UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de un simulador gráfico de la Máquina Enigma

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Alexandre Olau Moreno Francés

Tutor: Xavier Molero

Curso 2020-2021

Resum

La màquina Enigma és, possiblement, el dispositiu criptogràfic més conegut del món. El seu impacte en el desenvolupament de la II Guerra Mundial és àmpliament conegut, així com la participació d'una de les figures més importants de la història de la informàtica, Alan Turing, en el procés de desxifrar el seu codi. Aquest treball tracta sobre el disseny i la implementació d'un simulador d'aquesta màquina criptogràfica, alhora que detalla el seu funcionament, el context històric en què es va utilitzar i altres detalls rellevants. El desenvolupament del propi simulador s'ha fet en llenguatge informàtic Java, que és ràpid, flexible i ha sigut utilitzat en diverses assignatures durant els estudis d'aquesta carrera. Una vegada finalitzat el projecte, esperem que el simulador, capaç d'emular diversos models d'aquesta màquina, siga d'utilitat a l'hora de divulgar informació sobre els molts camps que abasta.

Paraules clau: Enigma, simulador, desenvolupament, diseny, historia, Alan Turing, II guerra mundial, criptografia, museu

Resumen

La máquina Enigma es, posiblemente, el dispositivo criptográfico más conocido del mundo. Su impacto en el desarrollo de la II Guerra Mundial es ampliamente conocido, así como la participación de una de las figuras más importantes de la historia de la informática, Alan Turing, en el proceso de descifrar su código. Este trabajo trata sobre el diseño y la implementación de un simulador de esta máquina criptográfica, a la vez que detalla su funcionamiento, el contexto histórico en el que se utilizó y otros detalles relevantes. El desarrollo del propio simulador se ha hecho en lenguaje informático Java, que es rápido, flexible y ha sido utilizado en varias asignaturas durante los estudios de esta carrera. Una vez finalizado el proyecto, esperamos que el simulador, capaz de emular varios modelos de esta máquina, sea de utilidad a la hora de divulgar información sobre los muchos campos que abarca.

Palabras clave: Enigma, simulador, desarrollo, diseño, historia, Alan Turing, II guerra mundial, criptografia, museo

Abstract

The Enigma machine is arguably the world's best known cryptographic device. Its impact on the development of World War II is widely known, as well as the participation of one of the most important figures in the history of computer science, Alan Turing, in the process of breaking its code. This paper deals with the design and implementation of a simulator of this cryptographic machine, while detailing its operation, the historical context around which it was used and other relevant details. The development of the simulator itself will be conducted in the Java computer language, which is fast, flexible and has been used in several courses during the studies of this degree. Once the project is completed, we hope that the simulator, capable of emulating various models of the machine, will be of value in disseminating information about the many fields it covers.

Key words: Enigma, simulator, development, design, history, Alan Turing, World War II, cryptography, museum

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Las Enigma son una familia de máquinas criptográficas utilizadas en Alemania a mediados del siglo XX. Su uso está íntimamente ligado al desarrollo de la II Guerra Mundial, al mundo de la criptografía, el criptoanálisis y a los orígenes de la computación.

En este trabajo se desarrollará un simulador moderno de esta máquina, a la vez que se estudiará su historia para asegurar la mayor similitud posible con la original.

1.1 Motivación

Como se ha comentado, la máquina Enigma forma parte de la historia de la criptografía, de la computación y de la II Guerra Mundial, todas ellas áreas de mi interés personal. Un simulador gráfico moderno y de uso intuitivo facilitará la divulgación de información sobre estos campos y, en lo personal, me ha permitido aprender sobre ellos más de lo que jamás habría hecho sin enfrascarme en este trabajo.

1.2 Objetivos

El objetivo principal de este trabajo es, evidentemente, el desarrollo de un simulador gráfico de la máquina Enigma y, dentro de este, la exactitud histórica: queremos un simulador que demuestre cómo haría uso de la máquina un usuario habitual en diferentes contextos reales.

Un objetivo secundario es que este simulador pueda ser utilizado como herramienta de divulgación en clases o charlas sobre cualquiera de los temas relacionados con la máquina Enigma, con la finalidad de mostrar su funcionamiento al público. Un ejemplo serían las exposiciones del Museo de Informática de la Universitat Politècnica de València.

1.3 Estructura de la memoria

Este trabajo está compuesto por seis capítulos; damos a continuación una breve descripción del contenido de cada uno de ellos:

1. Introducción: Donde se explica el motivo por el que la máquina Enigma es objeto de interés y la motivación y objetivos tras este trabajo.
2. Historia de la criptografía y la máquina Enigma: En este capítulo se expone una breve introducción al mundo de la criptografía, así como un resumen de la historia de la máquina Enigma.
3. Funcionamiento de la máquina Enigma: En este capítulo se muestra su funcionamiento, sus principios, componentes y defectos.
4. Modelos de máquina Enigma: Profundiza en los distintos modelos de Enigma y sus diferencias.
5. Diseño e implementación: Donde se expone el objetivo principal de este proyecto, nuestros razonamientos y las decisiones adoptadas a lo largo de su desarrollo.
6. Conclusiones: Capítulo final que recoge las conclusiones y sintetiza lo expuesto a lo largo del trabajo, a la vez que describe posibles mejoras que podrían añadirse al simulador en un futuro.

También se incluirá el código fuente del simulador en un apéndice, así como un análisis y explicación de su funcionamiento.

1.4 Uso de la bibliografía

Gran parte de la información recogida en este trabajo proviene de los artículos de *Cryptologia*, una publicación científica especializada en la criptografía y su historia que lleva publicándose desde 1977. Sus múltiples artículos sobre la máquina Enigma conforman una gran parte del conocimiento que se tiene a día de hoy sobre el dispositivo.

Otra fuente importante es la sección sobre la máquina Enigma de la página web del Cypto Museum,[1] un museo virtual situado en Holanda con una amplia colección de artículos criptográficos. Sus fuentes están añadidas a la bibliografía, pero el propio trabajo sintetizador del museo se reconoce en esta sección.

Figura 1.1: Logo del Crypto Museum: representa un rotor de la máquina Enigma.

[1] Pagina web del Crypto Museum: https://www.cryptomuseum.com/index.htm

CAPÍTULO 2
Criptografía y la historia de la máquina Enigma

Desde tiempos remotos siempre ha habido interés por mantener en secreto cierta información o determinados mensajes, lo que llevó a la creación de la criptografía, el arte de escribir con clave secreta o de modo enigmático. Este capítulo es una introducción a los orígenes de este arte, de cómo estos llevaron a la creación de la máquina Enigma y de sus predecesores, la evolución de sus múltiples modelos a lo largo de su ciclo de vida y las máquinas que se sucedieron, así como su impacto en la historia de nuestra nación.

2.1 Introducción a la criptografía

Los primeros indicios del uso de mensajes encriptados se han encontrado en las campañas militares de civilizaciones antiguas al tratar de evitar que, en el caso de que un mensajero fuese capturado, la información que portaba fuese revelada al enemigo. Existen todo tipo de métodos primitivos de criptografía, tanto mecánicos, como las escítalas del siglo V a.C., como no mecánicos, como el cifrado de César, una sustitución de letras por sus análogas de tres posiciones más adelante en el abecedario, así conocido por haber sido usado supuestamente por Julio César en sus campañas militares.

Figura 2.1: Una escítala.

Es importante definir los objetivos exactos de la criptografía: la confidencialidad, o la garantía de que únicamente las personas autorizadas puedan acceder a la información cifrada, y la integridad, o la garantía de que los mensajes encriptados mantendrán toda la información sin pérdidas. En el contexto contemporáneo, donde la criptografía se lleva a cabo a través de sistemas automatizados e informáticos, han aparecido nuevos objetivos como el de vinculación o autenticación, pero estos van más allá del ámbito de este trabajo.

2.2 Orígenes de Enigma

La máquina Enigma fue inventada en 1918 por el ingeniero eléctrico alemán Arthur Scherbius y llegó al mercado por primera vez en 1923. Esta primera versión, a diferencia de las que veremos a lo largo del trabajo, daba el resultado del cifrado escribiéndolo sobre papel como una máquina de escribir convencional. Estas máquinas eran conocidas como *Die Handelsmaschine* (la máquina comercial).

Figura 2.2: Arthur Scherbius, inventor de la máquina Enigma.

Sin embargo, este diseño demostró ser extremadamente caro de producir y propenso a fallos, lo que llevó a Scherbius a desarrollar una variante que mostraría el resultado del cifrado a través de una serie de lámparas marcadas con letras, que se iluminarían marcando la letra cifrada correspondiente. Esta revisión dio lugar al modelo Enigma A, con el primer uso de la denominación Enigma en uno de los modelos, que también era conocido como *Gluhlampenmaschine* (Máquina de lámpara incandescente) por las propias luces que empleaba.

Esta versión fue rápidamente sustituida por el modelo B, que a su vez no tardó en ser también reemplazado por el C, que sería el estándar de esta primera etapa de la vida de la máquina Enigma, y, aunque estas revisiones trajeron consigo ciertas mejoras de uso, todas seguían el mismo principio de actuación.[1]

[1] Tanto el principio de actuación como las diferencias entre modelos serán tratadas en los próximos capítulos.

En 1926 tuvo lugar el último gran cambio de diseño de las máquinas Enigma disponibles para el gran público con la aparición del modelo D, que adoptaría el teclado QWERTZ, el estándar alemán, y que simplificaría el acceso a los rotores para facilitar su manipulación. El modelo K, que salió al mercado el año siguiente y acentuó todavía más su facilidad de uso, se convertiría en el modelo comercial definitivo, aunque su funcionamiento en cuanto a encriptación es idéntico al modelo D.

2.3 Evolución de Enigma durante la II Guerra Mundial

A continuación centraremos nuestra atención en los modelos de la máquina Enigma que se desarrollaron durante la guerra y que fueron usadas principal, pero no únicamente, por las tropas alemanas.

La primera máquina Enigma militar fue una variante de la Enigma D, un prototipo de 1927 cuyas particularidades veremos más adelante. Esta variante se conoce como la *Reichwehr* Enigma D (*Reichwehr* se traduce como "guardia imperial", que es la denominación con la que se hizo referencia a la totalidad de las fuerzas armadas alemanas entre 1919 y 1935, por lo que su traducción sería la "Enigma D de las fuerzas armadas").

Este prototipo culminaría en 1932 en la Enigma I. A diferencia de otros modelos, donde la letra es efectivamente una letra, aquí hace referencia al numeral romano I y, por lo tanto, este modelo sería la Enigma "Uno". El estado alemán compró gran cantidad de estas máquinas en su preparación para la guerra, que serían utilizadas por las tropas de la *Wehrmacht* y la *Luftwaffe* (el ejército de tierra y las fuerzas aéreas).

Figura 2.3: Una foto del uso de una máquina Enigma M3 en un submarino U-boat.

Para la *Kriegsmarine*, la marina alemana, se desarrollaron los nuevos modelos de la serie M, que eran compatibles con la Enigma I pero con más opciones para su utilización conjunta con otras Enigma M. Esta serie culminó en 1942 con la Enigma M4, el modelo más seguro que se utilizó durante la guerra, pero de uso exclusivo entre los submarinos alemanes U-boat.

A lo largo de la guerra se desarrollaron otras variantes de la máquina Enigma para ser utilizadas con ciertos objetivos particulares, como la Enigma japonesa que veremos más adelante al examinar las diferencias entre modelos.

2.4 Máquinas Enigma en España

Durante la Guerra Civil española, el bando sublevado recibió ayuda de la Alemania nazi de Hitler y de la Italia fascista de Mussolini. Aunque la mayor parte de este apoyo se dio en forma de armas, municiones, bloqueo de suministros a la República desde el mediterráneo e incluso con el envío de tropas, el bando sublevado también recibió ayuda criptográfica por parte de los alemanes cuando más la necesitaba.

Figura 2.4: Foto de la máquina Enigma expuesta en el museo de la Academia de Artillería en Segovia. Se trata de un modelo K, funcionalmente idéntico al D.

Al principio de la guerra, los generales franquistas tuvieron problemas de comunicación por encontrarse llevando a cabo sus operaciones militares en puntos muy alejados de la península. Los nazis solucionaron este problema suministrando a sus altos mandos diez unidades de máquinas Enigma D, lo que les permitió realizar transmisiones abiertas por radio que el bando republicano nunca tuvo la oportunidad de descifrar. Pese a que a estas alturas ya existían modelos más avanzados, los alemanes eligieron suministrar el modelo comercial por falta de confianza en Franco, con el fin de proteger sus modelos más avanzados de los espías ingleses y soviéticos que vigilaban la península durante la guerra. El bando *nacional* quedó tan satisfecho que pidió una decena adicional de máquinas a los alemanes, aunque recibieron todavía más unidades de los italianos para que la marina pudiera coordinarse con mayor precisión y seguridad.

Se calcula que han sobrevivido alrededor de 30 máquinas Enigma españolas, algunas de las cuales se encuentran expuestas en museos militares a lo largo del país, como el ejemplar del Museo Histórico Militar en Valencia o el que se ve en la figura 2.4.

CAPÍTULO 3
Funcionamiento de la máquina Enigma

En este capítulo describiremos el mecanismo y los principios de actuación de la máquina Enigma. Dado que no existe una máquina única, sino diferentes modelos con sus propias particularidades, analizaremos el comportamiento general de la Enigma cubriendo todas sus posibilidades.

A grandes rasgos, el cifrado Enigma consiste en un circuito eléctrico: unos interruptores, cada uno de los cuales representa una letra, conectan con unos rotores (habitualmente tres), y estos a su vez conectan con un reflector, que devuelve el circuito a los rotores y, por último, a una lámpara, que corresponde a la letra final después del cifrado. Estos elementos se pueden apreciar en la figura 3.1.

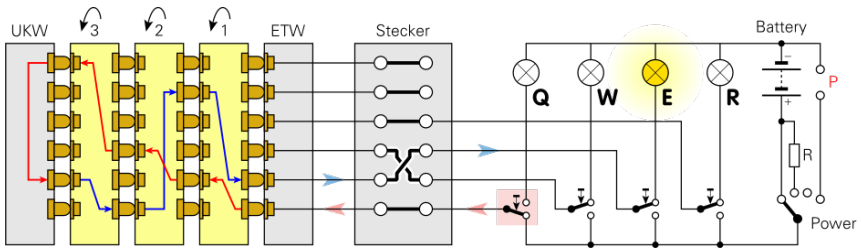

Figura 3.1: Diagrama simple del circuito de las máquinas Enigma. Cortesía del Crypto Museum.

