



Memoria proyecto:

Conoce tu ukelele

Colegio "Santa María la Nueva
y San José Artesano"
Desarrollo de aplicaciones
Web

Ángel Mori Martínez Díez

31 - 05 - 2022

**Quién me iba a decir a mí que un pato
de goma iba a ser el mejor sistema
para depurar errores.**

Contenido

Contenido	3
Índice de figuras.....	5
1. Introducción.....	6
2. Objetivos.....	7
3. Planificación y desarrollo	8
3.1. Planificación temporal.....	8
3.2. Recursos.....	8
3.2.1. Recursos hardware.....	8
3.2.2. Recursos software.....	8
4. Recogida de información.....	10
4.1. MonoDevelop	10
4.2. Electron.....	11
4.3. Lazarus.....	11
4.4. Xojo	12
4.5. Visual Studio 2022	12
4.6. Acordes y escalas.....	13
5. Especificación de requisitos	14
5.1. Requisitos funcionales.....	14
5.2. Requisitos no funcionales.....	14
6. Descripción mediante	15
6.1. Diagrama de clases	15
6.2. Diseño de interfaces.....	16
6.2.1. El mástil.....	16
6.2.2. Generación de acordes.....	17
6.3. Sección de escalas	20

6.4. Diálogo de afinación.....	21
7. Desarrollo futuro.....	23
7.1. Acordes avanzados	23
7.2. Añadido de escalas nuevas o personalizadas.....	23
7.3. Reproductor de sonidos	23
7.4. Version web.....	24
7.5. Version de pago bajo licencia o suscripción	25
7.5.1. Versión de escritorio.....	25
7.5.2. Version web.....	25
8. Conclusiones.....	26
9. Referencias	27
10. Apéndice: código del proyecto	28

Índice de figuras

Ilustración 1 Logo MonoDevelop	10
Ilustración 2 Logo Electron	11
Ilustración 3 Logo Lazarus	11
Ilustración 4 Logo Xojo	12
Ilustración 5 Logo Visual Studio	13
Ilustración 6 Diagrama de clases	15
Ilustración 7 Mástil	16
Ilustración 8 Mástil con notas ocultas.....	16
Ilustración 9 Detalle de los botones	17
Ilustración 10 Sección de generación de acordes	17
Ilustración 11 Bloques básicos de botones para generar acordes	18
Ilustración 12 Bloques de botones opcionales para generar acordes.....	18
Ilustración 13 Detalle del nombre de los acordes	19
Ilustración 14 Detalle del bloque de información.....	19
Ilustración 15 Sección de escalas.....	20
Ilustración 16 Bloque de botones de selección de escala	20
Ilustración 17 Bloque de información de la escala	21
Ilustración 18 Detalle botón de acceso al diálogo de afinación.....	21
Ilustración 19 Detalle del diálogo	22
Ilustración 20 Métodos de la clase NotasParser	28
Ilustración 21 Uso de expresiones regulares.....	28
Ilustración 22 Parte de la lógica de formación de acordes.....	29
Ilustración 23 Las escalas están predefinidas y asignadas a con un nombre en un diccionario	30

1. Introducción

El proyecto **Conoce tu ukelele** consiste en una aplicación de escritorio para Windows enfocada a la facilitación del aprendizaje del ukelele como herramienta para la interpretación y la creación musical. Con ella y algunos conocimientos del instrumento o de solfeo, una persona podrá aprender multitud de las posibles posiciones, acordes, escalas y notas del ukelele. Así como a crear con él y modificar su afinación con las implicaciones que esto conlleva.

La aplicación está escrita en C# y emplea Windows Forms, una librería libre y de código abierto incluida en el framework .NET. Con ella estas herramientas es posible crear interfaces bastante fácil y sencillamente.

A la hora del desarrollo se ha buscado obtener la información y el software de las fuentes oficiales para garantizar la veracidad la información y la optimización de los recursos empleados.

2. Objetivos

Existe mucha información a lo largo de la red sobre el ukelele, cómo tocarlo y cómo posicionar los dedos para distintas formas de un gran cantidad de acordes. Ya sea en forma de videos, diagramas o tablaturas. Sin embargo, esta información es estática y muchas veces limitada a lo básico o convencional.

Toda esta información es práctica y sencilla, pero demasiado sencilla cuando lo que quieras son cosas nuevas o no tan comunes, o cuando pretendes crear, modificar o improvisar música.

La estandarización de la música y los instrumentos no deja para nada libre al ukelele y los datos que se pueden encontrar sobre este. No digamos ya siquiera de su afinación, que, si bien existen dos o tres estándares, apenas se puede encontrar buena información que no sea de la afinación estándar.

Conoce tu ukelele pretende solventar este problema al presentarse como una aplicación práctica y dinámica que sirve de herramienta con la que aprender uno mismo a crear y buscar alternativas a lo convencional, además de lo estándar, lo fácil o lo simple.

El ukelele, como instrumento musical que es, presenta infinitas posibilidades que se van aprendiendo con práctica, teoría musical o solfeo y prueba y error. Esto queda facilitado con la herramienta que es la aplicación.

3. Planificación y desarrollo

3.1. Planificación temporal

Las fases programadas del proyecto han sido las siguientes:

- Búsqueda y documentación sobre las diferentes tecnologías disponibles y selección de la más apropiada
- Estudio la lógica y los algoritmos:
 - o de conversión y representación de las notas musicales como números y viceversa
 - o de creación y representación de acordes y escalas a partir de la matematización realizada a las notas y las selecciones del usuario
- Diseño del menú principal
- Diseño de las distintas pantallas de la aplicación
- Programación de la lógica de las distintas pantallas
- Pruebas de uso, testeo de casos límite y excepciones y corrección de errores encontrados
- Implementación y prueba de pequeñas mejoras

3.2. Recursos

3.2.1. Recursos hardware

Tanto para la ejecución como para el desarrollo de la aplicación, no es necesario nada más que un ordenador que sea capaz de ejecutar el software especificado en el punto siguiente.

