus relaciones. De este modo, la interfaz deja de estar orientada exclusivamente al contenido secuencial y pasa a reflejar la estructura interna de la prueba.

El diseño funcional también contempla la existencia de distintos niveles de acceso. Esto implica que no todos los usuarios interactúan de la misma forma con el sistema, algunos pueden crear y modificar información, mientras que otros quedan limitados a su consulta. Esta decisión aporta coherencia con el modelo de datos y con la lógica general del backend.

5.4 Implementación del frontend

La implementación del frontend se ha orientado a ofrecer una interfaz clara para trabajar con teoremas, bloques y dependencias. En lugar de centrar toda la interacción en la edición textual tradicional, la aplicación traslada al navegador las operaciones principales del sistema y organiza la información de manera que el usuario pueda identificar con facilidad tanto el contenido de cada bloque como su posición dentro de la demostración.

Una parte esencial del frontend es la pantalla de gestión de teoremas, donde el usuario puede consultar los

teoremas disponibles y crear nuevos. A partir de esta vista general, la aplicación permite acceder al detalle de cada teorema, que constituye el espacio principal de trabajo sobre la demostración. En esta pantalla se concentran la creación y edición de bloques, la asignación de dependencias y la visualización del estado de validación asociado a cada uno de ellos.

La representación visual de las dependencias entre bloques se ha implementado como uno de los elementos diferenciadores del sistema. Para ello, el frontend construye una visualización gráfica a partir de la información recibida del backend y la utiliza para mostrar el grafo de relaciones entre bloques. Esta vista no solo tiene una función estética, sino también estructural, ya que permite comprender de manera inmediata qué bloques dependen de otros y cómo se organiza el razonamiento global. Para esta visualización se utilizaron React Flow, como librería de representación de nodos y conexiones en la interfaz, y Dagre, para calcular la disposición automática de los nodos y mejorar la claridad del grafo generado. La Fig. 3 muestra un ejemplo de esta representación gráfica de dependencias entre bloques, al final del documento el Apéndice A1 recoge capturas adicionales de estas pantallas, tanto en modo editor como en modo visualizador.

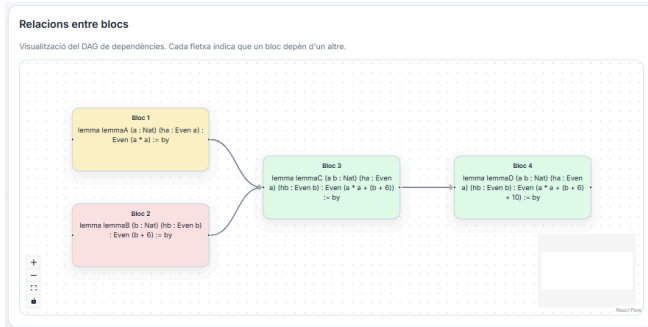


Fig. 3. Grafo acíclico dirigido (DAG) de un teorema

5.5 Implementación del backend

El backend se ha implementado como una API encargada de dar soporte a toda la lógica del sistema mediante los endpoints implementados (ver Tabla 1). Este componente es responsable de recibir las peticiones del frontend, aplicar las comprobaciones necesarias, acceder a la base de datos, coordinar la validación con Lean y devolver una respuesta al cliente.

Una de sus funciones principales es la gestión y autenticación de usuarios. Para ello, el backend incorpora los mecanismos necesarios para registrar usuarios, iniciar sesión, recuperar la identidad del usuario autenticado y aplicar restricciones en función del rol asignado. En relación con la seguridad de la API, el sistema utiliza JWT (JSON Web Tokens) como mecanismo de autenticación para el acceso a los endpoints protegidos. Asimismo, las contraseñas de los usuarios se almacenan en la base de datos en forma de *hash*, evitando su conservación en texto plano. Esta parte resulta fundamental para mantener la coherencia del comportamiento multiusuario planteado en los objetivos del trabajo.

Además de la autenticación, el backend gestiona la creación, consulta, edición y eliminación de teoremas y bloques. También se encarga de almacenar y recuperar las dependencias entre bloques, de forma que el frontend pueda representar correctamente la estructura del grafo.

Otra responsabilidad importante del backend es la gestión de estados de validación. Cuando un bloque se envía a validar, el backend prepara la información requerida, la remite al servicio Lean y actualiza el estado correspondiente a partir de la respuesta recibida. De esta forma, la aplicación mantiene una traza coherente entre el contenido introducido por el usuario, la comprobación formal realizada y el resultado mostrado posteriormente en la interfaz.

