Tsubame: simulador de redes informáticas basado en tecnologías web

Gonzalo García Navarro

Resumen– Dada la complejidad de las herramientas de simulación de redes que existe actualmente en el mercado, nace la necesidad de una herramienta sencilla e intuitiva que permita a los estudiantes crear redes virtuales de forma rápida y segura. Este artículo pretende presentar una herramienta pensada para el uso académico que permita a alumnos crear redes virtuales, aprender conceptos básicos de redes y usar un entorno seguro para experimentar, obteniendo así una base sólida de conocimiento.

Palabras clave– Redes informáticas, redes virtuales, simulador, tecnologías web

Abstract– Given the complexity of the network simulation tools currently available on the market, there is a need for a simple and intuitive tool that allows students to create virtual networks quickly and safely. This article aims to present a tool designed for academic use that allows students to create virtual networks, learn basic networking concepts and use a secure environment to experiment, thus obtaining a solid knowledge base.

Keywords– Computer networks, simulator, virtual networks, web technologies

1 Introducción

A CAUSA del avance incesante de la tecnología, cada vez los humanos se encuentran más interconectados, lo cual implica a su vez un mayor riesgo para la intimidad de los mismos. Para velar por la seguridad de todos, es estrictamente necesario contar con buenos profesionales en el sector de la gestión de redes y ciberseguridad. Para en un futuro tener estos profesionales, los alumnos que se encuentran estudiando actualmente en el área de redes deben obtener una base sólida de conocimiento. Es por eso que nace la necesidad de una herramienta enfocada completamente a los estudiantes que les permita simular redes, experimentar en un entorno seguro y por consiguiente entender mejor los conceptos básicos. Este artículo pretende plantear de forma detallada el desarrollo de una herramienta de simulación de redes que intenta diferenciarse de las herramientas clásicas ya existentes, enfocándose en ser un recurso en el que los alumnos de redes informáticas puedan apoyarse.

El artículo consta de diferentes secciones bien diferenciadas. Empezando por la situación actual o el estado del arte, donde se muestra en que tipo de software está basada la herramienta y en que se diferencia de las demás. A continuación se enumeran de forma concisa los objetivos (y las tareas) marcados que se han tenido en cuenta para el correcto desarrollo de la misma. En las secciones contiguas se detalla el desarrollo de la herramienta y de todos los componentes que la rodean, también se documenta el método seguido para cumplir los objetivos marcados. En últimas secciones se describen los resultados conseguidos, además de una breve explicación de cuál es el alcance del proyecto en un futuro, de las conclusiones obtenidas y una pequeña reflexión sobre de la experiencia adquirida de dichos resultados.

2 Estudio de la situación actual

Actualmente, existe una basta cantidad de software que es capaz de simular redes muy completas; sin embargo, la mayoría de estos resultan ser extremadamente complejos en cuanto a su uso, además de ser muy costosos para el usuario medio.

La mayoría de herramientas usadas para el aprendizaje de redes suelen ser las famosas aplicaciones de CISCO. Si bien estas herramientas suelen ser de muy alta calidad y son muy útiles para profesionales ya formados, resultan muy poco flexibles y prácticas para los estudiantes. Un ejemplo

• E-mail de contacte: gonzalo.garcian@autonoma.cat
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Kevin Chow (departament)
• Curs 2022/23

de ello es Cisco Packet Tracer (CPT) [1], una aplicación de escritorio para la simulación de redes sencillas. En los últimos años, es cierto que la herramienta CPT se ha actualizado para garantizar un uso más sencillo para usuarios menos avanzados, pero sigue teniendo pequeñas barreras para acceder a estos recursos. Una de estas barreras es la necesidad de registrarse y cursar varios cursos de CISCO, además de solo tener acceso a la herramienta si es descargada e instalada en la máquina de forma local. Otra referencia en la que se basa el proyecto es un software llamado Graphical Network Simulator (GNS) [2] que es una excelente herramienta de escritorio para la simulación de redes, que a diferencia de CPT es open source y a penas tiene limitaciones en sus funcionalidades.

También existen herramientas de aprendizaje, pero que no son simuladores, como puede ser CS4G Netsim [3] o Netsim Bossom [4]. Esta última herramienta es muy similar al proyecto planteado, un simulador de redes que existe íntegramente en la web; sin embargo, este resulta ser poco flexible, ya que no permite la creación topologías de red personalizadas. Se trata de una serie de lecciones con sus respectivas topologías prefabricadas.

Todas las herramientas mencionadas tienen sus puntos fuertes y puntos débiles y son una gran fuente de inspiración para este proyecto; sin embargo, es necesario nuevas herramientas que se adapten a tiempos modernos y que sean completamente accesibles para todos.

3 DESARROLLO

En esta sección se integran tres aspectos relevantes del proyecto. La subsección (3.1) presenta los objetivos, que refleja los puntos más importantes del proyecto. A continuación, la subsección (3.2) se describe detalladamente las metodologías utilizadas, cómo se han implementado y por qué. Por último, la subsección (3.3) explica minuciosamente el proceso de desarrollo y en líneas generales cómo se ha gestionado todo el desarrollo del proyecto.

3.1. Objetivos

Este proyecto tiene como objetivo principal crear una pequeña plataforma dónde los alumnos puedan, gracias a un simulador de redes, crear redes virtuales o topologías simples con las que poder experimentar y aprender. Todo ello a través de un navegador web. Existen tres partes que constituyen el proyecto, las cuales a su vez tienen sus propios objetivos:

- Crear un simulador de redes que actúa como núcleo de la plataforma. El simulador debe permitir al usuario crear sus propias topologías y, por lo tanto, debe ser capaz de:
  - Crear un lienzo dónde poder añadir nodos.
  - Poder posicionar los nodos por el lienzo.
  - Interaccionar e interconectar los nodos a través redes.
  - Gestionar la lógica de las redes y sus nodos.
  - Simular protocolos básicos.
  - Permitir al usuario ver de forma sencilla las conexiones y paquetes enviados de un nodo a otro.
  - Permitir al usuario gestionar las características los nodos.
- Implementar un sistema de usuarios. Los usuarios deben poder:
  - Registrarse en la base de datos.
  - Crear redes en el simulador.
  - Guardar el estado del simulador en la base de datos. Este es representado como un proyecto.
  - Tener un espacio privado en el que aparezca un listado de proyectos que poder cargar en el simulador.
  - Organizar los proyectos así como configurar sus características.
  - Poder compartir a otros usuarios los proyectos guardados.
  - Usar el simulador sin necesidad de registro.
- Construir una plataforma que:
  - Guíe a los alumnos en conceptos básicos de redes.
  - Guiar a los alumnos en el uso de todas las funcionalidades del simulador.
  - Incentivar a los alumnos a contribuir al proyecto y por consiguiente generar una estructura óptima para ello en la plataforma de control de versiones seleccionada.