3.2.2. Recursos software

3.2.2.1. Para la ejecución

- Windows 10 o posterior
- Windows Server 2016 o posterior
- Parallels (para la ejecución de Windows en macOS, si se usa este SO)

3.2.2.2. Durante el desarrollo

- Parallels
 - Ya que se ha usado un ordenador Mac
- Windows 11
 - En una máquina virtual de Parallels
- Visual Studio 2022
 - Como entorno de desarrollo integrado
- GitHub Desktop y versión web
 - Para el control de versiones.

4. Recogida de información

En un principio había dudas cómo hacer la aplicación, qué tecnologías usar y qué beneficios y desventajas tenía cada una a la hora tanto del desarrollo como del uso de la aplicación.

Por un lado se buscaba la simpleza y por otro la versatilidad a la hora de ejecutar la aplicación en distintos sistemas operativos.

También existía un debate sobre si emplear los conocimientos adquiridos durante el curso sobre ciertas tecnologías o decantarse por probar otras distintas.

A continuación, algunas de las tecnologías que se han estudiado emplear a la hora de desarrollar el proyecto.

4.1. MonoDevelop

Entorno de desarrollo integrado multiplataforma que permite el desarrollo de aplicaciones en Linux, macOS y Windows.

Soporta los siguientes lenguajes: C#, F#, Visual Basic .NET y Vala.

En la propia web oficial te ofrecen usar *Visual Studio for Mac*, si dispones de este tipo de ordenador como es mi caso.

Esto se debe a que la aplicación está desarrollada sobre una versión de MonoDevelop con extensiones y opciones extra disponibles en Visual Studio.

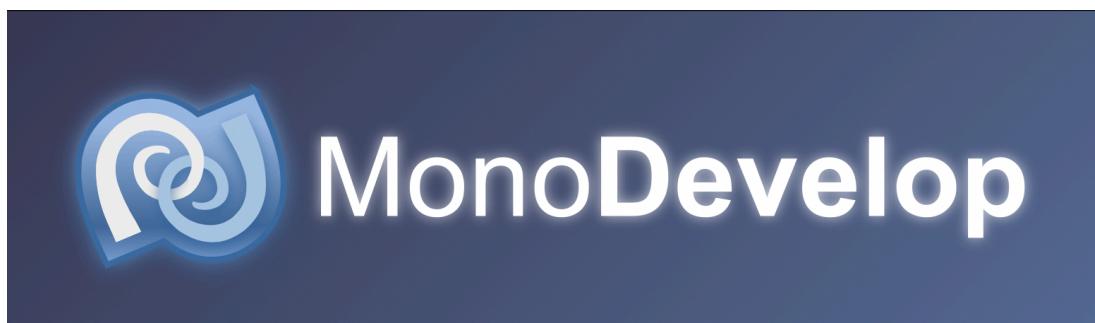


Ilustración 1 Logo MonoDevelop

4.2. Electron

Es un framework libre y de código abierto desarrollado y mantenido por GitHub. Está preparado para la creación de aplicaciones desarrolladas con HTML, CSS y JavaScript.

Emplea las tecnologías web Chromium y Node.js para crear aplicaciones multiplataforma de escritorio, por lo que no ofrece un desarrollo nativo.



Ilustración 2 Logo Electron

4.3. Lazarus

Es otro entorno de desarrollo integrado multiplataforma libre y de código abierto, diseñado para el desarrollo rápido de aplicaciones. Es compatible con Delphi, pero está basado en Free Pascal y Objetive Pascal.

Dispone de elementos de interfaz ya creados, como botones, cajas de texto, menús y diálogos.



Ilustración 3 Logo Lazarus

4.4. Xojo

Tambien es un entorno de desarrollo integrado multiplataforma, diseñado para el desarrollo rápido de aplicaciones.

Sin embargo este posee una licencia propietaria, por lo que es de pago, y usa su propio lenguaje de programación homónimo, bastante parecido a Visual Basic.

Dispone de diferentes tipos de licencias según se pretenda usar para desarrollo a web, de escritorio, iOS, Android, consola...

La primera versión salió en 1996 y desde entonces se actualiza y mantiene añadiendo cada vez más funcionalidades nuevas.

Se mantiene en el mercado gracias a que se adapta con rapidez a las nuevas tendencias del mercado



Ilustración 4 Logo Xojo

4.5. Visual Studio 2022

Es el entorno de desarrollo integrado multiplataforma de Microsoft diseñado para la creación de aplicaciones de web, de escritorio y móviles.

Este es entorno que se ha utilizado durante el curso, en concreto con la tecnología de Windows Forms.

Este entorno es el que finalmente ha sido seleccionado para realizar la aplicación tras la investigación de los anteriores, debido a la sencillez y practicidad de Windows Forms a la hora de crear interfaces gráficas.

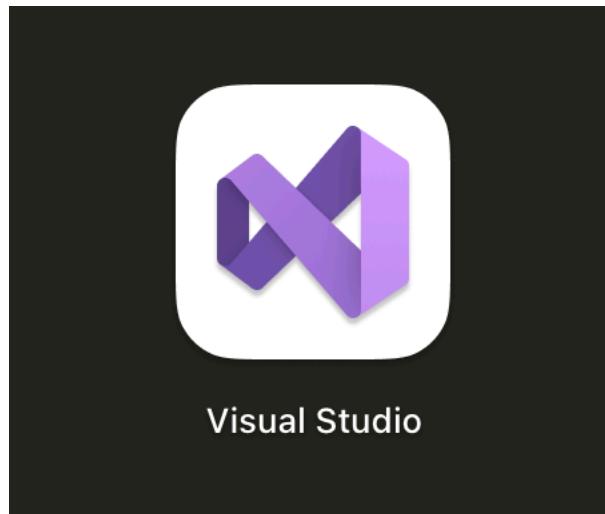


Ilustración 5 Logo Visual Studio

4.6. Acordes y escalas

Dados mis conocimientos musicales, no ha sido difícil encontrar la formas más optimas de desarrollar la aplicación.

