Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Ampliación de la versión multiplataforma de PredWin+

Autor: Carlos Carrasco Álvarez

Tutora: Sira E. Palazuelos Cagigas

2023

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de Computadores

Trabajo Fin de Grado

Ampliación de la versión multiplataforma de PredWin+

Autor: Carlos Carrasco Álvarez

Tutora: Sira E. Palazuelos Cagigas

Tribunal:

Presidente: Biel Piero Eloy Alvarado Vásquez

Vocal 1º: Carlos Julián Martín Arguedas

Vocal 2º: Sira E. Palazuelos Cagigas

Fecha de depósito: 30 de junio de 2023

Agradecimientos

Este proyecto no habría sido posible sin el apoyo y guía incondicional de mi tutora, Sira Elena Palazuelos. Desde el primer momento, ha sido capaz de motivarme y animarme a pesar de las dificultades inherentes a la situación en la que se encontraba el proyecto. Agradezco sinceramente el tiempo, energía y compromiso que ha invertido en ayudarme. Su dedicación ha sido fundamental para alcanzar las metas que nos propusimos juntos. Siempre ha estado ahí para brindarme un enfoque distinto de las cosas y me ha ayudado a plantearlas desde otro punto de vista.

Debo dar también las gracias especialmente a Oumayma Laaroussi, sin ella no habría podido desarrollar la mitad de las cosas que he hecho. A pesar de su poco tiempo, siempre ha sido capaz de sacar un par de horas para ayudarme a resolver mis dudas y es algo de lo que estoy muy agradecido.

También doy las gracias a mi pareja, Daniela, por haberme acompañado durante estos meses de estrés y por estar siempre presente para apoyarme y darme ánimos. Eres la persona que más ha confiado en mí y nunca podré agradecértelo lo suficiente.

Gracias a mi mejor amiga, Elena. Agradezco tu disposición para probar todo lo que te pido y por estar siempre presente. Tu apoyo incondicional significa mucho para mí.

No puedo olvidar mencionar a mi compañero de clase y amigo Mario. Quiero agradecerte por todas esas mañanas en las que me animaste a dejar de lado el TFG y unirme a ti a jugar. Aunque esas mañanas no fueran las más productivas del mundo, siempre nos lo pasábamos bien.

Finalmente, gracias a mis padres por haberme apoyado durante esta etapa de mi vida y darme la oportunidad de estudiar lo que quería.

Resumen

PredWin es un editor de texto orientado a personas con discapacidades físicas. Este TFG parte de una versión multiplataforma existente y la actualiza para aumentar su capacidad y que usuarios con diferentes capacidades se puedan beneficiar de ella. La actualización incluye mejoras significativas, como un nuevo menú llamado "oír" para facilitar la comunicación y su uso por personas con baja visión. También se ha añadido la función de predicción de palabras y escritura automática de ciertos caracteres para agilizar la escritura, el modo de barrido lineal para reducir las pulsaciones necesarias para escribir los textos y la posibilidad de realizar autoguardado.

Palabras clave: Editor de texto, personas con discapacidad física, teclado virtual, barrido lineal, predicción de palabras.

Abstract

PredWin is a text editor oriented to people with physical disabilities. This Bachelor's Degree Final Project builds on an existing cross-platform version and updates it to increase its capacity so that users with different capabilities can benefit from it. The updated version includes significant improvements, such as a new menu called "hear" to facilitate communication and use by people with low vision. Also added are word prediction and automatic insertion of certain characters to speed up typing, linear scanning mode to reduce keystrokes needed to type texts and the possibility to autosave.

Keywords: Text editor, people with physical disabilities, virtual keyboard, linear scanning, word prediction.

Índice general

Índice de figuras

Índice de listados de código fuente

Lista de acrónimos

LLM	Large Lenguage Model.
SAAC	Sistemas Aumentativos y Alternativos de Comunicación.
TTS	Text To Speech.

Capítulo 1

Introducción

1.1 Presentación

La escritura y la comunicación son fundamentales en la vida de las personas, permitiendo expresar deseos, pensamientos y necesidades. Es habitual que las personas con ciertas discapacidades (parálisis cerebral, ELA, etc.) tengan problemas de comunicación y también físicos, que les impiden tener acceso a herramientas habituales como la escritura en papel/pizarra o los teléfonos móviles y ordenadores si no están adaptados a sus necesidades físicas. Estas dificultades generan frustración y pueden tener una gran repercusión en su calidad de vida. En la actualidad es posible adaptar los elementos tecnológicos para que puedan ser utilizados con personas por capacidades diferentes, por ejemplo, permitiendo el uso de comunicadores o editores de texto con pulsadores de distintos tipos, pantallas táctiles etc., siendo de gran ayuda para garantizar la inclusión de estas personas y la igualad de oportunidades.

PredWin es una aplicación diseñada específicamente para personas con discapacidad física, que no pueden utilizar el teclado/ratón convencionales, dándoles la posibilidad de escribir texto de manera sencilla y adaptada a sus limitaciones de movilidad. Su objetivo principal es ofrecer un medio de escritura y comunicación eficiente y accesible.

La aplicación se basa en el barrido de las distintas opciones (menús, letras, etc.) que se van resaltando secuencialmente y se seleccionan cuando el usuario utiliza el dispositivo externo que mejor pueda controlar: una única tecla, un pulsador o un sensor de movimiento o soplido. Esto permite al usuario escribir texto con el mínimo esfuerzo físico necesario.

Una vez que el texto es escrito, PredWin tiene la capacidad de convertirlo en voz, permitiendo que el mensaje sea hablado por la propia aplicación. De esta manera, se facilita la comunicación, ya que no solo permite escribir sus pensamientos, sino también transmitirlos oralmente, pudiendo llamar la atención de personas que no se encuentran en la misma habitación, no están mirando la pantalla o comunicarse con interlocutores que no saben leer.

PredWin constituye una herramienta de gran valor para las personas con discapacidad física, brindándoles la oportunidad de escribir y comunicarse de manera efectiva y autónoma, superando las barreras físicas que podrían limitar su capacidad de expresión.

Esta memoria está estructurada de la siguiente forma: en el capítulo 2 presentamos un breve estado del arte. En el capítulo 3 se describe la arquitectura de la aplicación y las opciones que se han añadido. El capítulo 4 contiene las conclusiones y líneas futuras y, a continuación, se puede encontrar la bibliografía, el manual de usuario, el código fuente de varias funciones, el anexo con el procedimiento de instalación y configuración del proyecto para los distintos sistemas operativos y el pliego de condiciones.

Capítulo 2

Estudio teórico

2.1 Introducción

Con el paso del tiempo y el avance tecnológico, nos encontramos en pleno proceso de migración y desarrollo de PredWin hacia sistemas operativos más modernos y versátiles. Inicialmente, la aplicación se desarrolló para entornos como Windows 3.11 o Windows 2000, y estamos trabajando para adaptarla y añadir nuevas funcionalidades en sistemas operativos actuales, como Windows 11, Linux, Android e iOS.

2.2 Estado del Arte

Los problemas que afectan a la comunicación tienen un impacto significativo en la calidad de vida de las personas. La capacidad de comunicarse de manera efectiva es fundamental para llevar a cabo tareas cotidianas, como transmitir información, expresar necesidades y captar la atención de otras personas. Para aquellos que experimentan dificultades en el habla, estas actividades básicas pueden resultar desafiantes e incluso frustrantes.

Los Sistemas Aumentativos y Alternativos de Comunicación (SAAC) [1] son recursos y herramientas que permiten a las personas con diversidad funcional y dificultades en el habla o el lenguaje expresarse y comunicarse de manera efectiva. Estos sistemas proporcionan formas de expresión diferentes al lenguaje hablado convencional, con el objetivo de aumentar el nivel de expresión (aumentativo) y/o compensar las dificultades de comunicación (alternativo) que enfrentan estas personas.

El uso de SAACs permite a las personas con diversidad funcional comunicarse con los demás, brindándoles la oportunidad de expresar sus decisiones, emociones, sentimientos, opiniones y cualquier otra información relevante. Estos sistemas pueden incluir el uso de imágenes, símbolos, gestos, señales, escritura, dispositivos de comunicación electrónica y otras tecnologías, adaptadas a las necesidades individuales de cada persona.

Estos son algunos ejemplos de ayuda a la comunicación que existen en la actualidad:

- MegaBee: es un comunicador específico [1] basado en un sistema de escritura asistida conocido como seguimiento ocular. Este sistema utiliza el movimiento y parpadeo de los ojos como método de selección de letras o frases. Las opciones de selección se presentan en una pantalla ubicada en la parte inferior del dispositivo. A través de la conexión Bluetooth, el texto puede enviarse, almacenarse y visualizarse en un ordenador, además de cargar vocabulario específico del usuario [2].

[1] Dispositivo electrónico, habitualmente portátil, diseñado específicamente como ayuda a la comunicación.

- Speak for Yourself: es un sistema de comunicación basado en plataformas estándar [2]. Utiliza un vocabulario basado en palabras que se utilizan con mayor frecuencia en la comunicación. Permite a los usuarios aprender a utilizar el dispositivo de comunicación siguiendo los mismos principios que se utilizan para tocar un instrumento musical o escribir en un teclado [3].

- AraBoard: es una aplicación basada en tabletas o dispositivos móviles que utiliza una interfaz visual y táctil para proporcionar una plataforma para que las personas con dificultades en la comunicación, especialmente aquellas con discapacidades del habla o del lenguaje, se expresen utilizando símbolos y texto. Permite a los usuarios seleccionar símbolos, palabras o frases predefinidas para comunicarse de manera efectiva. La aplicación presenta una variedad de sistemas de símbolos, como pictogramas, fotografías o letras, y puede adaptarse a las necesidades individuales de los usuarios [4].

[2] Sistema convencional en los que se instalan determinadas aplicaciones que lo convierten en un sistema de comunicación.

Capítulo 3

PredWin

3.1 Introducción

PredWin es una aplicación que permite editar texto a personas que no pueden utilizar el teclado o ratón convencionales. En este TFG se ha partido de una versión nueva que se está desarrollando en entorno de Qt que permite trabajar en proyectos multiplataforma y generar versiones para Windows, Linux, Android e iOS, y se han añadido diferentes funcionalidades para facilitar su uso. En la figura 3.1, podemos observar la ventana principal, que se divide en tres módulos: un menú superior, un cuadro de texto y una matriz de letras.

Figura 3.1: Ventana principal de PredWin.

En esta versión, nos hemos centrado en mejorar la experiencia del usuario al introducir nuevas funcionalidades y menús que facilitarán la comunicación y agilizarán la escritura. Nuestro objetivo principal es ofrecer a los usuarios una experiencia más fluida y eficiente, permitiéndoles comunicarse de manera más efectiva y escribir con mayor rapidez.

Hemos introducido una nueva opción, OÍR, que ofrece a los usuarios con problemas de comunicación una amplia gama de opciones para facilitar su expresión y personalizar la voz de la aplicación. Esta funcionalidad también será beneficiosa para aquellos usuarios con problemas de visión, ya que hemos integrado un sintetizador de voz que les permitirá escuchar las opciones de los menús en los distintos modos de acceso de la aplicación, posibilitando el uso de la aplicación a usuarios que no pueden leer los menús por problemas visuales, o posturales (no pueden orientar adecuadamente la cabeza para ver la pantalla, o no les resulta cómodo).

Además, hemos añadido un nuevo modo de acceso, el barrido lineal, para los usuarios que tengan problemas al realizar numerosas pulsaciones debido a la fatiga o a daños en el músculo que utilicen para hacerlas. Este nuevo modo de barrido, aunque más lento, permite seleccionar cada opción utilizando la mitad de las pulsaciones que en modo filas y columnas, siendo más adecuado para este tipo de usuarios.

También hemos incluido mejoras significativas en la escritura, como la función de predicción de palabras, que permitirá a los usuarios escribir de manera más rápida y eficiente y la inserción automática de ciertos caracteres que reducirán el número de pulsaciones necesarias para escribir.

Además, se ha añadido la posibilidad de que la aplicación realice el guardado automático del texto que se está escribiendo cada cierto tiempo configurable. Esto garantizará que los usuarios no tengan que preocuparse por guardar su trabajo manualmente, y en caso de que la aplicación se cierre inesperadamente, su progreso no se pierda.

3.2 Arquitectura de PredWin

En este apartado, vamos a profundizar en la implementación y los elementos que conforman PredWin. Como se puede observar en la figura 3.2 [5], la aplicación se compone de diferentes módulos, como la gestión de texto, gestión de barrido y menús superiores. En este TFG se han agregado nuevas funcionalidades generales a estos módulos con dos objetivos fundamentales, además de mejorar su funcionamiento: por una parte, acelerar el proceso de escritura y, por otra, aumentar las posibilidades de personalización para que lo puedan utilizar personas con capacidades diferentes.

El módulo de gestión de texto se encarga de manejar los elementos que componen el cuadro de texto principal, proporcionando al usuario un acceso intuitivo a la edición y escritura de texto. En esta versión mejorada de PredWin, se han añadido nuevas funcionalidades, como el teclado predictivo de la matriz de letras o la implementación de mejoras en la escritura que reducen el número de pulsaciones requeridas para introducir el texto y mejoran la calidad del texto generado.

En el módulo de gestión de barrido, se ha incorporado un nuevo modo de barrido lineal, que permite al usuario navegar por la aplicación de manera secuencial y acceder a las diferentes funcionalidades opción a opción. La diferencia fundamental con el barrido por filas y columnas disponible en la versión anterior se produce a la hora de escribir, en el barrido de la matriz de letras: el barrido por filas y columnas permite escribir más rápido, pero requiere 2 pulsaciones por letra. Este esfuerzo es muy grande para algunos usuarios, que necesitan un sistema que necesite menos pulsaciones y más separadas, como es el barrido lineal.

En la sección de menús superiores, se ha añadido el nuevo menú oír, que ofrece una variedad de opciones que mejoran la experiencia de los usuarios que utilizan el sistema para comunicarse o que tienen problemas de visión que no se pueden solventar aumentando el tamaño de letra.

Además, también se ha incorporado el autoguardado, permitiendo configurar el tiempo entre guardados.

En los apartados siguientes se describirán las nuevas características de PredWin. Para cada una se describirá su funcionamiento a nivel de usuario, indicando las opciones añadidas a los menús, los usuarios beneficiados por ellas y, a continuación, una explicación técnica de su implementación.

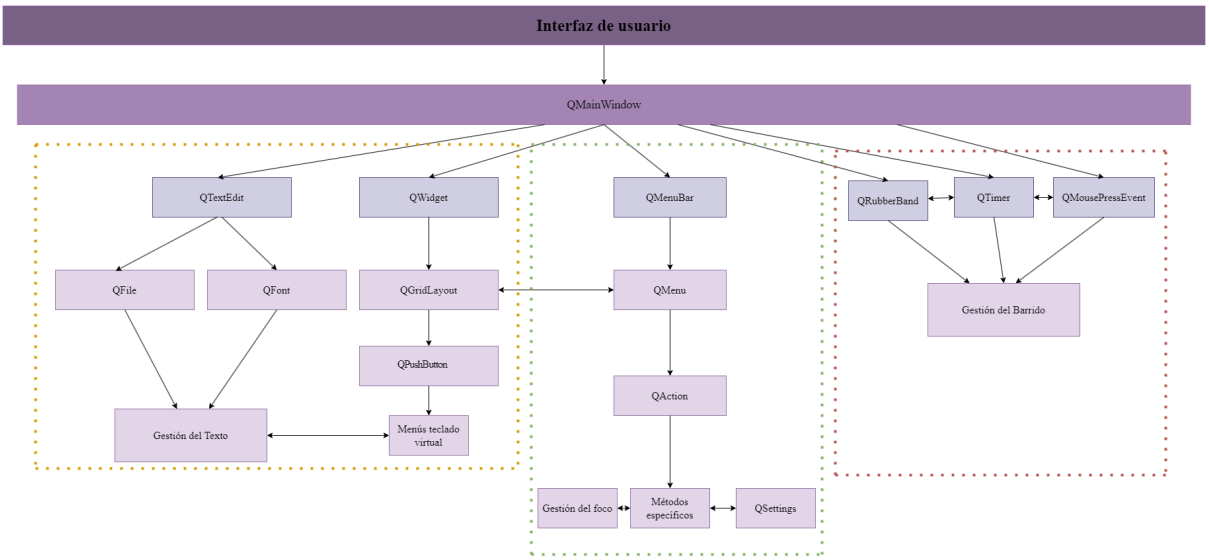


Figura 3.2: Diagrama de bloques de PredWin.

3.3 Ampliación de PredWin

3.3.1 Menú Superior

En el diagrama de la figura 3.3 [5] podemos ver el módulo que representa los menús en nuestra aplicación. La clase QMenuBar implementa la barra de menú superior en PredWin, donde se ubican todos los menús de la aplicación. Para lograr esto, se utiliza la clase QMenu, que permite crear menús desplegables y agregar acciones para que el usuario las ejecute. Estas acciones están representadas por la clase QAction.

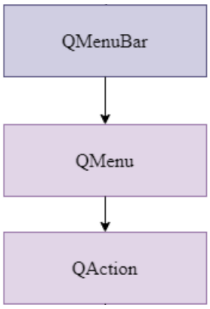


Figura 3.3: Menú superior.

Hemos añadido la opción “Oír” al listado de opciones del menú superior, quedando las siguientes:

- Archivo
- Edición
- Oír
- Formato
- Opciones
- Ayuda

3.3.1.1 Menú Oír

En la figura 3.4 podemos observar las opciones incluidas en este nuevo menú, que contribuirán a mejorar la experiencia de los usuarios que utilicen la aplicación, tanto para comunicarse como por problemas de visión que no les permitan leer los menús o el texto escrito.

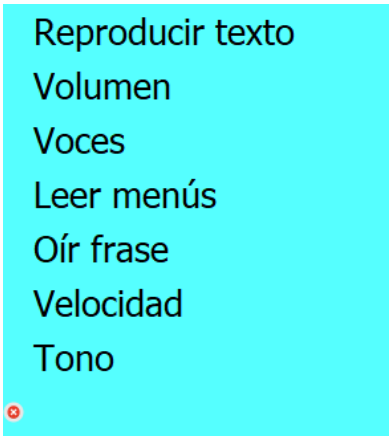


Figura 3.4: Menú Oír.

Este nuevo menú incluye las siguientes opciones:

- Reproducir texto
- Volumen
- Voces
- Leer menús
- Oír frase
- Velocidad
- Tono

3.3.1.1.1 Reproducir texto

La opción "Reproducir texto" se encargará de reproducir todo el contenido que se haya escrito en el cuadro de texto. Esta funcionalidad resulta útil para los usuarios con dificultades de comunicación, ya que les permite transmitir lo que han escrito en la aplicación de manera auditiva.

Además, esta opción ha sido añadida a los menús de "AYUDA", como se muestra en la figura 3.5, para que así se pueda reproducir el texto escrito en cada uno de ellos.

Figura 3.5: Botones del menú de ayuda.

3.3.1.1.2 Volumen

La opción "Volumen" abrirá una nueva ventana, que se muestra en la figura 3.6, en la que aparecerán las opciones para subir y bajar el volumen. El rango ajustable del volumen será de 0 % a 100 %, y se podrá modificar en intervalos de 10 %.

