Trabajo de Fin de Grado

Visor de grafos en procesos de ensamblaje de novo de genomas, integrable en cuadros de mando

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Emilio Martel Díaz

TUTORIZADO POR:
Mª Dolores Afonso Suárez
Daniel Reyes Parrilla

Fecha [06/2025]

ULPGC Universidad de Las Palmas de Gran Canaria | EII ESCUELA DE INGENIERÍA INFORMÁTICA

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D./Dª Emilio Martel Díaz, autor del trabajo Visor de grafos en procesos de ensamblaje de novo de genomas, integrable en cuadros de mando, correspondiente a la titulación Grado en Ingeniería Informática, en colaboración con el Instituto Tecnológico de Canarias, Jardín Botánico Canario Viera y Clavijo / Gestión Evolutiva de la Diversidad Vegetal Terrestre Endémica de la Macaronesia (NextGenDem).

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[X] se autoriza / [ ] no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).

[X] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/La estudiante

Fdo._____

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa:(firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)	Negativamente (justificación en TFT05)
Fdo._____	Fdo._____

DIRECTORA DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

Quienes construyen sueños en el anonimato merecen más que palabras: merecen nuestro reconocimiento eterno, y hoy mi voz se alza en su honor.

Finalmente llega la última parte de este TFT. Lo más importante ahora es agradecer y plasmar por escrito la fuente de motivación, energía y fuerza que me han acompañado hasta el día de hoy.

A mi madre y a mi hermana, por el amor incondicional y la comprensión tras recorrer juntos duros obstáculos. Yo solo soy el resultado de una causa aún mayor: ustedes, que son el motor y el combustible que me hacen seguir, independientemente del terreno, del momento y del lugar.

A mi padre, por enseñarme el significado del sacrificio. Aunque pasen los años y los inviernos se acumulen en nuestros hombros, sigues esforzándote para darme siempre lo mejor.

Después de un año realmente duro, con victorias acompañadas de golpes que me obligaron a evolucionar, quiero agradecer especialmente a mi novia. Gracias por darme la mano y saber decirme exactamente lo que necesitaba en cada momento: ese pasito más que me permitió aguantar cualquier obstáculo que se cruzara en mi camino.

A mis amigos y compañeros de la universidad, por todos esos *carrileos* en las distintas asignaturas, las horas de programación hasta el amanecer, las bromas para aliviar el estrés y la complicidad que nos hizo salir victoriosos de cualquier reto.

Por mucho que intente explayarme, nunca será suficiente para agradecerles todo lo que me han aportado. Incluso esas situaciones difíciles que a veces quisiéramos olvidar: gracias a que las vivimos y superamos, somos quienes somos.

Quiero dejar constancia de que esta experiencia no se olvidará jamás. ¿Cómo?, se preguntarán. Pues alzando mi voz por todos ustedes y dejándolo por escrito para la eternidad.

Los llevo siempre conmigo, y con la mano en el corazón grito al cielo:

Por ustedes y por los que ya no están, este logro nos pertenece.

Infinitamente gracias.

Agradecimientos Académicos

Se agradece de forma especial a Daniel Reyes Parrilla, Mª Dolores Afonso Suárez y Juli Caujapé Castells; al equipo del Jardín Botánico Viera y Clavijo —Unidad Asociada al CSIC—; al Instituto Tecnológico de Canarias; y a todas las personas que han aportado su tiempo y conocimiento a este proyecto.

A Daniel, por su trato cercano y su disposición constante. Desde el primer contacto supo transmitir confianza y acompañamiento, y aunque el ritmo de trabajo impidiera más encuentros, cada conversación dejó un aprendizaje valioso.

A Marilola, por su compromiso, su capacidad de escucha y su acompañamiento pedagógico. Su manera de enseñar, siempre desde el respeto y la exigencia constructiva, ha sido un impulso fundamental en el desarrollo de este trabajo.

A Juli Caujapé Castells y al equipo del Jardín Botánico, por abrir las puertas a los estudiantes y permitirles formar parte de un entorno científico de primer nivel, donde aprender y aportar van de la mano.

Al Instituto Tecnológico de Canarias, por proporcionar el soporte técnico y metodológico necesario, y por promover activamente la colaboración con la universidad para el desarrollo de soluciones aplicadas al entorno real.

Y al resto de participantes, gracias por vuestra dedicación, vuestro entusiasmo y por cada gesto de colaboración que hizo posible este trabajo.

Expreso mi más sincero reconocimiento a la Universidad de Las Palmas de Gran Canaria, por facilitarme los recursos y el entorno académico que han sido esenciales para el desarrollo de este proyecto.

Y, por último, a Romen, con quien he compartido experiencias desarrollando este trabajo, y a mis compañeros y compañeras de promoción, cuyo espíritu de camaradería y apoyo mutuo ha hecho de estos años una experiencia inolvidable.

Resumen

En el proyecto Nextgendem, los investigadores del ITC y sus colaboradores utilizaban herramientas de escritorio como Bandage para visualizar grafos de ensamblaje de novo de genomas. Esto limitaba el acceso a la aplicación y complicaba la colaboración remota. Para resolverlo, se desarrolló un componente web que carga un fichero `.gfa`, lo envía a un servicio Dockerizado que genera un archivo `.layout` con coordenadas y metadatos, y devuelve un JSON al navegador. El visor web procesa estos datos para mostrar el grafo de forma interactiva (zoom, desplazamiento y filtros básicos), sin necesidad de instalar software en local. El backend orquesta las llamadas al microservicio y expone una API REST, mientras que el frontend renderiza el grafo en Angular. Todo el sistema se despliega con Docker Compose, garantizando entornos reproducibles. La solución se validó con el ITC, demostrando que cumple el objetivo de ofrecer un acceso sencillo y colaborativo a los datos genómicos visualizados como grafos.

Esta iniciativa se realiza en colaboración con el Jardín Botánico Viera y Clavijo, unidad asociada del CSIC (JBVCUACSIC), que actúa como Product Owner científico de los productos desarrollados en el marco del proyecto NextGenDem del ITC. El ITC propone la tecnología y las estructuras metodológicas, y la ULPGC ha concretado e implementado dos tecnologías clave que ahora se integrarán en NextGenDem.

Abstract

Within the Nextgendem project, ITC researchers and partners relied on desktop tools like Bandage to visualize de novo genome assembly graphs. This approach restricted access and hindered remote collaboration. To address these drawbacks, a web component was built that uploads a `.gfa` file to a Dockerized service, which produces a `.layout` file containing node coordinates and metadata, then returns JSON to the browser. The web viewer uses this data to render an interactive graph (zoom, pan, and basic filters) without local software installation. The backend coordinates microservice calls and provides a REST API, while the frontend renders the graph in Angular. Docker Compose orchestrates the entire system for reproducible environments. Validation with the ITC confirmed that the solution achieves the goal of offering simple, collaborative access to genomics data visualized as graphs.

This work is carried out in collaboration with the Jardín Botánico Viera y Clavijo, CSIC-associated unit (JBVCUACSIC), which acts as the scientific Product Owner of the products developed within the scope of ITC's NextGenDem project. ITC provides the technological framework and methodologies, while ULPGC has defined and implemented two key technologies that will now be integrated into NextGenDem.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

> "Los datos no son datos hasta que se analizan."
>
> Ben Fry

El estudio de los **datos genómicos** es fundamental para avanzar en disciplinas tan diversas como la conservación de la biodiversidad, la biomedicina y la agricultura. Procesar secuencias de ADN permite identificar variabilidad genética, detectar mutaciones asociadas a enfermedades y comprender los mecanismos evolutivos de las especies. En particular, en el Archipiélago Canario existen numerosas especies endémicas cuya conservación depende de proyectos que integren datos geográficos y genéticos.

En este contexto surge **Nextgendem** [23], una iniciativa que agrupa a varias entidades, entre ellas el **Instituto Tecnológico de Canarias**, el **Jardín Botánico Viera y Clavijo** y la empresa **GesPlan**, para aplicar técnicas de análisis genómico y cartografía en favor de la protección de la flora autóctona. Estas instituciones aportan, respectivamente, la capacidad tecnológica, el acervo biológico y la gestión territorial, constituyendo el núcleo de Nextgendem. Su objetivo principal es organizar, analizar y aplicar el conocimiento científico más relevante para guiar actuaciones ïn situƶ .[ex] situ"que mejoren el estado de conservación de las floras de Gran Canaria (Islas Canarias) y Santiago (Cabo Verde), aunque se prevé incorporar otros territorios macaronésicos en futuros proyectos.

En este trabajo se presenta la creación de un componente web para la visualización de **grafos** (estructuras de nodos y aristas que representan relaciones entre fragmentos de ADN) en procesos de **ensamblaje de novo de genomas** (reconstrucción de la secuencia completa de un genoma sin emplear una referencia previa). Dicho componente, integrable en **cuadros de mando** (paneles interactivos de control y visualización de datos), carga un grafo almacenado en un fichero `.gfa` (*Graphical Fragment Assembly*, formato estándar para describir nodos y conexiones en ensamblajes de secuencias) y lo envía a un servidor, donde se procesa para generar un archivo `.layout` (fichero con coordenadas y atributos de cada nodo), que luego la aplicación web utiliza para:

1. Recuperar la información completa del grafo (número de nodos, aristas, metadatos).
2. Mostrarlo gráficamente en el navegador, permitiendo al usuario navegar, acercar/alejar (*zoom*) y desplazar la vista (*pan*).
3. Consultar y modificar propiedades de los nodos (anchura, color, etiquetas) y exportar el resultado final en formatos estándar (por ejemplo, `.layout` o imagen SVG).

De este modo, se aporta a Nextgendem una solución accesible desde cualquier navegador, sin necesidad de instalar software de escritorio, que facilita el análisis colaborativo de datos genómicos en el ITC y sus entidades asociadas.

1.1. Contexto y motivación

1.1.1. Plataforma Nextgendem

Nextgendem es una plataforma bioinformática diseñada para el análisis científico de datos geográficos, genéticos y ecológicos de especies endémicas de la biodiversidad vegetal terrestre de la Macaronesia. Ha sido desarrollada por un equipo multidisciplinar del Instituto Tecnológico de Canarias (ITC), el Jardín Botánico Canario 'Viera y Clavijo' y Gesplan. Su objetivo es facilitar el acceso y tratamiento de datos para analizar la diversidad filogenética a escala geográfica y aplicarla a la conservación de especies en territorios insulares como Canarias y Cabo Verde.[17]

1.1.2. Efectos del cambio climático en la biodiversidad

Investigadores de la ULPGC y la Universidad de La Laguna (ULL) han presentado estudios sobre las consecuencias del cambio climático en Canarias, dentro del Proyecto CanBIO, financiado por el Gobierno de Canarias y la Fundación Loro Parque[28]. Este proyecto analiza el impacto del cambio climático en especies vegetales amenazadas del archipiélago e incidiendo en cómo el impacto humano afecta negativamente a las especies.

1.1.3. Secuenciación de genomas

La secuenciación de genomas consiste en descifrar el orden de los nucleótidos que conforman el ADN de un organismo con el fin de estudiar su estructura genética, identificar variantes y profundizar en los procesos evolutivos. Entre sus aplicaciones más relevantes se encuentran la conservación de la biodiversidad, al permitir detectar y evaluar la variabilidad genética de poblaciones para proteger especies en peligro; la biomedicina, gracias a la identificación de mutaciones vinculadas a enfermedades; y la agricultura y la ecología, al aportar información sobre la capacidad de adaptación de distintos organismos a su entorno. En este contexto, el Instituto Tecnológico de Canarias (ITC) colabora en proyectos como Nextgendem, en el que realizó una plataforma diseñada para analizar datos genéticos y geográficos de las plantas endémicas de la Macaronesia, que facilita la toma de decisiones para preservar la flora canaria y macaronésica, frente a los efectos del cambio climático.[20]

1.1.4. Papel del visor de grafos en el proceso de secuenciación de genomas

La lectura de genomas se hace leyendo varios duplicados del mismo y solapando estas lecturas para crear la secuencia completa final. Esto es así porque la tecnología actual no permite leer el genoma completo con precisión por lo que se tiene que dividir en fragmentos. En este contexto, un *contig* se define como una secuencia lineal de nucleótidos obtenida al concatenar lecturas que presentan solapamientos consistentes y no ambiguos, de modo que constituye el tramo continuo más largo que puede inferirse sin lagunas. Un *repeat* es un segmento de ADN cuya secuencia se encuentra representada múltiples veces, de forma idéntica o casi idéntica, en distintas posiciones del genoma; esta redundancia introduce ambigüedades que dificultan la correcta asignación de lecturas y la estimación del número real de copias.

La visualización del grafo de ensamblaje, en el que los nodos representan *contigs* y las aristas sus posibles uniones, resulta imprescindible porque permite identificar de manera explícita las ambigüedades provocadas por dichos *repeats*. La edición interactiva de este grafo, eliminando conexiones espurias o resolviendo bifurcaciones con apoyo de datos adicionales, facilita la depuración del ensamblaje y la generación de una representación más fiel y completa del genoma estudiado. [18]

1.2. Organización del documento

El resto del documento se organiza de la siguiente manera:

- ✔ **Estado actual y objetivos iniciales.** Recoge el estado del arte de la materia en cuestión y establece los objetivos específicos que guían el desarrollo del trabajo.
- ✔ **Aportaciones del trabajo.** Detalla las competencias adquiridas, las contribuciones realizadas y su alineamiento con los Objetivos de Desarrollo Sostenible (ODS).
- ✔ **Desarrollo.** Expone el diseño, la arquitectura y la implementación de la solución propuesta, incluyendo esquemas, diagramas y decisiones técnicas relevantes.
- ✔ **Resultados.** Presenta los resultados obtenidos durante las pruebas o experimentos, analiza su validez y compara el desempeño con los objetivos planteados.
- ✔ **Conclusiones y trabajo futuro.** Resume las conclusiones extraídas del TFT, valora el cumplimiento de los objetivos y propone posibles líneas de investigación o mejoras a futuro.
- ✔ **Otros capítulos y anexos.** Incluye el glosario de términos, la bibliografía y cualquier material complementario (anexos, documentación adicional, etc.).

Capítulo 2

Estado actual y objetivos iniciales

2.1. Motivación y antecedentes

En las últimas décadas, el análisis de **datos genómicos** (conjunto de secuencias de ADN que contienen información sobre la estructura y función de los genes) se ha convertido en una herramienta clave para la biomedicina, la conservación de la biodiversidad y la agricultura. Procesar secuencias de ADN permite, entre otros objetivos, identificar variabilidad genética, detectar mutaciones asociadas a enfermedades o comprender procesos evolutivos. En el Archipiélago Canario, la protección de especies endémicas requiere integrar datos geográficos y genéticos para diseñar estrategias de conservación más efectivas.

Con este propósito se constituyó **Nextgendem** (plataforma de bioinformática y georreferenciación orientada a especies endémicas de la Macaronesia), que agrupa a varias entidades:

- ✔ **Instituto Tecnológico de Canarias (ITC)**: centro de I+D+i especializado en soluciones tecnológicas para el Archipiélago, con experiencia en proyectos Interreg MAC.
- ✔ **Jardín Botánico Viera y Clavijo**: Liderazgo del proyecto, colección científica de flora canaria, encargada de aportar muestras y conocimiento taxonómico.
- ✔ **GesPlan**: consultora en gestión de espacios naturales, responsable de integrar la información genética en planes de conservación territorial.

Nextgendem se centra en dos conceptos básicos:

- ✔ **Ensamblaje de novo**: reconstrucción de un genoma completo a partir de fragmentos de ADN, sin usar un genoma de referencia.
- ✔ **Pangenoma**: conjunto de todas las secuencias y variantes genómicas observadas en múltiples individuos o cepas de una misma especie.