Sin embargo, sobre todo en cuanto a escalas, ha sido necesario consultar alguna fuente para asegurar la veracidad y la versatilidad de los acordes y escalas disponibles en el programa.

Para ello se han utilizado dos fuentes: PianoChord.org y Wikipedia.

5. Especificación de requisitos

5.1. Requisitos funcionales

- ✓ Ofrecer una gran variedad de acordes y escalas
- ✓ Mostrar las diferentes posibilidades a la hora de formar cada acorde
- ✓ Mostrar las diferentes escalas
- ✓ Realizar todo lo anterior con afinaciones especificadas por el usuario
- ✓ Ser portable y no requerir de conexión a internet
- ✓ Ser interactiva

5.2. Requisitos no funcionales

- ✓ Sencillez en la interfaz
- ✓ Facilidad de uso
- ✓ Ser intuitiva

6. Descripción mediante

6.1. Diagrama de clases



Ilustración 6 Diagrama de clases

6.2. Diseño de interfaces

6.2.1. El mástil

Elemento común de las dos formularios base del programa. Aun siendo común en ambos, cada uno tiene el suyo propio.

Las notas son elementos clicables siempre que el botón que regula esta función esté en modo ocultar al tocar, como en la imagen.

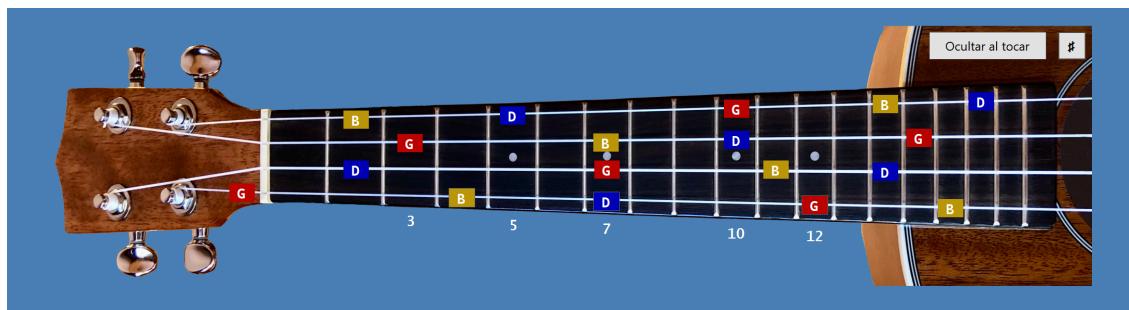


Ilustración 7 Mástil

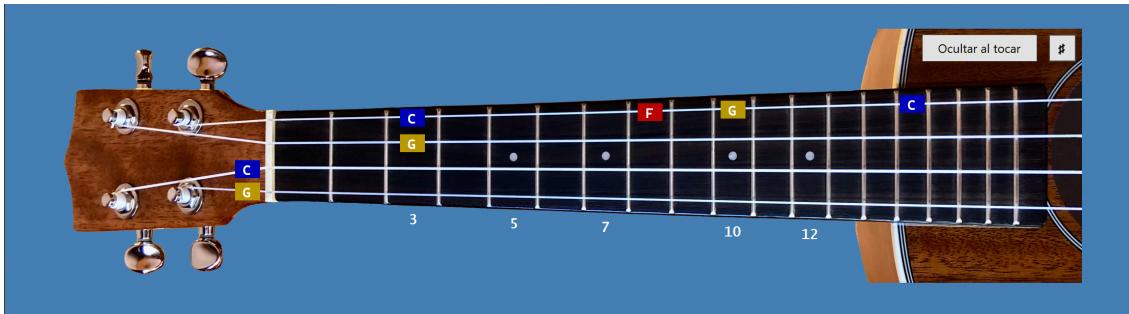


Ilustración 8 Mástil con notas ocultas

Dispone de dos botones: el antes mencionado para ocultar o no las notas y otro que cambia las notas del mástil entre sostenidos y bemoles.

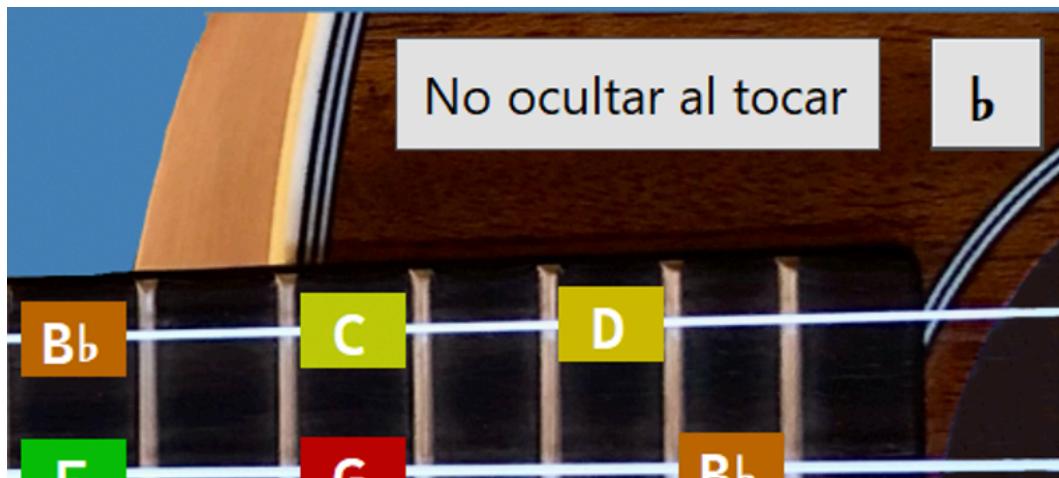


Ilustración 9 Detalle de los botones

6.2.2. Generación de acordes

Podemos ver la pantalla inicial de la aplicación, que también es el formulario de generación de acordes.