3.1 Entrada

La entrada de un carácter a la máquina se hace pulsando una tecla del teclado. Los diferentes modelos de Enigma han tenido diversas configuraciones de teclado, pero las más comunes son el orden alfabético y el QWERTZ alemán habitual.

Pulsar una tecla activa el mecanismo de giro, que veremos en detalle a continuación, y cierra el circuito que envía la señal eléctrica a través de la máquina para cifrar la entrada.

3.2 Giro de los rotores

Los rotores giran al pulsar una tecla. El primer rotor se mueve con cada pulsación del teclado, mientras que los que hay a continuación son empujados por una muesca en el rotor que les precede. Habitualmente los rotores tienen una única muesca y, por tanto, mueven el rotor siguiente una posición por cada vuelta completa. Sin embargo, este no es siempre el caso y hay modelos de Enigma cuyos rotores tienen hasta 17 muescas.

El giro de los rotores ocurre inmediatamente después de pulsar una tecla y antes de su cifrado. Por lo tanto, si tenemos tres rotores en posición "AAA" y pulsamos cualquier tecla, estos avanzarán a la posición "AAB" antes de cifrarla.

Figura 3.2: Fotografía de los rotores IV y VII de los modelos M.
Se pueden apreciar las dos muescas en el rotor VII. Cortesía del Crypto Museum.

3.2.1. Anomalía del doble paso

La forma exacta de funcionamiento del mecanismo de giro provoca que el rotor central avance dos veces seguidas cuando el tercer rotor también avanza. Es lo que se conoce como *anomalía del doble paso*, ya que un movimiento de los rotores también se conoce como paso.

Esto se debe al mecanismo de giro basado en palancas y muescas de las máquinas, cada vez que se pulsa una tecla unas palancas tratan de empujar los rotores. La primera palanca siempre hace avanzar a su rotor, pero las demás normalmente no alcanzan sus respectivos rotores. Las muescas en los rotores sirven para acercar estas palancas al siguiente rotor en la fila, permitiendo así que estas lo empujen.

Sin embargo, al liberarse de la muesca, la palanca también produce un paso en el rotor en el que estaba alojada. Con la segunda palanca alojada en el primer rotor esto no es perceptible, ya que este debe moverse con cada pulsación del teclado, pero cuando la tercera palanca queda alojada en el segundo rotor, después de que este avance a su posición muescada y pueda accionar el tercer rotor en la siguiente activación, producirá un paso adicional en la segunda rueda.

Las máquinas Enigma *Zählwerk* y las del modelo G tienen un mecanismo de giro diferente basado en engranajes, por lo que no presentan esta anomalía.

3.3 *Plugboard*

Las máquinas Enigma del ejército alemán tenían este mecanismo especial que consistía en un panel con entradas etiquetadas con las letras del abecedario. Conectar dos entradas con un cable provocaba que sus letras se intercambiasen en el circuito en dos ocasiones: o bien tras pulsar una de las teclas conectadas, antes de que esta pasase al cifrado por los rotores, o tras salir de este, justo antes de iluminar una de las lámparas. Hacer uso de este elemento de cifrado aumenta el numero de posibles combinaciones en millones.

Figura 3.3: Fotografía del panel de cables de un modelo militar I. Cortesía del Crypto Museum.

3.4 Cifrado con rotores y reflector

Los rotores son la pieza de la máquina Enigma donde tiene lugar la sustitución de letras. Un rotor es una rueda con 26 conexiones en cada uno de sus laterales, cada una de las cuales representa a una letra, que están interconectadas internamente.

Como se observa en la figura 3.3, por ejemplo, en el primero de los rotores es fácil ver cómo la entrada de B está conectada con la salida a la altura de C. Estas conexiones entre letras se representan mediante una cadena de caracteres que muestran la letra por la que se sustituirá la original según su posición en el abecedario si el rotor está en su posición original.

Fijándonos en ese primer rotor del ejemplo, vemos que sería FCEDBA (A se convierte en F, B se convierte en C, C en E, D no cambia, etc.).

Al girar los rotores, cambian las conexiones, lo que modifica el resultado del siguiente cifrado.

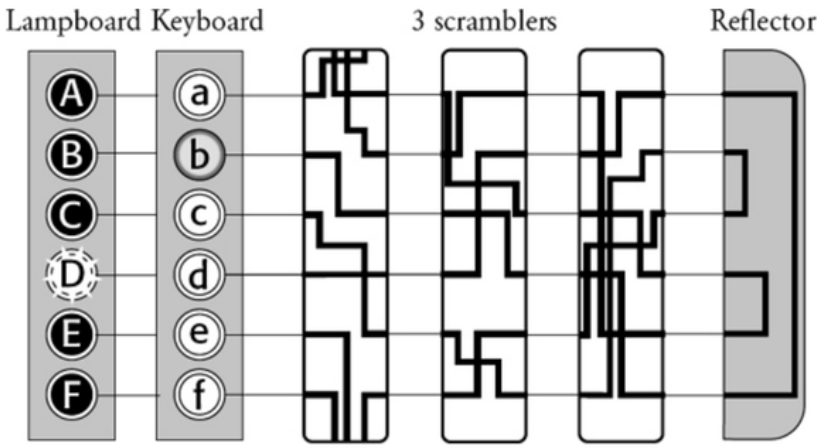

Figura 3.4: Diagrama del circuito de una máquina Enigma con tres rotores.

El reflector, como también se ve en la imagen, es un rotor con conexiones a un solo lado conectadas en pareja. Este mecanismo "refleja" la señal de vuelta a los rotores para una segunda vuelta de cifrado en sentido contrario (con la señal entrando a los rotores por el lado por donde antes salía). Esta forma de implementar el reflector es lo que da lugar a dos propiedades importantes de la máquina Enigma: la simetría de sus cifrados (para descifrar un mensaje simplemente había que reescribirlo en la máquina con la misma configuración inicial) y su mayor debilidad, la imposibilidad de que una letra se sustituyese por sí misma. Los primeros modelos de Enigma no tenían reflectores y por tanto carecían de estas propiedades. En su lugar tenían una palanca para cambiar entre modo de cifrado y descifrado.

3.5 Salida

Al completar el cifrado de sustitución mediante los rotores y reflector, la señal eléctrica llega hasta la lámpara que se corresponde con la última letra y esta se ilumina. La luz permanecía encendida mientras se mantenía presionada la tecla, dando el tiempo necesario para escribirla. El propio mensaje, tanto cifrado como descifrado, debía ser escrito manualmente por el operador de la máquina.

CAPÍTULO 4
Modelos de máquina Enigma

En este capítulo analizaremos los múltiples modelos de máquina Enigma que existieron a lo largo de la historia, las diferencias e innovaciones que había entre ellos y sus usos particulares.

4.1 *Die Handelsmaschine*

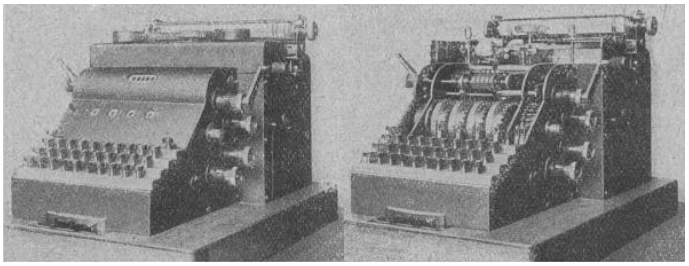

Figura 4.1: Una máquina comercial con el chasis cerrado a la izquierda y abierto a la derecha.

La primera máquina de la marca Enigma (pese a no incluir la palabra Enigma en el nombre o en el identificador del modelo) salió al mercado en 1923. Escribía sobre papel como una máquina de escribir convencional, pero tenía unas dimensiones más altas de lo habitual, como se aprecia en las fotos de la figura 4.2.

Tenía cuatro rotores con diferente número de muescas para giros y ningún reflector, por lo que no ofrecía un cifrado simétrico. En su lugar, un interruptor en la parte frontal permitía cambiar entre modo de cifrado, de descifrado y texto plano. Este último permitía utilizar *Die Handelsmaschine* como una máquina de escribir convencional.

4.1.1. *Die schreibende Enigma*

Este modelo, la "Enigma de escribir", sustituyó a la *Handelsmaschine* en 1926. Tenía un tamaño más reducido por haber situado el circuito en la base de la máquina, y gracias al rediseño mecánico, similar al de una máquina de escribir convencional, permitía escribir más rápido que su modelo predecesor.

Figura 4.2: Una máquina Enigma de escribir, cuya base forma parte de la máquina.

4.2 Enigma A

La primera máquina Enigma que utilizaba lámparas para mostrar el resultado del cifrado salió al mercado en 1924. Este nuevo modelo sirvió para reducir enormemente su coste, lo que potenció su cuota de mercado. Sin embargo, este abaratamiento también tuvo sus inconvenientes, como por ejemplo el avance de los rotores, que era manual mediante un botón. También se bloqueaba el teclado hasta que era pulsado de nuevo y únicamente tenía dos rotores.

Desafortunadamente, ningún modelo de la Enigma A parece haber llegado hasta nuestros días.

4.3 Enigma B

Sucesora de la Enigma A, salió al mercado a finales de 1924.

Incluía un reflector ajustable y realizaba el movimiento de rotores de forma automática. Como se puede apreciar en la figura 4.3, tanto el teclado como las lámparas están ordenados alfabéticamente, en vez de adoptar la configuración habitual de las máquinas de escribir.

También son interesantes las tres ruedas que se pueden observar en la máquina. La de la izquierda es el reflector y las otras dos son los rotores. La Enigma B es el único modelo de dos rotores que ha sobrevivido.

Figura 4.3: Fotografía de una Enigma B suiza.
Nótese la distribución de las teclas y lámparas.

4.4 Enigma C

Tercera máquina de la serie de Enigma con paneles de luces, el modelo C trajo consigo nuevas funcionalidades que facilitaban su uso, como un ajuste de la intensidad de la luz y entradas para fuentes de alimentación externas. Su mecanismo de cifrado consistía en tres rotores y un reflector interno con dos posiciones fijas, razón por la que solo se ven tres ruedas en su carrocería, al igual que en el modelo B.

No han sobrevivido ejemplares de este modelo, pero a diferencia del A, se conservan fotografías como la de la figura.

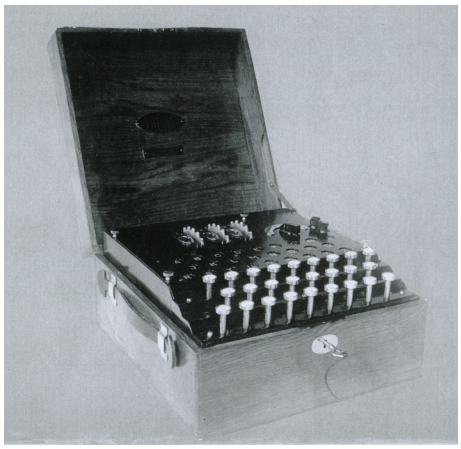

Figura 4.4: Fotografía de una Enigma C tomada por el fabricante original.

4.5 Enigma D

Figura 4.5: Fotografía de una Enigma D dentro de su caja. Nótese la nueva distribución del teclado y lamparas. Cortesía del Crypto Museum.

La Enigma D fue un modelo de suma importancia en la historia de estas máquinas criptográficas. Salió al mercado en 1926 y aportó un gran número de mejoras frente al modelo C. El teclado tenía la configuración QWERTZ habitual en Alemania que se puede observar en la figura 4.5 y contaba con un mecanismo de tres rotores, cuyo orden era modificable, y reflector único pero de posición ajustable. Estas mejoras convirtieron a la Enigma D en la principal máquina Enigma destinada al mercado comercial y la mayoría de las Enigmas posteriores estaban basadas en ella, hasta el punto de ser funcionalmente idénticas. Por su gran importancia y extendido uso, este modelo es uno de los que se implementará en el simulador.

4.5.1. Enigma K

El modelo K es una revisión del modelo D que lo sustituyó en 1927. Tenía alguna diferencia de diseño, pero era funcionalmente idéntica al modelo D y extremadamente difícil de diferenciar a simple vista. Una variante de este modelo, con un cableado diferente en los rotores y reflector, fue el utilizado por los oficiales ferroviarios alemanes y puede verse en la figura 4.6.

4.5.2. *Reishwehr D*

Primer prototipo de máquina Enigma para uso militar, desarrollado entre 1927 y 1929, consistía en una Enigma D con el reflector fijo, pero incluía la primera aparición del *plugboard*, un panel con entradas para cables que permitían "conectar" dos letras para que se intercambiasen entre sí.

Esta máquina desembocó en la Enigma I, que sí tuvo un uso real.

Figura 4.6: Fotografía de una Enigma K sin su caja de transporte. Es prácticamente indistinguible del modelo D. Este modelo concreto fue usado por oficiales ferroviarios. Cortesía del Crypto Museum.

4.5.3. Enigma Z

Figura 4.7: Fotografía de una Enigma Z.

Máquina Enigma numérica, la única capaz de cifrar números, basada en el modelo D. Dispone únicamente de diez teclas y lámparas numeradas del cero al nueve, y tiene el mismo mecanismo que el modelo en el que se basa, con tres rotores y un reflector.

4.6 Enigma I

Máquina Enigma empleada por las tropas de tierra y aire alemanas durante la II Guerra Mundial, introducida en 1932. Al igual que su prototipo, utilizaba tres rotores y un reflector fijo, pero compensaba este inconveniente con la posibilidad de cambiar de reflector, siendo los más comunes los reflectores denominados UKW-B y UKW-C, y suplementando los rotores con dos adicionales entre los que poder elegir. El *plugboard* que estaba presente en este prototipo también incrementaba exponencialmente las posibles combinaciones del modelo, que puede verse en la figura 4.8 y fue el más utilizado por el bando alemán durante la guerra.

Figura 4.8: Fotografía de una Enigma I. Nótese el panel de cables.

Por estas razones, en un principio se consideró la posibilidad de implementar este modelo en nuestro simulador, pero finalmente se optó por no hacerlo, ya que, como veremos a continuación, el modelo M3 incluye todas sus funcionalidades y es perfectamente retrocompatible.

4.7 Serie Enigma M

La serie M fue una serie de máquinas Enigma I adaptadas para la marina alemana, la *Kriegsmarine*. Los primeros modelos solo tenían diferencias estéticas, pero con los años se fueron añadiendo complementos que aumentaron su complejidad.