En conjunto, el backend centraliza la lógica de negocio del sistema, coordina la persistencia de la información y actúa como intermediario entre la interfaz web y el servicio de validación Lean. Esta organización facilita tanto la coherencia del comportamiento global de la aplicación como su evolución futura.

Tabla 1. Endpoints de la API

Endpoint	Método	Descripción
/auth/register	POST	Registra un nuevo usuario en el sistema
/auth/login	POST	Inicia sesión y devuelve el token de autenticación
/auth/me	GET	Devuelve la información del usuario autenticado
/theorems	GET	Obtiene la lista de teoremas disponibles
/theorems	POST	Crea un nuevo teorema
/theorems/{theorem_id}	GET	Obtiene el detalle de un teorema concreto
/theorems/{theorem_id}	PUT	Actualiza la información de un teorema
/theorems/{theorem_id}	DELETE	Elimina un teorema existente
/blocks	POST	Crea un nuevo bloque asociado a un teorema
/blocks/{block_id}	GET	Obtiene el detalle de un bloque concreto
/blocks/{block_id}	PUT	Actualiza la información de un bloque
/blocks/{block_id}	DELETE	Elimina un bloque existente
/blocks/{block_id}/validate	POST	Envía un bloque al servicio de validación Lean y actualiza su estado

5.6 Integración con Lean

La integración con Lean constituye el componente más específico del proyecto, ya que introduce la dimensión de demostración formal dentro de una arquitectura web convencional. Su finalidad es comprobar si el contenido

asociado a cada bloque es aceptado por Lean y trasladar ese resultado al estado correspondiente dentro de la plataforma.

Para ello, se ha diseñado un flujo en el que el backend envía al servicio de validación la información necesaria para construir el contexto de comprobación de un bloque. A partir de esta información, el servicio genera un fichero temporal de Lean, como el mostrado en la Fig. 4, que incluye las importaciones necesarias, las declaraciones auxiliares derivadas de los bloques dependientes y, finalmente, la declaración y la prueba del bloque que se desea validar. Una vez generado, este fichero se ejecuta mediante Lean y el servicio devuelve una respuesta con el estado resultante. La plataforma interpreta dicha respuesta y la utiliza para marcar el bloque como válido, pendiente o erróneo, según corresponda. En la implementación actual, esta validación se realiza de forma independiente para cada bloque, por lo que el estado mostrado en la interfaz refleja el resultado individual de esa comprobación y no una propagación automática del estado a través de todas las dependencias del grafo.

```

backend > lean_validator > LeanValidator > ValidationTmp > ▾ GeneratedBlock.lean
1  import Mathlib
2
3  axiom lemmaA : Even 2
4
5  axiom lemmaB : Even 4
6
7  lemma lemmaC : Even 6 := by
8    have hA : Even 2 := lemmaA
9    have hB : Even 4 := lemmaB
10   simp using Even.add hA hB
11

```

Fig. 4. Ejemplo de fichero temporal generado para la validación de un bloque en Lean

Cabe destacar que las declaraciones auxiliares derivadas de los bloques dependientes se incorporan como axiomas (axiom) en lugar de como demostraciones completas, lo que permite validar cada bloque de forma aislada asumiendo como ciertas sus dependencias, con independencia de que estas estén incompletas, pendientes de revisión o erróneas en ese momento. Esta decisión resulta clave para el carácter modular y colaborativo de la plataforma, ya que permite repartir el trabajo de una misma demostración entre varias personas trabajando en paralelo sobre bloques distintos, sin que cada una deba esperar a que las demás completen o corrijan los bloques de los que depende. El diseño de esta integración también ha buscado evitar un acoplamiento excesivo entre la lógica principal de la aplicación y la ejecución de Lean, situando la validación formal en un servicio separado que permite aislar mejor los problemas técnicos y mantener separadas las responsabilidades de cada parte del sistema.

6 RESULTADO PRUEBAS Y DISCUSIÓN

6.1 Funcionalidades implementadas

El resultado principal del trabajo es un prototipo funcional en entorno local que materializa la propuesta de construcción modular de demostraciones mediante Lean. La aplicación desarrollada permite gestionar usuarios, autenticar sesiones y aplicar una distinción básica de roles entre usuarios con permisos de edición y usuarios con permisos de visualización. Sobre esta base, el sistema ofrece las operaciones necesarias para crear, consultar y mantener teoremas dentro de la plataforma.