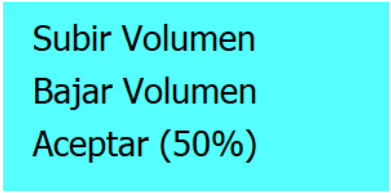


Figura 3.6: Ventana de ajuste del volumen.

3.3.1.1.3 Voces

La opción "Voces" desplegará la ventana que se muestra en la figura 3.7, que permite al usuario elegir la voz que se utilizará tanto para la lectura de los menús como para las funciones de reproducción de texto. Esta funcionalidad será útil para los usuarios que tengan dificultades de comunicación y deseen personalizar la voz para que se asemeje a la suya propia. Esto les permitirá sentir una mayor identificación y familiaridad al utilizar la aplicación.

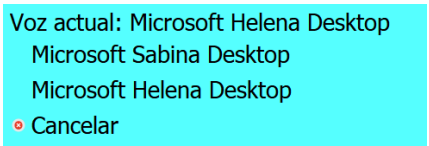


Figura 3.7: Ventana para modificar la voz.

En el caso en el que tengamos más de cinco voces instaladas en nuestro sistema, aparecerá la opción "..." en la lista. Si seleccionamos esta opción, se mostrará una nueva ventana en la que aparecerán las voces siguientes, como se muestra en la figura 3.8. En el caso en el que todavía queden más de cinco voces por mostrar, seguirá apareciendo la opción "..." de manera recurrente hasta que ya no queden más.

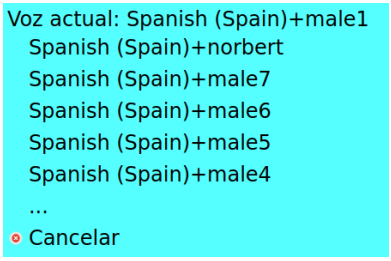


Figura 3.8: Ventana cuando hay más de 5 voces disponibles.

3.3.1.1.4 Leer menús

La opción "Leer menús" permitirá la lectura de los menús en los modos de barrido, barrido lineal y modo directo.

Cuando se habilita esta opción en modo directo, se leerán los menús, las opciones desplegadas y los botones al pasar el ratón sobre ellos. Esto será de gran ayuda para las personas con buena precisión controlando del ratón pero tengan problemas de visión. Por otro lado, si se habilita en los modos de barrido, se leerán los menús a medida que el barrido va resaltando las opciones. Esta función resultará útil para las personas que no puedan utilizar el ratón o teclado y además tengan problemas de visión.

3.3.1.1.5 Oír frase

La opción "Oír frase" se encargará de reproducir la frase actual, la cual se encuentra delimitada por dos puntos. En caso de que sea la primera frase, solo será necesario un punto. Está principalmente orientada a usuarios que presenten dificultades en la comunicación y deseen que las frases se escuchen cuando se escriban. Es especialmente útil en casos donde se necesite expresar frases cortas y directas, como "quiero agua" o "ábreme la puerta", por ejemplo.

3.3.1.1.6 Velocidad

La opción de "Velocidad" desplegará una nueva ventana, tal como se muestra en la figura 3.9. Esta ventana nos permitirá ajustar la velocidad en un rango de -50 % a 50 %. Si el valor es menor, la reproducción de la voz será más lenta, mientras que si es mayor, la reproducción será más rápida.

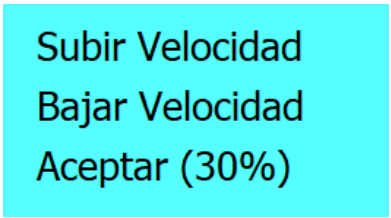


Figura 3.9: Ventana para modificar la velocidad de la voz.

Esta opción proporcionará la posibilidad de personalizar este parámetro a los usuarios que deseen modificar la voz de la aplicación, tanto para comunicarse, como para ajustar la velocidad de lectura de

los menús. De esta manera, se adaptará a las necesidades individuales de cada usuario, mejorando su experiencia de uso.

3.3.1.1.7 Tono

La opción "Tono" desplegará una ventana emergente, como la mostrada en la figura 3.10. Desde aquí podemos modificar el tono de la voz en un rango entre -100 % y 100 %, siendo el de menor valor el tono más grave, y el de mayor valor el más agudo.

Al igual que la "Velocidad", esta opción permitirá una mayor personalización de la voz para los usuarios que deseen modificarla según sus preferencias para comunicarse y también beneficiará a aquellos que tengan problemas de visión y deseen ajustar aún más la voz de la aplicación. De esta manera, se brinda una mayor flexibilidad y adaptabilidad en la experiencia de los usuarios.

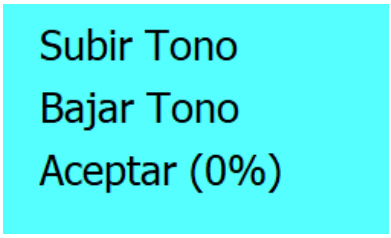


Figura 3.10: Ventana para modificar el tono de la voz.

Es muy importante destacar que no todas las voces permiten configurar su tono y velocidad. También influye el sistema operativo, las voces en distintos sistemas como Windows, Android o Linux son totalmente diferentes. Los usuarios de Linux pueden encontrarse con una selección más limitada de voces de calidad y opciones de configuración en comparación con otros sistemas operativos. Además, pueden encontrar menos aplicaciones y herramientas específicas de Text To Speech (TTS) disponibles y optimizadas para Linux.

3.3.1.2 Implementación del menú oír

Para implementar este nuevo menú desplegable, nos hemos basado en la funcionalidad proporcionada por la clase QTextToSpeech [6]. A través de los métodos y propiedades de esta clase, hemos añadido la funcionalidad necesaria para la síntesis de voz. Esta clase ofrece métodos para configurar y controlar las características de la síntesis de voz, como la velocidad, el tono, el volumen. Además, permite obtener información sobre las voces disponibles en el sistema y sus características específicas.

3.3.1.2.1 Reproducir texto

Para dar funcionalidad a esta opción hemos creado un objeto "QTextToSpeech" que recibirá lo que haya escrito en el cuadro de texto y lo reproducirá el sintetizador de voz mediante la función 'say()'.

3.3.1.2.2 Volumen

La opción de volumen despliega una nueva ventana que permite al usuario modificar el volumen. En la sección 3.3.4 del documento se explica en detalle la creación de esta ventana, sus opciones y su

funcionalidad en los modos de barrido. Utilizaremos los métodos 'getVolume()' y 'setVolume()' para obtener el volumen actual del sintetizador de voz y para modificarlo cada vez que se presione el botón de subir o bajar el volumen.

3.3.1.2.3 Voces

La opción de modificar la voz del sintetizador también se implementa mediante una nueva ventana en la que se agregan las voces disponibles.

Para establecer la voz que el usuario elija utilizaremos la función 'setVoice()' que se encargará de asignar al sintetizador la voz que debe usar.

Para añadir la opción "..." crearemos una nueva función llamada 'showMoreVoices()' [B.5], en la que se fijará un número máximo de voces a mostrar, en nuestro caso 5, y se irán añadiendo las voces como opciones de esta nueva ventana.

Para el funcionamiento de esta función 'showMoreVoices()' obtendremos un vector de voces disponibles de nuestro sistema. Recorremos el vector de voces y creamos una acción para cada una de ellas, agregándola a la nueva ventana. Si hay menos de 5 voces disponibles, conectaremos las acciones de las voces a la función 'modoVoz(int index)'. Esta función recibirá un índice y seleccionará la voz correspondiente utilizando dicho índice como parámetro.

En cambio, si hay más de 5 voces disponibles, se conectará a la misma función 'showMoreVoices()' para que, al presionar la opción "...", se muestren las voces adicionales hasta que se haya mostrado la lista completa de voces disponibles.

3.3.1.2.4 Leer menús

Para implementar la funcionalidad de que el sintetizador de voz lea las opciones de PredWin, al presionar la opción "Leer menús", la variable 'sonidoMenus' se establecerá a 'true'. Esto nos permitirá saber si se debe leer la opción resaltada ya sea en modo directo o en los modos de barrido.

Para que esta opción funcione en el modo de barrido directo sobre los menús superiores y las opciones desplegables, hemos utilizado la función 'hovered()' proporcionada por Qt.

Esta función 'hovered()' presenta un problema, ya que se activa cada vez que la posición del ratón se modifica. Esto significa que si el cursor se encuentra sobre la opción "Volumen" y se mueve ligeramente dentro de la zona de esa opción, el sintetizador de voz repetirá constantemente la lectura de "Volumen".

Para abordar la problemática de la función 'hovered()', que se invoca cada vez que el cursor se posiciona sobre el objeto, hemos implementado una solución para que la lectura de la opción solo ocurra una vez.

Para las opciones desplegables del menú superior, hemos creado una variable booleana para cada opción con un valor inicial establecido a 'false'. Cuando el cursor se encuentra sobre una opción, si la variable está en 'false', se establecerá en 'true', mientras que las demás se establecerán en 'false'. Luego, se leerá la opción actual donde se encuentra el cursor. De esta manera, si el cursor no se retira de la opción en la que está posicionado, no se volverá a leer esa opción.

Para leer las opciones de las ventanas emergentes, como los ajustes de volumen, hemos creado una función receptora de señales llamada 'actionHovered()'. En esta función, comprobaremos si se recibe un puntero de tipo 'QAction'. Si se recibe, se leerá la opción sobre la cual está posicionado el cursor.

Conectaremos cada una de las acciones de las ventanas emergentes a la señal 'QAction:hovered()'. Si se dispara la señal, la función 'actionHovered()' se ejecutará como respuesta.

Para solucionar el problema de que se lea constantemente la opción en las ventanas emergentes, hemos creado un 'QMap<QAction*, bool>' donde introduciremos todas las acciones con el valor inicial en 'false'. Esto nos permite replicar el mismo funcionamiento que para las opciones desplegables del menú superior. Si el cursor se encuentra sobre una opción y su acción correspondiente en el QMap está en 'false', se leerá la opción y se establecerá en 'true', asegurando así que solo se lea una vez, como se muestra en la función 'actionHovered()' [B.3].

Además debemos conectar la señal hovered de todas las acciones al slot 'actionHovered()':

```
// Conectar la señal "hovered" de las acciones al slot actionHovered
connect(subirVolumen, &QAction::hovered, this, &MainWindow::actionHovered);
connect(bajarVolumen, &QAction::hovered, this, &MainWindow::actionHovered);
connect(menuprincipal, &QAction::hovered, this, &MainWindow::actionHovered);
```

Para leer los botones de los teclados, hemos creado una función llamada 'eventFilter(QObject* object, QEvent* event)' que simula el funcionamiento de la función 'hovered()'. Esta función verificará el tipo de evento que se está produciendo y la ubicación en la que se está produciendo. Si el tipo de evento es "Enter", lo que significa que el ratón ha entrado en la zona de pulsación del objeto, y si el objeto es un 'QPushButton', se leerá el botón sobre el cual está posicionado el cursor.

En los modos de barrido, hemos utilizado la variable 'sonidoMenus' en todas las funciones que realizan barrido de menús y teclados. Si la variable está establecida como 'true', se leerá la opción que está siendo resaltada en cada momento durante el proceso de barrido.

La opción de "Leer menús" generaba un problema en relación con la función "Reproducir texto" cuando se utilizaba en conjunto con la lectura de menús, ya que el barrido continuaba y se interrumpía la reproducción del texto. Para solucionarlo, tuvimos que separar los lectores de TTS y asignar uno para la lectura de menús y otro exclusivamente para la reproducción de texto. Además, verificamos si la opción de lectura de menús estaba habilitada para esperar a que finalizara la reproducción de texto en el TTS correspondiente. Para ello, comprobamos el estado del TTS de reproducción de texto. Si está en proceso de lectura, su estado será 'Speaking', y si está listo para reproducir, su estado será 'Ready'. Solo cuando esté en estado 'Ready', podremos continuar con la lectura de menús sin interrupciones [B.4].

3.3.1.2.5 Oír frase

Para reproducir únicamente la última frase del cuadro de texto hemos guardado el texto escrito en una variable hemos dividido las frases con la función 'split(':')'. Una vez tenemos las frases del texto obtendremos la última frase entre dos puntos y el sintetizador de voz la reproduce únicamente si la frase tiene contenido.

3.3.1.2.6 Velocidad

La opción "Velocidad" desplegará una ventana que permitirá modificar la velocidad del sintetizador de voz a través de las funciones 'getRate()' y 'setRate()'. Utilizaremos estos métodos para obtener la velocidad actual del sintetizador de voz y para modificarla cada vez que se presione el botón de subir o bajar velocidad.

3.3.1.2.7 Tono

La opción “Tono” desplegará una ventana que permitirá al usuario modificar el tono del sintetizador de voz. Para lograr esto, utilizaremos las funciones ‘getPitch()’ y ‘setPitch()’ que nos permitirán obtener el tono actual del sintetizador de voz y modificarlo cada vez que se presione el botón de subir o bajar tono.

3.3.1.3 Menú Opciones

En la figura 3.11 se muestran las opciones del menú “Opciones”, en las que hemos añadido “Guardado Auto.” que permite configurar cada cuanto tiempo PredWin realiza un guardado automático del texto que se está escribiendo.

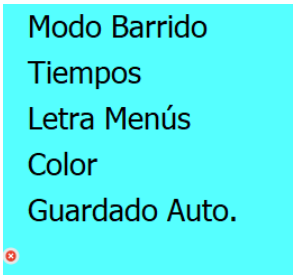


Figura 3.11: Menú Opciones.

3.3.1.3.1 Guardado automático

El guardado automático es una función esencial en PredWin. Sus usuarios requieren más tiempo para escribir y pueden enfrentar dificultades adicionales, como fatiga o interrupciones inesperadas. El guardado automático garantiza que su trabajo se guarde regularmente, evitando la posibilidad de perderlo debido a eventos imprevistos como apagones de luz, errores del sistema u otros problemas.

Además, el guardado automático elimina la necesidad de recordar y realizar manualmente la acción de guardar, permitiendo a los usuarios concentrarse en su tarea en lugar de preocuparse constantemente por guardar su trabajo.

Esta función brinda una sensación de seguridad y tranquilidad a los usuarios, ya que saben que su trabajo se guarda en segundo plano. Además, en caso de cualquier problema, el guardado automático facilita la recuperación de su trabajo previamente guardado.

Al pulsar esta opción se desplegará la ventana de la figura 3.12 en la que podremos seleccionar la frecuencia con la que se realice un guardado automático. El rango está establecido entre 0 y 60 minutos. Si el valor está configurado en 0, significa que el autoguardado se ha deshabilitado, cualquier otro valor habilitará el autoguardado a partir del momento en el que se pulse el botón de aceptar.

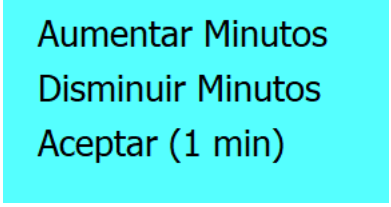


Figura 3.12: Menú de autoguardado.

El autoguardado funciona de la siguiente manera:

- Si el usuario abre la aplicación y escribe sin dar nombre al fichero (no realiza ningún guardado manual), cuando pase el tiempo de autoguardado, el sistema creará un fichero llamado "PredWin-BackupNew" en el que se guardará el texto escrito hasta ese momento. Este proceso se lleva a cabo de forma periódica, siempre y cuando el usuario no desactive la función de autoguardado.
- Si el usuario decide guardar manualmente el trabajo realizado en un archivo, el archivo de backup existente, en caso de haberlo, será eliminado y el autoguardado empezará a guardar en el nuevo archivo creado por el usuario.
- Si el usuario decide salir de la aplicación, se le presentará la opción de guardar el archivo. En caso de que el usuario elija guardar, el archivo se guardará con el nombre proporcionado y luego la aplicación se cerrará. Sin embargo, si el usuario decide no guardar, la aplicación se cerrará sin guardar ninguna información" y borrará el fichero "PredWinBackupNew" si existe.
- Si el usuario abre un nuevo archivo, se le preguntará si desea guardar los cambios del fichero actual si no está vacío. Tanto si la respuesta es positiva como si es negativa, el archivo de backup existente, en caso de haberlo, se eliminará y el autoguardado comenzará a guardar en el nuevo archivo abierto por el usuario.
- Si el usuario crea un nuevo archivo a través de la opción "Nuevo" y no ha guardado los cambios anteriores, los cambios seguirán siendo guardados en el archivo de backup. Sin embargo, si el usuario ha guardado los cambios anteriores, el archivo de backup se eliminará y se creará uno nuevo durante el periodo de autoguardado en caso de que no se haya guardado el nuevo archivo.
- El temporizador de autoguardado se reinicia en los siguientes casos: al abrir la aplicación, al cambiar el tiempo de autoguardado y cada vez que se realiza un autoguardado.
- Si la aplicación se cierra de forma inesperada, al volver a abrirla se abrirá automáticamente el archivo de backup si existe.

3.3.2 Gestión del barrido

En la siguiente figura 3.13 [5] se muestra el diagrama del módulo gestión del barrido.

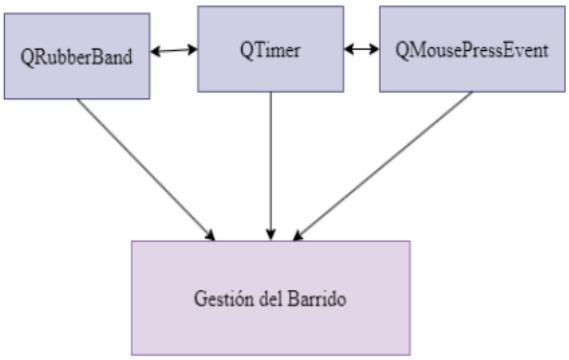


Figura 3.13: Diagrama de gestión del barrido.

Hasta la fecha, PredWin ofrecía dos modos de acceso: el modo directo y el modo de barrido por filas y columnas.

El modo directo es el modo de acceso directo para el ratón. Cuando movemos el ratón por los menús y pinchamos sobre ellos, se van seleccionando las opciones correspondientes. El modo barrido de filas y columnas sigue un patrón secuencial donde primero se exploran todas las filas de los menús disponibles. Una vez que se ha seleccionado una fila en particular, se procede a explorar las columnas correspondientes a esa fila.

3.3.2.1 Barrido lineal

En esta nueva versión, hemos introducido el modo de barrido lineal como alternativa al barrido por filas y columnas.

Esta función es especialmente útil para usuarios que tienen limitaciones en su capacidad para realizar pulsaciones de manera continuada en el tiempo. Por ejemplo, hay usuarios que tienen un número limitado de acciones controladas con precisión (el movimiento de un dedo, el cuello,...) que utilizan para controlar PredWin. Si el número de pulsaciones requeridas escribir son muchas o durante mucho tiempo, puede sobrecargarse, creando fatiga e incluso molestias (tendinitis, etc.) y dificultando la posibilidad de escribir o comunicarse en el futuro.

El modo de barrido lineal ha sido desarrollado específicamente para reducir la cantidad de pulsaciones requeridas en comparación con el modo de barrido por filas y columnas. Esto mejora la accesibilidad y la usabilidad de la aplicación para aquellos que tienen más limitaciones físicas o dificultades motoras, teniendo como contrapartida que se necesita un tiempo mayor para escribir el texto. Al reducir la cantidad de pulsaciones requeridas, se reduce el esfuerzo y se minimiza el riesgo de lesiones o molestias para los usuarios.