4.7.1. Enigma M1, M2 y M3

La marina alemana adoptó la máquina Enigma con el modelo M1 en 1934, que fue sustituido por el M2 en 1938 y por el M3 en 1940. Prácticamente no hay diferencias entre ellos.

Las únicas diferencias con el modelo I eran el orden de las entradas en el *plugboard*, que siguen el orden alfabético en estos modelos mientras que en el anterior están ordenadas como el teclado de una máquina de escribir, y los rotores indican su posición mediante letras en lugar de números.

En 1939 la marina añadió a la serie M tres nuevos rotores, aumentando el total a ocho. Estos nuevos rotores tenían dos muescas de giro en lugar de una, que era lo habitual en los modelos derivados de la Enigma D. Cabe destacar que la marina siempre usaba al menos uno de estos rotores adicionales en sus comunicaciones internas, lo que reducía el número real efectivo de posibles combinaciones.

Ya que esta máquina ofrece más funcionalidades que la Enigma I, con total compatibilidad con esta simplemente eligiendo no usar los tres nuevos rotores, es una de las máquinas que se ha implementado en nuestro simulador.

Figura 4.9: Fotografía de una caja donde se almacenan los rotores de las Enigma M que no estaban en uso. Aquí se ven cinco, más los tres que habría en la máquina, suman ocho.

4.7.2. Enigma M4

La máquina Enigma M4 fue introducida a finales de la II Guerra Mundial, en 1942, y representa la versión definitiva de la rama militar de la familia Enigma. Al igual que el resto de la serie M, fue utilizada exclusivamente por la marina, específicamente por los submarinos U-boat y una puede verse en la figura 4.10.

Es la primera máquina Enigma de cuatro rotores. El nuevo rotor se denominaba rotor extra (*Zusatzwalze*) y se encuentra entre el último rotor y el reflector; es ajustable pero no gira durante el uso de la máquina, a diferencia de los tres rotores habituales. El rotor extra es un modelo diferente y no es compatible con los ocho rotores normales. Solo dos rotores se usaron como rotores extra, denominados Beta y Gamma. Finalmente, la Enigma M4 contaba con un nuevo par de reflectores, UKW-b y UKW-c (nótese la minúscula), que al combinarse con los rotores Beta y Gamma respectivamente en su primera posición, resultaban ser funcionalmente idénticos a los reflectores UKW-B y UKW-C de los modelos anteriores, asegurando así su compatibilidad.

Por la complejidad de esta Enigma y por su importancia durante la guerra, este es uno de los modelos seleccionados para nuestro simulador.

Figura 4.10: Fotografía de la parte superior de una Enigma M4. Nótese la rueda negra, que es la del rotor extra. Cortesía del Crypto Museum.

4.8 Enigma G

Originalmente conocida como *Zählwerk Enigma*, toma su nombre del pequeño contador situado al lado izquierdo de los rotores, que mide la longitud de los mensajes cifrados y descifrados por la máquina, tal como puede verse en la figura 4.11. Pese a ser la función que dio su nombre al modelo, en realidad se trataba de su innovación más insignificante. Sin embargo, se comercializaron muy pocas unidades y no cobró popularidad hasta el lanzamiento de la Enigma G, una revisión desarrollada en 1931 que reducía sus medidas y peso para convertirla en portátil.

Figura 4.11: Fotografía de la parte superior izquierda de una Enigma G, donde se puede ver el contador de longitud y la palanca para bloquear los engranajes de giro. Cortesía del Crypto Museum.

Al tratarse de una Enigma comercial, su funcionamiento y diseño están basados en el popular modelo D. El cableado de los rotores y el reflector fueron idénticos a los de la Enigma D en todos los modelos previos al denominado G, incluso en la primera remesa de estos. La Enigma G contó con dos pequeñas revisiones que alteraron el cableado de los rotores, pero mantenían idénticas todas las demás funciones.

La innovación más importante de este modelo fue la completa reconstrucción del mecanismo de giro de los rotores, que pasó a basarse en engranajes. Esto permitió añadir una entrada a la que se le podía introducir una palanca incluida en la máquina para avanzar y retroceder el mecanismo de giro sin tener que hacer ajustes manuales, lo que facilitaba la corrección de errores e incluso añadir pasos manuales a la clave de cifrado. Esta palanca puede verse en la figura 4.12. También permitía el bloqueo total de los engranajes mediante un interruptor, visible en la figura 4.11, que detenía los rotores e impedía su giro durante la operación de la máquina.

Además, el número de muescas en los rotores fue aumentado enormemente: en concreto, los rotores pasaron de tener una sola muesca a tener 17, 15 y 11 respectivamente, todos ellos números primos y no divisibles por 26, que es el número de letras cifrables por Enigma. Con esto se lograba que los rotores girasen siguiendo un patrón irregular y, en consecuencia, mucho más difícil de calcular.

Figura 4.12: Fotografía de la parte superior derecha de una Enigma G. Nótese la manivela para avanzar y retroceder el mecanismo de giro. Cortesía del Crypto Museum.

Pese a no tratarse de una máquina Enigma militar y, por tanto, no poseer un *plugboard*, la Enigma G trajo consigo importantes innovaciones que aumentaban la complejidad de sus cifrados, siendo utilizada por diversas instituciones del gobierno nazi, entre ellas su servició secreto, el *Abwehr*.

4.9 Enigma T

Figura 4.13: Fotografía de una Enigma T.

Modelo especializado en comunicaciones entre oficiales alemanes y japoneses utilizado a partir de 1943, también conocido por su nombre en clave, "Tirpitz". Los japoneses querían emplear los modelos militares alemanes dotados con *plugboards*, pero los alemanes se negaron a compartir sus Enigma más avanzadas y llegaron al acuerdo de desarrollar este modelo especializado que combina aspectos de las series M y G.

Estas máquinas, al igual que las Enigma G, están dotadas de un único reflector con posición ajustable (se trata de la rueda negra visible en la figura 4.13) y suplementado por rotores con múltiples muescas para añadir complejidad al mecanismo de giro. Sin embargo, a diferencia del modelo G, todos los rotores tenían exactamente cinco muescas, como se puede ver en la figura 4.14.

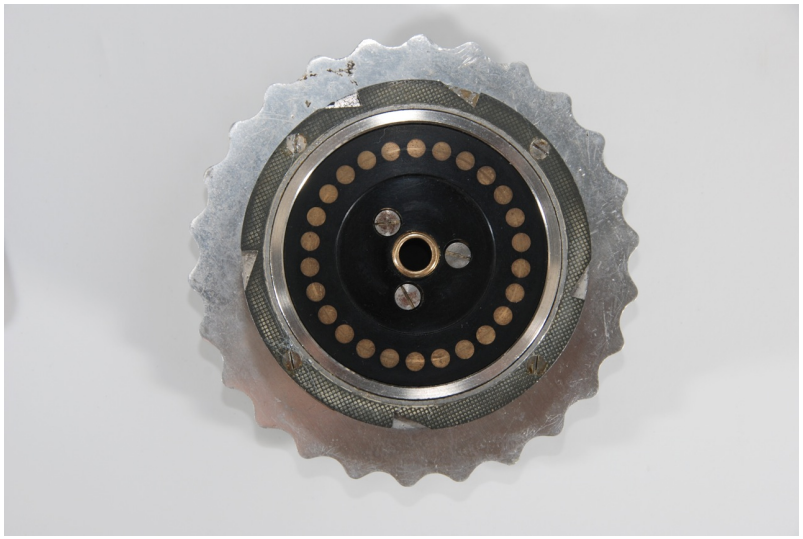

Figura 4.14: Fotografía del reverso de un rotor de la Enigma T, donde se pueden ver a la perfección las cinco muescas.

Aunque son menos muescas que sus predecesoras, la Enigma T seguía utilizando un número primo no divisible por 26, con lo que el giro seguía siendo irregular y más complejo que el del modelo D. Esto se compensaba con la aportación de la rama M: ocho rotores con diferente cableado entre los que optar, una capacidad de elección que antes solo estaba en manos de la marina. Cabe destacar que, aunque el cableado es diferente en cada uno de los rotores, solo hay cuatro combinaciones diferentes de muescas, con dos rotores utilizando cada una.

4.10 Tablas de cableado por modelo

Esta sección incluirá una serie de tablas que describirán el cableado de los rotores de los modelos anteriores. Que un modelo no aparezca en la lista significa que, a fecha de publicación, no hay constancia de su cableado. El capítulo 3 describe la nomenclatura usada en estas tablas.

4.10.1. Cableado Enigma D y K

Rotor	Cableado	Muescas
I	LPGSZMHAEOQKVXRFYBUTNICJDW	Y
II	SLVGBTFXJQOHEWIRZYAMKPCNDU	E
III	CJGDPSHKTURAWZXFMYNQOBVLIE	N
Reflector	IMETCGFRAYSQBZXWLHKDVUPOJN	

Tabla 4.1: Cableado de la máquina Enigma D, que es idéntico en el modelo K.

4.10.2. Cableado Enigma I y serie M

Rotor	Cableado	Muescas
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Q
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	E
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	V
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	J
V	VZBRGITYUPSDNHLXAWMJQOFECK	Z
VI	JPGVOUMFYQBENHZRDKASXLICTW	Z, M
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT	Z, M
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV	Z, M
UKW-B	YRUHQSLDPXNGOKMIEBFZCWVJAT	
UKW-C	FVPJIAOYEDRZXWGCTKUQSBNMHL	

Tabla 4.2: Cableado de la Enigma M1, M2 y M3. Idéntico en la Enigma I eliminando los rotores VI, VII y VIII.

4.10.3. Cableado Enigma M4

Rotor	Cableado	Muescas
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ	Q
II	AJDKSIRUXBLHWTMCQGZNPYFVOE	E
III	BDFHJLCPRTXVZNYEIWGAKMUSQO	V
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB	J
V	VZBRGITYUPSDNHLXAWMJQOFECK	Z
VI	JPGVOUMFYQBENHZRDKASXLICTW	Z, M
VII	NZJHGRCXMYSWBOUFAIVLPEKQDT	Z, M
VIII	FKQHTLXOCBJSPDZRAMEWNIUYGV	Z, M
UKW-b	ENKQAUYWJICOPBLMDXZVFTHRGS	
UKW-c	RDOBJNTKVEHMLFCWZAXGYIPSUQ	
Beta	LEYJVCNIXWPBQMDRTAKZGFUHOS	
Gamma	FSOKANUERHMBTIYCWLQPZXVGJD	

Tabla 4.3: Cableado de la Enigma M4. Los rotores son iguales a los otros modelos M, pero los reflectores y rotores extra son nuevos.

4.10.4. Cableado Enigma G

Rotor	Cableado	Muescas
I	LPGSZMHAEOQKVXRFYBUTNICJDW	SUVWZABCEFGIKLOPQ
II	SLVGBTFXJQOHEWIRZYAMKPCNDU	STVYZACDFGHKMNQ
III	CJGDPSHKTURAWZXFMYNQOBVLIE	UWXAEFHKMNR
Reflector	IMETCGFRAYSQBZXWLHKDVUPOJN	

Tabla 4.4: Cableado de la máquina Enigma G, que es idéntico en el modelo D excepto por el gran número de muescas.

4.10.5. Cableado Enigma T

Rotor	Cableado	Muescas
I	KPTYUELOCVGRFQDANJMBSWHZXI	W, Z, E, K, Q
II	UPHZLWEQMTDJXCAKSOIGVBYFNR	W, Z, F, L, R
III	QUDLYRFEKONVZAXWHMGPJBSICT	W, Z, E, K, Q
IV	CIWTBKXNRESPFLYDAGVHQUOJZM	W, Z, F, L, R
V	UAXGISNJBVERDYLFZWTPCKOHMQ	Y, C, F, K, R
VI	XFUZGALVHCNYSEWQTDMRBKPIOJ	X, E, I, M, Q
VII	BJVFTXPLNAYOZIKWGDQERUCHSM	Y, C, F, K, R
VIII	YMTPNZHWKODAJXELUQVGCBISFR	X, E, I, M, Q
UKW	GEKPBTAUMOCNILJDXZYFHWVQSR	

Tabla 4.5: Cableado de la máquina Enigma T.

4.10.6. Cableado Enigma Z

Rotor	Cableado	Muescas
I	6418270359	9
II	5841097632	9
III	3581620794	9
Reflector	5079183642	

Tabla 4.6: Cableado de la máquina Enigma Z.

CAPÍTULO 5
Diseño e Implementación

A continuación describimos los objetivos del simulador, así como la metodología empleada y las decisiones adoptadas a lo largo del proceso de su desarrollo.

5.1 Objetivos de diseño

Nos hemos propuesto una serie de objetivos a alcanzar durante el desarrollo del simulador con el fin de que nos sirvieran de guía para conseguir un producto final más útil a la tarea de divulgación. Son los que se exponen a continuación:

5.1.1. Precisión y rigor histórico

Queríamos lograr un simulador que emulara el funcionamiento real de estas máquinas hasta el punto de ser totalmente compatible con ellas. Por lo tanto, se ha investigado para localizar sus cableados reales y se ha tratado de mantener todas las funcionalidades posibles.

5.1.2. *Ease of Use*

Las máquinas Enigma originales eran dispositivos relativamente difíciles de operar y habitualmente se utilizaban entre dos usuarios. Aprovechando las ventajas de las nuevas tecnologías, hemos tratado de conseguir un simulador cuyo uso fuera lo más sencillo posible sin perder de vista el resto de objetivos.

5.1.3. Abanico de modelos

Como hemos visto, no existe una máquina Enigma única, sino una serie de diferentes modelos con diferentes cualidades. En el mejor de los casos, habríamos podido implementar todos los modelos, pero no ha sido posible por falta de tiempo e información sobre algunos de los más antiguos. Por lo tanto, hemos tratado de seleccionar los más relevantes.

5.2 Fase de diseño

Decidimos tempranamente que, para acomodar los diferentes modelos de la máquina Enigma y sus particularidades, se desarrollaría el simulador en dos capas. La primera sería una capa interfaz que gestionaría las opciones del usuario (el modelo, los rotores seleccionados y sus posiciones, etc.) y en la segunda se trataría de una serie de clases estáticas que representaran las máquinas Enigma y sus particularidades (cableado interno de los rotores, posición de las muescas, métodos de cifrado, etc.).