Los objetivos generales y el concepto básico del proyecto están bien reflejados en la lista anterior, pero es preciso insistir en que este también pretende ser un proyecto de código abierto que incite a los alumnos a añadir todas las herramientas y funcionalidades que vean necesarias en el simulador. Esta característica es imprescindible, puesto que más allá de enseñar a los alumnos los conceptos básicos, se alienta a estos a seguir indagando en el funcionamiento de las redes para implementar nuevas, expandiendo así en gran medida su entendimiento de la materia.

Para poder llegar a cumplir todos los objetivos anteriormente mencionados es imprescindible planificar las tareas a realizar. Para gestionar todas las tareas también es necesario utilizar una metodología capaz de dar un apoyo en la gestión de estas.

3.2. Metodología

Existen muchas metodologías y la gran mayoría son útiles cuando los proyectos se realizan en grandes o medianos equipos de desarrolladores. Dada la naturaleza del proyecto, este está realizado por una persona, con lo cual es conveniente seleccionar una metodología que encaje en estas características. En general, las metodologías Agile [5] están muy extendidas en el mundo del desarrollo de software, ya que son muy útiles para proyectos que necesitan flexibilidad. Es una filosofía basada en trocear el proyecto en muchas pequeñas tareas y en concreto la metodología Agile Kanban [6] parece encajar muy bien en equipos de

desarrollo de muy pocas personas. Dado que es el caso, la metodología a seguir para este proyecto es Kanban.

Kanban se trata de un tipo de metodología Agile basada principalmente en la división de un proyecto en pequeñas tareas. Uno de los puntos fuertes reside en lo visual que es, ya que, propone además de trocear el proyecto en pequeñas tareas, situar estas en un tablero conformado de varios carriles o columnas. Estos carriles muestran el estado de las tareas. Lo más usual en Kanban es tener cuatro carriles, uno para las tareas que aún no se han asignado, un segundo para las tareas que se están realizando en ese momento, un tercero para las tareas que se están revisando y un último para las tareas ya realizadas. Esto ayuda mucho a ver de forma rápida y visual el proceso del proyecto y si existen cuellos de botella, es decir, si se están haciendo demasiadas tareas a la vez.

Junto a Kanban es conveniente añadir una metodología de pruebas. El desarrollo dirigido por pruebas (TDD) se basa en la creación de pruebas en primera instancia, a continuación la creación del código fuente, pasar correctamente la prueba y reestructurar el código. El flujo de vida de una tarea, si estas dos metodologías se combinan, se puede ver con más claridad en la figura (1).

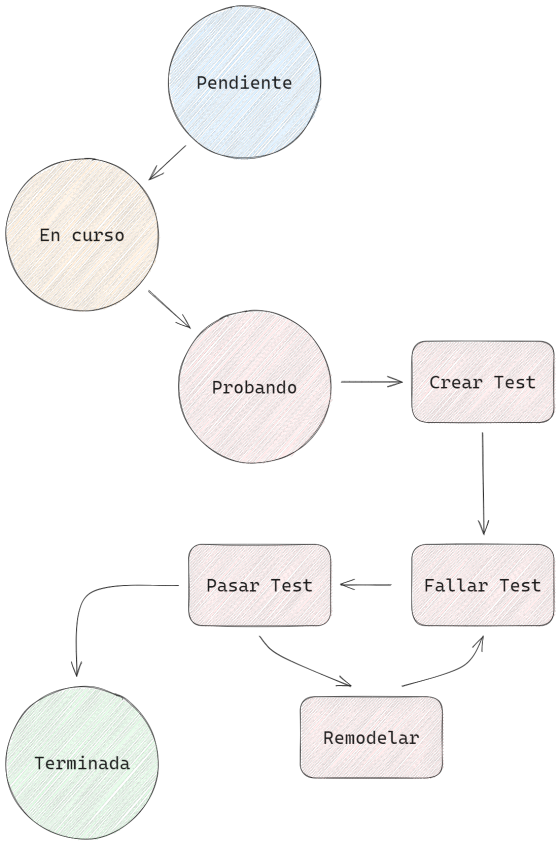


Fig. 1: Vida de una tarea

Para combinar estas dos metodologías se añade a Kanban una columna extra llamada 'probando' donde las tareas antes de pasar a 'terminada' son probadas con el método de desarrollo dirigido por pruebas (TDD).

Adicionalmente se está utilizando la herramienta online de Click Up [21]. Esta herramienta es extremadamente útil ya que permite implementar la metodología Ágile Kanban a la perfección que permite crear un espacio, este espacio puede contener varios directorios los cuales en su interior a su vez pueden contener su propio tablero de Kanban, en la figura (2) se puede observar esta jerarquía.

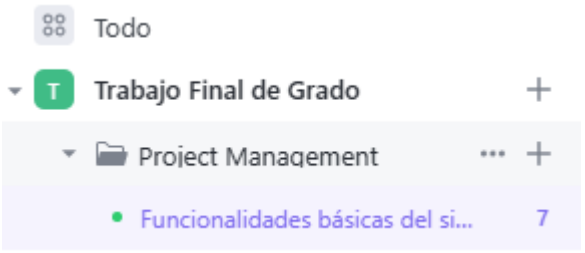


Fig. 2: Directorio de "Funcionalidades básicas del sistema"

Para organizar mejor el proyecto se ha creado un directorio para cada tarea que aparece en la sección V, así cada tarea puede tener su propio espacio y su propia tabla de Kanban para poder añadir las subtareas que se vean necesarias. Además de eso en la figura (3) se puede observar como cada tarea tiene su propia bandera. Estas banderas indican la prioridad, es un añadido que no estaba en la planificación inicial, pero que ha surgido de la necesidad de ordenar de alguna forma las subtareas. El rojo implica alta prioridad, el amarillo prioridad media-alta, el azul prioridad normal y el gris prioridad baja.