Al iniciar la aplicación por primera vez, está establecido el modo directo por defecto. Para cambiar el modo de acceso tendremos que ir al menú de opciones y seleccionar la opción “Modo de barrido” y se nos mostrará la ventana de la figura 3.14.

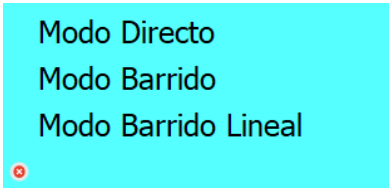


Figura 3.14: Menú para cambiar el modo de barrido.

Si activamos la opción de “Modo Barrido Lineal”, un rectángulo comenzará a resaltar el “Menú Archivo” y se desplazará por el resto de menús superiores de forma secuencial, como se muestra en las figuras 3.15 y 3.16.

Figura 3.15: Barrido lineal en “Menú Archivo”.

Figura 3.16: Barrido lineal en “Menú Oír”

Si el usuario hace clic con el ratón o toca la pantalla en caso de dispositivos táctiles o móviles, se desplegará el menú que esté resaltado en ese momento y se comenzarán a barrer las opciones de dicho menú, como se muestra en la figura 3.17.

Figura 3.17: Barrido lineal del menú desplegable “Edicion”.

Para acceder al teclado, el usuario debe seleccionar la opción “Editar” del menú de edición. Una vez seleccionada esta opción, el sistema comenzará a barrer cada tecla de izquierda a derecha, comenzando desde la primera fila. Esto se ilustra en las figuras 3.18 y 3.19. Si el usuario hace clic o toca la pantalla nuevamente, se llevará a cabo la acción correspondiente a la tecla seleccionada.

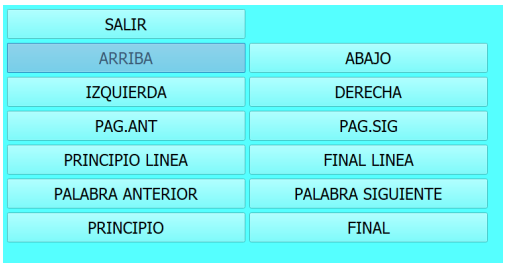
Si el usuario desea volver a barrer los menús superiores, puede hacerlo presionando el botón “MEN” que se encuentra en la última fila del teclado.

Figura 3.18: Barrido lineal del teclado.

Figura 3.19: Barrido lineal del teclado.

También se ha incorporado el barrido lineal a los otros teclados y menús de los que dispone PredWin, que son: teclado numérico o de símbolos, mostrado en la figura 3.20a, al cual se accede mediante el botón "SIG", el menú de selección, mostrado en la figura 3.20b, al que se accede mediante el botón "<->" y el menú de borrado, mostrado en la figura 3.20c, al que se accede mediante el botón "BOR". Estos menús también se barrerán de arriba a abajo y de izquierda a derecha.

(a) Barrido lineal del teclado de símbolos.



(b) Barrido lineal del menú de selección.

(c) Barrido lineal del menú de borrado.

Figura 3.20: Barrido lineal de distintos menús.

3.3.2.2 Implementación del barrido lineal

Para agregar este nuevo modo de acceso a PredWin, hemos tomado como base el modo de barrido existente y hemos realizado una serie de modificaciones para implementarlo. Primero, hemos añadido una variable booleana llamada 'barridoLineal' que se establece en 'true' cuando se selecciona el modo de barrido lineal.

A continuación, hemos creado una versión lineal para cada uno de los barridos existentes. Por ejemplo, para el barrido del teclado teníamos las funciones 'barridoTeclado()' y 'barridoTecladoColumn()', donde 'barridoTeclado()' se encargaba de barrer las filas del teclado y 'barridoTecladoColumn()' se encargaba de barrer las columnas una vez seleccionada la fila. Para agregar el barrido lineal, hemos creado la función 'barridoTecladoLineal()' que realiza el barrido tecla por tecla, como se explicó anteriormente.

Además, hemos tenido que crear una versión lineal de las funciones de 'reset' de estos barridos, que se utilizan cuando el usuario ha entrado en un menú por error y desea salir sin elegir ninguna de las opciones disponibles. Cada función que tenga un método de 'reset' ahora cuenta con su versión lineal que reinicia el barrido (reiniciando todas las variables involucradas en el proceso).

La gestión de pulsaciones es esencial a la hora de que el usuario seleccione la opción deseada. Para ello se utiliza la función 'mousePressEvent(QMouseEvent *event)' que detecta cuando se hace clic con el ratón o se pulsa la pantalla. Hemos añadido las condiciones necesarias para que se traten correctamente las pulsaciones en el modo lineal, en el que haremos uso de las funciones de barrido y 'reset' que hemos mencionado anteriormente.

En resumen, hemos creado una versión lineal de todos los métodos que ya estaban implementados para el modo de barrido existente. Esto nos ha permitido incorporar de manera efectiva el nuevo modo de barrido lineal a PredWin.

3.3.3 Gestión de texto

La figura 3.21 [5] ilustra el diagrama del módulo de gestión de texto. En este módulo, utilizamos la clase 'QTextEdit' para facilitar la edición y visualización de texto en formato enriquecido. La principal mejora de esté módulo es la introducción del teclado predictivo y la introducción de mejoras en la escritura que facilitarán al usuario la escritura rápida y ágil, reduciendo el número de pulsaciones requeridas para introducir el texto.

3.3.3.1 Teclado predictivo

En esta versión de PredWin, una de las grandes implementaciones a destacar es la introducción del teclado predictivo.

El teclado predictivo es una herramienta crucial para personas con discapacidad que no pueden usar un teclado convencional. Estas personas pueden tener limitaciones físicas o dificultades en la coordinación motora, lo que les dificulta escribir de manera precisa o rápida. Este teclado predictivo les proporcionará eficiencia en la comunicación, reducción de errores, facilidad de acceso y autocorrección, mejorando su independencia y calidad de vida.

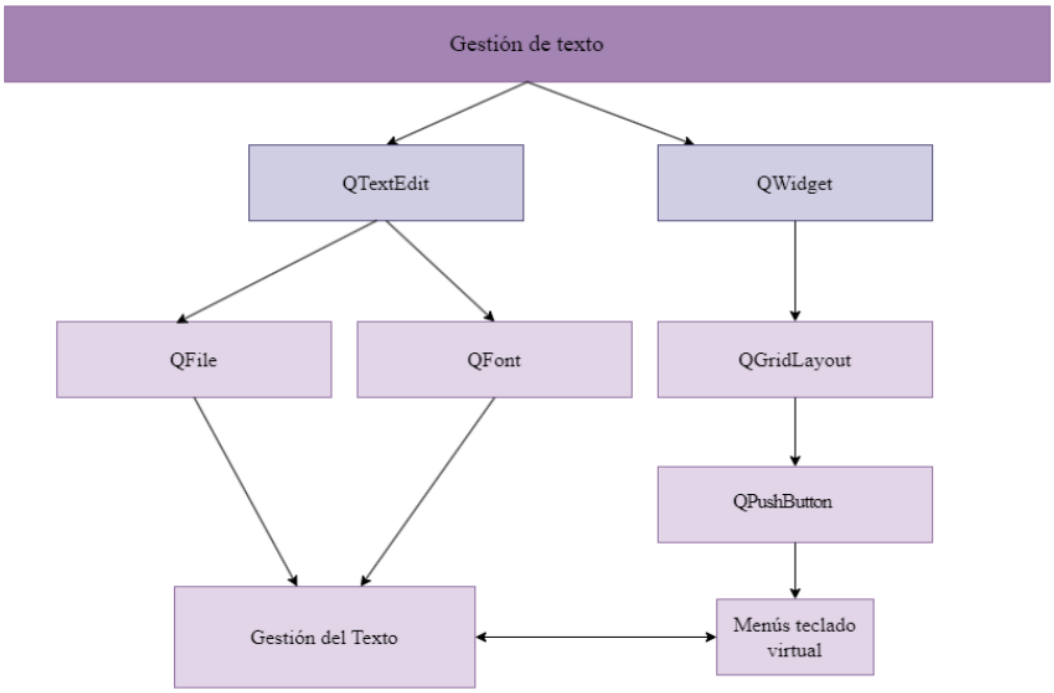


Figura 3.21: Gestión de texto.

Para mostrar el teclado predictivo hemos implementado una serie de botones que mostrarán las sugerencias actualizadas, como ilustra la figura 3.22.

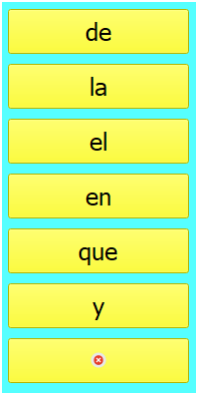


Figura 3.22: Botones que muestran las sugerencias del teclado predictivo.

Estos botones se actualizarán con las palabras sugeridas cada vez que se realice un cambio en el cuadro de texto, y son accesibles desde todos los modos de acceso. En el modo directo, el usuario puede hacer clic o tocar el botón correspondiente a la palabra que desea insertar, y esta se insertará en el cuadro de texto. En el modo de barrido por filas y columnas o el lineal, se accede al teclado predictivo a través de cualquiera de los botones "=>" ubicados en la primera columna del teclado de letras, como se muestra en las figuras 3.23 y 3.24.

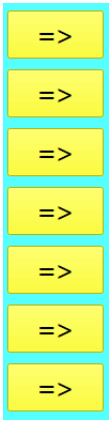


Figura 3.23: Botones que dan acceso al teclado predictivo.

Figura 3.24: Teclado de letras.

El barrido del teclado predictivo actualmente es de arriba a abajo, sin importar qué botón de acceso se presione. Sin embargo, en futuras versiones se planea implementar la adición de botones adicionales a la derecha de las palabras sugeridas para facilitar la escritura de texto, como la predicción de sufijos. Esto implicará una modificación en el barrido para que se desplace únicamente por la fila seleccionada, luego por la palabra sugerida y finalmente por el sufijo sugerido.

3.3.3.2 Implementación del teclado predictivo

El algoritmo de predicción se activa cada vez que se realiza una modificación en el cuadro de texto, utilizando la función 'getCompletions()' [B.6]. Esta función utiliza una expresión regular para identificar el primer carácter de la palabra en proceso. Luego, se busca en el diccionario las seis primeras palabras que comiencen por esa subcadena de la palabra actual. Estas coincidencias se agregan a una lista que se utiliza para mostrar las palabras sugeridas en los botones.

El diccionario utilizado contiene palabras ordenadas por frecuencia de uso y se divide en dos columnas. La columna izquierda contiene las palabras con acento y la columna derecha contiene las mismas palabras sin acento. Esto se hizo para que, incluso si el usuario no utiliza acentos al escribir, las palabras sugeridas en los botones aparezcan correctamente acentuadas. Por ejemplo, el usuario quiere escribir "público", si comienza a escribir "pub-" se mostrará "público" como sugerencia, aunque no haya utilizado el acento. Del mismo modo, si el usuario escribe "púb-", también se mostrará "público" como sugerencia. Cuando se seleccione esa palabra, se insertará en el texto debidamente acentuada, generándose texto correcto y ahorrándose escribir la tilde expresamente.

Para lograr esto, se crearon columnas con palabras tanto acentuadas como sin acentuar en el diccionario, como mencionamos anteriormente. Durante la búsqueda, se busca en ambas columnas y, en caso de coincidencia, siempre se añade a la lista la palabra de la columna que contiene el acento.

La función 'showCompleter()' [B.7] se encarga de invocar la función 'getCompletions()' para obtener las palabras sugeridas y las muestra muestra en los botones correspondientes. Por otro lado, la inserción de palabras se realiza a través de la función 'insertarPrediccion()' [B.8] que está asociada a cada botón. Esta función se encarga de obtener la parte de la palabra que el usuario ha escrito y completarla con la parte que falta de la palabra sugerida, incluyendo el acento si es necesario y manteniendo las mayúsculas que el usuario haya escrito en el fragmento inicial de la palabra..

3.3.3.3 Mejoras en la escritura

En esta versión de PredWin, hemos implementado reglas ortotipográficas [7] para insertar/eliminar caracteres automáticamente antes o después de ciertos símbolos y signos de puntuación, para agilizar el proceso de escritura, reducir el número de pulsaciones necesarias para introducir texto y mejorar la calidad del mismo y su formato. A continuación, detallamos las reglas implementadas:

- Añadir automáticamente un espacio después de introducir una palabra a través del teclado predictivo. Esta funcionalidad agiliza la escritura al evitar que el usuario tenga que insertar manualmente un espacio después de cada palabra, reduciendo las pulsaciones necesarias y mejorando la fluidez en la escritura del texto. Es una medida que tiene impacto en el aumento de la velocidad de escritura, ya que asegura el ahorro de un caracter adicional por cada palabra seleccionada de la lista.
- Si el usuario inserta alguno de los siguientes signos de puntuación: '.', ',', ':', ';', '?', '!', ']', '}', o ')' en el texto, se realizará una verificación para determinar si hay un espacio entre estos caracteres y la palabra anterior. Si se encuentra un espacio, se eliminará, lo que permitirá obtener un mejor formato para el texto sin necesidad de tener que borrarlo manualmente. Esta mejora garantiza que no haya espacios innecesarios antes de los signos de puntuación, lo que contribuye a la corrección del texto.
- Además, después de que el usuario agregue uno de los signos de puntuación anteriores, se insertará automáticamente un espacio, lo que facilitará la inserción de la siguiente palabra, ahorrando pulsaciones y tiempo al usuario.
- Cuando se inserta un punto en el texto, se realizará automáticamente un ajuste de mayúsculas en el siguiente carácter. Esto implica que el carácter posterior al punto será insertado directamente en mayúsculas, lo que facilita la escritura y mejora la presentación del texto. Esta funcionalidad agiliza el proceso de escritura al evitar que el usuario tenga que realizar manualmente el cambio de mayúsculas después de un punto.
- Mejoras en el teclado predictivo: la capacidad de reconocer palabras y proporcionar sugerencias con o sin acento, aunque el usuario no haya utilizado el acento en la escritura. Además, PredWin es capaz de mantener el formato del texto introducido por el usuario, preservando las mayúsculas y minúsculas en el fragmento de palabra al insertar la sugerencia del teclado predictivo.

3.3.4 Implementación de ventanas

En esta sección, veremos cómo implementar ventanas emergentes para las opciones de los distintos menús desplegables. Configuraremos las opciones de estas ventanas para que sean funcionales tanto en el modo directo como con barrido.

Para explicar este procedimiento, utilizaremos como ejemplo la ventana de modificación de la velocidad de la voz. Primero, crearemos una función llamada 'setActionsRate()' [B.13] que se encargará de crear la ventana y añadir las acciones que deseemos que aparezcan. Debemos tener en cuenta que para crear una nueva ventana debemos ir a Interfaz.ui y crear un nuevo "PopUp" y un 'gridLayout', para esto podemos coger cualquiera de los ya creados para las otras ventanas y copiarlo cambiando los nombres como se ilustra en la figura 3.25.

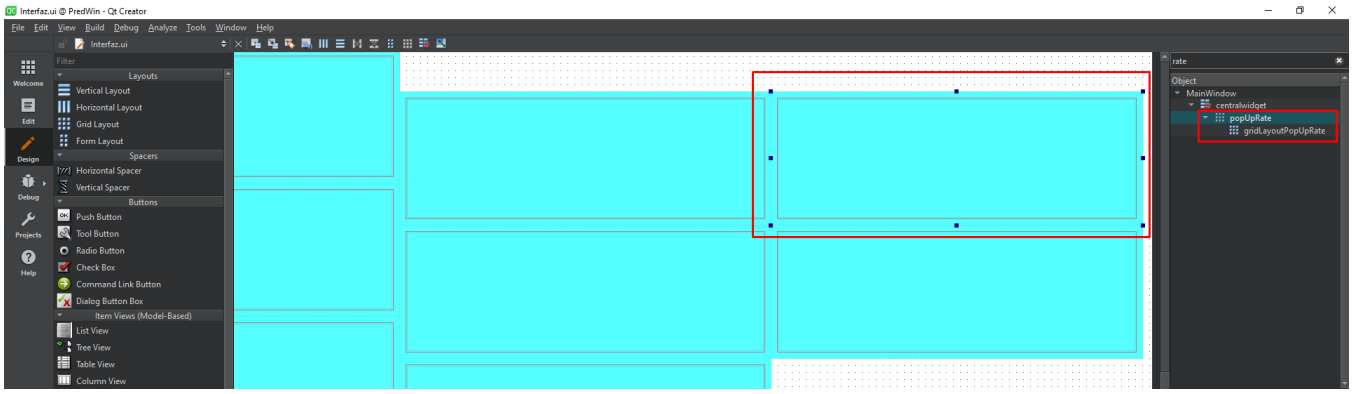


Figura 3.25: GridLayout en Interfaz.ui

Luego, crearemos la función 'm_ajustesRate()' [B.10] que permitirá que la ventana sea barrida. Esta función establecerá todos los parámetros necesarios para que los modos de barrido funcionen correctamente.

Si queremos que la ventana cambie el valor de una variable, como en el caso de la velocidad (que se modifica al pulsar las opciones de subir o bajar velocidad), crearemos la función opcional 'm_ajustesRateActualizar()' [B.11]. Esta función actualizará los valores de las acciones para que aparezcan correctamente cada vez que se presione una opción.

A continuación, crearemos la función 'barridoAjustesRate()' [B.12] que permitirá que la ventana sea barrida en ambos modos de barrido.

Finalmente, nos dirigiremos a la función 'trigger()', que se creará automáticamente al acceder a la Interfaz.ui y hacer clic derecho en la opción que deseamos que despliegue esta ventana. Luego, seleccionaremos "go to slot" y 'triggered()'. En esta función, agregaremos las funciones que hemos creado anteriormente para que puedan ser implementadas tanto en el modo directo como en los modos de barrido [B.13].

Para habilitar que estas opciones sean accesibles en los modos de barrido, debemos realizar modificaciones en la función 'BarridoMenuPrincipal()'. Aquí iremos a la parte donde hemos añadido la opción, en este caso, en el "menú oír", y desplegaremos una serie de condiciones para garantizar el correcto funcionamiento de la nueva ventana en los modos de barrido [B.14].

3.3.5 Modificación del fichero de configuración

PredWin permite cambiar los valores de numerosas variables para adaptarse a las capacidades de usuarios muy diferentes. La configuración para unos usuarios puede estar totalmente contraindicada para otros, por lo que es importante almacenar los valores fijados para futuras sesiones.

La aplicación guarda el estado de algunas variables al cerrarse utilizando las clases 'QCloseEvent' y 'QSettings'. La primera es una clase de tipo evento que se ejecuta al cerrarse la aplicación, y la segunda

clase se encarga de guardar la configuración de algunas variables de la aplicación en un archivo llamado "miapp.ini".

En esta nueva versión hemos tenido que añadir nuevas variables con las nuevas funcionalidades que hemos implementado. Estas nuevas variables son:

- "modoSonido", variable booleana que guarda el estado de activación o desactivación de la lectura de menús.
- "volumen", guarda el volumen del sintetizador de voz.
- "rate", guarda la velocidad del sintetizador de voz.
- "pitch", guarda el tono del sintetizador de voz.
- "modoBarridoLineal", variable booleana que guarda el estado de activación o desactivación del modo de barrido lineal.
- "velBarridoSonido", variable que guarda la velocidad del barrido cuando la lectura de menús está activada.
- "currentVoice", variable que guarda la voz actual del sintetizador de voz.
- "currentFile", variable que guarda el archivo actual con el que esta trabajando el usuario. El valor de esta variable es actualizado cada vez que se abra o guarde un fichero, ya sea mediante guardado manual o guardado automático.
- "minutosGuardadoAuto", guarda la configuración de los minutos establecidos para la ejecución del guardado automático. Estos minutos determinan cada cuánto tiempo se realiza automáticamente la función de autoguardado de la aplicación.