5.2.1. Diseño de la interfaz de usuario

La capa de interfaz es responsable de la entrada y salida de información por y para los usuarios, y, por ende, es la capa cuyo diseño visual es más importante, puesto que debe ser de uso intuitivo y, a la vez, ceñirse a nuestros objetivos de diseño. A continuación analizaremos las diversas partes de la interfaz.

Entrada de caracteres

Las máquinas Enigma utilizaron, a lo largo de su historia, dos formas de teclado para la entrada de texto: la ordenación de las letras por orden alfabético o aplicando la distribución QWERTZ típica alemana. Una alternativa más que se nos presenta por conveniencia moderna sería un teclado en pantalla QWERTY actual.

Dado que a día de hoy se desconoce el cableado de los modelos con entrada en orden alfabético por no haberse localizado ningún ejemplar, hemos descartado esta opción. A su vez, entre QWERTZ y QWERTY, hemos optado por el primero para respetar la autenticidad histórica del simulador.

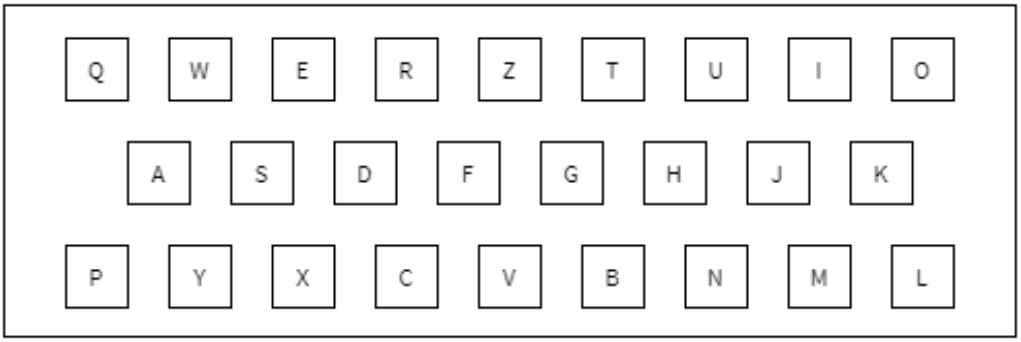

Figura 5.1: Diagrama del teclado de entrada.

La entrada se realizará mediante un teclado en pantalla, tal como muestra la figura. Hacer click en una de estas teclas virtuales iniciará el método de cifrado con los parámetros seleccionados en las demás partes de la interfaz.

Entrada de rotores y selección de modelo

La entrada de información de los rotores es la parte que más varía entre los distintos modelos de Enigma y también donde más información se debe introducir. Por lo tanto, utilizaremos un panel diferente para cada modelo que se adapte a sus componentes, pero siempre empleando una estructura similar.

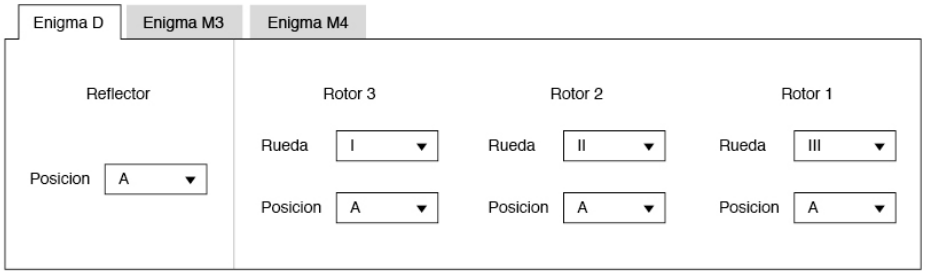

Figura 5.2: Diagrama de la entrada de la información de rotores del modelo D. El modelo T utilizaría un menú idéntico.

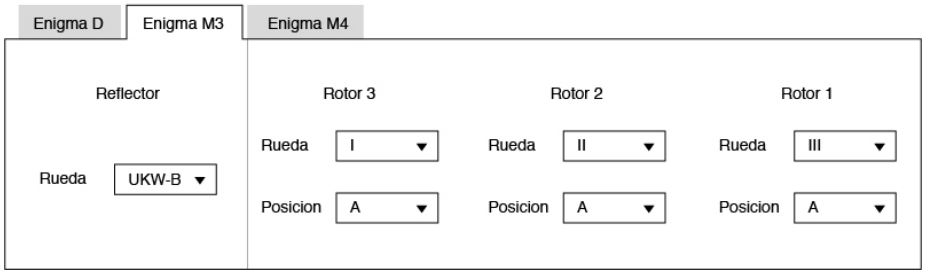

Figura 5.3: Diagrama de la entrada de la información de rotores del modelo M3.

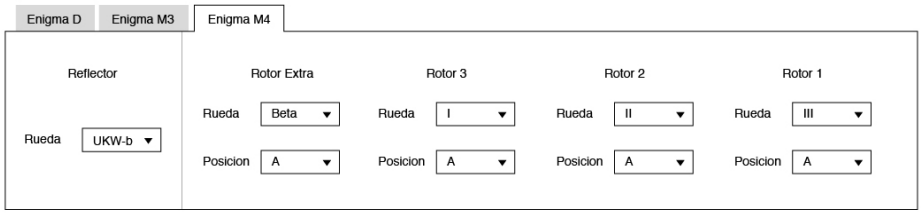

Figura 5.4: Diagrama de la entrada de la información de rotores del modelo M4.

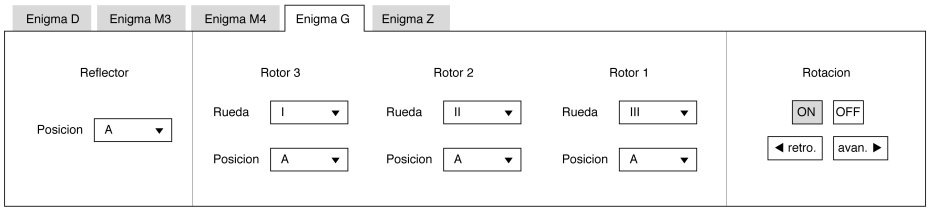

Figura 5.5: Diagrama de la entrada de la información de rotores del modelo G, que incluye botones adicionales para sus funciones únicas.

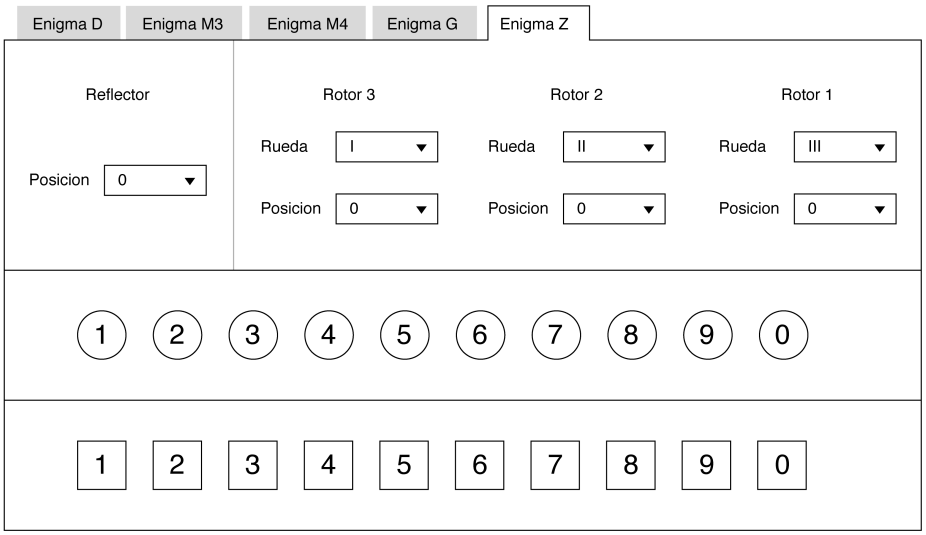

Figura 5.6: Diagrama completo para el modelo Z, que requerirá su propio teclado y set de lámparas.

Panel de cables

El panel de cables estará solo disponible para los modelos militares, ya que eran los únicos equipados con esa función. Para la interfaz, tendríamos dos opciones: el orden alfabético de la serie M o el orden QWERTZ de los modelos I. Optamos por la segunda opción para mantener el orden con las demás estructuras del simulador, pero, si dispusiéramos de suficiente tiempo extra, también sería una opción que cada modelo tuviese su propia estructura de *plugboard*.

Imitaremos el panel con una serie de botones con un círculo asociado debajo. Al presionar alguno de los botones, su círculo adoptará un color. Al presionar otro botón, su círculo adoptará el mismo color, mostrando que están enlazados. Si se presiona un botón de una pareja, ambos pierden su color y la pareja se deshace. Pulsar un botón dos veces seguidas cancela su emparejamiento. Por comodidad, también se ha añadido un botón para deshacer rápidamente todos los emparejamientos.

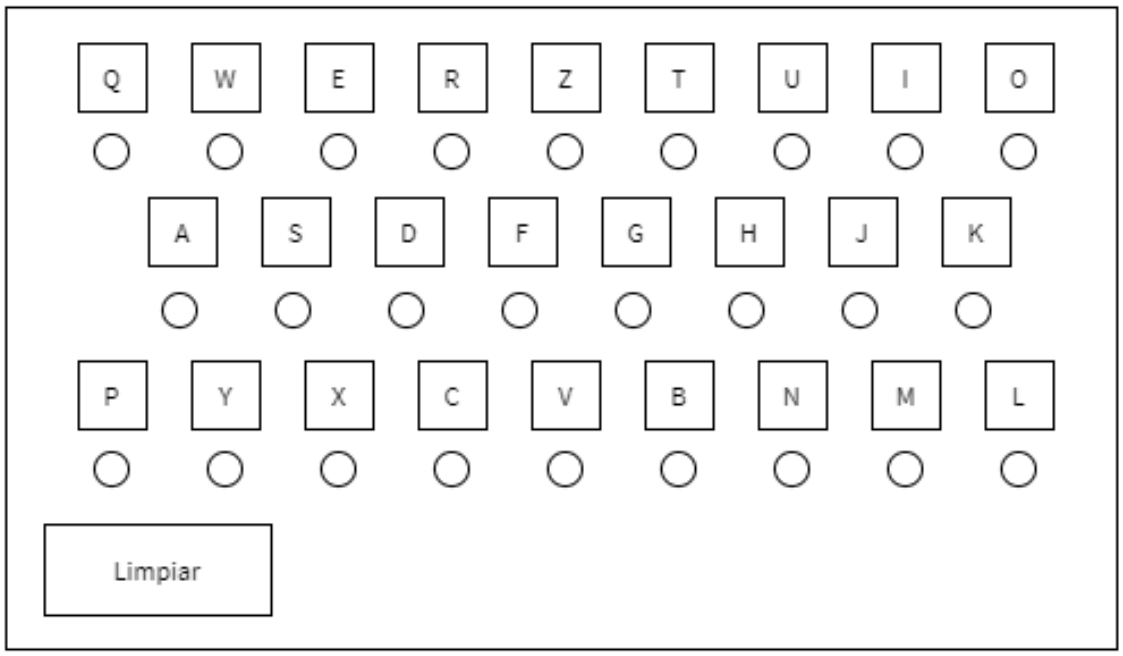

Figura 5.7: Diagrama del *plugboard* del simulador

Salida

Para mostrar los resultados del cifrado, emplearemos un panel con círculos etiquetados que cambiarán de color para "iluminar" el resultado, tal como lo harían las lámparas de los modelos originales. El orden de estas lámparas es igual al del teclado en la gran mayoría de modelos y, por lo tanto, seguiremos un razonamiento similar a la hora de diseñarlo.

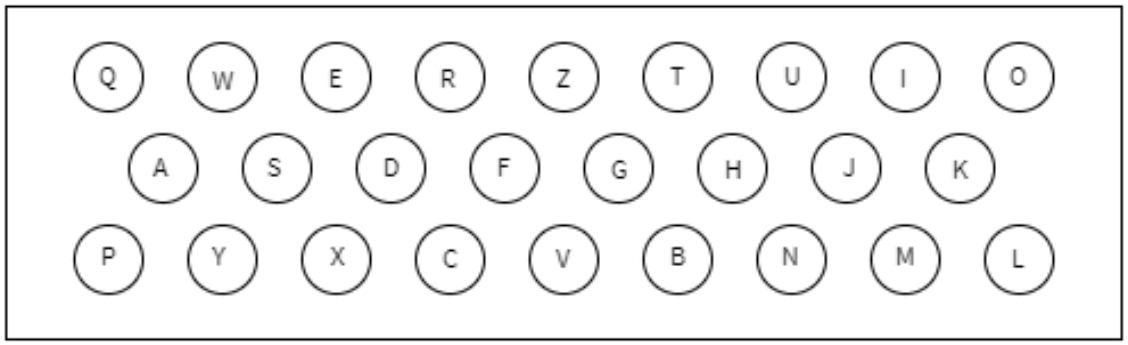

Figura 5.8: Diagrama del panel de lámparas del simulador.

La lámpara correspondiente con la letra resultante del cifrado se iluminará y se mantendrá así hasta que se presione otra tecla, simulando el comportamiento real de las originales, donde la lámpara se mantenía encendida tanto tiempo como se mantuviese pulsado el botón. No obstante, aquí, eliminaremos la necesidad de mantener el botón presionado.

5.2.2. Diseño de las clases Enigma

Las clases Enigma de nuestro proyecto serán clases estáticas que contendrán la información del cableado y del circuito de los modelos Enigma originales que replican. Tendrán métodos "cifrar" que serán llamados desde la capa de interfaz con los datos seleccionados en esta. Los modelos originales de Enigma descendían entre ellos como en un árbol genealógico y, por lo tanto, construyen a partir de algo ya existente. Emplearemos herencia de Java para aplicar esta filosofía de diseño a nuestro simulador. Estas clases contendrán información sobre los rotores de su modelo, su cableado y la posición de las muescas. Tendrán un método "cifrar" que recibirá como parámetros los rotores seleccionados y sus posiciones, así como el carácter a cifrar, y devolverán el resultado de este cifrado, así como un método getter para que la capa de interfaz pueda acceder a la posición de las muescas y efectuar el giro de los rotores.

5.3 Fase de implementación

En esta sección expondremos el proceso de implementación del simulador. Explicaremos los procedimientos adoptados, pero no profundizaremos en la forma exacta de la implementación del código, puesto que lo haremos en el anexo.

5.3.1. Herramientas utilizadas

Figura 5.9: Logos de herramientas usadas en el desarrollo del simulador.
De izquierda a derecha: Java, NetBeans y SceneBuilder.