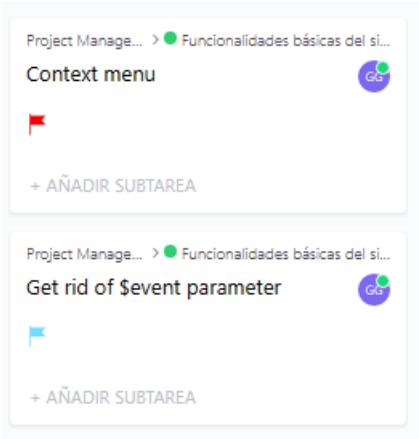


Fig. 3: Tabla Kanban de "Funcionalidades básicas del sistema"

También cabe destacar que se ha utilizado el esquema principal de Kanban dónde existen 4 estados de la tarea Open, In progress, Review y Closed. En Open se ponen todas las tareas y se le asigna una prioridad, en in progress se añaden aquellas tareas que estén en progreso, en Review se pasan todas las tareas las cuales se estén revisando y por último en Closed aquellas tareas que no se vayan a modificar más. En el estado de Review se utilizan dos herramientas de pruebas, una Unit Testing llamada Vitest [22] y una End2End llamada Cypress [23]. Vite test sirve para probar funciones de javascript para que dado un input salgan los output deseados y Cypress realiza de forma automática simulaciones en la página web como si fura un usuario normal que está navegando por la página.

3.3. Proceso de desarrollo

El proceso del desarrollo va muy ligado a los objetivos mostrados en la sección (3.1) dónde se muestra, que el proyecto se divide en tres partes. Lo prioritario ha sido crear un base sólida, en este caso el propio simulador. A continuación, cuando se ha conseguido construir el núcleo del proyecto, se ha creado alrededor una plataforma más enfocada a los usuarios. Por último se ha generado una serie de recursos para enriquecer la plataforma y que los usuarios sean capaces de entender por completo el proyecto. A continuación se describen las tareas realizadas para construir las partes que constituyen el trabajo.

3.3.1. Configuración del entorno de trabajo

Para todo proyecto de software es importante planificar un buen entorno y flujo de trabajo, así como la pila de tecnologías DevOps [7] a utilizar. En este caso existen tres partes esenciales; GitHub para el control de versiones de archivos, un servicio de Supabase para el almacenaje de datos y un servicio de Cloudflare para el hosting de la página web.

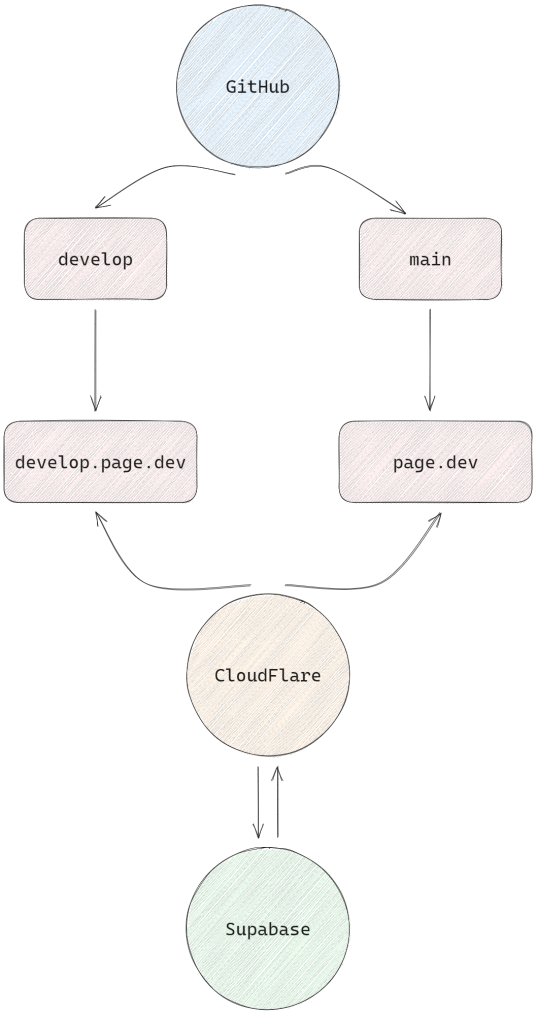


Fig. 4: Pila de tecnologías del entorno de trabajo

Las tareas realizadas para crear la base del proyecto son:

- Seleccionar la pila de tecnologías DevOps: tal como se ha mostrado en la figura (4), se utiliza Git/GitHub como control de versiones, CloudFlare para que todo el proyecto esté alojado en un servidor y Supabase para almacenar los datos de los usuarios.
- Crear un repositorio de GitHub. En este servicio se almacenan todas las versiones del proyecto y está organizado por ramas. Cada rama tiene su rol en el proyecto.
  - Crear una rama llamada main: en esta rama se suben todos los cambios que no son experimentales y que están, después de una serie de exhaustivas pruebas, presuntamente libre de fallos. Esto es muy útil para ofrecer al usuario la máxima calidad posible del producto.
  - Crear una rama llamada develop: en esta rama se suben todos los cambios experimentales y que aún no han sido probados. Gracias a esta rama, se tiene un entorno seguro para poder probar funcionalidades nuevas sin afectar al usuario final.
- Configurar el hospedaje del proyecto en el servicio de Cloudflare. Este tiene un servicio llamado 'Pages' que permite crear proyectos en los que alojar páginas web estáticas. Este servicio es ideal para el proyecto, ya que, además de ser gratuito, permite asociar una cuenta de GitHub en los que este servicio estará mirando constantemente el repositorio que le indiques para que cada vez que se detecte un cambio, la página web se actualice con dichos cambios.
  - Crear un proyecto del servicio "Pages".
  - Conectar la cuenta de GitHub con el proyecto de "Pages".
  - Configurar el despliegue del código para ambas ramas.
- Seleccionar la pila de tecnologías para backend.
- Configurar el servicio de base de datos Supabase. Este, al igual que Cloudflare, permite crear proyectos independientes de forma grauita. En este caso se trata de un servicio que permite crear bases de datos relacionales, además de ofrecer un sistema de autorización usuario ya implementado.
  - Crear un proyecto por cada rama.
- Seleccionar la pila de tecnologías para frontend.
  - Framework para la plataforma.
  - Framework JavaScript para el simulador.
  - Adaptador o puente entre él los dos frameworks anteriores.
  - Framework para realizar pruebas tanto end-to-end como unit-test.
  - Framework para los estilos de la página.
  - Librería para la creación e integración del proyecto.
- Creación del proyecto e integración de todas las tecnologías.

Primero el código se programa en local desde el IDE y una vez se han realizado cambios significativos, estos se suben al repositorio de GitHub [8], principalmente se suben primero las cosas a la rama de desarrollo (develop) y luego más tarde en el desarrollo, cuando se ha comprobado que todo funciona a la perfección, se actualiza la rama principal (main). A continuación, el servicio de CloudFlare Pages [9], que está constantemente mirando los cambios del repositorio, hace una copia de los nuevos cambios, construye el proyecto de nuevo y lo almacena en el espacio asignado en sus servidores para que la página web [10] (que está asociada al proyecto) se actualice. Toda la información de los usuarios se guarda en la base de datos de Supabase [11] y el proyecto alojado en Cloudflare utiliza la base de datos para gestionar los estados de los usuarios.

