Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Mención en Ingeniería de Software

Nomad WebDraw. Aplicación web para la creación de trabajos de fresado para máquinas CNC

Autor

Ricardo Villanueva Ojosnegros

Tutoras

Yania Crespo González-Carvajal

María Margarita Gonzalo Tasis

Agradecimientos

A mis padres.
A mi familia.
A Carla, mi novia.
A mis amigos y compañeros de universidad.
A mis tutoras Yania y Margarita.
A Nomad Technologies.
Por el apoyo y ayuda, gracias a todos.

Resumen

El objetivo de este proyecto es el desarrollo de una aplicación web para la generación de trabajos de fresado en máquina CNC. La aplicación tendrá un lienzo en el que se podrá dibujar y, posteriormente, exportar el dibujo en código G (lenguaje para máquinas CNC). Este proyecto se ha realizado para un cliente, la empresa Nomad Technologies S.L.

El proyecto se ha desarrollado utilizando el framework Angular, Typescript, la biblioteca three js de JavaScript, HTML y CSS siguiendo una metodología de desarrollo UPEdu.

La aplicación web desarrollada se ha llamado Nomad WebDraw.

Abstract

The objective of this project is the development of a web application for generating works for milling for CNC machines. The application will have a canvas in which you would draw and, later, export the drawing in Gcode (languaje used un CNC machines). This project has been done for a client, the enterprise Nomad Technologies S.L.

The project has been developed using the framework Angular, Typescript, the JavaScript library three js, HTML and CSS and following a software development methodology UPEdu.

The application has been named Nomad WebDraw.

Índice general

Lista de figuras

Lista de tablas

Capítulo 1

Introducción y contexto

1.1 Introducción

1.1.1 Motivación

En los últimos años han ganado en popularidad y número de usuarios las aplicaciones web con una utilidad comparable a la de un software tradicional. Esta revolución, ayudada por las ventajas que supone (fácil mantenimiento y actualización, monitorización continua, multiplataforma, simplicidad de acceso, distribución de tareas entre servidor y cliente...), también ha llegado a la fabricación digital. Las máquinas CNC (impresoras 3D, fresadoras, grabadoras láser) necesitan del llamado "software CAM" para la preparación de trabajos a partir de un diseño previo. Históricamente estos programas han sido complejos, pesados y difíciles de utilizar.

Por esta razón la empresa Nomad Technologies S.L. deseaba tener una aplicación web, propia y sencilla para realizar los trabajos de dibujo que más tarde se exportarían a un archivo GCode para que lo ejecutara una fresadora o grabadora láser.

Nomad Technologies es una joven empresa de Valladolid dedicada al diseño, fabricación y venta de maquinaria CNC de bajo coste y pequeño-medio formato.

Después se acordaron los límites y el contenido de la aplicación para que se adecuase a la duración y complejidad de un TFG. La propuesta final de TFG es la siguiente:

> *Diseño y desarrollo de una aplicación web que permita dibujar formas geométricas básicas 2D en un lienzo, realizar operaciones de transformación de los dibujos en el lienzo (traslación, reescalado, rotación…), calcular las trayectorias que debe seguir una herramienta de fresado para realizar operaciones de perfilado y vaciado, y exportar los dibujos así obtenidos a código G (lenguaje estándar de texto simple para máquinas CNC fresadoras), teniendo en cuenta opciones que se permitirán configurar tales como el diámetro de la herramienta y la velocidad.*

1.1.2 Objetivos

El TFG consiste en elaborar un proyecto en el que haya que aplicar todos (o la mayoría) de los conocimientos, habilidades y actitudes adquiridos durante los años de estudio del grado.

Los objetivos de formación del TFG son:

- Buscar, ordenar y estructurar información para la realización de un proyecto informático.
- Realizar un trabajo teniendo un cliente real que tiene sus propias necesidades e intereses.
- Elaborar una memoria completa de un proyecto informático.
- Elaborar y defender una presentación pública del trabajo realizado.

Además de los objetivos de formación, están los objetivos propios de la aplicación. Éstos engloban desde la adquisición de los conocimientos necesarios para el desarrollo como los que describen la finalidad de la aplicación. Los objetivos de la aplicación son:

- Estudiar cómo funciona una fresadora y cómo se controla.
- Estudiar el código G y aprender a usarlo para realizar los trabajos deseados.
- Estudiar el framework elegido para realizar la aplicación, así como las librerías necesarias.
- Permitir dibujar y transformar diseños simples en dos dimensiones.
- Permitir generar trayectorias de trabajos de fresado a partir del diseño y ciertos parámetros necesarios para la fresadora.
- Permitir generar el código para una máquina fresadora que realice el trabajo.
- Preparar la aplicación para su evolución a lo largo del tiempo.
- Desplegar la aplicación una vez ya acabada en un servidor.
- Conseguir una aplicación fácil de usar para los usuarios finales de la aplicación, la mayoría sin mucha experiencia con las tecnologías.

1.2 Contexto

En este apartado se explica brevemente el contexto del mundo del fresado, sobre el que gira todo el tema del TFG. Qué es, cómo se hace, qué máquinas se utilizan, etc.

1.2.1 Fresado

El fresado, en el contexto en el que nos encontramos, consiste en un proceso realizado por una herramienta similar a lo que podría ser una taladradora moviéndose en un plano horizontal sobre una plancha de material (normalmente madera, pero también se pueden usar otros materiales como plástico o metales blandos). Lo que se consigue con esto es que la herramienta, al moverse, va abriendo un surco en el material y se programa la máquina para que siga el recorrido predefinido para que “dibuje” las formas y figuras que se deseen. No únicamente se mueven en el plano horizontal, también tienen movilidad en vertical porque necesitan subir y bajar para moverse por encima del material cuando no están

realizando ningún trabajo, además de poder penetrar a más o menos profundidad en el material.

Las aplicaciones del fresado, principalmente, son dos. La primera sería crear relieves. Su finalidad es la de obtener un bloque de material en el que se incrusta un dibujo o diseño en él y de esta manera decorarlo. Esta aplicación del fresado se puede ver en la Figura 1 y en la Figura 2.

Figura 1. Ejemplo de fresado [6]

Figura 2. Ejemplo de fresado [12]

La otra aplicación del fresado es el corte. Aprovechando la movilidad en vertical de la máquina, se puede llevar la herramienta hasta el final del material, consiguiendo así desprender la pieza del resto del tablero. A continuación, algunos ejemplos de corte con fresadora.

Figura 3. Ejemplo de corte con fresado [3]

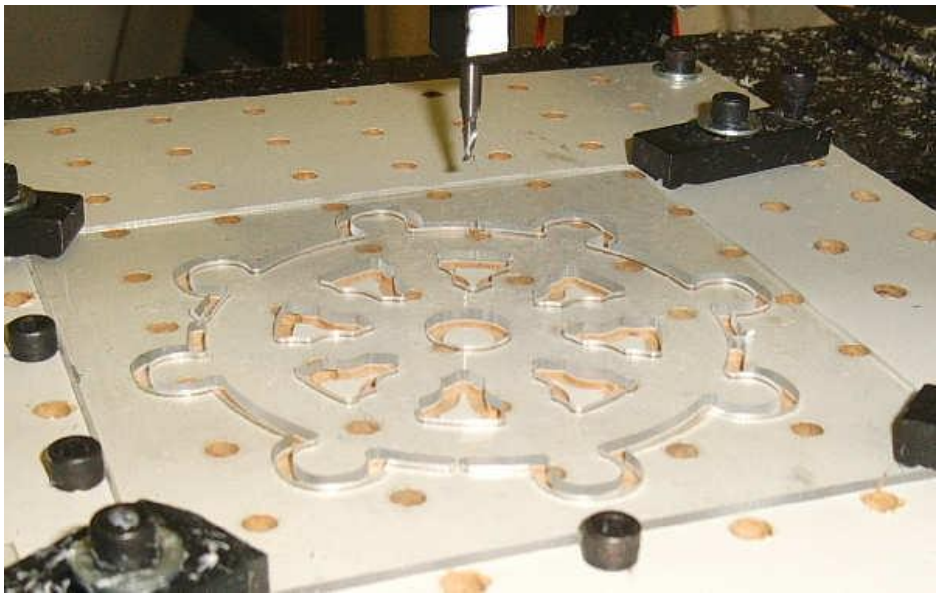

Figura 4. Ejemplo de corte con fresado [15]

1.2.2 Máquinas fresadoras

Las fresadoras pueden ser muy dispares, pero principalmente tienen los siguientes componentes:

- Herramienta. Una especie de taladro, con un motor y una broca.
- Dos raíles. Uno para poder mover la herramienta por cada uno de los ejes de coordenadas X e Y.

- Tres motores. Encargados de mover a la herramienta en los ejes X, Y y Z a través de los raíles y subiendo y bajando la herramienta.

A continuación, se presentan algunas máquinas fresadoras de uso industrial para poder observar cómo son.

Figura 5. Maquina fresadora industrial [9]

Figura 6. Maquina fresadora industrial [7]

La Figura 7, la Figura 8, la Figura 9 y la Figura 10 muestran las máquinas desarrolladas por Nomad Technologies y las que finalmente realizarán los trabajos diseñados en la aplicacion desarrollada en el TFG. Nomad Technologies se dedica a fabricar fresadoras de uso más artesanal en vez de industrial. Se puede ver que la diferencia con las máquinas fresadoras industriales es que son más simples, por lo que abarata su precio y simplifica su manejo y montaje. Las imágenes fueron obtenidas de la página web de Nomad Technologies [14].

Figura 7. Kit Fox Basic S

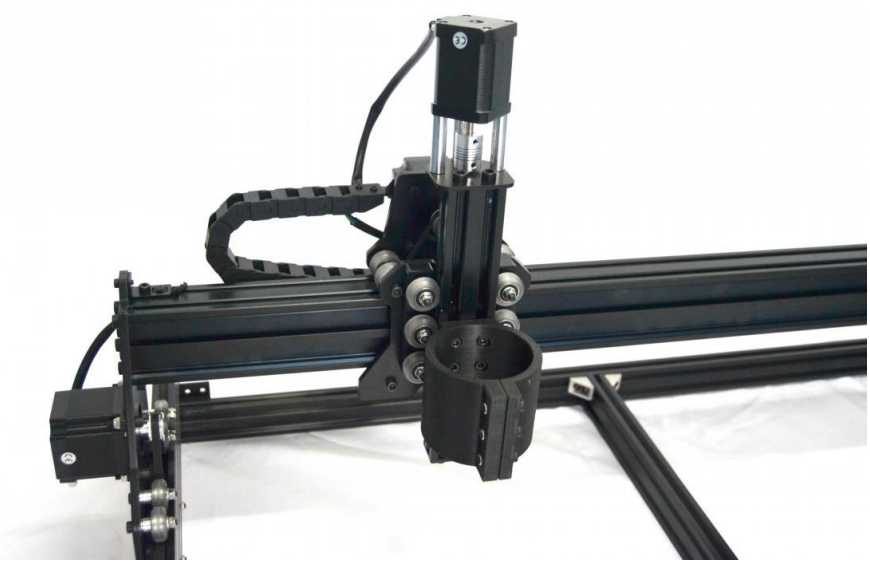

Figura 8. Detalle de herramienta de Kit Fox Basic S

Figura 9. Red Fox S

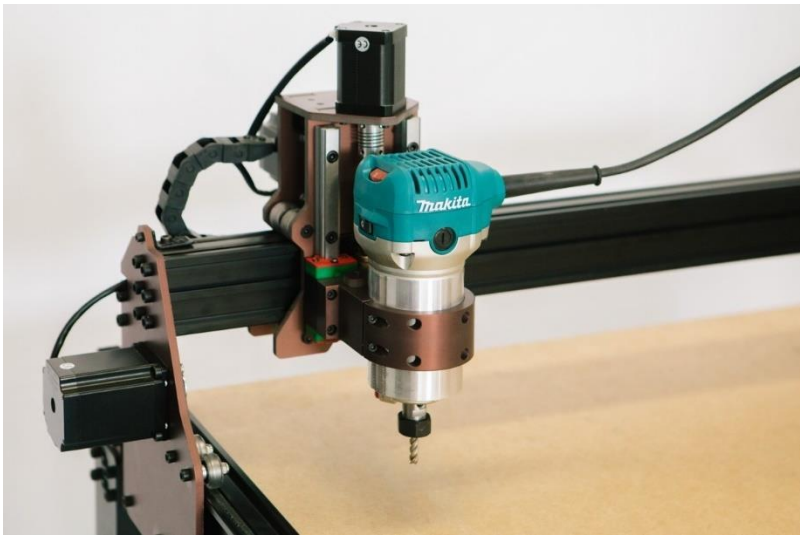

Figura 10. Detalle de herramienta de Red Fox S

1.2.3 Software relacionado

1.2.3.1 Inkscape

Es una aplicación de escritorio desarrollado para crear dibujos vectoriales. Gracias a que es de código abierto la comunidad desarrolló un plugin que transforma el dibujo a GCode. Debido a todos estos motivos es uno de los programas que principalmente usan en Nomad Technologies actualmente.

1.2.3.2 Aspire

Es una aplicación de escritorio desarrollada por la empresa Vectric. Éste es un software específico para crear trabajos para máquinas CNC por lo que lleva integrado de serie además del dibujo, el cálculo de trayectorias y la conversión a GCode. Es más completa que Inkscape, pero menos personalizable, al no ser de código abierto.

1.2.3.3 jscut

Es una aplicación web de código abierto, pero sin función de dibujado. En ésta se importan los dibujos hechos en otra aplicación de dibujado vectorial y calcula la trayectoria y el GCode.

1.2.3.4 Comparativa

Tanto Inkscape como Aspire son programas profesionales, pero Nomad Technologies buscaba una aplicación web por las ventajas que ello conlleva, como no tener que instalar el programa o sus actualizaciones (algo que puede ser complicado para los usuarios de sus productos), etc. Encontraron como opción jscut, pero no era viable por necesitar de software adicional para realizar los dibujos.

Que fuera de código abierto no era indispensable, pero sí de ayuda si el programa era de terceros, ya que podía no ajustarse a sus necesidades, pero de serlo se podría personalizar para solucionarlo.

	Aplicación web	Dibujado	Generación de Gcodes	Código abierto
Inkscape	No	Sí	Sí	Sí
Aspire	No	Sí	Sí	No
jscut	Sí	No	Sí	Si

Tabla 1. Comparativa de software relacionado

Como ninguna de las opciones se adecuaba correctamente a las necesidades del cliente hay que descartarlas. La otra opción es desarrollar un nuevo software que satisfaga sus necesidades. Los principales requisitos son los que no encontraban juntos en las demás aplicaciones, que se pudiera dibujar y generar Gcodes del dibujo y que además fuera una aplicación web.

1.2.4 Código G

Según Wikipedia el código G o G-code es:

> *G-code es un lenguaje mediante el cual las personas pueden decir a máquinas herramienta controladas por computadora qué hacer y cómo hacerlo. Esos "qué" y "cómo" están definidos mayormente por instrucciones sobre a dónde moverse, cuán rápido moverse y qué trayectoria seguir. Las máquinas típicas que son controladas con G-code son fresadoras, cortadoras, tornos e impresoras 3D.* [5]

Al ser un lenguaje muy genérico para multitud de máquinas CNC tiene muchas funciones que para el caso que nos ocupa son irrelevantes, como podrían ser los comandos para encender apagar un láser o una aspiradora, cambiar la temperatura del extrusor u otros para el control de tarjetas SD.

A continuación, se detallan los comandos necesarios para el desarrollo del TFG, su función y un ejemplo:

- G0. Este comando indica que es un movimiento rápido, es decir, de desplazamiento. Por ello se realiza mucho más rápido que un movimiento de trabajo. La velocidad a la que lo hace viene predefinida. A continuación del comando deben venir las coordenadas. Por ejemplo, si se quisiera volver rápido al origen el comando sería el siguiente:

```
G0 X0 Y0 Z0
```

- G1. Este comando indica que es un movimiento de trabajo. Se realiza bastante más despacio que un G0, a la velocidad que le haya indicado un comando F, es por ello por lo que antes de un G1 debe haberse llamado previamente a un F. A continuación del comando deben venir las coordenadas. Por ejemplo, si se quisiera mover la herramienta trabajando al punto (45, 200), sin desplazarnos en el plano vertical, el comando sería el siguiente:

```
G1 X45 Y200
```

- F. Este comando indica la velocidad de trabajo. Esa velocidad de trabajo se mantiene hasta que se encuentre una nueva F que cambie la velocidad. Si se quisiera cambiar la velocidad de trabajo solo por una línea (por ejemplo, para penetrar el material) después habría que volver a indicar de nuevo la velocidad de trabajo previa. Pueden escribirse en una línea él solo o al final de un comando GCode. Si se quisiera indicar que la velocidad de trabajo es de 2500 mm/min el comando sería el siguiente:

```
F2500
```

Las coordenadas vienen por defecto de forma absoluta, pero existen los comandos G90 y G91 que establecen que las coordenadas sean absolutas o relativas a partir de ese punto, respectivamente.

GCode no hace diferenciación de mayúsculas y minúsculas. Tampoco tiene en cuenta espacios. En cambio, sí que considera los saltos de línea ya que estos se interpretan como el fin del comando. A pesar de esto se suelen escribir con el formato antes descrito para favorecer la legibilidad. Además, se pueden escribir comentarios. Para ello el comentario debe comenzar con un punto y coma (;) y finalizar en el salto de línea.

El código G, al no ser un lenguaje de programación como tal, no es compilable. Esto significa que no hay comprobación de su corrección antes de ejecutarlo, por lo que la responsabilidad de asegurarse de que está bien es de quien lo genera. Para comprobar la corrección del código G generado, se han desarrollado simuladores de máquinas de fresado para no tener que probar con máquinas reales, puesto que son muy costosas y un fallo en el código G podría estropearlas. Un ejemplo de estos simuladores es CAMotics, usada en este TFG para comprobar la corrección del código G generado por la aplicación.

1.3 Estructura de la memoria

El presente documento ha sido organizado de la siguiente forma:

- Capítulos
    - En el Capítulo 2 se presentan los requisitos, validados por el cliente y los casos de uso, así como las ampliaciones de los mismos que fueron necesarias.
    - En el Capítulo 3 se presentan la planificación y seguimiento del proyecto. Se detalla la calendarización del mismo, el presupuesto, los riesgos y los ajustes a los mismos realizados durante el seguimiento.
    - En el Capítulo 4 se presenta el análisis de los requisitos y los casos de uso. También se pueden encontrar los diagramas de actividad y el modelo de domino.
    - En el Capítulo 5 se presentan las tecnologías utilizadas en el desarrollo de la aplicación, así como el control de versiones utilizado.
    - En el Capítulo 6 se presenta el diseño de la aplicación incluyendo la arquitectura de la misma.
    - En el Capítulo 7 se presenta la implementación del código y las pruebas realizadas para testear su funcionalidad y usabilidad.
    - En el Capítulo 8 se presentan las conclusiones y el trabajo futuro sobre la aplicación.
- Bibliografía
- Anexos
    - En el Apéndice A se presenta el manual de instalación del entorno para desarrolladores.
    - En el Apéndice B se presenta el manual de despliegue.
    - En el Apéndice C se presenta el manual de usuario.
    - En el Apéndice D se presenta manual de mantenimiento.
    - En el Apéndice E se presenta el contenido del CD-ROM entregado al tribunal.
    - En el Apéndice F se presenta el diccionario de datos y términos usados en la memoria

Capítulo 2

Requisitos

2.1 Contexto

El objetivo es desarrollar una herramienta que sea una aplicación web para dibujar formas (en principio polígonos básicos) y posteriormente exportar un código G para una fresadora. Esta fresadora tiene variables como el tamaño de la broca o la velocidad de movimiento de la máquina.

2.2 Requisitos funcionales

ID	Vers.	Nombre	Descripción	Prioridad
RF01	1.0	Lienzo dibujo	El sistema tendrá un lienzo de dibujo.	Alta
RF02	1.0	Dibujo formas	El sistema deberá permitir dibujar formas básicas en el lienzo (cuadrado, triángulo equilátero y círculo).	Alta
RF03	1.0	Transformación dibujo	El sistema deberá permitir hacer transformaciones a las figuras del lienzo (traslación, escalado y rotación).	Alta
RF04	1.0	Cálculo trayectorias	El sistema deberá calcular las trayectorias de perfilado y vaciado de los dibujos.	Alta
RF05	1.0	Parámetros fresado	El sistema deberá tener la opción de cambiar el tamaño de broca y velocidad de fresado.	Alta
RF06	1.0	Exportación código G	El sistema deberá exportar el código G del dibujo que se ha hecho.	Alta

Tabla 2. Requisitos funcionales

Los expuestos anteriormente son los requisitos que se obtuvieron del cliente en la elicitación de requisitos de la primera iteración. Al final de la fase de construcción se volvió a quedar con el cliente en una reunión para mostrarle los avances y recibir su retroalimentación. En esta reunión, el cliente vio que el producto no se ajustaba a la idea inicial que tenía sobre la aplicación. Es por ello por lo que se han elicitado unos nuevos requisitos para satisfacer las necesidades del cliente y una nueva iteración de las fases de inicio y elaboración. Los nuevos requisitos son los expuestos a continuación:

ID	Vers.	Nombre	Descripción	Prioridad
RF07	1.0	Trayectorias animadas	El sistema mostrará las trayectorias calculadas de forma animada, que simulará el movimiento de la herramienta.	Alta
RF08	1.0	Controles trayectorias	El sistema deberá implementar controles de visualización de la animación de las trayectorias (play, pause y reset).	Alta
RF09	1.0	Parámetros fresado adicionales	El sistema deberá tener la opción de indicar la altura de seguridad, la profundidad de pasada, la profundidad total y la velocidad de entrada.	Media
RF10	1.0	Deshacer / rehacer	El sistema deberá tener funcionalidad para deshacer y rehacer los cambios que se realicen en las figuras (creación, borrado, traslación, rotación y escalado).	Media
RF11	1.0	Escalar ejes	Se podrá escalar para cada uno de los dos ejes de coordenadas individualmente.	Baja
RF12	1.0	Lista figuras	La aplicación deberá mostrar las figuras en una lista.	Alta
RF13	1.0	Seleccionar figura	La aplicación deberá permitir seleccionar la figura sobre la que se desea trabajar de entre las dibujadas.	Alta
RF14	1.0	Borrar figura	La aplicación deberá permitir borrar la figura seleccionada.	Alta

Tabla 3. Requisitos funcionales. Segunda iteración

2.3 Requisitos no funcionales

ID	Vers.	Nombre	Descripción	Importancia
RNF01	1.0	Aplicación web	La aplicación deberá ser una aplicación web.	Crítica
RNF02	1.0	Compatibilidad navegadores	La aplicación deberá ser compatible con Chrome 56+, Firefox 51+, Safari 11+ y Edge.	Crítica
RNF03	1.0	Requisitos SO	Los requisitos mínimos de SO serán Windows 7 y MacOS 10.11.	Crítica
RNF04	1.0	Desarrollo gráfico	La parte gráfica de la aplicación se desarrollará en Three js.	Crítica
RNF05	1.0	Usabilidad	Un usuario sin experiencia deberá ser capaz de usar la aplicación en media hora con la ayuda y explicaciones de alguien con experiencia.	Deseable

Tabla 4. Requisitos no funcionales

2.4 Requisitos de información

Sólo hay un tipo de usuario, por lo tanto, todos los usuarios tienen acceso a toda la funcionalidad de la aplicación y no hay que iniciar sesión para utilizar la aplicación. Por ello no hay ningún requisito de información relativo a identificación de usuarios.

ID	Vers.	Nombre	Descripción
RIN01	1.0	Información de dibujo	El sistema deberá guardar la información de las formas dibujadas en el lienzo. De todas guardará las coordenadas del centro y el grado de inclinación. Del círculo además guardará la longitud del radio, Y del cuadrado y el triángulo, la longitud del lado.
RIN02	1.0	Información de fresado	El sistema deberá guardar la información de las opciones de fresado. Tipo de trayectoria (perfilado o vaciado), velocidad de fresado y tamaño de la broca.

Tabla 5. Requisitos de información

2.5 Restricciones

ID	Vers.	Nombre	Descripción	Prioridad
RES01	1.0	Visualización 3D	El sistema deberá tener un espacio reservado para el dibujo 3D para futuras ampliaciones.	Alta

Tabla 6. Restricciones

En la reunión con el cliente al final de la primera iteración de la fase de construcción (explicada anteriormente en el apartado de requisitos funcionales) además de requisitos funcionales se extrajeron también restricciones a mayores de las previamente obtenidas. Las nuevas restricciones se detallan a continuación:

ID	Vers.	Nombre	Descripción	Prioridad
RES02	1.0	Color del lienzo	El color del lienzo deberá ser de un color claro (blanco, gris claro, etc.).	Alta
RES03	1.0	Color cuadrícula	El color de la cuadrícula del lienzo deberá ser de un color claro pero un poco más oscuro que el lienzo	Alta
RES04	1.0	Barra figuras	Para crear las figuras tendrá que haber una barra con cada una de las opciones (con la intención de que se puedan añadir nuevas figuras en un futuro)	Alta
RES05	1.0	Seleccionar figura	La aplicación deberá resaltar tanto en el lienzo como en la lista la figura que se haya seleccionado.	Alta
RES06	1.0	Apariencia figuras	Las figuras deberán tener un color sólido, en vez de ser representadas con el wireframe.	Alta
RES07	1.0	Opciones trayectoria	Las opciones para el cálculo de trayectorias deberán estar en el mismo apartado que las de dibujo.	Alta

RES08	1.0	Rotación absoluta	La rotación de los objetos deberá ser tratada de forma absoluta.	Alta
RES09	1.0	Trayectoria transición	La línea que indica la trayectoria de transición entre figuras deberá ser de un color diferente al de la trayectoria de las figuras.	Alta
RES10	1.0	Perfilado por fuera	La trayectoria de perfilado deberá contornear la figura por el exterior.	Alta
RES11	1.0	Tamaño lienzo	El lienzo deberá mostrar una sección de 500x800 en disposición vertical.	Alta

Tabla 7. Restricciones. Segunda iteración

2.6 Diagrama de casos de uso

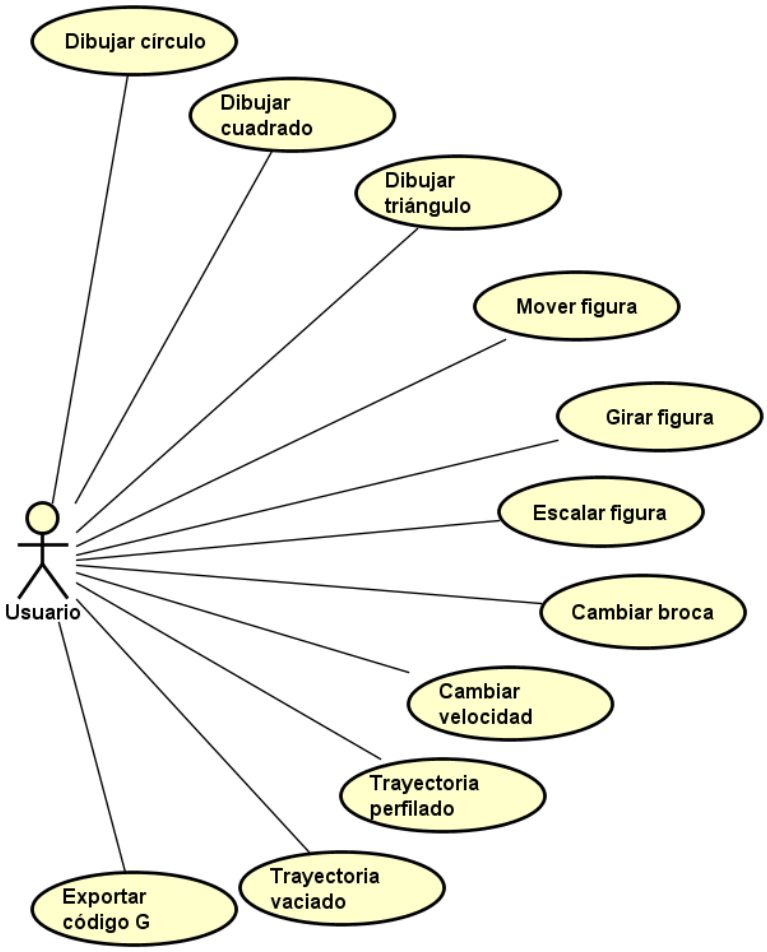

Figura 11. Diagrama de casos de uso obtenido en la primera iteración

Para la segunda iteración, el diagrama de casos de uso quedaría de la siguiente manera, resaltando los nuevos casos de uso en blanco:

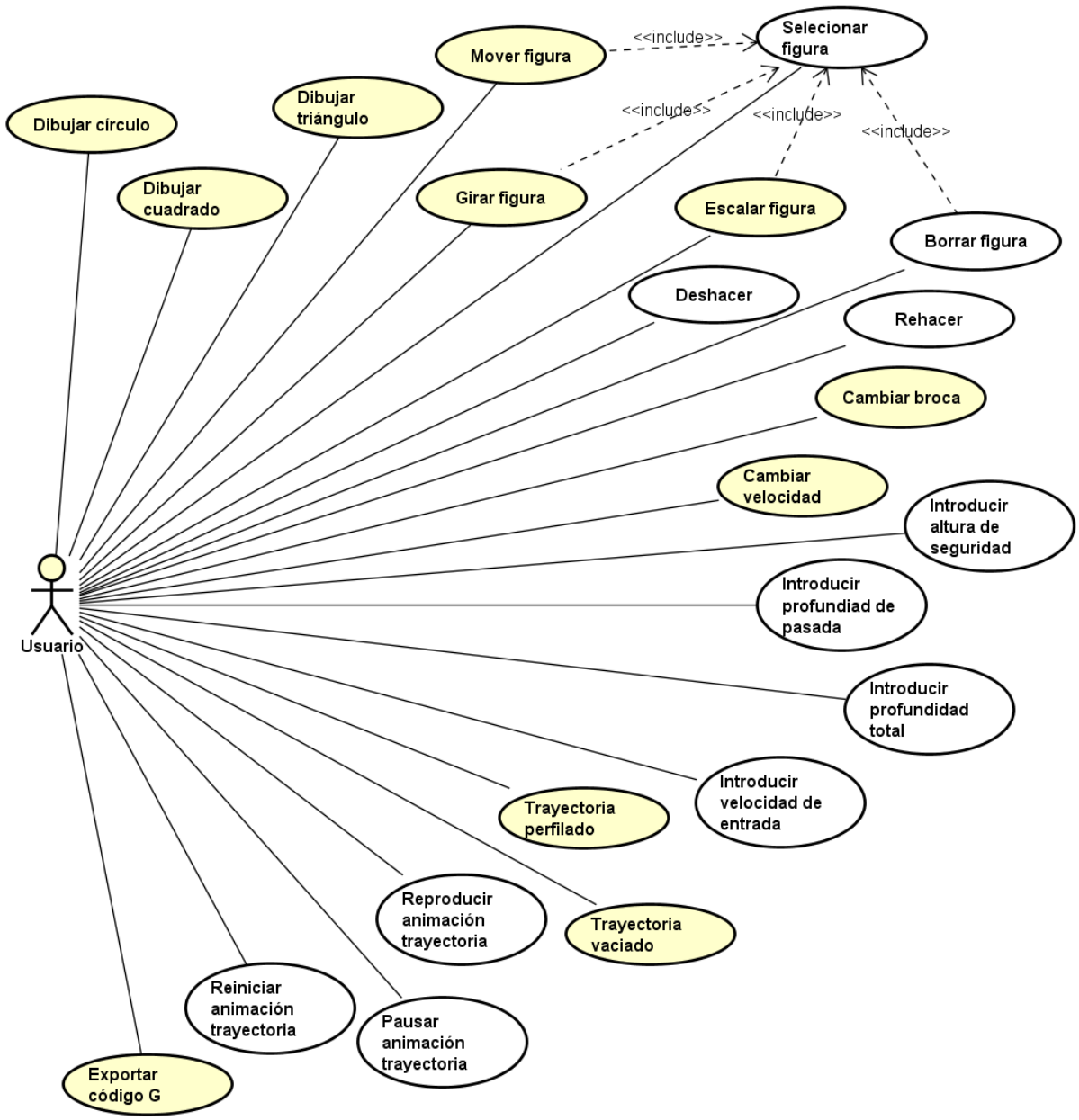

Figura 12. Diagrama de casos de uso. Segunda iteración

2.7 Matriz de incidencia requisitos/casos de uso

	RF01	RF02	RF03	RF04	RF05	RF06
CU01	✓	✓				
CU02	✓	✓				
CU03	✓	✓				
CU04			✓			
CU05			✓			
CU06			✓			
CU07				✓		
CU08				✓		
CU09					✓	
CU10					✓	
CU11						✓

Tabla 8. Matriz de incidencia de la primera iteración

A continuación, se muestra la matriz de incidencia de la segunda iteración.