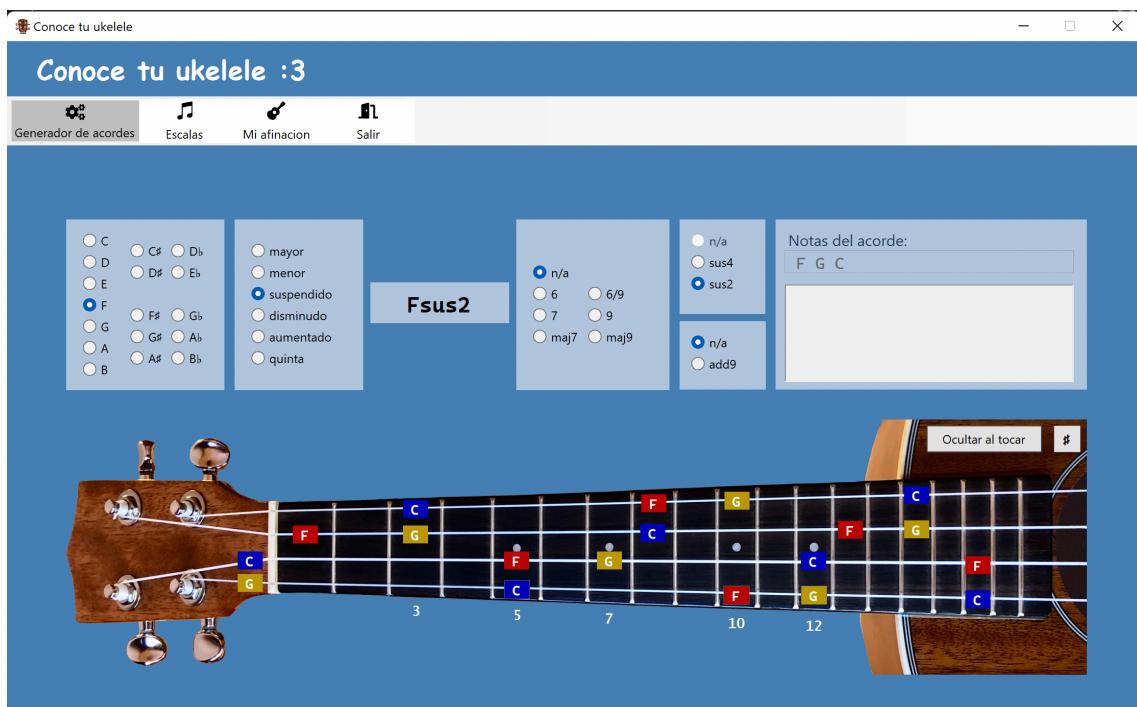


Ilustración 10 Sección de generación de acordes

Para la formación de los acordes podemos ver que hay distintos bloques de botones:

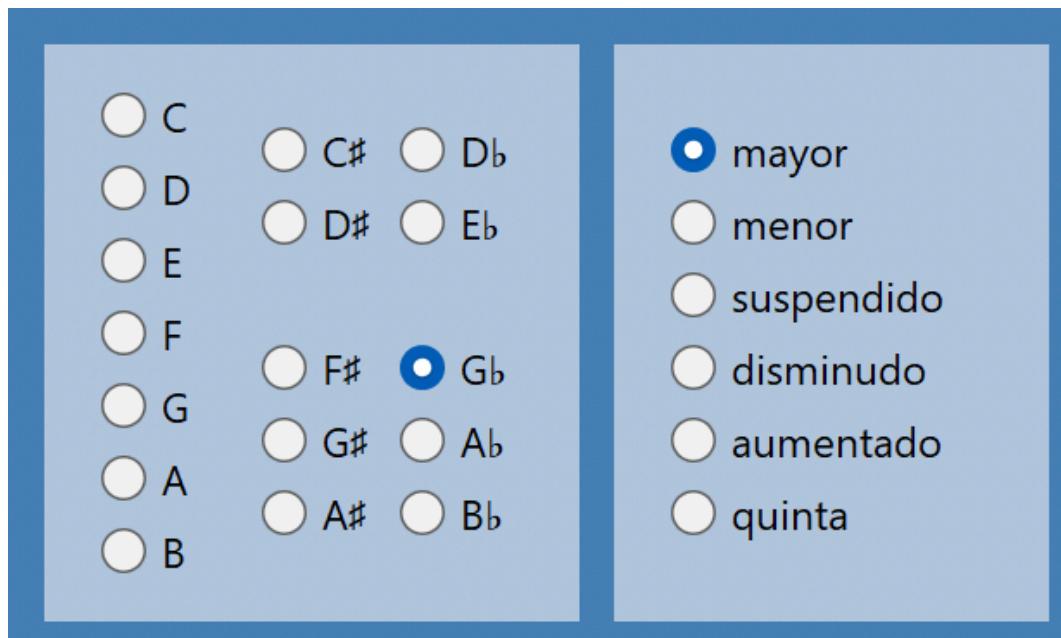


Ilustración 11 Bloques básicos de botones para generar acordes

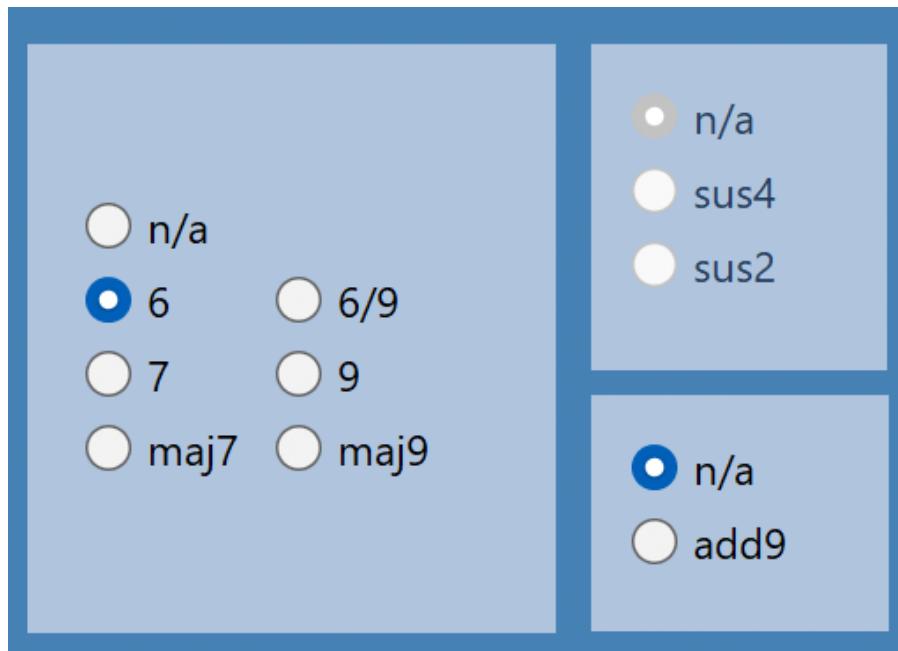


Ilustración 12 Bloques de botones opcionales para generar acordes

El resultado de la combinación de los botones seleccionados un acorde con su nombre y otro bloque con información.

Este último contiene un cuadro de texto que estará vacío o relleno según se necesite mostrar información al usuario o no.

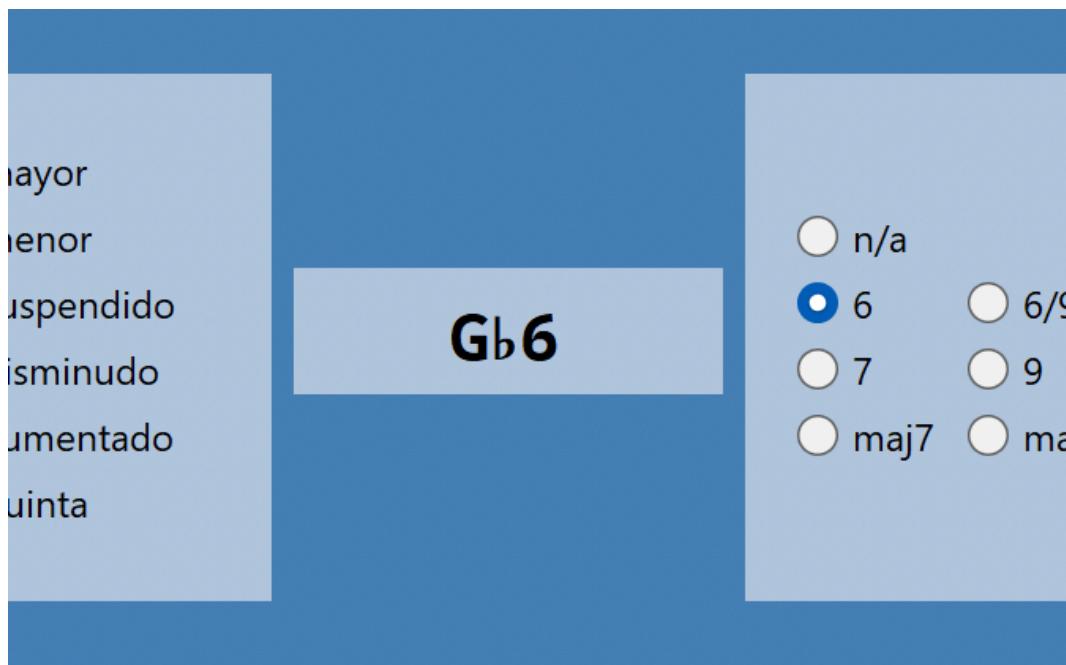


Ilustración 13 Detalle del nombre de los acordes

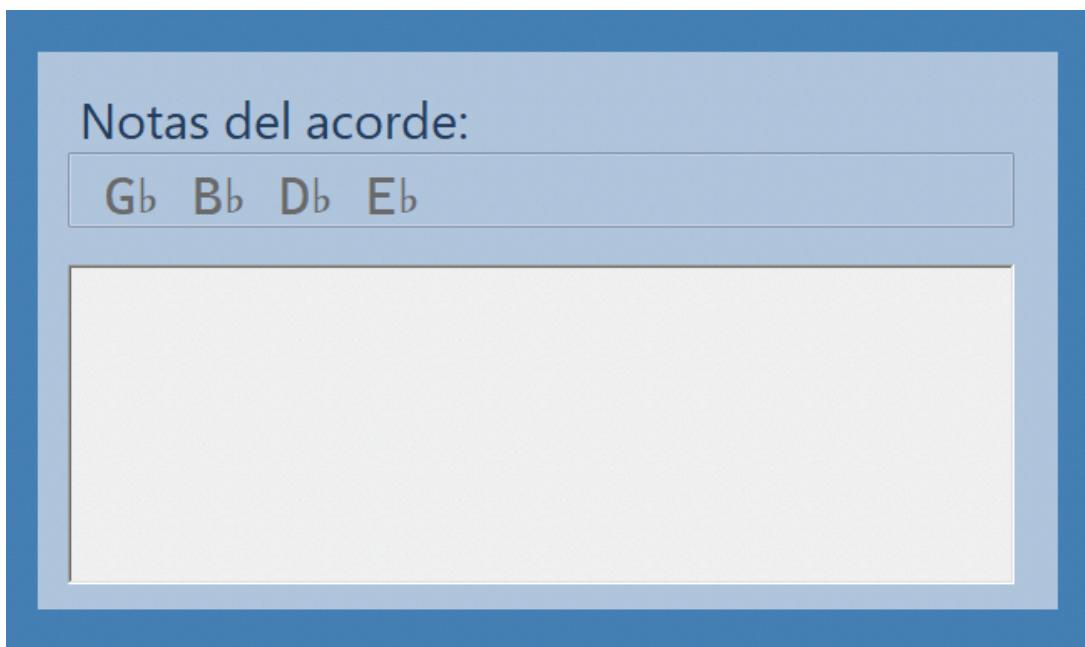


Ilustración 14 Detalle del bloque de información

6.3. Sección de escalas

Similar a la sección anterior, permite mostrar escalas en el mástil con los distintos bloques de botones y muestra información de las escalas en otro.