El archivo de configuración se lee al iniciar la aplicación. En la primera ejecución, si el archivo "miapp.ini" no existe, se genera a partir de los valores por defecto de las variables. Este archivo de configuración permite almacenar los valores configurados por el usuario, evitando la necesidad de personalizar los distintos valores cada vez que entramos en la aplicación.

Para cargar estos valores desde el archivo "miapp.ini", utilizaremos la función 'definicionValores(...)' [B.15]. Esta función se encargará de actualizar los valores en la aplicación, de modo que cuando se vuelva a abrir, se cargue la configuración con la que se cerró anteriormente. Además, para que estos valores se guarden, se utilizará la función 'closeEvent(QCloseEvent *event)' [B.16] para guardar los valores antes de cerrar la aplicación.

Capítulo 4

Conclusiones y líneas futuras

4.1 Conclusiones

PredWin es un editor de textos orientado a personas con diversidad funcional que no pueden utilizar el teclado/ratón convencionales. Está siendo objeto de una importante actualización desde sus inicios como un simple editor de texto en entornos ya obsoletos como Windows 3.1 o Windows 2000 [8]. Con la evolución de las tecnologías y la demanda de los usuarios, se está realizado un rediseño para hacerla multiplataforma, pudiendo ser instalada y ejecutada en Windows, Linux, Android e iOS. Este proceso de rediseño es de vital importancia para garantizar la accesibilidad y usabilidad de la aplicación en los dispositivos preferidos por los usuarios en la actualidad.

En este TFG, las incorporaciones que se han realizado sobre la versión existente en Qt son las siguientes:

- La síntesis de voz: Esta nueva capacidad permite aumentar su potencia como comunicador y leer las opciones al ser resaltadas, conveniente cuando los usuarios tienen dificultades de visión. Se han incorporado menús para cambiar los parámetros de la voz (volumen, tono, velocidad, tipo de voz, etc.) y para definir cuando debe utilizarse (para leer los menús o las frases del texto) dependiendo de las necesidades principales del usuario.

- La predicción de palabras: se ha incluido la gestión básica de la predicción de palabras. Mientras se escribe, se muestran en el menú las palabras más frecuentes que empiezan con el fragmento escrito de la palabra en curso y cuando se selecciona una de dichas palabras se inserta en el texto.

- La inserción/borrado automático de caracteres antes o después de la escritura de signos (punto, coma y otros signos de puntuación), automayúscula después de punto, etc., que mejoran tanto la velocidad de escritura como la calidad del texto ya que estará correctamente formateado y se generará con menor número de pulsaciones.

- El barrido lineal, para reducir las pulsaciones necesarias al utilizar el programa, orientado a las personas que tienen dificultades para realizar las pulsaciones por fatiga o dolor.

- La herramienta de autoguardado que guarda los trabajos de manera automática, rápida y segura. En caso de un cierre inesperado, los usuarios podrán recuperar fácilmente lo que habían escrito, evitando la pérdida de información.

- La capacidad de configuración de la aplicación y almacenamiento de los parámetros modificados han mejorado, brindando a los usuarios una mayor flexibilidad. Esta mejora permitirá una experiencia más personalizada y adaptada a las necesidades de cada usuario evitando, además, que sea necesario volver a configurar cada modificación al entrar de nuevo al programa.

Para finalizar, hemos logrado alcanzar el principal objetivo de garantizar que todas las nuevas herramientas añadidas en la versión actual de PredWin sean compatibles con los sistemas operativos Windows, Android y Linux.

4.2 Líneas futuras

Si realizamos una visión global del trabajo realizado, es natural que surjan nuevas ideas y aspectos que deben ser considerados para futuras versiones. La naturaleza del proyecto es evolucionar y adaptarse a las peticiones de los usuarios para mejorar la accesibilidad de PredWin y llegar a más usuarios. A continuación se muestran una serie de propuestas y aspectos a mejorar para que los usuarios tengan una mejor experiencia:

- Aumentar la potencia de la predicción de palabras: En esta versión la predicción está basada en un diccionario general con más de 100.000 palabras ordenadas por frecuencia de uso. El listado de palabras que se muestra está basado solamente en la frecuencia y no tiene en cuenta las palabras anteriores. Sería conveniente incorporar la predicción proveniente de un Large Lenguage Model (LLM) que mejore la calidad de las palabras predichas.

- Agregar diccionarios temáticos. A pesar de la extensión del diccionario general, este puede ser insuficiente y poco efectivo para textos de temas concretos, que contengan términos y expresiones muy específicos. Los diccionarios temáticos vienen a suplir esta carencia. Se crean a partir de textos que traten de los temas más frecuentemente utilizados, que contengan el vocabulario específico.

- Mejora en la comunicación mediante la gestión de expresiones, que permitirá al usuario crear y guardar frases o expresiones personalizadas que podrán ser insertadas directamente en el texto mediante un botón. Al tener la opción de utilizar estas expresiones predefinidas, el usuario ahorrará tiempo y esfuerzo al no tener que escribir repetidamente frases que use frecuentemente.

- Aumentar la seguridad de la aplicación para prevenir posibles pérdidas del fichero: Al utilizar el barrido, existe la posibilidad de cometer errores en pulsaciones y responder de forma errónea a las preguntas del programa (si nos pregunta si borra un fichero o si lo guarda) y que derive en la pérdida de todo el trabajo realizado o parte de él. Por esta razón, agregar nuevas ventanas de confirmación para salir de la aplicación o cuando el usuario decida no borrar, es una mejora importante para brindar una mayor seguridad al usuario.

- Modificar la matriz de letras en modo barrido lineal, para que las letras más frecuentes aparezcan primero. Esta mejora permitirá al usuario una escritura más rápida cuando se utilice dicho barrido.

- Incluir un botón de “OÍR” en la matriz de letras que despliegue las opciones comunicadoras (reproducir texto, leer frase, leer palabra...), para agilizar el proceso de comunicación en la aplicación.

- Agregar un modo de aprendizaje que permita incorporar las letras progresivamente permite enseñar a leer a los niños que utilizan PredWin. En este modo, se ocultarán todas las letras en la matriz de letras, excepto las vocales. A medida que los niños vayan aprendiendo las letras, estas se irán

mostrando gradualmente en sus respectivas posiciones en la matriz de letras. Esto facilitará el proceso de aprendizaje y ayudará a los niños a familiarizarse con las letras y su posición en el teclado de PredWin.

- Incluir la posibilidad de que sea multiusuario, permitiendo configurarlo de forma diferente para varias personas con distintas capacidades en el mismo ordenador/tablet, modificando los distintos parámetros independientemente para cada usuario.
- Poder ejecutar PredWin en dispositivos iOS.

Bibliografía

[1] *Información sobre los SAACs*, https://arasaac.org/aac/es [Último acceso 12/junio/2023].

[2] *Información técnica sobre MegaBee*, https://www.e2l.uk.com/megabee/megabee_technicales.html [Último acceso 12/junio/2023].

[3] *Información sobre la aplicación Speak for Yourself*, https://apps.apple.com/us/app/speak-for-yourself/id482508198 [Último acceso 12/junio/2023].

[4] *Información sobre el comunicador AraBoard*, https://www.tecnoaccesible.net/catalogo/araboard [Último acceso 12/junio/2023].

[5] O. L. Lamkhair, "Implementación multiplataforma de un editor predictivo: PredWin+", Trabajo Fin de Grado, EPS, UAH, Madrid, 2023.

[6] *Documentación de la clase QTextToSpeech 5.15*, https://doc.qt.io/qt-5/qtexttospeech.html [Último acceso 15/junio/2023].

[7] *Información sobre reglas ortotipográficas*, https://blognisaba.wordpress.com/2010/10/10/ortotipografia-coma-punto-y-coma-punto-dos-puntos/ [Último acceso 28/junio/2023].

[8] S. D. González, "ACTUALIZACIÓN DEL EDITOR DE TEXTOS PARA PERSONAS CON DISCAPACIDAD FÍSICA PREDWIN", Trabajo Fin de Carrera, EPS, UAH, Madrid, 2008.

[9] *Información sobre Qt Installer Framework*, https://doc.qt.io/qtinstallerframework/index.html [Último acceso 11/junio/2023].

[10] *Información sobre Qt 5.15.2*, https://www.qt.io/blog/qt-5.15.2-released [Último acceso 11/junio/2023].

[11] *Página oficial de Android Studio*, https://developer.android.com/studio [Último acceso 11/junio/2023].

[12] *Información sobre Gradle 6.3*, https://docs.gradle.org/6.3/release-notes.html [Último acceso 11/junio/2023].

[13] *Información sobre CQtDeployer*, https://wiki.qt.io/CQtDeployer [Último acceso 11/junio/2023].

[14] *Información sobre Microsoft Windows en Wikipedia*, https://es.wikipedia.org/wiki/Microsoft_Windows [Último acceso 11/junio/2023].

[15] *Información sobre GNU/Linux en Wikipedia*, http://es.wikipedia.org/wiki/GNU/Linux [Último acceso 11/junio/2023].

[16] *Información sobre Android en Wikipedia*, https://es.wikipedia.org/wiki/Android [Último acceso 11/junio/2023].

[17] *Página oficial de VirtualBox*, https://www.virtualbox.org/ [Último acceso 28/junio/2023].

Apéndice A

Manual de usuario

A.1 Introducción

PredWin es un editor de textos diseñado especialmente para personas con discapacidad física. Está enfocado en aquellos individuos que enfrentan dificultades en la manipulación del teclado debido a su condición física. En esta nueva versión, comenzamos con las características más básicas y esenciales, y a partir de ellas hemos desarrollado nuevas funcionalidades que mejorarán la experiencia de escritura y comunicación para los usuarios.

A.2 Instalación y ejecución de PredWin

Con esta nueva versión vamos a poder ejecutar PredWin en distintos sistemas operativos. Vamos a explicar el proceso de instalación de cada uno de ellos:

- Windows: para la instalación de PredWin en el sistema operativo Windows, es necesario ejecutar el instalador "PredWinInstaller.exe", para ello pulsamos con el ratón el ejecutable y seguimos los pasos para la instalación del programa:

    1. Se muestra la ventana de bienvenida del instalador, pulsa sobre la opción "Siguiente".

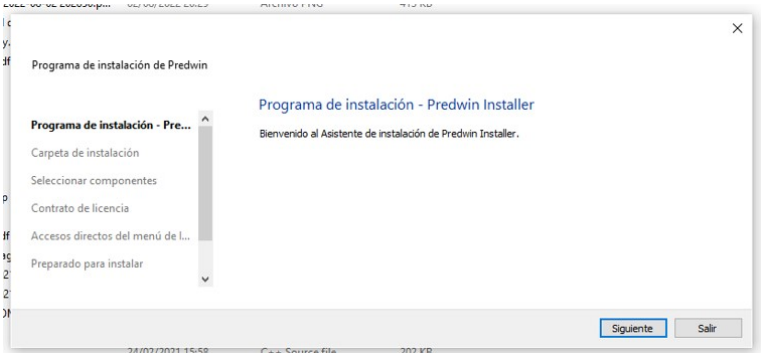


Figura A.1: Ventana de bienvenida del instalador

    2. La siguiente ventana te permite seleccionar la carpeta de instalación de Predwin. Si deseas cambiar el directorio de instalación, pulsa en la opción "Examinar..." para elegir otra carpeta. Si no deseas realizar cambios, pulsa en la opción "Siguiente".

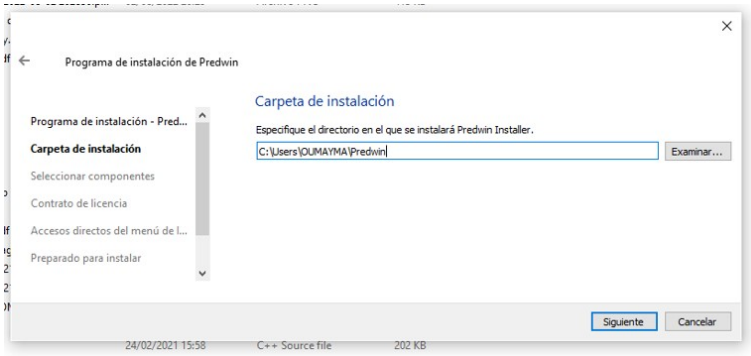


Figura A.2: Ventana para elegir la carpeta de instalación

3. A continuación, se muestra la ventana de selección de componentes. Pulsa en “Siguiente”.

Figura A.3: Ventana de selección de componentes

4. Se muestra la ventana de accesos directos. Pulsa en “Siguiente”.

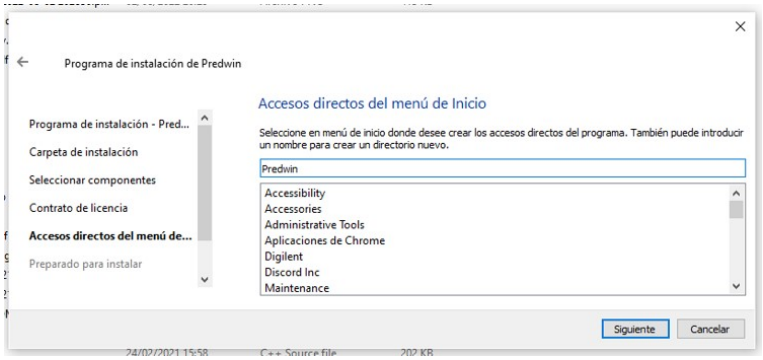


Figura A.4: Ventana de accesos directos

5. Vemos la ventana de instalación. Pulsa en “Instalar”.

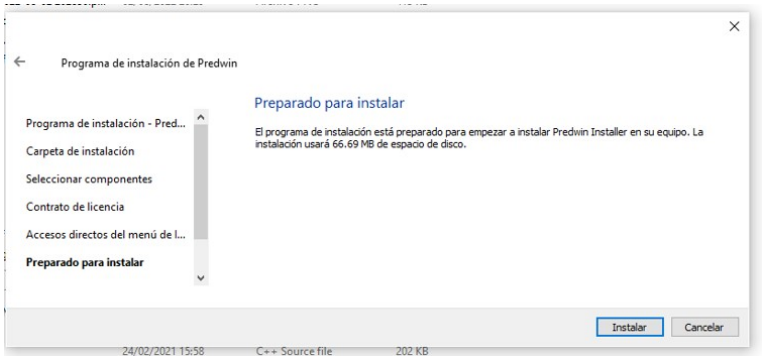


Figura A.5: Ventana de instalación

6. Por ultimo, se muestra la ventana de finalización de la instalación. Pulsa en “Finalizar”.

Figura A.6: Ventana de finalización

Siguiendo los pasos de instalación, el programa estará instalado y listo para ser ejecutado. Para ello, dirígete al escritorio o al directorio de instalación y haz doble clic en el icono de “Predwin”.

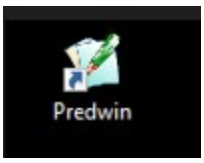


Figura A.7: Ejecutable de PredWin

- Linux: Para instalar PredWin en Linux, el proceso es similar al de Windows, con la única diferencia de que el instalador se llama “PredWinInstaller.run”.

  En Linux, para poder ejecutar el instalador, se deben otorgar permisos de ejecución al instalador. Para ello, se hace clic derecho sobre el instalador, se selecciona “Propiedades” en el menú y en la sección de permisos, se marca la opción “Permitir ejecutar el archivo como programa”.

- Android: para instalar la aplicación de PredWin en tu dispositivo Android, debes descargar el archivo “PredWin.apk” que te proporcionaremos. Una vez descargado, deberás realizar algunas configuraciones en tu teléfono o tablet para permitir la instalación.

  1. Accedes a las descargas de tu dispositivo y pinchas en “PredWin.apk” para que comience la instalación. Verás una ventana emergente que te indica que no se pueden instalar aplicaciones de fuentes desconocidas.

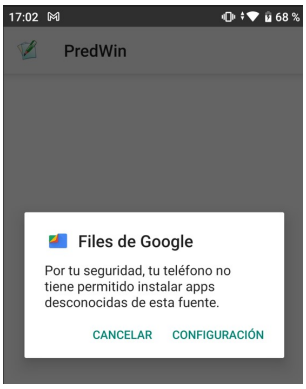


Figura A.8: Bloqueo de instalación

2. Seleccionas la opción de "CONFIGURACIÓN" para que te redirija directamente a los ajustes y puedas activar la instalación de aplicaciones con origen desconocido.

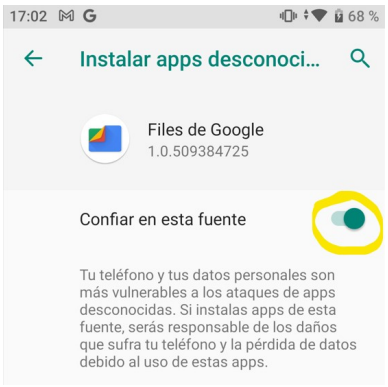


Figura A.9: Activar fuente desconocida

3. Vuelves atrás para continuar con la instalación. Ahora le das clic en "INSTALAR".

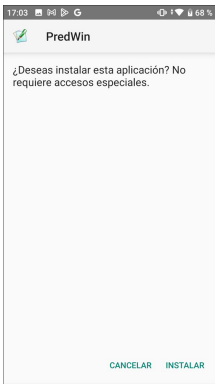


Figura A.10: Instalar

4. Se te mostrará la ventana de confirmación y debes seleccionar “Instalar de todas formas” para continuar con la instalación.

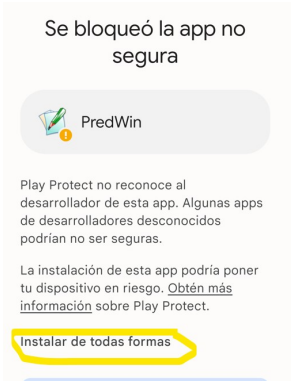


Figura A.11: Ventana de confirmación

5. Finalmente, la aplicación se instala y ahora puedes buscarla en tu dispositivo. Una vez seleccionada, se ejecutará sin problemas.

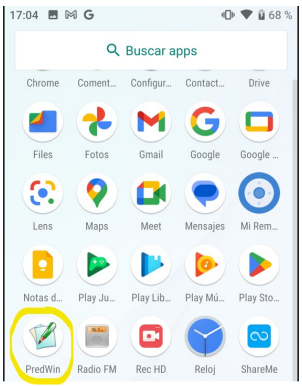


Figura A.12: Predwin Android

A.3 La predicción

Sirve para ayudar a los usuarios a escribir sus textos a mayor velocidad. A medida que se va escribiendo, el programa va mostrando en la parte derecha del menú de caracteres las palabras que estima como más probables para escribirse a continuación.

No es la única ayuda a la escritura, ya que el editor también reduce el número de pulsaciones necesarias para escribir un texto insertando caracteres automáticamente, por ejemplo, después de un punto escribe un espacio y pone la siguiente letra en mayúscula, y al elegir una palabra predicha, inserta directamente un espacio detrás.