No tener un backend propio tiene sus ventajas y desventajas, por una parte, no existen preocupaciones en cuanto a brechas de seguridad y carga de peticiones. Por otra parte, a veces puede resultar poco flexible o escalable y por supuesto, al ser un servicio gratuito, tiene ciertas limitaciones.

Para el frontend la pila de tecnologías que se usan en el proyecto son las que aparecen en la figura (5).

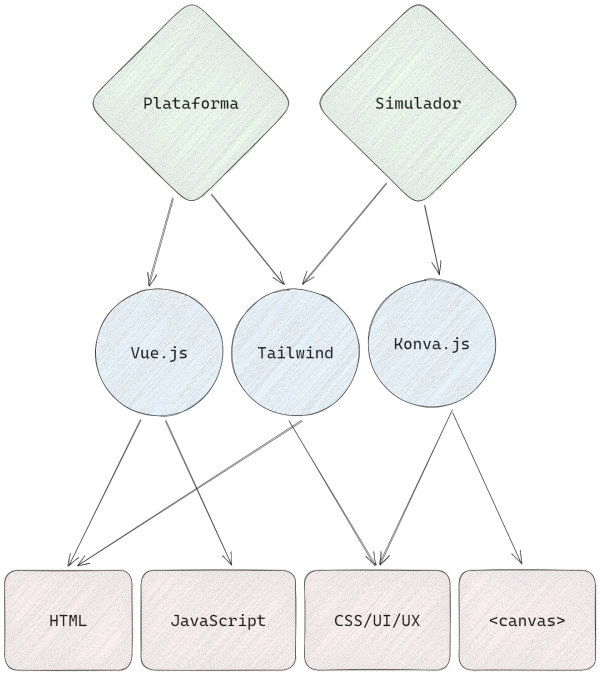


Fig. 5: Pila de tecnologías frontend

Para todos los elementos de la página web se utiliza un framework de Javascript llamado Vue [12], este se encarga de controlar el DOM de la página web de forma dinámica. Para la parte más visual, dada la inexperiencia en la maquetación de páginas web y creación de interfaces de usuario (UI), se utiliza Tailwind [13], que es un framework de CSS y permite crear UI de forma muy sencilla. En cuanto al simulador se refiere, este está construido a partir de HTML y Javascript también, más concretamente basado en la etiqueta 'canvas' de HTML que permite la manipulación de píxeles en un lienzo. Para poder realizar el proyecto con éxito y a tiempo, es prácticamente obligatorio una librería capaz de manipular estos píxeles y que además proporcione una capa de abstracción para la manipulación de conjuntos de píxeles. La librería a utilizar es Konva.js que tiene todo lo que se necesita para este proyecto. Además, para la construcción del proyecto se ha utilizado Vite [14] que sirve para construir de forma automática la estructura base del sistema de ficheros del proyecto y Pinia [15] que es un framework que complementa al framework de Vue para gestionar la memoria de la página. Esto es especialmente útil, ya que, de esa forma, se puede tener un estado global al cual puedan acceder todos los componentes y así existe una forma de comunicación eficiente entre estos. Como punto a destacar, al principio la planificación inicial contenía una librería extra llamada Vue-Konva.js [15] que en principio convertía la librería de Konva.js en etiquetas especiales de Vue.js, esto parecía muy buena idea en un principio, ya que mantendría un orden lógico y visual en el código, sin embargo, lejos de la realidad, esta librería empezó a complicar de forma innecesaria el código. Por lo tanto, al final se ha acabado optando por usar el modelo básico de la librería de Konva.

3.3.2. Rutas internas de la página web

Un paso importante a realizar para crear una plataforma, es la planificación de un buen sistema de 'routing' interno de la página web. En este caso, la página web es de tipo Single Page Application (SPA) [4] . Este sistema define como los usuarios navegan por la página y qué contenido podrán encontrar en esta. Las tareas realizadas se han dividido en:

- Crear página de inicio que incluye:
  - Página de documentación (docs): en este se explica a los interesados el proceso a seguir para contribuir al proyecto y documentación que explica cómo funciona por dentro.
  - Página para hablar sobre el proyecto (blog): actualizaciones que el autor da sobre los últimos cambios del proyecto.
  - Login/Signin: Los usuarios pueden registrarse o acceder a los recursos de su cuenta a través de un proceso de autenticación.
  - Sandbox: en este el usuario puede acceder al simulador sin necesidad de tener una cuenta.
- Crear página de dashboard que incluye:
  - Perfil usuario.
  - Proyectos: los estados de cada topología creada hasta el momento.
- Crear página del simulador.
- Integrar las páginas en un sistema SPA.

La página web está pensada para ser una Single Page Application, es decir, una página web en el que en ningún momento se recargue la página. Para conseguir este efecto de fluidez, se ha usado usar una librería de View llamada Routing [20] que permite cargar en una misma página varias vistas de Vue a la vez. Esto es un concepto importante porque se debe tener muy en cuenta a la hora de programarlo. La figura (6) muestra tanto la jerarquía de las vistas o caminos que se puede seguir en la página web, como el flujo que sigue el usuario para llegar hasta el simulador.

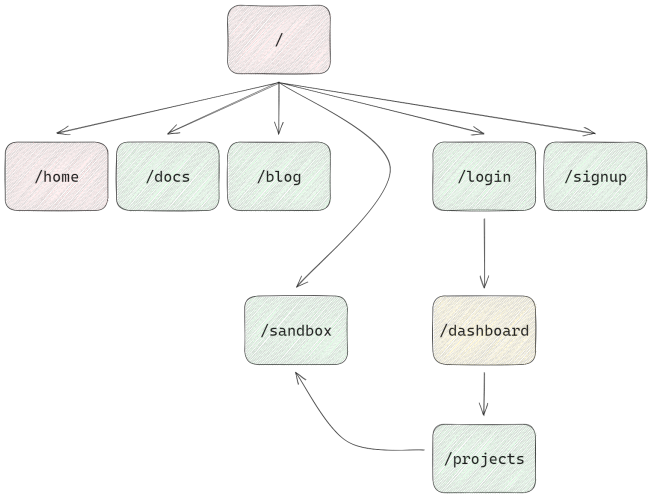


Fig. 6: Rutas básicas de la SPA

Primero está el path principal '/' que es el que aparece cuando un usuario ingresa en la página. Este está programado de tal forma que directamente redirecciona al usuario a la vista de 'home', a partir de ahí, el usuario puede acceder a través de la barra de navegación a 'docs' para obtener más información sobre el proyecto, a 'blog' para ver las últimas noticias publicadas por el autor, 'login' para acceder a su cuenta o 'signup' para crearse una, por último 'sandbox' que lleva a un simulador el cual no se encuentra vinculado a ninguna cuenta y en el que se puede experimentar sin registro previo. Si el usuario accede a su cuenta a través de 'login', se carga directamente 'dashboard' que es una vista que contiene una barra de navegación con proyectos. En proyectos se muestra un listado de proyectos guardados en la nube, los cuales están vinculados a la cuenta del usuario. Si este accede a uno de estos proyectos, puede acceder a la topología de red asociada al proyecto y se abre directamente el simulador cargando así los datos para que pueda seguir trabajando en el mismo punto en el que se encontraba la última vez que lo abrió.