En relación con la construcción de las pruebas, la funcionalidad central implementada consiste en la creación y edición de bloques de demostración asociados a un teorema. Cada bloque actúa como una unidad independiente de trabajo formal y puede almacenar tanto su contenido como la información necesaria para situarlo dentro de la estructura global de la demostración. Además, la aplicación permite establecer dependencias explícitas entre bloques, de forma que la prueba no se trate como un único texto lineal, sino como un conjunto de elementos relacionados entre sí.

Otro de los aspectos implementados es la visualización de estas dependencias mediante una representación gráfica basada en grafos acíclicos dirigidos (DAG). Esta funcionalidad permite observar de forma más clara la estructura interna de la demostración, identificar relaciones entre subpruebas y comprender el flujo lógico del razonamiento formal. Junto a ello, la plataforma incorpora un mecanismo de validación conectado con Lean, encargado de comprobar el contenido de los bloques y de devolver un estado asociado a su resultado. En conjunto, estas funcionalidades permiten afirmar que el prototipo cubre el núcleo operativo previsto en los objetivos del trabajo.

6.2 Pruebas realizadas

La validación del sistema se ha realizado principalmente mediante pruebas funcionales en entorno local, orientadas a comprobar el comportamiento de las operaciones más relevantes de la aplicación. En primer lugar, se verificó el flujo de autenticación, incluyendo el registro de usuarios, el inicio de sesión y la restricción de acceso a determinadas funciones según el rol asignado. Estas pruebas permitieron comprobar que la aplicación mantenía un comportamiento coherente en escenarios básicos de uso multiusuario.

En segundo lugar, se realizaron pruebas sobre la gestión de teoremas y bloques. Se comprobó la correcta creación de nuevos teoremas, su persistencia en base de datos y su recuperación posterior desde la interfaz. Del mismo modo, se verificó la creación, edición y almacenamiento de bloques, así como su asociación correcta al teorema correspondiente. También se probaron las operaciones relacionadas con las dependencias entre bloques, asegurando que estas se guardaban de forma consistente

y que podían representarse posteriormente en la visualización gráfica del sistema.

Una tercera línea de prueba se centró en la validación formal. Para ello se utilizaron ejemplos sencillos y controlados, con el objetivo de verificar que el backend enviaba correctamente la información al servicio Lean, que este devolvía una respuesta interpretable y que el estado del bloque se actualizaba de forma coherente dentro de la aplicación. Estas pruebas permitieron comprobar el funcionamiento del flujo completo de validación en entorno local, incluyendo la comunicación entre frontend, backend y servicio de validación.

Además de estas pruebas funcionales, se realizaron pruebas de integración entre componentes. En particular, se verificó que los cambios introducidos desde la interfaz se reflejaban correctamente en la base de datos, que el backend recuperaba la información sin inconsistencias y que la representación gráfica de dependencias correspondía a las relaciones realmente definidas entre los bloques.

6.3 Resultados obtenidos

Los resultados obtenidos permiten afirmar que la propuesta desarrollada es técnicamente viable dentro del alcance planteado para el proyecto. En primer lugar, se ha conseguido implementar una aplicación web capaz de gestionar teoremas y sus demostraciones de forma modular. Esto supone ya una diferencia significativa respecto al flujo lineal tradicional, ya que la prueba puede organizarse en partes diferenciadas con relaciones entre ellas.

En segundo lugar, se ha logrado integrar una representación visual de las dependencias entre bloques. Esta parte del sistema resulta especialmente relevante porque traduce a una estructura gráfica una organización que, en un flujo puramente textual, suele permanecer implícita. El uso de un grafo acíclico dirigido (DAG) permite reflejar mejor la estructura lógica de la demostración y aporta una visión más clara del papel de cada subprueba dentro del conjunto.

En tercer lugar, el sistema ha demostrado que es posible integrar la validación formal de bloques dentro de una arquitectura web separada en componentes. En el entorno local, la comunicación entre la aplicación y el servicio Lean ha permitido validar contenido formal y reflejar sus resultados en la interfaz. Este hecho confirma que la propuesta modular no sólo es conceptualmente coherente, sino también implementable en un prototipo real.

De forma global, el resultado del trabajo no es simplemente una prueba de concepto, sino un prototipo operativo que permite experimentar con una forma alternativa de construir demostraciones formales. Aunque el sistema no pretende sustituir las herramientas oficiales de Lean ni cubrir todos los escenarios posibles de uso, sí ofrece una base sólida para estudiar nuevas formas de

organización y validación de pruebas dentro de una aplicación propia.