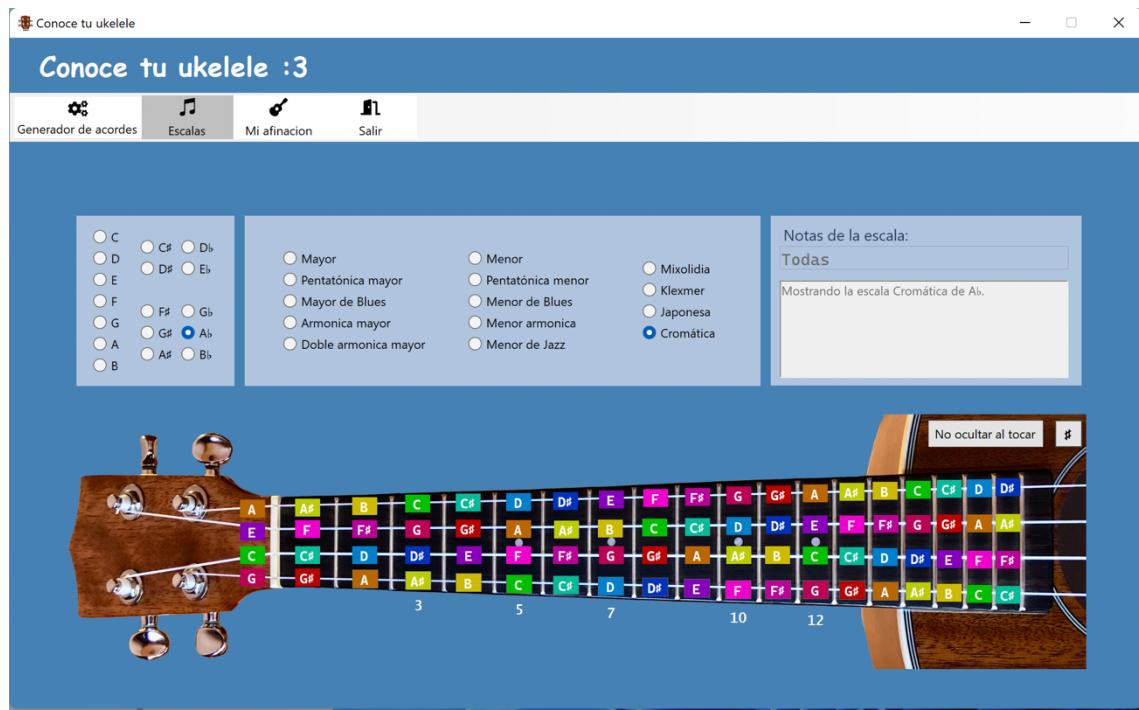


Ilustración 15 Sección de escalas



Ilustración 16 Bloque de botones de selección de escala

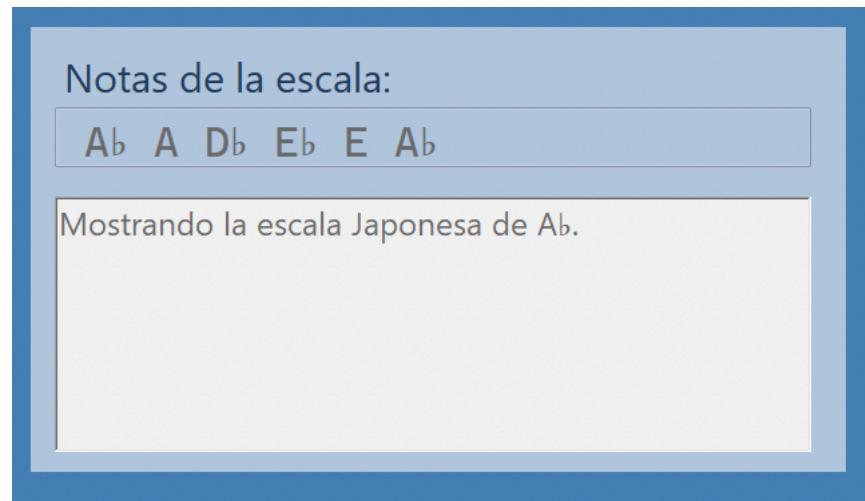


Ilustración 17 Bloque de información de la escala

6.4. Diálogo de afinación

El diálogo de afinación, accesible desde el homónimo botón, permite cambiar la afinación del ukelele para adaptarla a cada ocasión.