3.3.3. Funcionalidades básicas del simulador

El simulador se basa en un lienzo infinito en el cual el usuario puede añadir nodos. Entendiendo como nodos cualquier elemento que represente un objeto real como un ordenador, un router... Este también se divide en 3 partes, el canvas, que es el lienzo controlado por Konva.js en el cual el usuario puede crear topologías. Un panel en el que se muestran los nodos disponibles para añadir a la topología. Por último, un terminal, que permite ejecutar comandos e interactuar con la topología. En este apartado se reflejan las funcionalidades creadas únicamente dentro del simulador:

- Añadir nodos al lienzo del simulador.
- Mover los nodos (drag and drop).
- Rectangulo de selección de nodos.
- Cambiar el tamaño de los nodos (transform).
- Ampliar o alejar el lienzo (zoom in/out).
- Moverse por el lienzo.

En esta fase del proyecto era importante poder tener funcionalidades básicas del simulador, estas funcionalidades consisten en poder tener un conjunto de píxeles (nodo) y manipularlo a través de todo el lienzo. También era imprescindible poder navegar por el lienzo, seleccionar varios nodos a la vez, hacerlos más grandes o más pequeños, etc. Gracias a Konva y a la documentación que esta librería tiene, todas las funcionalidades han podido ser implementadas:

- Añadir nodos al lienzo del simulador: esta funcionalidad depende de la lógica de red básica con lo cual ha sido implementada en las siguientes tareas.
- Mover los nodos (drag and drop). Konva.js tiene una magnífica documentación que contiene una cantidad enorme de ejemplos, es por eso mismo que tanto esta funcionalidad como las siguientes a mencionar han sido sacadas de ejemplos ya creados que solo necesitaban ser adaptados. En concreto, el drag and drop de los nodos [16] ha sido muy sencillo de implementar.
- Cambiar el tamaño de los nodos (transform). También ha podido ser implementado con relativa facilidad y este ha tomado de ejemplo el código de la documentación [17]
- Ampliar o alejar el lienzo (zoom in/out). Este caso en concreto ha sido ligeramente complejo, puesto que tenía que ser compatible a su vez con la siguiente funcionalidad. El código base se ha sacado de la documentación [18] y ha sido modificado ligeramente. En concreto, se ha tenido que utilizar la posición relativa del ratón y no la posición actual. Esto es debido a que al permitir que el lienzo se mueva y que este sea infinito, si usamos las coordenadas del puntero del ratón correspondientes a su posición con respecto a la ventana del navegador, jamás podremos saber realmente dónde se encuentra en el lienzo el objeto al cual se quiere hacer zoom/seleccionar o cualquier acción que necesite obtener coordenadas con respecto al lienzo.
- Moverse por el lienzo. Al igual que el anterior, este ha sido ligeramente modificado con respecto al ejemplo de la documentación [19] para que los controles se adapten a lo que el usuario está acostumbrado a utilizar en este tipo de software.
- La selección de varios nodos a la vez [17] con un rectángulo de selección, al igual que ocurre en los sistemas operativos como Windows. Al igual que las anteriores, ha sido modificado para tener en cuenta las coordenadas relativas.

3.3.4. Nodos básicos y estructura interna

En esta tarea se han creado las estructuras básicas de los nodos principales: host, router, network. Cada nodo, excluyendo el nodo que representa la red en sí, tiene estas características y su lógica asociada:

- Lógica e interfaz que representen la dirección física.
- Lógica e interfaz para representar una interfaz de red.

- Una interfaz de usuario integrada para representar las interfaces de red.
- La capacidad de conectar los nodos a un nodo que represente una red a través de un cable.

La figura (7) muestra la estructura de cada uno de los elementos.

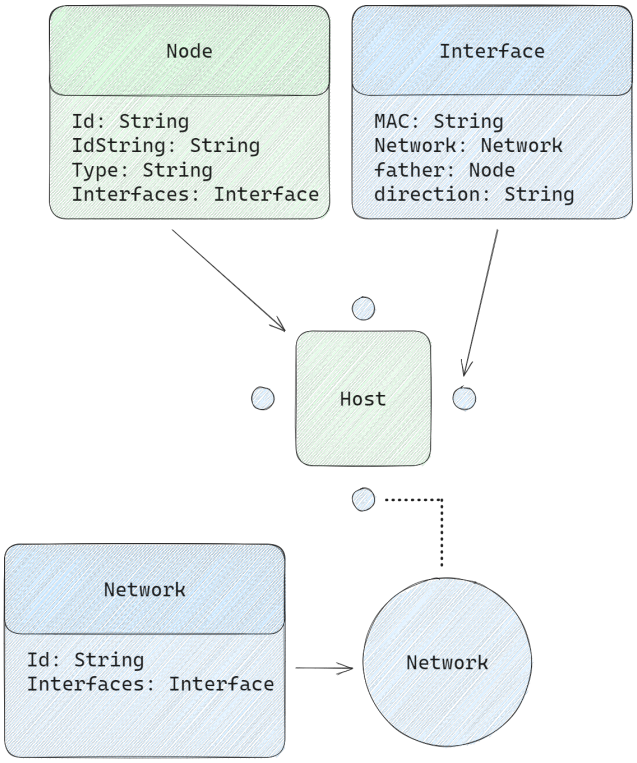


Fig. 7: Estructura interna de los nodos

Cada nodo está representado por una clase interna de Javascript. Internamente, un ordenador y un router son la misma clase, estos dos tienen asociados una lista de interfaces de red que a su vez también tienen su propia clase. El nodo de red, es una clase a aparte y contiene sus propios atributos.

Cada red contiene una lista de nodos que a su vez contiene una lista de interfaces, las cuales también contienen un campo de red el cual contiene una referencia a la red en la que se encuentra.

Esta estructura se ha creado para que el algoritmo de enrutamiento que se explica en la siguiente sección resulte menos complejo de tratar.