Optamos por utilizar herramientas de desarrollo vistas a lo largo de los estudios cursados en la carrera de Ingeniería Informática en la ETSInf.

Decidimos usar Java 8 con Netbeans como IDE junto a SceneBuilder para el desarrollo, como en la asignatura Interfaces Persona Computador (IPC). Java 8 ha sido usado en múltiples asignaturas a lo largo de la carrera e incluye su propia implementación de JavaFX para manejar la interfaz gráfica de usuario sin requerir instalaciones adicionales.

Otra razón para usar Java es el uso de herencia para implementar las clases Enigma, como mencionamos en la sección anterior.

Ya que hemos escogido la versión 8 de Java, también hemos utilizado la versión 8.5.0 de SceneBuilder para construir la interfaz, puesto que es la versión recomendada por los desarrolladores para trabajar con esta versión del lenguaje.

Otro motivo para usar Java es el uso de herencia para implementar las clases Enigma, como mencionamos en la sección anterior.

Ya que escogimos la versión 8 de Java, también usaremos la versión 8.5.0 de SceneBuilder para hacer la interfaz, ya que es la versión recomendada por los desarrolladores para trabajar con esta versión del lenguaje.

5.3.2. Primeros pasos

Antes de proceder a la creación del simulador, optamos por hacer una toma de contacto con la implementación de un cifrado de sustitución de caracteres. La clase de caracteres de Java, char, es una clase numérica que interpreta su contenido como el equivalente en Unicode. Por tanto, es posible aplicar un offset sumando o restando directamente a la variable que contiene el carácter a cifrar. Sin embargo, finalmente no hemos utilizado esta propiedad sino que hemos tomado el índice de los caracteres por su posición en el abecedario.

5.3.3. Interfaz de usuario

El siguiente paso ha sido crear la interfaz de usuario en SceneBuilder tratando de asemejarnos lo máximo posible a los diseños mostrados anteriormente. Se asigna una ID única a las lámparas, conectores del panel y menús de rotores y posición, pues su contenido es relevante durante la ejecución y debe poder ser llamado por el programa sin importar qué tecla active el cifrado. En el caso de las teclas y los botones del panel, se les asigna un método para ejecutar sus tareas, pero no requieren de IDs, pues es posible consultar la letra que etiqueta el botón que activa sus eventos.

La barra de ventanas para elegir modelo de Enigma también tiene una ID y evento asociado. La ID se usa para consultar qué modelo de Enigma hay seleccionado cuando se intenta ejecutar el cifrado, mientras que el método controla los cambios en las partes de la interfaz que no pertenecen a la propia ventana, como el *plugboard*.

Hemos posibilitado la ocultación del panel de cables durante el uso del simulador del mismo modo que se podía esconder dentro de la caja de los originales.

5.3.4. Clase Main y controlador de la interfaz

Clase Main

Esta clase simplemente se dedicará a llamar a la escena, abriendo el archivo FXML. Su implementación es extremadamente simple.

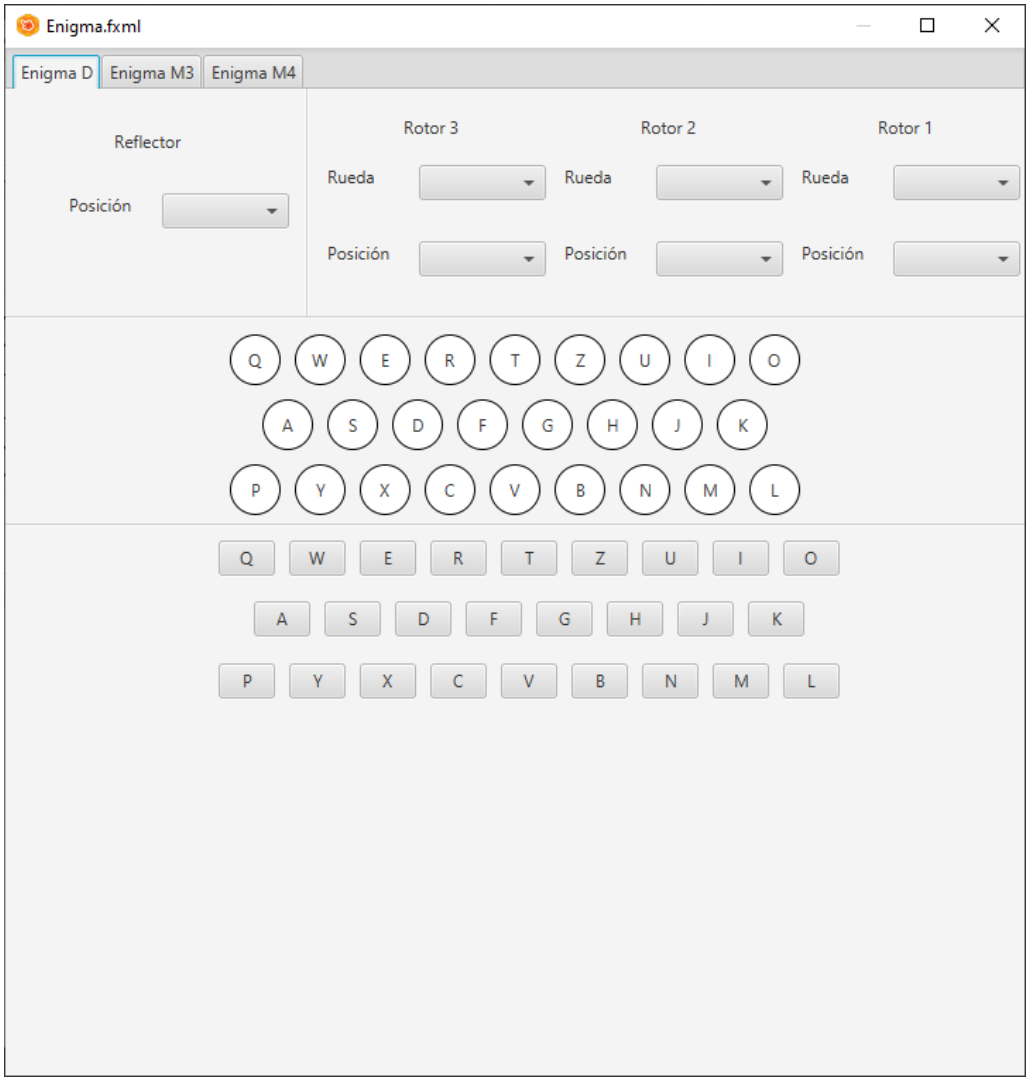

Figura 5.10: Maqueta del simulador en la ventana del modelo D, sin panel de cables.

Clase controlador

Esta será la clase que contendrá la mayor parte de los métodos necesarios para la funcionalidad del simulador y probablemente será la de mayor envergadura en cuanto a código. El primer paso en su elaboración es ejecutar la orden *Make Controller* de NetBeans, que genera una plantilla de clase con la información ya incorporada en nuestra interfaz.

El siguiente paso es la implementación de muchos de los métodos vacíos que esta plantilla nos ofrece, como un "Initialize" que dé sus valores a los campos de selección de los rotores y sus posiciones, a la vez que sus valores por defecto. El resultado de iniciar las maquetas de las figuras 5.10 y 5.11 se puede ver en las figuras 5.12, 5.13 y 5.14.

También contendrá los métodos que llamarán a las clases Enigma para efectuar los cifrados, controles de la interfaz (como mostrar u ocultar el panel de cables, según el modelo seleccionado, o encender y apagar las lámparas), así como gestionar las conexiones del *plugboard*.

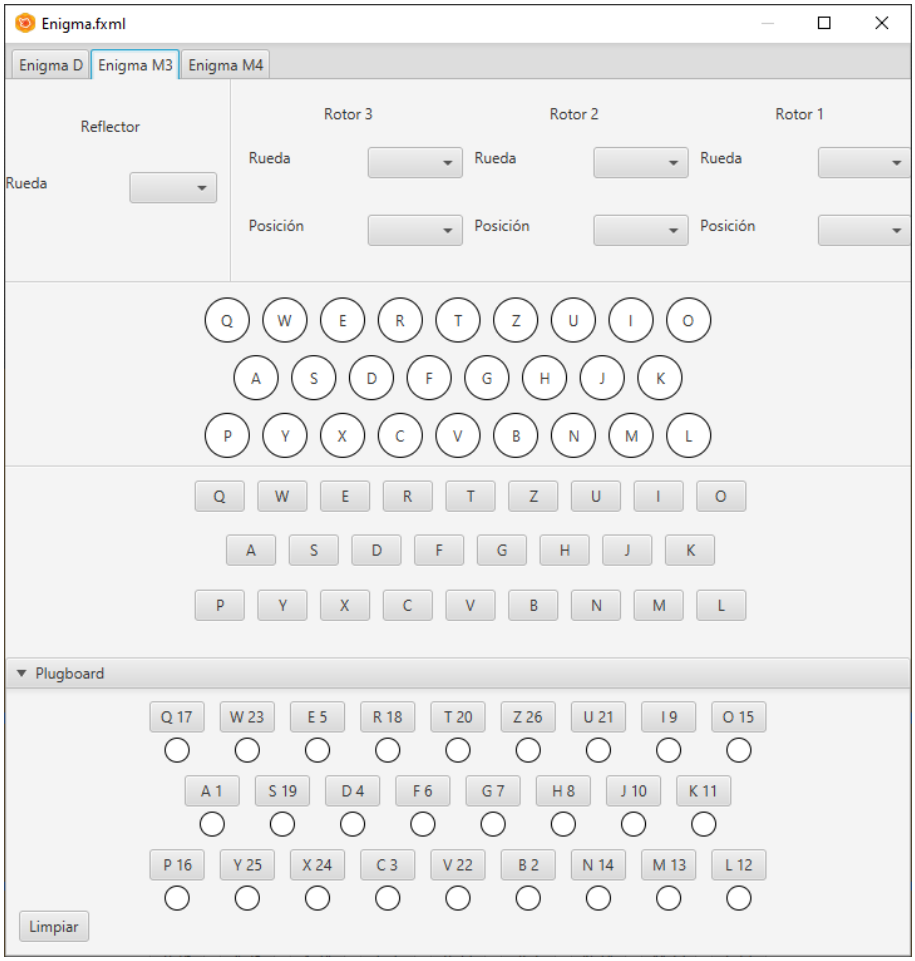

Figura 5.11: Maqueta del simulador en la ventana del modelo M3, que sí tiene panel de cables.

La mayoría de estos métodos son relativamente simples y cumplen un solo cometido, siendo la excepción el método cuyo objetivo es recoger la información y ejecutar el cifrado. Este método se activa al pulsar uno de los botones del teclado, almacena el carácter que este contiene, comprueba el modelo Enigma seleccionado basándose en la pestaña actual y ejecuta la sustitución del panel de cables antes y después del cifrado.

La selección de modelo se llevará a cabo mediante submétodos que serán llamados por este, cada uno de ellos recogerá la información pertinente a su modelo y calculará la siguiente posición de los rotores (ya que esta debe efectuarse antes del cifrado, como ya hemos comentado). Una vez llevado a cabo todo el proceso, se invocará el método “Cifrar” del modelo en cuestión y se iluminará la lámpara apropiada.

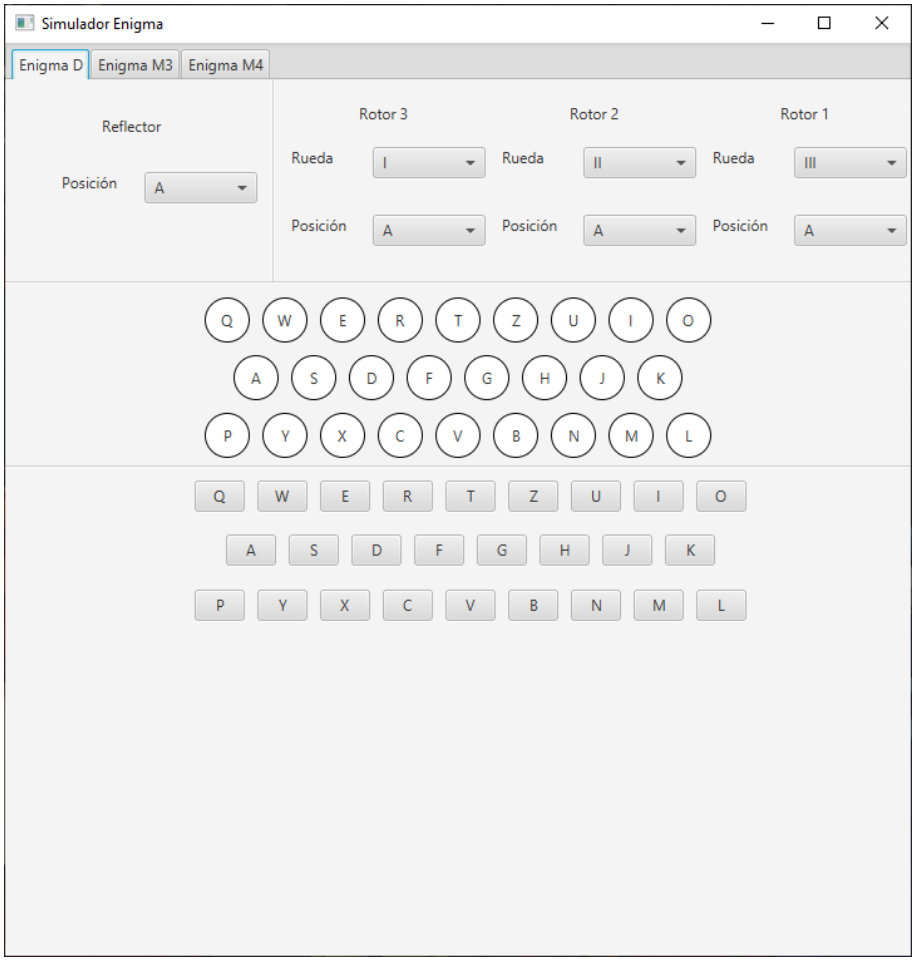

Figura 5.12: Interfaz de usuario ya inicializada mostrando el simulador de Enigma D.

Clase auxiliar Helper

Con el objetivo reducir el volumen de las clases Enigma y el controlador, optamos por desarrollar una clase auxiliar que contuviera métodos y constantes necesarios para el buen funcionamiento del simulador, pero que pueden estar almacenados en otra clase para reducir el tamaño de estas.