A.3.1 Teclado predictivo

La predicción general existe siempre: cuando escribes un carácter, el programa buscará cuáles son las palabras más frecuentes en español que empiecen por lo que se lleva escrito desde el último espacio en blanco o signo de puntuación. Por ejemplo, tras escribir “ele” predecirá: “elecciones”, “electoral”,

"elección" etc. Para ello utilizará un diccionario de palabras ordenadas por frecuencia, proporcionado en la instalación, y que no se cambiarán nunca.

Para acceder al teclado predictivo desde los modos de barrido se debe presionar el botón "=>" que se encuentra en la primera columna del teclado de letras.

A.4 Menús

A.4.1 Menú principal

El menú principal está situado en la parte superior de la ventana. Algunos de sus elementos son típicos de las aplicaciones de edición de texto como "Archivo", "Edición", "Ayuda", mientras que el resto son específicos de PredWin. Las opciones de este menú dan acceso a otros submenús que veremos a continuación.

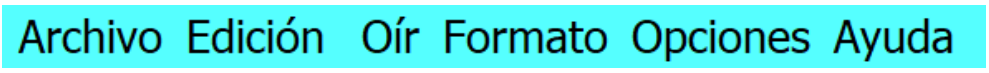


Figura A.13: Menú principal

A.4.2 Menú archivo

A continuación, se describe la funcionalidad de las opciones del menú archivo.

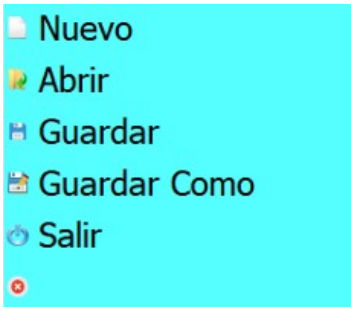


Figura A.14: Menú Archivo

A.4.2.1 Nuevo

Permite crear un documento en blanco. Si el documento actual no ha sido salvado, pedirá confirmación para salvarlo antes de crear el nuevo documento, ya que sólo se puede trabajar con un documento a la vez.

A.4.2.2 Abrir

Abre un documento almacenado en el disco duro. Previamente, se da la opción de guardar el documento actual antes de perderlo. Para seleccionar el documento que se desea abrir se muestra el listado de selección de ficheros.

Como vemos, se nos muestra un total de 5 ficheros que se puedes seleccionar directamente para su apertura, y otra la opción "..." que visualiza otro grupo de ficheros de ese directorio para optimizar el

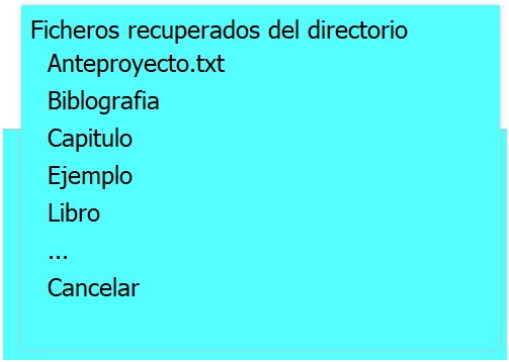


Figura A.15: listado de selección de ficheros

espacio de la ventana. Pulsando sucesivamente sobre "..." navegaremos por todos los ficheros guardados hasta encontrar el que deseamos abrir.

A.4.2.3 Guardar

Graba el fichero actual en el disco duro. En caso de que el fichero no tenga nombre solicita uno.

A.4.2.4 Guardar como

Guarda el fichero actual en el disco duro mostrando una ventana con la opción de poner nombre al fichero.

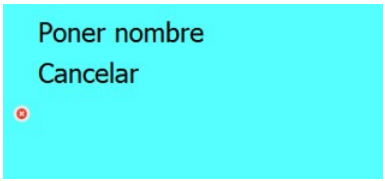


Figura A.16: Guardar como

Cuando se pulsa la opción "Poner nombre", se despliega otra ventana con un área de edición donde escribir el nombre. Podremos escribir el texto usando la matriz de letras o el teclado. Al terminar, en modo barrido seleccionamos la opción "MEN" y el barrido pasa a las opciones "Aceptar" y "Cancelar" del cuadro, con las que se podrá aceptar el nombre escrito o cancelar la operación entera. En modo directo pulsamos la opciones de "Aceptar" o "Cancelar" que tienen el mismo efecto que en modo barrido.

A.4.2.5 Salir

Finaliza la aplicación y cierra la ventana principal. Si el fichero actual no ha sido guardado, se nos da la opción de salvarlo antes de abandonar la aplicación.

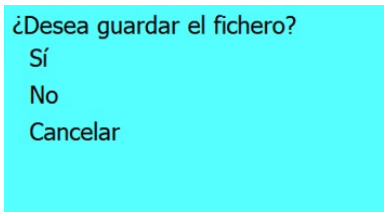


Figura A.17: El cuadro de confirmación de salir

Figura A.18: Menú edición

A.4.3 Menú edición

A.4.3.1 Editar

Entra en el modo de edición. El barrido pasa automáticamente a la matriz de letras, se hace visible el cursor sobre el texto y se habilita el teclado. Para salir de este modo y tener acceso a los menús se usa la opción “MEN” de la última línea de la matriz de letras.

A.4.3.2 Deshacer / Rehacer

Permite deshacer la última acción realizada en la edición de texto. También es posible rehacer lo deshecho, simplemente seleccionando la opción Rehacer.

Figura A.19: Rehacer

A.4.3.3 Seleccionar

Permite seleccionar texto para luego actuar sobre él con los comandos de gestión de bloques “Cortar”, “Copiar”, “Pegar” etc. La selección se realiza en dos fases y es guiada por el sistema.

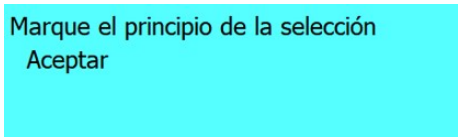


Figura A.20: Inicio de la selección

Primero se advierte de que se va a proceder a marcar el principio de la selección. Inmediatamente se muestra la matriz de movimiento del cursor y se hace visible el mismo. Con esta matriz es posible desplazar el cursor a cualquier parte del texto. También es posible usar el teclado.

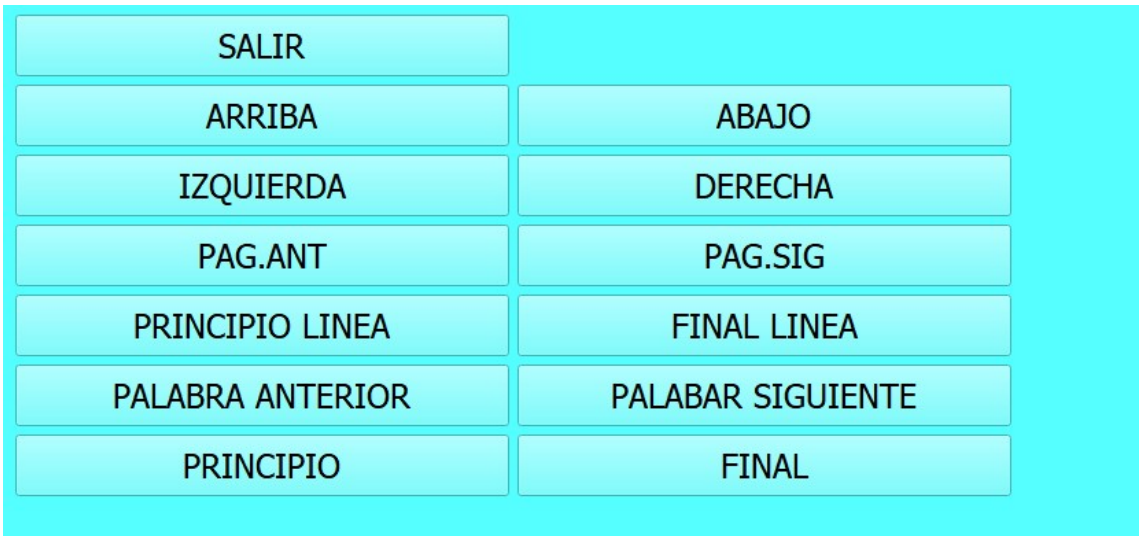


Figura A.21: La matriz de movimiento de cursor

- Arriba: mueve el cursor una línea hacia arriba.
- Abajo: mueve el cursor una línea hacia abajo.
- Izquierda: mueve el cursor un carácter a la izquierda.
- Derecha: mueve el cursor un carácter a la derecha.
- Pag. Ant.: mueve el cursor una página hacia arriba.
- Pag. Sig.: mueve el cursor una página hacia abajo.
- Principio línea: mueve el cursor al principio de la línea.
- Final línea: mueve el cursor al final de la línea.
- Palabra anterior: mueve el cursor al principio de la palabra anterior.
- Palabra siguiente: mueve el cursor al principio de la palabra siguiente.
- Principio: mueve el cursor al principio del documento.

- Final: mueve el cursor al final del documento.

Una vez situado el cursor en la posición deseada, saldremos de la ventana con la opción “Salir”. A continuación seguiremos el mismo proceso para marcar el final de la selección. Al concluir el texto seleccionado quedará resaltado:

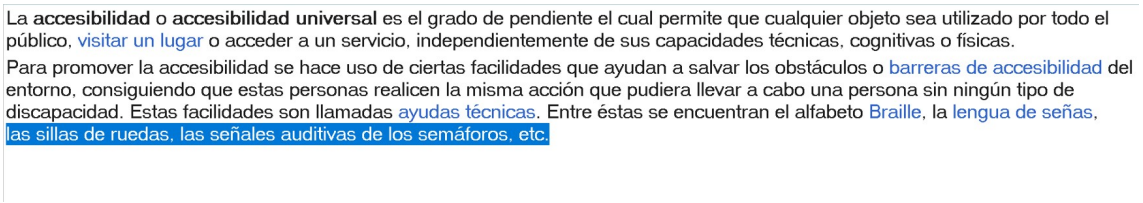


Figura A.22: Texto resaltado

En este momento la selección puede ser manejada como en cualquier otro editor: cortada, copiada, pegada, borrada, reemplazada por la pulsación de una tecla, etc. Para usuarios capaces de utilizar el teclado o el ratón, la selección puede realizarse en cualquier momento en el modo de edición. Con el ratón, pulsando en el principio de la selección, arrastrando y soltando el botón al final de la selección, o con el teclado, con la tecla “Shift” pulsada y usando las teclas de dirección o de movimiento del cursor.

A.4.3.4 Cortar

Con “Cortar” borraremos el texto seleccionado, y lo copiaremos en el portapapeles. Si no hay texto seleccionado, la opción no tendrá efecto.

A.4.3.5 Copiar

Con “Copiar” copiaremos el texto seleccionado en el portapapeles. Si no hay texto seleccionado, la opción no tendrá efecto.

A.4.3.6 Pegar

Con “Pegar” se inserta el texto que contenga el portapapeles en la posición actual del cursor. Si el portapapeles no contiene ningún texto, no se produce ninguna acción. Si tenemos texto seleccionado, la selección es reemplazada por el texto del portapapeles.

A.4.4 Menú Oír

El menú oír incorpora la síntesis de voz. Este submenú permite configurar las características del sonido en sí, (volumen, tipo de voz, etc.) mientras que otros se refieren al uso de la síntesis de voz.

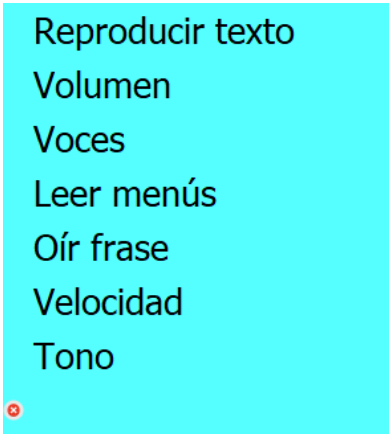


Figura A.23: Menú Oír.

A.4.4.1 Reproducir texto

Reproduce el contenido del cuadro de texto al completo. Esta opción también ha sido añadida a los menús de ayuda para reproducir el contenido de cada tema.

A.4.4.2 Volumen

Permite modificar el volumen de la voz.

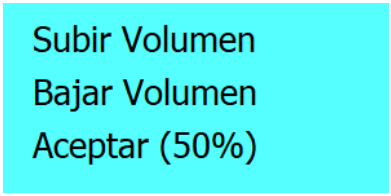


Figura A.24: Ventana de ajuste del volumen.

A.4.4.3 Voces

Permite seleccionar la voz. Las opciones disponibles serán las voces descargadas en el sistema. Al seleccionar la opción "..." se mostrarán más voces adicionales.

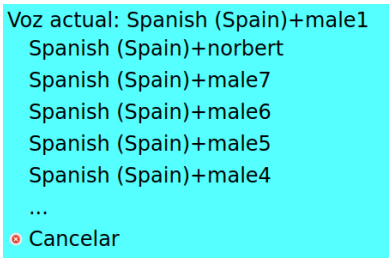


Figura A.25: Ventana de configuración de voz.

A.4.4.4 Leer menús

Esta opción activa o desactiva si queremos que se lean los menús al pasar por estos. Cuando esté activada lee cada opción al ser resaltada, tanto en modo directo, como en modo barrido y modo barrido lineal.

A.4.4.5 Oír frase

Al seleccionar esta opción se reproducirá la frase actual (el contenido entre dos puntos).

A.4.4.6 Velocidad

Permite modificar la velocidad de la voz. Da paso a una nueva ventana en la que se muestra el valor de la velocidad, y se permite aumentarla o reducirla.

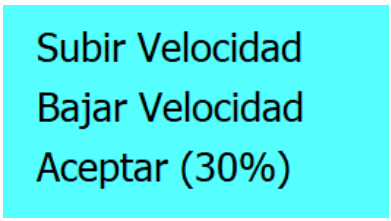


Figura A.26: Ventana de configuración de velocidad de la voz.

A.4.4.7 Tono

Especifica la agudeza o gravedad de la voz. Da paso a una nueva ventana en la que se muestra el valor del tono, y se permite aumentar el valor (más agudo) o reducirlo (más grave).

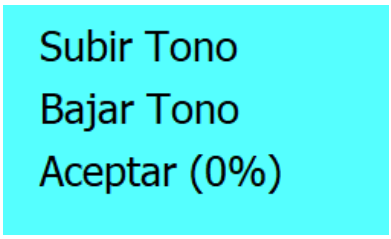


Figura A.27: Ventana de configuración del tono de la voz.

A.4.5 Menú formato

En la figura A.28 se puede ver el menú formato y en la A.29 ejemplos de texto formateado.

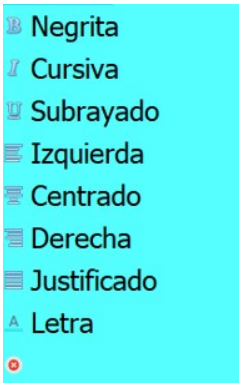


Figura A.28: Menú formato

A.4.5.1 Negrita, Cursiva y Subrayado

Permite resaltar el texto seleccionado o el que se procede a escribir en Negrita, Cursiva o Subrayado. También permite realizar la acción opuesta.

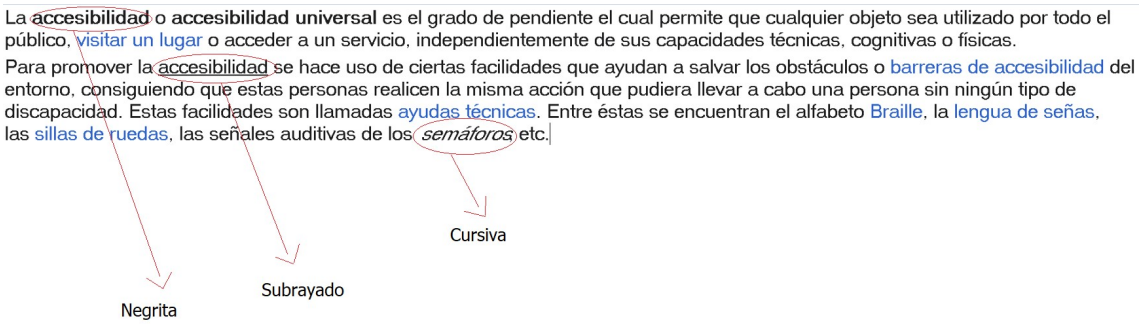


Figura A.29: Ejemplo Negrita, Cursiva y Subrayado

A.4.5.2 Justificar párrafo

Existen cuatro tipos de justificación:

- Izquierda: permite justificar a la izquierda el párrafo sobre el que está situado el cursor.
- Centrado: permite justificar al centro el párrafo sobre el que está situado el cursor.
- Derecha: permite justificar a la derecha el párrafo sobre el que está situado el cursor.
- Justificado: permite justificar a izquierda y derecha el párrafo sobre el que está situado el cursor.

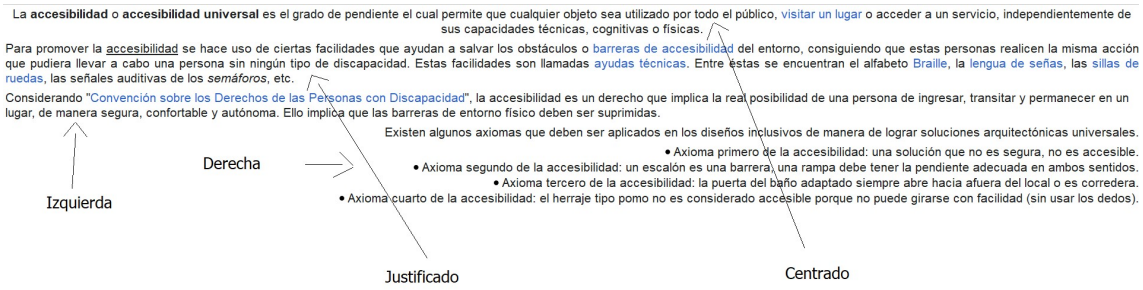


Figura A.30: Justificación de texto

A.4.5.3 Letra

Al seleccionar esta opción aparece un submenú que permite cambiar el Tamaño, la Fuente o el Color del texto seleccionado o del que el usuario procede a escribir.

Figura A.31: Submenú letra

A.4.5.3.1 Tamaño

El usuario puede cambiar el tamaño de la letra que procede a escribir o de los caracteres seleccionados. El tamaño está limitado entre un valor mínimo y máximo.

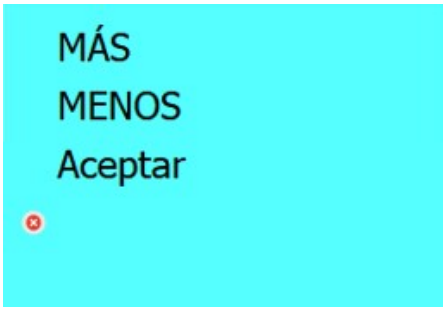


Figura A.32: Tamaño de texto del editor

El aumento / disminución del tamaño del texto va en incrementos / decrementos de 2 puntos. Para evitar problemas a la hora de corregir el tamaño, se ha puesto un límite máximo y mínimo de 72 como máximo y 12 como mínimo.