También se ha creado un sistema de almacenamiento local con el framework Pinia. Este sistema almacena todas las instancias de los nodos y una referencia a sus respectivas formas gráficas en el lienzo. Este sistema se ha implementado para tener un estado general del simulador para poder posteriormente enviar este estado al servidor y guardar el progreso del proyecto. Esto también se aplica al cargar el estado que recibe el usuario desde el servidor.

3.3.5. Algoritmo de enrutamiento

Antes, se pretendía simular de forma precisa todos los elementos de una red, sin embargo, no era un enfoque realista, además tampoco era un enfoque cercano a la idea de crear un sistema simplificado de una red para enseñar conceptos básicos a alumnos. Por lo tanto, se ha buscado una alternativa mucho más simplista y realista. Ahora toda la simulación depende de una función que simplemente, dada una topología y dado un nodo de inicio y un nodo de fin, devuelve el camino entre estos. Esto a efectos prácticos se podría decir que es un símil a lo que ocurre en una red real, lo que se suele llamar enrutamiento. Esta función es un algoritmo muy similar al algoritmo de Depth First Search (DFS) [3] pero que está modificado para que se comporte como un verdadero enrutamiento de datagramas. Gracias a esta base, se pueden simular prácticamente todos los protocolos básicos de red.

3.3.6. Protocolos implementados y otros comandos

El algoritmo anterior permite saber, dependiendo de una topología, el camino de un nodo a otro, esto permite crear un conjunto de comandos básicos que simulen el comportamiento de un nodo.

- Integración del protocolo ICMP parcialmente para el uso de la herramienta de ping.
- Integración del protocolo ARP: protocolo de resolución de direcciones físicas.
- Integración del protocolo DHCP: protocolo para conseguir una dirección lógica en una red.
- Integración del protocolo UDP: con fragmentación, transporte de información sin gestión de perdida.
- Integración del protocolo TCP: transporte de información con gestión de pérdida, no se ha incluido la funcionalidad de las windows ni fragmentación.
- Comando save: para indicar que se quiere guardar el estado de la topología en el servidor.
- Comando clear: para limpiar la terminal.
- Comando help: para imprimir por pantalla un texto que indica todos los comandos disponibles.

Todos los algoritmos anteriormente mencionados, son adaptaciones creadas para este proyecto, pero siguen los protocolos reales muy de cerca. Se han diseñado para ser una pequeña abstracción de los verdaderos protocolos.

3.3.7. Sistema de visualización de tramas

Puesto que la idea es que los alumnos puedan ver las tramas que van de un lado para otro, es conveniente crear un sistema en el que estos sean capaces de observar la estructura de estas. Para ello se ha creado un sistema de terminales y resultados. La terminal no es más que un intento de simular una terminal clásica de Linux en el que el usuario puede introducir comandos. En este caso los comandos introducidos se ejecutan mostrando por la misma terminal un resultado. Este resultado se compone de varias tablas que representan todas las tramas de todas las redes por las que pasa el datagrama. El resumen de la tarea realizada es:

- Sistema visual de terminales y datagramas.
- Construcción visual de los protocolos ICMP, ARP, DHCP, TCP y UDP.

- Lógica interna para simular un programa que intercepta el tráfico en la red: esto se consigue de forma individual para cada comando.

Además, cada vez que se crea un comando nuevo que simule un protocolo, a su vez se desarrolla la visualización personalizada del resultado de este (la tabla personalizada). Tener dos sistemas completamente separados hace que añadir código sea mucho más sencillo. En el componente del terminal está el código del comando, el cual utiliza el sistema de resultados para imprimir por pantalla la tabla resultante dependiendo del comando introducido. Todos estos sistemas son muy visuales, con lo cual quedarán mucho más claros cuando se presenten los resultados finales en la sección (4).

3.3.8. Sistema de autorización y navegación avanzada

Esta tarea ha consistido en conectar el frontend al backend e implementar el sistema de autorización de usuarios (login/signup) así como el control de acceso a las páginas de la web. La tarea ha resultado muy sencilla, ya que la base de datos utilizada (Supabase) ya tiene un sistema de autorización. Se ha creado la página de login y signup con el respectivo código para poderse conectar desde el cliente a la base de datos.

3.3.9. Sistema de almacenaje de estados

Creación de un sistema de almacenaje en el que el usuario pueda guardar el estado en el que se encuentra el simulador en ese momento. Esto incluye todos los nodos y sus características. También incluye la comunicación a base de datos para la carga de estados y proyectos.

3.3.10. Diseño de la plataforma y documentación

Se ha creado una documentación y un entorno para ayudar al usuario a utilizar la herramienta.

4 Exposición de resultados

La plataforma es completamente visual, lo que hace complejo exponer todos los resultados obtenidos. Para una buena comprensión de los mismos, a continuación se expone un caso de uso de la plataforma desde el punto de un usuario que jamás ha entrado a la plataforma. Para una mejor visualización del resultado, lo recomendado es acceder a la propia página. Para empezar, el usuario ingresa en la página principal donde tiene dos opciones, entrar en 'sandbox' y empezar experimentando con el simulador, o bien registrarse para posteriormente tener la opción de guardar el estado de una o varias topologías. Si el usuario se registra y vuelve a esta vista, tiene la opción de 'login' para acceder a su área de proyectos. Como se muestra en la figura (8) el usuario tiene una lista de los proyectos ya creados.

Al seleccionar uno de esos proyectos, se redirecciona al usuario al simulador en la vista llamada 'sandbox'. Aquí están las partes principales del proyecto, al lado izquierdo, una lista de los nodos con los que se puede experimentar, abajo un terminal para poder interactuar con la topología y por último un lienzo en el que poder crear las topologías.