Para apoyar al controlador, Helper se encargará del cálculo del giro de los rotores mediante una serie de guardas basadas en la posición actual del rotor, y la posición muescada almacenada en la clase Enigma pasará los números romanos de los rotores y números de serie de reflectores a números naturales para ser usados como parámetros.

También contendrá métodos para apoyar a las clases Enigma; algunos de sustitución entre caracteres y su índice en el abecedario y viceversa, útiles en el algoritmo de cifrado, así como un método que garantizará que los números con los que trabajemos siempre se encuentren entre 0 y 26, es decir, dentro de los límites del abecedario, y en caso de rebasarlo volver a situarnos dentro de este margen.

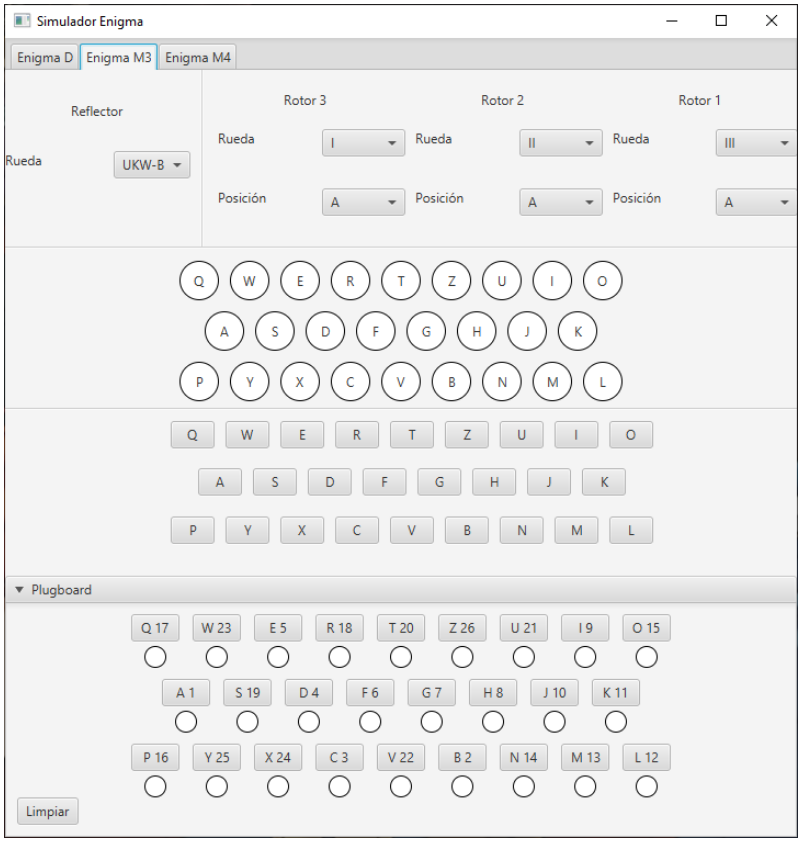

Figura 5.13: Interfaz de usuario ya inicializada mostrando el simulador de Enigma M3, diferente respecto al D por el panel de cables y la selección de "rueda" en el reflector, en lugar de "posición".

5.3.5. Clases Enigma y sus métodos

Estas clases contendrán los datos propios de cada modelo de la máquina Enigma, así como métodos que, tomando como parámetros un carácter a cifrar y rotores a usar en cierto orden y sus posiciones, devuelvan el resultado que ese carácter de entrada o input habría producido en una máquina Enigma real con esa configuración.

Cada posición representa un rotor determinado y, como mencionamos anteriormente, los Strings la letra por la que se sustituye la letra del abecedario de ese índice en su posición inicial. Las posiciones muescadas en una matriz de caracteres, donde cada fila representa un rotor, en la mayoría de los casos los rotores solo tienen una muesca, pero este modelo facilita la implementación de los rotores con más de una sin añadir apenas coste computacional. Para que el controlador pueda acceder a las muescas, incluiremos un método getter que, dado el número de un rotor, devuelva sus muescas. Las muescas están representados por el carácter que se muestra en la rueda del rotor cuando se hace contacto con la muesca.

El evento principal de estas clases es el método "cifrar". Este método toma como parámetros tres elementos: el carácter a cifrar, un array de números que marcan los rotores seleccionados y su orden y otro array numérico que representa las posiciones de estos rotores y, por tanto, el offset que se debe aplicar al cifrado.

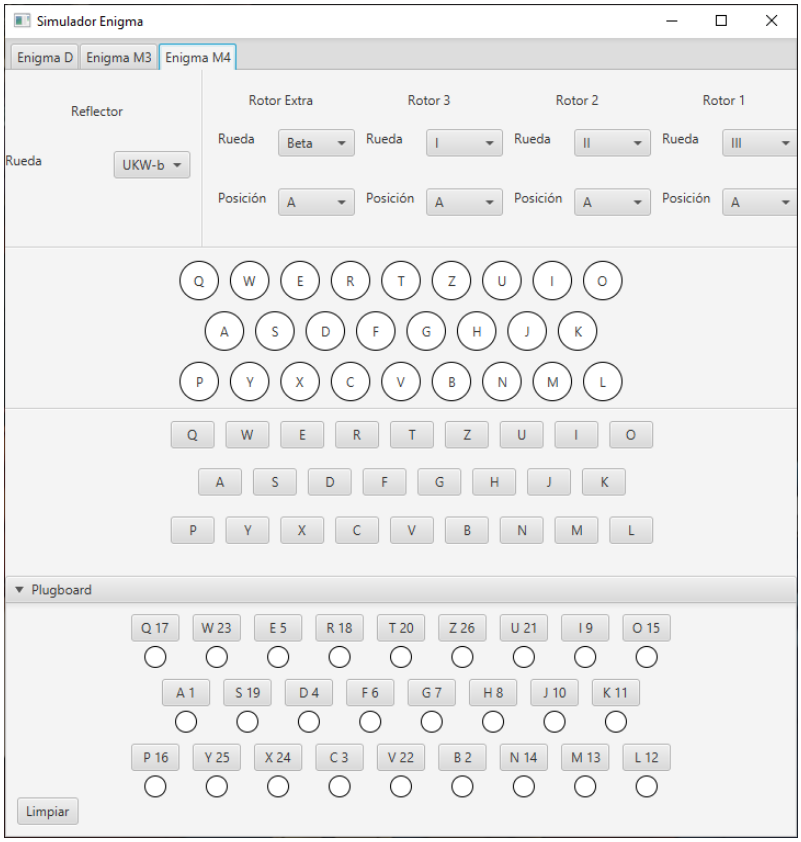

Figura 5.14: Interfaz de usuario ya inicializada mostrando el simulador de Enigma M4, diferente respecto al M3 por la presencia del rotor extra.

El método de cifrado tiene tres partes y se divide en dos submétodos. Uno hace las sustituciones para cada rotor en dirección hacia el reflector y se ejecuta una vez por cada rotor en un bucle, y una vez adicional para el reflector con parámetros ligeramente distintos. El otro método hace la sustitución de un rotor en sentido contrario y se ejecuta dentro de otro bucle que va en sentido contrario al anterior. El resultado final de este último bucle se vuelve a convertir en carácter y es la respuesta final del cifrado de sustitución (sin contar una posible pasada adicional por el *plugboard*).

Para cifrar hacia el reflector, o de derecha a izquierda, tomamos como parámetros el número que representa la letra a sustituir, el String del rotor con el que se cifra y el offset de este. La sustitución consiste en incrementar el input por el offset, pasar el nuevo número a un carácter y obtener el índice de este en el String rotor. Finalmente, se le resta el offset a este índice y ese número es devuelto por el método. Este número representa el índice en el abecedario del resultado del cifrado. Adicionalmente, después de cada operación con el offset y el índice, hacemos uso del método que asegura que el índice esta entre 0 y 26.

El cifrado en sentido contrario, de izquierda a derecha, es extremadamente similar. La diferencia es que convertimos el resultado de la primera operación en el carácter índice, elegimos la letra del rotor y la sustituimos por el índice en el abecedario. Posteriormente se efectúa la resta al igual que en el otro método.

Estos son los elementos en común de todas las clases Enigma. A continuación describiremos las diferencias entre cada una de ellas.

Clase Enigma D

Esta clase, al igual que el modelo de la máquina Enigma que imita, será la base de las demás clases mediante herencia de Java. Por ende, también será la clase Enigma con más contenido y tendrá todos los métodos y todas las constantes mencionadas anteriormente.

Las constantes en cuestión para en modelo representan lo visto en la tabla 4.1.

Clase Enigma M3

Modelo Enigma que representa los modelos militares y marines estándar, con un número mucho mayor de rotores y dos reflectores, algunos de ellos con más de una muesca. Las constantes son muy diferentes de su clase precedente, pero cambia el método cifrar para alterar el cifrado por reflector, ya que esta máquina tenía más de una opción para ello, pero sin posibilidad de alterar su posición. El getter debe ser sobre-escrito para que esta clase devuelva sus muescas y no las del modelo D.

Las constantes de este modelo están basadas en la información de la tabla 4.2.

Clase Enigma M4

La única diferencia entre este modelo y el M3 es el rotor adicional y los diferentes reflectores. Como el rotor adicional no gira, no tiene muescas y, por tanto, la matriz de muescas puede ser heredada de la M3 sin problemas, pero el array de rotores debe ser implementado de nuevo.

Las constantes para este modelo aparecen en la tabla 4.3.

CAPÍTULO 6
Conclusiones

Concluimos este trabajo con un resumen de lo expuesto hasta ahora y un pequeño análisis de futuros objetivos.

6.1 Reflexiones finales

Hemos hablado de la historia de la máquina Enigma, de sus modelos, de su funcionamiento y de su importancia histórica. Hemos visto que se trataba de un dispositivo potente, pero con una serie de debilidades que eventualmente permitieron a los analistas británicos romper su código.

El diseño y desarrollo del simulador han supuesto un gran esfuerzo, puesto que, aunque no se puede considerar ningún apartado como especialmente complejo, la búsqueda de información sobre el funcionamiento y la traducción de estos mecanismos a código informático representaron un ejercicio interesante. También lo ha sido la elaboración de una interfaz de usuario que debería ser intuitiva para los usuarios.

6.2 Posible trabajo futuro

Debido a la falta de tiempo y a otras dificultades como consecuencia de la situación sanitaria de este curso, no se ha explotado todo el potencial de este simulador. Dedicándole más tiempo, habríamos añadido más modelos, concretamente los modelos G, T y Z ya vistos, así como otras variantes como la K ferroviaria, e incluso se habría intentado añadir accesorios no tratados en este trabajo, a la vez que habríamos ampliado el capítulo dedicado a la historia de Enigma.

También consideramos que sería útil la creación de una página web que alojara el simulador, posiblemente dentro de la propia web del Museo de Informática.

Sin embargo, tenemos plena confianza en que, en su estado actual, este simulador es capaz de cumplir con sus objetivos como herramienta de divulgación.

Bibliografía

[1] DAVID HAMER. "Actions involved in the 'double stepping' of the middle rotor". En *Cryptologia* 1997, Vol. 20, no. 1. ISSN 0161-1194

[2] DAVID HAMER, GEOFF SULLIVAN y FRODE WEIERUD. "Enigma Variations: An Extended Family of Machines". En *Cryptologia* 1998, Vol. 22, no. 3, pp. 211-229. ISSN 0161-1194

[3] FRIEDRICH L. BAUER. "An error in the history of rotor encryption devices". En *Cryptologia* 1999, Vol. 23, no. 3. ISSN 0161-1194

[4] LOIUS KRUH y CIPHER A. DEAVORUS. "The Commercial Enigma: Beginnings of Machine Cryptography". En *Cryptologia* 2002, Vol. 26, no. 1. ISSN 0161-1194

[5] KARL DE LEEUW. "The Dutch invention of the Rotor Machine, 1915-1923". En *Cryptologia* 2003, Vol. 27, no. 1, pp. 73-94. ISSN 0161-1194

[6] JOSÉ RAMÓN SOLER FUENSANTA, FRANSISCO JAVIER LÓPEZ-BREA ESPIAU y FRODE WEIERUD. "Spanish Enigma: A History of the Enigma in Spain". En *Cryptologia* 2010, Vol. 34, no. 14, pp. 301-328. ISSN 0161-1194

[7] CIPHER DEAVOURS y LOIUS KRUH. *Machine Cryptography and Modern Cryptanalysis*. Artech House, 1985. ISBN: 9780890061619

[8] SIMON SINGH. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Knopf Doubleday Publishing Group, 2001. ISBN 978-0-307-78784-2.

[9] Dr. A. RAY MILLER. *The Cryptographic Mathematics of Enigma*. 3ª Edición, 2002. NSA. Center for Cryptologic History, 1996. ISBN 9781478379270.

[10] TOM PERERA. *The Story of the ENIGMA: History, Technology and Deciphering*. 2ª Edición, 2004. Artifax Books. ISBN 1-890024-06-6

[11] BRIAN J. WINKEL, CIPHER DEAVOURS y LOIUS KRUH. *The German Enigma Cipher Machine: Beginnings, Success, and Ultimate Failure*. Artech House, 2005. ISBN: 9781580539968

[12] MAVIS BATEY. *Dilly: The Man Who Broke Enigma*. Dialogue, 2009. ISBN 978-1-906447-01-4.