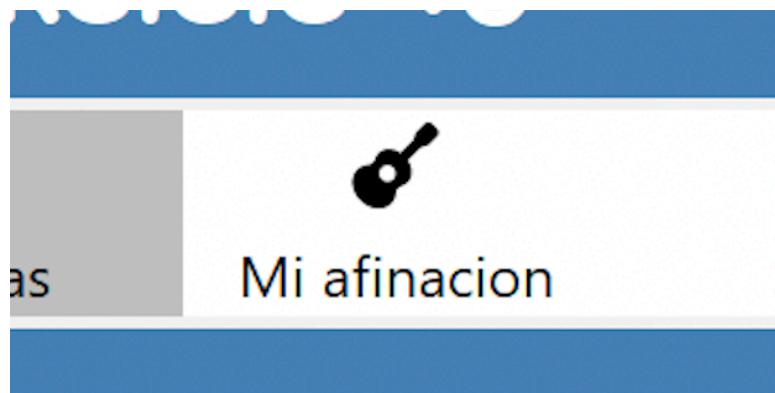


Ilustración 18 Detalle botón de acceso al diálogo de afinación

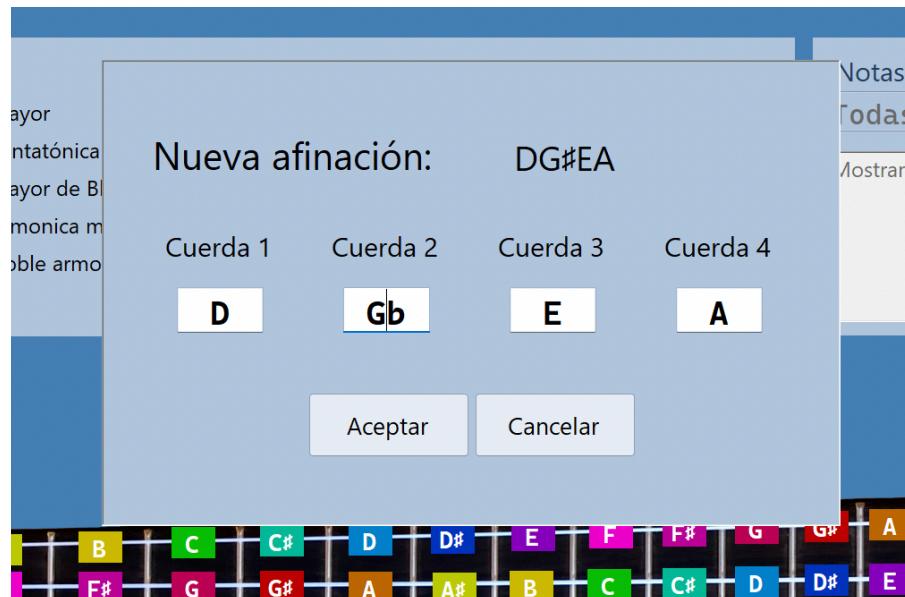


Ilustración 19 Detalle del diálogo

7. Desarrollo futuro

7.1. Acordes avanzados

A pesar de los más de 500 acordes que es posible formar con la versión actual del programa, existen muchas más tipos de acordes que se pueden tocar con el ukelele y que para hacer más amigable la interfaz no se ha dado la posibilidad de mostrarlos en la aplicación.

Es posible tanto añadir estas variedades nuevas manteniendo el estilo de la interfaz o empleando *drop lists*, lo que simplificaría la interfaz, pero, a su vez, la haría ser menos visual.

7.2. Añadido de escalas nuevas o personalizadas

Como se puede apreciar en algunas de las imágenes de esta memoria, la aplicación dispone de 14 escalas distintas ya definidas. Esta variedad de escalas será, con certeza, más que suficiente para la mayoría de usuarios de la aplicación. Sin embargo, habrá a quien le hubiese gustado poder mostrar otras menos comunes o incluso añadir las suyas propias.

Estos datos podrían ser almacenados de forma local en archivos internos del programa, lo que iría con la filosofía con la que se creó la aplicación, o de forma remota en bases de datos, lo que implicaría tener conexión a internet para tener acceso a esta funcionalidad.

7.3. Reproductor de sonidos

Si bien la representación visual de las notas puede ser suficiente, no estaría de más agregar la posibilidad de escuchar como sonaría las 130.321 combinaciones de las 76 notas diferentes representadas en el mástil del ukelele.

Esta mejora es una mejora y no ya una funcionalidad de la versión actual por dos motivos distintos. En primer lugar, esta no es una funcionalidad imprescindible, y en segundo, dada la libertad que ofrece el programa, la cantidad de combinaciones posibles están grande que no sería funcional la grabación de todas ellas por separado, lo que obligaría a grabar simplemente las 76 notas y reproducirlas en bloques de cuatro notas secuenciales.

Existen muchas aplicaciones que hacen esto y el resultado deja bastante que desear por falta de calidad en el resultado.

7.4. Version web

Durante mi periodo de formación en el centro de trabajo, he pasado a versión web con ASP:NET una aplicación desarrollada en Windows Forms.

El hecho es que es bastante sencilla la transformación, ya prácticamente todo el código detrás de los formularios es fácilmente reutilizable. El mayor trabajo es el de desarrollar de nuevo la interfaz utilizando HTML, CSS y ASP.

No se hizo la aplicación en versión web desde un principio ya que se pretendía priorizar el uso de las herramientas que se habían usado ante la creación de una aplicación tan basada en las tecnologías web.

La posibilidad de pasar la aplicación a versión web nos permite poder ampliar fácilmente las opciones que se ofrecen en la aplicación sin requerir al usuario descargar nuevo contenido para actualizar a las nuevas funcionalidades. Bastaría con actualizar en el servidor de la aplicación.

La aplicación podría de esta forma llegar a un público más amplio al poderse usar cómodamente en cualquier sistema operativo o dispositivo electrónico con acceso a internet.

7.5. Versión de pago bajo licencia o suscripción

Todas o algunas de estas mejoras, junto con otras que se puedan concretar en el futuro, podrían estar disponibles para los usuarios que las adquieran a través de un pago por una licencia de uso o una suscripción.

Se nos plantean dos opciones a la hora de hacer eso: pasar a versión web o quedarse en el formato de aplicación de escritorio.

7.5.1. Versión de escritorio

Manteniendo la filosofía con la que se originó la aplicación, se podría continuar el desarrollo del programa en formato de escritorio, trabajando de manera puramente local.

7.5.2. Versión web

Se nos plantean aquí dos opciones: transformar el programa para dejarlo como un web o crear el programa en modo web y mantener la versión de escritorio como un extra, de pago o no.

7.5.2.1. Sólo web

Simplemente la web, con las ventajas y desventajas que ya han sido mencionadas en el punto 7.4 de esta memoria.

7.5.2.2. Web + escritorio

Esto presenta las ventajas de ambas opciones, ya que la aplicación web podría estar disponible para todas las plataformas y dispositivos y disponer de una versión de pago y otra gratuita.