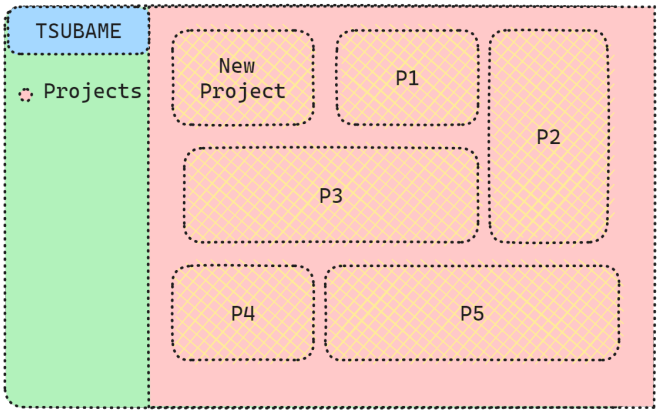


Fig. 8: Proyectos de un usuario

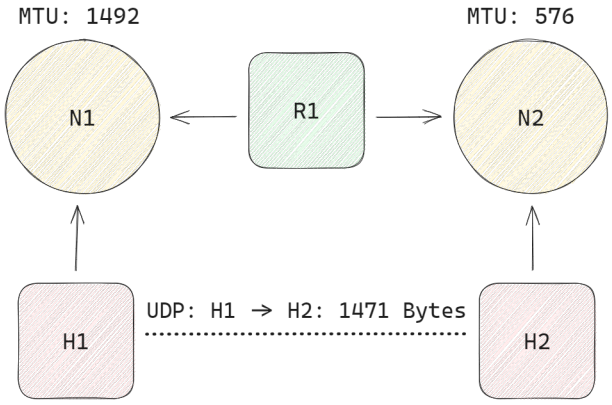


Fig. 9: Posible topología a simular

En esta casuística el usuario puede querer crear una topología como la mostrada en la figura (9) dónde tiene un router, dos redes y un host para cada una de las redes. También puede querer usar el comando udp con lo cual refleja que las redes tienen una MTU específica por la que enviar un mensaje de 1479 bytes. Para conseguir esto primero debe usar el menú izquierdo del simulador e ir arrastrando los elementos necesarios para construir la topología.

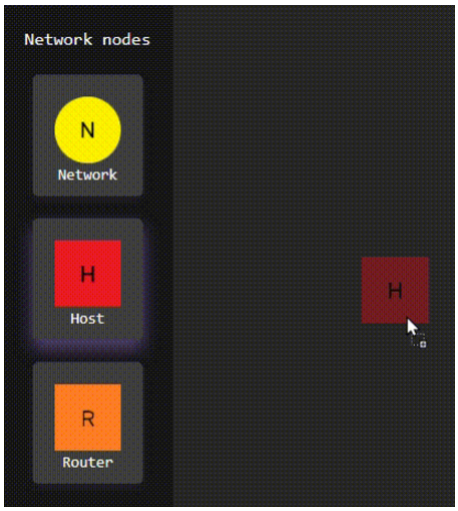


Fig. 10: Arrastrar nodos al lienzo

En la figura (10) se puede ver como el usuario puede ir arrastrando los nodos al lienzo. Una vez arrastrados todos

los nodos, debe interconectar los 'host' a través de las redes conectadas por el router.

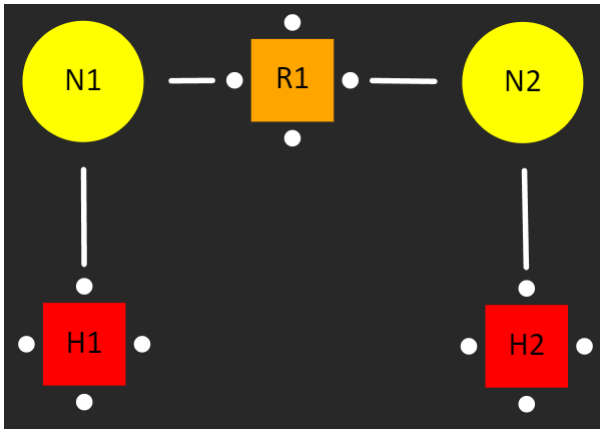


Fig. 11: Nodos conectados

Este resultado es posible gracias a que el usuario es capaz de poner el cursor encima de las interfaces y crear las líneas mostradas en la topología de la figura (13) simplemente arrastrando la interfaz a la red a la que se quiere conectar. Todas las acciones realizadas no solo generan un resultado gráfico en el lienzo sino que además está creando automáticamente toda la lógica para que después el usuario pueda introducir comandos en la terminal y generar los resultados acorde. En la figura (11) se muestra como se introduce un comando en la terminal para realizar la simulación. Una vez se introduce y envía el comando, no es posible volver a escribir otro comando hasta que aparezca el resultado del mismo por la terminal.

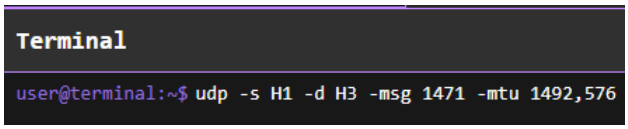
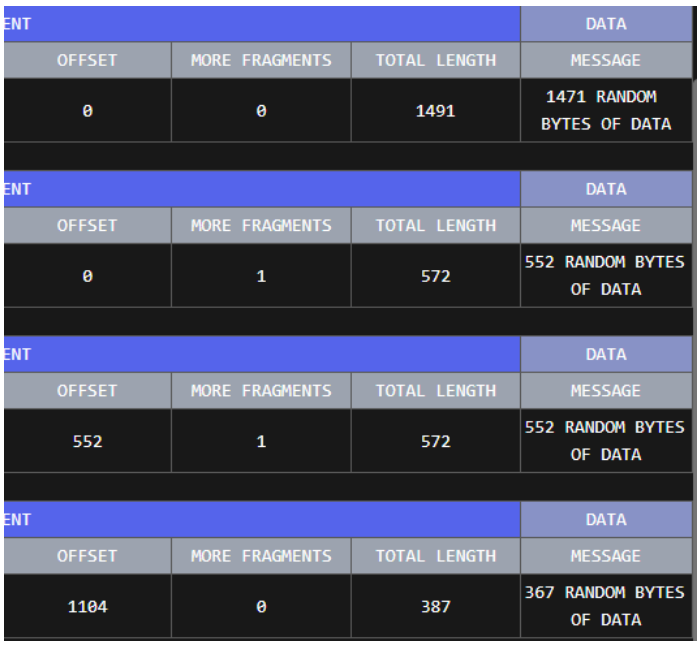


Fig. 12: Comando udp de ejemplo

En este caso el comando utilizado es 'udp -s H1 -eth UP -d H2 -eth UP -msg 1471 -mtu 1492,576' el cual indica que el usuario quiere enviar con el protocolo udp una cierta cantidad de bytes desde el host H1 al host H2 a través de unas redes que fragmentarán el datagrama.

Una vez ejecutado el comando, en el propio terminal aparece el resultado en forma de tablas. Para cada datagrama enviado, aparece una tabla donde se muestra cómo se ha fragmentado cada datagrama, por dónde se ha enviado y una gran variedad de información útil. Por supuesto, los resultados están simplificados para un mayor entendimiento del alumnado, así que no aparecen todas las cabeceras que aparecerían si se usara un programa para capturar paquetes de verdad. En la figura (13) se muestra una pequeña porción del resultado. En este resultado se puede observar el segmento udp que tiene las cabeceras donde se indican atributos como el offset, la longitud total del datagrama, etc.