6.4 Limitaciones encontradas

A lo largo del desarrollo se identificaron varias limitaciones relacionadas con el alcance del prototipo y con el nivel de madurez alcanzado por la solución. En primer lugar, la plataforma se ha diseñado como un sistema funcional orientado a validar la viabilidad de una aproximación modular a la demostración formal, no como un entorno completo que cubra todos los escenarios posibles de uso de Lean. Por ello, ciertas capacidades avanzadas propias de entornos más maduros han quedado fuera del alcance del proyecto.

Una de las limitaciones del proyecto es que la validación formal se ha planteado dentro del marco de Lean y Mathlib, que constituye actualmente la biblioteca de referencia más completa y utilizada en este asistente de demostración. Por ello, el trabajo no se ha orientado a cubrir entornos ajenos a dicha biblioteca, sino a demostrar la viabilidad de la propuesta en el contexto más representativo y habitual de uso de Lean.

Otra limitación importante es que la colaboración implementada no debe entenderse como edición simultánea en tiempo real. Aunque la aplicación incorpora gestión de usuarios y control de acceso mediante roles diferenciados, no dispone de mecanismos avanzados de concurrencia, sincronización instantánea o resolución de conflictos entre modificaciones paralelas. Sin embargo, este aspecto tiene un impacto reducido en el contexto de la demostración formal, ya que el contenido de un bloque Lean tiene una validez binaria, es correcto o no lo es, por lo que no existen versiones parcialmente correctas que deban fusionarse o rectificarse.

También deben señalarse algunas limitaciones relacionadas con la expresividad y evolución de la propia plataforma. La visualización del grafo de dependencias cumple adecuadamente su función estructural, pero podría mejorarse con mecanismos más avanzados de navegación, filtrado o agrupación de bloques. Del mismo modo, la trazabilidad del proceso de validación podría ampliarse con información más detallada sobre el contexto formal, los errores devueltos por Lean o la evolución histórica de cada bloque.

Finalmente, se realizaron pruebas de despliegue mediante Docker en un servidor con el objetivo de comprobar la portabilidad del sistema. Aunque la integración general de la solución era viable, el componente de validación Lean presentó problemas de compatibilidad con las versiones disponibles en dicho entorno. En particular, durante la importación de determinados ficheros necesarios para la validación, el proceso quedaba bloqueado, lo que impedía completar de forma estable ese escenario de ejecución.

En conjunto, estas limitaciones no invalidan el resultado

del trabajo, sino que delimitan con precisión el alcance actual del prototipo. La plataforma obtenida permite demostrar la viabilidad técnica de la propuesta en entorno local y establece una base suficientemente sólida para futuras ampliaciones orientadas a mejorar la validación, la colaboración y la portabilidad del sistema.

6.5 Discusión de los resultados

Los resultados obtenidos deben interpretarse en relación con los objetivos planteados al inicio del trabajo. Desde esta perspectiva, puede considerarse que el proyecto ha cumplido satisfactoriamente su finalidad principal: demostrar que es posible construir una plataforma web capaz de organizar demostraciones formales en bloques, representar sus dependencias de manera explícita y conectar dicha estructura con un proceso de validación basado en Lean.

Uno de los aspectos más relevantes del trabajo es que la aportación no se limita a integrar Lean dentro de una aplicación web, sino que introduce una forma distinta de estructurar la prueba. Esta organización modular y visual constituye el principal valor diferencial del prototipo. El sistema no elimina la necesidad de escribir contenido formal correcto, pero sí ofrece una capa adicional de organización que puede mejorar la comprensión de la estructura de una demostración y facilitar su gestión progresiva.

Desde un punto de vista de la ingeniería del software, el proyecto también muestra la utilidad de aplicar principios de separación de responsabilidades al ámbito de la demostración formal. La arquitectura en componentes, el desacoplamiento del servicio de validación y la persistencia estructurada de teoremas, bloques y dependencias refuerzan la idea de que el caso abordado puede tratarse como un problema software bien definido y no únicamente como una cuestión de formalización matemática.

En conjunto, los resultados expuestos permiten concluir que se ha alcanzado un nivel de desarrollo adecuado para el alcance del proyecto: existe una arquitectura funcional, una implementación operativa en local y una demostración razonable de que la aproximación modular propuesta puede constituir una vía útil para organizar, visualizar y validar demostraciones formales mediante Lean.