Por su lado, al versión de escritorio, sería una versión quizás reducida de la aplicación web, presentándose así como un extra a esta última. Sin tener que disponer de todas las opciones, estaría disponible para su descarga y uso sin conexión.

Si debería estar disponible para usuarios de pago o no sería algo a estudiar.

8. Conclusiones

La aplicación cumple con los requisitos que se pretendía cumplir en un principio, resultando una aplicación práctica y sencilla.

El desarrollo no llevó más tiempo del esperado y sus fases se fueron sucediendo sin ningun problema.

Se podría decir que una de las partes más complicadas fue el desarrollo de los algoritmos para la transformación de la música en datos funcionales para el programa y la combinación de las distintas opciones de construcción de acordes y la formación final de estos.

Las interfaces han resultado ser bastante visuales y comprensibles. Las pruebas de uso con usuarios reales lo confirmaron.

El proyecto final es funcional y satisface una necesidad de forma bastante adecuada.

Las mejoras futuras prometen y, aunque al principio no se había planteado la posibilidad, es bastante factible la rentabilización del producto, sobre todo dinfundiéndolo en su versión web.

9. Referencias

- Opciones entornos y métodos de programación:
 - <https://www.monodevelop.com/>
 - <https://www.electronjs.org/>
 - <https://www.lazarus-ide.org/>
 - <http://xojo.com/>
 - <https://visualstudio.microsoft.com/es/vs/>
- Parallels:
 - <https://www.parallels.com/es/>
- Acordes y escalas:
 - <https://es.wikipedia.org/>
 - <https://www.pianochord.org/>
- Documentación de programación:
 - <https://docs.microsoft.com/>
 - <https://stackoverflow.com/>
- Expresiones regulares:
 - <https://regex101.com/>
-

10. Apéndice: código del proyecto

La clase NotasParser se dedica a dar valor numérico a las notas musicales o sacar su valor en notación musical a partir del valor numérico

```
public static class NotaParser
{
    1 reference
    → private static bool UsarSostenidos(string clave) ...
    20 references
    → public static string GetNota(string clave, int value) ...
    4 references
    → public static int GetValorNota(string raiz) ...
}
```

Ilustración 20 Métodos de la clase NotasParser

```
public partial class DialogAfinacion : Form
{
    1 reference
    → private static readonly string pattern = "^[ABCDEFG] [#b#b]?$";
    1 reference
    → private static readonly Regex regex = new(pattern);
    18 references
}
```

Ilustración 21 Uso de expresiones regulares

```
    1 reference
    +--> private string GetRaiz() ...
    |
    |    1 reference
    |    +--> private string GetCategoria() ...
    |
    |    1 reference
    |    +--> private string GetColor() ...
    |
    |    1 reference
    |    +--> private string GetSusN() ...
    |
    |    1 reference
    |    +--> private string GetAdd() ...
    |    1 reference
    |    +--> private string GetAcorde()
    |    {
    |        notas.Clear();
    |
    |        string acorde;
    |        string raiz = GetRaiz();
    |        string categoria = GetCategoria();
    |        string susN = GetSusN();
    |        string color = GetColor();
    |        string add = GetAdd();
    |
    |        if (string.Compare(categoria, "maj") == 0 || string.Compare(categoria, "sus") == 0)
    |            categoria = "";
    |        else if (string.Compare(categoria, "5") != 0)
    |        {
    |            if (string.Compare(color, "maj7") == 0)
    |                color = "Maj7";
    |            if (string.Compare(color, "maj9") == 0)
    |                color = "Maj9";
    |        }
    |
    |        acorde = raiz + categoria + color + susN + add;
    |
    |        return acorde;
    |    }
```

Ilustración 22 Parte de la lógica de formación de acordes

```

25 references
private readonly List<int> notas = new();
1 reference
private static readonly int[] Mayor = { 0, 2, 4, 5, 7, 9, 11 };
1 reference
private static readonly int[] Pentatonicamayor = { 0, 2, 4, 5, 9 };
1 reference
private static readonly int[] MayordeBlues = { 0, 2, 3, 4, 7, 9 };
1 reference
private static readonly int[] Armonicamayor = { 0, 2, 4, 5, 7, 8, 11 };
1 reference
private static readonly int[] Doblearmonicamayor = { 0, 1, 4, 5, 7, 8, 11 };
1 reference
private static readonly int[] Menor = { 0, 2, 3, 5, 7, 8, 10 };
1 reference
private static readonly int[] Pentatonicamenor = { 0, 3, 5, 7, 10 };
1 reference
private static readonly int[] MenordeBlues = { 0, 3, 5, 6, 7, 10 };
1 reference
private static readonly int[] Menorarmonica = { 0, 2, 3, 5, 7, 8, 11 };
1 reference
private static readonly int[] MenordeJazz = { 0, 2, 3, 5, 7, 9, 11 };
1 reference
private static readonly int[] Mixolidia = { 0, 2, 4, 5, 7, 9, 10 };
1 reference
private static readonly int[] Klexmer = { 0, 1, 4, 5, 7, 8, 10 };
1 reference
private static readonly int[] Japonesa = { 0, 1, 5, 7, 8 };
1 reference
private static readonly int[] Cromatica = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 };
1 reference
private static readonly Dictionary<string, int[]> escalas = new()
{
    {"Mayor", Mayor },
    {"Pentatónica mayor", Pentatonicamayor },
    {"Mayor de Blues", MayordeBlues },
    {"Armonica mayor", Armonicamayor },
    {"Doble armonica mayor", Doblearmonicamayor },
    {"Menor", Menor },
    {"Pentatónica menor", Pentatonicamenor },
    {"Menor de Blues", MenordeBlues },
    {"Menor armonica", Menorarmonica },
    {"Menor de Jazz", MenordeJazz },
    {"Mixolidia", Mixolidia },
    {"Klexmer", Klexmer },
    {"Japonesa", Japonesa },
    {"Cromática", Cromatica }
};

```

Ilustración 23 Las escalas están predefinidas y asignadas a con un nombre en un diccionario