Para finalizar con el ejemplo, el usuario puede guardar la topología creada con el comando 'save', así, si este cierra el navegador no pierde el estado del proyecto, ya que este se guarda en la base de datos.



ENT			DATA
OFFSET	MORE FRAGMENTS	TOTAL LENGTH	MESSAGE
0	0	1491	1471 RANDOM BYTES OF DATA

ENT			DATA
OFFSET	MORE FRAGMENTS	TOTAL LENGTH	MESSAGE
0	1	572	552 RANDOM BYTES OF DATA

ENT			DATA
OFFSET	MORE FRAGMENTS	TOTAL LENGTH	MESSAGE
552	1	572	552 RANDOM BYTES OF DATA

ENT			DATA
OFFSET	MORE FRAGMENTS	TOTAL LENGTH	MESSAGE
1104	0	387	367 RANDOM BYTES OF DATA

Fig. 13: Reultados de ejecución de comando udp

4.1. Conclusión de los resultados

No ha sido un ejercicio sencillo, sin embargo, ha resultado ser un desafío muy provechoso. No solo se han aprendido tecnologías nuevas, sino que además se ha mejorado la gestión de recursos a la hora de diseñar un sistema de software.

Los resultados del proyecto son bastante asequibles para un alumno de Ingeniería Informática, ya que a lo largo de todo el proceso de desarrollo se han aplicado casi en su totalidad conceptos que se han cursado y que no resultan desconocidos para ningún estudiante. La gestión de proyectos, el desarrollo con tecnologías web, el uso de algoritmos para encontrar caminos en un grafo, conceptos de redes, de algoritmia y de programación son algunos de los ejemplos de conceptos cursados en Ingeniería y que se han usado en este proyecto.

5 Conclusiones

En líneas generales, el proyecto ha seguido su curso y se han conseguido buenos resultados. El prototipo final es bastante completo y fiel a lo que se deseaba en un principio. Es una herramienta capaz de generar topologías básicas y basándose en eso simular de forma simple y adecuada para alumnos los diferentes conceptos básicos de redes, mostrando resultados de los mismos de forma comprensible y ordenada.

Si bien se han cumplido las expectativas iniciales, es cierto que se esperaba que las funcionalidades gráficas consumieran menos tiempo de desarrollo y fueran menos complejas de implementar. Estas funcionalidades gráficas se han convertido en funcionalidades basadas en una consola de comandos, que al final ha resultado ser una aproximación de la solución más satisfactoria, ya que ha resultado mucho menos complejo de implementar y resultará mucho más sencillo de utilizar para los alumnos, además es mucho más modular con lo cual tanto el autor del programa como el alumnado pueden añadir varios protocolos sin complicar

mucho el programa.

Para un futuro se han quedado muchas ideas en el tintero, así que se han introducido las funcionalidades más necesarias, dejando así funcionalidades muy útiles, pero costosas en tiempo para más tarde. Por ejemplo, una de las funcionalidades que, seguramente, ha entrado dentro del ciclo de desarrollo, es el sistema de filtros. Este sistema de filtros constaría de varias opciones para mostrar u ocultar las tramas dependiendo de dónde y cuándo se hayan creado. También sería esencial tener un sistema que permitiera a los alumnos quitar capas de abstracción en el caso de que estos ya dominen los conceptos más básicos y quieran generar resultados más realistas.

En general ha sido un buen ejercicio para probar y aprender nuevas tecnologías y aprender sobre cómo organizar un proyecto de forma realista y gestionar de forma eficiente el tiempo del que se dispone, acorde a las tareas que se quieren realizar y sobre todo acorde de la experiencia que se tiene de las tecnologías utilizadas.

AGRADECIMIENTOS

Primero, quiero agradecer a mi tutor Kevin Chow por guiarme en este proyecto desde sus inicios. También me gustaría agradecer a todos los profesores de ingeniería informática por su compromiso y dedicación en la enseñanza. Sus enseñanzas y materiales han sido esenciales para el desarrollo del proyecto.

Por último quiero expresar mis agradecimientos a mis amigos y familiares quienes me han apoyado en todo momento y cuyas palabras me han ayudado a seguir adelante.

REFERENCIAS

[1] Cisco packet tracer - *networking simulation tool (2022)*. https://www.netacad.com/es/courses/packet-tracer.

[2] Getting started with GNS3 - *GNS3 documentation*. https://docs.gns3.com/docs/.

[3] CS4G Netsim - *Simulator game*. https://netsim.erinn.io/.

[4] NetSim™ Network Simulator™ - *Router Simulator*. https://www.boson.com/netsim-cisco-network-simulator.

[5] Frank K.Y. Chan et al. (2008) Acceptance of Agile Methodologies: A Critical Review and conceptual framework, Decision Support Systems. North-Holland.

[6] M. O. Ahmad, J. Markkula and M. Oivo, "Kanban in software development: A systematic literature review,"2013 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander, Spain, 2013, pp. 9-16, doi: 10.1109/SEAA.2013.28.

[7] ¿Qué es devops? *Explicación de DevOps: Microsoft Azure* https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops.

[8] Tsubame *GitHub repository* https://github.com/gonzalo-garcian/tsubame.

[9] CloudFlare *Pages* https://pages.cloudflare.com/.

[10] Tsubame *Network Simulation* https://develop.tsubame.pages.dev/sandbox.

[11] Supabase *Supabase PostgreSQL Auth Service* https://supabase.com/docs/guides/auth.

[12] Vue *Progressive Javascript Framework* https://vuejs.org/api/.

[13] Tailwind *CSS Framework* https://tailwindcss.com/docs/installation.

[14] Vite *Build Vue* https://vitejs.dev/guide/.

[15] Pinia *Vue store* https://pinia.vuejs.org/.

[16] Konva *Drag and Drop* https://konvajs.org/docs/drag_and_drop/Drag_and_Drop.html.

[17] Konva *Select and transform shapes* https://konvajs.org/docs/select_and_transform/Basic_demo.html.

[18] Konva *Zoom relative to pointer* https://konvajs.org/docs/sandbox/Zooming_Relative_To_Pointer.html.

[19] Konva *Drag Stage* https://konvajs.org/docs/drag_and_drop/Drag_a_Stage.html.

[20] Vue Routing *SPA Routing* https://vuejs.org/guide/scaling-up/routing.html#official-router.

[21] ClickUp *Ímplementación Kanban* https://app.clickup.com/9008141662/v/s/90080255102.

[22] Vitest *Unit Testing with Vue* https://vitest.dev/api/.

[23] Cypress *E2E Test* https://docs.cypress.io/api/.