7 CONCLUSIONES Y TRABAJO FUTURO

7.1 Conclusiones

El presente trabajo ha permitido diseñar y desarrollar una plataforma web orientada a la construcción y validación modular de demostraciones formales mediante Lean. El proyecto parte de una limitación práctica observada en el flujo habitual de trabajo con asistentes de demostración: la dificultad de gestionar pruebas complejas cuando estas se expresan de forma lineal y textual. Frente a ello, se ha planteado una implementación basada en bloques de

demostración con dependencias, representadas mediante grafos acíclicos dirigidos (DAG), con el objetivo de aportar una estructura más clara, modular y visual al proceso de construcción formal.

A lo largo del desarrollo se ha conseguido implementar un prototipo funcional en entorno local que integra los principales elementos previstos desde el inicio: gestión de usuarios, creación de teoremas, edición de bloques, definición de dependencias entre bloques, visualización gráfica de dichas relaciones e integración con un servicio de validación Lean. Este resultado permite afirmar que la propuesta planteada es técnicamente viable y que puede materializarse en una aplicación software coherente con los principios de modularidad y separación de responsabilidades propios de la ingeniería del software.

Uno de los principales valores del trabajo reside en haber trasladado estos principios al ámbito de la demostración formal. En lugar de tratar cada prueba como un único bloque de texto, la solución desarrollada permite descomponerla en partes relacionadas entre sí, favoreciendo una organización más explícita del razonamiento. Esta aportación no pretende sustituir a Lean ni a sus entornos oficiales, sino ofrecer una capa adicional de estructuración e interacción que facilite la comprensión y gestión de las demostraciones.

En conjunto, se han cumplido satisfactoriamente los propósitos del proyecto, al transformar un problema concreto en una solución software analizada, diseñada, implementada y evaluada. Aunque el sistema obtenido no constituye un producto final plenamente evolucionado, sí proporciona una base funcional, coherente y ampliable para trabajos futuros.

7.2 Grado de cumplimiento de los objetivos

En relación con los objetivos definidos al inicio del trabajo, puede considerarse que el objetivo general se ha alcanzado de forma satisfactoria. Se ha diseñado e implementado una plataforma web capaz de gestionar demostraciones formales en Lean mediante una aproximación modular, con bloques independientes, dependencias explícitas y validación formal integrada.

Del mismo modo, los objetivos específicos vinculados al análisis del problema, al diseño de un modelo basado en bloques, a la representación de dependencias mediante un grafo acíclico dirigido (DAG) y al desarrollo de una aplicación web funcional han sido alcanzados. También se ha implementado una gestión básica de usuarios y roles, así como un mecanismo de visualización de dependencias y un flujo de validación conectado con Lean en entorno local.

Por tanto, puede concluirse que el proyecto ha alcanzado sus metas principales dentro del alcance previsto. El resultado obtenido permite considerar que la propuesta planteada es válida como solución software aplicada al problema identificado y constituye una base funcional y

coherente sobre la que podrían desarrollarse futuras ampliaciones.

7.3 Líneas de trabajo futuro

A partir del trabajo desarrollado, se identifican varias líneas de continuidad que permitirían ampliar y mejorar la solución obtenida. Una primera línea de evolución consiste en reforzar el modelo de colaboración entre usuarios. En la versión actual, la plataforma distingue entre permisos de edición y visualización, pero futuras versiones podrían incorporar mecanismos más avanzados de trabajo compartido, como control de versiones, gestión de cambios concurrentes o resolución de conflictos entre modificaciones simultáneas. Este tipo de cambios permitirían acercar la aplicación a un entorno colaborativo más completo, manteniendo la estructura modular planteada en el proyecto.

Otra línea de mejora se sitúa en la visualización del grafo de dependencias. Aunque la representación actual cumple adecuadamente su función estructural, sería interesante enriquecerla con información adicional, opciones de filtrado por niveles de dependencia, agrupación de subpruebas y mecanismos de navegación más avanzados. Estas mejoras permitirían explotar con mayor profundidad una de las aportaciones diferenciales del trabajo: la representación explícita de la estructura lógica de la demostración mediante grafos acíclicos dirigidos (DAG).

Desde la perspectiva de la validación formal, una línea futura especialmente interesante consistiría en ampliar el alcance actual del sistema más allá de Mathlib, incorporando soporte para otros contextos, bibliotecas o configuraciones del ecosistema Lean. Esta evolución permitiría aumentar la flexibilidad del prototipo y facilitar su aplicación a escenarios de demostración más diversos.