[13] MORENO IZQUIERDO, Rafael. El arma secreta de Franco. En: *El País*. Otoño 2008. [consulta: junio 2021]. ISSN 1576-3757. Disponible en: https://elpais.com/diario/2008/10/12/domingo/1223783554_850215.html

[14] *History of the Enigma*. Crypto Museum, ©2012-2019 [consulta: mayo 2021]. Disponible en: https://www.cryptomuseum.com/crypto/enigma/hist.htm

[15] *Working principle of the Enigma*. Crypto Museum, ©2009-2020 [consulta: mayo 2021]. Disponible en: https://www.cryptomuseum.com/crypto/enigma/working.htm

[16] *Enigma wiring*. Crypto Museum, ©2009-2021 [consulta: mayo 2021]. Disponible en: https://www.cryptomuseum.com/crypto/enigma/wiring.htm

[17] *Java Platform Standard Edition 8 Documentation*. Oracle, ©1993-2021 [consulta: mayo 2021]. Disponible en: https://docs.oracle.com/javase/8/docs/

APÉNDICE A
Código fuente del simulador

A.1 FXMain.java

```
1  package enigmafx;
2
3  import java.io.IOException;
4  import javafx.application.Application;
5  import javafx.fxml.FXMLLoader;
6  import javafx.scene.Parent;
7  import javafx.scene.Scene;
8  import javafx.stage.Stage;
9
10 /**
11  * @author Alexandre Olau Moreno Frances
12  */
13 public class FXMain extends Application {
14
15     private final String rutaEscena = "/enigmafx/Enigma.fxml";
16     private final String Titulo = "Simulador Enigma";
17
18     @Override
19     public void start(Stage stage) throws IOException {
20
21         Parent root = FXMLLoader.load(getClass().getResource(rutaEscena
               ));
22
23         Scene scene = new Scene(root);
24
25         stage.setScene(scene);
26         stage.setTitle(Titulo);
27         stage.show();
28     }
29
30     public static void main(String[] args) {
31         launch(args);
32     }
33 }
```

A.2 EnigmaController.java

```
1  /*
2   * To change this license header, choose License Headers in Project
        Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package enigmafx;
7
8  import enigmamachine.*;
9  import auxiliar.helper;
10 import java.net.URL;
11 import java.util.ArrayList;
12 import java.util.Iterator;
13 import java.util.List;
14 import java.util.ResourceBundle;
15 import javafx.collections.FXCollections;
16 import javafx.event.ActionEvent;
17 import javafx.fxml.FXML;
18 import javafx.fxml.Initializable;
19 import javafx.scene.control.Button;
20 import javafx.scene.control.ChoiceBox;
21 import javafx.scene.control.TabPane;
22 import javafx.scene.control.TitledPane;
23 import javafx.scene.input.MouseEvent;
24 import javafx.scene.media.AudioClip;
25 import javafx.scene.paint.Color;
26 import javafx.scene.shape.Circle;
27
28 /**
29  * FXML Controller class
30  *
31  * @author Alexandre Olau Moreno Frances
32  */
33 public class EnigmaController implements Initializable {
34
35     private boolean emparejandoPlugs;
36     private List<String> plugboard;
37     private String plugboardPareja;
38     private Color colorPareja;
39
40     @FXML
41     private ChoiceBox<String> rot1D, rot2D, rot3D, rot1M3, rot2M3,
           rot3M3,
42             rot1M4, rot2M4, rot3M4, rotExtraM4, refM3, refM4;
43
44     @FXML
45     private ChoiceBox<Character> pos1D, pos2D, pos3D, pos1M3, pos2M3,
           pos3M3,
46             pos1M4, pos2M4, pos3M4, posExtraM4, posRefD;
47     @FXML
48     private Circle lampA, lampB, lampC, lampD, lampE, lampF, lampG,
           lampH,
49             lampI, lampJ, lampK, lampL, lampM, lampN, lampO, lampP,
                lampQ,
50             lampR, lampS, lampT, lampU, lampV, lampW, lampX, lampY,
                lampZ;
51     @FXML
```

```
52      private Circle plugA, plugB, plugC, plugD, plugE, plugF, plugG,
            plugH,
53              plugI, plugJ, plugK, plugL, plugM, plugN, plugO, plugP,
                    plugQ,
54              plugR, plugS, plugT, plugU, plugV, plugW, plugX, plugY,
                    plugZ;
55
56      private Circle ultimaLampIluminada;
57
58      @FXML
59      private TabPane enimgaTabs;
60      @FXML
61      private TitledPane plugboardPane;
62
63      public EnigmaController() {
64          this.plugboard = new ArrayList<>();
65          this.ultimaLampIluminada = null;
66          this.emparejandoPlugs = false;
67      }
68
69      /**
70       * Initializes the controller class.
71       *
72       * @param url
73       * @param rb
74       */
75      @Override
76      public void initialize(URL url, ResourceBundle rb) {
77
78          //Enigma D
79          this.rot1D.setItems(FXCollections.observableArrayList("I", "II"
                , "III"));
80          this.rot2D.setItems(FXCollections.observableArrayList("I", "II"
                , "III"));
81          this.rot3D.setItems(FXCollections.observableArrayList("I", "II"
                , "III"));
82          this.pos1D.setItems(FXCollections.observableArrayList('A', 'B',
                'C',
83                  'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
84                  'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
85          this.pos2D.setItems(FXCollections.observableArrayList('A', 'B',
                'C',
86                  'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
87                  'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
88          this.pos3D.setItems(FXCollections.observableArrayList('A', 'B',
                'C',
89                  'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
90                  'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
91          this.posRefD.setItems(FXCollections.observableArrayList('A', 'B
                ', 'C',
92                  'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
```

```
93                  'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
94
95          //Enigma M3
96          this.rot1M3.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
97                  "IV", "V", "VI", "VII", "VIII"));
98          this.rot2M3.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
99                  "IV", "V", "VI", "VII", "VIII"));
100         this.rot3M3.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
101                 "IV", "V", "VI", "VII", "VIII"));
102         this.refM3.setItems(FXCollections.observableArrayList("UKW-B",
                "UKW-C"));
103         this.pos1M3.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
104                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
105                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
106         this.pos2M3.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
107                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
108                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
109         this.pos3M3.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
110                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
111                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
112
113         //Enigma M4
114         this.rot1M4.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
115                 "IV", "V", "VI", "VII", "VIII"));
116         this.rot2M4.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
117                 "IV", "V", "VI", "VII", "VIII"));
118         this.rot3M4.setItems(FXCollections.observableArrayList("I", "II
                ", "III",
119                 "IV", "V", "VI", "VII", "VIII"));
120         this.rotExtraM4.setItems(FXCollections.observableArrayList("
                Beta", "Gamma"));
121         this.refM4.setItems(FXCollections.observableArrayList("UKW-b",
                "UKW-c"));
122         this.pos1M4.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
123                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
124                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
125         this.pos2M4.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
126                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
```

```
127                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
128         this.pos2M4.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
129                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
130                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
131         this.pos3M4.setItems(FXCollections.observableArrayList('A', 'B'
                , 'C',
132                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
133                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
134         this.posExtraM4.setItems(FXCollections.observableArrayList('A',
                'B', 'C',
135                 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
                        'O',
136                 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'))
                        ;
137
138         //Enigma D
139         this.rot1D.setValue("III");
140         this.rot2D.setValue("II");
141         this.rot3D.setValue("I");
142         this.pos1D.setValue('A');
143         this.pos2D.setValue('A');
144         this.pos3D.setValue('A');
145         this.posRefD.setValue('A');
146
147         //Enigma M3
148         this.rot1M3.setValue("III");
149         this.rot2M3.setValue("II");
150         this.rot3M3.setValue("I");
151         this.refM3.setValue("UKW-B");
152         this.pos1M3.setValue('A');
153         this.pos2M3.setValue('A');
154         this.pos3M3.setValue('A');
155
156         //Enigma M4
157         this.rot1M4.setValue("III");
158         this.rot2M4.setValue("II");
159         this.rot3M4.setValue("I");
160         this.rotExtraM4.setValue("Beta");
161         this.refM4.setValue("UKW-b");
162         this.pos1M4.setValue('A');
163         this.pos2M4.setValue('A');
164         this.pos3M4.setValue('A');
165         this.posExtraM4.setValue('A');
166     }
167
168     @FXML
169     private void pulsacionTeclado(ActionEvent event) {
170         playAudio();
171
172         char input = ((Button) event.getSource()).getText().charAt(0);
173         char output;
174
```

```
175         int maquinaEnigma = enimgaTabs.getSelectionModel().
                getSelectedIndex();
176         switch (maquinaEnigma) {
177             case 0:      //EnigmaD
178                 output = cifrarEnigmaD(input);
179                 break;
180             default:     //Enigma M3
181                 input = cifrarPlugboard(input);
182                 output = cifrarEnigmaM3(input);
183                 output = cifrarPlugboard(output);
184                 break;
185             case 2:      //Enigma M4
186                 input = cifrarPlugboard(input);
187                 output = cifrarEnigmaM4(input);
188                 output = cifrarPlugboard(output);
189                 break;
190         }
191         iluminacion(lampSelect(output));
192     }
193
194     @FXML
195     private void plugboardHandler(MouseEvent event) {
196         int indexTabActual = enimgaTabs.getSelectionModel().
                getSelectedIndex();
197         if (indexTabActual != 0) {
198             plugboardPane.setExpanded(true);
199             plugboardPane.setVisible(true);
200         } else {
201             plugboardPane.setExpanded(false);
202             plugboardPane.setVisible(false);
203         }
204     }
205
206     @FXML
207     private void plugboardConnection(ActionEvent event) {
208         String input = ((Button) event.getSource()).getText().substring
                (0, 1);
209
210         if (!plugSelect(input.charAt(0)).getFill().equals(Color.WHITE))
                {
211             if (estaEnchufadoPlugboard(input)) {
212                 deshacerParejaPlugboard(input);
213             } else {
214                 cancelarEmparejamientoPlugboard(input);
215             }
216         } else if (!emparejandoPlugs) {
217             emparejarPlugsParte1(input);
218             emparejandoPlugs = true;
219         } else {
220             emparejarPlugsParte2(input);
221             emparejandoPlugs = false;
222         }
223     }
224
225     @FXML
226     private void limpiarPlugboard(ActionEvent event) {
227         plugboard.clear();
228         emparejandoPlugs = false;
229         for (char i = 'A'; i <= 'Z'; i++) {
```

```
230             plugSelect(i).setFill(Color.WHITE);
231         }
232     }
233
234     private char cifrarEnigmaD(char input) {
235
236         int[] rotor = {helper.rotorNameToNum(rot1D.getValue()),
237             helper.rotorNameToNum(rot2D.getValue()),
238             helper.rotorNameToNum(rot3D.getValue())};
239
240         char[] pos = {pos1D.getValue(), pos2D.getValue(), pos3D.
                getValue(),
241             posRefD.getValue()};
242
243         char[] turnoverRotor1 = EnigmaD.getTurnover(rotor[0]);
244         char[] turnoverRotor2 = EnigmaD.getTurnover(rotor[1]);
245
246         /*
247         En las maquinas Enigma, se da el "paso" en los rotores antes de
248         recorrer el circuito. Como la UI se actualiza al terminar
249         el metodo, hay que calcular las nuevas posiciones antes de
                cifrar
250          */
251         helper.recalcularPosiciones(pos, turnoverRotor1, turnoverRotor2
                );
252
253         //Avanzar los rotores en la UI
254         pos1D.setValue(pos[0]);
255         pos2D.setValue(pos[1]);
256         pos3D.setValue(pos[2]);
257
258         int[] desplazamiento = helper.index(pos);
259
260         return EnigmaD.cifrarCaracter(input, rotor, desplazamiento);
261     }
262
263     private char cifrarEnigmaM3(char input) {
264
265         int[] rotor = {helper.rotorNameToNum(rot1M3.getValue()),
266             helper.rotorNameToNum(rot2M3.getValue()),
267             helper.rotorNameToNum(rot3M3.getValue()),
268             helper.rotorNameToNum(refM3.getValue())};
269
270         char[] pos = {pos1M3.getValue(), pos2M3.getValue(), pos3M3.
                getValue()};
271
272         char[] turnoverRotor1 = EnigmaM3.getTurnover(rotor[0]);
273         char[] turnoverRotor2 = EnigmaM3.getTurnover(rotor[1]);
274
275         /*
276         En las maquinas Enigma, se da el "paso" en los rotores antes de
277         recorrer el circuito. Como la UI se actualiza al terminar
278         el metodo, hay que calcular las nuevas posiciones antes de
                cifrar
279          */
280         helper.recalcularPosiciones(pos, turnoverRotor1, turnoverRotor2
                );
281
282         //Avanzar los rotores en la UI
```

```
283         pos1M3.setValue(pos[0]);
284         pos2M3.setValue(pos[1]);
285         pos3M3.setValue(pos[2]);
286 
287         int[] desplazamiento = helper.index(pos);
288 
289         return EnigmaM3.cifrarCaracter(input, rotor, desplazamiento);
290     }
291 
292     private char cifrarEnigmaM4(char input) {
293 
294         int[] rotor = {helper.rotorNameToNum(rot1M4.getValue()),
295             helper.rotorNameToNum(rot2M4.getValue()),
296             helper.rotorNameToNum(rot3M4.getValue()),
297             helper.rotorNameToNum(rotExtraM4.getValue()),
298             helper.rotorNameToNum(refM4.getValue())};
299 
300         char[] pos = {pos1M4.getValue(), pos2M4.getValue(), pos3M4.
                getValue(),
301             posExtraM4.getValue()};
302 
303         char[] turnoverRotor1 = EnigmaM4.getTurnover(rotor[0]);
304         char[] turnoverRotor2 = EnigmaM4.getTurnover(rotor[1]);
305 
306         /*
307         En las maquinas Enigma, se da el "paso" en los rotores antes de
308         recorrer el circuito. Como la UI se actualiza al terminar
309         el metodo, hay que calcular las nuevas posiciones antes de
                cifrar
310          */
311         helper.recalcularPosiciones(pos, turnoverRotor1, turnoverRotor2
                );
312 
313         //Avanzar los rotores en la UI
314         pos1M4.setValue(pos[0]);
315         pos2M4.setValue(pos[1]);
316         pos3M4.setValue(pos[2]);
317 
318         int[] desplazamiento = helper.index(pos);
319 
320         return EnigmaM4.cifrarCaracter(input, rotor, desplazamiento);
321     }
322 
323     private char cifrarPlugboard(char input) {
324         Iterator<String> plugboardIterator = plugboard.listIterator();
325         String pareja;
326         while (plugboardIterator.hasNext()) {
327             pareja = plugboardIterator.next();
328             if (pareja.contains(Character.toString(input))) {
329                 if (pareja.charAt(0) == input) {
330                     return pareja.charAt(1);
331                 } else {
332                     return pareja.charAt(0);
333                 }
334             }
335         }
336         return input;
337     }
338 
```

```
339     private void iluminacion(Circle lamp) {
340
341         if (ultimaLampIluminada != null) {
342             ultimaLampIluminada.setFill(Color.WHITE);
343         }
344         ultimaLampIluminada = lamp;
345         lamp.setFill(Color.YELLOW);
346     }
347
348     private Circle lampSelect(char lamp) {
349         switch (lamp) {
350             case 'A':
351                 return lampA;
352             case 'B':
353                 return lampB;
354             case 'C':
355                 return lampC;
356             case 'D':
357                 return lampD;
358             case 'E':
359                 return lampE;
360             case 'F':
361                 return lampF;
362             case 'G':
363                 return lampG;
364             case 'H':
365                 return lampH;
366             case 'I':
367                 return lampI;
368             case 'J':
369                 return lampJ;
370             case 'K':
371                 return lampK;
372             case 'L':
373                 return lampL;
374             case 'M':
375                 return lampM;
376             case 'N':
377                 return lampN;
378             case 'O':
379                 return lampO;
380             case 'P':
381                 return lampP;
382             default:
383             case 'Q':
384                 return lampQ;
385             case 'R':
386                 return lampR;
387             case 'S':
388                 return lampS;
389             case 'T':
390                 return lampT;
391             case 'U':
392                 return lampU;
393             case 'V':
394                 return lampV;
395             case 'W':
396                 return lampW;
397             case 'X':
```

```
398                 return lampX;
399             case 'Y':
400                 return lampY;
401             case 'Z':
402                 return lampZ;
403         }
404     }
405 
406     private Circle plugSelect(char plug) {
407         switch (plug) {
408             case 'A':
409                 return plugA;
410             case 'B':
411                 return plugB;
412             case 'C':
413                 return plugC;
414             case 'D':
415                 return plugD;
416             case 'E':
417                 return plugE;
418             case 'F':
419                 return plugF;
420             case 'G':
421                 return plugG;
422             case 'H':
423                 return plugH;
424             case 'I':
425                 return plugI;
426             case 'J':
427                 return plugJ;
428             case 'K':
429                 return plugK;
430             case 'L':
431                 return plugL;
432             case 'M':
433                 return plugM;
434             case 'N':
435                 return plugN;
436             case 'O':
437                 return plugO;
438             case 'P':
439                 return plugP;
440             default:
441             case 'Q':
442                 return plugQ;
443             case 'R':
444                 return plugR;
445             case 'S':
446                 return plugS;
447             case 'T':
448                 return plugT;
449             case 'U':
450                 return plugU;
451             case 'V':
452                 return plugV;
453             case 'W':
454                 return plugW;
455             case 'X':
456                 return plugX;
```

```
457                 case 'Y':
458                     return plugY;
459                 case 'Z':
460                     return plugZ;
461             }
462         }
463 
464         private void playAudio() {
465             AudioClip click = new AudioClip(this.getClass().getResource("
                    switch.mp3").toString());
466             click.play();
467         }
468 
469         private boolean estaEnchufadoPlugboard(String s) {
470             Iterator<String> plugboardIterator = plugboard.listIterator();
471             while (plugboardIterator.hasNext()) {
472                 String pair = plugboardIterator.next();
473                 if (pair.contains(s)) {
474                     return true;
475                 }
476             }
477             return false;
478         }
479 
480         private void emparejarPlugsParte1(String input) {
481             plugboardPareja = input;
482             colorPareja = Color.color(Math.random(), Math.random(), Math.
                    random());
483             plugSelect(input.charAt(0)).setFill(colorPareja);
484         }
485 
486         private void emparejarPlugsParte2(String input) {
487             String aux = plugboardPareja + input;
488             plugboard.add(aux);
489             plugSelect(input.charAt(0)).setFill(colorPareja);
490 
491             System.out.println(plugboard.toString());
492         }
493 
494         private void cancelarEmparejamientoPlugboard(String input) {
495             //Click en el que se esta intentando emparejar = Cancelar
496             emparejandoPlugs = false;
497             plugSelect(input.charAt(0)).setFill(Color.WHITE);
498         }
499 
500         private void deshacerParejaPlugboard(String s) {
501             for (int parejas = plugboard.size() - 1; parejas >= 0; parejas
                    --) {
502                 String pareja = plugboard.get(parejas);
503                 if (pareja.contains(s)) {
504                     plugSelect(pareja.charAt(0)).setFill(Color.WHITE);
505                     plugSelect(pareja.charAt(1)).setFill(Color.WHITE);
506                     plugboard.remove(pareja);
507                 }
508             }
509             System.out.println(plugboard.toString());
510         }
511 }
```