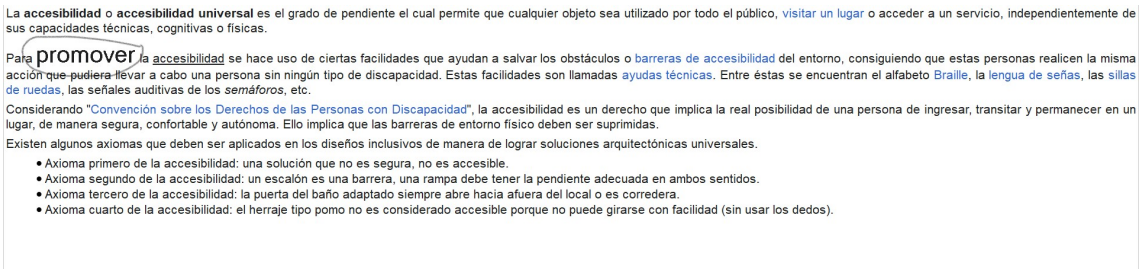


Figura A.33: Texto que muestra el tamaño seleccionado

A.4.5.3.2 Fuente

Cambia la fuente del texto seleccionado o del que se proceda a escribir. A la hora de seleccionar la opción Fuente, se muestran las siguientes 5 opciones:

Figura A.34: Menú para elegir el tipo de fuente a emplear

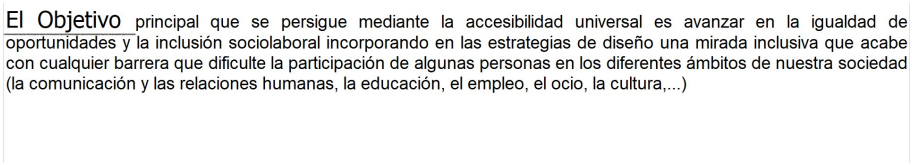


Figura A.35: Ejemplo, tipo de letra cambiado en una parte del escrito

A.4.5.3.3 Color

Para la opción del color sucede lo mismo: Al seleccionar la opción de Color, aparece un menú como el de la siguiente figura:

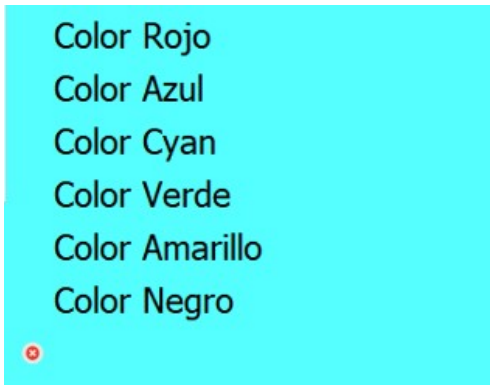


Figura A.36: Menú selección de color para la fuente actual

Al seleccionar cualquiera de los colores por defecto se aplicará el cambio sobre el texto seleccionado o el que se proceda a escribir.

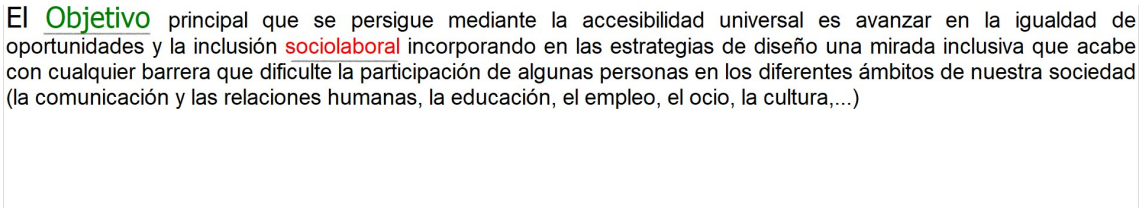


Figura A.37: Ejemplo, cambio de color de la letra en una parte del escrito

A.4.6 Menú opciones

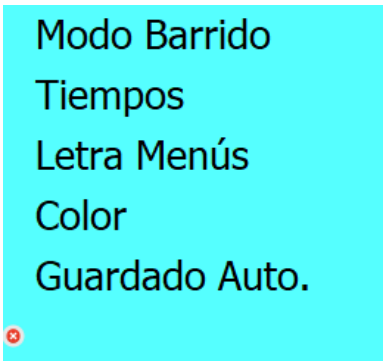


Figura A.38: Menú opciones

En este menú se configuran las diferentes opciones del programa. La mayoría de ellas tienen un efecto inmediato sobre la aplicación. Los valores de las opciones son almacenados en el fichero "miapp.ini", que se detalla al final del manual.

A.4.6.1 Modo barrido

Permite seleccionar el modo de acceso de la aplicación. Existen tres modos: directo, barrido por filas y columnas, y barrido lineal.

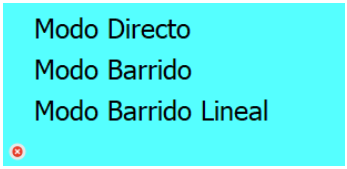


Figura A.39: Modos de barrido

A.4.6.1.1 Modo Directo

Es el modo de acceso directo para el ratón. Cuando movemos el ratón por los menús y pinchamos sobre ellos, se van seleccionando las opciones correspondientes.

A.4.6.1.2 Modo Barrido Filas/Columnas

Este modo de barrido es el convencional, de tal manera que primero se barren las filas de los menús y, tras seleccionar una de ellas, se barren las columnas de esa fila.

A.4.6.1.3 Modo Barrido lineal

En este modo se barren todas las opciones de los diferentes menús de forma lineal, esto es, que se barran todas las opciones una a una.

A.4.6.2 Tiempos

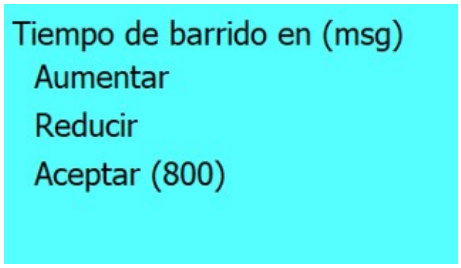


Figura A.40: La ventana de modificar tiempos

En la primera línea se indica que tiempo estamos modificando y las unidades en que se mide. En la 2ª y 3ª se dan las opciones de aumentar o reducir el valor actual y en la 4ª la opción de aceptar y se muestra el valor actual. Es el tiempo que tarda en desplazarse la selección en el barrido. El rango varía entre los 500 y los 3000 milisegundos. Es el tiempo más importante de configurar correctamente, ya que un barrido demasiado lento es desesperante y uno demasiado rápido provoca errores en la selección. Es recomendable empezar con un tiempo grande e ir reduciéndolo según ganemos en precisión por el uso.

A.4.6.3 Letra Menús

Con esta opción se puede especificar el tamaño que se usará para los elementos de las matrices y menús. La mayoría de las aplicaciones no incluyen esta funcionalidad. La razón de incluirla aquí es ayudar a las personas con problemas de visión. La ventana que aparece para seleccionar el tamaño es la misma que la que aparece dentro de la opción Letra del menú principal Formato, con la diferencia que en este caso las modificaciones se aplican a la letra de los menús.

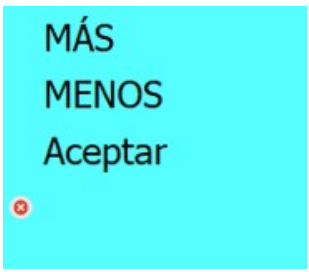


Figura A.41: Cambio del tamaño de una fuente

A.4.6.4 Color

El sistema permite cambiar el color del fondo de las matrices y los menús. Utilizar contrastes de colores fuertes y tipos de letras grandes puede servir de ayuda a las personas con dificultades de visión.

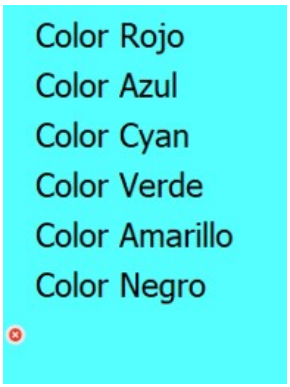


Figura A.42: La ventana de selección de colores

Figura A.43: Ejemplo de cambio de color

A.4.6.5 Autoguardado

El sistema realiza una copia de seguridad del documento actual automáticamente cada cierto tiempo, que se configura desde este menú. Los valores varían entre 0 y 60 minutos. Un tiempo de 0 minutos desactiva la copia de seguridad. El cambio tiene lugar cuando confirmas el tiempo de autoguardado. La copia de seguridad es guardada con el nombre de PredWinBackupNew, en el directorio de textos. En caso de una salida anómala del programa (como un corte de corriente), la siguiente vez que se arranca se abre automáticamente el fichero y el usuario tiene la opción de salvarlo.

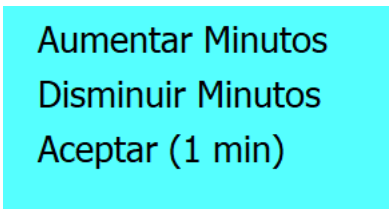


Figura A.44: Ventana de ajustes del autoguardado

A.4.7 Menú ayuda

Esta opción, como su nombre indica nos presenta un diálogo de ayuda para los usuarios que así lo deseen sobre el manejo y funcionamiento del programa. Da lugar a un submenú que nos permite escoger los temas: “Introducción”, “Matriz de letras”, “Menú principal” y “Acerca de”.

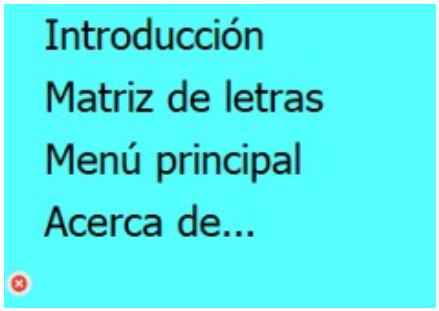


Figura A.45: Menú de selección de temas de Ayuda

A.5 Matrices

Son cuatro las matrices disponibles: caracteres, signos y números, movimiento del cursor y borrado. Las matrices están organizadas en filas y columnas, y el barrido se realiza primero por filas y luego por columnas, de modo que para acceder a un elemento deberemos seleccionar primero la fila y luego el elemento.

A.5.1 La matriz de caracteres

=>	<-	ESP	Enter	∘	<->	BOR	SIG	de
=>	E	A	I	∘	R	T	B	la
=>	O	S	D	∘	U	Q	V	el
=>	N	L	M	∘	G	Y	Z	en
=>	C	P	F	∘	H	J	K	que
=>	´	,	.	∘	X	Ñ	W	y
=>	MEN	MAY	∘					∘

Figura A.46: La matriz de caracteres

La matriz de caracteres es la más utilizada. Contiene el alfabeto español, los signos de puntuación más utilizados, puntos de acceso a las otras tres matrices y acceso a la barra de menús. La disposición de los elementos en la matriz no es aleatoria. Responde a estudios sobre la frecuencia con que son usados los caracteres en el idioma castellano. Los caracteres más usados son dispuestos en las zonas de más rápido acceso (cerca del ángulo superior izquierdo).

A.5.1.1 Elementos de la matriz de caracteres

- Es un símbolo de escape de barrido cuando nos hemos equivocado y hemos seleccionado una línea: seleccionaremos este símbolo, y saldremos de ella sin necesidad de elegir una opción y luego deshacer su efecto.

- Borra el carácter anterior del texto. Produce el mismo efecto que la tecla “back delete” del teclado.

- Inserta un espacio en blanco en el texto. Equivalente a la barra espaciadora del teclado.

- Inserta un retorno de carro en el texto, igual que la tecla “intro” del teclado.

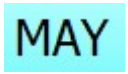
- Da acceso a la matriz de movimiento del cursor. Dicha matriz se trata más adelante en este manual.

- Da acceso a la matriz de borrado. Dicha matriz se trata más adelante en este manual.

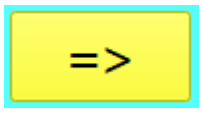
- Da acceso a la matriz de signos y números. Dicha matriz se trata más adelante en este manual.

- Acento, funciona igual que en el teclado. Es decir al pulsarlo no ocurre nada aparentemente, pero el siguiente carácter aparece acentuado.

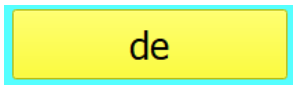
- Sale del modo de edición y pasa el barrido al menú principal. Al salir del modo de edición, el cursor desaparece del texto y el teclado queda inhabilitado para la introducción de texto.



- Cambia el estado del bloqueo de mayúsculas, exactamente igual que la tecla “bloq. mayús.” del teclado. Al cambiar, toda la matriz cambia a mayúsculas / minúsculas, de modo que un carácter siempre es escrito en el texto tal y como aparece en la matriz.



- En el modo de barrido por filas y columnas o en el modo de barrido lineal, para acceder al teclado predictivo, el usuario deberá hacerlo a través de cualquiera de estos botones ubicados en la primera columna del teclado de letras.



- Inserta en el cuadro de texto la palabra que aparece en el botón, preservando las mayúsculas y minúsculas que el usuario haya escrito previamente.

A.5.2 La matriz de signos y números

Se accede a través del elemento “SIG” de la matriz de caracteres, y se sale con el elemento “SALIR”. Contiene signos de puntuación, números y caracteres matemáticos así como ciertos caracteres especiales propios de otros idiomas.

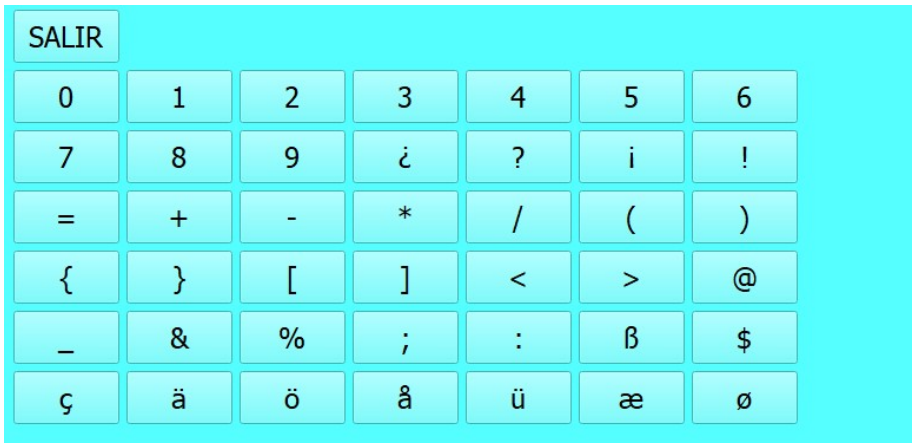


Figura A.47: La matriz de signos

A.5.3 La matriz de movimiento del cursor

Se accede a ella a través del elemento "<->" de la matriz de caracteres, y se sale a través del elemento "SALIR". Contiene comandos de movimiento del cursor equivalentes a los de un editor normal. Estos comandos también pueden ser ejecutados a través de combinaciones del teclado.

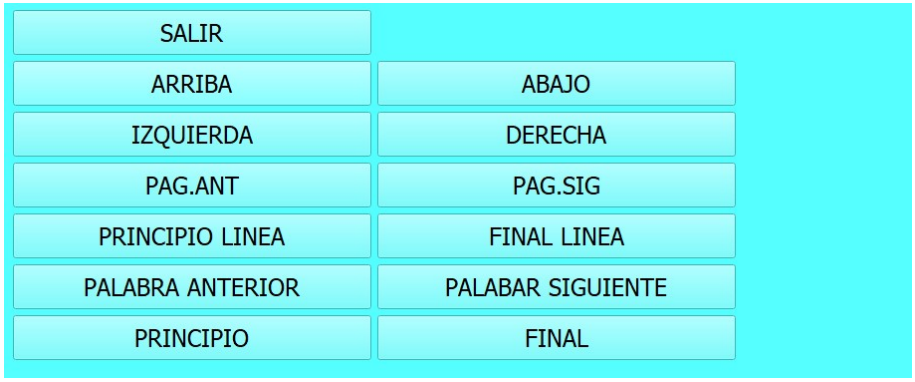


Figura A.48: La matriz de movimiento del cursor

- Arriba: mueve el cursor una línea hacia arriba.
- Abajo: mueve el cursor una línea hacia abajo.
- Izquierda: mueve el cursor un carácter a la izquierda.
- Derecha: mueve el cursor un carácter a la derecha.
- Pag. Ant.: mueve el cursor una página hacia arriba.
- Pag. Sig.: mueve el cursor una página hacia abajo.
- Principio línea: mueve el cursor al principio de la línea.
- Final línea: mueve el cursor al final de la línea.
- Palabra anterior: mueve el cursor al principio de la palabra anterior.
- Palabra siguiente: mueve el cursor al principio de la palabra siguiente.

- Principio: mueve el cursor al principio del documento.
- Final: mueve el cursor al final del documento.

A.5.4 Matriz de borrado

Se accede a través del elemento “BOR” de la matriz de caracteres, y se sale con el elemento “SALIR”. Contiene cinco comandos que permiten efectuar distintos tipos de borrado sobre el documento, sin tener que seleccionar primero el texto y después suprimirlo. La opción de “borrar parrafo” es nueva en esta versión de Predwin.

Figura A.49: La matriz de borrado

A.6 Configuración

A.6.1 El fichero de inicialización

La configuración de algunas variables de la aplicación en un archivo llamado “miapp.ini”.

La variables que guardamos en este documento son las siguientes:

1. “color”, guarda el color del fondo de los menús
2. “activo”, variable booleana que guarda si el modo barrido esta activado o no.
3. “tMenus”, guarda el tamaño de los menús de la aplicación
4. “tiempos”, guarda la velocidad del barrido.
5. “fuenteletra”, guarda la fuente del texto.
6. “fuenteletraTam”, guarda el tamaño de la fuente del texto.
7. “ColorLetra”, guarda el color de la fuente del texto.
8. “modoSonido”, variable booleana que guarda el estado de activación o desactivación de la lectura de menús.
9. “volumen”, guarda el volumen del sintetizador de voz.
10. “rate”, guarda la velocidad del sintetizador de voz.
11. “pitch”, guarda el tono del sintetizador de voz.
12. “modoBarridoLineal”, variable booleana que guarda el estado de activación o desactivación del modo de barrido lineal.
13. “velBarridoSonido”, variable que guarda la velocidad del barrido cuando la lectura de menús está activada.
14. “currentVoice”, variable que guarda la voz actual del sintetizador de voz.
15. “currentFile”, variable que guarda el archivo actual con el que esta trabajando el usuario.
16. “minutosGuardadoAuto”, guarda la configuración de los minutos establecidos para la ejecución del guardado automático. Estos minutos determinan cada cuánto tiempo se realiza automáticamente la función de autoguardado de la aplicación.