Finalmente, también resultaría especialmente interesante evaluar la utilidad real de la plataforma en contextos docentes o de aprendizaje. Dado que una de las motivaciones del trabajo es mejorar la comprensión estructural de las demostraciones, una continuación razonable consistiría en analizar si esta aproximación modular y visual aporta ventajas efectivas frente al flujo textual tradicional en tareas de estudio, revisión o enseñanza de demostración formal.

USO DE LA IA GENERATIVA

Durante la elaboración de este trabajo se han utilizado herramientas de inteligencia artificial generativa como apoyo puntual en tareas auxiliares de redacción, reformulación de fragmentos de texto, corrección lingüística y traducción entre idiomas.

Asimismo, estas herramientas se han empleado de forma ocasional para ayudar en la resolución de problemas

concretos de código durante el desarrollo del proyecto. En todo momento, su uso ha sido complementario y limitado, manteniéndose siempre el control, la revisión y la validación final del contenido por parte del autor.

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a mi tutor, Antonio Lozano Bagen, por su orientación, seguimiento y disponibilidad a lo largo del transcurso de este trabajo. Sus observaciones y recomendaciones han sido de gran utilidad tanto para delimitar el alcance del proyecto como para mejorar su planteamiento y desarrollo. Sin su ayuda no habría sido posible alcanzar los resultados obtenidos.

También quiero agradecer a mi familia y a mi pareja, que me han acompañado durante este proceso, por su apoyo, paciencia y confianza constantes. Este trabajo supone el cierre de una etapa muy bonita para mí que no habría sido posible sin su presencia.

BIBLIOGRAFÍA

- [1] The Lean Language Reference. Available: lean-lang.org/doc/reference/latest/.
- [2] J. Avigad, L. de Moura, S. Kong, and S. Ullrich, Theorem Proving in Lean 4. Lean Project, 2026. Available: lean-lang.org/theorem_proving_in_lean4/.
- [3] "Mathematics in Lean". Lean Community, 2025. Available: leanprover-community.github.io/mathematics_in_lean/.
- [4] Lean Project, "Lake," in The Lean Language Reference. Available: lean-lang.org/doc/reference/latest/Build-Tools-and-Distribution/Lake/.
- [5] Rocq Development Team, "Rocq Prover," official website. Available: rocq-prover.org/docs.
- [6] Isabelle Developers, "Isabelle," official website. Available: isabelle.in.tum.de/documentation.html.
- [7] leanprover, Lean 4 VS Code Extension Manual. GitHub repository documentation. Available: github.com/leanprover/vscode-lean4/blob/master/vscode-lean4/manual/manual.md.
- [8] React, "Learn React," official documentation. Available: react.dev/learn.
- [9] Vite, "Getting Started," official documentation. Available: vite.dev/guide/.
- [10] FastAPI, "FastAPI", official documentation. Available: fastapi.tiangolo.com.
- [11] PostgreSQL Global Development Group, PostgreSQL Documentation. Available: postgresql.org/docs/.
- [12] Docker, Inc., "Docker Documentation". Available: docs.docker.com/guides/.
- [13] XYFlow, "React Flow," official documentation. Available: reactflow.dev.
- [14] Dagre, "Dagre," GitHub repository documentation. Available: github.com/dagrejs/dagre.
- [15] L. de Moura and S. Ullrich, "The Lean 4 Theorem Prover and Programming Language," in Automated Deduction – CADE 28, A. Platzer and G. Sutcliffe, Eds. Cham, Switzerland: Springer, 2021, pp. 625–635. Available: https://doi.org/10.1007/978-3-030-79876-5_37.
- [16] The mathlib Community, "The Lean Mathematical Library," in Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '20), New York, NY, USA: Association for Computing Machinery, 2020, pp. 367–381. Available: <https://doi.org/10.1145/3372885.3373824>.