Hasta ahora, los ensamblajes y pangenomas en Nextgendem se han visualizado principalmente con herramientas de escritorio (por ejemplo, Bandage [30]), que muestran nodos y

conexiones, pero requieren instalación local. Sobre esta base surge la necesidad de un visor *web* que:

- ✔ Sea accesible desde cualquier navegador, sin instalaciones previas.
- ✔ Permita integrar la visualización de grafos en cuadros de mando corporativos del ITC.
- ✔ Ofrezca filtros y esquemas de color dinámicos basados en metadatos genómicos.

2.1.1. Estado de visores en Nextgendem

Hasta la fecha se ha utilizado Bandage [31] (aplicación de escritorio) para:

- ✔ Cargar archivos `.gfa` (formato *Graphical Fragment Assembly*) generados por ensambladores como SPAdes o Velvet.
- ✔ Aplicar algoritmos de *layout* (por ejemplo, force-directed) que calculan coordenadas (x, y) para cada nodo.
- ✔ Visualizar nodos y aristas con anotaciones de cobertura, longitud y resultados de BLAST.
- ✔ Exportar un fichero `.layout` con las coordenadas para su uso en informes.

Este flujo, aunque potente, presenta dos limitaciones:

1. Depende de la instalación local de la aplicación (Qt, bibliotecas de parsing GFA).
2. No se integra fácilmente en cuadros de mando web ni facilita la colaboración remota.

Por tanto, la propuesta de este TFT es crear un componente web que reproduzca (y amplíe) esas funcionalidades sin requerir software de escritorio.

2.2. Visores web de grafos genómicos

Los **visores web de grafos genómicos** permiten explorar ensamblajes y pangenomas directamente en el navegador. A continuación se describen algunas de las opciones más representativas, con sus ventajas y limitaciones:

2.2.1. Visores de variantes genómicas

- ✔ **SequenceTubeMap[29]:**
  - *Pros:*
    - Representa el grafo como un “mapa de metro”, facilitando la comparación de variantes múltiples.

    - Interactividad avanzada (zoom, selección de rutas, resaltado de nodos) gracias a D3.js.
    - Permite superponer anotaciones (genes, SNPs) directamente sobre los tubos.
  - *Contras:*
    - Curva de aprendizaje empinada para usuarios no familiarizados con gráficos genómicos tubulares.
    - Depende del rendimiento del navegador; en grafos muy grandes puede ralentizarse.
    - Requiere preprocesar datos GFA/rGFA para extraer rutas, añadiendo pasos al pipeline.

✔ **MoMI-G[19]:**

  - *Pros:*
    - Visualiza variantes estructurales (inserciones, deleciones, inversiones) en vistas lineal y en grafo.
    - Filtrado avanzado de regiones genómicas y anotaciones en tiempo real (VCF, GFF/GTF, BAM).
  - *Contras:*
    - Consume recursos importantes en grafos grandes (pangenomas, ensamblajes metagenómicos).
    - Configuración inicial compleja (GFA + VCF + BAM).

✔ **JBrowse 2 con JBrowse Web[16]:**

  - *Pros:*
    - Arquitectura modular basada en plugins.
    - Renderizado rápido con Canvas/WebGL, incluso en grafos de decenas de miles de nodos.
    - Compatible con formatos BAM, VCF, GTF y rutas de pangenoma en rGFA.
  - *Contras:*
    - Requiere servidor Node.js y configuración de índices genómicos.
    - Personalizar la visualización de GFA puede ser complejo para usuarios sin experiencia en la API de JBrowse 2.

2.2.2. Visores alternativos a Bandage

✔ **PanGraphViewer Web[26]:**

- *Pros:*
  - Basado en Python/Django + Bokeh, ideal para entornos bioinformáticos.
  - Paneles de control para filtrar nodos por longitud, cobertura o atributos personalizados.
- *Contras:*
  - Despliegue complejo (servidor Django, dependencias Bokeh/Tornado).
  - Interactividad limitada en pangenomas muy grandes.

✔ **VRPG[2]:**

- *Pros:*
  - Basado en D3.js, soporta pangenomas completos con rutas y variantes anidadas.
  - Renderizado optimizado con Canvas/WebGL para equipos modernos.
- *Contras:*
  - Requiere preprocesar y generar índices VG (XG, GBWT).
  - Curva de aprendizaje pronunciada (línea de comandos de VG, API de `vg.js`).

En conclusión, estos visores web ofrecen distintos equilibrios entre interactividad, rendimiento y facilidad de despliegue. A continuación, se definen los objetivos de este trabajo, que busca cubrir las limitaciones detectadas.

2.3. Objetivo principal

Este proyecto tiene como objetivo desarrollar un **componente web** para la visualización de grafos de ensamblaje de novo de genomas. Ésta es una técnica utilizada en bioinformática para representar y analizar el proceso de ensamblaje de secuencias de ADN sin usar un genoma de referencia. Su propósito es:

- ✔ Integrarse en **cuadros de mando** (paneles interactivos de control) del ITC.
- ✔ Cargar un archivo `.gfa` (*Graphical Fragment Assembly*) y enviarlo a un backend que:
  - Aplique un algoritmo de *layout* (force-directed) para calcular coordenadas (x, y) de cada nodo.
  - Genere un archivo `.layout` con la estructura (nodos, aristas y metadatos) lista para visualizar en la web.
- ✔ Renderizar interactivamente el grafo en el navegador, permitiendo navegación, zoom, desplazamiento y consulta de metadatos.
- ✔ Facilitar el filtrado y la personalización de propiedades (anchura, color, etiquetas) de los nodos.

2.3.1. Objetivos específicos

1. Implementar una interfaz gráfica interactiva y amigable para visualizar y manipular grafos de ensamblaje de genomas con Angular.
2. Desarrollar un backend que procese la estructura de los grafos y devuelva un `.layout` para su representación web.
3. Validar la funcionalidad del componente mediante demostraciones con el equipo del ITC y optimizar su rendimiento en escenarios reales de uso.

Capítulo 3

Aportaciones del trabajo

3.1. Principales aportaciones

El presente Trabajo de Fin de Título aborda un desafío clave en el ámbito de la bioinformática: **la visualización de grafos generados en procesos de ensamblaje de novo de genomas**.

La implementación de un componente web interactivo basado en las funcionalidades de la herramienta Bandage permitirá la integración de esta visualización en cuadros de mando, facilitando la interpretación de datos complejos por parte de investigadores y expertos en biología computacional.

Las principales contribuciones de este trabajo pueden agruparse en tres ámbitos: técnico, científico y socioeconómico.

3.1.1. Aportaciones técnicas

Desde el punto de vista tecnológico, el desarrollo de un visor web para grafos de ensamblaje de novo de genomas representa un avance significativo en la accesibilidad y usabilidad de herramientas bioinformáticas. La implementación de tecnologías modernas, como Angular para el frontend y Node.js para el backend, garantizará tiempos de respuesta adecuados y compatibilidad con entornos de investigación avanzada. Los aspectos clave incluyen:

- ✔ **Interfaz gráfica interactiva**: Permite la visualización y manipulación intuitiva de estructuras de grafos, optimizando el análisis de datos genómicos.
- ✔ **Automatización en la organización de grafos**: Un backend especializado procesará la estructura de los grafos y generará una representación visual optimizada, reduciendo la carga cognitiva en la interpretación de resultados.
- ✔ **Adaptabilidad y escalabilidad**: El componente se diseñará con un enfoque modular para facilitar su integración en otros proyectos de bioinformática, maximizando su

utilidad en diferentes contextos de investigación.

3.1.2. Aportaciones científicas

El desarrollo de este visor de grafos contribuirá al avance de la bioinformática y la genética al proporcionar una herramienta accesible para el estudio de organismos sin genoma de referencia. Entre sus aportaciones destacan:

- ✔ **Facilitación de estudios sobre biodiversidad**: La herramienta permitirá a los investigadores analizar la evolución y variabilidad genética de especies vegetales endémicas de la Macaronesia, con aplicaciones directas en biología de conservación.
- ✔ **Optimización de la representación de datos**: La visualización efectiva de grafos facilitará la interpretación de conexiones entre secuencias de ADN, permitiendo un análisis más profundo de patrones genéticos complejos.
- ✔ **Aplicabilidad en estudios evolutivos**: Contribuirá a investigaciones centradas en la evolución de especies vegetales en ecosistemas insulares, ofreciendo datos clave sobre la adaptación genética y la diversidad evolutiva.

3.1.3. Aportaciones socioeconómicas

Más allá del impacto técnico y científico, este trabajo también posee implicaciones en el ámbito socioeconómico y medioambiental:

- ✔ **Apoyo a la conservación de la biodiversidad**: La herramienta será utilizada en el proyecto "NextGenDem"del Instituto Tecnológico de Canarias, facilitando la gestión evolutiva de la diversidad vegetal terrestre endémica de la Macaronesia.
- ✔ **Impulso a la investigación aplicada**: La colaboración con el Jardín Botánico Canario Viera y Clavijo refuerza la conexión entre la bioinformática y la conservación, promoviendo soluciones tecnológicas en el análisis genético.
- ✔ **Fomento del desarrollo tecnológico en Canarias**: Al ser desarrollado en la Universidad de Las Palmas de Gran Canaria, este trabajo aporta a la consolidación de un ecosistema de innovación local, fortaleciendo la presencia de tecnologías avanzadas en el archipiélago.

3.1.4. Repercusión esperada

Se espera que este visor de grafos tenga un impacto positivo tanto en el ámbito académico como en la comunidad investigadora. La disponibilidad de una herramienta accesible para la visualización de datos genómicos contribuirá a mejorar la eficiencia de estudios sobre biodiversidad y adaptación genética, facilitando la integración de análisis computacionales en proyectos de conservación. Asimismo, al emplear tecnologías modernas y escalables, el componente podrá ser adaptado y extendido a otros contextos dentro de la bioinformática y biología molecular.

La combinación de bioinformática, desarrollo de software y conservación ambiental convierte este Trabajo de Fin de Título en una contribución relevante tanto para la ciencia como para la sociedad.

3.2. Competencias específicas

Las **competencias específicas** relacionadas de forma más directa con el trabajo desarrollado, y cómo se han cubierto con este TFT son las indicadas en los cuadros 3.1 y 3.2.

3.3. Alineamiento con los objetivos de desarrollo sostenible

Este Trabajo de Fin de Grado se alinea especialmente con los ODS 3, 9 y 17. En relación con el ODS 3 (Salud y Bienestar), el componente web propuesto permite visualizar grafos de ensamblaje de novo de genomas, lo cual es fundamental en investigaciones biomédicas y de salud pública, facilitando el análisis de ADN de organismos no estudiados anteriormente. Por otro lado, se alinea con el ODS 9 (Industria, Innovación e Infraestructuras) al desarrollar una herramienta tecnológica avanzada basada en Bandage, utilizando Angular y un backend de organización automática para nodos. Finalmente, se vincula estrechamente con el ODS 17 (Alianzas para lograr objetivos) ya que el proyecto parte de una colaboración con el Instituto Tecnológico de Canarias y se enmarca en el programa europeo Interreg MAC, destacando el carácter multidisciplinar y transnacional del trabajo. Esta información está ubicada en el cuadro 3.3.

Cuadro 3.1: Competencias específicas: comunes a la rama de informática [13]

Código	**Descripción**
CI1	Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.
CI2	Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
CI4	Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
CI7	Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema.
CI8	Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
CI16	Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería del software.
CI17	Capacidad para diseñar y evaluar interfaces persona–computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.
CI18	Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Cuadro 3.2: Competencias específicas: tecnologías de la información [13]

Código	**Descripción**
TI1	Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.
TI3	Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

Cuadro 3.3: Grado de relación del TFT con los objetivos de desarrollo sostenible.

ODS	Grado de relación			
	0 No procede	**1** Bajo	**2** Medio	**3** Alto
1 Fin de la Pobreza	X			
2 Hambre cero	X			
3 Salud y Bienestar				X
4 Educación de calidad			X	
5 Igualdad de género	X			
6 Agua limpia y saneamiento	X			
7 Energía Asequible y no contaminante	X			
8 Trabajo decente y crecimiento económico			X	
9 Industria, Innovación e Infraestructuras				X
10 Reducción de las desigualdades	X			
11 Ciudades y comunidades sostenibles		X		
12 Producción y consumo sostenibles	X			
13 Acción por el clima	X			
14 Vida submarina	X			
15 Vida de ecosistemas terrestres	X			
16 Paz, justicia e instituciones sólidas	X			
17 Alianzas para lograr objetivos				X

Capítulo 4

Desarrollo

4.1. Metodologías aplicadas

Para el desarrollo de este TFT se siguió un proceso dividido en fases de corta duración, según lo indicado en la propuesta TFT01[9]. Cada fase atendía a objetivos concretos y se revisaba al finalizar para incorporar feedback continuo. Además, se adoptaron principios de arquitectura limpia, integración y entrega continuas, y contenerización. A continuación se presenta una versión resumida de las prácticas adoptadas.

4.1.1. Iteraciones cortas según TFT01

El desarrollo se estructuró en las siguientes fases, con entregas parciales al final de cada una:

- ✔ **Planificación y Análisis de Requisitos:** definición de funcionalidades esenciales y MVP.
- ✔ **Diseño de Arquitectura e Interfaz:** elaboración de diagramas de capas y bocetos de la UI.
- ✔ **Desarrollo e Implementación:** codificación del frontend, backend y microservicio en contenedores.
- ✔ **Validación:** revisión con el equipo del ITC para confirmar el cumplimiento de requisitos.
- ✔ **Ajustes y correcciones:** adaptación según resultados de la validación y análisis de calidad.

Cada fase tuvo una duración estimada de dos a cuatro semanas, permitiendo evaluar el progreso y priorizar tareas en ciclos cortos.

4.1.2. Clean Architecture

Se adoptaron principios de arquitectura limpia para desacoplar la lógica de negocio de detalles de infraestructura y facilitar la mantenibilidad.

- ✔ **Capas:**
  - *Entidades:* reglas de negocio puras.
  - *Casos de uso:* orquestan operaciones sobre entidades.
  - *Adaptadores:* conectan la lógica con tecnologías externas (controladores, persistencia).
  - *Frameworks y Drivers:* capas exteriores (UI, base de datos, servicios).
- ✔ **Regla de dependencia:** Las capas externas pueden referirse a capas internas, no al revés.
- ✔ **Beneficio:** Alta capacidad de prueba y flexibilidad para cambiar tecnologías sin afectar el núcleo.

4.1.3. SonarQube

Se utilizó la herramienta SonarQube para realizar análisis estáticos del código fuente del proyecto, permitiendo detectar errores potenciales antes de la ejecución. Este análisis incluyó la identificación de vulnerabilidades de seguridad, duplicidad de código y malas prácticas (*code smells*)

- ✔ **Análisis estático:** examina el código sin necesidad de ejecutarlo, identificando posibles fallos lógicos o de estilo.
- ✔ **Vulnerabilidades:** detecta patrones de código inseguros o propensos a fallos.
- ✔ **Mantenimiento:** proporciona una puntuación de mantenibilidad, facilitando el control técnico del proyecto a largo plazo.

4.1.4. Docker y Docker Compose

Para garantizar entornos consistentes, tanto localmente como en producción, se utilizó Docker para contenerizar cada componente y Docker Compose para orquestarlos.

- ✔ **Docker:** empaqueta frontend, backend y microservicio de procesamiento de grafos en contenedores aislados.
- ✔ **Docker Compose:** define servicios y dependencias en un solo fichero, facilitando el despliegue conjunto.

- ✔ **Beneficio:** elimina problemas de "funciona en mi máquina" y permite reproducir la infraestructura con un único comando.

En conjunto, estas prácticas (Fases con iteraciones cortas, Clean Architecture, CI/CD con SonarQube y contenerización) fueron seleccionadas para maximizar la agilidad, la calidad y la escalabilidad del desarrollo, manteniendo un ciclo de retroalimentación rápido y entornos reproducibles.

4.2. Tecnologías involucradas

A continuación se describen las principales tecnologías empleadas en cada uno de los tres repositorios, **visor de grafos** [6], **servidor principal** [8] y el **servidor bandage** [7]

4.2.1. Visor de grafos

- ✔ **Angular v19**: Framework front-end basado en TypeScript para construir la interfaz de usuario.[1]
- ✔ **TypeScript**: Tipado estático en componentes, servicios y modelos de datos.[27]
- ✔ **RxJS**: Manejo de flujos de datos y comunicación asíncrona con el backend mediante observables.[24]
- ✔ **NG-Zorro**: Biblioteca o conjunto de utilidades para componentes visuales y diseño de la aplicación.[21]
- ✔ **D3.js**: Librería para renderizar y organizar visualmente los nodos y aristas del grafo.[3]
- ✔ **Docker**: Empaqueta la aplicación Angular en un contenedor con NGINX.[5]

4.2.2. Servidor principal

- ✔ **Node.js** : Entorno de ejecución para JavaScript en servidor.[22]
- ✔ **Express.js**: Framework minimalista para definir rutas y manejar peticiones REST.[14]
- ✔ **TypeScript**: Tipado estático para rutas, controladores y lógica de orquestación.
- ✔ **SonarQube**: Análisis estático dedicado a detectar *code smells*, duplicación de código y posibles vulnerabilidades de seguridad en el backend.[25]
- ✔ **Docker**: Empaqueta el servidor en un contenedor.

4.2.3. Servidor Bandage

- ✔ **Node.js**: Entorno de ejecución para JavaScript en servidor.
- ✔ **Express.js**: Framework para gestionar rutas y peticiones REST, implementando la lógica de conversión de `.gfa` a `.layout` en JSON. y obtención de informacón del grafo.
- ✔ **SonarQube**: Análisis estático dedicado a detectar *code smells*, duplicación de código y posibles vulnerabilidades de seguridad en este microservicio.
- ✔ **Docker**: Empaqueta el microservicio en un contenedor.

Toda esta infraestructura está orquestada mediante **docker-compose**[4], un archivo de configuración centralizado para lanzar y controlar múltiples servicios al mismo tiempo.

4.3. Fases de desarrollo

En este proyecto se desarrolló en las siguientes fases, logrando cumplir los objetivos pactados en el TFT01.[9]

4.3.1. Planificación y Análisis de Requisitos

Siguiendo la propuesta descrita en el documento TFT01, esta fase inicial se ha centrado en entender y replicar las funciones esenciales de Bandage para definir un MVP incremental. Las tareas principales son:

1. **Familiarización con BandageNG**
   - ✔ Instalar y ejecutar BandageNG.
   - ✔ Cargar varios archivos `.gfa` generados con diferentes ensambladores (SPAdes, Velvet, MEGAHIT) y observar:
     - Cómo Bandage organiza internamente la estructura de grafo (nodos, aristas, metadatos de cobertura y longitud).
     - El algoritmo de layout automático (force-directed) que calcula posiciones (x, y) para cada nodo.
   - ✔ Explorar la interfaz gráfica para identificar cómo se mapean los datos del grafo a los elementos visuales (colores, tamaños de nodos según longitud o cobertura).
2. **Identificación de elementos visuales principales** Tras la exploración, se documentan los componentes que deberán existir en la versión web:
   - ✔ *Lienzo de grafo*: área donde se dibujan nodos (segmentos de secuencia) y aristas (enlaces), con propiedades visuales (color, grosor) que reflejan metadatos.

- ✔ *Panel lateral de filtros*: controles para establecer umbrales de longitud mínima, cobertura mínima y selección de esquemas de color.
- ✔ *Menú de exportación*: opciones para descargar el layout actual en formato `.layout` (coordenadas en JSON) o generar imágenes (SVG/PNG) del grafo.
- ✔ *Indicadores de progreso*: barras de avance o iconos que muestren el estado de operaciones costosas (cálculo de layout, parsing de GFA) y alertas de errores (archivo mal formado).

3. **Definición del flujo funcional** Para reflejar el comportamiento de BandageNG, se establece el siguiente diagrama de flujo simplificado:

   a) *Carga de archivo* `.gfa`: el usuario selecciona un fichero desde su sistema o repositorio.4.2

   b) *Parsing en el microservicio*: el backend invoca al microservicio `bandage` para leer y validar el `.gfa`, generando estructuras de nodos y aristas con metadatos.

   c) *Cálculo de layout*: el microservicio hace uso del cliente de consola del Bandage para hacer uso de un comando, el cual convierte el archivo .gfa a .layout.

   d) *Envío de datos al frontend*: el servidor devuelve el JSON que contiene lista de nodos (con coordenadas y propiedades) y lista de aristas.

   e) *Renderizado interactivo*: el frontend recibe el JSON y utiliza D3.js para dibujar el grafo en un SVG, habilitando zoom, pan y selección de nodos. 4.5 4.7

   f) *Filtrado dinámico*: el usuario ajusta filtros (longitud, cobertura, color), lo que provoca una nueva capa de filtrado en el frontend sin recalcular completamente el layout.4.3 4.4 4.6

4. **Definición del MVP e iteraciones** Con el flujo anterior, se establecen los requisitos mínimos del MVP:

   - ✔ Cargar un archivo `.gfa` y mostrar el grafo con coordenadas generadas en el microservicio.
   - ✔ Permitir zoom in / zoom out en el grafo renderizado.
   - ✔ Exportar el layout actual en formato `.layout` (JSON).

   Tras validar este MVP con el equipo del ITC, se planifica un roadmap iterativo para incorporar en cada sprint (de duración semanal) funcionalidades adicionales como:

   - ✔ Panel de filtros de longitud y cobertura.
   - ✔ Exportación a SVG/PNG del grafo visible.

   Cada iteración culminará con una demostración al equipo del Instituto Tecnológico de Canarias para recibir retroalimentación continua y adaptar prioridades según las necesidades reales.

A continuación veremos la aplicación de Bandage con sus diferentes funcionalidades para migrar en nuestro componente web.

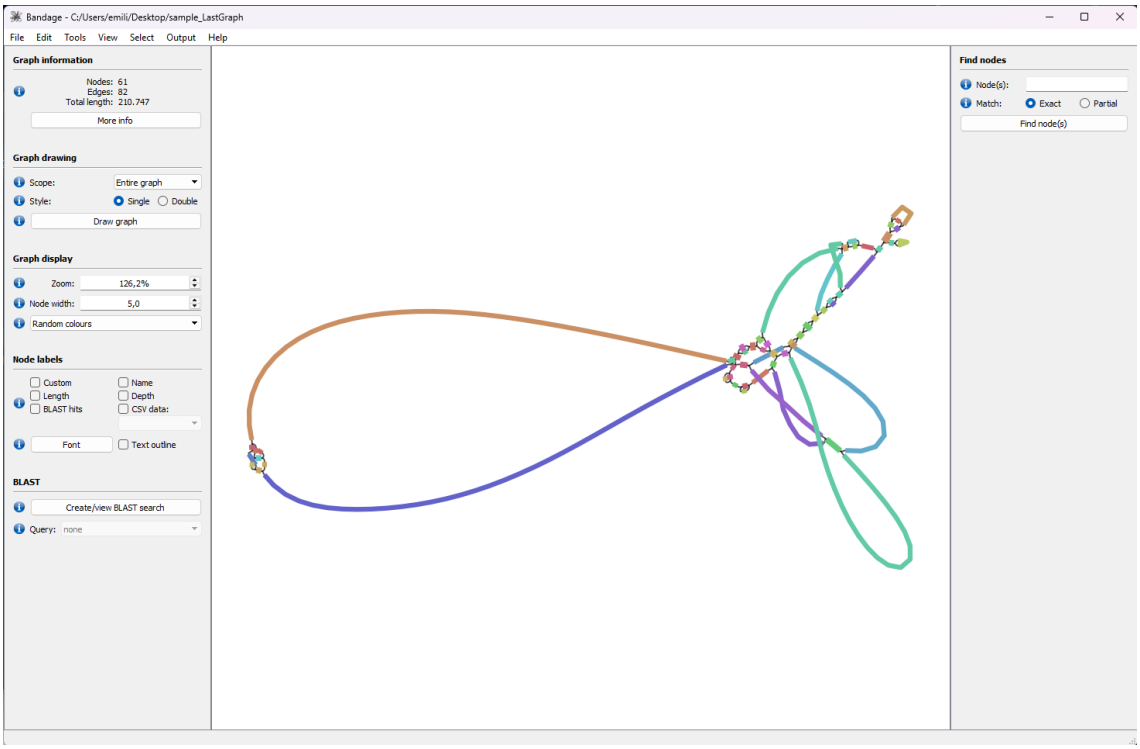

Ilustración 4.1: Vista completa de la aplicación de escritorio Bandage

Como se muestra en la Ilustración 4.1, la interfaz de Bandage tiene una interfaz sencilla pero cuenta con un amplio número de funcionalidades que iremos desglosando, en las cuales nos hemos enfocado en replicar en nuestro proyecto.

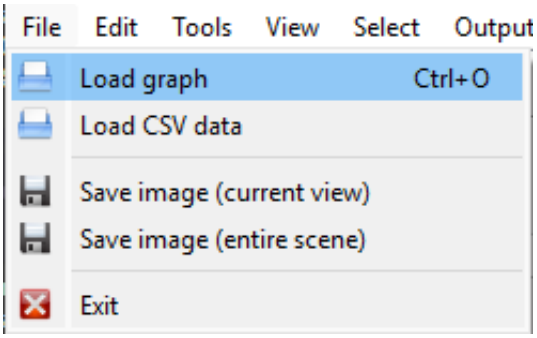

Ilustración 4.2: Detalle de la ventana de carga de archivo `.gfa` en Bandage

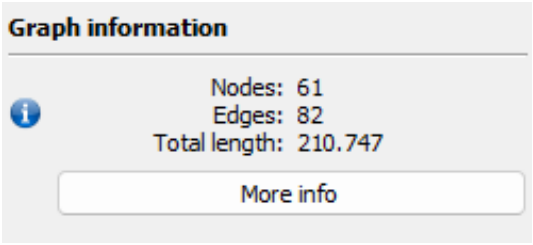

Ilustración 4.3: Detalle del panel de información del grafo en Bandage

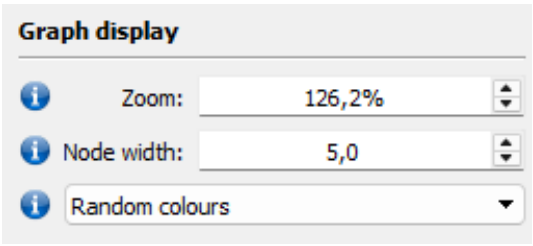

Ilustración 4.4: Detalle de las opciones de visualización del grafo en Bandage

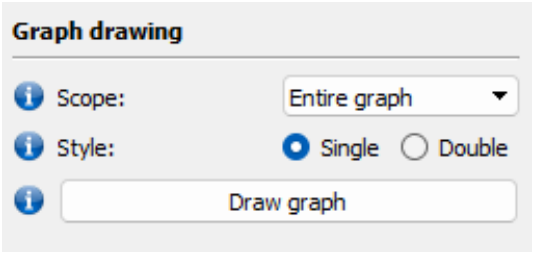

Ilustración 4.5: Detalle de la representación gráfica del grafo en Bandage

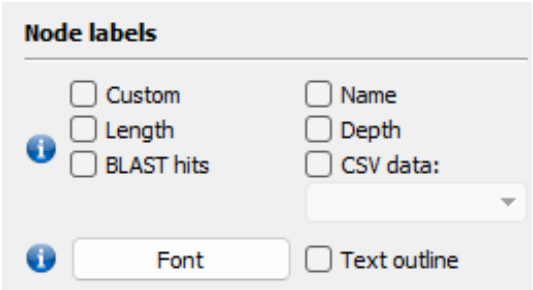

Ilustración 4.6: Detalla de las etiquetas de nodos y metadatos visibles en Bandage

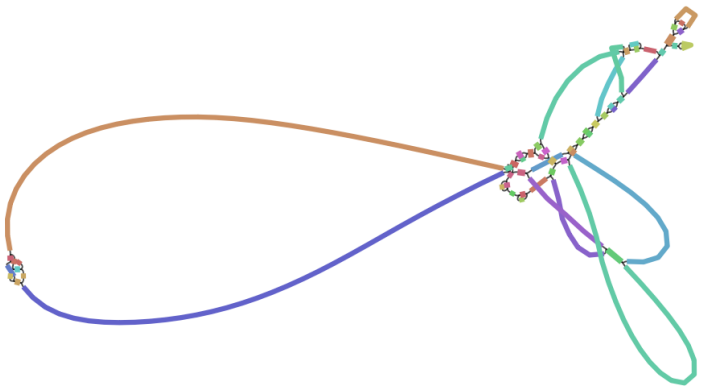

Ilustración 4.7: Detalle de la representación de la visualización en Bandage

4.3.2. Desarrollo e Implementación

A continuación se describe la infraestructura general del proyecto, constituido por tres repositorios principales: `tft-frontend`, `tft-backend` y `bandage`. En el servidor (`tft-backend`) se ha aplicado una Clean Architecture para mantener la lógica de negocio desacoplada de los detalles de infraestructura. En el frontend (`tft-frontend`) se ha modularizado la aplicación utilizando *standalone components* de Angular para facilitar la mantenibilidad y escalabilidad.

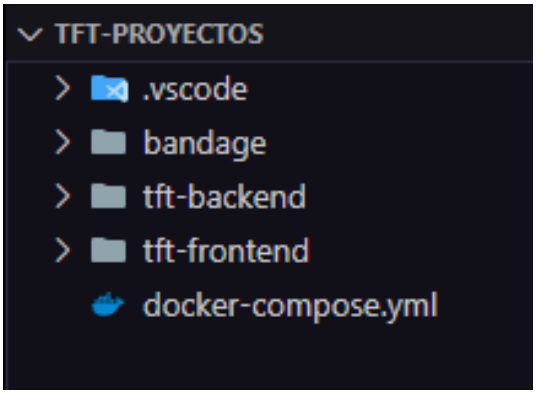

Ilustración 4.8: Detalle de la estructura de carpetas principal del proyecto TFT

La Figura 4.8 muestra el directorio raíz, donde conviven los tres subproyectos y el fichero `docker-compose.yml` encargado de orquestar los contenedores Docker de cada uno.

4.3.3. Frontend: `tft-frontend`

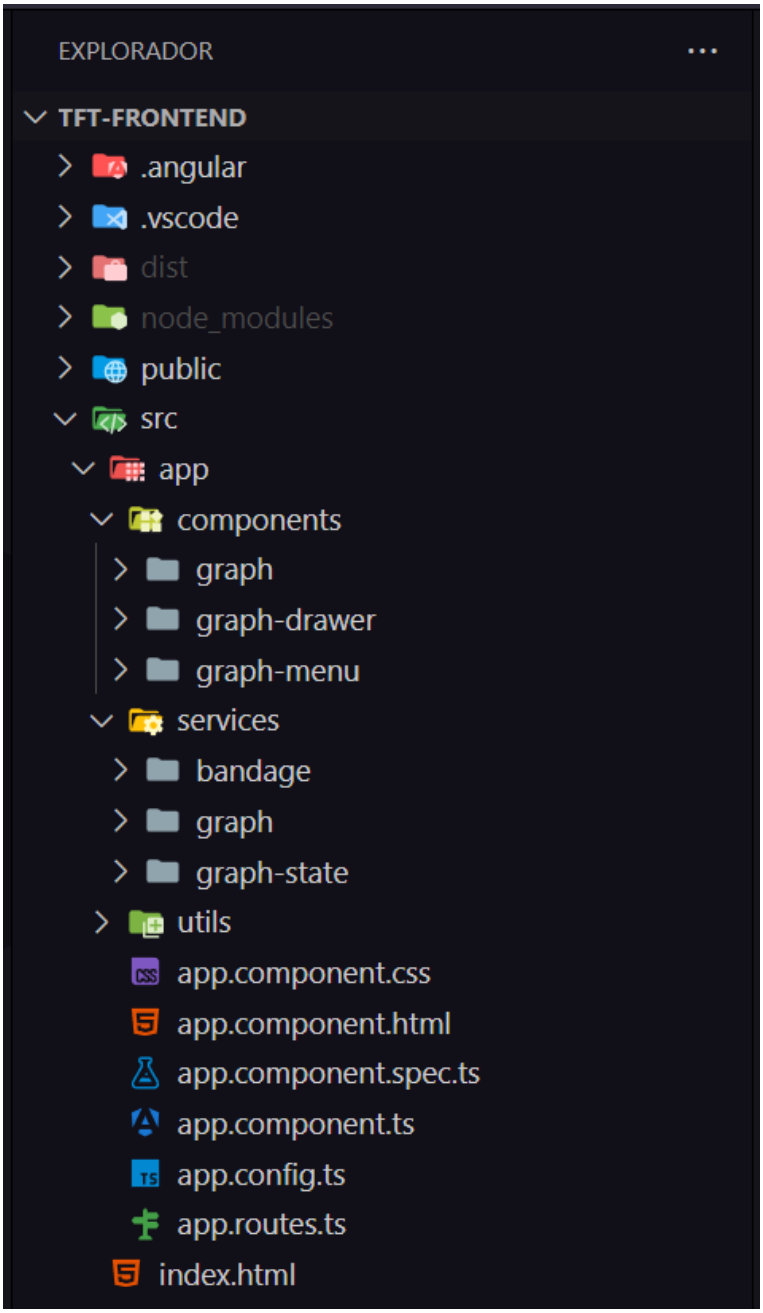

Ilustración 4.9: Estructura interna de `tft-frontend`

En `tft-frontend` (Figura 4.9) se ha organizado la aplicación Angular en *standalone components*:

- ✔ **components/graph**: módulo que agrupa los componentes encargados de la representación estática del grafo.

- ✔ **components/graph-drawer**: componente especializado en dibujar nodos y aristas sobre un `canvas` usando D3.js.
- ✔ **components/graph-menu**: menú lateral que contiene controles de filtros, esquemas de color y exportación.
- ✔ **services/bandage**: servicio que encapsula las llamadas al backend para procesar archivos `.gfa` y obtener el layout.
- ✔ **services/graph, graph-state**: servicios para gestionar el estado global del grafo y sus propiedades (nodos, aristas, metadatos).
- ✔ **utils**: funciones auxiliares reutilizables (por ejemplo, parsers de JSON, transformaciones de datos).
- ✔ `app.config.ts` y `app.routes.ts`: configuración de puntos finales (endpoints) y rutas de navegación de la aplicación.

El uso de los *standalone components*,permite que cada pieza de la UI se declare de forma independiente y luego se importen únicamente en los módulos o rutas que requieran su uso, reduciendo dependencias innecesarias.

4.3.4. Backend: `tft-backend`

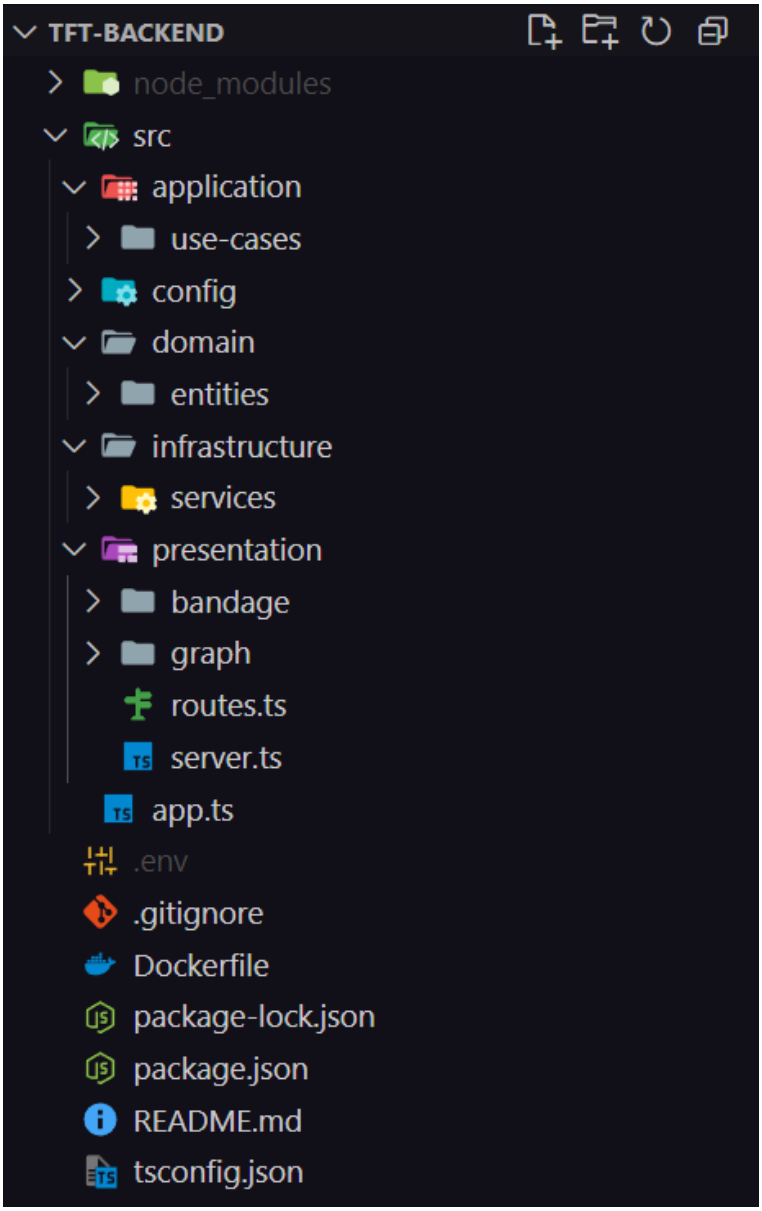

Ilustración 4.10: Estructura interna de `tft-backend`

En `tft-backend` (Figura 4.10) se ha implementado una Clean Architecture que se articula en cuatro capas concéntricas:

- ✔ **domain/entities**: contiene las clases y modelos de dominio puro (*Entities*) que representan nodos y aristas del grafo, sin depender de librerías externas.
- ✔ **application/use-cases**: define los casos de uso (*Interactors*) que implementan la lógica de negocio, tales como: *Convertir GFA a Layout* y *Obtener información del grafo*.

- ✔ **infrastructure/services**: adaptadores que implementan detalles concretos de persistencia o llamadas a microservicios (por ejemplo, cliente HTTP hacia el contenedor `bandage`), cumpliendo las interfaces definidas en el dominio.
- ✔ **presentation/routes.ts y server.ts**: expone los endpoints REST mediante Express.js, mapeando cada ruta a su caso de uso correspondiente.

La separación estricta de responsabilidades permite que la capa de dominio y aplicación no conozcan nada sobre Express, Docker o cualquier otro framework, facilitando pruebas unitarias y futuros cambios tecnológicos.

4.3.5. Microservicio de procesamiento: `bandage`

A continuación se muestra la estructura interna del microservicio `bandage`, donde se puede observar la organización de carpetas y archivos principales:

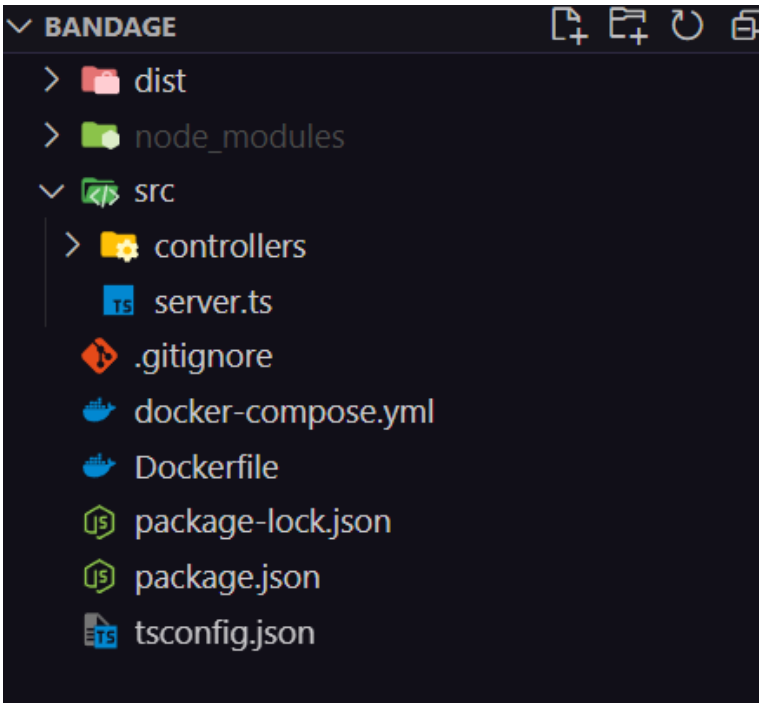

Ilustración 4.11: Estructura interna del microservicio `bandage`

El repositorio `bandage` (Ilustración 4.11) contiene la imagen Docker que agrupa el cliente de consola de BandageNG para realizar las conversiones y extraer información del grafo mediante comandos. Concretamente, este contenedor se encarga de:

- ✔ Recibir el archivo `.gfa` a través de un endpoint REST.
- ✔ Invocar internamente el cliente de consola de Bandage para generar un layout o extraer metadatos del grafo (por ejemplo, posiciones de nodos o estadísticas de cobertura).

- ✔ Recuperar la salida del comando de BandageNG en formato JSON o texto estructurado con coordenadas (x, y) de cada nodo y lista de enlaces.
- ✔ Devolver al backend el resultado completo de la operación (layout o información solicitada) para que éste lo procese y reenvíe al frontend.

En este microservicio no se implementa directamente la lógica de parsing en TypeScript: en su lugar, se utiliza el cliente de consola de BandageNG dentro de un contenedor Docker, accediendo a todas sus funciones de conversión y análisis mediante comandos preconfigurados.

4.3.6. Despliegue y Contenerización

Toda la infraestructura se integra mediante un único fichero `docker-compose.yml` ubicado en la raíz (Ilustración 4.8). Dicho fichero orquesta tres servicios:

- ✔ **frontend**: contiene la imagen Docker generada a partir de `tft-frontend/Dockerfile`, sirviendo la aplicación Angular compilada sobre NGINX.
- ✔ **backend**: contiene la imagen Docker generada a partir de `tft-backend/Dockerfile`, exponiendo los endpoints REST.
- ✔ **bandage**: contiene la imagen Docker generada a partir de `bandage/Dockerfile`, exponiendo su API para el procesamiento de archivos GFA.

Los contenedores se comunican mediante una red interna definida en `docker-compose.yml`, de modo que el frontend llama al backend y el backend invoca a bandage para solicitar la conversión de archivos GFA a layout.

Esta arquitectura modular y contenerizada permite desplegar cada componente de forma aislada y escalar únicamente aquel que lo requiera (por ejemplo, replicar múltiples instancias de `bandage` para procesar grafos de gran tamaño), manteniendo al mismo tiempo la flexibilidad de actualizar individualmente cada repositorio sin interrumpir el servicio global.

4.3.7. Validación

Se han implementado todas las funcionalidades definidas y, tras varias reuniones con el equipo del ITC, se confirmó que el comportamiento de la aplicación coincide con lo solicitado. A continuación, se listan las funcionalidades validadas:

- ✔ Carga de archivo `.gfa` y visualización inicial del grafo.
- ✔ Generación automática de layout y renderizado interactivo (zoom y pan).
- ✔ Modificación de nodos por sus propiedades:
  - anchura
  - color
  - mostrar/ocultar su id
- ✔ Selección de esquemas de color basados en metadatos.
- ✔ Exportación del layout actual en formato `.layout`.

Esto garantiza que el proyecto cumple con los requisitos planteados por el ITC. En la sección siguiente se presentan los resultados obtenidos al aplicar estas funcionalidades en distintos casos de uso reales, evaluando su rendimiento, usabilidad y efectividad en la representación de grafos de ensamblaje.

4.3.8. Metodología Iterativa

Durante el desarrollo del trabajo se adoptó una metodología iterativa basada en entregas parciales y ciclos de retroalimentación rápida. Se llevaron a cabo cuatro iteraciones principales, cada una centrada en cumplir objetivos progresivos relacionados con la replicación de funcionalidades de BandageNG en un entorno web. No todas las fases del proceso se aplicaron en cada iteración; a continuación, se detalla qué fases se ejecutaron en cada una de ellas.

Iteración 1 ✔ **Estudio previo / Análisis:** Se realizó una investigación exhaustiva del funcionamiento interno de BandageNG, identificando sus componentes clave y los requisitos del MVP.

✔ **Diseño / Desarrollo / Implementación:** Se diseñó la arquitectura general del sistema (frontend, backend y microservicio) y se elaboraron los primeros diagramas de capas y flujos de datos.

✔ **Evaluación / Validación / Prueba:** Se validó la viabilidad técnica del flujo de procesamiento de archivos `.gfa` a `.layout` en local.

Iteración 2 ✔ **Diseño / Desarrollo / Implementación:** Se desarrollaron las funcionalidades mínimas del MVP, incluyendo carga de archivos, renderizado interactivo del grafo y exportación a `.layout`.

✔ **Evaluación / Validación / Prueba:** Se realizó una demostración inicial al equipo del ITC, donde se validó el comportamiento básico del MVP.

✔ **Correcciones en Desarrollo:** Se realizaron ajustes menores en los flujos de comunicación entre el backend y el microservicio, a partir del feedback recibido.

Iteración 3 ✔ **Diseño / Desarrollo / Implementación:** Se ampliaron las funcionalidades con la incorporación de filtros por cobertura y longitud, así como esquemas de color.

✔ **Evaluación / Validación / Prueba:** Se validó con el ITC el funcionamiento correcto de los filtros y la visualización interactiva del grafo con distintas configuraciones.

✔ **Correcciones en Desarrollo:** Se corrigieron errores relacionados con la visualización de nodos y se mejoró la estabilidad del sistema frente a archivos `.gfa` mal formateados.

Iteración 4 ✔ **Diseño / Desarrollo / Implementación:** Se completó el sistema de exportación a formatos SVG/PNG y se preparó el entorno contenerizado para su despliegue conjunto.

✔ **Evaluación / Validación / Prueba:** Se validó globalmente la aplicación con todas sus funcionalidades integradas mediante reuniones finales con el ITC.

✔ **Correcciones en Desarrollo:** Se aplicaron las últimas mejoras derivadas de las pruebas finales, asegurando cumplimiento de requisitos.

✔ **Documentación / Presentación:** Se elaboró la documentación técnica y de usuario, y se preparó la presentación final del trabajo.

Capítulo 5

Resultados

A continuación se presenta el flujo completo de interacción con el componente web, desde la pantalla inicial hasta la modificación de propiedades del grafo. Este recorrido muestra cómo el usuario:

1. Accede a la pantalla principal sin ningún grafo cargado. Ver ilustración 5.2
2. Abre el menú lateral para desplegar las opciones de control.5.3
3. Selecciona y carga un archivo `.gfa` desde su sistema de archivos.5.4
4. Envía el archivo al backend, que invoca el microservicio Bandage para generar el layout.5.6
5. Visualiza el grafo resultante en el lienzo, con zoom y desplazamiento habilitados.5.7
6. Ajusta propiedades del grafo como el ancho de nodos, esquema de color y visibilidad de etiquetas mediante el panel de control.5.9
7. Observa el resultado del grafo tras aplicar dichos cambios, confirmando que la interfaz responde a las necesidades definidas por el ITC.

Dicho flujo podemos verlo representado en el diagrama de secuencias UML que se encuentra en la ilustración 5.1

5.1. Flujo de procesamiento de grafos

El proceso comienza cuando el usuario, desde el *frontend*, selecciona y carga un fichero en formato GFA; a continuación, el *frontend* envía este archivo al *backend*, que a su vez lo reenvía al microservicio Bandage para convertir la descripción topológica en un fichero `.layout` con coordenadas y estilos. Una vez Bandage devuelve dicho fichero, el *backend* envía al *frontend* toda la información necesaria para renderizar el grafo (posiciones de nodos, rutas de aristas y etiquetas) y presentar una vista interactiva. Finalmente, el usuario puede modificar propiedades (colores, etiquetas, tamaños) o desplazar nodos, de modo que cada

cambio genera una solicitud del *frontend* al *backend* y, si es necesario recalcular el layout, de nuevo a Bandage, tras lo cual el *frontend* actualiza la representación gráfica en pantalla.

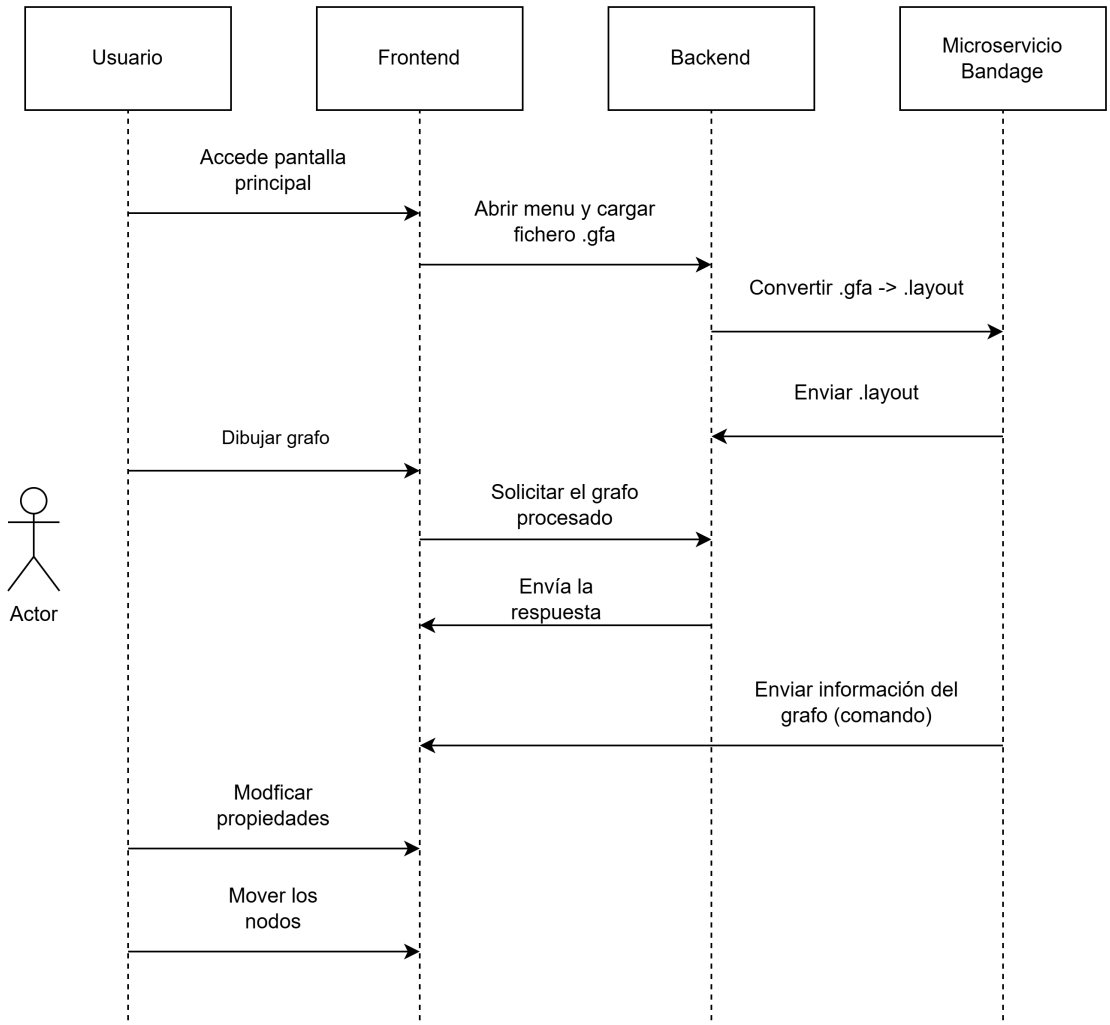

Ilustración 5.1: Diagrama UML del flujo del proyecto

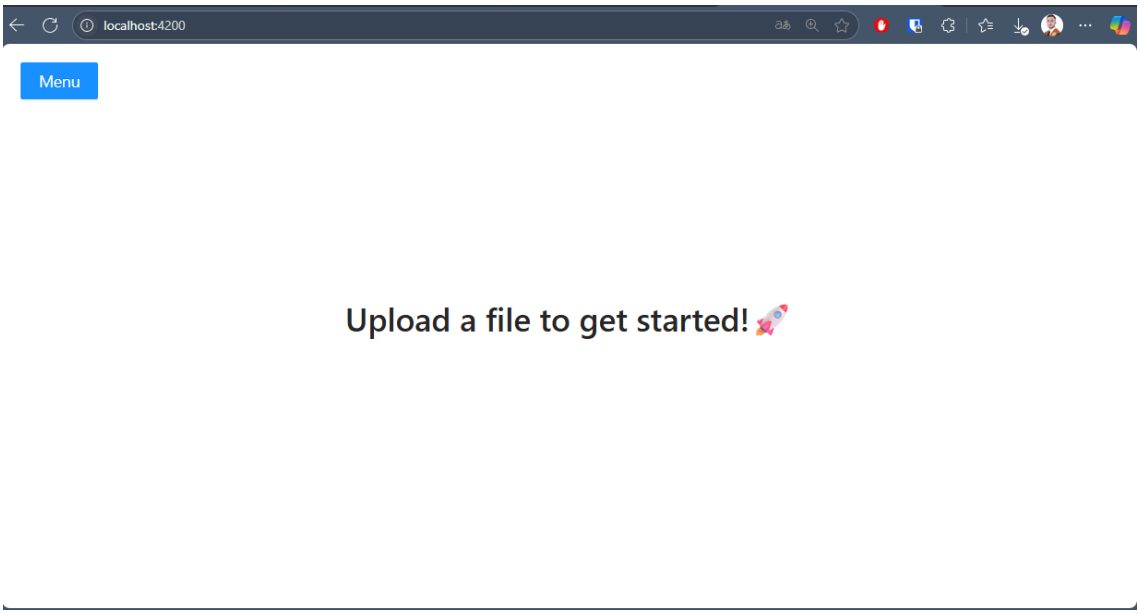

Ilustración 5.2: Pantalla inicial de la aplicación TFT antes de cargar un archivo

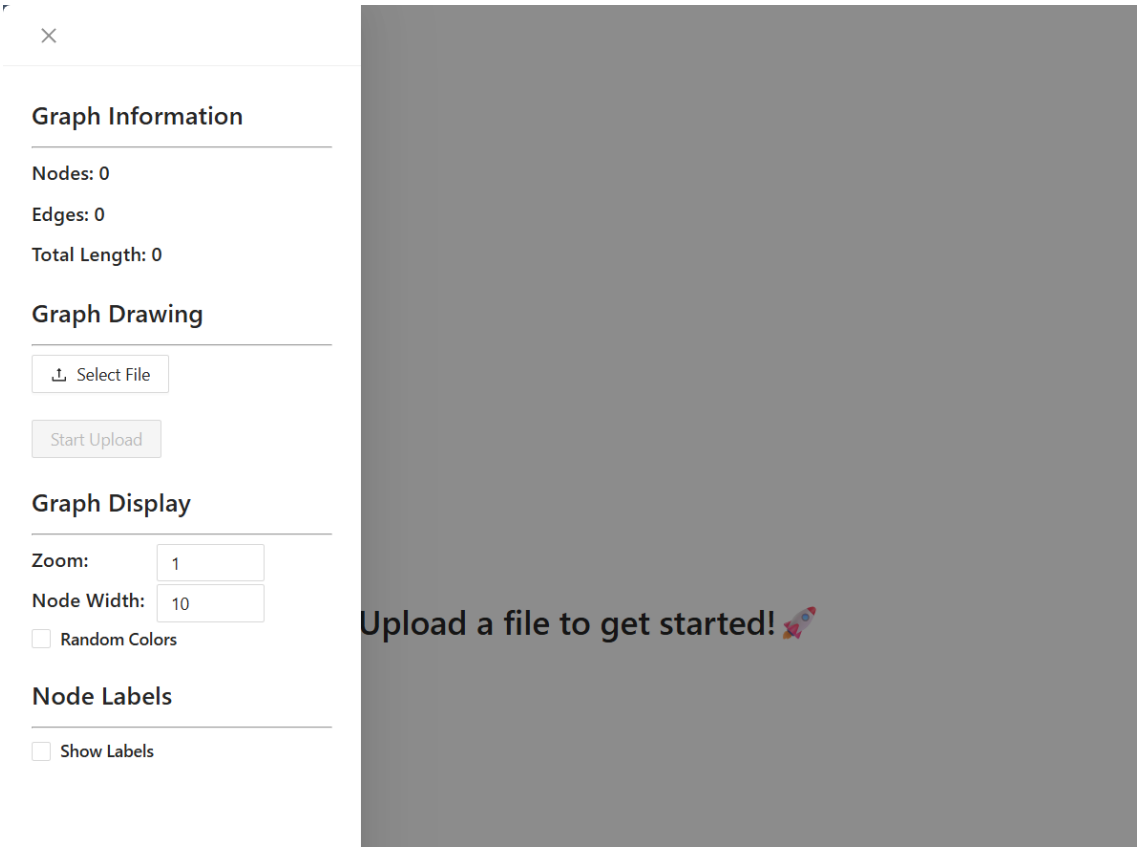

Ilustración 5.3: Apertura del menú lateral con las opciones de control del grafo

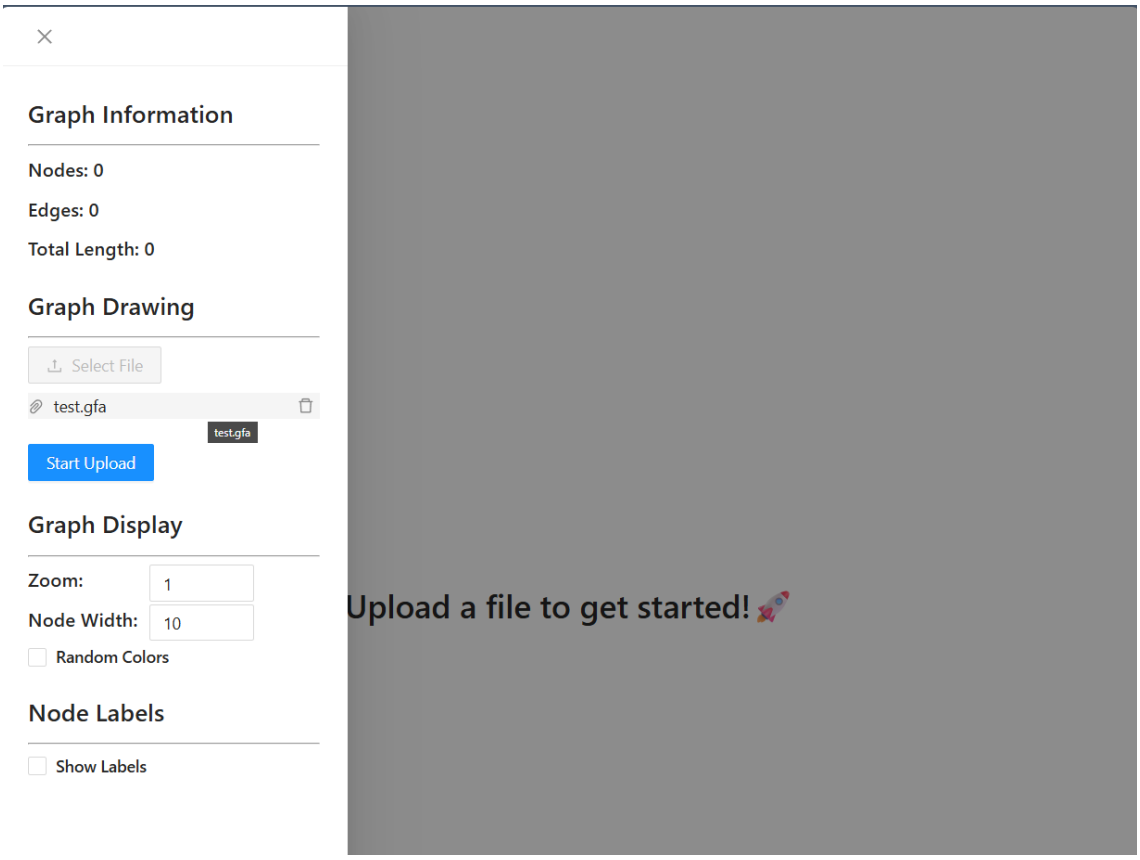

Ilustración 5.4: Selección de un archivo `.gfa` en el panel de carga

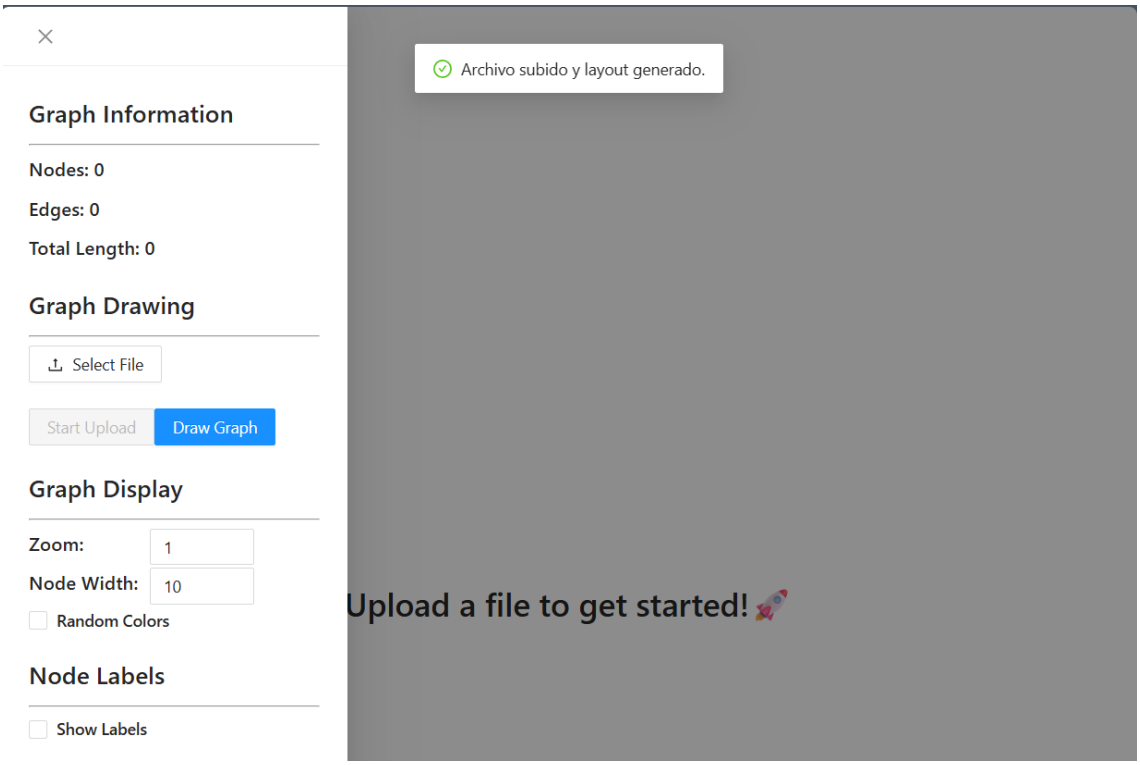

Ilustración 5.5: Archivo `.gfa` subido y layout generado exitosamente

```
export class GraphRoutes {
  static get routes(): Router {
    const router = Router();
    const graphController = new GraphController();

    router.get('/', graphController.getTodos);

    router.post("/upload", upload.single("file"), graphController.uploadFile);

    router.get("/parsed-gfa", graphController.getParsedGfa);

    return router;
  }
}
```

Ilustración 5.6: Endpoint que recoge el fichero cargado y pasa de .gfa a .layout

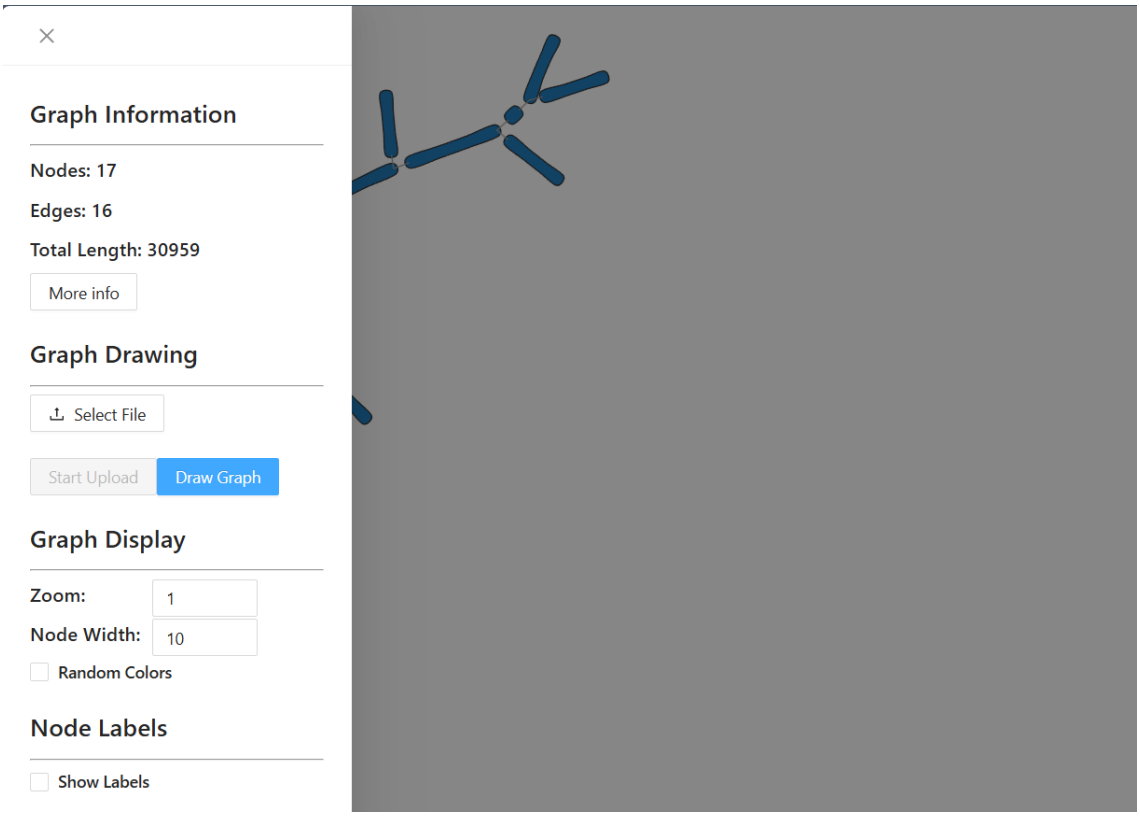

Ilustración 5.7: Visualización inicial del grafo generado en el lienzo

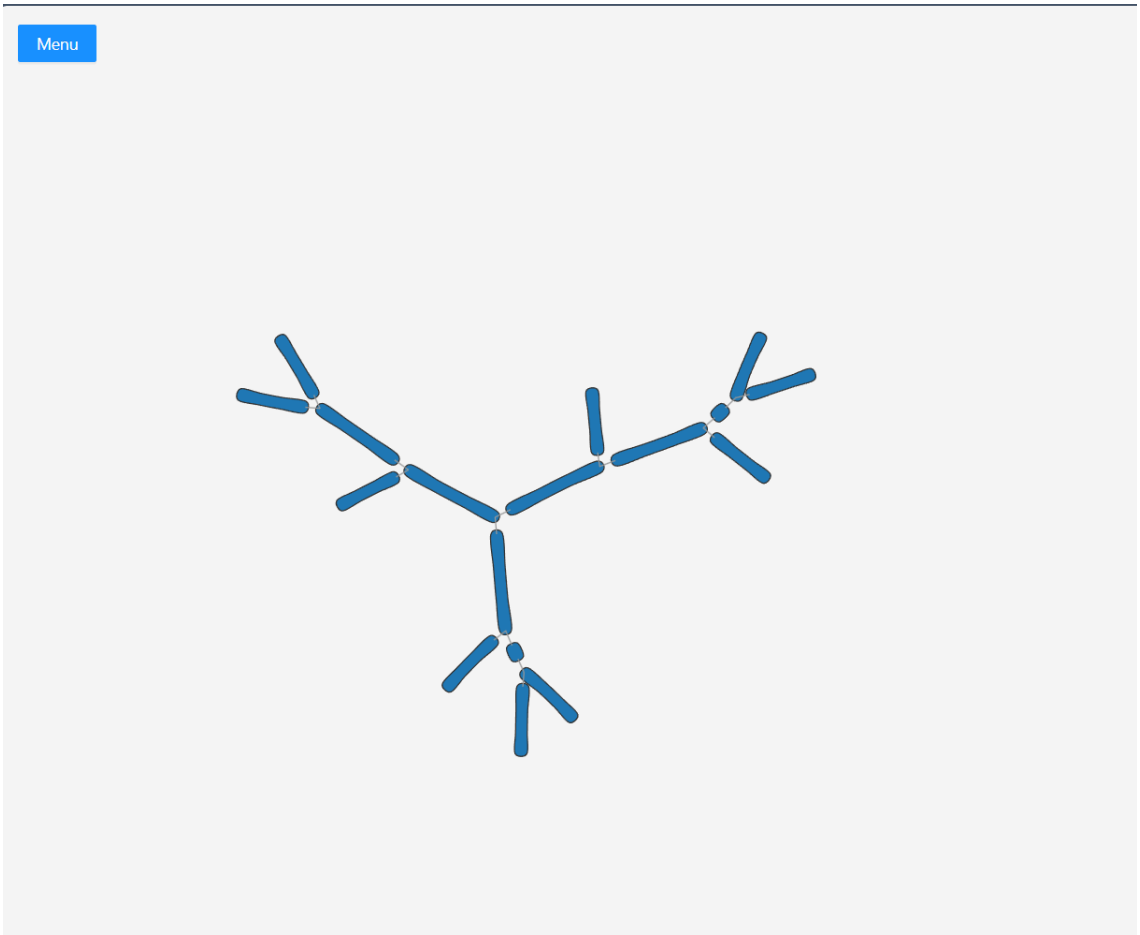

Ilustración 5.8: Vista completa del grafo renderizado en el componente Angular

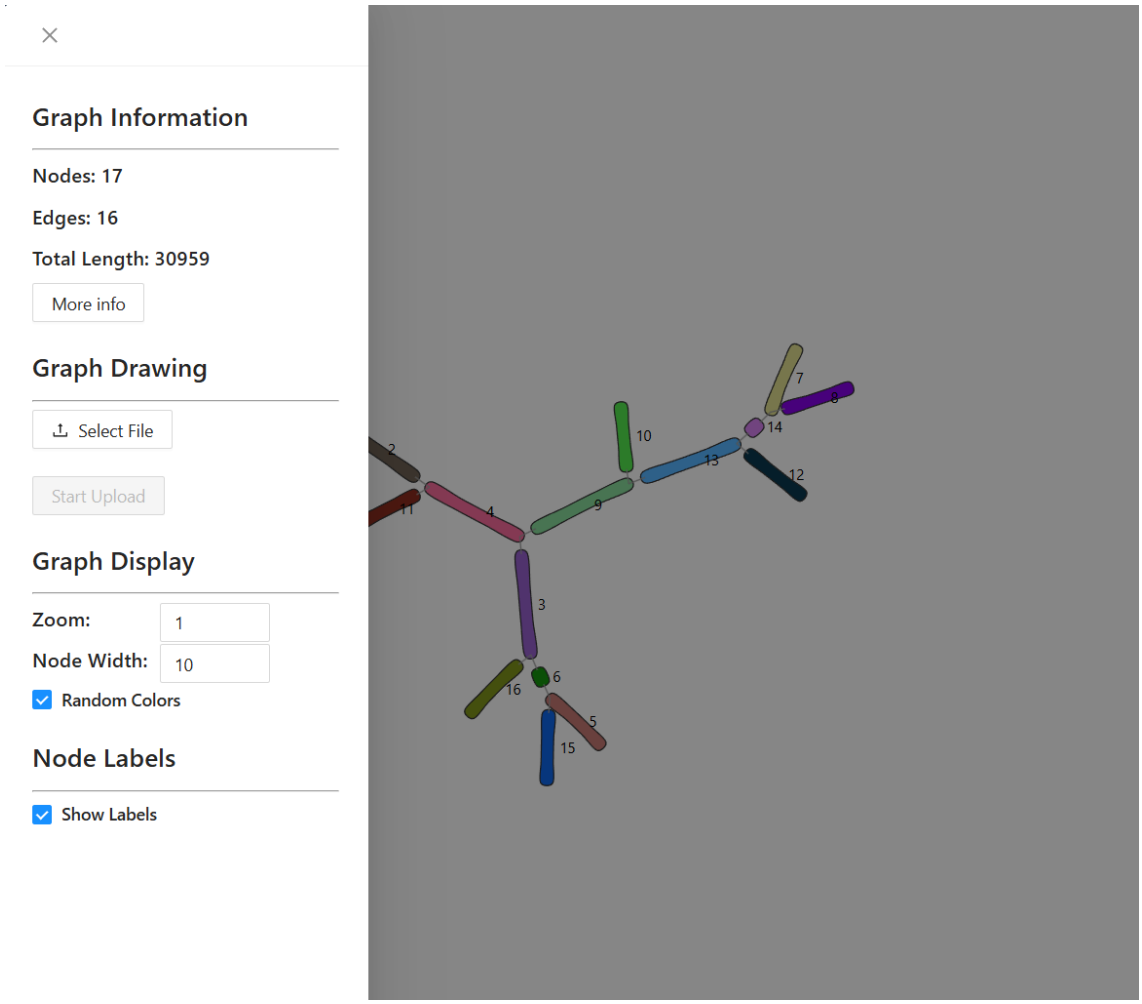

Ilustración 5.9: Panel de propiedades del grafo para modificar anchura, color y etiquetas de nodos

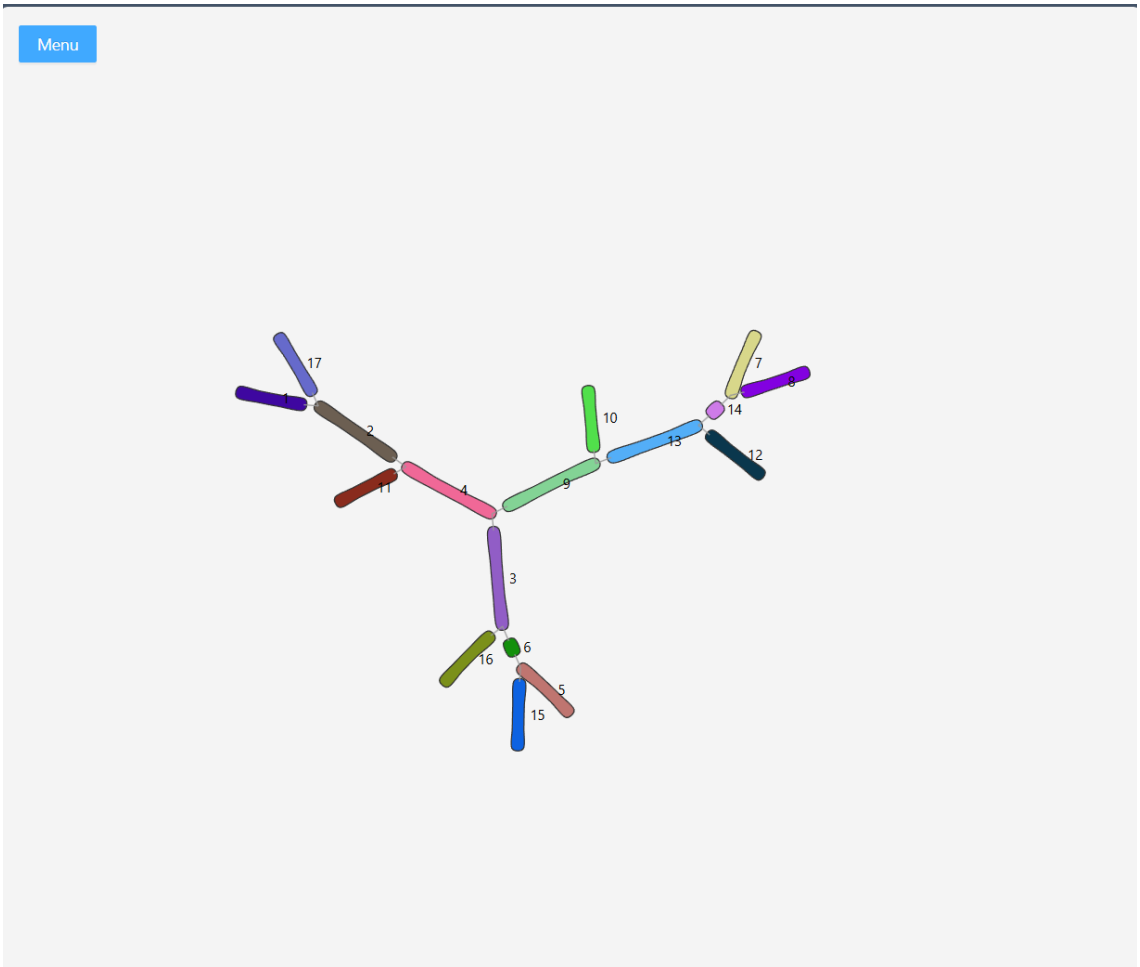

Ilustración 5.10: Resultado del grafo tras aplicar cambios en propiedades de nodos

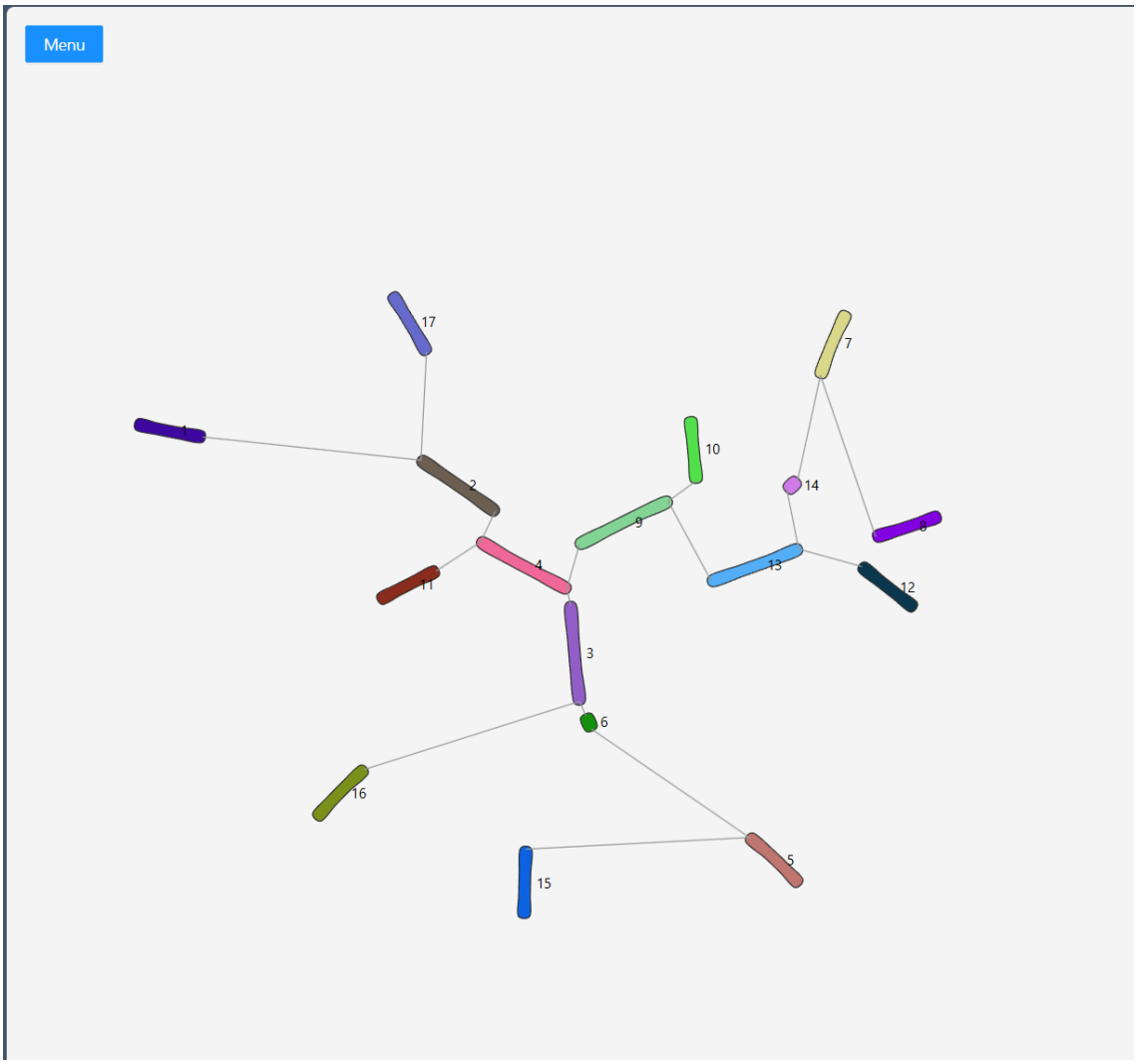

Ilustración 5.11: Ubicación de nodos tras la recalibración del layout

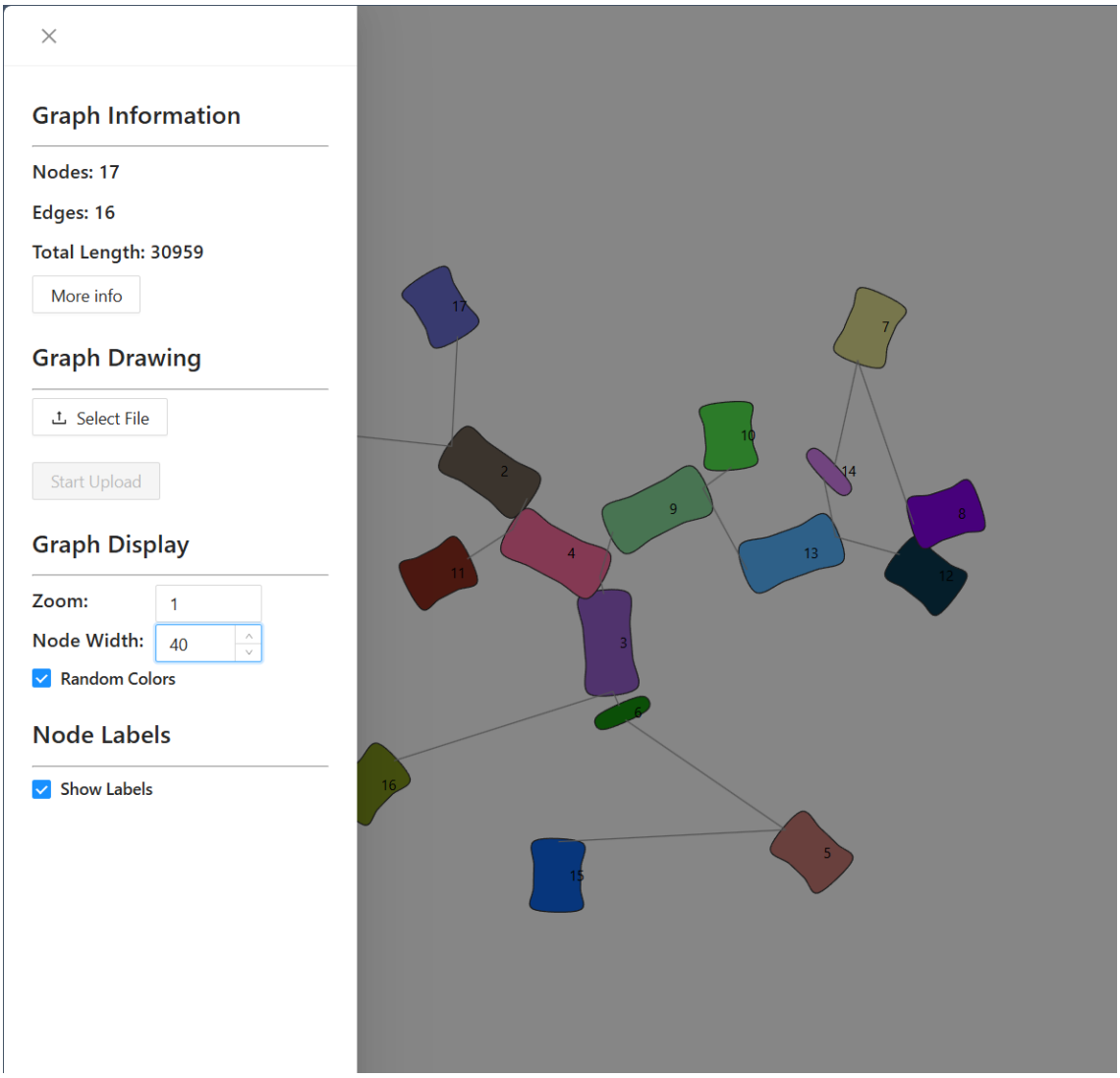

Ilustración 5.12: Modificación del ancho de los nodos

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

A lo largo de este Trabajo de Fin de Título se han cumplido todos los objetivos planteados:

- ✔ Desarrollo de un componente web capaz de visualizar grafos de ensamblaje de novo de genomas, replicando las funcionalidades principales de la aplicación de escritorio Bandage.
- ✔ Personalización de color, ajuste de propiedades de nodos y exportación de layout en formato `.layout`, tal y como requería el ITC.
- ✔ Validación continua con el equipo del ITC, asegurando que la solución web se ajusta a las necesidades reales.

6.1.1. Beneficios de una solución web frente a escritorio local

- ✔ **Accesibilidad remota:** Al migrar Bandage a un componente web, cualquier investigador puede acceder a la herramienta desde un navegador, sin necesidad de instalar software ni gestionar actualizaciones locales.
- ✔ **Distribución sencilla:** La versión web se despliega desde un servidor centralizado, evitando problemas de compatibilidad con diferentes sistemas operativos (Windows, Linux, macOS).
- ✔ **Colaboración:** Los usuarios pueden compartir enlaces directos a grafos almacenados en el servidor, facilitando la colaboración entre distintos grupos de trabajo del ITC.
- ✔ **Integración con cuadros de mando:** Al ser un componente web, puede integrarse fácilmente en paneles corporativos o workflows de bioinformática basados en Angular o frameworks similares.

Contenerización y despliegue

Se ha dockerizado cada artefacto (frontend, backend y microservicio BandageNG) para garantizar entornos reproducibles y portables. Cada repositorio genera su propia imagen Docker:

- ✔ `emmartel/bandage-api`: imagen que incluye el cliente de consola de BandageNG configurado para recibir archivos `.gfa` y devolver un layout en JSON.[10]
- ✔ `emmartel/tft-backend`: servidor Node.js/Express con Clean Architecture, responsable de orquestar la llamada al contenedor Bandage y exponer API REST.[11]
- ✔ `emmartel/tft-frontend-prod`: aplicación Angular compilada y servida desde NGINX.[12]

Todas las imágenes se han subido al Docker Hub con etiquetas multiplataforma (Linux amd64 y Mac ARM64). Para simplificar el despliegue, se ha proporcionado un fichero `docker-compose.yml` que levanta simultáneamente los tres contenedores y los conecta mediante una red interna. De esta manera, el usuario solo necesita:

```
docker-compose up -d
```

y obtendrá el sistema operativo completo, sin tener que instalar dependencias locales ni preocuparse por versiones de Node.js, Angular o librerías de parsing GFA.

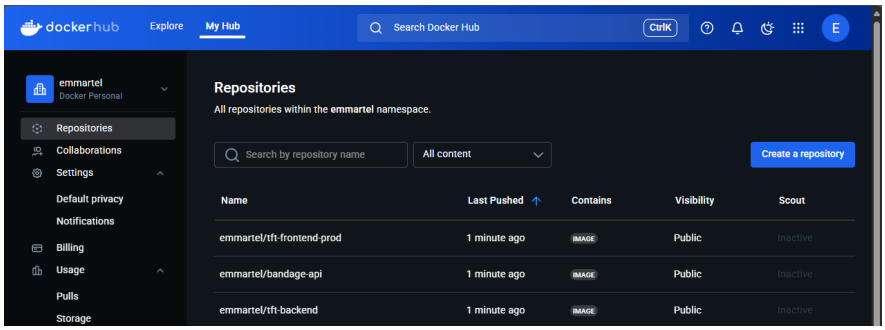

Ilustración 6.1: Listado de imágenes Docker en Docker Hub para los tres componentes

Análisis de calidad con SonarQube

Para garantizar la robustez y mantenibilidad del código, se integró SonarQube en el pipeline CI/CD de cada repositorio. El análisis estático se configuró para detectar:

- ✔ **Vulnerabilidades de seguridad:** Uso de dependencias vulnerables, inyección de código o patrones de programación inseguros.
- ✔ **Código duplicado:** Fragmentos de código repetido que podrían refactorizarse para mejorar la cohesión.
- ✔ **Code smells:** Prácticas de codificación que pueden derivar en problemas de mantenimiento.

Todos los proyectos superaron el *Quality Gate* 6.5 de SonarQube con estado "Passed", sin issues críticos pendientes. A continuación se muestran capturas de pantalla de los dashboards de SonarQube para cada componente:

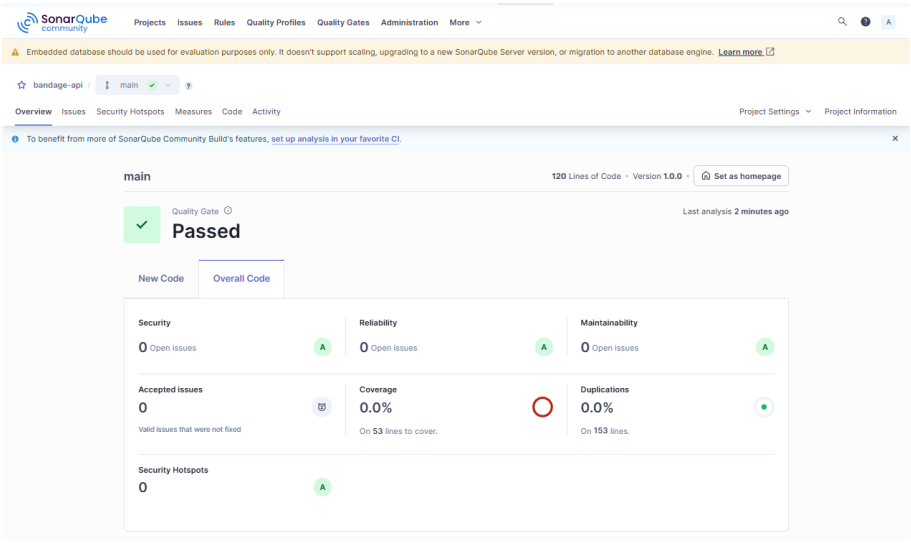

Ilustración 6.2: Resultado de SonarQube para la imagen `bandage-api`

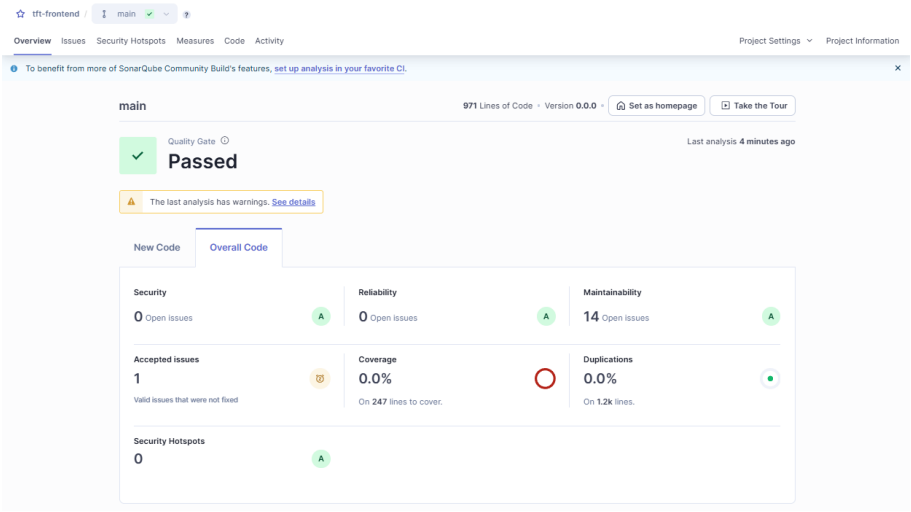

Ilustración 6.3: Resultado de SonarQube para `tft-frontend`

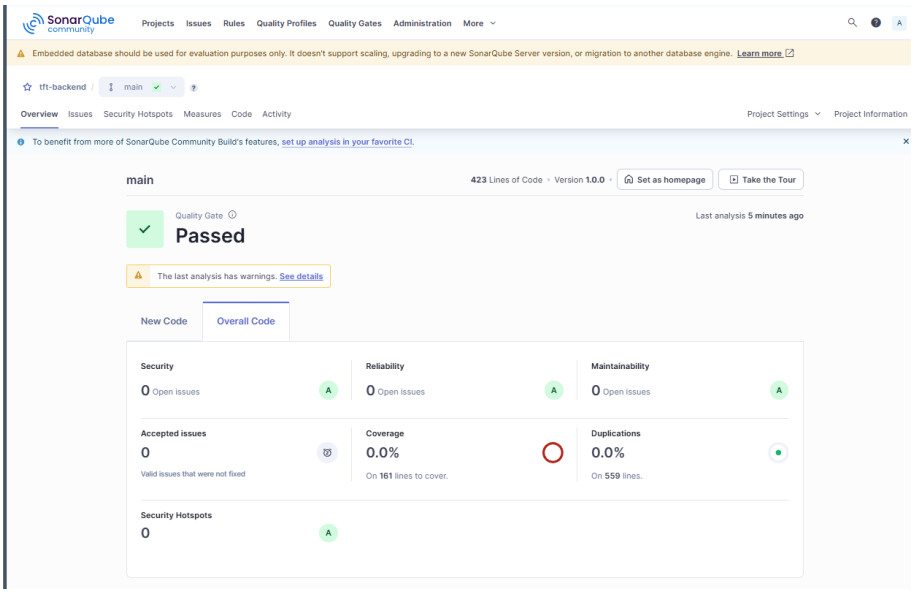

Ilustración 6.4: Resultado de SonarQube para `tft-backend`

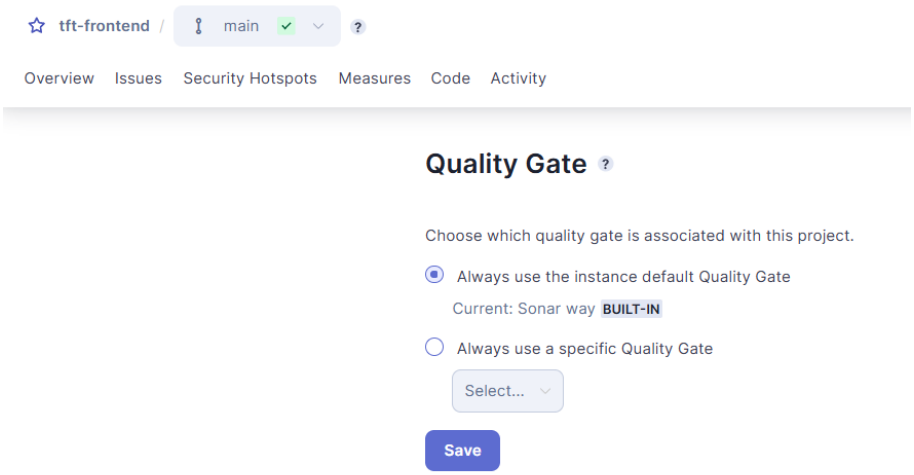

Ilustración 6.5: Configuración general del QualityGate para los proyectos

6.1.2. Conclusión final

El componente web desarrollado satisface completamente los objetivos definidos por el ITC, proporcionando una experiencia de usuario similar a 'Bandage' de escritorio, pero con las ventajas propias de una solución web: despliegue centralizado, accesibilidad multiplataforma y fácil integración en cuadros de mando. La contenerización de cada artefacto y la disponibilidad de imágenes Docker en Docker Hub garantizan una instalación rápida y

sin conflictos de versiones. Finalmente, el uso de SonarQube ha asegurado que el código se mantenga libre de vulnerabilidades, duplicaciones innecesarias y smells, aportando una base sólida para futuros desarrollos y mejoras continuas.

6.2. Trabajo futuro

En fases posteriores se ampliarán las capacidades del visor web incorporando funcionalidades avanzadas que actualmente ofrece Bandage de escritorio, adaptándolas a un entorno completamente nativo. Entre las líneas de trabajo futuro se incluyen:

- ✔ **Localización de nodos por identificador:** permitir al usuario introducir o seleccionar un ID concreto para centrar el grafo en el nodo correspondiente y destacar su vecindario inmediato.
- ✔ **Agrupación y estilización personalizada de nodos:** ofrecer la posibilidad de definir grupos de nodos (por ejemplo, por rango de cobertura o longitud) y aplicar estilos específicos (colores, formas, tamaños) de forma dinámica.
- ✔ **Modo de visualización centrado en la topología:** habilitar una vista simplificada que muestre únicamente la estructura del grafo (nodos y aristas) sin metadatos adicionales, facilitando el análisis de la generación del ensamblaje.
- ✔ **Procesamiento nativo de archivos de Bandage:** estudiar y extraer la lógica interna de Bandage para implementar las operaciones de parsing de GFA y cálculo de layout directamente en el visor, sin depender exclusivamente del cliente de consola en contenedor.

Estas extensiones permitirán ofrecer una experiencia más completa y flexible, acercando todas las herramientas esenciales de Bandage a un entorno web integrado y optimizado para los cuadros de mando del ITC.

Capítulo 7

Otros capítulos y anexos

7.1. Manual de usuario y software

A continuación se describe el procedimiento para obtener, instalar y ejecutar el proyecto completo mediante Docker Compose. Para facilitar la colaboración y futuras actualizaciones, todo el código fuente está alojado en GitHub.

7.1.1. Requisitos previos

- ✔ **Git:** para clonar los repositorios desde GitHub. [15]
- ✔ **Docker Desktop:** incluye Docker Engine y Docker Compose. Disponible para Windows, macOS (Intel y Apple Silicon) y Linux. [5]

7.1.2. Clonado de los repositorios

El proyecto consta de tres repositorios principales en GitHub. Cree una carpeta nueva, abra una terminal y ejecute los siguientes comandos para descargarlos localmente:

```
# Clonar el frontend (visor de grafos)
git clone https://github.com/EmilioMartel/tft-frontend.git

# Clonar el backend (servidor principal)
git clone https://github.com/EmilioMartel/tft-backend.git

# Clonar el microservicio Bandage
git clone https://github.com/EmilioMartel/bandage.git
```

Tras estos pasos, en su carpeta local tendrá tres subdirectorios:

- ✔ `tft-frontend/`

- ✔ `tft-backend/`
- ✔ `bandage/`

7.1.3. Instalación de Docker Desktop

1. Visite https://www.docker.com/products/docker-desktop y descargue el instalador para su sistema operativo.
2. Ejecute el instalador y siga las indicaciones:
   - ✔ En Windows: habilite WSL 2 si aún no lo tiene configurado.
   - ✔ En macOS (Apple Silicon o Intel): siga las instrucciones del paquete .dmg.
   - ✔ En Linux: instale Docker Engine y Docker Compose según la documentación oficial (generalmente no necesita Docker Desktop).
3. Una vez instalado, abra Docker Desktop y espere a que el indicador muestre "Docker is running".

7.1.4. Configuración del proyecto

Dentro de la carpeta raíz (donde clonó los tres repositorios), cree o verifique que existe el fichero `docker-compose.yml` con el siguiente contenido mínimo:

```
services:
  backend:
    env_file:
      - ./tft-backend/.env
    volumes:
      - ./files:/app/files
      - ./gfa:/app/gfa
    build:
      context: ./tft-backend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    restart: always

  frontend:
    build:
      context: ./tft-frontend
      dockerfile: Dockerfile
    ports:
      - "4200:80"
```

```
    depends_on:
      - backend
    restart: always

  bandage:
    build:
      context: ./bandage
      dockerfile: Dockerfile
    volumes:
      - ./files:/app/files
      - ./gfa:/app/gfa
    ports:
    - "3001:3000"
    depends_on:
      - backend
    restart: always
```

Si su repositorio ya incluye este fichero con rutas relativas, no necesita modificar nada. Solo asegúrese de estar ubicado en la carpeta que contiene `tft-frontend/`, `tft-backend/` y `bandage/`.

7.1.5. Compilación y despliegue

1. Abra una terminal en la carpeta raíz del proyecto (donde está `docker-compose.yml`).
2. Ejecute el siguiente comando para compilar todas las imágenes Docker y levantar los contenedores:

   ```
   docker-compose up --build -d
   ```

   - ✔ La opción `--build` fuerza la creación de cada imagen a partir del `Dockerfile` correspondiente.
   - ✔ La opción `-d` lanza los contenedores en segundo plano (modo "detached").
3. Verifique que los contenedores estén en funcionamiento:

   ```
   docker-compose ps
   ```

   Debería ver tres contenedores con estado `Up`.

   - ✔ `tft-frontend` en el puerto 4200.
   - ✔ `tft-backend` en el puerto 3000.
   - ✔ `tft-bandage` en el puerto 3001.

7.1.6. Uso de la aplicación web

1. Abra su navegador y vaya a http://localhost:4200.
2. Se mostrará la pantalla de inicio del visor de grafos.
3. Para cargar un grafo, haga clic en Çargar archivo", seleccione un fichero `.gfa` desde su equipo y pulse .Enviar".
4. El backend enviará el `.gfa` al microservicio Bandage, que generará las coordenadas y devolverá un JSON.
5. El grafo aparecerá renderizado en el lienzo; utilice las opciones de zoom y desplazamiento para explorarlo.
6. Desde el panel lateral, aplique filtros de longitud o cobertura, cambie esquemas de color y muestre u oculte etiquetas de nodos.
7. Para exportar el layout, pulse .Exportar layoutz descargue el fichero `.layout` resultante.

7.1.7. Detención y limpieza

Cuando desee detener y eliminar todos los contenedores, redes y volúmenes asociados, ejecute:

```
docker-compose down
```

Si además quiere eliminar las imágenes creadas, use:

```
docker-compose down --rmi all
```

7.1.8. Acceso al código fuente

Para explorar el código fuente, realice un `git pull` en cada repositorio clonado:

```
cd tft-frontend
git pull

cd ../tft-backend
git pull

cd ../bandage
git pull
```

Así obtendrá las últimas actualizaciones. Si desea contribuir o presentar problemas, abra un *issue* o *pull request* en el repositorio correspondiente de GitHub.

7.1.9. Notas adicionales

- ✔ Asegúrese de usar Docker Desktop en versiones recientes (mínimo 20.10.x) para compatibilidad con Compose v2.
- ✔ Si trabaja en macOS con Apple Silicon (M1/M2), compruebe que las imágenes y dependencias sean compatibles con ARM64.
- ✔ En caso de conflicto de puertos, edite `docker-compose.yml` y cambie las asignaciones (por ejemplo, `"4201:80"` para frontend).

Con estos pasos, cualquier usuario podrá clonar, construir y ejecutar la aplicación completa de forma sencilla, sin necesidad de instalaciones manuales adicionales.

7.2. Herramientas de IA empleadas

En el transcurso del Trabajo de Fin de Título se han integrado distintas herramientas de Inteligencia Artificial con el fin de optimizar la generación de contenido, documentación técnica y asistencia durante el desarrollo del código:

- ✔ **GitHub Copilot (VS Code):** Se usó para acelerar la escritura de fragmentos de código tanto en los diferentes proyectos. Copilot sugirió estructuras de control, importaciones automáticas y fragmentos repetitivos para la configuración de módulos, servicios y componentes. Gracias a sus propuestas contextuales, se redujo el tiempo invertido y se identificaron patrones de diseño recomendados para la comunicación con el backend.
- ✔ **ChatGPT (OpenAI):** Empleado como asistente de redacción y revisión, facilitó la elaboración de explicaciones sobre la arquitectura del proyecto, la generación de plantillas LaTeX para la memoria y ejemplos de tablas y figuras. ChatGPT ayudó a depurar sintaxis, ofrecer sugerencias de buenas prácticas en la organización del documento y proveer resúmenes claros sobre temas complejos. Además, sirvió para resolver dudas puntuales durante la integración de bibliotecas externas y para estructurar secciones del manual de usuario.
- ✔ **Gemini (Google AI):** Utilizado para consultar fuentes oficiales y actualizadas, como la documentación de Angular, guías de estilo de LaTeX, normativas de accesibilidad web y API públicas. A través de consultas en lenguaje natural, Gemini localizó enlaces a manuales técnicos, ejemplos de configuración de Docker Compose y recomendaciones de seguridad en entornos Node.js. De este modo se obtuvo información verificada sobre versiones de dependencias, se contrastaron mejores prácticas y se citaron correctamente recursos oficiales en la bibliografía.

Bibliografía

[1] Angular Team (2025). Angular: One framework. mobile & desktop. https://angular.dev/.

[2] codeatcg (2025). Vrpg: Visor de grafos de pangenoma de variantes con d3.js. https://github.com/codeatcg/VRPG/tree/main.

[3] D3.js Consortium (2025). D3.js: Data-driven documents. https://d3js.org/.

[4] Docker Inc. (2025a). Docker compose documentation. https://docs.docker.com/compose/.

[5] Docker Inc. (2025b). Docker: Empowering app development for developers. https://www.docker.com/.

[6] Emilio Martel (2025). tft-frontend: Componente angular para visualización de grafos. https://github.com/EmilioMartel/tft-frontend.

[7] Emilio Martel Díaz (2025a). bandage: Microservicio para conversión de gfa a layout en json. https://github.com/EmilioMartel/bandage.

[8] Emilio Martel Díaz (2025b). tft-backend: Servicio node.js/express para procesamiento de grafos. https://github.com/EmilioMartel/tft-backend.

[9] Emilio Martel Díaz (2025c). Tft01: Documento del trabajo de fin de titulación. https://drive.google.com/file/d/1pMWQ3yOH9kEnp-UWlcNnZG6TDLSmwMI0/view?usp=sharing.

[10] emmartel (2025a). emmartel/bandage-api Docker image. https://hub.docker.com/repository/docker/emmartel/bandage-api/general.

[11] emmartel (2025b). emmartel/tft-backend Docker image. https://hub.docker.com/repository/docker/emmartel/tft-backend/general.

[12] emmartel (2025c). emmartel/tft-frontend-prod Docker image. https://hub.docker.com/repository/docker/emmartel/tft-frontend-prod/general.

[13] Escuela de Ingeniería Informática, Universidad de Las Palmas de Gran Canaria (2019). Grado en ingeniería informática: Memoria del plan de estudios (2019). https://www.eii.ulpgc.es/sites/default/files/2022-05/Grado%20en%20Ingenier%C3%ADa%20Inform%

C3%A1tica%20-%20memoria%20del%20plan%20de%20estudios%20-%202019.pdf. Páginas 12–13.

[14] Express.js Team (2025). Express: Fast, unopinionated, minimalist web framework for node.js. https://expressjs.com/.

[15] Git SCM (2025). Git – distributed version control system. https://git-scm.com/.

[16] GMOD (2025). Jbrowse components: Visualización genómica en la web usando html canvas. https://github.com/GMOD/jbrowse-components.

[17] Instituto Tecnológico de Canarias (ITC) (2023). Nextgendem: la bioinformática al servicio de la conservación de la flora endémica de la macaronesia. https://www.itccanarias.org/web/es/actualidad/noticias/nextgendem-la-bioinformatica-al-servicio-de-la-conservacion-de-la-flora-endemica-de-la-macaronesia.

[18] Mikheenko, A. and Kolmogorov, M. (2019). Assembly graph browser: interactive visualization of assembly graphs. *Bioinformatics*, 35(18):3476–3478.

[19] MoMI-G Project (2025). Momi-g: Visor integrado de genomas modular multiescala (d3.js). https://github.com/MoMI-G/MoMI-G/.

[20] National Human Genome Research Institute (NHGRI) (2019). Secuenciación del adn. https://www.genome.gov/es/about-genomics/fact-sheets/Secuenciacion-del-ADN.

[21] NG-ZORRO (Ant Design for Angular) (2025). Ng-zorro: Ant design components for angular. https://ng.ant.design/docs/introduce/en.

[22] Node.js Foundation (2025). Node.js: Javascript runtime built on chrome's v8 engine. https://nodejs.org/es.

[23] Project NEXTGENDEM (2021). Nextgendem: información genética, geoespacial y supercomputación para mejorar la gestión de especies y espacios en macaronesia. https://lupus.itccanarias.org/nextgendem/es.

[24] ReactiveX Team (2025). Rxjs: Reactive extensions for javascript. https://rxjs.dev/.

[25] SonarSource (2025). Sonarqube: Continuous code quality monitoring. https://www.sonarsource.com/.

[26] TF-Chan-Lab (2025). Pangraphviewer: Visualización de grafos de pangenoma con python, bokeh y dna_features_viewer. https://github.com/TF-Chan-Lab/panGraphViewer.

[27] TypeScript Team (2025). Typescript: Javascript with syntax for types. https://www.typescriptlang.org/.

[28] Universidad de Las Palmas de Gran Canaria (ULPGC) (2025). Investigadores de la ulpgc y ull presentan los resultados de sus trabajos sobre las consecuencias del cambio climático en canarias. https://www.ulpgc.es/noticia/2025/05/06/investigadores-ulpgc-y-ull-presentan-resultados-sus-trabajos-consecuencias-del.

[29] vgteam (2025). Sequencetubemap: Visualización de grafos genómicos con d3.js. https://github.com/vgteam/sequenceTubeMap.

[30] Wick, R. R. (2015). Bandage: interactive visualization of de novo genome assemblies. https://github.com/rrwick/Bandage.

[31] Wick, R. R., Schultz, M. B., Zobel, J., and Holt, K. E. (2015). Bandage: interactive visualization of *de novo* genome assemblies. https://doi.org/10.1093/bioinformatics/btv383.

Glosario

datos genómicos conjunto de secuencias de ADN que contienen información acerca de la estructura y función de los genes. 1

ensamblaje de novo Proceso de reconstrucción de un genoma a partir de lecturas de ADN sin usar un genoma de referencia previo. 5

GesPlan consultora en gestión de espacios naturales. 1, 5

Instituto Tecnológico de Canarias centro especializado en I+D+i para el desarrollo de soluciones tecnológicas. 1, 5

Jardín Botánico Viera y Clavijo colección científica de flora canaria. 1, 5

MVP versión mínima con funcionalidades esenciales para validar y recibir retroalimentación temprana con los usuarios. 18

Nextgendem plataforma de bioinformática y georreferenciación orientada a especies endémicas de la Macaronesia. 1, 5

pangenoma Conjunto completo de secuencias genómicas y variantes de todos los individuos de una especie, que representa la diversidad genética dentro del clado estudiado. 5

UML Lenguaje de Modelado Unificado (Unified Modeling Language): estándar para especificar, visualizar, construir y documentar los artefactos de un sistema de software. 31