A.3 Clases Enigma

A.3.1. EnigmaD.java

```
1  package enigmamachine;
2
3  import auxiliar.helper;
4
5  /**
6   * @author Alexandre Olau Moreno Frances
7   */
8  public class EnigmaD {
9
10     protected static final String[] ROTOR = {
11         "LPGSZMHAEOQKVXRFYBUTNICJDW",
12         "SLVGBTFXJQOHEWIRZYAMKPCNDU",
13         "CJGDPSHKTURAWZXFMYNQOBVLIE",
14         "IMETCGFRAYSQBZXWLHKDVUPOJN"};
15     //ROTOR[0-2]: Rotores disponibles, se suele usar uno de cada en
           cualquier orden
16     //ROTOR[3]: Rotor reflector
17     //Esta maquina solo tiene un reflector disponible, pero puede
           variar su posicion
18
19     protected static final char[][] TURNOVER_NOTCHES = {{'Y'}, {'E'}, {
           'N'}};
20
21     protected static final int N_ROTORES_ACTIVOS = 3;
22
23     public static char cifrarCaracter(char input, int[] rot, int[]
           offset) {
24         int cif = helper.index(input);
25
26         for (int i = 0; i < N_ROTORES_ACTIVOS; i++) {
27             cif = cifradoRotor(cif, ROTOR[rot[i]], offset[i], true);
28         }
29
30         cif = cifradoRotor(cif, ROTOR[3], offset[3], true);
31
32         for (int i = N_ROTORES_ACTIVOS - 1; i >= 0; i--) {
33             cif = cifradoRotor(cif, ROTOR[rot[i]], offset[i], false);
34         }
35
36         return helper.index(cif);
37     }
38
39     protected static int cifradoRotor(int input, String rotor, int
           desplazamiento, boolean DerIzq) {
40         int intAux;
41         char charAux;
42
43         intAux = input + desplazamiento;        //Input avanza el
               offset
44         intAux = helper.modABC(intAux);         //La suma se ajusta
               para estar entre 0-26
45
46         if (DerIzq) {
```

```
47              charAux = rotor.charAt(intAux);      //Cambio de indice con
                    el rotor
48              intAux = helper.index(charAux);      //Pasar char a int
49          } else {
50              charAux = helper.index(intAux);      //Cambio de indice con
                    el ABC...
51              intAux = rotor.indexOf(charAux);     //Pasar char a int
52          }
53
54          intAux -= desplazamiento;                //Segunda aplicacion
                del offset
55          intAux = helper.modABC(intAux);          //La resta se ajusta
                para estar entre 0-26
56
57          return intAux;
58      }
59
60      public static char[] getTurnover(int posRotor) {
61          return TURNOVER_NOTCHES[posRotor];
62      }
63  }
```

A.3.2. EnigmaM3.java

```
1   /*
2    * To change this license header, choose License Headers in Project
         Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package enigmamachine;
7
8   import auxiliar.helper;
9
10  /**
11   * @author Alexandre Olau Moreno Frances
12   */
13  public class EnigmaM3 extends EnigmaD {
14
15      private static final String[] ROTOR = {
16          "EKMFLGDQVZNTOWYHXUSPAIBRCJ", //ROT-1
17          "AJDKSIRUXBLHWTMCQGZNPYFVOE", //ROT-2
18          "BDFHJLCPRTXVZNYEIWGAKMUSQO", //ROT-3
19          "ESOVPZJAYQUIRHXLNFTGKDCMWB", //ROT-4
20          "VZBRGITYUPSDNHLXAWMJQOFECK", //ROT-5
21          "JPGVOUMFYQBENHZRDKASXLICTW", //ROT-6
22          "NZJHGRCXMYSWBOUFAIVLPEKQDT", //ROT-7
23          "FKQHTLXOCBJSPDZRAMEWNIUYGV", //ROT-8
24          "YRUHQSLDPXNGOKMIEBFZCWVJAT", //UKW-B
25          "FVPJIAOYEDRZXWGCTKUQSBNMHL"};//UKW-C
26      //ROTOR[0-7]: Rotores disponibles, se eligen 3 en cualquier orden
27      //ROTOR[8-9]: Reflectores disponibles, se elige uno
28
29      protected static final char[][] TURNOVER_NOTCHES = {{'Q'}, {'E'}, {
            'V'},
30      {'J'}, {'Z'}, {'Z', 'M'}, {'Z', 'M'}, {'Z', 'M'}};
31
```

```
32      public static char cifrarCaracter(char input, int[] rot, int[]
            offset) {
33          int cif = helper.index(input);
34
35          for (int posRotor = 0; posRotor < N_ROTORES_ACTIVOS; posRotor
                ++) {
36              cif = cifradoRotor(cif, ROTOR[rot[posRotor]], offset[
                    posRotor], true);
37          }
38
39          cif = cifradoRotor(cif, ROTOR[rot[3]], 0, true);
40
41          for (int posRotor = N_ROTORES_ACTIVOS - 1; posRotor >= 0;
                posRotor--) {
42              cif = cifradoRotor(cif, ROTOR[rot[posRotor]], offset[
                    posRotor], false);
43          }
44
45          return helper.index(cif);
46      }
47
48      public static char[] getTurnover(int posRotor) {
49          return TURNOVER_NOTCHES[posRotor];
50      }
51  }
```

A.3.3. EnigmaM4.java

```
1   /*
2    * To change this license header, choose License Headers in Project
         Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package enigmamachine;
7
8   import auxiliar.helper;
9
10  /**
11   * @author Alexandre Olau Moreno Frances
12   */
13  public final class EnigmaM4 extends EnigmaM3 {
14
15      private static final String[] ROTOR = {
16          "EKMFLGDQVZNTOWYHXUSPAIBRCJ", //ROT-1
17          "AJDKSIRUXBLHWTMCQGZNPYFVOE", //ROT-2
18          "BDFHJLCPRTXVZNYEIWGAKMUSQO", //ROT-3
19          "ESOVPZJAYQUIRHXLNFTGKDCMWB", //ROT-4
20          "VZBRGITYUPSDNHLXAWMJQOFECK", //ROT-5
21          "JPGVOUMFYQBENHZRDKASXLICTW", //ROT-6
22          "NZJHGRCXMYSWBOUFAIVLPEKQDT", //ROT-7
23          "FKQHTLXOCBJSPDZRAMEWNIUYGV", //ROT-8
24          "ENKQAUYWJICOPBLMDXZVFTHRGS", //UKW-b
25          "RDOBJNTKVEHMLFCWZAXGYIPSUQ", //UKW-c
26          "LEYJVCNIXWPBQMDRTAKZGFUHOS", //Beta
27          "FSOKANUERHMBTIYCWLQPZXVGJD"};//Gamma
28      //ROTOR[0-7]: Rotores disponibles, se eligen 3 en cualquier orden
```

```
29      //ROTOR[8-9]: Rotores extra, se elige uno
30      //ROTOR[10-11]: Reflectores disponibles, se elige uno
31
32      protected static final int N_ROTORES_ACTIVOS = 4;
33
34      //Este modelo tiene los mismos "notches" que el M3 y los hereda de
            este
35      public static char cifrarCaracter(char input, int[] rot, int[]
            offset) {
36          int cif = helper.index(input);
37
38          for (int i = 0; i < N_ROTORES_ACTIVOS; i++) {
39              cif = cifradoRotor(cif, ROTOR[rot[i]], offset[i], true);
40          }
41
42          cif = cifradoRotor(cif, ROTOR[rot[3]], 0, true);
43
44          for (int i = N_ROTORES_ACTIVOS - 1; i >= 0; i--) {
45              cif = cifradoRotor(cif, ROTOR[rot[i]], offset[i], false);
46          }
47
48          return helper.index(cif);
49      }
50  }
```

A.4 helper.java

```
1   /*
2    * To change this license header, choose License Headers in Project
         Properties.
3    * To change this template file, choose Tools | Templates
4    * and open the template in the editor.
5    */
6   package auxiliar;
7
8   /**
9    * @author Alexandre Olau Moreno Frances
10   */
11  public class helper {
12
13      private static final String ABC = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
14
15      public static int rotorNameToNum(String rotorName) {
16          switch (rotorName) {
17              case "I":
18                  return 0;
19              case "II":
20                  return 1;
21              case "III":
22                  return 2;
23              case "IV":
24                  return 3;
25              case "V":
26                  return 4;
27              case "VI":
28                  return 5;
```

```
            case "VII":
                return 6;
            case "VIII":
                return 7;
            case "UKW-B":
            case "UKW-b":
                return 8;
            case "UKW-C":
            case "UKW-c":
                return 9;
            case "Beta":
                return 10;
            case "Gamma":
                return 11;
            default:
                return 0;
        }
    }

    public static void recalcularPosiciones(char[] pos, char[] turn1o,
        char[] turn2o) {
        if (turn1o.length > 1 && turn2o.length > 1) {
            if (pos[1] == turn2o[0] || pos[1] == turn2o[1]) {
                pos[1]++;
                pos[2]++;
            } else if (pos[0] == turn1o[0] || pos[0] == turn1o[1]) {
                pos[1]++;
            }
        } else if (turn1o.length > 1) {
            if (pos[1] == turn2o[0]) {
                pos[1]++;
                pos[2]++;
            } else if (pos[0] == turn1o[0]) {
                pos[1]++;
            }
        } else if (turn2o.length > 1) {
            if (pos[1] == turn2o[0] || pos[1] == turn2o[1]) {
                pos[1]++;
                pos[2]++;
            } else if (pos[0] == turn1o[0] || pos[0] == turn1o[1]) {
                pos[1]++;
            }
        } else {
            if (pos[1] == turn2o[0]) {
                pos[1]++;
                pos[2]++;
            } else if (pos[0] == turn1o[0]) {
                pos[1]++;
            }
        }

        pos[0]++;

        noPasarseDeZ(pos);
    }

    private static void noPasarseDeZ(char[] pos) {
        for (int i = 0; i < pos.length; i++) {
            if (pos[i] > 'Z') {
```

```
87                  pos[i] = 'A';
88              }
89          }
90      }
91
92      public static int index(char ch) {
93          return ABC.indexOf(ch);
94      }
95
96      public static int index(String str) {
97          return ABC.indexOf(str.charAt(0));
98      }
99
100     public static char index(int x) {
101         return ABC.charAt(x);
102     }
103
104     public static int[] index(char[] ch) {
105         int res[] = new int[ch.length];
106         for (int i = 0; i < ch.length; i++) {
107             res[i] = index(ch[i]);
108         }
109         return res;
110     }
111
112     public static int modABC(int x) {
113         int mod = x;
114         while (mod < 0) {
115             mod += ABC.length();
116         }
117         return mod %ABC.length();
118     }
119 }
```