APÉNDICE

A1. CAPTURAS DE LA INTERFAZ



Fig. 5. Pantalla de login

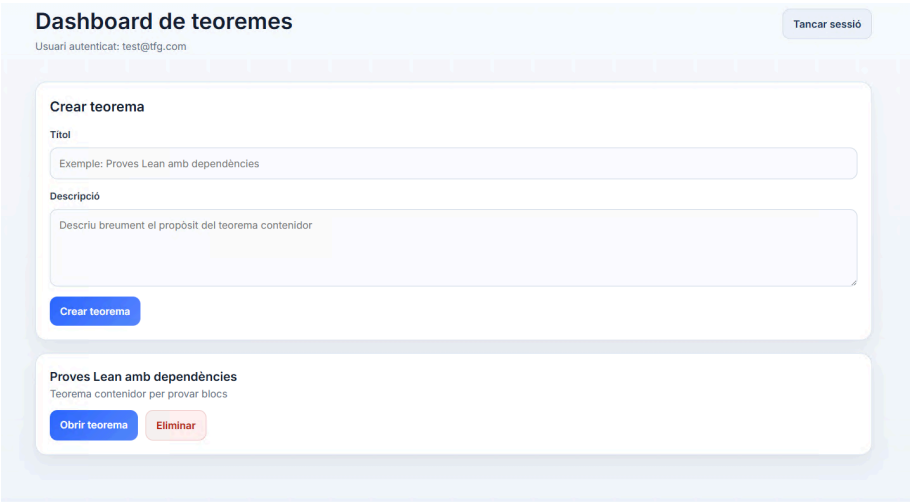


Fig. 6. Dashboard en modo editor

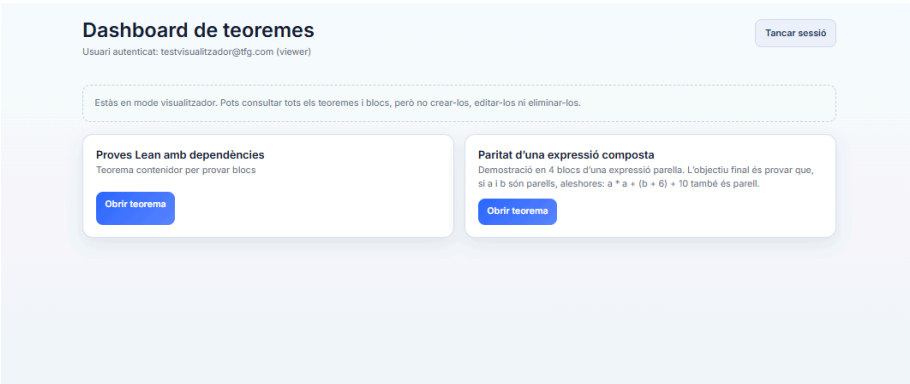


Fig. 7. Dashboard en modo visualizador

Proves Lean amb dependències

Tornar al dashboard

Teorema contenidor per provar blocs

Crear bloc

Escriu la declaració Lean amb `:= by`.

Les dependències han de ser els ids reals dels blocs, separats per comes.

Lean declaration

lemma lemmaA : Even 2 := by

Proof content

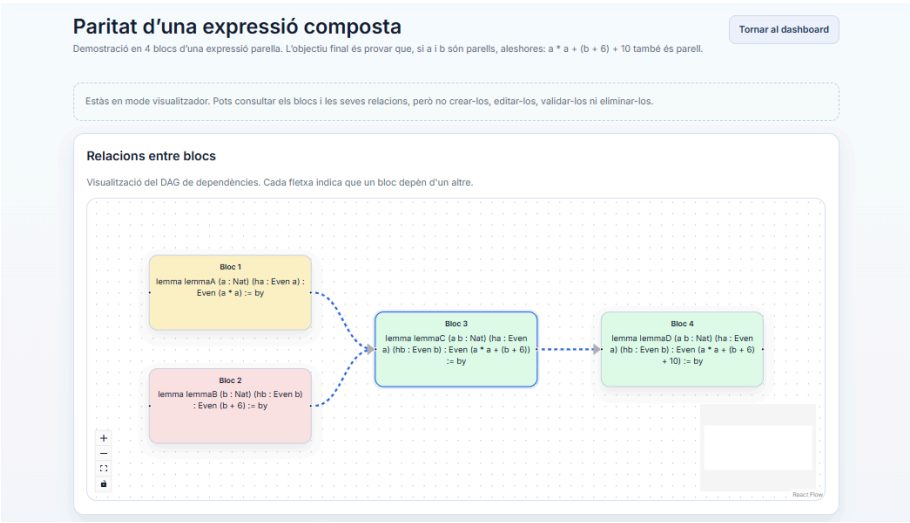
norm_num

Dependencies (ids reals separats per comes)