Apéndice B

Anexo de códigos fuente

B.1 Ampliación de PredWin

B.1.1 Sintetizador de voz

Listado B.1: Función 'resetHovVariables()'

```
void MainWindow::resetHovVariables() {
    //Se crea un vector con todas las variables hov
    hovVariables = {
            &hovNuevo, &hovAbrir, &hovGuardar, &hovGuardarC,
            &hovImprimir, &hovSalir,
            &hovEditar, &hovSeleccionar, &hovDeshacer,
            &hovCortar, &hovCopiar,
            &hovPegar, &hovReproducir, &hovAjustesVol,
            &hovSonido, &hovNegrita,
            &hovCursiva, &hovSubrayado, &hovIzquierda, &hovCentrado, &hovDerecha,
            &hovSubrayado, &hovJustificado, &hovLetra, &hovModo, &hovTiempos,
            &hovLetraMenus, &hovColor, &hovIntroduccion, &hovMatriz,
            &hovMenuPrincipal, &hovAcerca, &hovCambiarVoz, &hovGuardadoAuto,
            &hovFrase, &hovRate, &hovPitch
        };

        // Ponemos todas las variables a false
        for (bool* hovVariable : hovVariables) {
            *hovVariable = false;
        }
}
```

Listado B.2: Ejemplo de código de la función 'hovered()' de un 'QAction()'

```
void MainWindow::on_actionNuevo_hovered()
{
    //Comprobamos si esta activado la lectura de menus,
    //si esta desactivado el modo barrido
    //y si la variable hovNuevo es false
    if (sonidoMenus && !activo && !hovNuevo) {
        //Ponemos todas las hov variables a false
        resetHovVariables();
        //Marcamos que hovNuevo esta a true
        hovNuevo = true;
        //Leemos nuevo
        m_speech->say("Nuevo");
    }
}

 //Así para cada opción del menu desplegable
```

Listado B.3: Función 'actionHovered()'

```
// Implementación del slot para manejar las acciones
void MainWindow::actionHovered()
{
    //Si la lectura de menus esta activa
    // y el barrido esta desactivado
    if (sonidoMenus && !activo)
    {
        //Obtenemos la action
        QAction* action = qobject_cast<QAction*>(sender());
        //Si existe action y la variable hov está a false
        if (action && !actionReadMap[action])
        {
            // Restablecer el valor de los booleanos de todas las acciones
            for (auto it = actionReadMap.begin(); it != actionReadMap.end(); ++it)
            {
                if (it.key() != action)
                {
                    it.value() = false;
                }
            }
            //Ponemos el valor de la action actual a true
            actionReadMap[action] = true;

            //Leemos el texto
            if(action->text()!=""){
                 m_speech->say(action->text());
            }else{
                 m_speech->say("cancelar");
            }
        }
    }
}
```

Listado B.4: Solución a la interrupción de "Reproducir texto" debido a la lectura de menús en los modos de barrido.

```
m_speech->stop(); //paramos el TTS del barrido
ui->actionReproducir_texto->trigger(); //Llamamos a la funcion para reproducir texto
if(sonidoMenus){

    //Desactivamos temporalmente la lectura de menus
    sonidoMenus=false;

    //Esperar a que el texttospeech termine de hablar para poder volver a barrer
    QEventLoop loop;
    QTimer timer2;

    //Cuando la señal timeout del temporizador timer2 se emite,
    //se comprobará el if
    QObject::connect(&timer2, &QTimer::timeout, &loop, [&](){

        //Esperamos a que el lector este disponible
        if (m_speech2->state() == QTextToSpeech::State::Ready) {
            qDebug()<<"estado ready";
            loop.quit();
        }
    });
    qDebug()<<"estado speaking";
    timer2.start(100); // Revisar el estado cada 100 milisegundos
    loop.exec();

    //Volvemos a habilitar la lectura de menús
    sonidoMenus=true;
}
```

Listado B.5: Función 'showMoreVoices()'

```
void MainWindow::showMoreVoices()
{
    ...Resto del código...

    //Lista con las voces disponibles
    QVector<QVoice> availableVoices = m_speech->availableVoices();
    QList<QVoice> voiceList = QList<QVoice>::fromVector(availableVoices);


    //Numero de voces
    int totalVoices = voiceList.count();
    if (totalVoices <= 0) {
        return;
    }

    // Crear una acción para cada voz disponible
    int maxVoicesToShow = 5; // Mostrar solo las siguientes 5 voces

    //Recorremos la lista de voces
    for (int i = m_startIndex; i < totalVoices && i < m_startIndex
    + maxVoicesToShow; i++) {
        QAction* action = new QAction(voiceList.at(i).name(), this);
        //Agregamos la voz como action
        menuCambiarVoz->addAction(action);
        //La conectamos a la función modoVoz
        connect(action, &QAction::triggered, this, [this, i]() { modoVoz(i); });
    }

    //Si hay mas de 5 voces
    if (m_startIndex + maxVoicesToShow < totalVoices) {

        //Agregamos la action ...
        QAction* moreVoicesAction = new QAction("...", this);
        menuCambiarVoz->addAction(moreVoicesAction);

        //Conectamos la action ... a showMoreVoices
        connect(moreVoicesAction, &QAction::triggered, this, [this]() {
        showMoreVoices(); });

        // Actualizar el valor de m_startIndex
        m_startIndex += maxVoicesToShow;
    }

    ...Resto de código

}
```

B.1.2 Predicción de palabras

Listado B.6: Función 'getCompletions()'

```
QStringList MainWindow::getCompletions()
{
    //Creamos la lista de sugerencias
    QStringList completions;

    QString currentText = ui->textEdit->toPlainText();
    int cursorPos = ui->textEdit->textCursor().position();

    // Busca hacia atrás desde el cursor en busca del separador más cercano
    int start = currentText.lastIndexOf(QRegExp("[␣\\n¡¿({\\[]"),
    cursorPos - 1) + 1;

    if (start < 0) start = 0;

    // Obtiene la subcadena correspondiente a la palabra actual
    QString currentWord = currentText.mid(start, cursorPos - start);

    int cnt = 0;

    // Busca coincidencias en el diccionario
    for (const Diccionario& entrada : diccionario) {
        if (entrada.palabraSinAcento.startsWith(currentWord, Qt::CaseInsensitive)
        || entrada.palabraAcento.startsWith(currentWord, Qt::CaseInsensitive)) {
            if (cnt <= 5) {

                //Añadimos la palabra de la columna con acento
                completions.append(entrada.palabraAcento);

                cnt++;
                // Contamos el número de sugerencias
                // para solo barrer las necesarias
                numPred = cnt;
            }
        }
    }

    qDebug() << completions;
    return completions;
}
```

Listado B.7: Función 'showCompleter()'

```
void MainWindow::showCompleter()
{
    //obtiene las sugerencias
    QStringList predicciones = getCompletions();
    ui->textEdit->setFocus();

    //Si estas en el teclado de letras
    if(menuTecladoDes==false && menuTecladoBor==false && menuTecladoSig==false){
        int length = predicciones.length();

        for (int i = 0; i < 6; i++) {
            QPushButton* button = nullptr;
            switch (i) {
                //Obtiene el boton al que añadir el texto
                case 0:
                    button = buttonPred0;
                    break;
                case 1:
                    button = buttonPred1;
                    break;
                case 2:
                    button = buttonPred2;
                    break;
                case 3:
                    button = buttonPred3;
                    break;
                case 4:
                    button = buttonPred4;
                    break;
                case 5:
                    button = buttonPred5;
                    break;
                default:
                    break;
            }

            if (i < length) {
                //Asigna el texto a cada botón
                button->setText(predicciones[i]);
            } else {
                //Si no existe sugerencia
                button->setText("-");
            }
        }
    }
}
```

Listado B.8: Función para insertar la palabra sugerida en el texto

```
void MainWindow::insertarPrediccion0(){

    //Obtenemos la palabra sugerida
    QString palabraSugerida = buttonPred0->text();

    //Obtenemos el cursor
    QTextCursor cursor = ui->textEdit->textCursor();

    //Obtenemos el texto del cuadro de texto
    QString texto = ui->textEdit->toPlainText();

    //Encontramos la posición del cursor al inicio de la última palabra completa
    int posInicio = texto.lastIndexOf(QRegExp("[␣¡¿\\[{(]"),
    cursor.position() - 1) + 1;
    // obtener la última palabra completa
    QString palabraActual = texto.mid(posInicio, cursor.position() - posInicio);

    //Obtenemos el cacho de palabra restante de la palabra sugerida
    QString palabraRestante = palabraSugerida.mid(palabraActual.length());

    //Eliminamos la palabra actual
    cursor.setPosition(posInicio);
    cursor.setPosition(cursor.position() + palabraActual.length(),
    QTextCursor::KeepAnchor);

    cursor.removeSelectedText();

    // insertar la palabra sugerida respetando las mayúsculas y minúsculas
    QString nuevaPalabra;
    for (int i = 0; i < palabraActual.length(); i++) {
        QChar cActual = palabraActual.at(i);
        //Respetamos mayusculas y minusculas
        if (cActual.isUpper()) {
            nuevaPalabra += palabraSugerida.at(i).toUpper();
        } else {
            nuevaPalabra += palabraSugerida.at(i).toLower();
        }
    }

    //Insertarmos la palabra nueva
    nuevaPalabra += palabraRestante;
    cursor.insertText(nuevaPalabra + "␣");

    //Restablecer el foco en el QTextEdit
    ui->textEdit->setFocus();
}
```

B.2 Implementación de ventanas

Listado B.9: Función 'setActionsRate()'

```
void MainWindow::setActionsRate()
{
    //Añade tu ventana en closeMenu()
     closeMenu();

     //Crear ventana
     menuCambiarRate = new QMenu(this);

     //booleano para condicion en closeMenu()
     menuAjustesRate = true;

     //Acciones
     QString velocidad = QString::number((m_speech->rate())*100);
     subirVel = new QAction(tr("Subir Velocidad"),this);
     bajarVel = new QAction(tr("Bajar Velocidad"),this);
     menuprincipal = new QAction("Aceptar ("+velocidad+"%)",this);

    //Añadir acciones
     menuCambiarRate->addAction(subirVel);
     menuCambiarRate->addAction(bajarVel);
     menuCambiarRate->addAction(menuprincipal);

     //Mostrar ventana
     menuCambiarRate->show();

     //Ponemos el popUp y gridLayout que hemos creado.
     ui->popUpRate->setGeometry(popupCentralSize(menuCambiarRate,fuenteMenus));
     ui->popUpRate->setHidden(false);
     ui->popUpRate->setEnabled(true);
     ui->gridLayoutPopUpRate->addWidget(menuCambiarRate);

}
```

Listado B.10: Función 'm_ajustesRate()'

```
void MainWindow::m_ajustesRate()
{
    //Desconectamos el barrido del menu principal
    timer.stop();
    timer.disconnect(&timer, SIGNAL(timeout()), this, SLOT(BarridoSubMenu()));

    //Deshabilitamos el resaltador del menu principal
    Highlight->~QRubberBand();
    qDebug()<<"m_ajustesrate"<<p_menuprincipal;

    //Añadimos las acciones
    setActionsRate();

    //Creamos el resaltador para la nueva ventana
    Highlight = new QRubberBand(QRubberBand::Line,menuCambiarRate);

    //guardamos las acciones en esta variable
    ajustesRateHijos =  menuCambiarRate->actions();

    //mostramos la ventana
    menuCambiarRate->show();

    //conectamos el barrido al barrido de la ventana
    connect(&timer,SIGNAL(timeout()), this, SLOT(barridoAjustesRate()));
    connect(&timer, SIGNAL(timeout()), this, SLOT(update()));

    //comenzamos el barrido
    timer.start(velBarrido);

    //booleano que indica que la ventana esta abierta
    popUpAjustesRate = true;
}
```

Listado B.11: Función 'm_ajustesRateActualizar()'

```
void MainWindow::m_ajustesRateActualizar()
{
    //Desconectamos el barrido de la ventana
    timer.stop();
    timer.disconnect(&timer, SIGNAL(timeout()), this, SLOT(barridoAjustesRate()));

    //Volvemos a añadir las acciones
    setActionsRate();

    //Creamos el resaltador
    Highlight = new QRubberBand(QRubberBand::Line,menuCambiarRate);

    //guardamos las aciones en esta variable
    ajustesRateHijos =  menuCambiarRate->actions();

    //reconectamos el barrido
    connect(&timer,SIGNAL(timeout()), this, SLOT(barridoAjustesRate()));
    connect(&timer, SIGNAL(timeout()), this, SLOT(update()));
    timer.start(velBarrido);

}
```

Listado B.12: Función 'barridoAjustesRate()'

```
void MainWindow::barridoAjustesRate(){
    //Ajustamos el resaltador a las dimensiones de la ventana
    //Se debe cambiar la variable de los Hijos y el menu
    Highlight->move(menuCambiarRate->actionGeometry(
    ajustesRateHijos.at(botones)).x(),menuCambiarRate->
    actionGeometry(ajustesRateHijos.at(botones)).y());

    Highlight->setGeometry(menuCambiarRate->actionGeometry(
    ajustesRateHijos.at(botones)));

    Highlight->show();

    //Condicion por si esta habilitada la lectura de menus
    if(sonidoMenus){
        QString actionParaLeer = ajustesRateHijos.at(botones)->text();
        actionParaLeer.replace(QString("&"),QString(""));
        m_speech->say(actionParaLeer);
    }

    //Barre posicion por posicion
    botones++;

    //Si llega a la ultima posicion se reinicia el barrido a la primera
    if(botones==ajustesRateHijos.length()){
        botones=0;
    }
    qDebug()<<botones;

}
```

Listado B.13: Función 'setActionsRate()'

```
void MainWindow::setActionsRate()
{
    //Añade tu ventana en closeMenu()
     closeMenu();

     //Crear ventana
     menuCambiarRate = new QMenu(this);

     //booleano para condicion en closeMenu()
     menuAjustesRate = true;

     //Acciones
     QString velocidad = QString::number((m_speech->rate())*100);
     subirVel = new QAction(tr("Subir Velocidad"),this);
     bajarVel = new QAction(tr("Bajar Velocidad"),this);
     menuprincipal = new QAction("Aceptar ("+velocidad+"%)",this);

    //Añadir acciones
     menuCambiarRate->addAction(subirVel);
     menuCambiarRate->addAction(bajarVel);
     menuCambiarRate->addAction(menuprincipal);

     //Mostrar ventana
     menuCambiarRate->show();

     //Ponemos el popUp y gridLayout que hemos creado.
     ui->popUpRate->setGeometry
     (popupCentralSize(menuCambiarRate,fuenteMenus));
     ui->popUpRate->setHidden(false);
     ui->popUpRate->setEnabled(true);
     ui->gridLayoutPopUpRate->addWidget(menuCambiarRate);

}
```

Listado B.14: Implementar el barrido de las opciones de la nueva ventana en 'BarridoMenuPrincipal()'

```
//Cada opcion tiene un indice que va en CONSTANTS.h
 }else if(auxSubMenu == INDICEOIR5){
     //Si la ventana esta cerrada
     if(!popUpAjustesRate){
         m_ajustesRate();

     }else if(popUpAjustesRate && botones == 1){
         //Si la ventana esta abierta y se selecciona la primera opcion
         modoSubirRate();
         m_ajustesRateActualizar();

     //Si la ventana esta abierta y se selecciona la segunda opcion
     }else if(popUpAjustesRate && botones == 2){
         modoBajarRate();
         m_ajustesRateActualizar();

     //Si la ventana esta bierta y se da a aceptar
     }else if(popUpAjustesRate && botones == 0){
         reset = true; //reset barrido
         actionResetBarrido_triggered();
         ui->popUpRate->setHidden(true);

         //A false para cerrarla
         popUpAjustesRate = false;
         return;
     }
     reset=true;
     return;
 }
```

B.3 Modificación del fichero de configuración

Listado B.15: Función 'definicionValores(...)'

```
void MainWindow::definicionValores(bool modo, QString type, QString color,
int tmnenu, int tbarrido,QString typeLetra,int tLetra,QString colorLetra,
bool modoSonido, bool modoBarridolineal, int minutosGuardadoAuto,
double volumen, double rate, double pitch, int velBarridoSonido,
QString currentFile, QString currentVoice)
{

    ... //Más codigo antes

    if(!modoSonido){
        sonidoMenus = false;
    }else{
        sonidoMenus = modoSonido;
    }if(!modoBarridolineal){
        barridolineal = false;
    }else{
        barridolineal = modoBarridolineal;
    }if(minutosGuardadoAuto==INTEGERVALUE){
        minutos = 0;
        guardadoAuto = false;
    }else{
        minutos = minutosGuardadoAuto;
         guardadoAuto = true;
    }if(volumen==INTEGERVALUE){
        m_speech2->setVolume(0.0);
        m_speech->setVolume(0.0);
    }else{
        m_speech2->setVolume(volumen);
        m_speech->setVolume(volumen);
    }if(rate==INTEGERVALUE){
        m_speech2->setRate(0.0);
        m_speech->setRate(0.0);
    }else{
        m_speech2->setRate(rate);
        m_speech->setRate(rate);
    }if(pitch==INTEGERVALUE){
        m_speech2->setPitch(0.0);
        m_speech->setPitch(0.0);
    }else{
        m_speech2->setPitch(pitch);
        m_speech->setPitch(pitch);
    }if(velBarridoSonido==INTEGERVALUE){
        tiempoBarridoSonido=TIEMPOBARRIDOSONIDO;
    }else{
        tiempoBarridoSonido = velBarridoSonido;
    }

    //Resto de codigo

}
```

Listado B.16: Función 'closeEvent(QCloseEvent *event)'

```
void MainWindow::definicionValores(bool modo, QString type, QString color,
int tmnenu, int tbarrido,QString typeLetra,int tLetra,QString colorLetra,
bool modoSonido, bool modoBarridolineal, int minutosGuardadoAuto,
double volumen, double rate, double pitch, int velBarridoSonido,
QString currentFile, QString currentVoice)
{

   void MainWindow::closeEvent(QCloseEvent *event)
{
    if(!ui->textEdit->toPlainText().isEmpty()){
        salirapp = true;
        on_actionNuevo_triggered();
    }
    QSettings settings("miapp.ini", QSettings::IniFormat);
    settings.setValue("typeFont", fuenteMenus.family());
    settings.setValue("color",backgroundColor);
    settings.setValue("activo",activo);
    settings.setValue("tMenus",LetraMenus);
    settings.setValue("tiempos",velBarrido);
    settings.setValue("fuenteletra",fontLetra.family());
    settings.setValue("fuenteletraTam",FuenteTexto);
    settings.setValue("ColorLetra",ui->textEdit->textColor());
    settings.setValue("modoSonido",sonidoMenus);
    settings.setValue("modoBarridolineal",barridolineal);
    settings.setValue("minutosGuardadoAuto",minutos);
    settings.setValue("volumen",m_speech->volume());
    settings.setValue("rate",m_speech->rate());
    settings.setValue("pitch",m_speech->pitch());
    settings.setValue("velBarridoSonido",tiempoBarridoSonido);
    settings.setValue("currentFile",curFile);
    settings.setValue("currentVoice",m_speech->voice().name());

    QMainWindow::closeEvent(event);

}
```

Apéndice C

Instalación y configuración del proyecto

En esta sección, abordaremos la configuración e instalación del software requerido para trabajar en la aplicación en diferentes sistemas operativos.

C.1 Instalación y configuración de Qt

C.1.1 Windows

Para utilizar Qt en Windows debemos descargarnos el instalador online de Qt desde el siguiente enlace: https://www.qt.io/download-open-source.

Una vez tenemos el instalador descargado, lo ejecutamos y en el apartado de agregar componentes agregamos Qt 5.15.2 y en el apartado "Developer and Designer tools" agregamos Qt Creator, Debugging Tools for Windows, MinGW8.1 64bit y 32 bit y Qt Installer Framework.

Para abrir el proyecto debemos abrir Qt Creator y seleccionar la accion "Open Project", donde seleccionaremos el fichero "PredWin.pro". Tras haberlo seleccionado nos pedirá que asignamos un kit a nuestro proyecto, en nuestro caso hemos utilizado el kit "Desktop Qt 5.15.2 MinGW 64-bit" ya que es la versión más estable y con mayor compatibilidad respecto a anteriores versiones.

Si queremos ejecutar el proyecto seleccionaremos la opción "Play" de abajo a la izquierda en modo "Debug".