	RF 01	RF 02	RF 03	RF 04	RF 05	RF 06	RF 07	RF 08	RF 09	RF 10	RF 11	RF 12	RF 13	RF 14
CU 01	✓	✓										✓		
CU 02	✓	✓										✓		
CU 03	✓	✓										✓		
CU 04			✓											
CU 05			✓											
CU 06			✓											
CU 07				✓										

CU 08				✓										
CU 09					✓									
CU 10					✓									
CU 11						✓								
CU 12										✓				
CU 13										✓				
CU 14									✓					
CU 15									✓					
CU 16									✓					
CU 17									✓					
CU 18							✓	✓						
CU 19								✓						
CU 20								✓						
CU 21														✓
CU 22													✓	

Tabla 9. Matriz de incidencia. Segunda iteración

Capítulo 3

Planificación y seguimiento

3.1 Planificación

Para este proyecto se va a usar el método de desarrollo de software UPEdu. Este método consta de 4 fases; inicio, elaboración, construcción y transición. Cada una de las fases puede tener 1 o 2 iteraciones. A continuación, se detallan las actividades que se van a desarrollar en cada fase, así como el tiempo estimado que van a llevar cada una de ellas teniendo en cuenta las 300 horas que debe llevar un TFG.

Para la estimación de fechas se suponen 4 horas de trabajo al día de lunes a viernes y el 4/2/2019 como fecha de inicio

3.2 Fases

3.2.1 Inicio

Estimación: 25 horas
Fecha de inicio: 4/2/2019
Fecha de fin: 12/2/2019

1ª Iteración

Actividades:

- Planificación (5 horas)
- Elicitación de requisitos (3 horas)
- Análisis de requisitos y casos de uso (7 horas)
- Estudio de three js (10 horas)

3.2.2 Elaboración

Estimación: 50 horas

Fecha de inicio: 12/2/2019

Fecha de fin: 28/2/2019

1ª Iteración

Actividades:

- Modelo de dominio (10 horas)
- Realización en análisis de casos de uso (10 horas)
- Arquitectura del sistema (30 horas)
  - Estudio de los diferentes frameworks (10 horas)
  - Estudio del framework elegido (15 horas)
  - Documentación de la arquitectura (5 horas)

3.2.3 Construcción

Estimación: 200 horas

Fecha de inicio: 28/2/2019

Fecha de fin: 9/5/2019

1ª Iteración

Actividades:

- Desarrollo del software (110 horas)
- Pruebas del sistema (25 horas)
- Documentación de la memoria (15 horas)

2ª Iteración

Actividades:

- Desarrollo del software (30 horas)
- Pruebas del sistema (15 horas)
- Documentación de la memoria (5 horas)

3.2.4 Transición

Estimación: 25 horas

Fecha de inicio: 9/5/2019

Fecha de fin: 17/5/2019

1ª Iteración

Actividades:

- Documentación final del TFG (25 horas)

3.3 Diagrama de Gantt

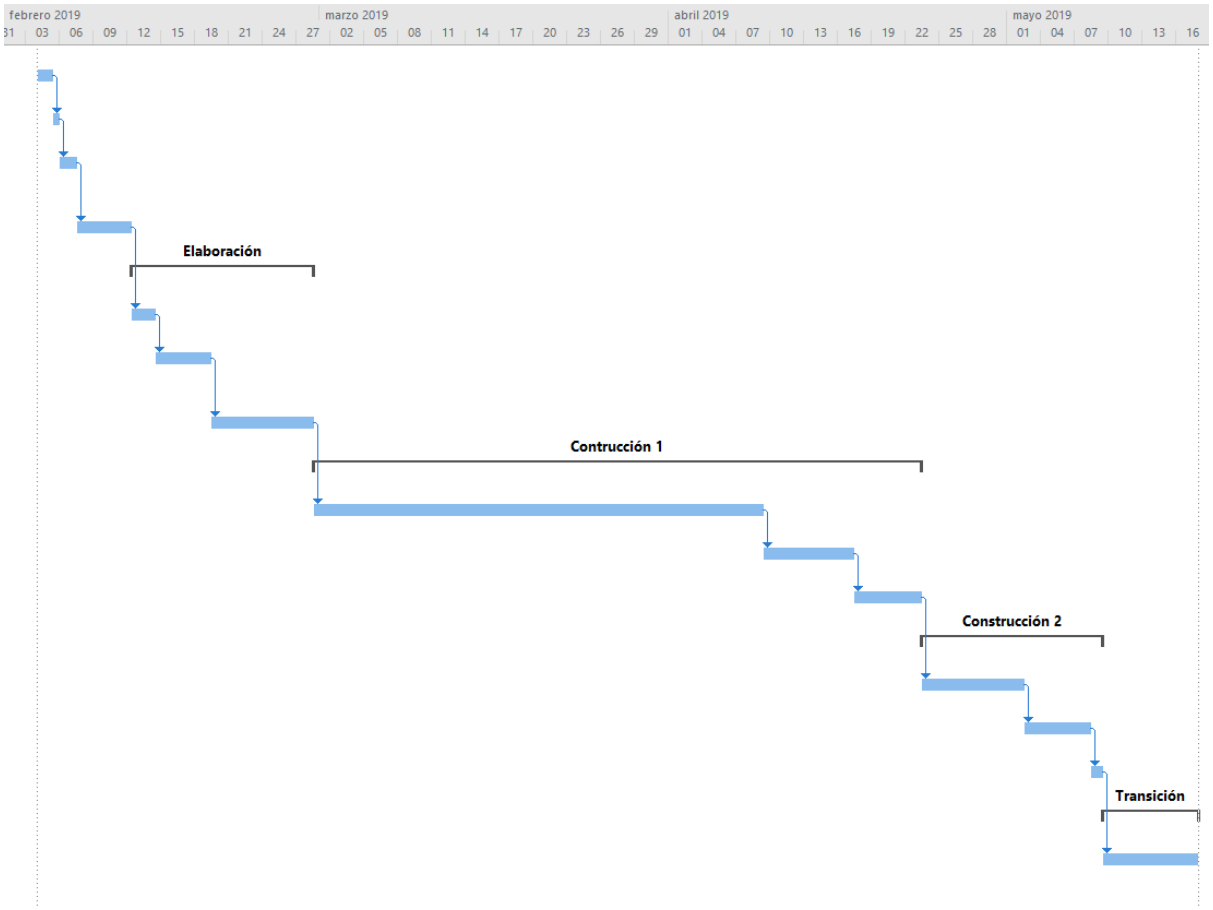

Figura 13. Diagrama de Gantt

3.4 Gestión de riesgos

Durante la realización del proyecto pueden surgir riesgos que lo pongan en peligro, por ello es una buena práctica listar los posibles riesgos y analizarlos con el objetivo de evitarlos o minimizar su impacto todo lo posible.

Cada riesgo detectado tendrá los siguientes atributos:

1. Nombre: Nombre del riesgo.
2. Descripción: Resumen del riesgo y sus características
3. Tipo: Determina de qué tipo es cada riesgo. Hay tres tipos, que se detallan a continuación.
    1. De proyecto: Ponen en riesgo el plan del proyecto. Pueden aumentar la planificación temporal y conllevar sobrecostes
    2. Técnico: Ponen en riesgo la calidad del proyecto y/o la planificación. Provocarían fallos en el diseño, interfaz, etc.

3. De producto: Ponen en riesgo la viabilidad de la aplicación; como que no tenga ninguna utilidad real o que no sea lo que satisfaga las necesidades del cliente.

1. Probabilidad: Estimación de la probabilidad de que ese riesgo suceda. Se va a indicar con un porcentaje.
2. Impacto: Señala la magnitud de las consecuencias si se produjera el riesgo. En la Tabla 10 se muestran los valores con los que se medirá el impacto previsto del riesgo en forma de escala ordinal de 1 a 4.

Catastrófico	4
Crítico	3
Marginal	2
Despreciable	1

Tabla 10. Impacto del riesgo

1. Consecuencia: Consecuencias que acarrearía la materialización del riesgo
2. Estrategia: Estrategia a usar con la intención de prevenir o paliar el riesgo
3. Plan de acción: Plan de acción en caso de que el riesgo suceda

Nombre	1. Enfermedad del alumno
Descripción	De enfermar el alumno el proyecto podría sufrir demoras por tener que parar hasta recuperarse
Tipo	De proyecto
Probabilidad	50%
Impacto	2
Consecuencia	Retraso en el proyecto con respecto a la planificación prevista
Estrategia	Añadir un colchón a la planificación a la hora de estimar las horas para así intentar que la enfermedad afectase lo menos posible a la calendarización

Plan de acción	Actualizar la planificación con el retraso que se ha sufrido

Tabla 11. Riesgo 1

Nombre	2. Indisponibilidad alguna de las tutoras
Descripción	En algún momento puede necesitarse la ayuda de alguna de las tutoras para que solucione una duda y éstas no puedan resolverla por tener algún otro compromiso
Tipo	De proyecto
Probabilidad	30%
Impacto	3
Consecuencia	Posible retraso en el proyecto con respecto a la planificación prevista
Estrategia	Tener una buena comunicación para saber las fechas en las que no podrán atenderme
Plan de acción	Realizar otra actividad que no requiera de su intervención. Si no fuera posible, actualizar la planificación con el retraso que se ha sufrido

Tabla 12. Riesgo 2

Nombre	3. Indisponibilidad del cliente
Descripción	En algún momento se puede necesitar quedar con el cliente (enseñarle artefactos, verificar requisitos, etc.) y que el cliente no esté disponible
Tipo	De proyecto
Probabilidad	75%
Impacto	1

Consecuencia	No debería causar grandes consecuencias. Una ligera demora como mucho
Estrategia	Acordar una cita con cierta anterioridad para asegurar la disponibilidad
Plan de acción	Acordar una cita lo antes posible

Tabla 13. Riesgo 3

Nombre	4. Avería en el ordenador
Descripción	El ordenador podría estropearse o dejar de funcionar por diferentes factores. Las consecuencias serían nefastas para poder seguir con el proyecto
Tipo	De proyecto
Probabilidad	10%
Impacto	4
Consecuencia	Imposibilidad de seguir con el proyecto en el mismo terminal y una posible pérdida de información
Estrategia	Realizar un control de versiones y backups en un repositorio remoto para salvar el proyecto
Plan de acción	Usar otro ordenador disponible (del laboratorio general de la universidad, por ejemplo)

Tabla 14. Riesgo 4

Nombre	5. Cambio en los requisitos del cliente
Descripción	El cliente puede cambiar de opinión durante el desarrollo y querer cambiar o añadir algún requisito
Tipo	De producto
Probabilidad	15%
Impacto	4

Consecuencia	Es probable que conlleve graves retrasos, una reestructuración de la planificación y aumento del presupuesto
Estrategia	Ir enseñando al cliente el producto durante el desarrollo para asegurarnos de que es realmente lo que quiere y si quiere añadir o cambiar algún requisito detectarlo cuanto antes para así intentar minimizar el impacto
Plan de acción	Evaluar si es necesaria una reestructuración de la planificación para cobijar el nuevo requisito y realizarla si procede

Tabla 15. Riesgo 5

Nombre	6. Mala planificación
Descripción	Debido a la poca experiencia en planificación de proyectos se ha podido cometer algún error de planificación y que ésta perjudique la realización del proyecto
Tipo	De proyecto
Probabilidad	70%
Impacto	3
Consecuencia	Retrasos en la realización del proyecto
Estrategia	Mantener una relación fluida con las tutoras con los avances en la planificación con el objetivo de que detecten errores en ella
Plan de acción	Corregir los errores de planificación que se hayan cometido

Tabla 16. Riesgo 6

Nombre	7. Mal diseño
Descripción	Puede tener lugar alguna mala decisión de diseño del proyecto
Tipo	Técnico

Probabilidad	60%
Impacto	2
Consecuencia	Retrasos en la realización del proyecto
Estrategia	Mantener una relación fluida con las tutoras con los avances en el diseño con el objetivo de que detecten errores en él
Plan de acción	Corregir los errores de diseño que se hayan cometido

Tabla 17. Riesgo 7

Nombre	8. Complicaciones con las tecnologías de desarrollo
Descripción	Dada la inexperiencia con three js y JavaScript es bastante posible que surjan complicaciones a la hora de usarlos
Tipo	Técnico
Probabilidad	80%
Impacto	2
Consecuencia	Retrasos en la realización del proyecto
Estrategia	Realizar un exhaustivo estudio de las tecnologías
Plan de acción	Búsqueda de información adicional por internet

Tabla 18. Riesgo 8

Nombre	9. Pérdida de información
Descripción	Por algún fallo en el ordenador podrían perderse avances en el proyecto o incluso éste entero
Tipo	De proyecto
Probabilidad	30%
Impacto	4

Consecuencia	Posible pérdida del proyecto o parte de él y obligación de rehacer todo el trabajo perdido
Estrategia	Realizar copia de seguridad tanto de la memoria y archivos relacionados como del proyecto con el código de la aplicación
Plan de acción	Recuperar la última versión del proyecto y trabajar a partir de ahí, rehaciendo el trabajo perdido

Tabla 19. Riesgo 9

Nombre	10. Caída de la red
Descripción	Es posible que haya alguna caída de la red, que dependiendo del tiempo que dure podría afectar al seguimiento de la planificación
Tipo	De proyecto
Probabilidad	20%
Impacto	1
Consecuencia	Imposibilidad de seguir avanzando con el proyecto
Estrategia	Es un riesgo ajeno al proyecto. No se puede prever ni evitar
Plan de acción	Desplazarse a otro lugar con red hasta que se solucione la incidencia (A la UVa, por ejemplo)

Tabla 20. Riesgo 10

3.5 Presupuesto

3.5.1 Costes hardware

Para el desarrollo de la aplicación se necesita de un ordenador con conexión a internet. El modelo que se usará es el del propio estudiante, un Samsung 305V5A. Para calcular el coste que supone únicamente durante el desarrollo del proyecto se calculará la parte proporcional a los 5 meses que se ha estimado que durará y una vida útil del portátil de 4 años (48 meses).

Concepto	Coste
Samsung 305V5A	(700 € / 48 meses) * 5 meses = 72.9 €
TOTAL	**72.9 €**

Tabla 21. Costes hardware

3.5.2 Costes software

Aquí se detallan las licencias necesarias para el desarrollo del proyecto. Microsoft Project es necesario para realizar el diagrama de Gantt de la planificación y el Astah UML para la realización de los diagramas de casos de uso, actividad, modelo de dominio, etc. No se añade la licencia del procesador de textos que utilizo puesto que es Google docs. para compartir en tiempo real mi trabajo con las tutoras y es gratuito.

Concepto	Coste
Microsoft Project Professional 2019	1509 €
Astah UML	125 $ (110.6 €)
TOTAL	**1619.6 €**

Tabla 22. Costes software

3.5.3 Costes de recursos humanos

Suponiendo las 300 horas estimadas para el desarrollo del TFG y el sueldo medio de un ingeniero informático se calcula el coste de los honorarios. Acorde al Boletín Oficial de la Provincia de Valladolid del 6 de febrero de 2018 y la revisión salarial según el convenio colectivo de oficinas y despachos de la provincia de Valladolid [8], el sueldo base de un titulado superior (nivel 1) es de 27567,40 €/año. Lo que supone aproximadamente un salario de 16 € la hora.

Concepto	Coste
Sueldo de ingeniero informático	16 € * 300 horas = 4800 €
TOTAL	**4800 €**

Tabla 23. Costes de recursos humanos

3.5.4 Costes totales

Sumando los anteriores costes obtenemos el total de costes presupuestados para el proyecto.

Concepto	Coste
Costes hardware	72.9 €

Costes software	1619.6 €
Costes de recursos humanos	4800 €
TOTAL	**6492.5 €**

Tabla 24. Costes totales

Se decide aplicar un colchón ya que, dada la naturaleza del trabajo, las horas estimadas pueden no ser suficientes y se puede acabar demorando. Por ello, se aplica un colchón para prevenir una demora de 50 horas sobre la planificación inicial. Esto supondría 800€ más, dejando los costes de recursos humanos en 5600€ y el total en 7292.5€.

3.6 Seguimiento

En esta sección se hace tanto el seguimiento de las horas hombre (De ahora en adelante, HH) reales que ha tomado realizar cada una de las tareas de las fases del proceso, como las variaciones de la planificación inicial que se hayan tenido que realizar por diversos motivos.

3.6.1 Ajustes de la planificación

Al final de la primera iteración de la fase de Construcción se tuvo una reunión con el cliente para mostrarle el progreso del desarrollo del software y así recibir su retroalimentación, con el objetivo de que el resultado final se ajustara lo más posible a la visión que tenían del producto inicialmente. Como resultado de esto, se obtuvieron algunos requisitos a mayores de los obtenidos en la primera iteración de la fase de Inicio. Por ello ha sido necesario rehacer la planificación inicial, añadiendo una segunda iteración tanto en las fases de Inicio como de Elaboración (también es necesaria una segunda iteración en la fase de Construcción, pero ya estaba contemplada en la planificación inicial.

A continuación, se detalla el resultado de la planificación rehecha:

3.6.1.1 Inicio

Estimación: 35 horas

1ª Iteración

Actividades:

- Planificación (5 horas)
- Elicitación de requisitos (3 horas)
- Análisis de requisitos y casos de uso (7 horas)
- Estudio de three js (10 horas)

2ª Iteración

Actividades

- Planificación (5 horas)
- Análisis de requisitos y casos de uso (5 horas)

3.6.1.2 Elaboración

Estimación: 65 horas

1ª Iteración

Actividades:

- Modelo de dominio (10 horas)
- Realización en análisis de casos de uso (10 horas)
- Arquitectura del sistema (30 horas)
  - Estudio de los diferentes frameworks (10 horas)
  - Estudio del framework elegido (15 horas)
  - Documentación de la arquitectura (5 horas)

2ª Iteración

Actividades:

- Realización en análisis de casos de uso (5 horas)
- Documentación de la arquitectura (10 horas)

3.6.1.3 Construcción

Estimación: 200 horas

1ª Iteración

Actividades:

- Desarrollo del software (110 horas)
- Pruebas del sistema (25 horas)
- Documentación de la memoria (15 horas)

2ª Iteración

Actividades:

- Desarrollo del software (30 horas)
- Pruebas del sistema (15 horas)
- Documentación de la memoria (5 horas)

3.6.1.4 Transición

Estimación: 25 horas

1ª Iteración

Actividades:

- Documentación final del TFG (25 horas)

3.6.1.5 Diagrama de Gantt

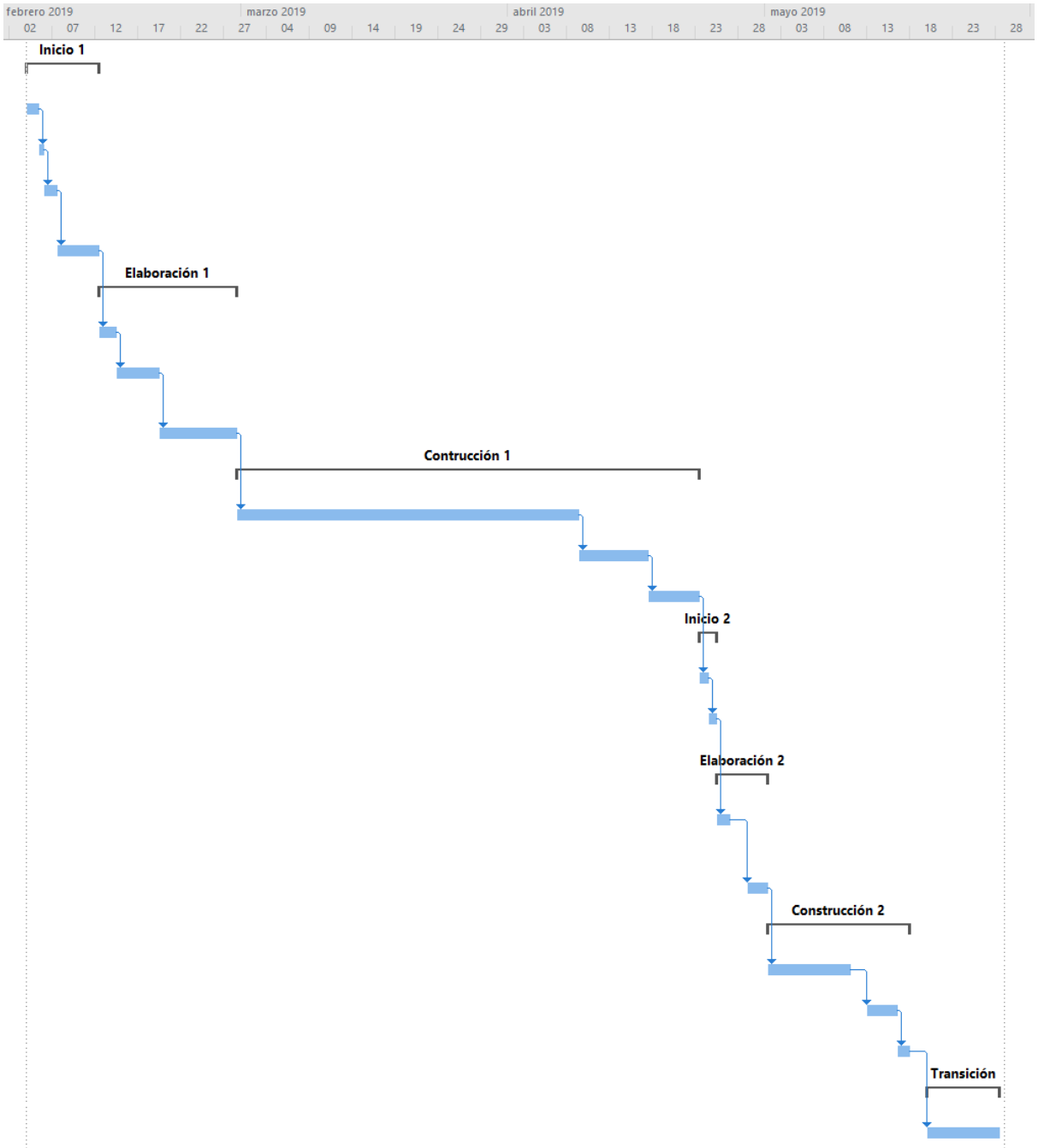

Figura 14. Diagrama de Gantt. Segunda iteración

Como resultado de haber rehecho la planificación, se han añadido 25 horas con respecto a la planificación inicial. Quedando en un total de 325 horas.

3.6.2 HH reales frente estimadas

El seguimiento de las HH se ha realizado con la finalidad de compararlas con las estimaciones hechas en la planificación y así analizar las diferencias que pudiera haber, para así corregir las desviaciones para próximos proyectos.

Para poder afinar más en el análisis se presentarán primero cada una de las fases con sus tareas desgranadas, permitiendo observar mejor dónde exactamente se presentan las mayores diferencias entre las horas/hombre estimadas y las reales.

Al final de este apartado se presenta un recuento total más general con todas las fases para realizar una valoración en conjunto del seguimiento.

3.6.2.1 Inicio

Tareas	**Horas estimadas**	**Horas reales**	**Diferencia**
1ª Iteración			
Planificación	5	12	+7
Elicitación de requisitos	3	3	0
Análisis de requisitos y casos de uso	7	4	-3
Estudio de Three js	10	10	0
2ª Iteración			
Planificación	5	3	-2
Análisis de requisitos y casos de uso	5	3	-2
		TOTAL	**0**

Tabla 25. HH reales frente a estimadas. Fase de Inicio

En la primera iteración es destacable la gran diferencia en las horas que realmente se han necesitado para realizar la planificación del proyecto con respecto a las que se habían estimado. Pero en general no hay un gran desvío, apenas un 7.25% más del tiempo estimado.

Conociendo cómo fue la estimación de la primera iteración, se hizo la de la segunda intentando ajustarse más, pecando de pesimista en la estimación para tener margen de maniobra por si por algún motivo se necesitase más tiempo para alguna de las tareas. No fue el caso y se ha acabado la iteración con 4 menos de las estimadas.

Como balance final de la fase, curiosamente, ha acabado con las mismas horas estimadas y reales.

3.6.2.2 Elaboración

Tareas	Horas estimadas	Horas reales	Diferencia
1ª Iteración			
Modelo de dominio	10	5	-5
Realización en análisis de casos de uso	10	2	-8
Estudio de los diferentes frameworks	10	8	-2
Estudio del framework elegido	15	25	+10
Documentación de la arquitectura	5	2	-3
2ª Iteración			
Realización en análisis de casos de uso	5	3	-2
Documentación de la arquitectura	10	8	-2
		TOTAL	**-12**

Tabla 26. HH reales frente a estimadas. Fase de Elaboración

En la primera iteración de esta fase hay una gran disparidad de diferencias. Mientras que el estudio del framework elegido ha llevado casi el doble de lo esperado, la estimación de la mayoría de las tareas ha sido bastante pesimista (se creía que llevarían bastante más de lo que en realidad han supuesto), por lo que compensan con creces y acaba habiendo un balance positivo, 8 horas menos de las estimadas.

Al igual que en la fase anterior, sabiendo cómo fue la primera iteración se ha intentado ajustar más la planificación a lo que realmente fue, pecando un poco de pesimista para tener margen si alguna tarea se complicase. Finalmente, la estimación fue correcta, sobrando 2 hora para cada una de las tareas, 4 horas en total.

La fase en general tiene un balance muy positivo, habiendo necesitado 12 horas menos para realizar las tareas de lo que se había estimado.

3.6.2.3 Construcción

Tareas	Horas estimadas	Horas reales	Diferencia
1ª Iteración			
Desarrollo del software	110	107	-3
Pruebas del sistema	25	20	-5
Documentación de la memoria	15	8	-7
2ª Iteración			
Desarrollo del software	30	8	-22
Pruebas del sistema	15	11	-4
Documentación de la memoria	5	4	-1
		TOTAL	**-42**

Tabla 27. HH reales frente a estimadas. Fase de Construcción

Se puede apreciar que para la primera iteración de esta fase se hizo una estimación muy pesimista, puesto que las tres tareas acaban en menos horas de las estimadas. Lo que nos deja un balance muy positivo, 15 horas menos.

En la segunda iteración se ha hecho un balance demasiado pesimista, especialmente en la tarea del desarrollo de software, puesto que se han necesitado 27 horas menos de las 50 planificadas.

En general, esta fase ha sido la mejor estimada, puesto que no se ha sobrepasado el tiempo estimado en ninguna de las tareas, a pesar de ser la fase más extensa y, por lo tanto, la más complicada de estimar.

3.6.2.4 Transición

Tareas	Horas estimadas	Horas reales	Diferencia
Documentación final del TFG	25	40	+15
		TOTAL	+15

Tabla 28. HH reales frente a estimadas. Fase de Transición

Se ha necesitado bastante más tiempo para la documentación final de la memoria, concretamente para las correcciones. Esto nos deja con 15 horas por encima de las estimadas inicialmente.

3.6.2.5 Total

Fases	Horas estimadas	Horas reales	Diferencia
Inicio	35	35	0
Elaboración	65	53	-12
Construcción	200	158	-42
Transición	25	40	+15
		TOTAL	-39

Tabla 29. HH reales frente a estimadas. Total

Finalmente, se acaba con un balance muy positivo en el que se puede apreciar la eficacia de una planificación pesimista, logrando así no sobrepasar las horas inicialmente estimadas. Se ha logrado realizar el proyecto con 39 horas menos de las estimadas, es decir, se ha realizado en 286 horas en vez de en 325.

Capítulo 4

Análisis de requisitos

4.1 Casos de uso

CU01	**Dibujar círculo**
Versión	1.0
Dependencias	RF01 y RF02
Descripción	El usuario quiere dibujar un círculo en el lienzo
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica que quiere dibujar un círculo 2. El sistema crea un círculo en el lienzo de radio y posición predeterminadas y lo añade a la lista de figuras dibujadas 3. El caso de uso finaliza
Postcondición	Hay un círculo dibujado en el lienzo de trabajo
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 30. CU01

CU02	**Dibujar cuadrado**
Versión	1.0
Dependencias	RF01 y RF02
Descripción	El usuario quiere dibujar un cuadrado en el lienzo
Actor primario	Usuario

Precondición	Ninguna
Flujo básico	1. El usuario indica que quiere dibujar un cuadrado 2. El sistema crea un cuadrado en el lienzo de longitud del lado y posición predeterminadas y lo añade a la lista de figuras dibujadas 3. El caso de uso finaliza
Postcondición	Hay un cuadrado dibujado en el lienzo de trabajo
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 31. CU02

CU03	**Dibujar triángulo**
Versión	1.0
Dependencias	RF01 y RF02
Descripción	El usuario quiere dibujar un triángulo en el lienzo
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica que quiere dibujar un triángulo 2. El sistema crea un triángulo en el lienzo de longitud de lado y posición predeterminadas y lo añade a la lista de figuras dibujadas 3. El caso de uso finaliza
Postcondición	Hay un triángulo dibujado en el lienzo de trabajo
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 32. CU03

CU04	**Mover figura**
Versión	1.0
Dependencias	RF03
Descripción	El usuario quiere mover la figura de posición
Actor primario	Usuario
Precondición	Tiene que haber una figura dibujada en el lienzo

Flujo básico	1. Se realiza el CU Seleccionar figura 2. El usuario indica las coordenadas a las que quiere mover el centro de la figura 3. El sistema comprueba que sean unas coordenadas correctas 4. El sistema actualiza las coordenadas del centro de la figura y muestra la nueva posición 5. El caso de uso finaliza
Postcondición	La figura se ha movido a la posición indicada por el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Coordenadas incorrectas
	1. Se muestra un mensaje de error indicando que las coordenadas son incorrectas porque no se ha introducido alguna de ellas 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Media

Tabla 33. CU04

CU05	**Girar figura**
Versión	1.0
Dependencias	RF03
Descripción	El usuario quiere girar la figura
Actor primario	Usuario
Precondición	Tiene que haber una figura dibujada en el lienzo
Flujo básico	1. Se realiza el CU Seleccionar figura 2. El usuario indica los grados que quiere girar la figura 3. El sistema comprueba que sea un número válido 4. El sistema actualiza el ángulo de giro de la figura y lo muestra aplicándolo 5. El caso de uso finaliza
Postcondición	La figura se ha girado los grados indicados por el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Grados incorrectos
	3. Se muestra un mensaje de error indicando que los grados son incorrectos porque no se ha introducido el dato 4. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Media

Tabla 34. CU05

CU06	Escalar figura
Versión	1.0
Dependencias	RF03
Descripción	El usuario quiere cambiar el tamaño de la figura
Actor primario	Usuario
Precondición	Tiene que haber una figura dibujada en el lienzo
Flujo básico	1. Se realiza el CU Seleccionar figura 2. El usuario indica el tamaño que quiere que tenga la figura (En porcentaje con respecto a la figura original) 3. El sistema comprueba que los datos introducidos sean válidos 4. El sistema actualiza el tamaño de la figura y lo muestra aplicándolo 5. El caso de uso finaliza
Postcondición	La figura tiene el tamaño que ha indicado el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Tamaño incorrecto
	1. Se muestra un mensaje de error indicando que el tamaño es incorrecto porque no se ha introducido el dato o es negativo 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Media

Tabla 35. CU06

CU07	Trayectoria perfilado
Versión	1.0
Dependencias	RF04
Descripción	El usuario quiere calcular la trayectoria de perfilado del dibujo
Actor primario	Usuario
Precondición	Tiene que haber una forma dibujada en el lienzo
Flujo básico	1. El usuario indica que quiere obtener la trayectoria de perfilado 2. El sistema realiza los cálculos necesarios para obtener la trayectoria 3. El sistema muestra la trayectoria que seguirá la broca para realizar el perfilado 4. El caso de uso finaliza

Postcondición	Se muestra la trayectoria calculada
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 36. CU07

CU08	**Trayectoria vaciado**
Versión	1.0
Dependencias	RF04
Descripción	El usuario quiere calcular la trayectoria de vaciado del dibujo
Actor primario	Usuario
Precondición	Tiene que haber una forma dibujada en el lienzo
Flujo básico	1. El usuario indica que quiere obtener la trayectoria de vaciado 2. El sistema realiza los cálculos necesarios para obtener la trayectoria 3. El sistema muestra la trayectoria que seguirá la broca para realizar el vaciado 4. El caso de uso finaliza
Postcondición	Se muestra la trayectoria calculada
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 37. CU08

CU09	**Cambiar broca**
Versión	1.0
Dependencias	RF05
Descripción	El usuario quiere cambiar el tamaño de la broca
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica el tamaño que quiere que tenga la

	broca 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza el tamaño de la broca que se va a utilizar 4. El caso de uso finaliza
Postcondición	La broca tiene el tamaño indicado por el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Tamaño incorrecto
	1. Se muestra un mensaje de error indicando que el tamaño es incorrecto porque no se ha introducido el dato o es negativo 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Media

Tabla 38. CU09

CU10	**Cambiar velocidad**
Versión	1.0
Dependencias	RF05
Descripción	El usuario quiere cambiar la velocidad de fresado
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica la velocidad de fresado que desea 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza la velocidad de fresado que se va a utilizar 4. El caso de uso finaliza
Postcondición	La velocidad ha cambiado a la que ha indicado el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Velocidad incorrecta
	1. Se muestra un mensaje de error indicando que la velocidad introducida es incorrecta porque no se ha introducido el dato o es negativo 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Media

Tabla 39. CU10

CU11	Exportar código G
Versión	1.0
Dependencias	RF06
Descripción	El sistema exportará el código G de la trayectoria previamente calculada
Actor primario	Usuario
Precondición	Debe haberse calculado antes alguna trayectoria
Flujo básico	1. El usuario indica que quiere exportar el código G 2. El sistema genera el código y lo exporta en un archivo de texto 3. El usuario indica al navegador el nombre del archivo y dónde quiere guardarlo 4. Descargar archivo en la ruta especificada 5. El caso de uso finaliza
Postcondición	El navegador ha descargado un archivo de texto con el código G
Flujo alternativo	FA01 El usuario cancela la acción en el paso 3
	1. El caso de uso queda sin efecto
Excepciones	Ninguna
Frecuencia	Media

Tabla 40. CU11

A continuación, se presentan los casos de uso pertenecientes a la segunda iteración.

CU12	Deshacer
Versión	1.0
Dependencias	RF10
Descripción	El usuario quiere deshacer una acción
Actor primario	Usuario
Precondición	Tiene que haber una forma dibujada en el lienzo
Flujo básico	1. El usuario indica que quiere deshacer 2. El sistema comprueba que hay alguna acción que deshacer 3. El sistema deshace el último cambio y muestra el resultado

	4. El caso de uso finaliza
Postcondición	El dibujo está en el estado previo a la anterior acción que se hubiera realizado antes de empezar el caso de uso
Flujo alternativo	2. El sistema comprueba que no hay ningún cambio que se pueda deshacer 3. El caso de uso finaliza
Excepciones	Ninguna
Frecuencia	Baja

Tabla 41. CU12

CU13	**Rehacer**
Versión	1.0
Dependencias	RF10
Descripción	El usuario quiere rehacer una acción previamente deshecha
Actor primario	Usuario
Precondición	Tiene que haberse dado el caso de uso CU12 y no haberse realizado ninguna otra después
Flujo básico	1. El usuario indica que quiere rehacer 2. El sistema comprueba que hay alguna acción que rehacer 3. El sistema rehace la última acción deshecha y muestra el resultado 4. El caso de uso finaliza
Postcondición	El dibujo está en el estado previo a empezar el caso de uso CU12
Flujo alternativo	2. El sistema comprueba que no hay ningún cambio que se pueda rehacer 3. El caso de uso finaliza
Excepciones	Ninguna
Frecuencia	Muy baja

Tabla 42. CU13

CU14	**Introducir altura de seguridad**
Versión	1.0

Dependencias	RF09
Descripción	El usuario quiere introducir la altura de seguridad
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica la altura de seguridad que desea 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza la altura de seguridad que se va a utilizar 4. El caso de uso finaliza
Postcondición	La altura de seguridad ha cambiado a la que ha indicado el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Altura de seguridad incorrecta
	1. Se muestra un mensaje de error indicando que la altura de seguridad introducida es incorrecta porque no se ha introducido el dato o es negativo 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Alta

Tabla 43. CU14

CU15	**Introducir profundidad de pasada**
Versión	1.0
Dependencias	RF09
Descripción	El usuario quiere introducir la profundidad de pasada
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica la profundidad de pasada que desea 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza la profundidad de pasada que se va a utilizar 4. El caso de uso finaliza
Postcondición	La profundidad de pasada ha cambiado a la que ha indicado el usuario
Flujo alternativo	Ninguno

Excepciones	EX01 Profundidad de pasada incorrecta
	1. Se muestra un mensaje de error indicando que la profundidad de pasada introducida es incorrecta porque no se ha introducido el dato, es negativo o es mayor que la profundidad total 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Alta

Tabla 44. CU15

CU16	**Introducir profundidad total**
Versión	1.0
Dependencias	RF09
Descripción	El usuario quiere introducir la profundidad total
Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica la profundidad total que desea 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza la profundidad total que se va a utilizar 4. El caso de uso finaliza
Postcondición	La profundidad total ha cambiado a la que ha indicado el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Profundidad total incorrecta
	1. Se muestra un mensaje de error indicando que la profundidad total introducida es incorrecta porque no se ha introducido el dato, es negativo o es menor que la profundidad de pasada 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Alta

Tabla 45. CU16

CU17	**Introducir velocidad de entrada**
Versión	1.0
Dependencias	RF09
Descripción	El usuario quiere introducir la velocidad de entrada

Actor primario	Usuario
Precondición	Ninguna
Flujo básico	1. El usuario indica la velocidad de entrada que desea 2. El sistema comprueba que los datos introducidos sean válidos 3. El sistema actualiza la velocidad de entrada que se va a utilizar 4. El caso de uso finaliza
Postcondición	La velocidad de entrada ha cambiado a la que ha indicado el usuario
Flujo alternativo	Ninguno
Excepciones	EX01 Velocidad de entrada incorrecta
	1. Se muestra un mensaje de error indicando que la velocidad de entrada introducida es incorrecta porque no se ha introducido el dato o es negativo 2. El flujo vuelve al paso 1 del flujo básico
Frecuencia	Alta

Tabla 46. CU17

CU18	**Reproducir animación trayectoria**
Versión	1.0
Dependencias	RF08
Descripción	El usuario quiere reproducir la animación de la trayectoria
Actor primario	Usuario
Precondición	Debe haber una trayectoria calculada y la animación estar parada
Flujo básico	1. El usuario indica que quiere reproducir la animación de la trayectoria 2. El sistema comprueba si puede reproducir la animación 3. El sistema reproducir la animación 4. El caso de uso finaliza
Postcondición	Se ha reproducido la animación
Flujo alternativo	Ninguno
Excepciones	EX01 No estaba parada antes la animación
	1. El caso de uso queda sin efecto
Frecuencia	Media

Tabla 47. CU18

CU19	Pausar animación trayectoria
Versión	1.0
Dependencias	RF08
Descripción	El usuario quiere pausar la animación de la trayectoria
Actor primario	Usuario
Precondición	Debe haberse calculado una trayectoria y que esté animándose
Flujo básico	1. El usuario indica que quiere pausar la animación de la trayectoria 2. El sistema comprueba si puede pausar la animación 3. El sistema pausa la animación 4. El caso de uso finaliza
Postcondición	Se ha pausado la animación
Flujo alternativo	Ninguno
Excepciones	EX01 La animación ya estaba parada o había acabado
	1. El caso de uso queda sin efecto
Frecuencia	Media

Tabla 48. CU19

CU20	Reiniciar animación trayectoria
Versión	1.0
Dependencias	RF08
Descripción	El usuario quiere reiniciar la animación de la trayectoria
Actor primario	Usuario
Precondición	Debe haberse calculado una trayectoria previamente
Flujo básico	1. El usuario indica que quiere reiniciar la animación de la trayectoria 2. El sistema reinicia la animación 3. El caso de uso finaliza
Postcondición	Se ha reiniciado la animación
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Media

Tabla 49. CU20

CU21	Borrar figura
Versión	1.0
Dependencias	RF14
Descripción	El usuario quiere borrar una figura
Actor primario	Usuario
Precondición	Tiene que haber una figura dibujada en el lienzo
Flujo básico	1. Se realiza el CU Seleccionar figura 2. El usuario indica que quiere borrar la figura 3. El sistema borra la figura 4. El caso de uso finaliza
Postcondición	La figura se ha borrado del lienzo de dibujo
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Media

Tabla 50. CU21

CU22	Seleccionar figura
Versión	1.0
Dependencias	RF13
Descripción	El usuario quiere seleccionar una figura
Actor primario	Usuario
Precondición	Tiene que haber una figura dibujada en el lienzo
Flujo básico	1. El usuario indica que quiere seleccionar una figura 2. El sistema selecciona la figura y lo muestra 3. El caso de uso finaliza
Postcondición	La figura está seleccionada
Flujo alternativo	Ninguno
Excepciones	Ninguna
Frecuencia	Alta

Tabla 51. CU22

4.2 Diagramas de actividad

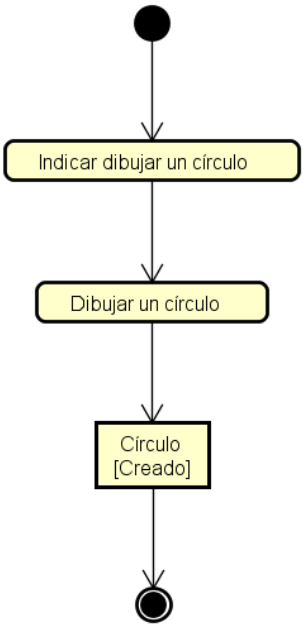

Figura 15. CU01. Dibujar círculo

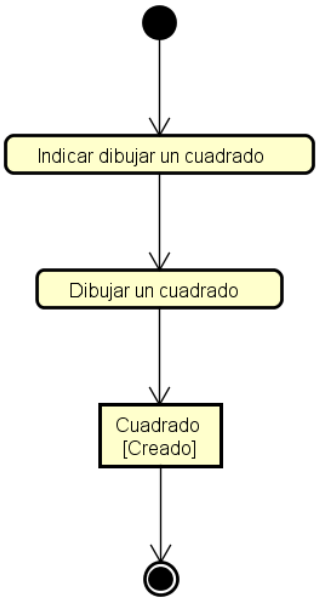

Figura 16. CU02. Dibujar cuadrado

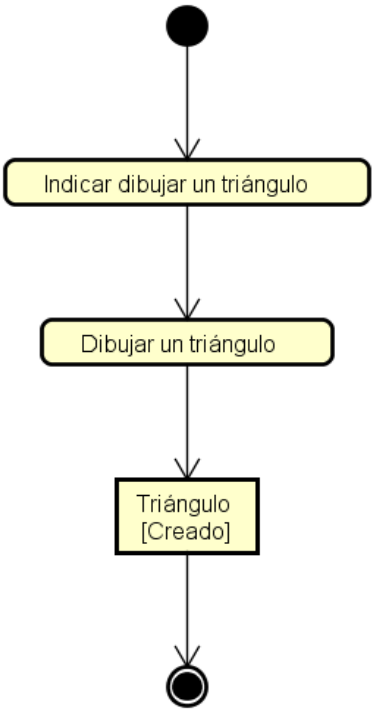

Figura 17. CU03. Dibujar triángulo

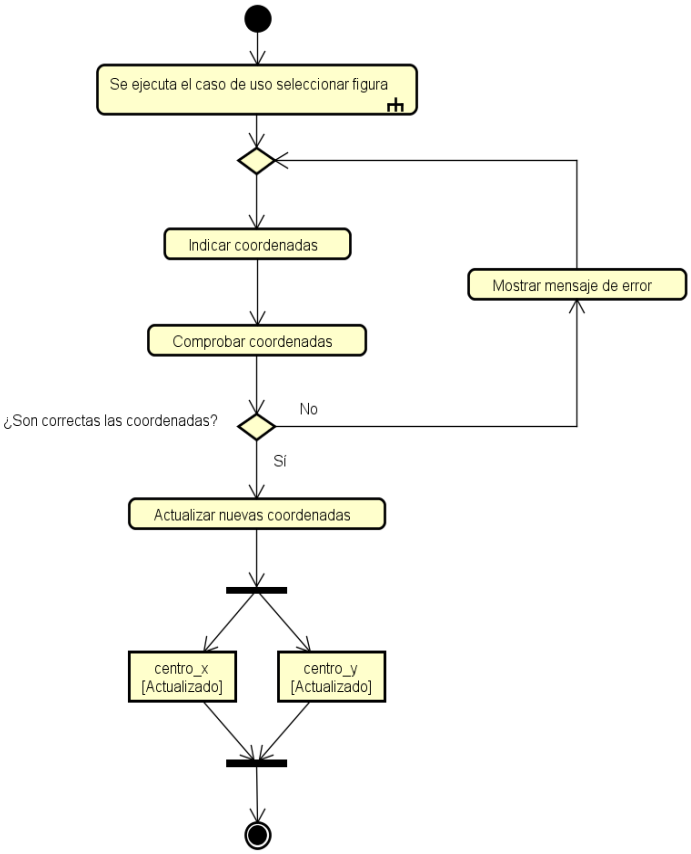

Figura 18. CU04. Mover figura

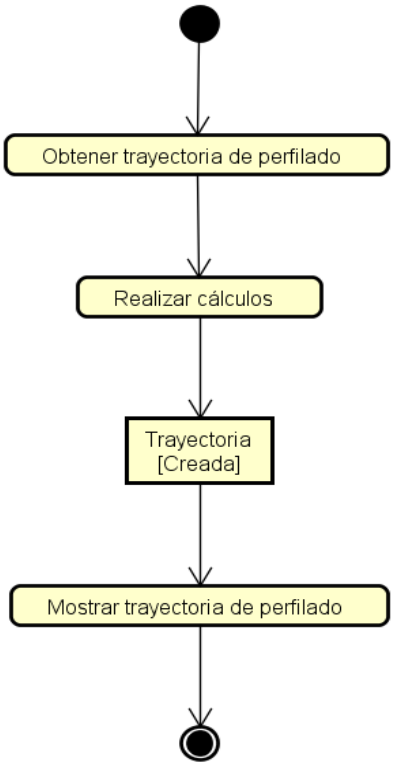

Figura 19. CU07. Trayectoria perfilado

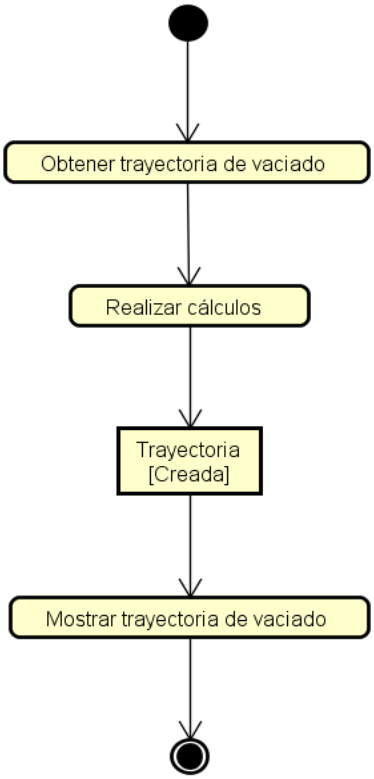

Figura 20. CU08 Trayectoria vaciado

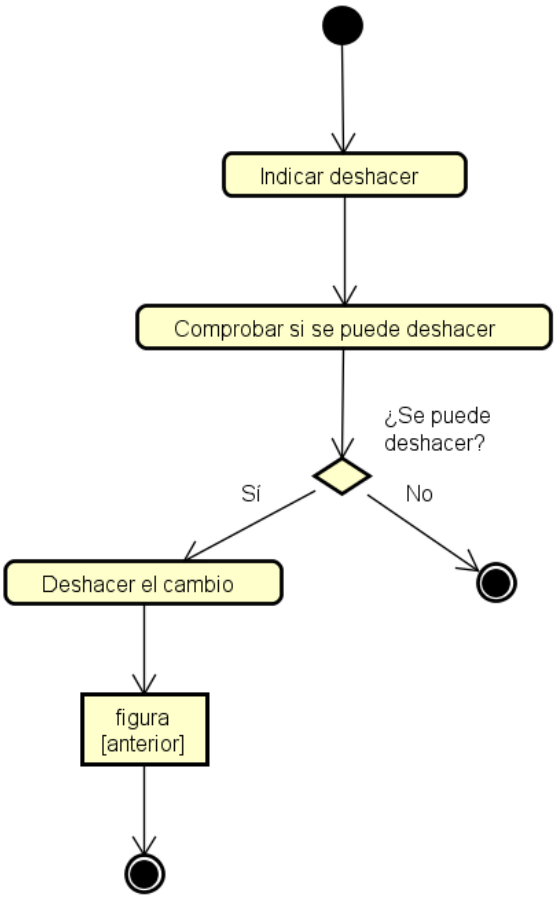

Figura 21. CU12. Deshacer

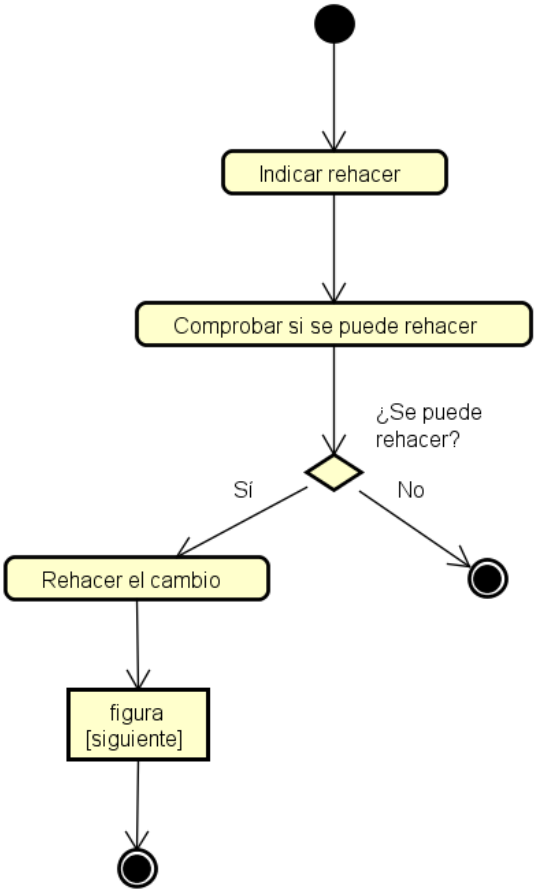

Figura 22. CU13. Rehacer

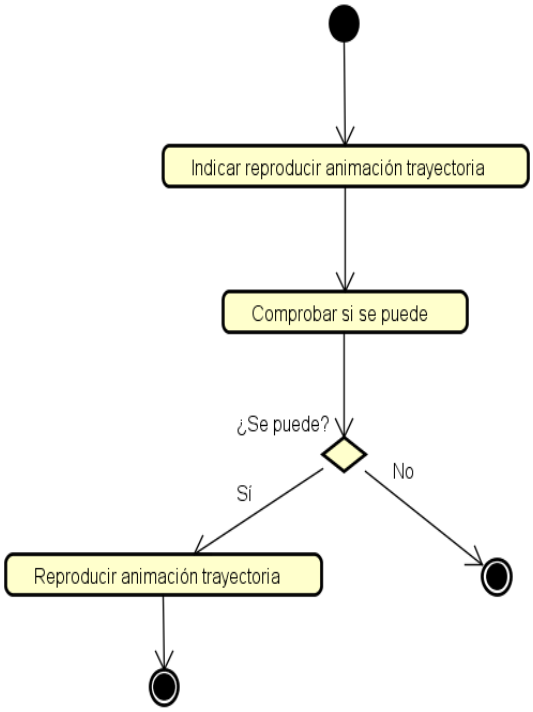

Figura 23. CU18. Reproducir animación trayectoria

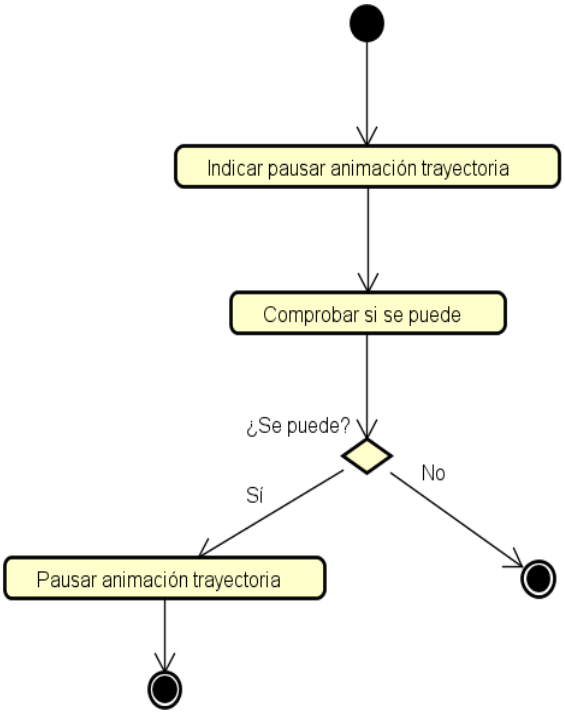

Figura 24. CU19. Pausar animación trayectoria

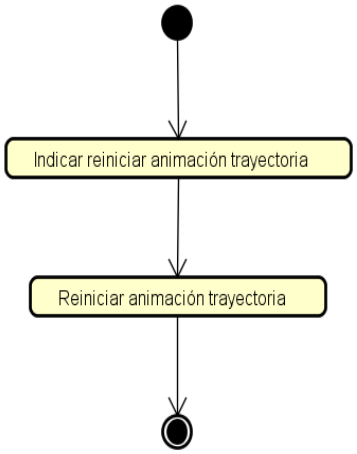

Figura 25. CU20. Reiniciar animación trayectoria

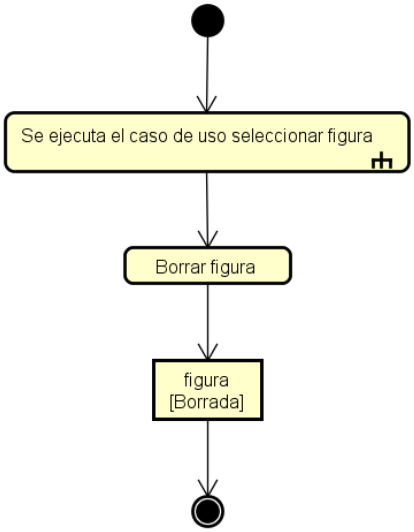

Figura 26. CU21 Borrar figura

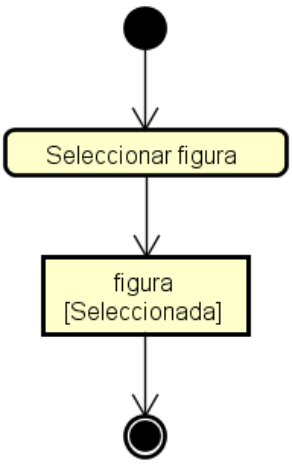

Figura 27. CU22 Seleccionar figura

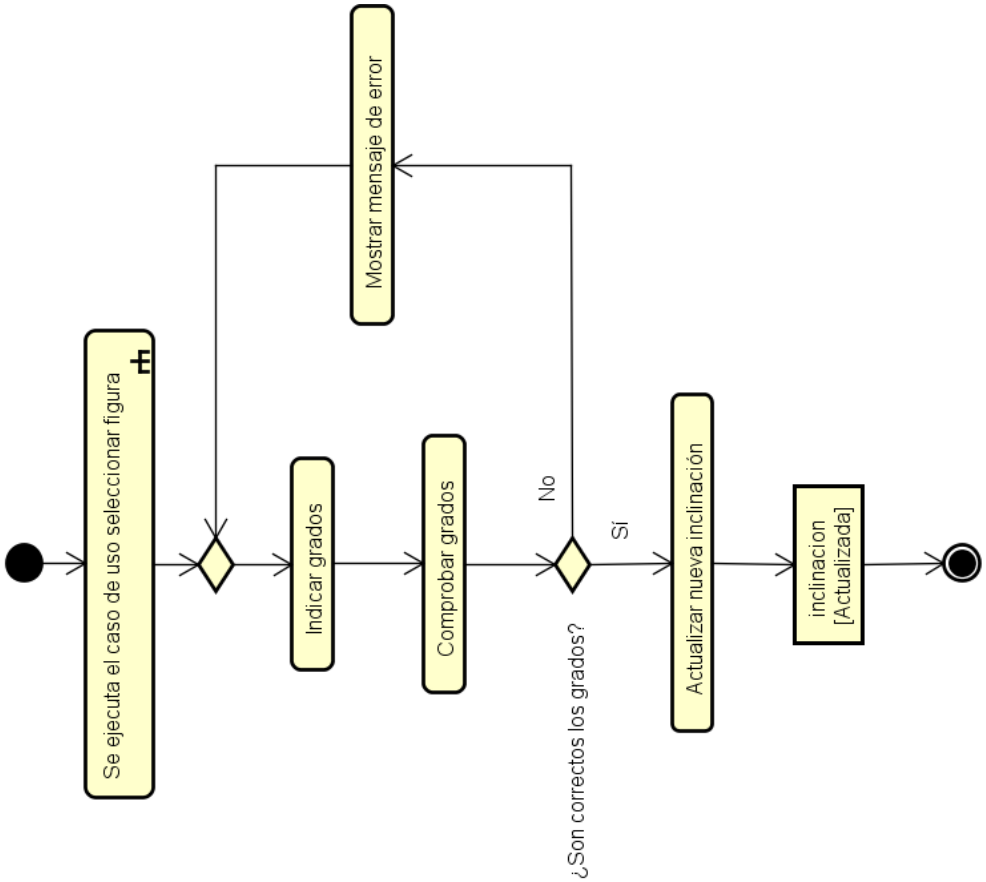

Figura 28. CU05. Girar figura

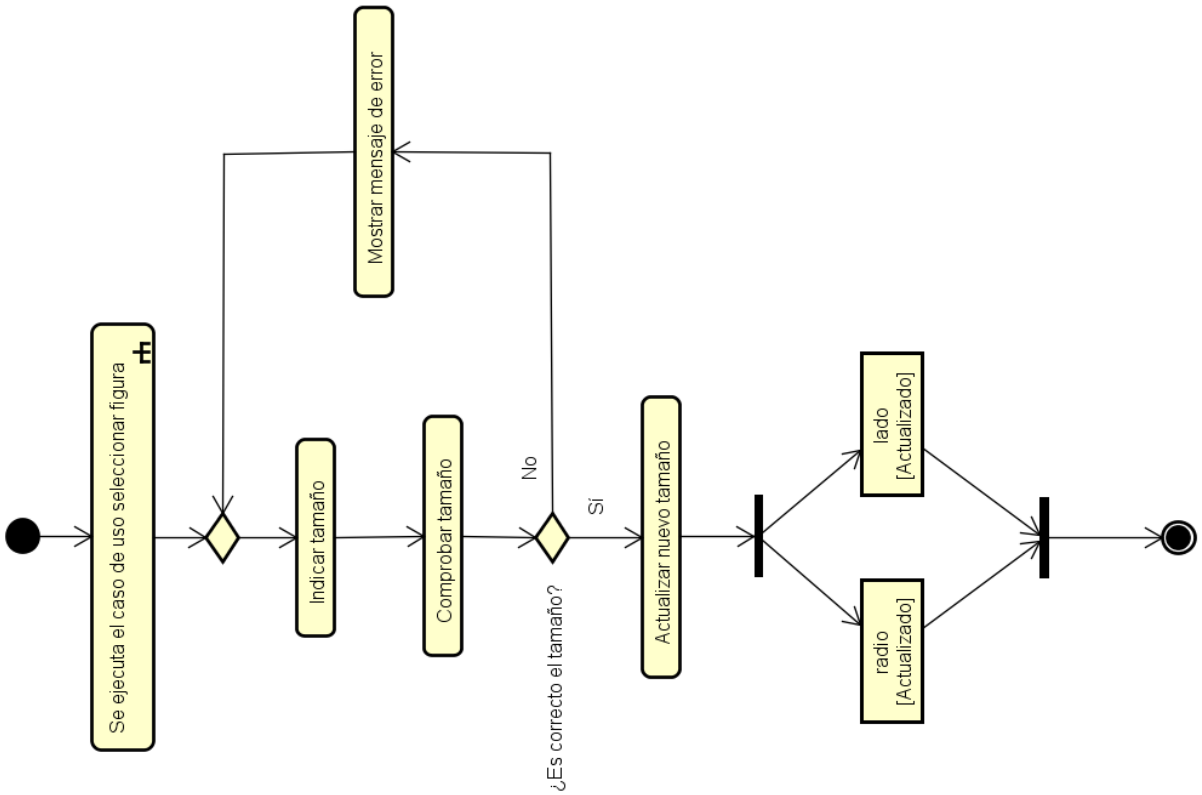

Figura 29. CU06. Escalar figura

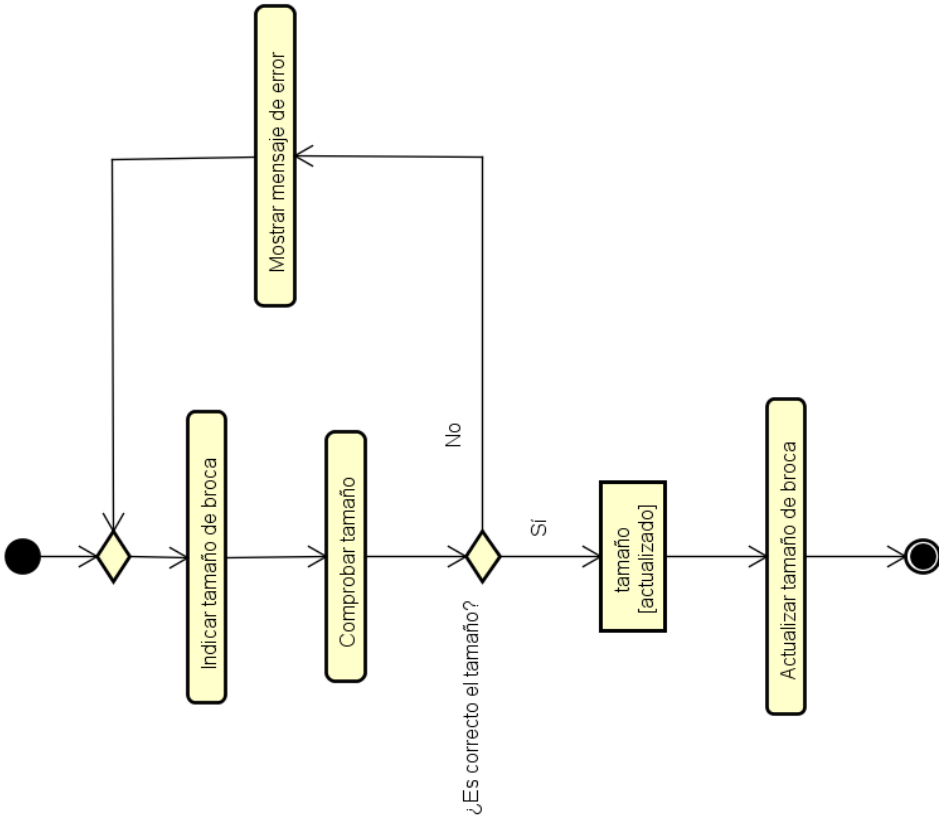

Figura 30. CU09. Cambiar broca

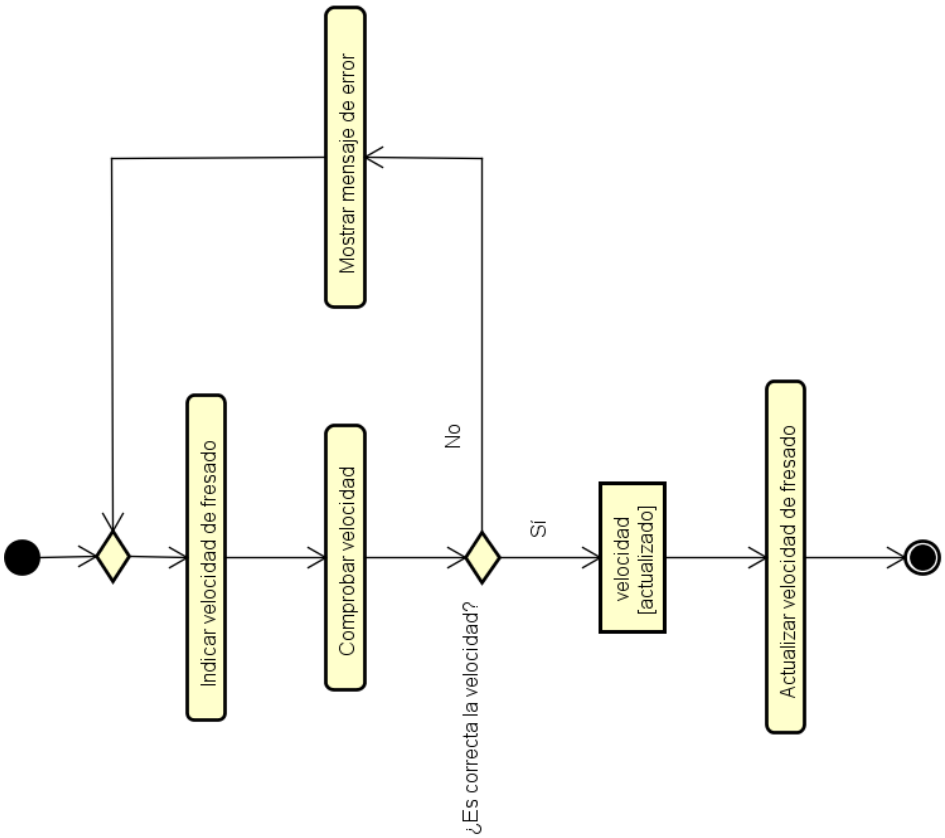

Figura 31. CU10. Cambiar velocidad

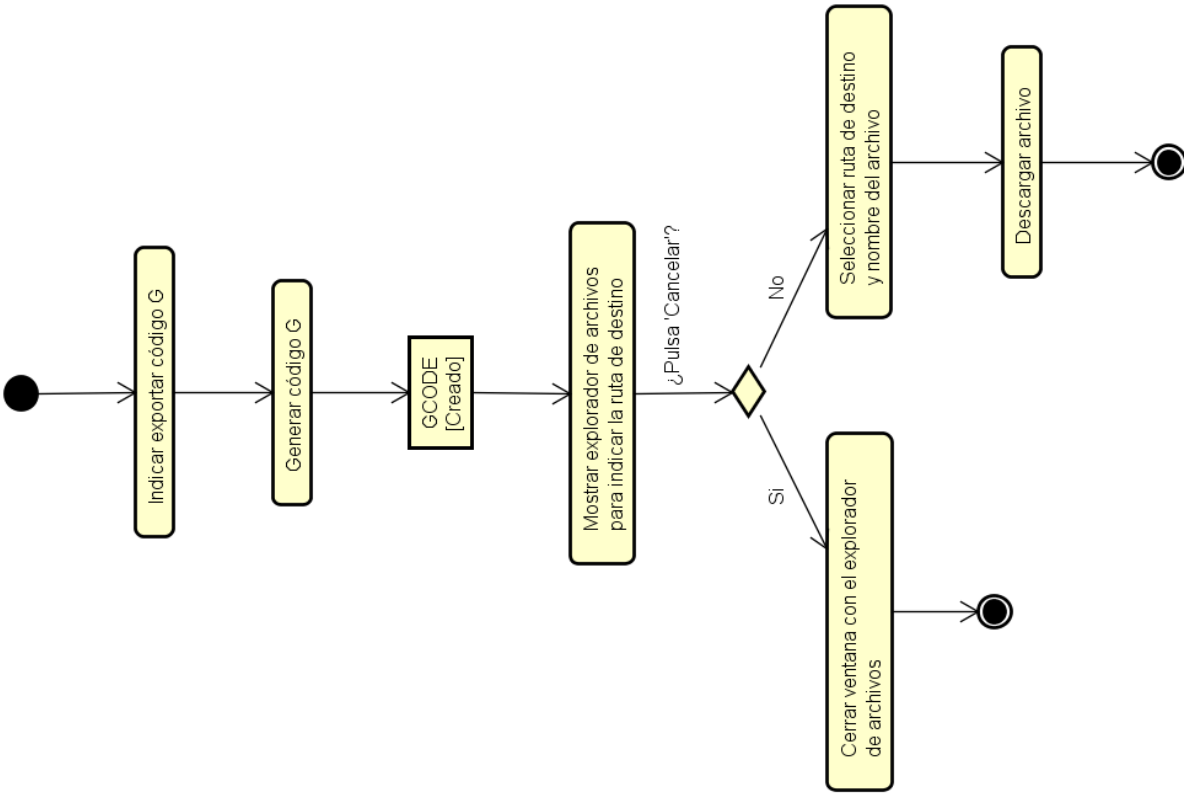

Figura 32. CU11. Exportar código G

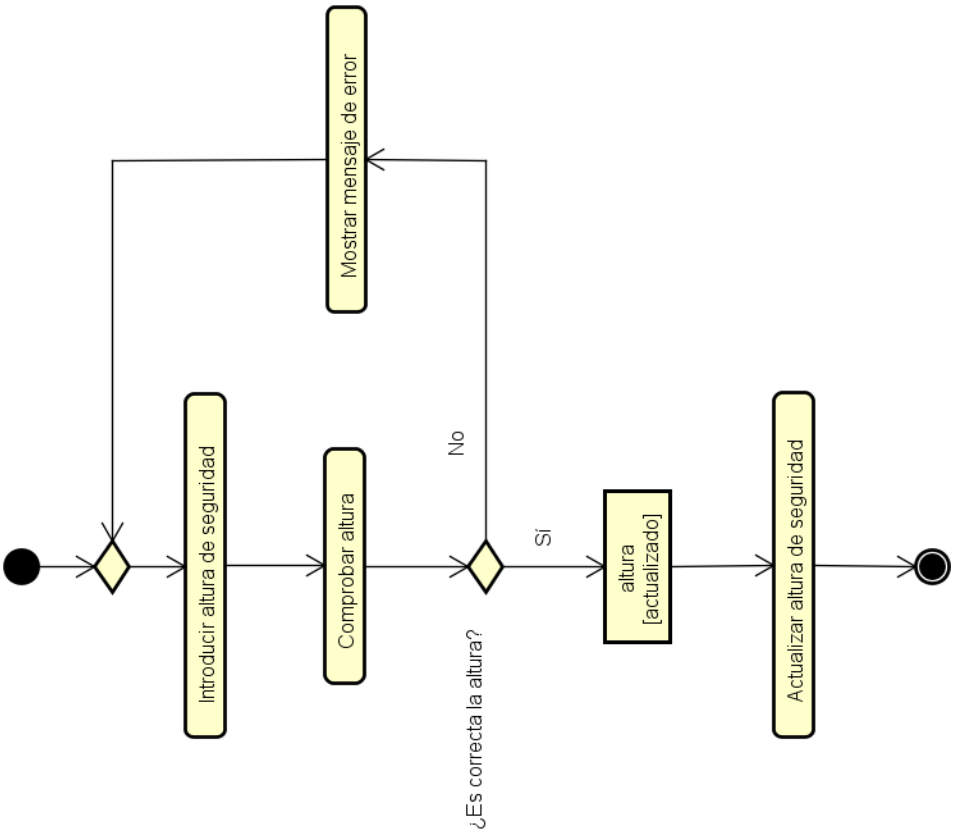

Figura 33. CU14. Introducir altura de seguridad

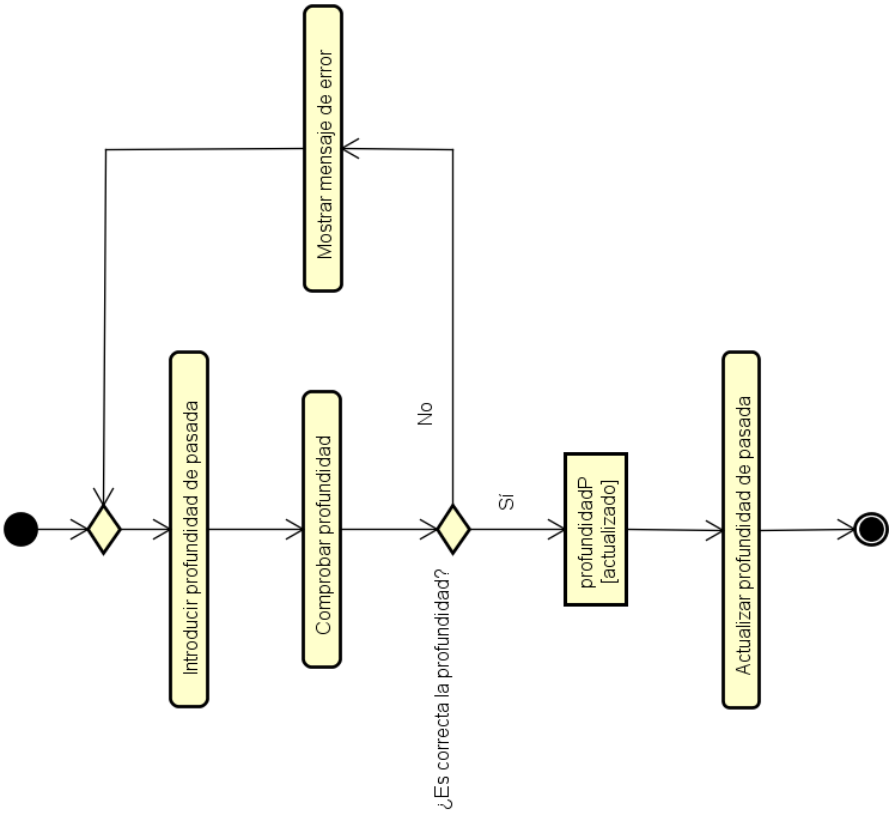

Figura 34. CU15. Introducir profundidad de pasada

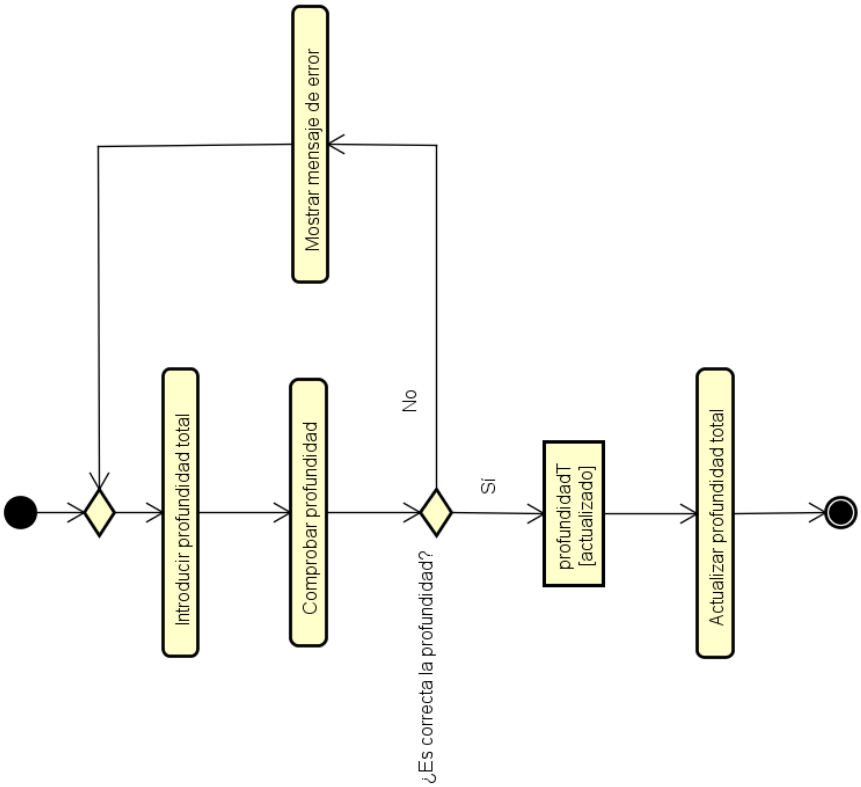

Figura 35. CU16. Introducir profundidad total

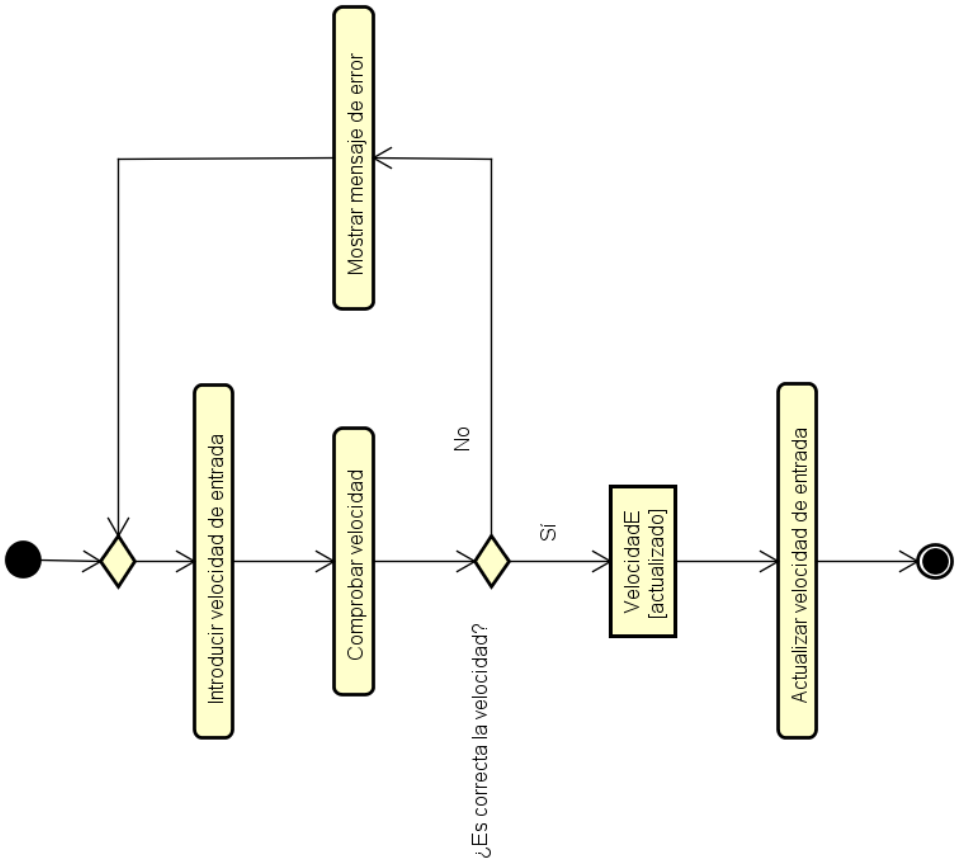

Figura 36. CU17. Introducir velocidad de entrada

4.3 Modelo de dominio

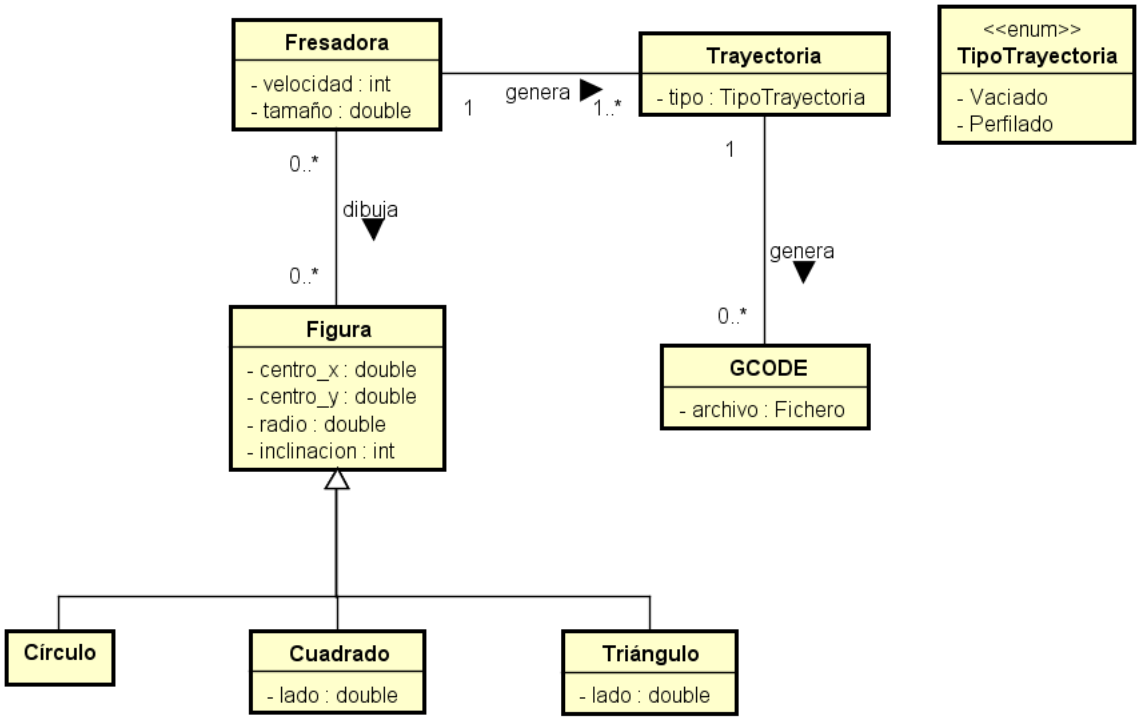

Figura 37. Modelo de dominio

Con los cambios realizados para la segunda iteración el modelo de dominio quedaría de la siguiente manera, resaltando las clases modificadas en blanco y los nuevos atributos en rojo:

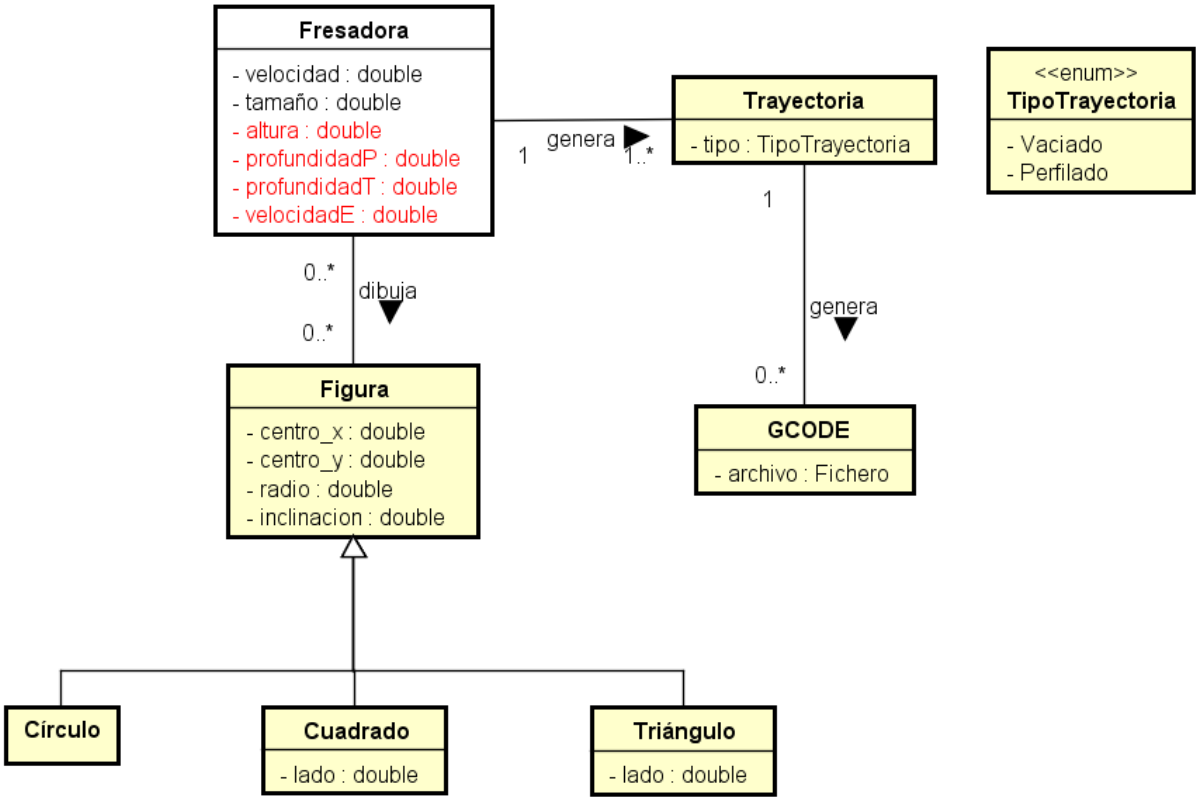

Figura 38. Modelo de dominio. Segunda iteración

Una vez finalizado el análisis de requisitos, como fase intermedia entre el análisis y el diseño, se produce una toma de decisiones sobre algunas tecnologías a utilizar que condicionan el diseño de la aplicación como los frameworks y bibliotecas. Por tanto, el siguiente capítulo será el Capítulo 5 Tecnologías utilizadas y el siguiente el Capítulo 6 Diseño.

Capítulo 5

Tecnologías utilizadas

5.1 Angular

Es un framework de JavaScript, creado y mantenido por Google, pero de código abierto, utilizado para realizar SPA (Single Page Application, aplicaciones web de una sola página). El objetivo de usar SPA es crear páginas reactivas que no recargan el navegador, por lo que se consiguen aplicaciones muy dinámicas y asíncronas, recayendo la carga computacional en el navegador en el que se está ejecutando.

Es de los frameworks más utilizados para realizar páginas web acorde a la encuesta que ha realizado Stack Overflow [18]. Esto tiene una enorme ventaja, tiene una gran base de usuarios que dan vida a una comunidad que comenta y resuelve las dudas generadas sobre el framework. Además de la estabilidad que da que un gigante como Google esté detrás suyo. [1]

5.2 Angular cli

Angular CLI es una herramienta utilizada para inicializar aplicaciones que utilicen el framework Angular, desarrollar componentes y para tareas de mantenimiento asociadas a ello. [2]

5.3 npm

Npm es un gestor de paquetes que permite a los desarrolladores de JavaScript compartir y reutilizar código. Gracias a este gestor podemos conseguir de forma sencilla y actualizada las dependencias que va a tener nuestra aplicación. este gestor de paquetes resulta esencial, ya que de él dependen Angular, Angular CLI y Node.js. [16]

5.4 Node js

Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables. Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor.

Es necesario para ejecutar JavaScript en el sistema operativo de nuestro ordenador y para instalar herramientas de JavaScript como Angular. [13]

5.5 three.js

Es una biblioteca liviana de JavaScript desarrollada para crear y mostrar gráficos animados por ordenador en 3D en un navegador web y puede ser utilizada en conjunción con el elemento canvas de HTML5, SVG o WebGL. Se ha vuelto bastante popular por lo simple que es utilizarla para crear gráficos 3D, en contraposición con lo laborioso que resutaba antes de la irrupción de estas librerías. [17]

5.6 Visual Studio Code

Visual Studio Code es un editor de código ligero pero potente desarrollado por Microsoft. Es una de las mejores opciones para desarrollar en Angular, puesto que viene con Typescript ya integrado, que es el lenguaje en el que se desarrolla la lógica en Angular, porque es un lenguaje también desarrollado por Microsoft. Es también bastante popular como herramienta de desarrollo web. [20]

5.7 Bootstrap

Es un conjunto de herramientas de código abierto para diseños de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript. Es únicamente para desarrollar el front-end, proporcionando estilo a las páginas web.

La versión 4, que es la que se ha utilizado, además tiene funcionalidad específica para Angular. [4]

5.8 Git

Git es un software de control de versiones diseñado por el creador de Linux, Linus Torvalds, usado en el proyecto tanto como control de versiones de la aplicación como medida de seguridad almacenando las versiones en el GitLab de la escuela (gitlab.inf.uva.es). [10]

5.9 Now

Now es un servicio de Zeit (empresa dedicada al cloud computing) que sirve para desplegar aplicaciones en un servidor real, para poder realizar pruebas, por ejemplo. Permite tener 3 aplicaciones subidas gratuitamente. Se ha usado para realizar las pruebas con usuarios para tener la aplicación en un servidor real. [21]

Capítulo 6

Diseño

6.1 Arquitectura del sistema

6.1.1 Framework. Angular

6.1.1.1 Introducción

Existen una gran cantidad de frameworks para desarrollar aplicaciones web. En el fondo, con cualquiera de ellos se puede hacer cualquier aplicación, pero cada uno de ellos está pensado para usarse en diferentes situaciones, dependiendo de las necesidades del proyecto. Por ello he dedicado un tiempo a investigar los diferentes frameworks que existen y las características que tienen para encontrar el que más se adecúe al proyecto que voy a desarrollar.

Después de haber estado estudiando los principales frameworks para JavaScript que existen (Angular, React y Vue principalmente) he decidido usar Angular para desarrollar el proyecto. Las razones que han decantado la balanza por este framework son:

- Es uno de los más populares y usados. Eso significa que hay una gran comunidad que plantea y resuelve dudas en los foros, algo que ayuda mucho cuando surge algún problema. Además, hay muchos tutoriales y documentación hechos por usuarios.

- Es muy completo. Es un framework que abarca todo, backend y frontend. También facilita el desarrollar aplicaciones híbridas para móvil si se quisiera.

- Sintaxis “heredada” de orientación a objetos. Uno de los puntos que destacan cuando comparan los diferentes es que para aprender Angular se lleva ventaja si se conocen Java o C++ puesto que usa una sintaxis parecida.

- Es muy demandada. Es una motivación más personal. Es el framework más requerido por las empresas para sus trabajadores así que aprenderlo para desarrollar este proyecto me beneficia de cara al futuro laboral.

6.1.1.2 Arquitectura interna

Internamente, Angular está pensado para un desarrollo basado en componente, pero ¿qué es un componente? Un componente es un conjunto de código independiente, con una funcionalidad propia que se encarga de realizar un trabajo específico dentro de la aplicación. Por ejemplo, en una red social, un componente podría ser el chat, otro el muro, otro el encargado de las stories, etc. En el fondo una aplicación creada con Angular es simplemente un conjunto de componentes trabajando juntos para que la aplicación funcione correctamente. Esta forma de desarrollo basado en componentes tiene ciertas ventajas:

- **Reutilización de software**. Al ser trozos de código independientes pueden ser reutilizados en diferentes aplicaciones. En el ejemplo antes propuesto, el componente que en una red social implementa el chat puede ser utilizado en cualquier otra aplicación que necesite de un chat, como podría ser un juego, una página de atención al cliente o un aula virtual.
- **Simplificación de las pruebas**. Se pueden probar los componentes por separado, lo cual facilita mucho el testing. Además, una vez desarrollado y probado, el componente se puede reutilizar sin tener que volver a probarlo puesto que ya se hizo previamente.
- **Simplificación del mantenimiento del sistema**. Si la aplicación comenzase a fallar, al estar los componentes débilmente acoplados es más fácil localizar el error, aislarle y solucionarlo.
- **Opacidad**. No es necesario saber cómo funciona un componente, estando bien documentado se puede usar perfectamente sin tener conocimiento alguno de su funcionamiento interno.
- **Ahorro**. Reutilizar un componente ya creado en vez de crearlo de cero ahorra mucho tiempo y esfuerzo, factores clave en el desarrollo de software.

  No todo son ventajas, el desarrollo basado en componentes también tiene algunos inconvenientes:

- **Documentación**. Para que un componente sea fácilmente reutilizable es necesario que esté muy bien documentado, para facilitar la reutilización del componente por terceros.
- **Esfuerzo adicional**. Cuanto menos específico se desarrolle un componente, con la intención de reutilizarlo en la mayoría de las ocasiones posibles, más complicado será y más pruebas habrá que realizar con el objetivo de que sea más fácil su reutilización. Es necesario quedarse en un término medio, capaz de reutilizarse, pero sin intentar abarcar cualquier situación.

A continuación, vamos a ver cómo son los componentes de angular, su estructura, funcionamiento y la relación entre ellos.

El núcleo de un componente es su controlador. Viene siempre precedido por el decorador @Component que indica cómo se comporta el componente. Tiene el siguiente aspecto:

```
@Component({
      selector: 'app-ejemplo';
      template: '<h1>Hola mundo</h1>',
      style: 'h1{color: blue}'
})
```

- Selector: Indica la manera de introducir el componente en la página web. Mediante esa etiqueta Angular interpreta que en su lugar es donde se incrusta el componente seleccionado.
- Template: Es el HTML asociado al componente. No es necesario que esté escrito ahí mismo. Es más, rara vez se usa, en su lugar se usa `templateUrl: './ejemplo.component.html'` al que se le puede enlazar el archivo HTML cuando es más extenso que unas pocas líneas.
- Style: Donde se define el estilo del HTML. Al igual que con el HTML se suele enlazar en archivos externos para mejorar la legibilidad del código. En este caso sería con `styleUrls: ['./ejemplo.component.css']`, pudiéndose enlazar varias páginas de estilo.

Se puede ver la representación de un componente y cómo sería la integración con otro componente, que sería su hijo, a continuación:

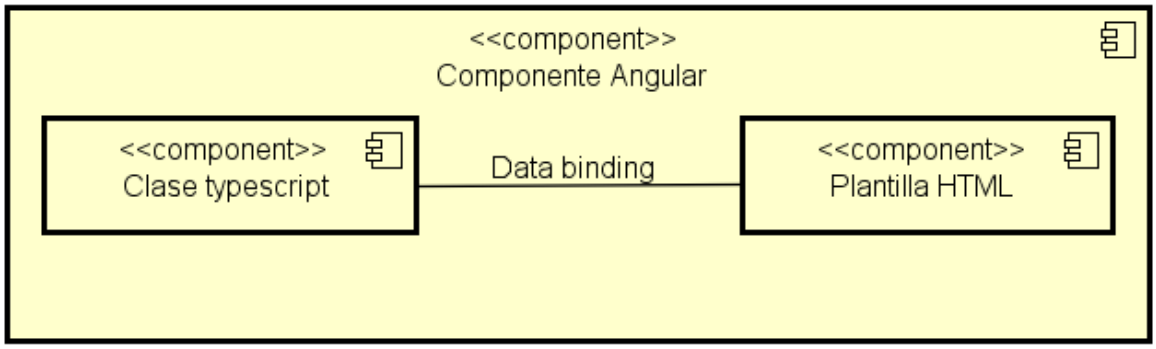

Figura 39. Componente Angular

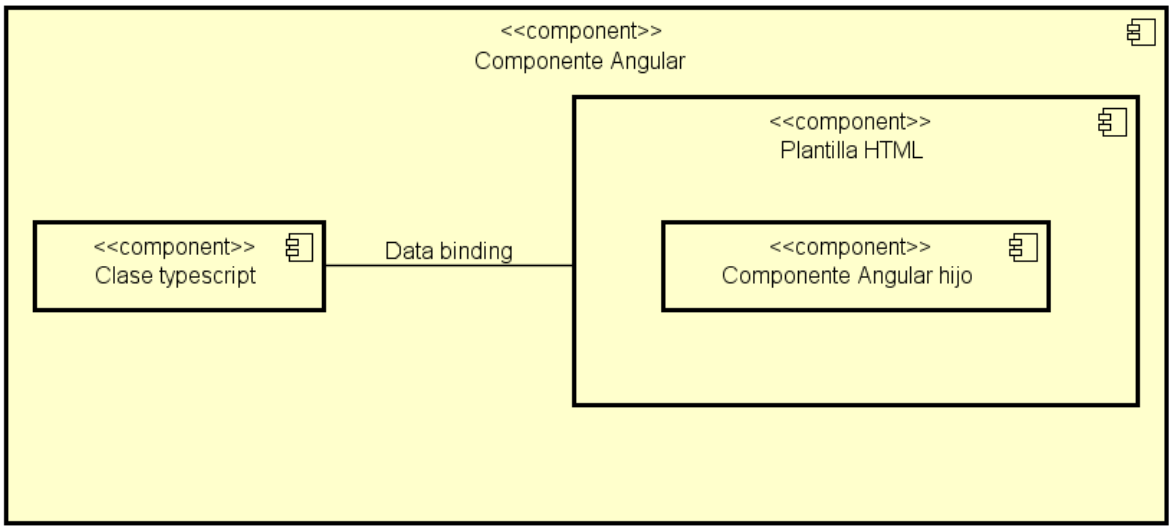

Figura 40. Componente Angular con componente hijo

En Angular también existen los Servicios. Sirven para la carga de datos y compartir datos entre componentes. Su representación es la siguiente:

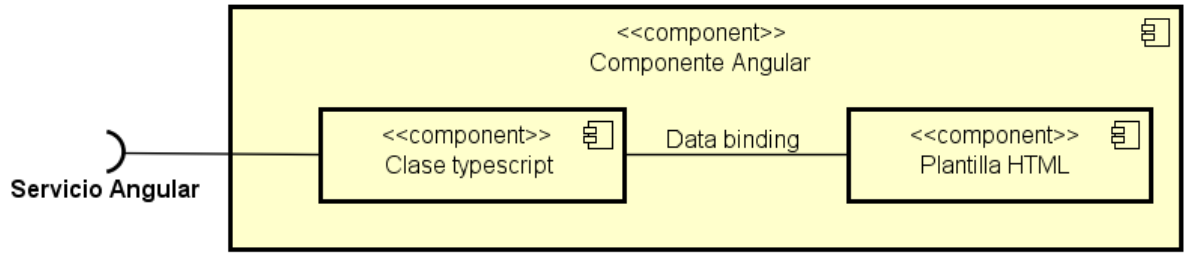

Figura 41. Componente Angular con servicio

6.1.2 Librería. three js

6.1.2.1 Introducción

En los requisitos del proyecto se especifica que en gran parte de la aplicación se necesitan gráficos, tanto en el lienzo de trabajo como en la representación 3D del resultado del cálculo de la trayectoria de fresado (parte que no se desarrollará en este TFG pero que aun así debe ser tenida en cuenta para próximas ampliaciones de la aplicación). Es por ello por lo que se necesita alguna librería específica para crear y mostrar gráficos para facilitar el desarrollo. Por tanto, con el consejo de las tutoras, decidimos usar three js.

6.1.2.2 Arquitectura interna

Para que Three js funcione únicamente se necesitan tres clases: Renderer, Scene y Camera.

- Renderer: Es el encargado de renderizar lo que hemos creado para poder verlo en el navegador. Esto sólo lo hace una vez, si queremos que actualice los cambios que se realicen, hay que llamarlo a renderizar periódicamente.

- Scene: Es la escena donde se van añadiendo todas las formas y figuras, una especie de contenedor. Luego es lo que renderiza el Renderer.
- Camera: Es un tipo especial de objeto que hay en la escena. Simularía el funcionamiento de una cámara, pudiéndola mover por la escena, dirigir dónde apunta, etc. Hay varios tipos de cámara, como PerspectiveCamera, utilizada para entornos 3D. O ArrayCamera, usada para renderizar escenas para realidad virtual. En el caso que nos ocupa he usado OrtographicCamera ya que al tratarse de un entorno 2D era la que mejor se ajustaba a nuestras necesidades.

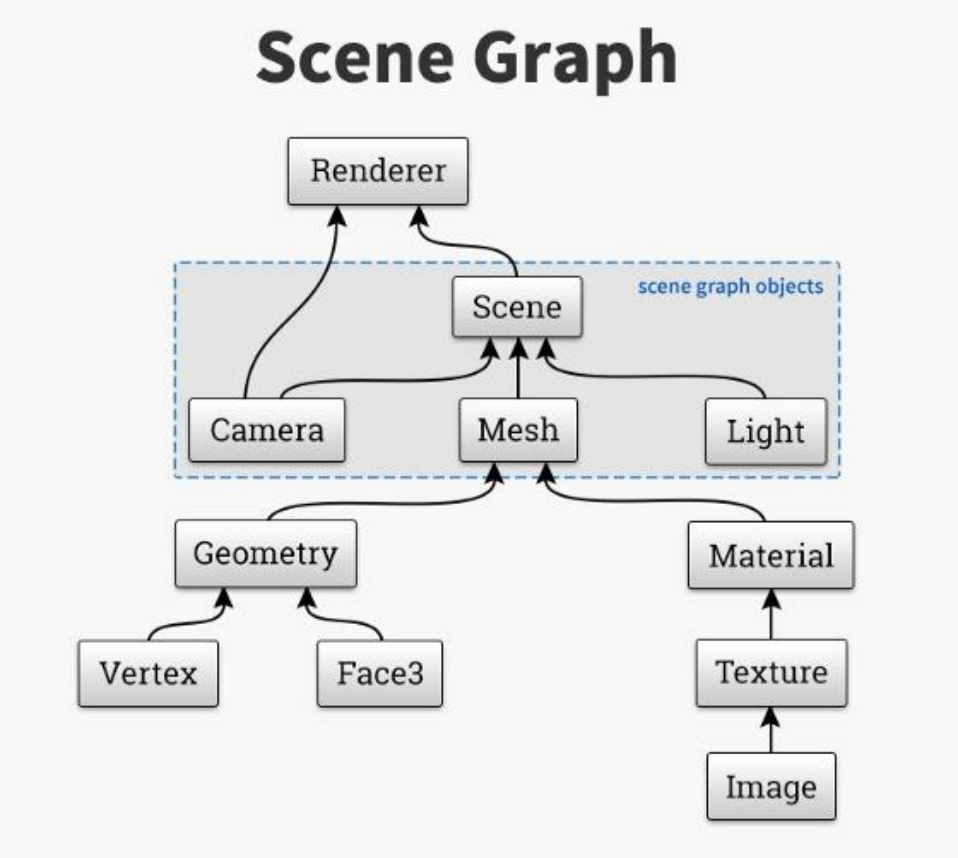

Figura 42. Arquitectura three.js [11]

Luego, como podemos ver en la figura, se añaden diferentes Mesh (que serían los objetos creados) a la escena para hacer las creaciones que deseemos. Cada Mesh tiene dos partes, Geometry y Material. Geometry es el encargado de la forma, creándolo con vectores de 3 dimensiones para crear los vértices y triángulos, uniendo 3 de esos vértices para formar las caras de los objetos. Material es el encargado de la apariencia del Mesh. Al material se le pueden añadir texturas e imágenes para crear una apariencia a voluntad.

Aparte de lo explicado, hay otros objetos que pueden añadirse a la escena. Quizás el más importante sería Light, el cual provee de luz a nuestra escena, creando luces y sombras.

6.1.3 Aplicación.

A continuación, se explica el patrón arquitectónico usado y se muestra la arquitectura interna de la aplicación, es decir, los componentes y servicios creados y su relación entre ellos, después, las modificaciones y/o adiciones realizadas en la segunda iteración.

6.1.3.1 MVC

Debido al uso de Angular, el controlador viene por defecto separado de la vista, por lo que este framework facilita mucho el implementar el patrón arquitectónico Modelo-Vista-Controlador. En el caso del modelo, es una buena práctica separar las acciones que tienen que ver con consulta de datos en los Servicios. Como en este proyecto no hay persistencia de datos, se ha decidido no separar el modelo del controlador, por lo que no sería una aplicación estricta del patrón MVC

En cuanto al reparto de carga entre el cliente y el servidor, al tratarse de JavaScript, nos encontramos con un cliente pesado sobre el que cae el grueso de la carga de cómputo. Como no hay base de datos, en este caso toda la acción se realiza en el navegador del cliente.

6.1.3.2 1ª Iteración

- App: Es el contenedor de toda la aplicación. Apenas tiene funcionalidad, solamente las pestañas que indican en qué parte de la aplicación nos encontramos.

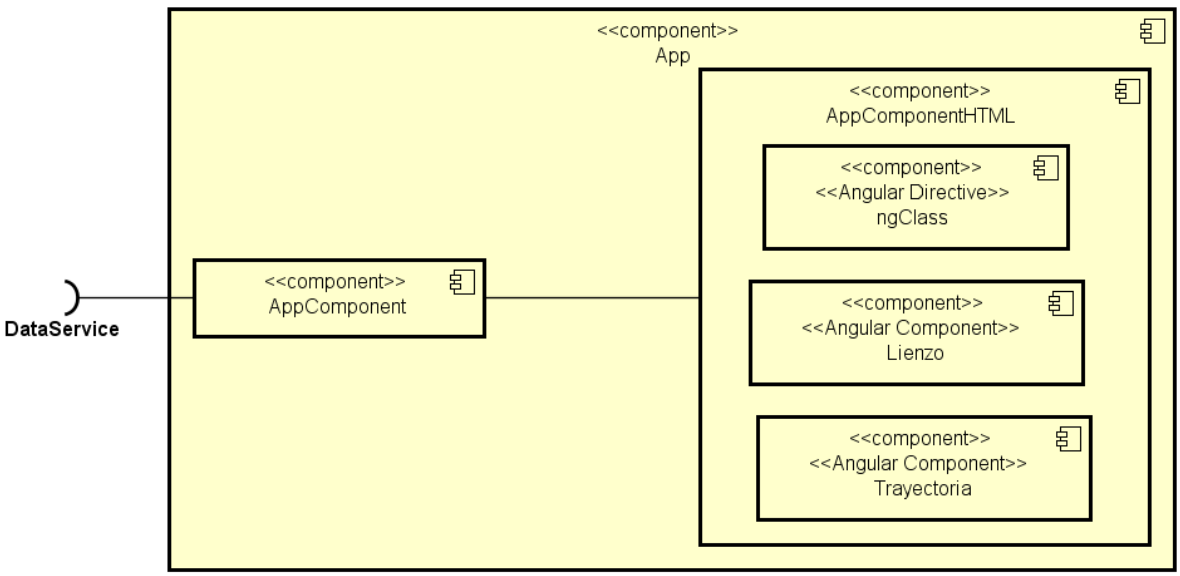

Figura 43. Componente App

- Lienzo: Uno de los dos componentes principales de la aplicación. Es el encargado del dibujo y sus transformaciones, así como de la recopilación de las opciones de fresado.

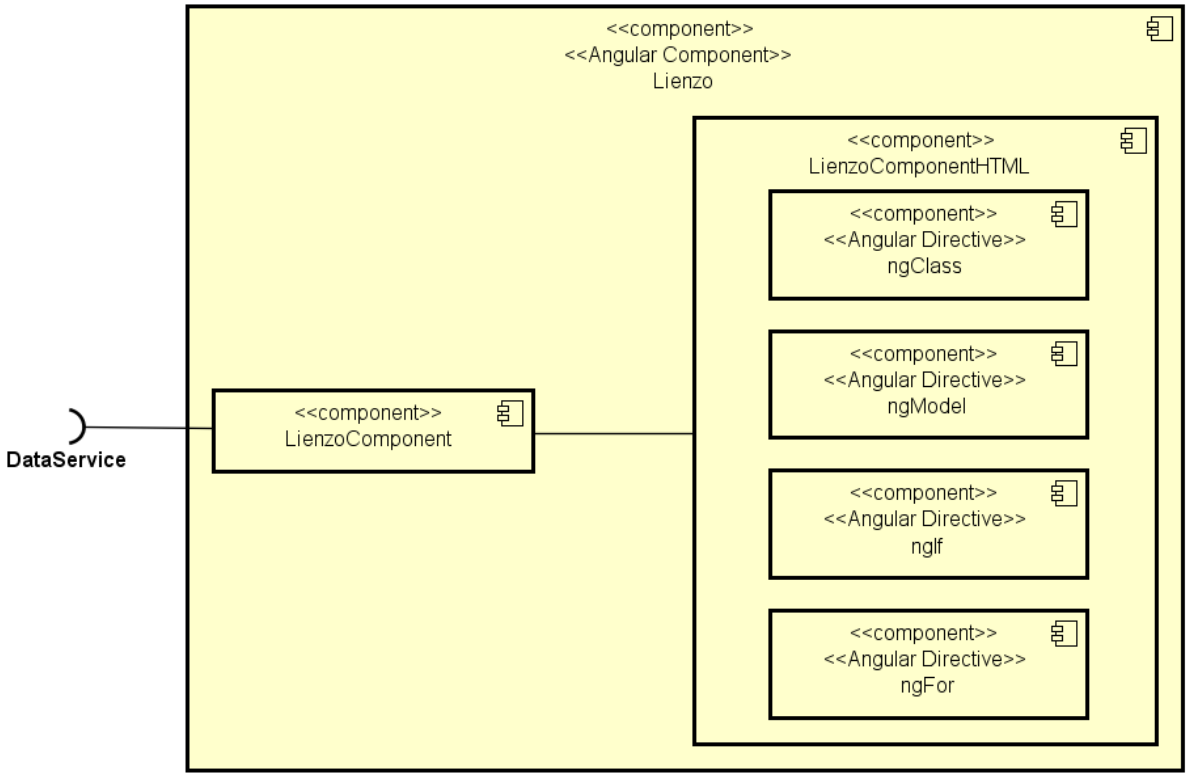

Figura 44. Componente Lienzo

- Trayectoria: El otro componente principal. Es el encargado de calcular las trayectorias con los datos obtenidos del componente Lienzo y de representar las mismas.

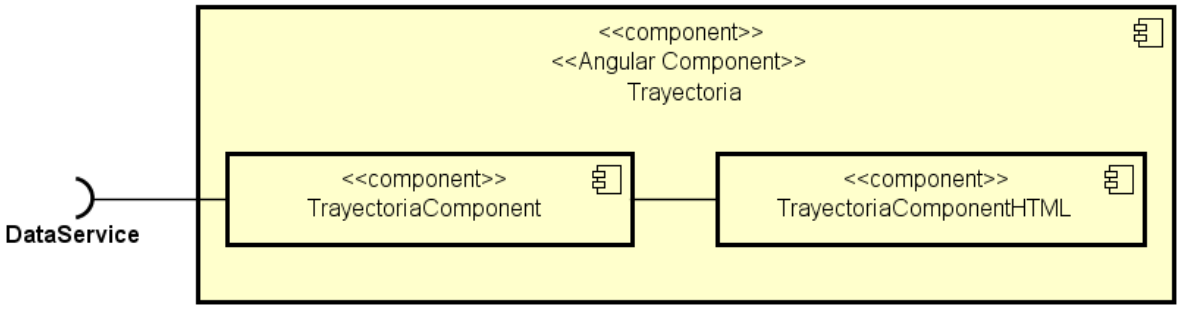

Figura 45. Componente Trayectoria

- DataService: Es el servicio creado para compartir datos entre componentes.

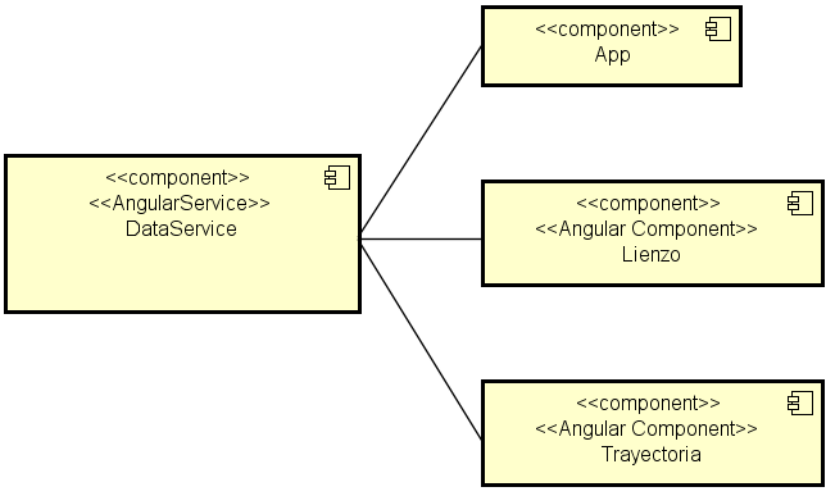

Figura 46. Servicio Data

Una vez vistos los componentes y servicios creados, ahora vamos a ver la estructura interna de las clases que los forman, sus atributos y funciones.

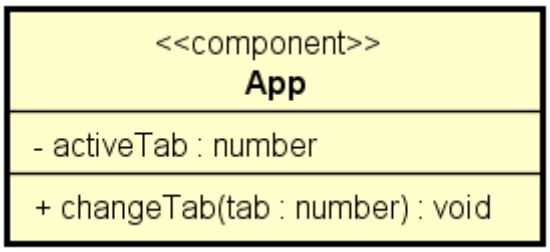

Figura 47. Clase App

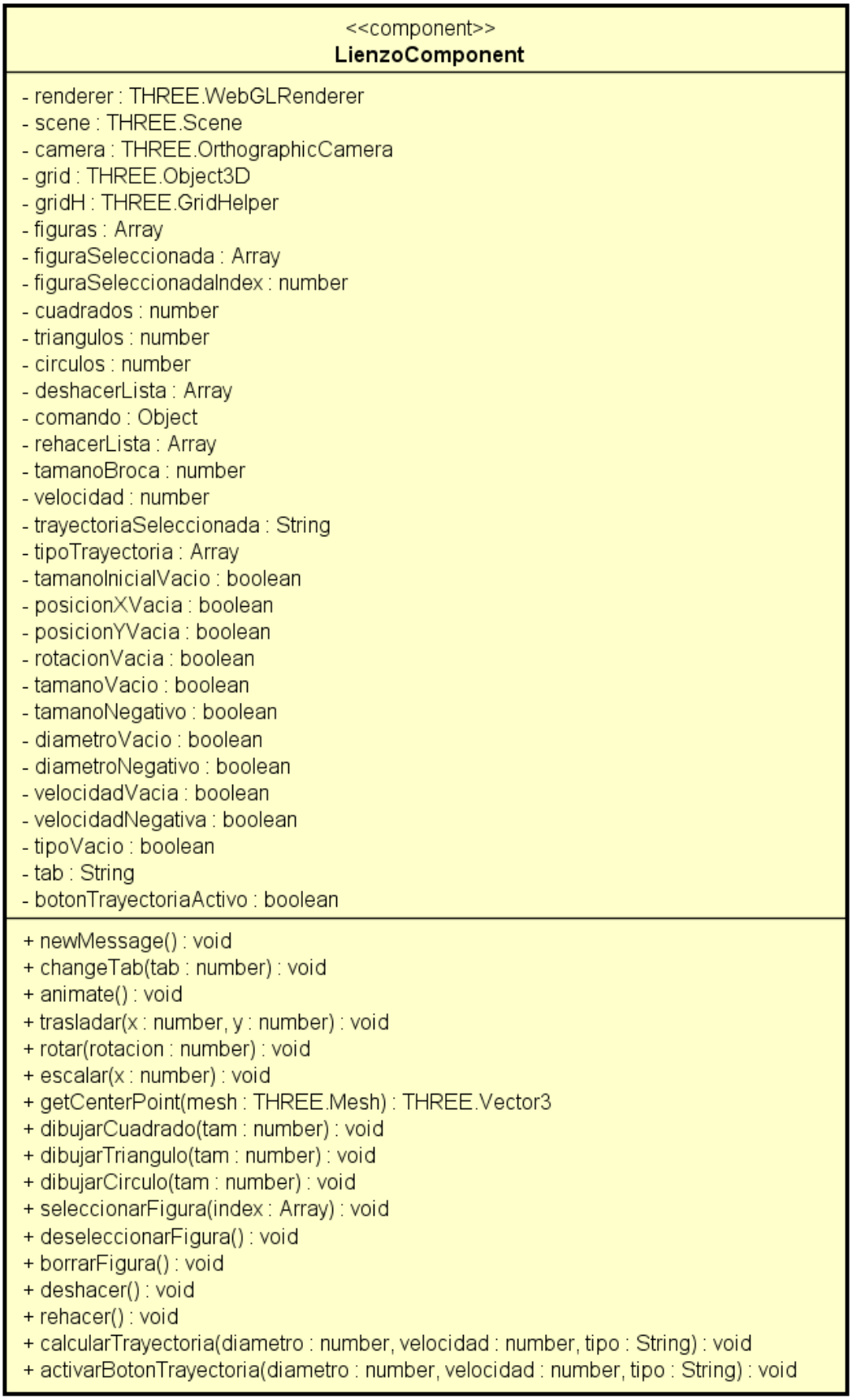

Figura 48. Clase LienzoComponent

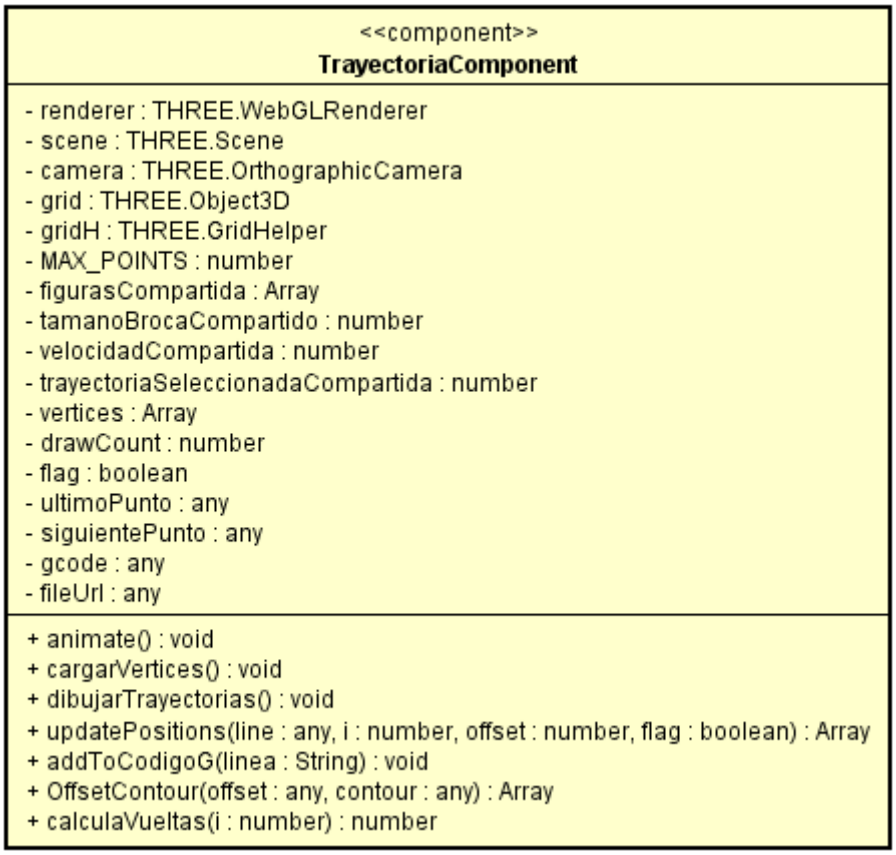

Figura 49. Clase TrayectoriaComponent

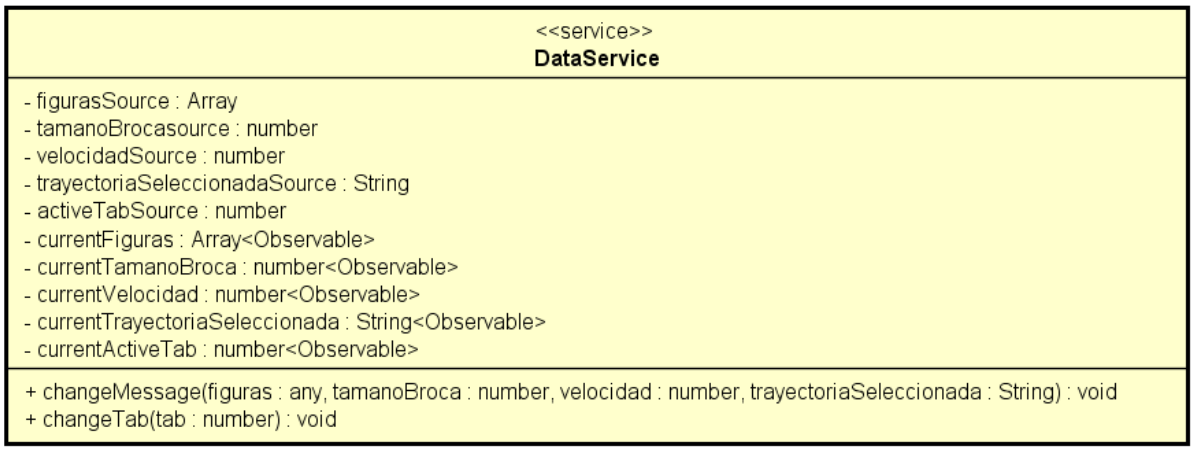

Figura 50. Clase DataService

6.1.3.3 2ª Iteración

En la segunda iteración no se ha creado ningún componente ni servicio, por lo que únicamente se detallará las clases de los componentes modificados, que en este caso son LienzoComponent, TrayectoriaComponent y DataService. Se han resaltado los cambios y adiciones en rojo.

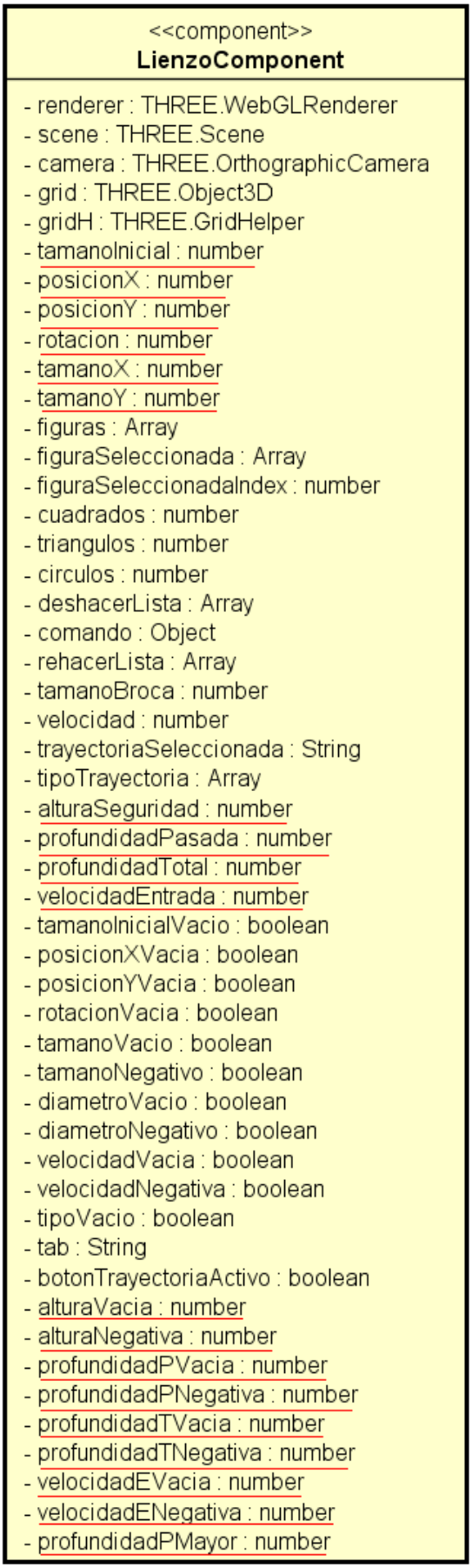

Figura 51. Atributos de la clase LienzoComponent. Segunda iteración

Figura 52. Operaciones de la clase LienzoComponent. Segunda iteración

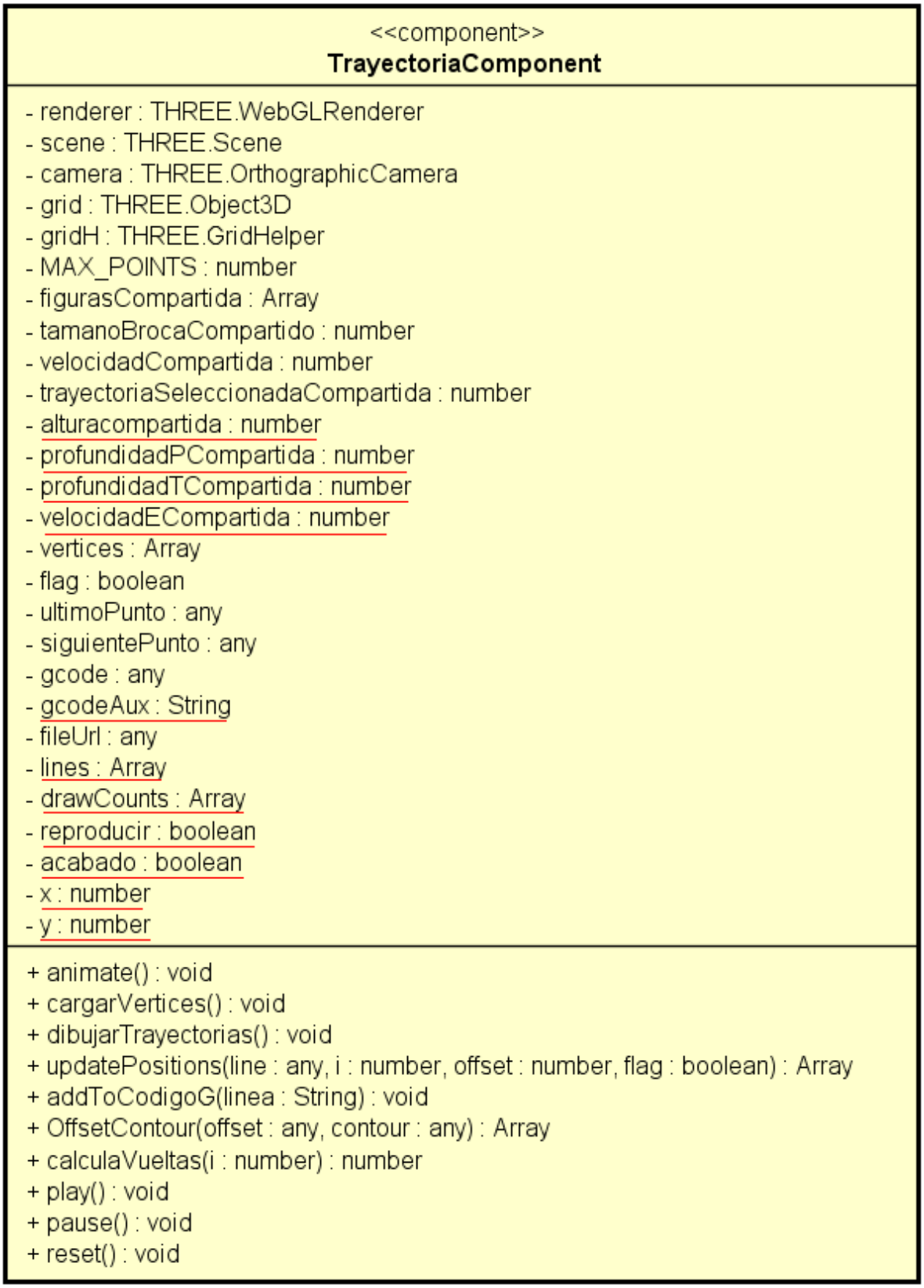

Figura 53. Clase TrayectoriaComponent. Segunda iteración

Figura 54. Clase DataService. Segunda iteración

Capítulo 7

Implementación y pruebas

7.1 Implementación

La implementación de la aplicación web ha sido realizada en dos iteraciones. A continuación, se desgrana un poco el orden en el que se han ido implementando cada una de las funcionalidades.

7.1.1 1ª Iteración

- Implementación de las transformaciones a las figuras (traslación, rotación y escalado).
- Implementación de la creación de los diferentes tipos de figura (cuadrado, triángulo equilátero y círculo).
- Implementación de la lista de figuras.
- Implementación del deshacer/rehacer.
- Implementación del servicio.
- Implementación de la generación de las trayectorias.
- Implementación de la generación de Gcodes.

Como se puede ver, aunque la funcionalidad de deshacer/rehacer está en los requisitos de la segunda iteración, se realizó en la primera. Esto es porque se decidió en un estado muy temprano de la implementación y tomé la decisión de implementarlo en ese momento, puesto que requería menos esfuerzo realizarlo en ese punto que una vez acabada la aplicación, teniendo que remodelarla para añadir la funcionalidad.

7.1.2 2ª Iteración

- Implementación de las nuevas opciones de fresado (altura de seguridad, profundidad de pasada, profundidad total y velocidad de entrada.
- Implementación de la animación de trayectorias.
- Implementación de los controles de la animación.

7.1.3 Patrones

7.1.3.1 Comando

Para implementar las funciones de deshacer y rehacer en el lienzo de dibujo se ha implementado el patrón de diseño comando. Este comando básicamente lo que dice es que

hay que encapsular las acciones u operaciones en objetos y así facilitar la parametrización de los métodos además de opacarlos. De esta manera se pueden almacenar las operaciones, pudiendo deshacerlas y rehacerlas o restaurar el estado de la aplicación a un momento dado.

En el caso concreto de la aplicación, se ha creado un objeto en el que se encapsularan los comandos. Este se llama `comando`, este almacena la operación que es (creación, desplazamiento, rotación, escalado o borrado) y un Array con los atributos necesarios para realizarla, como son en todos los casos la figura sobre la que se aplica y dependiendo de la operación, otros atributos como las coordenadas de la figura en el desplazamiento o los grados de rotación en la rotación.

Estos comandos se almacenan en un Array para poder deshacerlos y de hacerlo, pasan a otro Array para poder rehacer el cambio. Si luego de deshacer cambios se realiza alguno nuevo, el Array de rehacer se vacía entendiendo que se ha deshecho hasta donde se quería y se desea trabajar a partir de ahí, como suele ser implementado en el resto de las aplicaciones.

7.1.3.2 Singleton

Este patrón de diseño no se usa explícitamente en la aplicación, pero sí lo utiliza Angular internamente, por ello considero relevante el exponerlo. Este patrón consiste en restringir la creación de un tipo de objetos. Crea una única instancia de cierto objeto y, cuando se desea utilizar en otra parte, si ya está creado, en vez de duplicarlo se ofrece la instancia creada previamente para su uso y así tener una única instancia global en todo momento de esa clase.

Una de las razones para usar este patrón es cuando hay cierto tipo de datos que debe estar disponible para todos los demás objetos de la aplicación. Esa es la función principal de los Servicios en Angular, por lo que internamente el framework aplica el patrón singleton sobre este tipo de objetos mediante la inyección de dependencias para proporcionar siempre la misma instancia de la clase.

7.1.4 Algoritmos

En esta sección se explica cómo se han desarrollado los principales algoritmos de la aplicación. En este caso, aquellos para calcular las trayectorias para el fresado:

7.1.4.1 Perfilado

Para conseguir que la herramienta perfile las figuras por el borde exterior es necesario que ésta pase alejada de la figura (como se puede ver en la Figura 55. Explicación del offset) para que su borde pase justo por la silueta, ya que si el centro de la herramienta pasara justo por el borde se “comería” parte de la figura porque tiene un diámetro que se introduciría en la figura que se quiere perfilar.

Para conseguir perfilar justo por su borde es necesario calcular a qué distancia debe pasar la herramienta por el exterior de la pieza, esa distancia (a partir de ahora llamaba offset)

en el caso de perfilado, es la mitad del diámetro de la herramienta. Así el borde de ésta pasará justo por el borde de la figura, sin adentrarse en ningún momento en su interior.

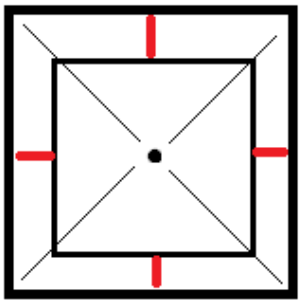

Figura 55. Explicación del offset [19]

El algoritmo se podría describir de la siguiente manera:

- Se obtienen los vértices de la figura
- Se les aplica el offset correspondiente
- Se añade el primer punto al final para cerrar la figura
- Se guarda el resultado
- Fin

De esta manera se conseguirían los puntos para una trayectoria de perfilado.

7.1.4.2 Vaciado

Este algoritmo es bastante más complicado que el de perfilado. El objetivo es calcular una trayectoria que vacíe el interior de la figura, dado un diámetro de herramienta. Para ello es necesario calcular el número de veces que hay que acercarse al centro mientras se incrementa el offset. Una vez calculadas, se va calculando el offset de cada pasada. El offset de la primera pasada es, igual que en el perfilado, la mitad del diámetro de la herramienta, solo que en este caso con valor negativo para ser hacia dentro. Esta primera pasada deja las esquinas interiores de la figura sin vaciar, pero debido al tamaño de la herramienta es imposible hacer más sin salirse de la figura. En las demás pasadas el offset no es el mismo que en la primera, ya que dejarían sin vaciar las esquinas como la primera, pero en este caso sí que se puede solucionar. Para solucionarlo, se calcula el offset necesario para que el borde de la herramienta empiece donde acababa el borde de la figura en la pasada anterior, para así pasar a partir del último sitio al que la última pasada pudo llegar. Esta medida es un poco ineficaz ya que, fuera de las esquinas, hay una parte de la herramienta que pasa por una zona en la que ya ha trabajado en la anterior pasada. A pesar de ello se ha realizado de esta manera porque es la manera convencional de realizar trabajos de vaciado. Esto se realiza hasta completar las pasadas que se habían calculado, que era cuando al aplicar el offset, se pasaba del centro. A pesar de que no había espacio para otra pasada, algunas veces el centro de la figura quedaba sin vaciar porque el borde de la herramienta no llegaba a pasar por él (Figura 56. Vaciado que no llegaba al centro**¡Error! No se encuentra el origen de la referencia.**). Para solucionarlo, si se detecta esa situación, se añade un último punto en la trayectoria justo en el centro de la figura y así asegurarnos de que le vacía.

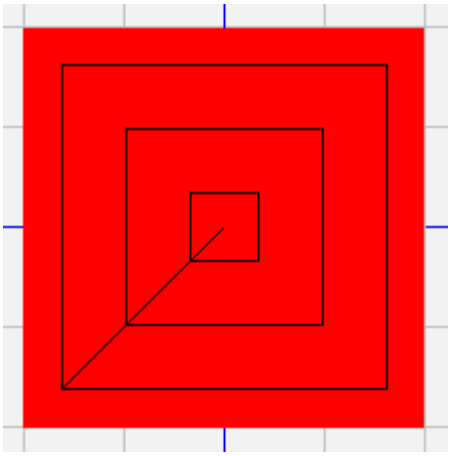

Figura 56. Vaciado que no llegaba al centro

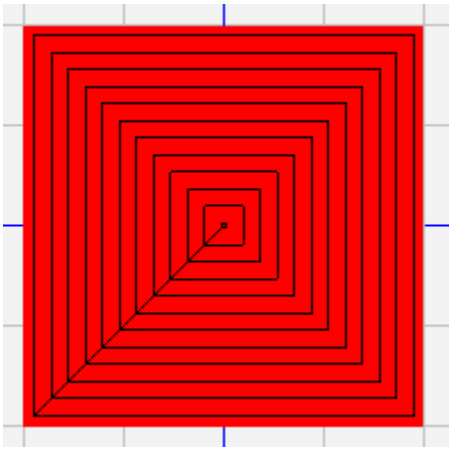

Figura 57. Vaciado normal

El algoritmo se podría describir de la siguiente manera:

- Se obtienen los vértices de la figura
- Se calculan las vueltas que se van a necesitar para llegar al centro
- Se les aplica el offset correspondiente tantas veces como vueltas se han calculado necesarias
- Cada vez que se aplica un offset se añade el primer punto obtenido al final para así cerrar la figura
- Se van almacenando todos los puntos obtenidos
- Si la última pasada no llega al centro, se añade un último punto a los obtenidos previamente
- Fin

De esta manera se conseguirían los puntos para una trayectoria de perfilado.

7.2 Despliegue

La aplicación se ha desplegado sobre un servidor de pruebas proporcionado por Now. Se ha decidido esto en vez de en el servidor definitivo puesto que al tratarse de una aplicación

para una empresa acabará desplegado en su servidor. Por lo que para realizar las pruebas y exponer ante el tribunal, se usará este servidor provisiona por comodidad. Una vez que se acabe el TFG, la aplicación ya se desplegará en el servidor correspondiente.

El servidor en el que se encuentra desplegada es el siguiente:

https://nomadwebdraw.rikrdo95.now.sh/

7.3 Pruebas

Para el testeo de la aplicación se han realizado tres tipos de pruebas: unitarias, de interacción y de usabilidad. Estas últimas sólo en la segunda iteración, puesto que requería de la versión final de la aplicación y que estuviera ya desplegada.

7.3.1 Pruebas unitarias

Las pruebas unitarias son aquellas pruebas que tienen como objetivo comprobar el correcto funcionamiento de una unidad de código, en nuestro caso, que las principales funciones de la aplicación dan como resultado aquello que se espera de ellas y, en caso, contrario, corregirlas para que tengan un correcto funcionamiento. Pueden ser tanto automatizadas como manuales. En este caso he elegido hacerlas manualmente.

7.3.1.1 1ª Iteración

Las funciones por probar son las siguientes, junto a las especificaciones de su prueba:

```
/**
* Función para trasladar una figura previamente seleccionada.
* Guarda la posición previa al cambio para poder deshacerlo.
*
* @param x coordenada X de la traslación
* @param y coordenada Y de la traslación
*/
trasladar(x: number, y: number)
```

Especificación del caso de prueba	Teniendo un cuadrado de tamaño 200 en el origen de coordenadas previamente dibujado y seleccionado, se llamará a la función con los parámetros (100, 100)
Resultado esperado	El centro de la figura se moverá del origen de coordenadas a las coordenadas (100, 100)
Resultado obtenido	El centro de la figura se ha movido del origen de coordenadas a las coordenadas (100, 100)

Tabla 52. Prueba unitaria de trasladar

```
  /**
  * Función para dibujar un cuadrado.
  * Guarda la acción para poder deshacerla.
  *
  * @param tam lado del cuadrado
  */
 dibujarCuadrado(tam: number)
```

Especificación del caso de prueba	Se llamará a la función con un parámetro que será 100
Resultado esperado	Se creará una figura que es un cuadrado de lado 100 y se habrá añadido a la lista de figuras
Resultado obtenido	Se creará una figura que es un cuadrado de lado 100 y se habrá añadido a la lista de figuras

Tabla 53. Prueba unitaria de dibujarCuadrado

```
  /**
  * Función que selecciona una figura para que resalte cuál es la que se
está modificando.
  * La cambia el color de rojo a azul y la pone un poco por delante de
las demás.
  *
  * @param index índice de la figura a seleccionar
  */
 seleccionarFigura(index: [String, Object])
```

Especificación del caso de prueba	Habiendo dibujado un cuadrado y un triángulo de tamaño 200 previamente, se llamará a la función con un parámetro que será la figura que representa al cuadrado, para así seleccionarle
Resultado esperado	El cuadrado estará seleccionado
Resultado obtenido	El cuadrado está seleccionado

Tabla 54. Prueba unitaria de seleccionarFigura

```
  /**
  * Función que pasa el dibujo al componente trayectoria para que la
calcule y muestre.
  * También valida los campos de opciones de fresado.
  *
```

```
  * @param diametro diámetro de la herramienta
  * @param velocidad velocidad del fresado
  * @param tipo tipo de fresado
  */
 calcularTrayectoria(diametro: number, velocidad: number, tipo: String)
```

Especificación del caso de prueba	Habiendo dibujado un cuadrado y un triángulo de tamaño 200 previamente, se llamará a la función con el diámetro introducido, la velocidad introducida y el tipo de fresado seleccionado
Resultado esperado	Enviará al componente Trayectoria los parámetros introducidos y la lista de figuras dibujadas. Después cambiará de pestaña
Resultado obtenido	Ha enviado al componente Trayectoria los parámetros introducidos y la lista de figuras dibujadas. Después cambió de pestaña, pero no indicaba correctamente en la que se encontraba.

Tabla 55. Prueba unitaria de calcularTrayectoria

```
  /**
  * Función encargada de ir representando las trayectorias de cada una
de las figuras.
  *
  */
 dibujarTrayectorias()
```

Especificación del caso de prueba	Teniendo las figuras dibujadas, el tamaño de herramienta, la velocidad y al tipo de fresado previamente, se llamará a la función
Resultado esperado	Obtendrá los puntos por los que tendrá que pasar la trayectoria
Resultado obtenido	No obtiene todos los puntos por un límite en la capacidad.

Tabla 56. Prueba unitaria de dibujarTrayectorias

```
 /**
  * Función que calcula los puntos de la trayectoria seleccionada para
una figura dada.
  *
  * @param line línea que representará la trayectoria
```

```
  * @param i figura sobre la que se hacen los cálculos
  * @param offset radio de la herramienta
  * @param flag bandera que indica si se calcula una figura (true) o la línea de transición (false)
  * @returns última trayectoria en el caso del vaciado y la única en el caso de perfilado
  */
 updatePositions(line: any, i: number, offset: number, flag: boolean)
```

Especificación del caso de prueba	Teniendo las figuras dibujadas, el tamaño de herramienta, la velocidad y al tipo de fresado previamente, se llamará a la función con el objeto en el cual se guardarán los puntos, el índice de la figura que se calcula, el radio de la herramienta y una bandera que indica que se están calculando los puntos de una figura
Resultado esperado	Obtendrá los puntos por los que tendrá que pasar la trayectoria para hacer el fresado de la figura
Resultado obtenido	No obtiene correctamente los puntos porque no calcula bien el ángulo entre vértices

Tabla 57. Prueba unitaria de updatePositions

```
 /**
  * Función que calcula los vértices de una figura aportando una compensación (offset)
  *
  * @param offset distancia a compensar
  * @param contour vértices originales
  * @returns los puntos de la figura aplicado el offset
  */
 OffsetContour(offset: any, contour: any)
```

Especificación del caso de prueba	Se llama a la función pasándola un offset de 50 y los puntos de un cuadrado
Resultado esperado	Devolverá los puntos del cuadrado aplicando el offset
Resultado obtenido	Ha devuelto los puntos del cuadrado aplicando el offset

Tabla 58. Prueba unitaria de OffsetContour

```
  /**
  * Método que calcula las vueltas necesarias para llegar al centro de
la figura para determinado tamaño de herramienta.
  *
  * @param i número de figura
  * @returns vueltas para llegar al centro
  */
 calculaVueltas(i: number)
```

Especificación del caso de prueba	Habiendo unas figuras dibujadas y un tamaño de herramienta, se llama a la función indicando el índice 0 para que calcule las vueltas para llegar al centro de la primera figura
Resultado esperado	Calculará y devolverá las vueltas necesarias para llegar al centro con una determinada herramienta
Resultado obtenido	Ha calculado mal las vuelas porque la figura no estaba en el origen y había una operación aritmética mal escrita

Tabla 59. Prueba unitaria de calculaVueltas

Después de realizar las pruebas han ayudado a encontrar algunos fallos:

1. La función dibujarTrayectorias fallaba cuando la trayectoria de una figura tenía demasiados puntos debido a un límite en la capacidad.
2. La función calcularTrayectoria cambiaba de pestaña, pero no indicaba correctamente en la que se encontraba.
3. La función updatePositions calculaba mal la trayectoria del vaciado porque en vez de coger el primer y último vértice para calcular el ángulo, cogía el primero dos veces.
4. La función calculaVueltas no devolvía el dato esperado cuando la figura no estaba en el origen porque realizaba incorrectamente una operación aritmética.

Una vez corregidos los fallos, se tiene la certeza de que las funciones probadas unitariamente tienen un correcto funcionamiento.

7.3.1.2 2ª Iteración

En la segunda iteración no se van a probar las funciones ya probadas en la primera iteración. Se probarán las funciones nuevas y aquellas que estuvieran en la primera iteración, pero hayan sufrido cambios. Esas son:

```
  /**
   * Función que pasa el dibujo al componente trayectoria para que la
calcule y muestre.
   * También valida los campos de opciones de fresado.
   *
   * @param diametro diámetro de la herramienta
   * @param velocidad velocidad del fresado
   * @param tipo tipo de fresado
   * @param alturaSeguridad altura de seguridad de la fresadora
   * @param profundidadPasada profundidad de pasada de la fresadora
   * @param profundidadTotal profundidad total de la fresadora
   * @param velocidadEntrada velocidad de entrada de la fresadora
   */
  calcularTrayectoria(diametro: number, velocidad: number, tipo: String,
alturaSeguridad: number, profundidadPasada: number, profundidadTotal:
number, velocidadEntrada: number)
```

Especificación del caso de prueba	Habiendo dibujado un cuadrado y un triángulo de tamaño 200 previamente, se llamará a la función con los parámetros de fresado introducidos
Resultado esperado	Enviará al componente Trayectoria los parámetros introducidos y la lista de figuras dibujadas. Después cambiará de pestaña
Resultado obtenido	Ha enviado al componente Trayectoria los parámetros introducidos y la lista de figuras dibujadas. Después ha cambiado de pestaña

Tabla 60. Prueba unitaria de calcularTrayectoria. Segunda iteración

```
  /**
   * Función de los controles de la animación de trayectorias.
   * Es la encargada de reproducir la animación.
   *
   */
  play()
```

Especificación del caso de prueba	Habiendo ya calculado todos los puntos de la trayectoria de fresado se llama a la función
Resultado esperado	Dibujará en la pantalla mediante una animación todos los puntos calculados
Resultado obtenido	Dibuja los puntos calculados menos el último de cada figura

Tabla 61. Prueba unitaria de play

```
  /**
  * Función encargada de ir representando las trayectorias de cada una
de las figuras.
  *
  */
 dibujarTrayectorias()
```

Especificación del caso de prueba	Teniendo las figuras dibujadas, el tamaño de herramienta, la velocidad y al tipo de fresado previamente, se llamará a la función
Resultado esperado	Obtendrá los puntos por los que tendrá que pasar la trayectoria
Resultado obtenido	Ha obtenido los puntos por los que tendrá que pasar la trayectoria

Tabla 62. Prueba unitaria de dibujarTrayectorias. Segunda iteracion

```
  /**
  * Función que calcula los puntos de la trayectoria seleccionada para
una figura dada.
  *
  * @param line línea que representará la trayectoria
  * @param i figura sobre la que se hacen los cálculos
  * @param offset radio de la herramienta
  * @param flag bandera que indica si se calcula una figura (true) o la
línea de transición (false)
  * @returns última trayectoria en el caso del vaciado y la única en el
caso de perfilado
  */
 updatePositions(line: any, i: number, offset: number, flag: boolean)
```

Especificación del caso de prueba	Teniendo las figuras dibujadas, el tamaño de herramienta, la velocidad y al tipo de fresado previamente, se llamará a la función con el objeto en el cual se guardarán los puntos, el índice de la figura que se calcula, el radio de la herramienta y una bandera que indica que se están calculando los puntos de una figura
Resultado esperado	Obtendrá los puntos por los que tendrá que pasar la trayectoria para hacer el fresado de la figura
Resultado obtenido	No obtiene correctamente los puntos porque no pasa por el centro y no calcula bien las pasadas para profundizar

Tabla 63. Prueba unitaria de updatePositions. Segunda iteración

Después de realizar las pruebas han ayudado a encontrar algunos fallos:

1. En la función updatePositions, añadía al GCode la línea de profundizar para una pasada más en el último punto del vaciado, en vez de en el primero.
2. En la función updatePositions, si la herramienta es muy grande deja el centro de la figura sin vaciar.
3. En la función updatePositions, si la profundidad total no es múltiplo de la de pasada, hacía una pasada de menos.
4. En la función play, no dibuja la última línea de la trayectoria de cada figura.

Una vez corregidos los fallos, se tiene la certeza de que las funciones probadas unitariamente tiene un correcto funcionamiento.

7.3.2 Pruebas de integración

Las pruebas de integración son aquellas que sirven para comprobar que la aplicación cumple los casos de uso redactados en el análisis. Con ellas se comprueba el correcto funcionamiento del conjunto de la aplicación, como la comunicación entre las funciones probadas en los test unitarios o la comunicación entre los componentes.

Al igual que las unitarias, las pruebas de integración pueden ser automatizadas o manuales. Pero dada la naturaleza de la aplicación es necesario observar los cambios y el correcto funcionamiento en el lienzo de dibujo, por lo que no hay opción de elegir, es necesario que sean manuales.

7.3.2.1 1ª Iteración

Los casos de uso a probar son los siguientes:

- Dibujar círculo
- Dibujar cuadrado
- Dibujar triángulo

- Mover figura
- Girar figura
- Escalar figura
- Cambiar broca
- Cambiar velocidad
- Trayectoria de perfilado
- Trayectoria de vaciado
- Exportar código G

Las pruebas se describen a continuación

	Dibujar círculo
Especificación del caso de prueba	Se introducirá un tamaño inicial de 200 y se seleccionará crear un circulo
Resultado esperado	Se creará un círculo de radio 200 en el origen de coordenadas
Resultado obtenido	Se ha creado un círculo de radio 200 con centro en el origen de coordenadas

Tabla 64. Prueba de integración de Dibujar círculo

	Dibujar cuadrado
Especificación del caso de prueba	Se introducirá un tamaño inicial de 200 y se seleccionará crear un cuadrado
Resultado esperado	Se creará un cuadrado de lado 200 con centro en el origen de coordenadas
Resultado obtenido	Se ha creado un cuadrado de lado 200 con centro en el origen de coordenadas

Tabla 65. Prueba de integración de Dibujar cuadrado

	Dibujar triángulo
Especificación del caso de prueba	Se introducirá un tamaño inicial de 200 y se seleccionará crear un triángulo
Resultado esperado	Se creará un triángulo de lado 200 con centro en el origen de coordenadas
Resultado obtenido	Se ha creado un triángulo de lado 200 con centro en el origen de coordenadas

Tabla 66. Prueba de integración de Dibujar triángulo

	Mover figura
Especificación del caso de prueba	Teniendo un cuadrado de tamaño 200 previamente dibujado, se moverá a las coordenadas (100, 100)
Resultado esperado	El centro de la figura se moverá del origen de coordenadas a las coordenadas (100, 100)
Resultado obtenido	El centro de la figura se ha movido del origen de coordenadas a las coordenadas (100, 100)

Tabla 67. Prueba de integración de Mover figura

	Girar figura
Especificación del caso de prueba	Teniendo un cuadrado de tamaño 200 previamente dibujado, se girará 30º
Resultado esperado	La figura se girará 30º
Resultado obtenido	La figura se ha girado 30º

Tabla 68. Prueba de integración de Girar figura

	Escalar figura
Especificación del caso de prueba	Teniendo un cuadrado de tamaño 200 previamente dibujado, se escalará un 200%
Resultado esperado	La figura se escalará un 200% acabando con un tamaño de 400
Resultado obtenido	La figura se ha escalado un 200% acabando con un tamaño de 400

Tabla 69. Prueba de integración de Escalar figura

	Cambiar broca
Especificación del caso de prueba	Se introducirá un tamaño de broca de 20 mm
Resultado esperado	El tamaño de broca será de 20 mm
Resultado obtenido	El tamaño de broca es de 20 mm

Tabla 70. Prueba de integración de Cambiar broca

	Cambiar velocidad
Especificación del caso de prueba	Se introducirá una velocidad de 4000 mm/min
Resultado esperado	La velocidad será de 4000 mm/min
Resultado obtenido	La velocidad es de 4000 mm/min

Tabla 71. Prueba de integración de Cambiar velocidad

	Trayectoria de perfilado
Especificación del caso de prueba	Se seleccionará que el tipo de fresado será un perfilado
Resultado esperado	El tipo de fresado será perfilado
Resultado obtenido	El tipo de fresado es perfilado

Tabla 72. Prueba de integración de Trayectoria de perfilado

	Trayectoria de vaciado
Especificación del caso de prueba	Se seleccionará que el tipo de fresado será un vaciado
Resultado esperado	El tipo de fresado será vaciado
Resultado obtenido	El tipo de fresado es vaciado

Tabla 73. Prueba de integración de Trayectoria de vaciado

	Exportar código G
Especificación del caso de prueba	Habiendo calculado la trayectoria de un dibujo previamente, se seleccionará generar un GCode
Resultado esperado	Se habrá descargado el archivo con el GCode para fresar la trayectoria generada
Resultado obtenido	Se ha descargado el archivo con el GCode para fresar la trayectoria generada

Tabla 74. Prueba de integración de Exportar código G

En todas las pruebas se ha obtenido el resultado esperado, por lo que corroboran que los casos de uso se cumplen a la perfección.

7.3.2.2 2ª Iteración

Los casos de uso a probar son los siguientes (los de la primera iteración no se prueban porque ya fueron probados):

- Introducir altura de seguridad
- Introducir profundidad de pasada
- Introducir profundidad total
- Introducir velocidad de entrada
- Reproducir animación de la trayectoria
- Pausar animación de la trayectoria
- Reiniciar animación de la trayectoria
- Seleccionar figura
- Deshacer
- Rehacer
- Borrar figura

Las pruebas se describen a continuación

	Introducir altura de seguridad
Especificación del caso de prueba	Se introducirá una altura de seguridad de 10 mm
Resultado esperado	La altura de seguridad será de 10 mm
Resultado obtenido	La altura de seguridad es de 10 mm

Tabla 75. Prueba de integración de Introducir altura de seguridad

	Introducir profundidad de pasada
Especificación del caso de prueba	Se introducirá una profundidad de pasada de 2 mm
Resultado esperado	La profundidad de pasada será de 2 mm
Resultado obtenido	La profundidad de pasada es de 2 mm

Tabla 76. Prueba de integración de Introducir profundidad de pasada

	Introducir profundidad total
Especificación del caso de prueba	Se introducirá una profundidad total de 7 mm
Resultado esperado	La profundidad total será de 7 mm
Resultado obtenido	La profundidad total es de 7 mm

Tabla 77. Prueba de integración de Introducir profundidad total

	Introducir velocidad de entrada
Especificación del caso de prueba	Se introducirá una velocidad de entrada de 100 mm/min
Resultado esperado	La velocidad de entrada será de 100 mm/min
Resultado obtenido	La velocidad de entrada es de 100 mm/min

Tabla 78. Prueba de integración de Introducir velocidad de entrada

	Reproducir animación de la trayectoria
Especificación del caso de prueba	Habiendo calculado la trayectoria de un dibujo previamente, se pulsará el botón de reproducir la animación
Resultado esperado	Se reproducirá la animación de la trayectoria hasta que finalice
Resultado obtenido	Se ha reproducido la animación de la trayectoria hasta que ha finalizado

Tabla 79. Prueba de integración de Reproducir animación de la trayectoria

	Pausar animación de la trayectoria
Especificación del caso de prueba	Habiendo calculado la trayectoria de un dibujo y reproducido la animación previamente, se pulsará el botón de pausar la animación antes de que finalice
Resultado esperado	Se pausará la animación de la trayectoria
Resultado obtenido	Se ha pausado la animación de la trayectoria

Tabla 80. Prueba de integración de Pausar animación de la trayectoria

	Reiniciar animación de la trayectoria
Especificación del caso de prueba	Habiendo calculado la trayectoria de un dibujo y reproducido la animación hasta que ha finalizado previamente, se pulsará el botón de reiniciar la animación
Resultado esperado	Se reiniciará la animación borrando las líneas previamente dibujadas al reproducirla
Resultado obtenido	Se ha reiniciado la animación borrando las líneas previamente dibujadas al reproducirla

Tabla 81. Prueba de integración de Reiniciar animación de la trayectoria

	Seleccionar figura
Especificación del caso de prueba	Habiendo dibujado un cuadrado y un triángulo de tamaño 200 previamente, se seleccionará el cuadrado
Resultado esperado	El cuadrado estará seleccionado
Resultado obtenido	El cuadrado está seleccionado

Tabla 82. Prueba de integración de Seleccionar figura

	Deshacer
Especificación del caso de prueba	Habiendo dibujado un cuadrado de tamaño 200 previamente, se deshará
Resultado esperado	Se deshará la creación del cuadrado
Resultado obtenido	Se ha deshecho la creación del cuadrado

Tabla 83. Prueba de integración de Deshacer

	Rehacer
Especificación del caso de prueba	Habiendo dibujado un cuadrado de tamaño 200 y deshecho previamente, se rehará
Resultado esperado	Se rehará la creación del cuadrado
Resultado obtenido	Se ha rehecho la creación del cuadrado

Tabla 84. Prueba de integración de Rehacer

	Borrar figura
Especificación del caso de prueba	Habiendo dibujado un cuadrado de tamaño 200 previamente, se borrará
Resultado esperado	Se borrará el cuadrado
Resultado obtenido	Se ha borrado el cuadrado

Tabla 85. Prueba de integración de Borrar figura

En todas las pruebas se ha obtenido el resultado esperado, por lo que corroboran que los casos de uso se cumplen a la perfección.

7.3.3 Pruebas de usabilidad

Las pruebas de sistema sirven para comprobar la usabilidad del sistema. El requisito no funcional RNF05 decía al respecto: Un usuario sin experiencia deberá ser capaz de usar la aplicación en media hora con la ayuda y explicaciones de alguien con experiencia. Para ello se realizarán unas pruebas con usuarios sin experiencia en este tipo de software, como podrían ser los artesanos de madera que serán los usuarios finales de la aplicación.

Estas pruebas se realizarán con la aplicación desplegada en un servidor de pruebas y consistirán en una pequeña introducción explicando el funcionamiento de la aplicación y conceptos básicos del fresado, seguido de un tiempo para que el usuario pruebe la aplicación y se acostumbre a los controles con ayuda de alguien con experiencia, en este caso, seré yo. Por último, se le pedirá que realice un diseño él solo, para así comprobar si en media hora un usuario sin experiencia podría aprender a usar la aplicación.

El diseño que se pide al usuario que realice el final es el siguiente:

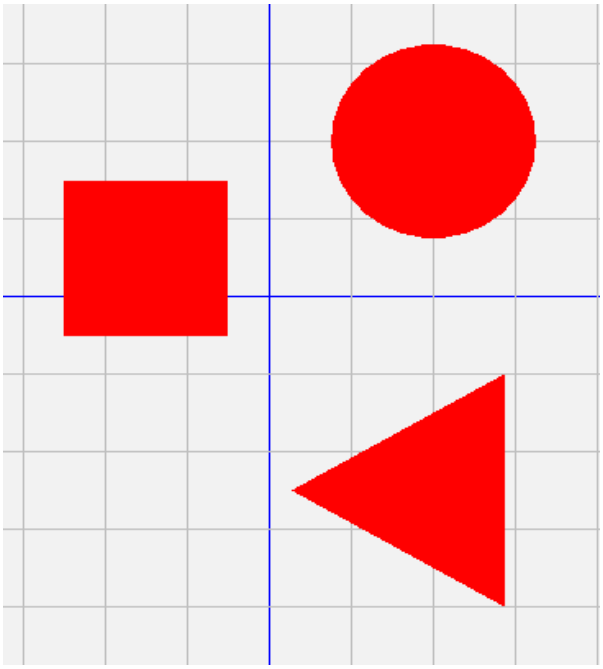

Figura 58. Dibujo de pruebas

También se le han dado los siguientes datos para que pueda hacerlo correctamente:

- El cuadrado tiene tamaño 200 mm y está en las coordenadas (-150, 50).
- El círculo tiene tamaño 250 mm y está en las coordenadas (200, 200).
- El triángulo tiene tamaño 300 mm, está en las coordenadas (200, -250) y está girado un ángulo de 30 grados.

Después del dibujo, debe generar una trayectoria de fresado y descargar el GCode. Los parámetros de fresado, también dados al usuario, son los siguientes:

- Diámetro de la herramienta: 20 mm.
- Velocidad de fresado: 4000 mm/min.
- Tipo de fresado: vaciado.
- Altura de seguridad: 20 mm.
- Profundidad de pasada: 2 mm.
- Profundidad total: 7 mm.
- Velocidad de entrada: 100 mm/min.

Además, se les pedirá que realicen las acciones de borrar, deshacer/rehacer y controlar la animación de la trayectoria (reproducir, parar o reiniciar) si no las han llegado a usar mientras realizaban el dibujo que se les pedía, para así probar todas las opciones que ofrece la aplicación.

Finalmente se han realizado las pruebas con 10 usuarios, todos ellos con un perfil como el usuario final de la aplicación. Una vez realizadas las pruebas, anoté algunos comentarios de los usuarios que la han probado, para así mejorarla en un futuro. Son los siguientes:

- Algunos usuarios no conocen (o no recuerdan) el concepto de plano cartesiano, coordenadas, centro de origen, etc. No es necesario cambiar nada de la aplicación, pero sí preguntar antes de empezar a explicar por si hay que explicarlo antes de todo.
- Algunos usuarios no conocen el concepto de “recargar página”, necesario para reiniciar la aplicación. Hay que explicarlo cuando se enseña a usar la aplicación, al menos hasta que se implemente alguna manera de volver a empezar o volver atrás dentro de la aplicación.
- Puede no quedar muy claro que, al mover una figura, lo que se mueve es el centro. Se soluciona fácilmente añadiéndolo en la explicación inicial.
- Muchos usuarios fallan a la hora de crear figuras. Casi todos dar al botón de crear figura sin acordarse del campo tamaño a pesar de haberse explicado cómo funciona, por lo que, o les da un error si el campo estaba vacío, o crean una figura del tamaño que ya estaba previamente (y no suelen darse cuenta). Esto requiere de rediseñar la manera de crear las figuras para hacerlo más intuitivo.

A pesar de estos pequeños fallos, todos los usuarios han sido capaces de aprender a usar la aplicación en media hora con la explicación y ayuda de alguien con experiencia, como especificaba el requisito RNF05.

Además de los resultados de las pruebas se han encontrado diversos errores en la aplicación, detallados a continuación:

- No funcionaba la validación para los cuatro parámetros añadidos en la segunda iteración (altura de seguridad, profundidad de pasada, profundidad total y velocidad de entrada).
- Habiendo mínimo dos figuras dibujadas, al borrar una, deshacer, rehacer y volver a deshacer, da error. Crea mal las figuras de nuevo y todo después no funciona como debería.
- Cuando se borraba un elemento de la lista de figuras (que no fuera el último) al deshacer el cambio se quedaban seleccionadas en el lienzo dos figuras. Además, si después de esto se seguía, la trayectoria la hacía con la figura borrada y sin la última de la lista.

Todos los errores detectados fueron solventados una vez acabadas las pruebas de sistema.

Capítulo 8

Conclusiones

8.1 Conclusiones

Tras finalizar el proyecto se procede a valorar la experiencia, primero valorando los objetivos definidos al principio de éste y posteriormente a nivel más personal.

Los objetivos que se definieron al principio del proyecto eran los siguientes:

- Buscar, ordenar y estructurar información para la realización de un proyecto informático.
- Realizar un trabajo teniendo un cliente real que tiene sus propias necesidades e intereses.
- Elaborar una memoria completa de un proyecto informático.
- Elaborar y defender una presentación pública del trabajo realizado.
- Estudiar cómo funciona una fresadora y cómo se controla.
- Estudiar el código G y aprender a usarlo para realizar los trabajos deseados.
- Estudiar el framework elegido para realizar la aplicación, así como las librerías necesarias.
- Permitir dibujar y transformar diseños simples en dos dimensiones.
- Permitir generar trayectorias de trabajos de fresado a partir del diseño y ciertos parámetros necesarios para la fresadora.
- Permitir generar el código para una máquina fresadora que realice el trabajo.
- Preparar la aplicación para su evolución a lo largo del tiempo.
- Desplegar la aplicación una vez ya acabada en un servidor.
- Conseguir una aplicación fácil de usar para los usuarios finales de la aplicación, la mayoría sin mucha experiencia con las tecnologías.

A excepción de la exposición (que se hará cuando haya acabado el proyecto), se han cumplido todos los objetivos planteados. Ha sido necesario documentarme y repasar conceptos estudiados durante toda la carrera para refrescar los conocimientos que han hecho falta para el proyecto. Se ha experimentado el realizar un proyecto para un cliente, teniendo que reunirse con él al final de las iteraciones para recibir retroalimentación, cambio de requisitos, añadir otros nuevos, etc. Por último, ha habido que documentar todo el proyecto en la presente memoria, con la orientación de las tutoras y leyendo TFG realizados por alumnos previamente, para buscar orientación en los contenidos y las formas de un documento de este tipo.

Aparte de los objetivos de formación, se han logrado completar todos los objetivos de la aplicación. Estos son desde comprender todo lo necesario para realizar la aplicación (como estudiar el funcionamiento de una fresadora, el código G o el framework y las librerías) hasta los que describían la finalidad de la aplicación (las funciones de dibujo, generación de trayectorias y la generación del GCode).

Además de los objetivos, también hay otros aspectos importantes a destacar. Se ha desarrollado toda la aplicación en tecnologías no usadas antes. Tanto Angular como JavaScript eran desconocidas, en tanto a que era la primera vez que se usaban. Es importante, porque implica que se han adquirido los conocimientos necesarios para aprender y desarrollar conocimientos por cuenta propia a partir de la base adquirida en la universidad.

Se ha aprendido a trabajar por cuenta propia. Antes se estaba acostumbrado a tener alguna fecha límite para entregar prácticas, ahora todo dependía de uno mismo. El poder realizarla en un año o en tres meses, dependía de uno mismo. Ese ha sido un aspecto bastante nuevo y ha costado adaptarse a la nueva dinámica, pero se ha aprendido a planificarse mejor y a trabajar de una manera nueva hasta el momento.

Y por último se ha aprendido a trabajar solo. Hasta ahora, prácticamente siempre se había trabajado en equipo, delegando tareas, aportando distintos tipos de perspectiva y repartiéndose el trabajo. Por lo que desarrollar un proyecto de esta envergadura por cuenta propia, a pesar de tener la orientación de las tutoras, ha sido un cambio muy grande. Al principio se veía inabarcable pero poco a poco se iba cambiando la mentalidad y, echando la vista atrás se veía mejor el progreso que se había conseguido hasta el momento, así que se veía que, aunque no se sintiera mucho progreso mientras se desarrolla, todo se va sumando hasta que se consigue completarlo.

8.2 Trabajo futuro

Las mejoras futuras a la aplicación son aquellas funcionalidades que el cliente quería para la aplicación, pero fueron descartadas para ajustarnos a las 300 horas que debiera llevar un TFG. Estas mejoras serían:

- Creación de un nuevo apartado para representar en 3D la animación de la trayectoria de la fresadora. Para poder ver la profundidad de las pasadas, simular una herramienta para poder ver su movimiento, etc.
- Ampliación de las figuras disponibles para dibujar. Por eso se creó un grupo de botones, con la intención de poder añadir más en el futuro.
- Crear la unión booleana de figuras. Para así poder combinarlas y crear dibujos más complejos.
- Cargar dibujos vectoriales. Dada la limitación de dibujar figuras tan básicas, para formas y composiciones más complejas sería mejor importarlas ya hechas, y así la aplicación sólo se preocuparía de generar las trayectorias para la fresadora.
- Crear nuevos tipos de fresado. Se eligieron únicamente el perfilado y vaciado para no sobrecargar el TFG, pero se podrían añadir nuevos tipos, como por ejemplo un perfilado, pero por el interior.

Además de lo propuesto por los clientes, surgen algunas propuestas para mejorar la usabilidad, como resultado de las pruebas de sistema. También se añaden líneas de futuro trabajo que se ocurren lógicas para la expansión de la aplicación.

- Rediseñar la forma de crear figuras. No queda muy clara, la mayoría de los usuarios sólo se queda con dar al botón de la figura para crearla, olvidándose de introducir antes el tamaño.
- Mejorar la navegabilidad de la aplicación. Algo que en este momento no se ha tenido en cuenta, pero para muchas personas el concepto de recargar la página es desconocido, por lo que debería incluirse en la página alguna manera de reiniciar todo. Además, sería una buena idea implementar de alguna manera el volver atrás desde la parte de trayectoria a la de dibujo, por si se quisiera modificar el que ya se tenía hecho.
- Mejora de los controles de animación de trayectorias. Podrían mejorarse los controles añadiendo la velocidad a la que quieres que se vea, una barra donde te muestre el progreso y la puedas manejar para ir atrás o adelante, etc.
- Introducir controles de ratón para el dibujo. Como podría ser pinchar sobre las figuras para seleccionarlas, mover las figuras arrastrándolas por el lienzo, girarlas pinchando en algún punto y moviendo o redimensionarlas de la misma manera.
- Visualización del lienzo. Añadir un zoom, para ayudar si las figuras son muy pequeñas.

Bibliografía

[1] Angular (2019). *Angular*. [online] Angular.io. Available at: https://angular.io/docs [Accessed 13 May 2019].

[2] Angular CLI (2019). *Angular CLI*. [online] Cli.angular.io. Available at: https://cli.angular.io/ [Accessed 13 May 2019].

[3] Fuente de la imagen: https://www.axyz.com/us/2018/01/16/plastic-fabrication-using-cnc-routers-part-one/

[4] Bootstrap (2019). *Introduction*. [online] Getbootstrap.com. Available at: https://getbootstrap.com/docs/4.0/getting-started/introduction/ [Accessed 13 May 2019].

[5] Es.wikipedia.org. (2019). *G-code*. [online] Available at: https://es.wikipedia.org/wiki/G-code [Accessed 25 Jun. 2019].

[6] Fuente de la imagen: https://www.europages.pt/Gravacao-a-FresaCNC/SOLASER-JPFREITAS-LDA/cpid-5453568.html

[7] Fuente de la imagen: https://eurotools-industrie.ro/masini-de-frezat-metal-cnc/7264-masina-de-frezat-cnc-optimum-f-105-808-d.html

[8] Diputación de Valladolid, (2019). [online] Available at: https://bop.sede.diputaciondevalladolid.es/boletines/2018/febrero/06/BOPVA-B-2018-026.pdf [Accessed 8 Jul. 2019].

[9] Fuente de la imagen: https://www.indiamart.com/proddetail/cnc-milling-machine-14874331991.html

[10] Git (2019). *About - Git*. [online] Git-scm.com. Available at: https://git-scm.com/about [Accessed 13 May 2019].

[11] Fuente de la imagen: https://medium.com/@PavelLaptev/three-js-for-beginers-32ce451aabda

[12] Fuente de la imagen: https://articulo.mercadolibre.com.ar/MLA-673205046-router-cnc-60x60-fresadora-pantografo-_JM

[13] Node.js, F. (2019). *Acerca | Node.js*. [online] Node.js. Available at: https://nodejs.org/es/about/ [Accessed 13 May 2019].

[14] Fuente de la imagen: https://nomadtech.es/es/

[15] Fuente de la imagen: https://nomadtech.es/es/content/27-que-puedo-hacer-con-mi-fox

[16] npm (2019). *About npm | npm Documentation*. [online] Docs.npmjs.com. Available at: https://docs.npmjs.com/about-npm/ [Accessed 13 May 2019].

[17] Threejs.org. (2019). *three.js - Javascript 3D library*. [online] Available at: https://threejs.org/ [Accessed 13 May 2019].

[18] Stack Overflow. (2019). *Stack Overflow Developer Survey 2019*. [online] Available at: https://insights.stackoverflow.com/survey/2019?utm_source=so-owned&utm#technology-_-web-frameworks_medium=blog&utm_campaign=dev-survey-2019&utm_content=launch-blog [Accessed 7 Jul. 2019].

[19] Fuente de la imagen: https://stackoverflow.com/questions/50957349/threejs-how-to-offset-all-points-on-a-2d-geometry-by-distance

[20] Visual Studio Code (2019). *Documentation for Visual Studio Code*. [online] Code.visualstudio.com. Available at: https://code.visualstudio.com/docs [Accessed 13 May 2019].

[21] Zeit.co. (2019). *Now - ZEIT*. [online] Available at: https://zeit.co/now [Accessed 11 Jun. 2019].

Anexos

Apéndice A

Manual de instalación del entorno para desarrolladores

A.1. Requisitos previos

Para preparar el entorno de desarrollo se requiere:

- Node JS versión 10+
- npm versión 6.4+
- git (en caso de descargar desde un repositorio)

A.2. Instalación

Para preparar el entorno de desarrollo:

- Primero habrá que descargar el código fuente mediante un git clone si se quiere descargar desde un repositorio o copiando desde el CD del TFG.
- En la carpeta donde se tenga el código se instalan las dependencias npm. Esto se hace con el siguiente comando:

```
$ npm install
```

Esto es todo lo necesario para tener el entorno de desarrollo preparado.

A.3. Comandos útiles

Una vez esté todo listo, se pueden usar los siguientes comandos, útiles en la labor de desarrollo:

- `ng serve`: Este comando permite desplegar el proyecto en local en un servidor de desarrollo, a través del puerto 4200 de nuestra máquina (localhost:4200 en cualquier navegador).
- `ng build --prod`: Construye la aplicación lista para producción bajo el directorio dist en el directorio que contiene la aplicación.

Apéndice B

Manual de despliegue

Para poder desplegar se necesita un artefacto. Para generarle, teniendo el entorno de desarrollo preparado, se ejecuta el siguiente comando (explicado en el APÉNDICE A.3):

```
$ npm build --prod
```

Una vez ejecutado, dentro del directorio dist habrá una carpeta con el nombre del proyecto. Para desplegar la aplicación habrá que publicar el contenido de esta carpeta en un servidor web.

Apéndice C

Manual de usuario

En el siguiente se muestra cómo utilizar la aplicación. Las explicaciones irán acompañadas de capturas de la propia aplicación para ilustrar los pasos.

C.1. Crear figura

Para crear figuras tenemos la parte superior de la aplicación, donde pone “Nueva figura”.

Figura 59. Crear figura

Para crear una figura es necesario introducir un tamaño antes. Una vez introducido el tamaño se pulsa sobre cualquiera de las figuras que vienen representadas debajo. Una vez creada la figura se verá en el lienzo de la izquierda, además de en la lista que se ha creado a la derecha.

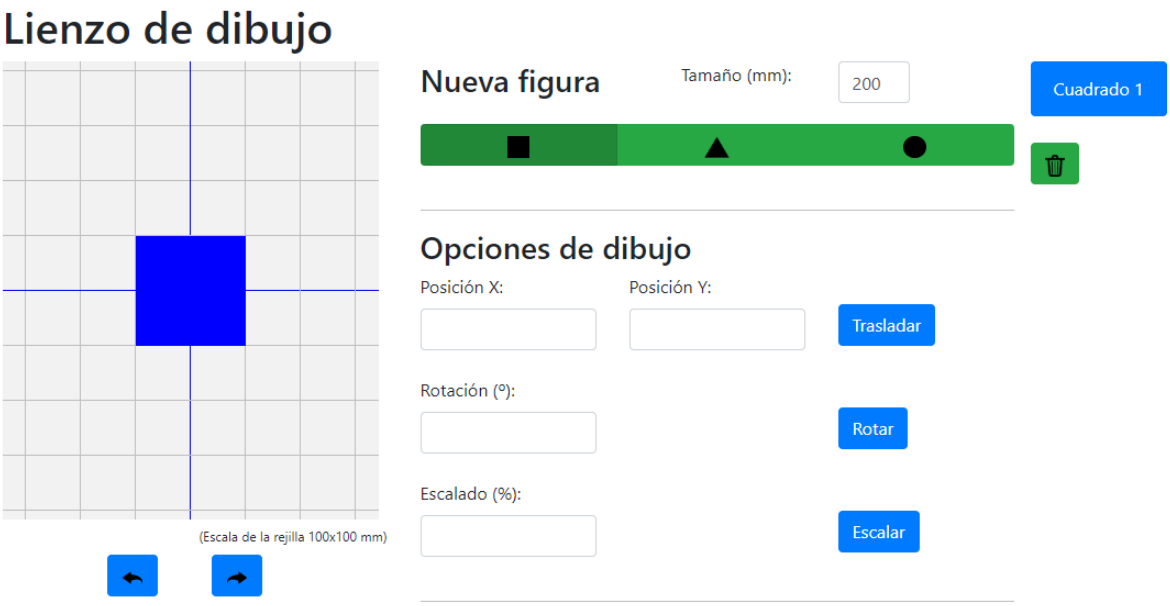

Figura 60. Crear figura

Si se intentara crear una figura sin haber introducido antes el tamaño nos avisaría la aplicación con un error.

Figura 61. Error al crear figura

C.2. Borrar figura

Una vez que haya creadas figuras, aparecerán a la derecha de la aplicación, junto a los botones para crear las figuras.

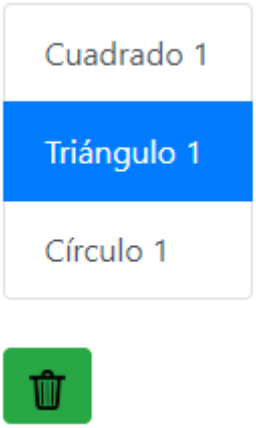

Figura 62. Borrar figura

Para borrar una figura, se selecciona pulsando sobre ella en la lista y después en el icono de la papelera que hay bajo la lista. Una vez borrada desaparecerá tanto de la lista como del lienzo de dibujo.

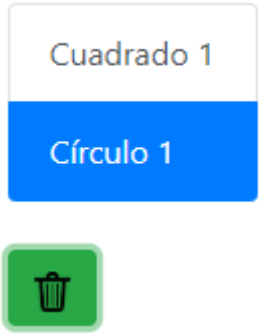

Figura 63. Borrar figura

C.3. Mover figura

Para mover una figura por el lienzo habrá que seleccionarla primero en la lista de la derecha e introducir las coordenadas a las que se desee mover el centro de la figura en los cuadros de texto Posición X y Posición Y, dentro de la parte de opciones de dibujo.

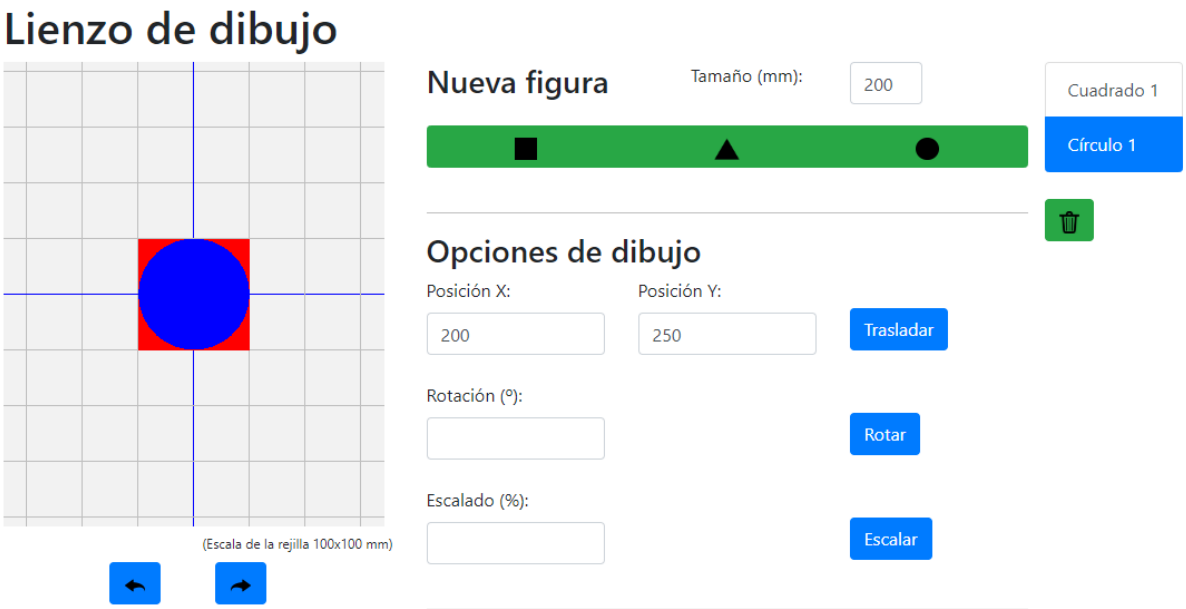

Figura 64. Mover figura

Después de esto, se pulsa en el botón Trasladar y la figura se moverá a las coordenadas que se hayan indicado.

Figura 65. Mover figura

Si se intentase mover sin haber introducido alguna de las coordenadas, la aplicación nos avisaría con un error.

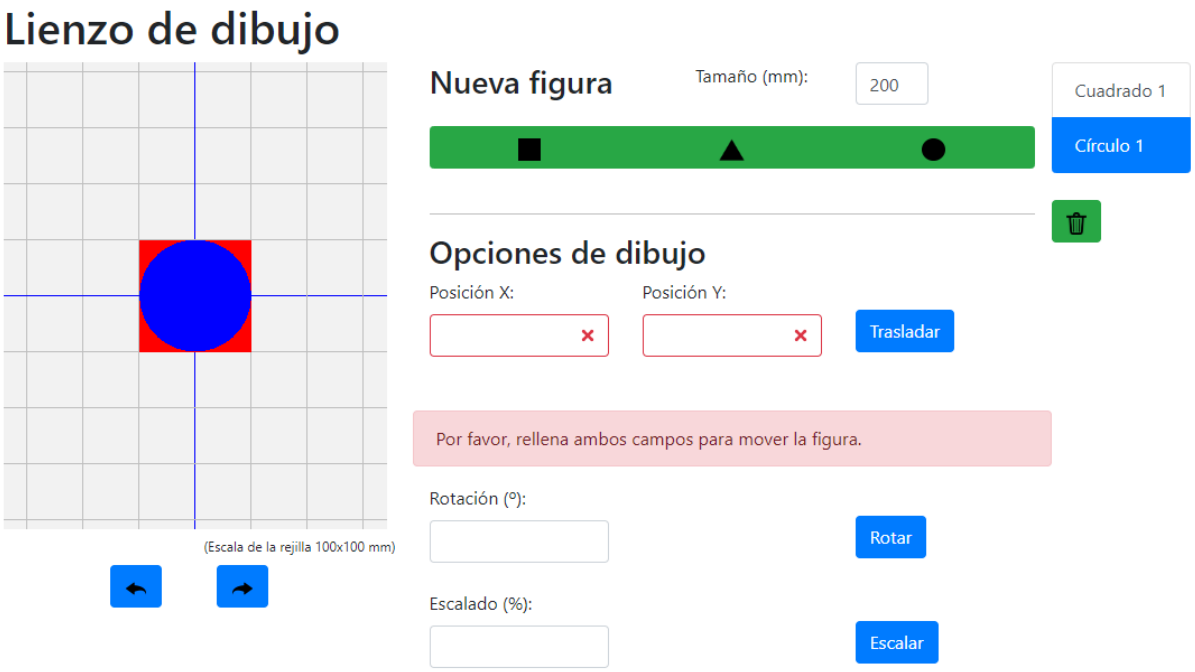

Figura 66. Error al mover figura

C.4. Rotar figura

Para rotar una figura habrá que seleccionarla primero en la lista de la derecha e introducir los grados a los que se desee girar en el cuadro de texto Rotación, dentro de la parte de opciones de dibujo.

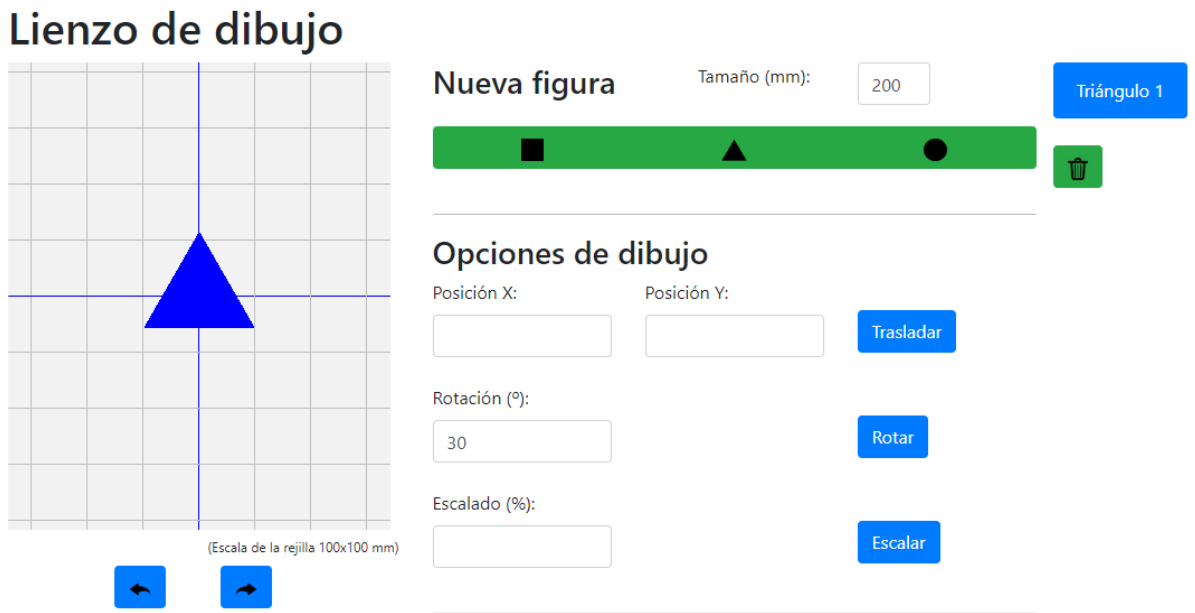

Figura 67. Rotar figura

Después de esto, se pulsa en el botón Rotar y la figura se girará los grados que se hayan indicado.

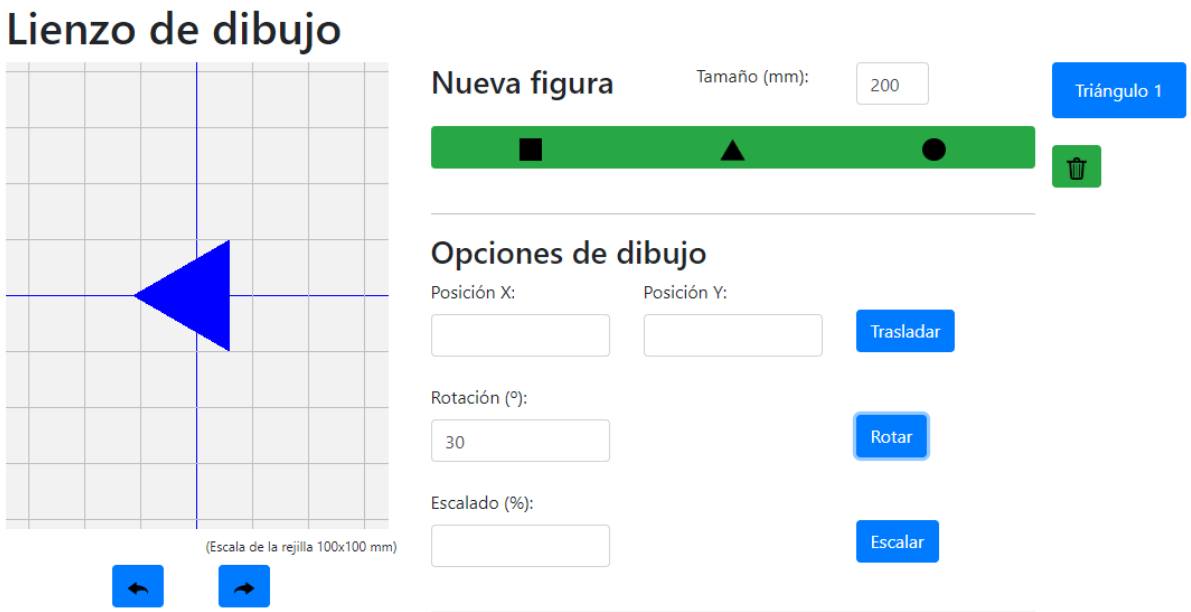

Figura 68. Rotar figura

Si se intentase girar sin haber introducido nada en el campo Rotación, la aplicación nos avisaría con un error.

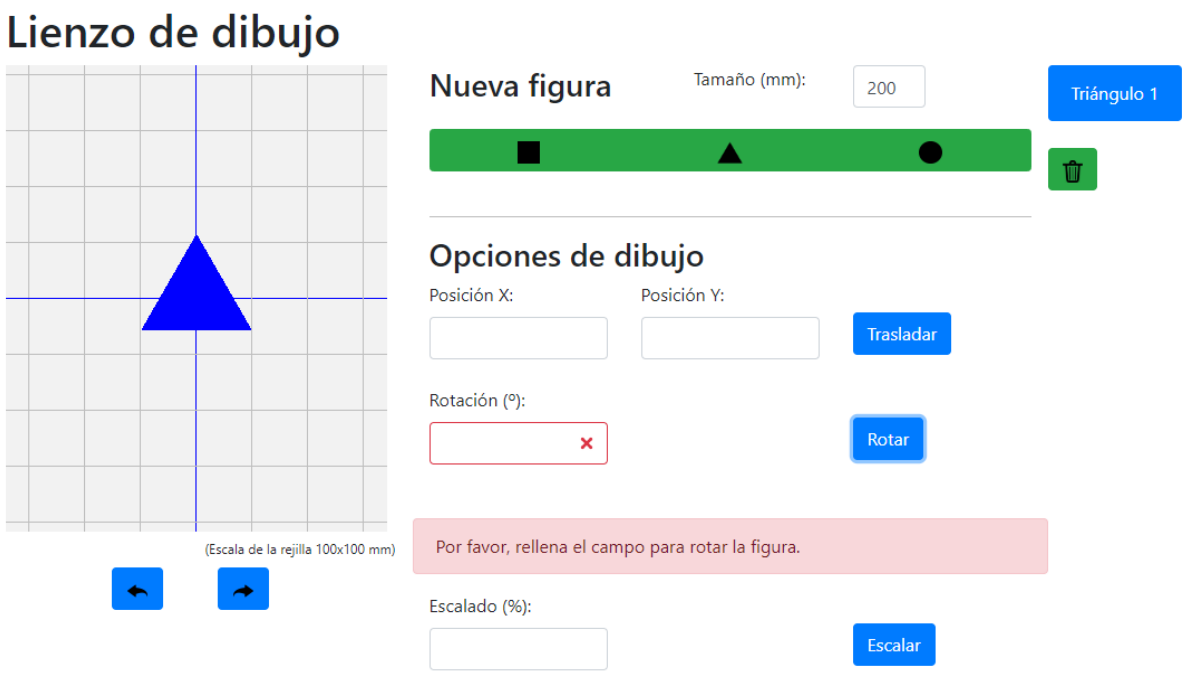

Figura 69. Error al rotar figura

C.5. Escalar figura

Para escalar una figura habrá que seleccionarla primero en la lista de la derecha e introducir el porcentaje al que se desee redimensionar en el cuadro de texto Escalado, dentro de la parte de opciones de dibujo.

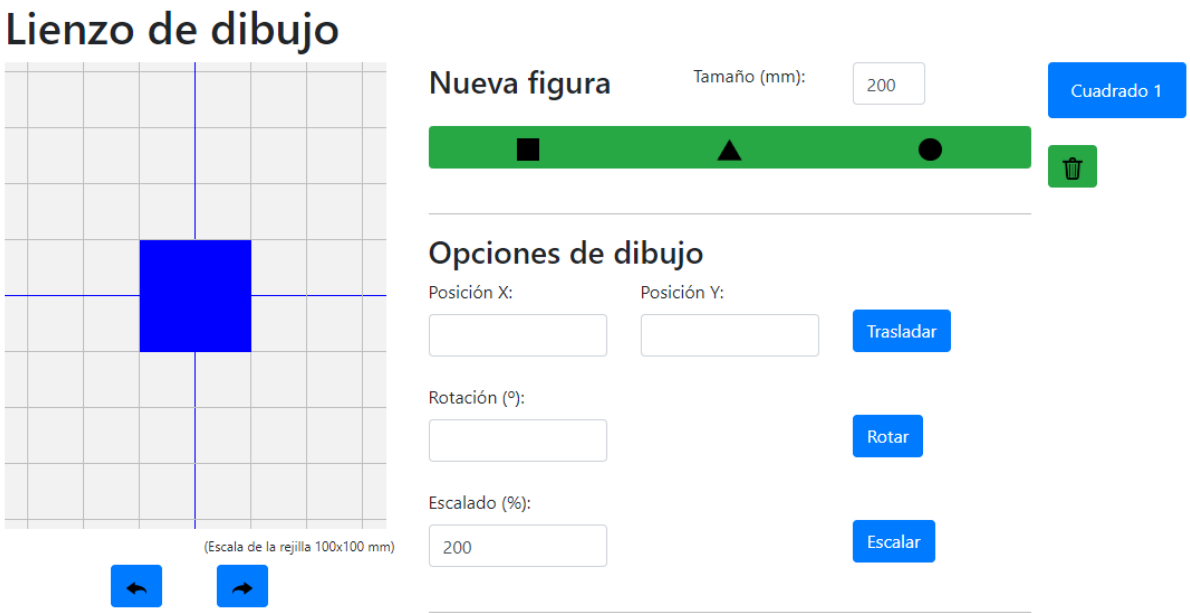

Figura 70. Escalar figura

Después de esto, se pulsa en el botón Escalar y la figura se escalará el porcentaje que se haya indicado.

Figura 71. Escalar figura

Si se intentase escalar sin haber introducido nada en el campo Escalado o si se intentase introducir un número negativo, la aplicación nos avisaría con un error.

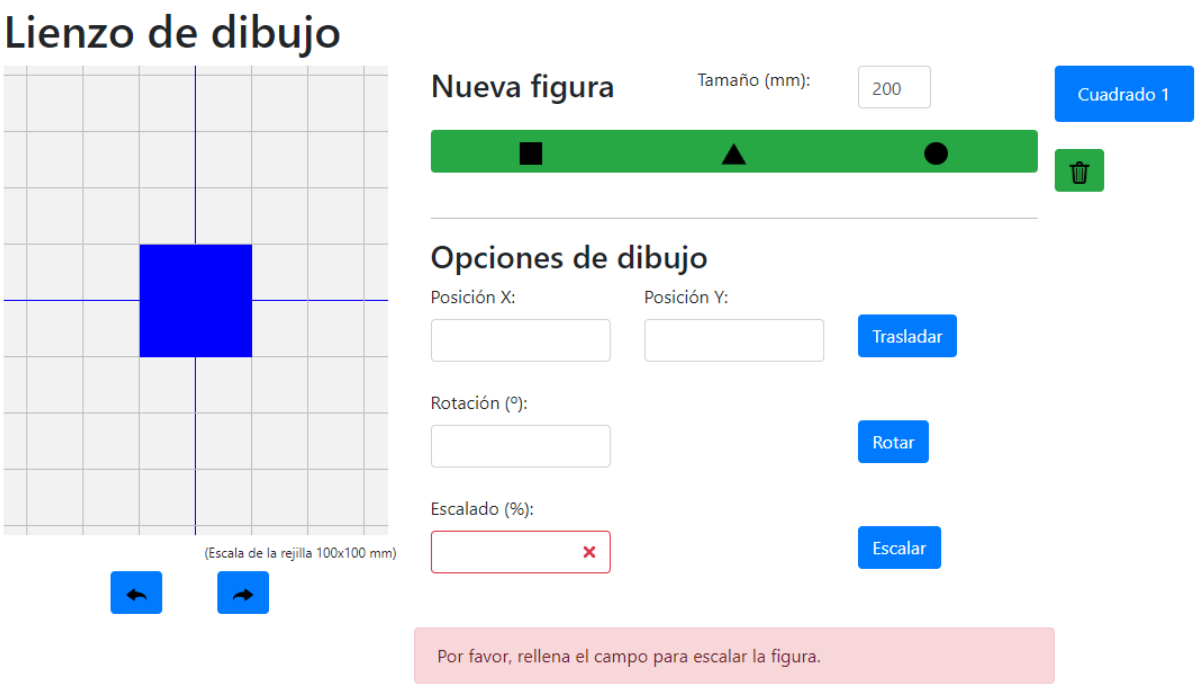

Figura 72. Error al escalar figura

Figura 73. Error al escalar figura

C.6. Deshacer/Rehacer cambios

Se pueden deshacer y rehacer todas las opciones de dibujo, en caso de error o de necesidad por cualquier motivo. Las acciones que se pueden deshacer y rehacer son la creación de figuras, el borrado, la traslación, la rotación y el escalado. Para ello sólo hay que

pulsar los botones que se encuentran bajo el lienzo de dibujo. El botón izquierdo es el botón Deshacer y el derecho el de Rehacer.

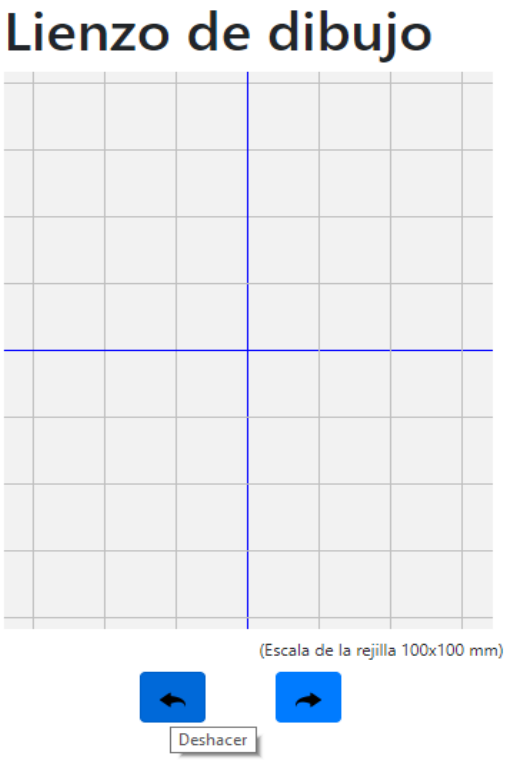

Figura 74. Deshacer cambio

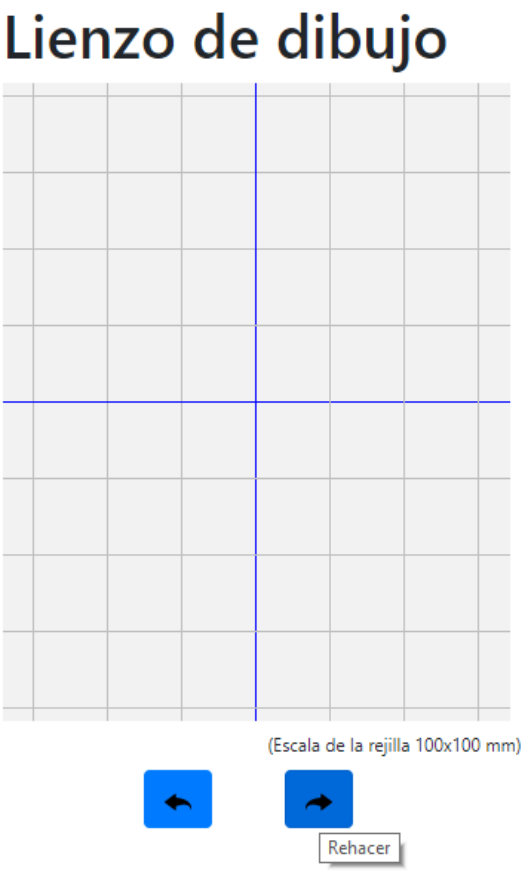

Figura 75. Rehacer cambio

C.7. Introducir opciones de fresado

Una vez finalizado el dibujo, se procede a introducir todos los parámetros necesarios para que la fresadora pueda realizar el dibujo. Cuando ya estén rellenados todos los campos, se pulsa sobre el botón Generar trayectoria.

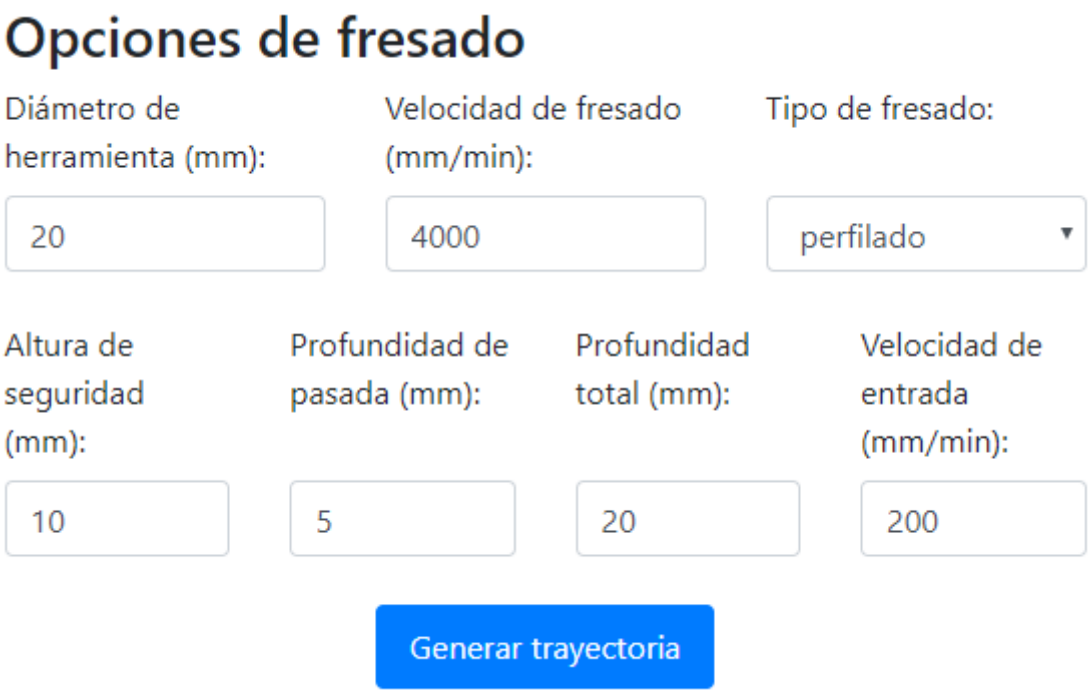

Figura 76. Introducir opciones de fresado

Si se deja algún campo sin rellenar, con un valor negativo o la profundidad de pasada mayor a la profundidad total, la aplicación avisará con errores.

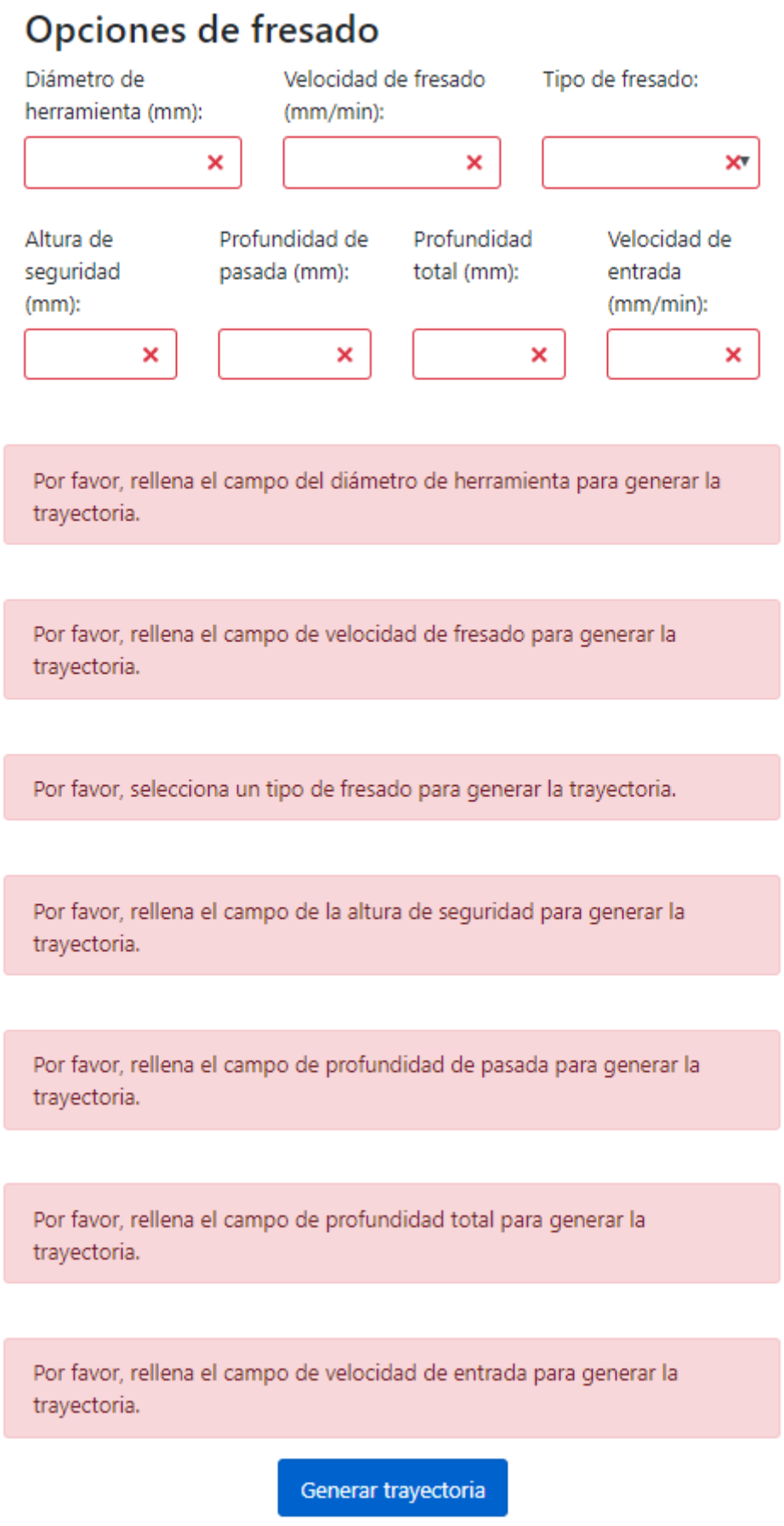

Figura 77. Error al introducir opciones de fresado

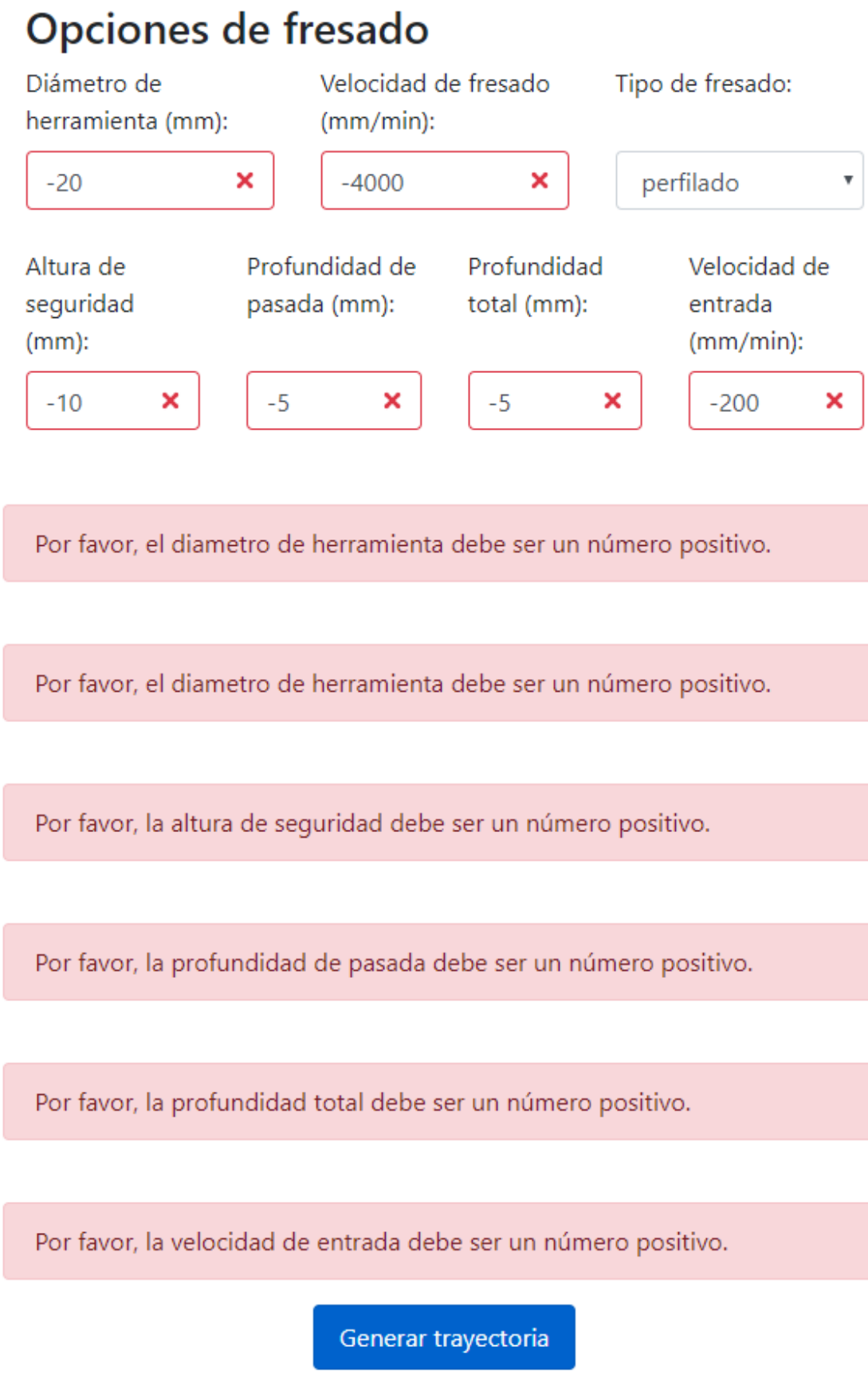

Figura 78. Error al introducir opciones de fresado

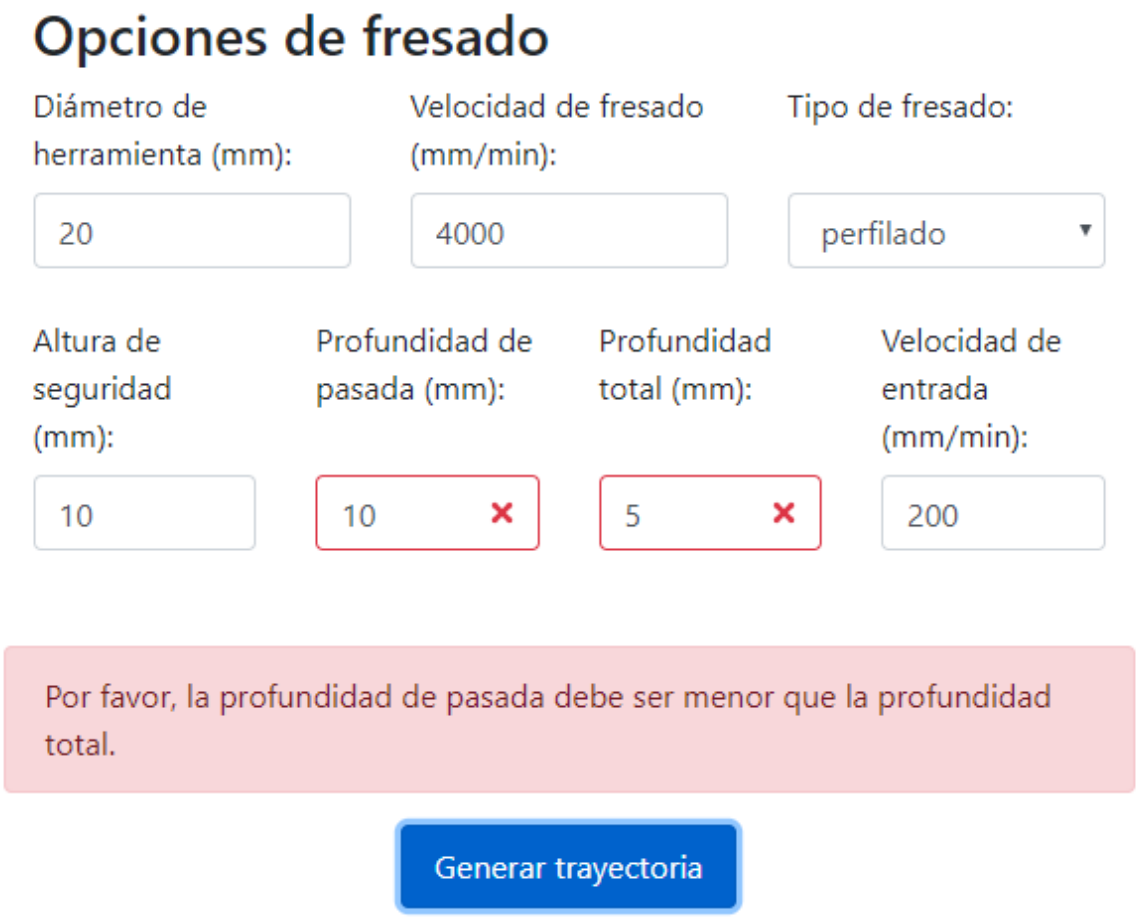

Figura 79. Error al introducir opciones de fresado

C.8. Controles de trayectoria

Una vez generada la trayectoria, llegamos a una nueva pestaña. En ella encontramos el dibujo que habíamos hecho previamente y la aplicación ya ha calculado la trayectoria a partir del dibujo y los parámetros que hemos introducido.

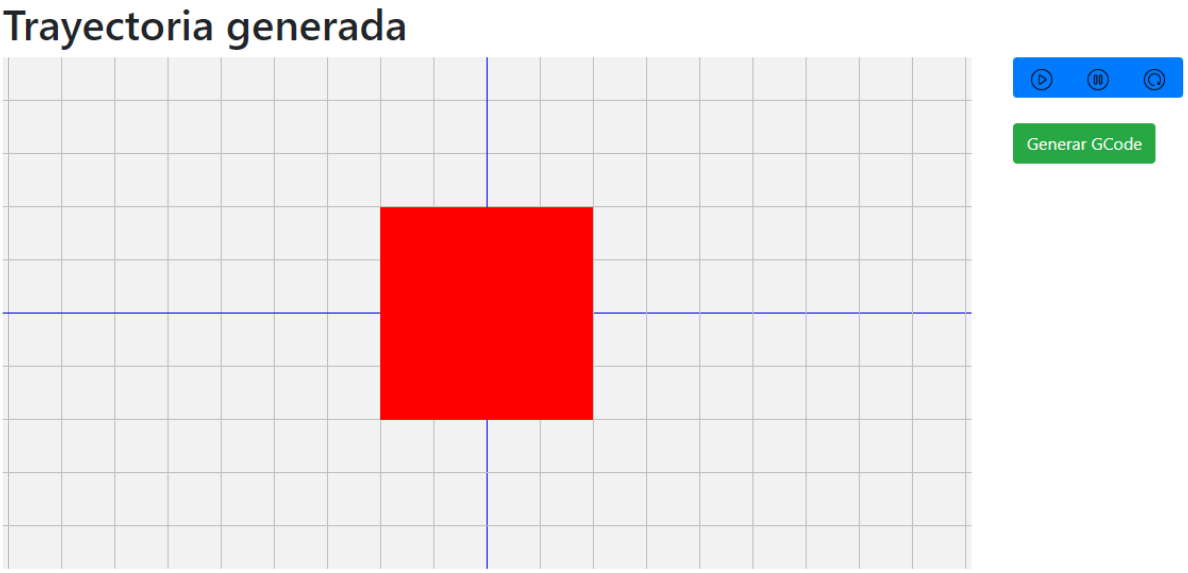

Figura 80. Controles de animación de la trayectoria

Si quisiéramos ver la animación de la trayectoria que seguirá la fresadora para realizar el dibujo, podemos pulsar en el botón Reproducir, el cual está representado por el típico triángulo de “play”.

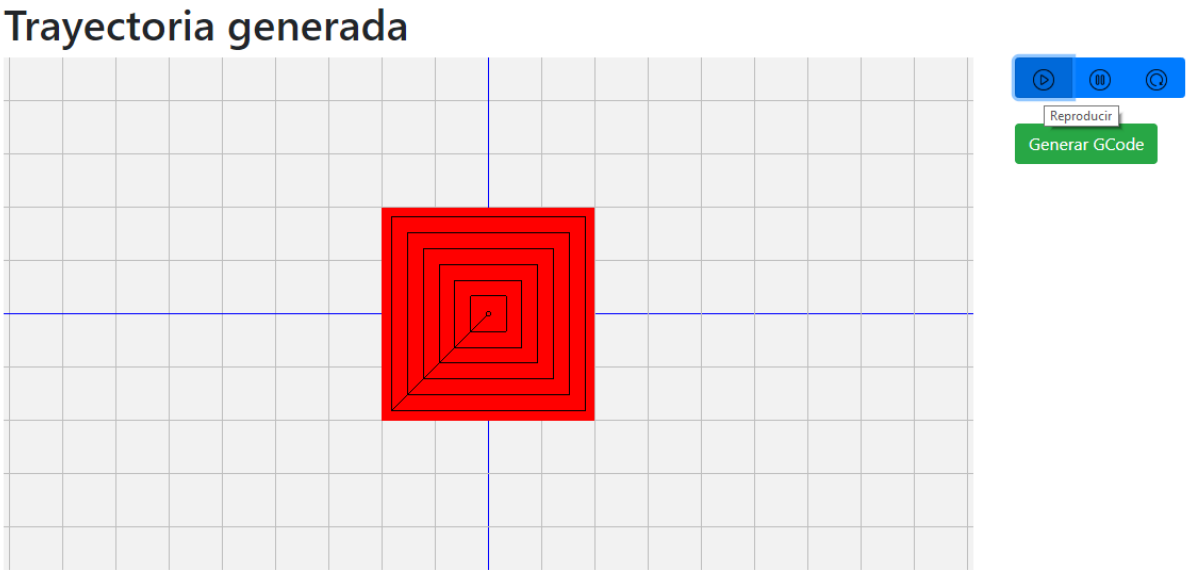

Figura 81. Reproducir animación de trayectoria

En cualquier momento de la animación se puede pausar, pulsando sobre el botón Parar, representado por las dos barras verticales de “pause”. Después de parar la animación se puede continuar volviendo a pulsar sobre Reproducir.

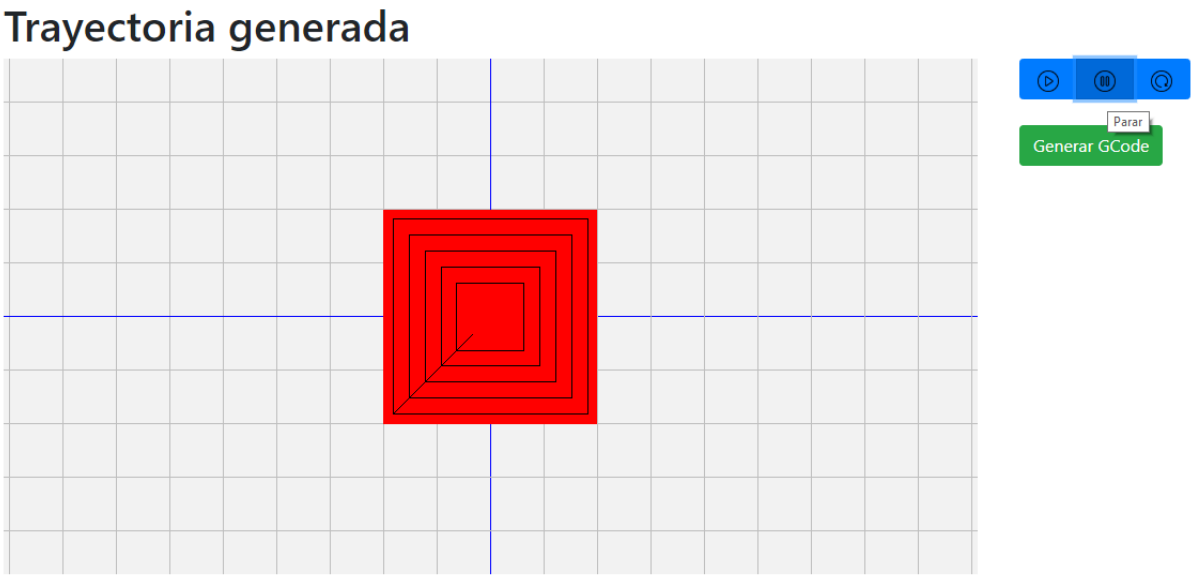

Figura 82. Pausar animación de trayectoria

Una vez finalizada la animación, se puede poner a cero con el botón Reiniciar, representado por la flecha circular de “reset”, para así poder reproducirla de nuevo.

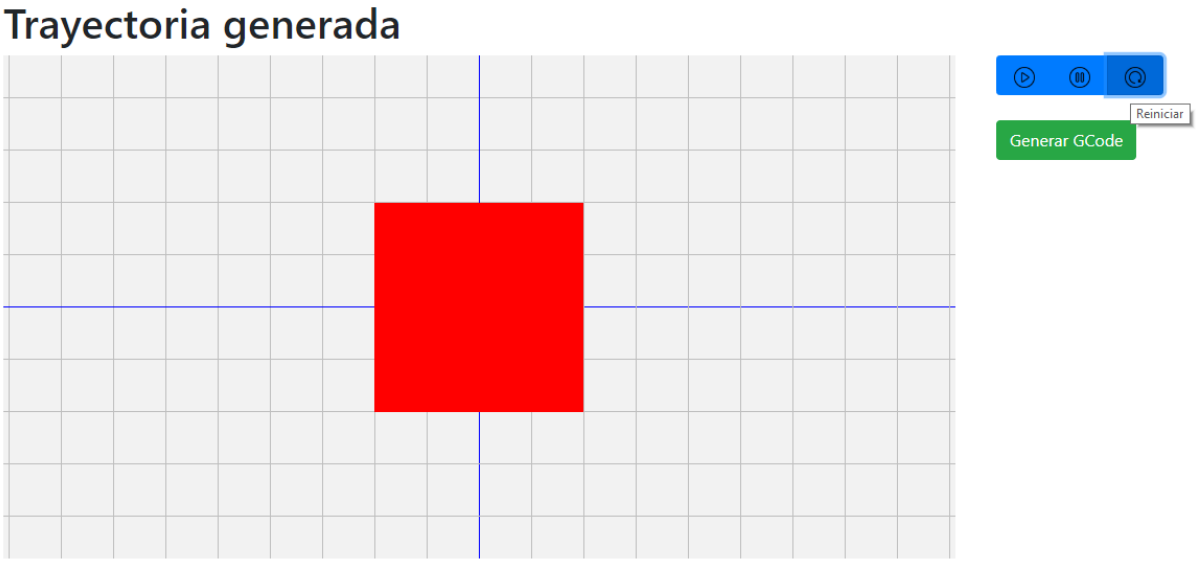

Figura 83. Reiniciar animación de trayectoria

C.9. Generar GCode

Una vez esté generada ya la trayectoria (no es necesario haber reproducido la animación), pulsando sobre el botón Generar GCode nos descargará un archivo `dibujo.gco` el cual es el que necesita la fresadora para poder realizar el trabajo.

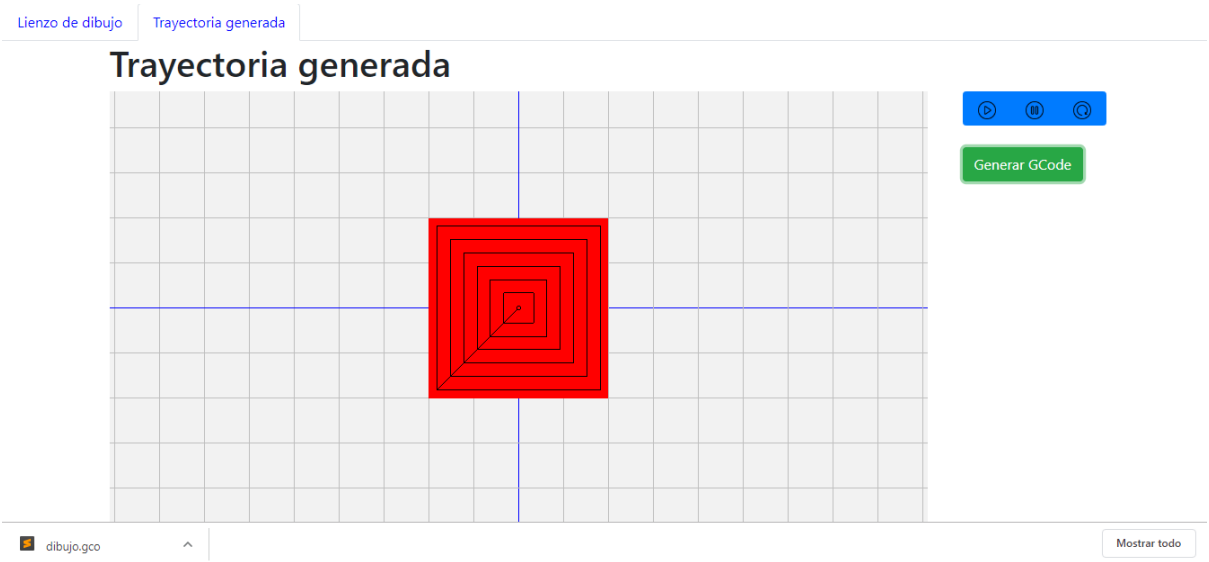

Figura 84. Generar GCode

C.10. Reiniciar aplicación

Si en cualquier momento se desea reiniciar la aplicación sólo es necesario pulsar F5 en el teclado del ordenador o en la flecha circular que se encuentra en el navegador en la parte superior, a la izquierda de la barra de dirección.

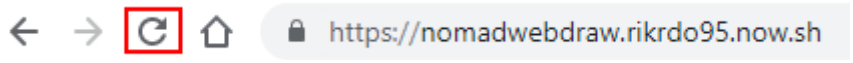

Figura 85. Reiniciar aplicación

Apéndice D

Manual de mantenimiento

Este manual está pensado para aquellos desarrolladores que deseen ampliar o modificar el código fuente de la aplicación. En él se explican un poco los conocimientos necesarios para hacerlo, así como la organización general del código para así aclarar dónde se encuentra cada función, por si se desease modificarlas.

D.1. Manual para desarrolladores

Para el desarrollo es necesario tener conocimientos de JavaScript (en realidad es Typescript, pero es muy similar), HTML y CSS. Conviene tener nociones básicas de Angular, aunque si lo que se va a modificar es un componente ya creado es menos necesario. Es más necesario si se va a crear un componente nuevo, utilizar un servicio o utilizar directivas en el código HTML.

Aparte de esto, hay que tener configurado correctamente el entorno de desarrollo, como se ha explicado en el APÉNDICE A.

D.2. Organización del código

La organización del código está bastante estructurada. Al tratarse de Angular, se divide en componentes. El primero de ellos, AppComponent, es el padre de los demás. En él simplemente está la representación de las pestañas (que representarían cada uno de los otros componentes).

En el componente LienzoComponent está implementada toda la parte de dibujo, así como los parámetros de fresado. Se ve fácilmente cuál es su alcance puesto que es todo aquello que se encuentra en la primera pestaña.

Por último, está el componente TrayectoriaComponent, representado por la segunda pestaña. Es el componente en el que se calculan las trayectorias, así como la representación de estas y su animación.

Por último, está el servicio, usado para comunicar datos entre componentes. Concretamente sirve para que LienzoComponent le pase el dibujo (un array con las figuras), y los parámetros de fresado a TrayectoriaComponent. También comunica a AppComponent y LienzoComponent para que este último le indique que seleccione la pestaña de Trayectoria cuando se cambia de componente.

D.3. Modificaciones del código

Aquí detallaré dónde se encuentran las distintas funcionalidades, por si se desease modificar alguna, o dónde se pueden añadir aquellas nuevas funcionalidades, como las explicadas en el Capítulo 7 en el apartado de trabajo futuro.

En AppComponent se encuentran básicamente los demás componentes. Si se desease añadir algún componente (como la representación 3D), habría que añadir aquí el selector. Si se desease eliminar algún componente de los existentes en la aplicación, también sería aquí.

En LienzoComponent está todo lo relacionado con el dibujo y las opciones de fresado. Desde crear los propios dibujos hasta la validación de los campos, pasando por la modificación de los dibujos. Si se quisieran añadir nuevas formas, más opciones de fresado, más transformaciones a los dibujos (como deformarlos por ejes) o añadir controles de ratón al lienzo de dibujo, sería aquí.

En TrayectoriaComponent se encuentran todas las funciones que calculan las trayectorias de la fresadora, así como las que la animan o las que las transforman en GCode. Si se quisieran añadir nuevos tipos de fresado habría que crear la función que los calcula aquí. También es el sitio donde se tendrían que añadir los comandos de GCode para que lo añadiera al código final si por ejemplo se quisieran añadir opciones para una cortadora de plasma o una grabadora láser. De querer ampliar los controles de la animación de la trayectoria, también sería el componente en el que se haría.

En DataService están aquellas funciones para compartir datos entre componentes. Si se añadiesen atributos que tuvieran que compartirse entre componentes, habría que declararlos aquí y añadirlos a las funciones existentes o crear nuevas si fuera preciso.

Apéndice E

Contenido del CD-ROM

El CD-ROM entregado al tribunal contiene lo siguiente:

- La carpeta Código contiene el proyecto completo, de nombre Nomad WebDraw, con el código fuente. También se añade el mismo código fuente, pero comprimido en un zip por si fuera de utilidad.
- La carpeta Diagramas con las figuras de los diagramas incluidos en la memoria, para facilitar su visualización si en la memoria fuese complicado de ver.
- La carpeta Manuales, en la que se incluyen el manual de instalación del entorno para desarrolladores, el manual de despliegue, el manual de usuario y el manual de mantenimiento.
- La presente memoria, en formato PDF, llamada Memoria.pdf.

Apéndice F

Diccionario de datos y términos

Altura de seguridad Altura a la que se eleva la herramienta por seguridad cuando tiene que moverse mientras no está realizando un trabajo porque la velocidad es mucho mayor que la velocidad de fresado. Se expresa en milímetros.

CNC Control Numérico Computerizado. Referente a aquellas máquinas que utilizan un sistema de automatización de máquinas herramienta que son operadas mediante comandos programados en un medio de almacenamiento, en comparación con el mando manual mediante volantes o palancas, por ejemplo.

Código G / GCode / G-code Lenguaje de programación usado para controlar máquinas de control numérico computerizado (CNC) como fresadoras, cortadoras láser o impresoras 3D.

Framework Entorno de desarrollo con una estructura conceptual y tecnológica de asistencia que puede servir de base para la organización y desarrollo de software

HH Horas/hombre. Las horas que una persona dedica a una tarea

Iteración Cada uno de los ciclos de desarrollo del proyecto.

Perfilado Acción de repasar con la broca el contorno de la figura dibujada.

Profundidad de pasada Profundidad que se incrementa para realizar cada una de las pasadas del fresado. Se expresa en milímetros.

Profundidad total Profundidad total a la que va a llegar el fresado después de realizar varias pasadas. Se expresa en milímetros.

Offset Distancia a la que se desean desplazar los lados de una figura para así conseguir una nueva figura en la que todos sus lados están a esa distancia de la original, provocando una ampliación (o reducción, si es negativo) de la figura.

Software CAM Computer-Aided Manufacturing. Es el software intermedio a los planos 2D y 3D y el lenguaje de programación de las máquinas-herramienta que ejecutan los diseños.

SPA Single Page Application. Páginas web consistentes en un sólo HTML, en las que es trabajo computacional recae en el cliente.

TFG Trabajo de fin de grado

Vaciado Acción de crear un “hueco” en el material con el contorno de la figura dibujada.

Velocidad de entrada Velocidad a la que penetra la fresadora en el material. Se expresa en milímetros por minuto.

Velocidad de fresado Velocidad a la que se mueve la fresadora mientras está realizando un trabajo. Se expresa en milímetros por minuto.

Wireframe Es una representación esquemática de las figuras, mostrando la silueta de los triángulos que forman las figuras.