8,11

Afegir bloc

Fig. 8. Vista inicial de un teorema en modo editor



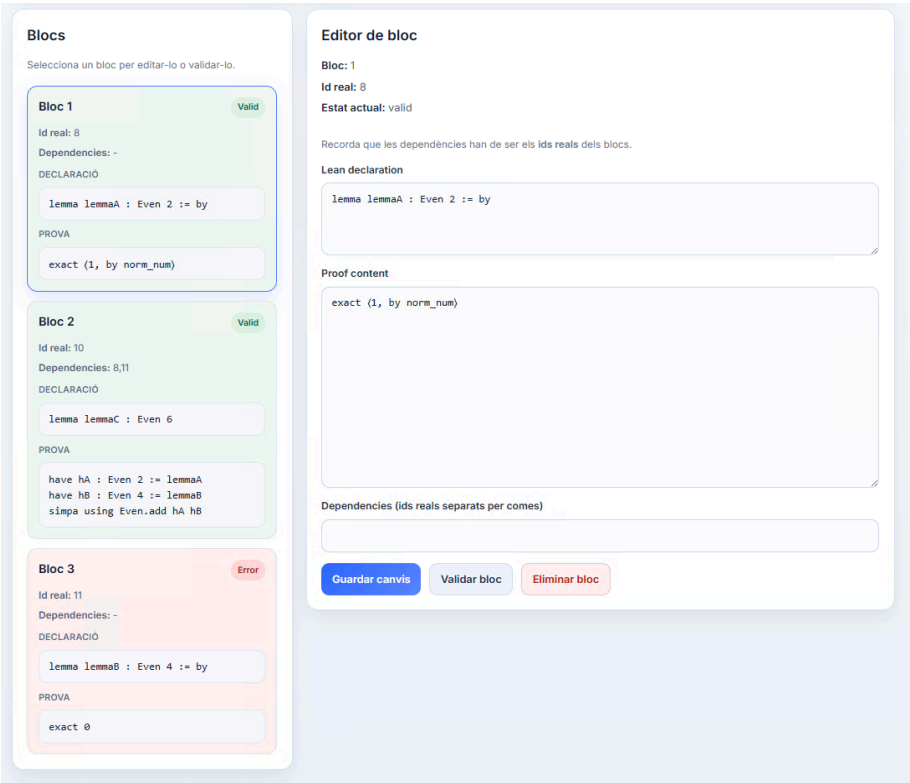


Fig. 10. Vista de bloques en modo editor

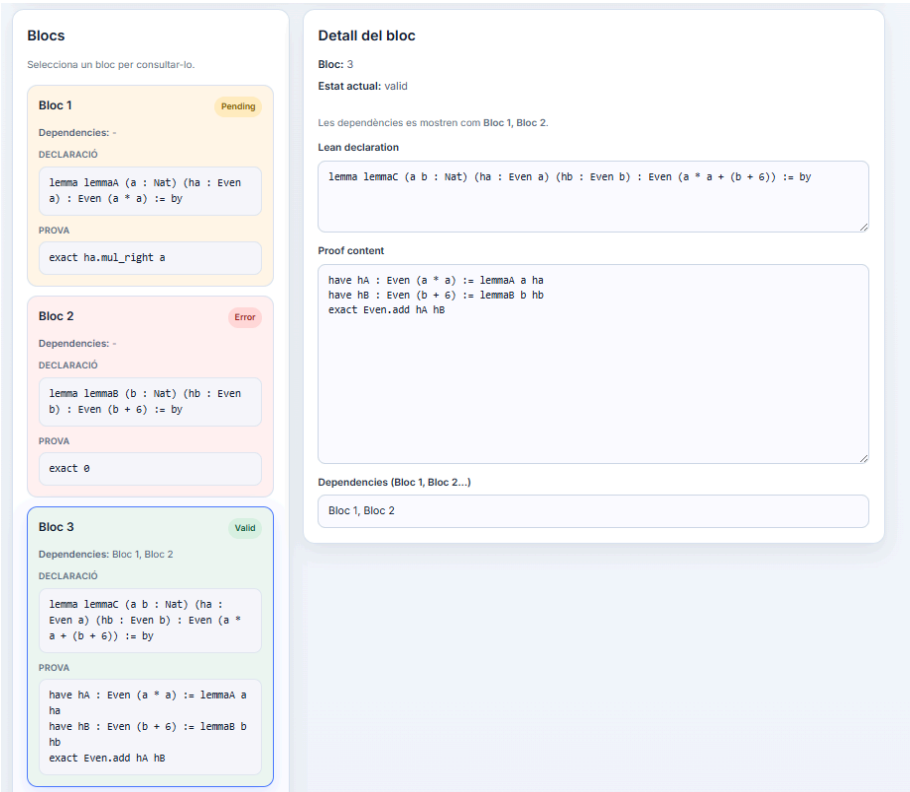


Fig. 11. Vista de bloques en modo visualizador