C.1.2 Android

Para poder probar el proyecto en Android debemos ir a Qt Creator->Edit->Preferences->Devices->Android y debemos configurar un Sdk válido junto a su localización en el sistema. Tras esto se realizará una comprobación y si son compatibles las versiones aparecerá "Android settings are OK" junto a la versión..

Una vez hayamos completado este paso, debemos descargar Android Studio https://developer.android.com/studio y crear un dispositivo virtual desde el "Device Manager" y lo abriremos. Una vez este abierto nuestro dispositivo, podremos ejecutar el proyecto desde Qt Creator dándole al botón de ejecutar habiendo seleccionado el kit de Android Qt 5.15.2 Clang Multi-Abi.

En el caso de que al ejecutar el programa ocurra algún tipo de error de "Gradle" iremos al directorio donde se compiló esta ejecución de Android y accedemos a android-build->gradle->wrapper->gradle-wrapper.properties y modificaremos esta línea "distributionUrl=https //services.gradle.org/distributions/gradle-6.3-bin.zip".

C.1.3 Linux

Si queremos probar nuestro proyecto en Linux debemos seguir los mismos pasos que con Windows.

C.2 Configuración del ejecutable según el sistema operativo

En esta sección vamos a ver paso a paso cómo generar el archivo ejecutable en cada uno de los sistemas operativos en el entorno de Qt. Tenemos dos modos de compilación, modo Debug (depuración) y modo Release (lanzamiento).

El modo de depuración se utiliza cuando la aplicación se encuentra en desarrollo, ya que proporciona información sobre posibles errores en el proyecto. El modo de lanzamiento se ejecuta para obtener la versión final de la aplicación.

C.2.1 Windows

Para obtener el fichero ejecutable en el sistema optativo Windows vamos a realizar los siguientes pasos:

1. Establecemos en Qt el modo de compilación Release, y ejecutamos la aplicación. Como muestra la figura C.1.

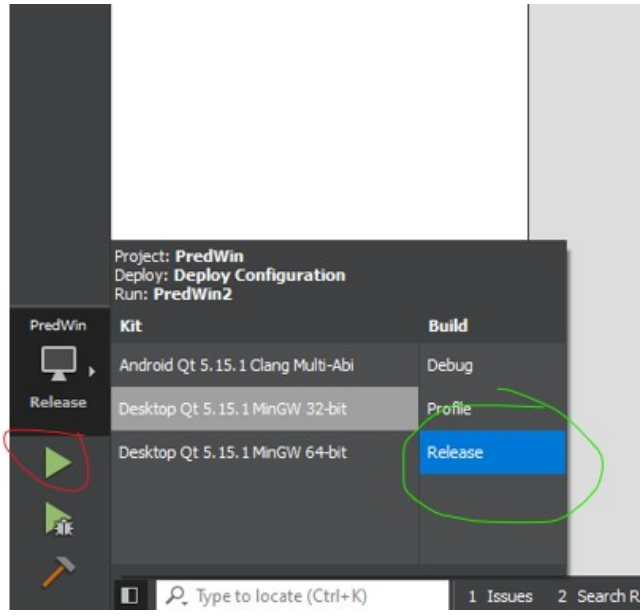


Figura C.1: Modo de compilación Release

2. Vamos al directorio donde se encuentra el fichero ejecutable que se ha generado.Podemos ver el directorio donde se ha generado en menú de "Projects->Build settings->Build Directory", como se muestra en la figura C.2.

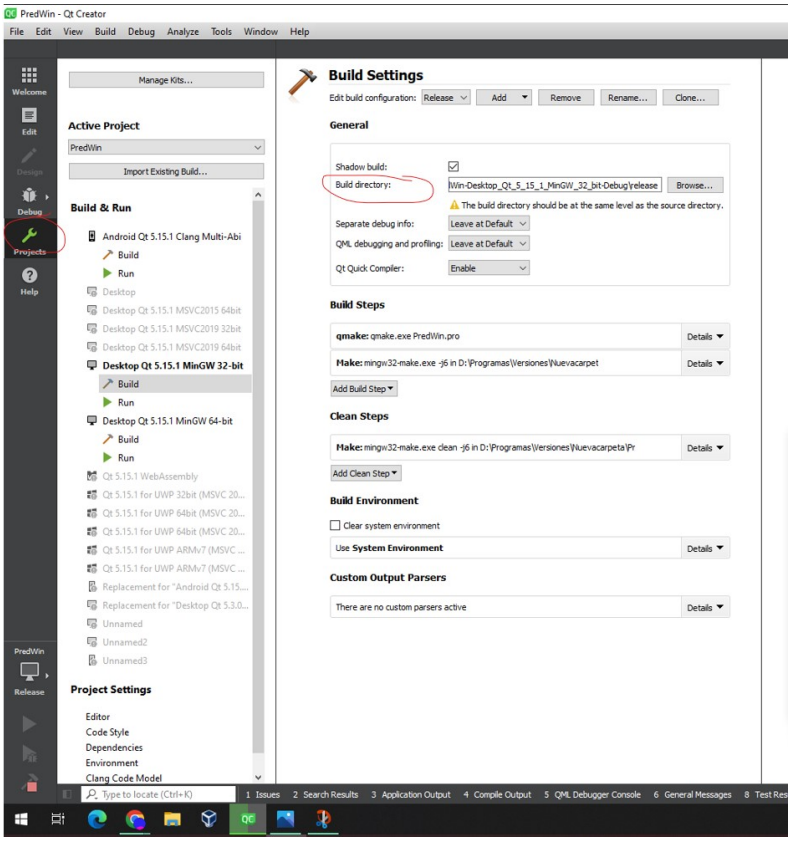


Figura C.2: Directorio de ejecución

3. Con el buscador de windows abrimos la terminal de QT. Como se ve en la figura C.3 y se nos mostrara la consola como se ve en la figura C.4.

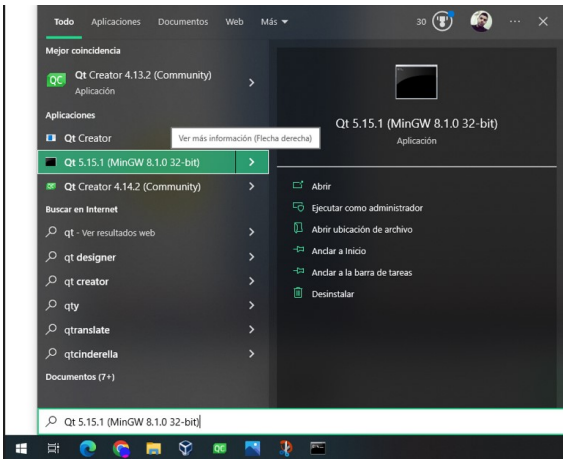


Figura C.3: Terminal de Qt

Figura C.4: Consola de comandos de Qt

4. Nos movemos al directorio de ejecución y generamos el fichero ejecutable poniendo en la consola los siguientes comandos:

   ```
   cd <directorio>
   ```

   ```
   windeployqt predwin.exe
   ```

5. Volvemos al directorio de ejecución y como vemos en la figura C.5 se han generado todos los ficheros necesarios para lanzar el ejecutable sin problemas. Si pulsamos dos veces sobre el fichero "PredWin.exe" se nos abre la aplicación.

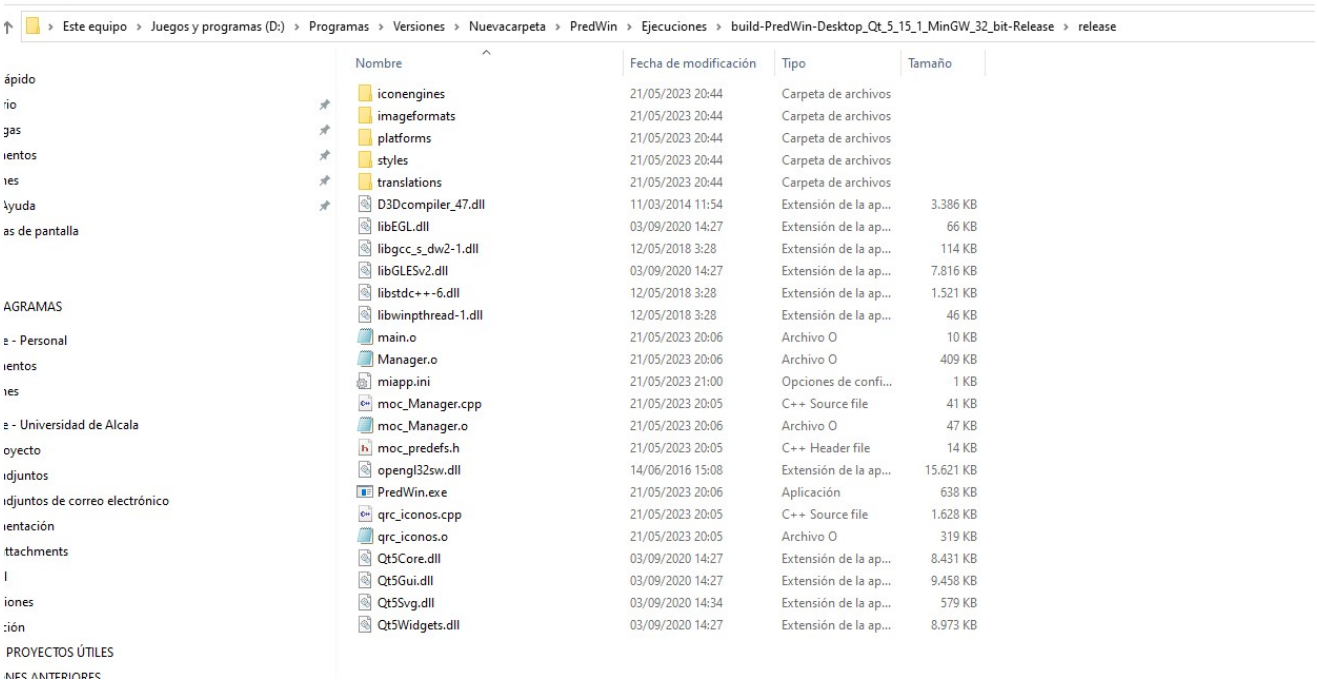


Figura C.5: Ejecutable Windows

6. Ahora para poder agrupar todos estos archivos en un instalador lo que haremos será usar la herramienta Qt Installer Framework [9].

   Descargamos esta herramienta del siguiente enlace https://download.qt.io/official_releases/qt-installer-framework/4.6.0/ y lo instalamos. Tras haberse completado la instalación, vamos al directorio donde se ha instalado, en mi caso es

`C:/Qt/Tools/QtInstallerFramework/4.6/bin`, copiamos esta ruta y la agregamos a nuestro "PATH" en editar variables de entorno del sistema.

Una vez tenemos agregada la ruta en nuestro "PATH", entramos a la carpeta "examples" y copiamos el ejemplo "startmenu". Lo pegamos en una nueva carpeta que creemos y tras abrirla veremos dos carpetas: config y packages, entramos en config y modificamos el archivo de config.xml como se muestra en la figura C.6.

config.xml: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?xml version="1.0" encoding="UTF-8"?>
<Installer>
    <Name>PredWin Installer</Name>
    <Version>1.0.0</Version>
    <Title>PredWin</Title>
    <Publisher>Carlos Carrasco Álvarez</Publisher>
    <!-- Directory name is used in component.xml -->
    <StartMenuDir>PredWin</StartMenuDir>
    <TargetDir>@HomeDir@/PredWin</TargetDir>
</Installer>
```

Figura C.6: Archivo config.xml

Después entramos en la carpeta "packages" y en la subcarpeta "data" borramos el README.txt y pegamos todos los archivos que se han generado en el paso 4. Ahora accedemos a la carpeta "meta" y abrimos el archivo "installscript.qs" y modificamos las últimas lineas como se muestra en la figura C.7.

```
Component.prototype.createOperations = function()
{
    // call default implementation to actually install README.txt!
    component.createOperations();

    if (systemInfo.productType === "windows") {
        component.addOperation("CreateShortcut", "@TargetDir@/PredWin.exe", "@StartMenuDir@/PredWin.lnk",
            "workingDirectory=@TargetDir@", "iconPath=%SystemRoot%/system32/SHELL32.dll",
            "iconId=2", "description=Open README file");
    }
}
```

Figura C.7: Archivo installscript.qs

Una vez completados los pasos anteriores, regresamos a la carpeta inicial del ejemplo "startmenu" y abrimos el archivo "README". En este archivo encontraremos las instrucciones para crear el ejecutable del instalador. Copiaremos la instrucción y abrimos una nueva consola de comandos en la ruta de "startmenu". Pegamos la instrucción y añadimos el título de nuestro instalador.

```
binarycreator --offline-only -c config/config.xml -p packages PredWinInstalador
```

Una vez finalizado el proceso, podremos encontrar el instalador generado en la carpeta "startmenu", tal como se muestra en la figura C.8. A partir de ahí, podremos distribuir el instalador para que las personas puedan instalar PredWin en sus sistemas Windows.

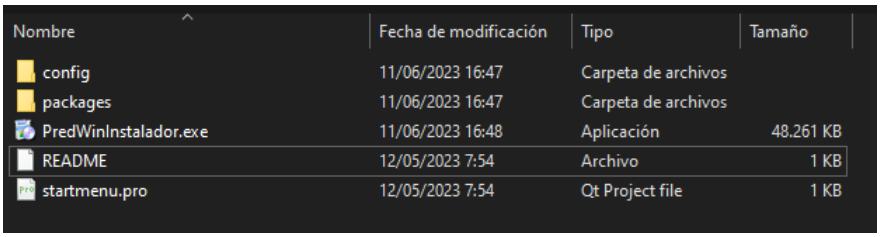


Figura C.8: Carpeta con el instalador

C.2.2 Android

Para obtener el archivo .apk de Android, simplemente debemos dirigirnos al directorio donde se ha compilado el proyecto de Android y buscar el archivo llamado .ªndroid-build.apk".

C.2.3 Linux

Para generar el instalador de PredWin en Linux hay que seguir unos pasos similares que los de Windows.

1. El primer y segundo paso son exactamente iguales en Windows y Linux. Iremos a Qt y ejecutaremos la aplicación en modo Release, lo que nos generará una carpeta con los archivos necesarios para ejecutar la aplicación.

2. En este paso tendremos que usar un equivalente a windeployqt para obtener las dependencias y librerias necesarias que necesita PredWin para ser ejecutado. Usaremos "CQtDeployer", para descargarlo abriremos nuestra terminal en Linux y ejecutaremos el comando `sudo snap install cqtdeployer`.

3. Una vez descargado, iremos a la carpeta en la que hemos compilado el proyecto en modo Release y abriremos una terminal en la que ejecutaremos la siguiente instrucción, `cqtdeployer -bin PredWin -qmake <dirqmake>`. Donde en mi caso, mi <dirqmake>es `/home/carlos/Qt/5.15.2/gcc_64/bin/qmake`. Esta instrucción es la equivalente en Linux a `windeployqt` y nos proporcionará todos los archivos que necesita PredWin para ser ejecutado desde cualquier sistema Linux, como se muestra en la figura C.9.

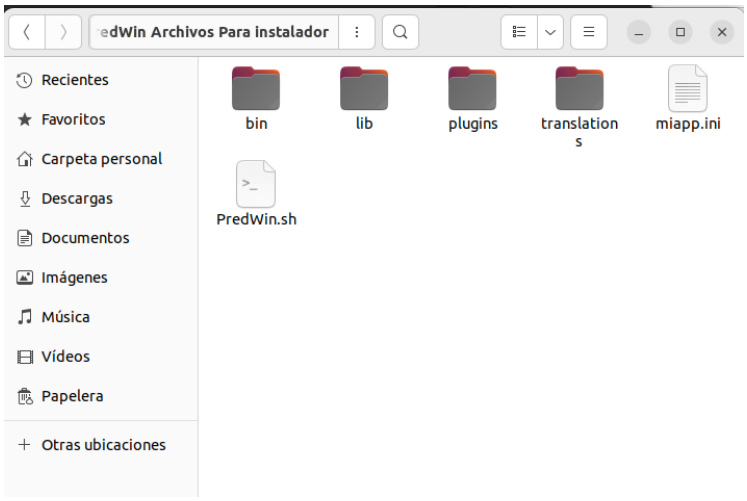


Figura C.9: Carpeta de Linux con los archivos generados tras usar CQtDeployer

4. Con todo esto ya podemos generar el instalador. Para ello tenemos que instalar en Linux la herramienta Qt Installer Framework https://download.qt.io/official_releases/qt-installer-framework/4.6.0/.

   Iremos a la carpeta dónde se instaló, y buscaremos de nuevo la subcarpeta "examples". Copiaremos el ejemplo de "tutorial" y lo pegaremos en una nueva carpeta. Accedemos a la subcarpeta "config" y modificaremos el archivo "config.xml" como se muestra en la figura C.10.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Installer>
    <Name>PredWin</Name>
    <Version>1.0.0</Version>
    <Title>PredWin Installer</Title>
    <Publisher>Carlos Carrasco Álvarez</Publisher>
    <StartMenuDir>PredWin</StartMenuDir>
    <TargetDir>@HomeDir@/PredWin</TargetDir>
</Installer>
```

Figura C.10: Archivo config.xml en Linux

Una vez modificado el archivo "config.xml" accederemos a la subcarpeta "packages" y veremos otras dos subcarpetas llamadas "data" y "meta". En la carpeta "data" pegaremos todos los archivos que se generarón tras usar el comando.

En la subcarpeta "meta" borraremos los archivos "installscript.qs" y "page.ui" modificaremos el archivo "package.xml" como se muestra en la figura C.11.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Package>
    <DisplayName>PredWin</DisplayName>
    <Description>PredWin</Description>
    <Version>0.1.0-1</Version>
    <ReleaseDate>2023-06-11</ReleaseDate>
    <Licenses>
        <License name="Beer Public License Agreement" file="license.txt" />
    </Licenses>
    <Default>true</Default>
</Package>
```

Figura C.11: Archivo package.xml en Linux

Tras estos pasos debemos obtener la ruta donde se encuentra el "binarycreator" y la copiaremos. En mi caso la ruta es `home/carlos/Qt/QtIFW-4.6.0/bin/binarycreator`. Abriremos una nueva terminal desde la carpeta donde veamos las subcarpetas "config" y "packages"y pegaremos la ruta que hemos copiado añadiendo el binarycreator y los siguientes parámetros: `<dirbinarycreator> -c config/config.xml -p packages PredWinInstaller.run`.

Esta instrucción nos abrá generado el ejecutable, como se muestra en la figura C.12.

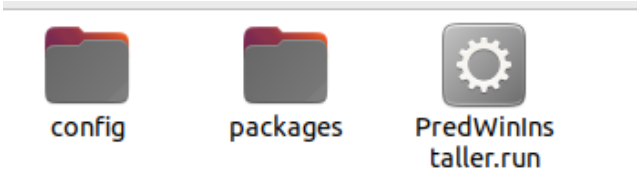


Figura C.12: PredWinInstaller.run

Apéndice D

Pliego de condiciones

Las herramientas necesarias para la elaboración del proyecto han sido:

- PC compatible.
- Qt 5.15.2 [10].
- Qt Installer Framework [9].
- Android Studio [11].
- Gradle 6.3 [12].
- CQtDeployer [13].
- Sistema operativo Windows [14].
- Sistema operativo GNU/Linux [15].
- Sistema operativo Android [16].
- VirtualBox [17].

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá