Escuela Politécnica Superior

Prototipo didáctico para la ejecución de secuencias de acordes en la guitarra

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Pablo Baudí Canales

Tutor/es: Pedro José Ponce de León Amador

Septiembre 2014

Índice de contenido

Índice de imágenes

Índice de Tablas

Capítulo 1: Introducción al problema

1.1 Introducción

El presente prototipo didáctico para la ejecución de secuencias en la guitarra, ha sido desarrollado para ayudar a los guitarristas a resolver uno de los dilemas más importantes a la hora de ejecutar una secuencia de acordes y es conocer si se está ejecutando de la forma más cómoda.

Debemos tener en cuenta que el mismo acorde puede ser tocado de varias formas sobre el mástil de la guitarra, por lo que es complejo determinar cuál sería la combinación ideal en una secuencia de acordes dada por un usuario, por este motivo se ha desarrollado *The Wise Guitar*, un prototipo de aplicación de apoyo para usuarios con el objetivo de obtener una solución adecuada a esta problemática. Los guitarristas, muchas veces, realizan un sobreesfuerzo a la hora de enlazar un acorde con otro, por ello el prototipo pretende minimizar el número de movimientos que se realizan sobre el mástil y facilitar el aprendizaje mostrando una secuencia de acordes adecuada y cómoda.

Hay que tener en cuenta, que la mayoría de las aplicaciones relacionadas con el ámbito de los acordes de la guitarra, muestran una secuencia donde nunca se obtiene el resultado más cómodo. Siempre facilitan los acordes más utilizados lo que no implica que sean los más cómodos. Esta fue, entre otros muchos aspectos, una de las motivaciones para desarrollar un prototipo como *The Wise Guitar*.

El prototipo, para realizar su cometido, utiliza un sistema experto de razonamiento basado en casos (*CBR*), una técnica para solucionar nuevos problemas basándose en soluciones conocidas a problemas similares, en este caso un conjunto de acordes dado por el guitarrista. Esta solución, no está exento de limitaciones, como por ejemplo la necesidad de obtener conocimiento de personas expertas en el dominio para alcanzar la suficiente información y que el sistema pueda devolver una secuencia de acordes lo más óptima posible.

1.2 Acordes en la guitarra

Para comprender mejor el dominio del problema, debemos saber qué es un acorde y cómo los guitarristas lo expresan de forma visual.

Antes de todo, tenemos que conocer el lugar donde se colocan los acordes en la guitarra, el mástil.

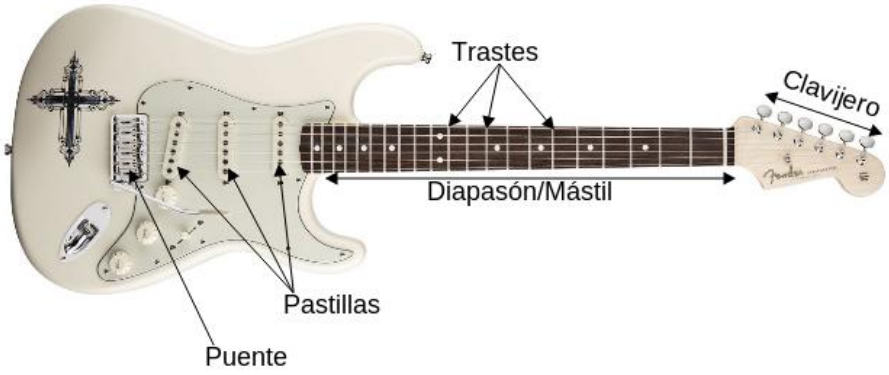


Figura 1: Partes de la guitarra

El mástil es una pieza primordial que sobresale del cuerpo de la guitarra, donde se coloca el diapasón. El diapasón se sitúa en la parte superior del mástil y es el lugar donde se encuentran los trastes, zona donde se pulsan las cuerdas para que suenen los distintos acordes. Los trastes son unos salientes metálicos que se insertan en el diapasón, para su mejor comprensión véase la *Figura 1*.

La guitarra, por defecto, suele tener 6 cuerdas y cada una de ellas de un grosor distinto. La sexta cuerda, la más gruesa de todas, y por tanto la que produce un sonido más grave está colocada por encima de las demás y, así, hasta llegar a la primera cuerda, la más fina de todas. Por defecto, la guitarra está afinada de forma estándar, de la cuerda más grave a la más aguda: E, A, D, G, B, E'.

En la *Figura 2*, se puede ver un ejemplo de acordes, donde cada línea vertical indica la cuerda y las horizontales los trastes. La cuerda que se encuentra totalmente a la izquierda indica la sexta cuerda (la más grave) y, así, hasta llegar a la cuerda de la derecha que es la primera cuerda (la más aguda). Para representar las cuerdas que no se

pulsan, se añade, visualmente, una *X* encima de la cuerda y las que se tocan al aire, es decir, que se tocan sin pulsar ningún traste, se representan con un círculo *O*.

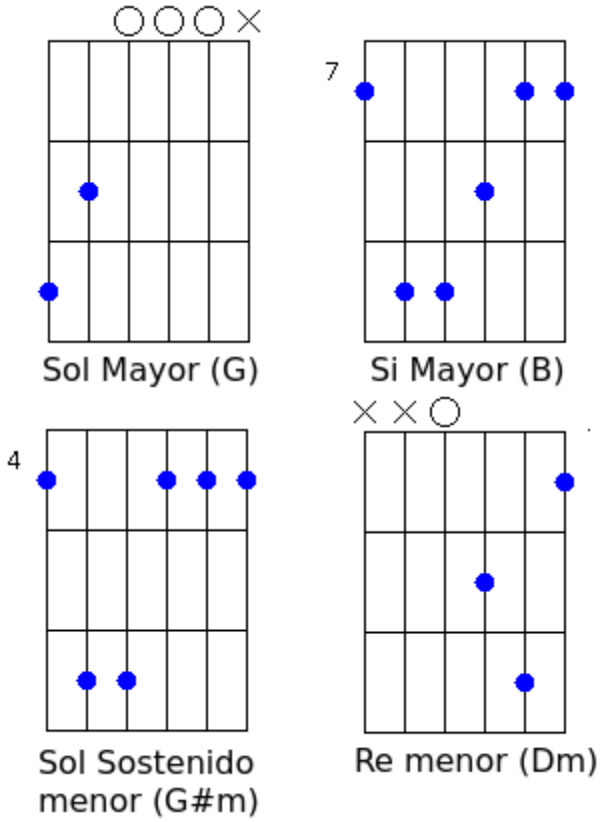


Figura 2: Ejemplo de acordes de guitarra

Los círculos de color azul indican el lugar donde se colocan los dedos, siempre teniendo como referencia en que zona del mástil nos encontramos. Esto se representa con un número al lado de los trastes, si no lo tuviera indicaría que se trata del primer traste de la guitarra.

Un acorde es un conjunto de 3 ó más notas diferentes tocadas al mismo tiempo. La diferencia de altura, en un pentagrama, entre notas se conoce como intervalo musical.

Los intervalos se denominan de la forma siguiente:

1 (Fundamental), 2 (Segunda), 3 (Tercera), 4 (Cuarta), 5 (Quinta), 6 (Sexta) y *8 (Octava).*

Los nombres de los intervalos representan la distancia del intervalo, a partir de la nota fundamental (la nota inferior). Véase el ejemplo de la *Figura 3*.

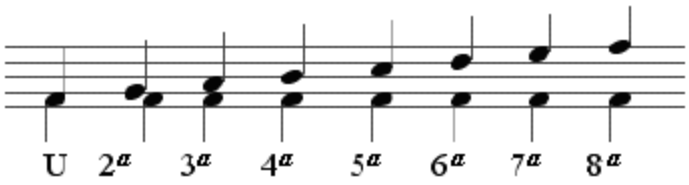


Figura 3: Intervalo musical simple

En la *Tabla 1* se muestran las distancias en semitonos. En la guitarra sería la distancia en trastes de una nota a otra.

	Segunda	Tercera	Cuarta	Quinta	Sexta	Séptima	Octava
Mayor	2	4	-	-	9	11	-
Menor	1	3	-	-	8	10	-
Aumentada	3	5	6	8	10	12	13
Disminuida	-	2	4	6	7	9	11
Justa	-	-	5	7	-	-	12

Tabla 1: Distancia en semitonos (Intervalo musical)

Los acordes están representados mediante una nota fundamental y la familia a la que pertenece, por ejemplo: *Do* sostenido menor (*C#m*).

Fundamental: **C#**.

Familia: **m**.

La notación utilizada en el ejemplo anterior, es la usada normalmente en el mundo de la música (Sistema de notación musical anglosajón). Las notas en el sistema latino son: Do, Re, Mi, Fa, Sol, La, Si y las del anglosajón son: C, D, E, F, G, A, B.

Cada familia, representa el total de notas a utilizar y la distancia entre ellas, a partir de la nota fundamental. Por ejemplo, los acordes *Mayores* hacen uso de una tríada (3 notas), *1º (Fundamental), 3º (Tercera Mayor)* y *5º (Quinta Justa)*. Si elegimos, *La Mayor (A)*, su distancia es la siguiente: De Fundamental a Tercera Mayor 4 semitonos y de Fundamental a Quinta Justa 7 semitonos, esto llevado a notación musical se representaría de la siguiente manera: *La (A) – Do Sostenido (C#) – Mi (E).*

De A a C# hay 4 semitonos y de A a E hay 7 semitonos.

Pongamos otro ejemplo, ahora con la familia de las séptimas (7). Elegimos *Do Séptima (C7)*, las séptimas se forman con una cuatriada (4 notas), *1º (Fundamental)*, *3º (Tercera Mayor)*, *5º (Quinta Justa)* y *7º (Séptima Menor)* su distancia es la siguiente: De Fundamental a Tercera Mayor 4 semitonos, de Fundamental a Quinta Justa 7 semitonos y de Fundamental a Séptima Menor 10 semitonos, esto llevado a notación musical se representaría de la siguiente manera:

Do (C) - Mi (E) - Sol (G) - Si Bemol (Bb).

De C a E hay 4 semitonos, de C a G hay 7 semitonos y de C a Bb hay 10 semitonos.

1.3 Familias y posiciones

Las familias, indica la cantidad de notas que deben aparecer en un acorde y la distancia entre ellas. Las posiciones son las diferentes formas de representar un mismo acorde de una familia. En la *Figura 4*, se puede observar un ejemplo, en este caso *Re menor (Dm)*, donde se muestra tres distintas.

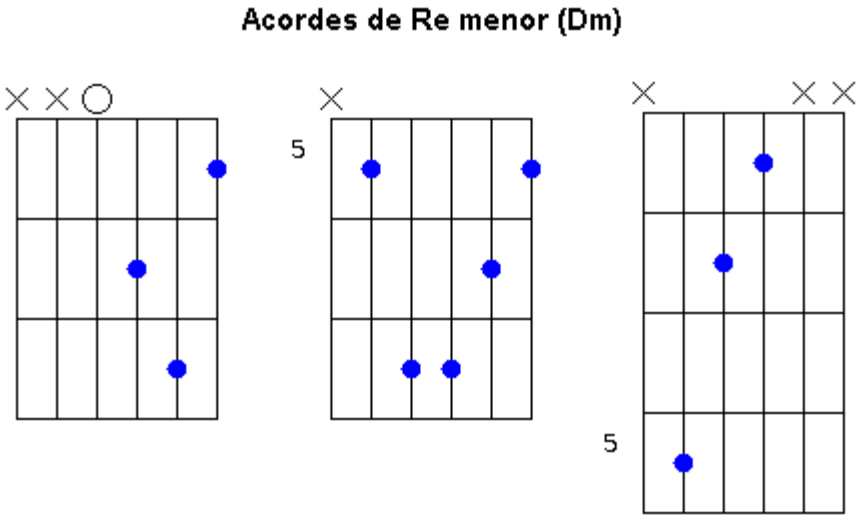


Figura 4: Tres acordes de Re menor (Dm)

En este trabajo se ha usado un total de cinco familias de acordes: *Mayores (M)*, *Menores (m)*, *Séptimas (7)*, *Mayor Séptima (M7)* y *Menor Séptima (m7)*. A pesar de ello, el prototipo está preparado para añadir posiciones como familias sin el menor problema, insertándolos en la BD.

1.4 Estado del arte

1.4.1 Artículos

Se realizó una investigación acerca de sistemas y trabajos teóricos que cumplieran el requisito mínimo de parecerse a *The Wise Guitar*.
Existen multitud de sistemas y artículos basados en sistemas expertos *CBR*, desde un sistema que ayuda a estimar el tiempo de los viajes[1] hasta la búsqueda del camino más óptimo de un punto A a otro punto B realizado por un Robot móvil[2].
A continuación vamos a citar dos trabajos, el primero de ellos puede ser complementado por el prototipo *The Wise Guitar* y el otro es un trabajo muy parecido a este proyecto.
El primero ha sido presentado en la *The International Society For Music Information Retrieval (ISMIR)*[3]. *ISMIR* es un foro internacional que investiga sobre la organización de datos relacionados con la música, donde se realizan cada año, desde el año 2000, conferencias presentando investigaciones relacionados con el mundo de la música. El trabajo se titula *Automatic chord transcription with concurrent recognition of chord symbols and boundaries*[4], se basa en reconocimiento de acordes utilizando señales de audio reales. Este es el trabajo que puede ampliar su funcionalidad gracias a *The Wise Guitar*. A la hora de reconocer los acordes, el sistema de *The Wise Guitar* puede utilizarse para que el otro trabajo pueda mostrar la secuencia de acordes de manera óptima.

El segundo trabajo es de la *Association for the Advancement of Artificial Intelligence*, entidad dedicada al estudio de la inteligencia artificial, fue fundada en 1979 y participa en las conferencias más importantes sobre la *IA*. El trabajo se titula *Automatic Decision of Piano Fingering Based on Hidden Markov Models*[5] y su finalidad consiste en resolver, de manera automática, las digitaciones de los dedos sobre las teclas del piano basándose en una pieza musical. Este es el trabajo más parecido a *The Wise Guitar* porque comparten la optimización de las posiciones, uno para guitarra y otro para el piano.

1.4.2 Aplicaciones similares

Existe una multitud de aplicaciones de distintas plataformas que se pueden considerar similares a *The Wise Guitar*, pero la mayoría solamente son utilizadas para la obtención de información sobre posiciones de acordes, de ahí que sean empleadas para obtener sólo información sobre ello y nada más, no sirven para resolver problemas específicos del dominio como realiza *The Wise Guitar*.

Algunas de esas aplicaciones son *Chord!* [6](*Android* y *iOS*), *Guitar Chord Generator* [7] (*Web*), *Guitar Tuna* [8](*Android*, *iOS* y *Windows Phone*), *ChordBank* [9](*Android* y *iOS*), *SongSterr*[10](*Web*).

La diferencia que existe entre las aplicaciones antes citadas y el prototipo, es que estas no utilizan ningún sistema para solucionar problemas reales de los usuarios como el de las secuencias, en cambio *The Wise Guitar* ayuda al usuario a mejorar a la hora de ejecutar acordes en la guitarra.

Capítulo 2: Planificación del trabajo

En este capítulo se van a detallar los objetivos principales y secundarios que se han llevado a cabo y, a su vez, la metodología empleada en las fases de análisis y desarrollo. También se comenta las herramientas utilizadas para desarrollar el prototipo.

2.1 Objetivos

2.1.1 General

El principal objetivo para el usuario del sistema, es seleccionar una secuencia y que el sistema le devuelva, transformada, en posiciones que puedan ser ejecutadas sobre el mástil de la guitarra de manera cómoda y rápida.

2.1.2 Específicos

Los objetivos específicos son características concretas para completar el prototipo desarrollado con más funcionalidades.

1. Ver todas las posiciones de acordes disponibles en el sistema y a qué familia pertenecen.
2. Seleccionar un acorde de partida en el cual el usuario podrá ver de manera gráfica su posición y, por otro lado, seleccionar los acordes siguientes a este en orden y en formato texto.
3. Obtener una aplicación con un alto grado de usabilidad que facilita acceder a sus funcionalidades.
4. Obtener el resultado de las posiciones de acordes de manera gráfica.
5. Ver todos los casos definidos por los expertos alojados en la base de datos de manera gráfica.
6. Exportar la secuencia de acordes en formato *XML* para que el usuario pueda gestionarla posteriormente con el prototipo.
7. Importar la secuencia de acordes (*XML)* para que el usuario pueda modificarla.
8. Exportar el resultado en formato *PDF*.
9. Reproducir con sonido las posiciones del acorde que se esté visualizando para poder escucharlas.
10. Configurar el prototipo con un conjunto de parámetros para que este sea más personalizable y versátil.

2.2 Metodología

2.2.1 Especificación del problema

Como se ha indicado previamente, el problema a resolver, consiste en devolver al usuario una secuencia de posiciones de acordes, para que pueda ejecutarlo con su guitarra de manera sencilla y rápida.

Al principio, el usuario aporta al sistema una secuencia de acordes en notación musical, donde este selecciona la posición para el primer acorde de la secuencia.

Cada acorde tiene varias posiciones para ser ejecutado en la guitarra y, por eso, a la hora de cambiar de un acorde a otro puede ser una tarea costosa.

Cada usuario piensa de manera diferente a la hora de decidir las posiciones más cómodas dentro de una secuencia, por esta razón se vuelve complejo al tratarse de problemas con soluciones *subjetivas*. Por tanto, debe ser resuelto con conocimiento previo sobre qué posición se debe utilizar dependiendo de la posición predecesora, para obsequiar un resultado lo más objetivo posible.

2.2.2 Diccionario de posiciones

Se decidió que el prototipo albergara una pequeña cantidad de posiciones. No obstante, el prototipo está diseñado para que se puedan añadir nuevas posiciones y familias en la base de datos.

La forma en la que me documenté sobre las posiciones de los acordes, fue investigando en webs especializadas y apoyándome en bibliografía específica de la materia [11].
Se puede consultar el diccionario de posiciones en el Anexo 2 (Diccionario de posiciones).

2.2.3 Selección de técnicas

Al principio de la fase de análisis se barajaron dos técnicas para solucionar el problema, una de ellas era utilizar los *grafos multietapas* y la otra *CBR* (*Case Based Reasoning*)

A continuación se explican cada una de estas técnicas:

2.2.4 Grafos multietapa

Esta fue la primera idea que estudié para comprobar si era factible aplicarlo a la problemática de obtener un conjunto de acordes óptimos. El objetivo era definir un grafo al que cada vértice representase una posición concreta de un acorde. Cada etapa del grafo correspondería a un acorde de la secuencia y estaría compuesta por tantos vértices como posiciones posibles tuviera ese acorde en la guitarra. Cada vértice tendría aristas hacia todos los vértices de la siguiente etapa. Las aristas que referenciaría el camino entre dos posiciones tendrían un peso para cuantificar la dificultad de la transición entre posiciones. En la *Figura 5* se puede observar la idea de grafo multietapa de manera gráfica.

Para obtener el conjunto de acordes óptimos, se iba a implementar un algoritmo de programación dinámica que recorriera los distintos caminos hasta llegar al que tuviese el menor valor de los pesos totales.

Ventajas y desventajas:

- **Ventaja:** Búsqueda del camino más corto óptimo al utilizar algún algoritmo de programación dinámica.
- **Desventaja:** Dificultad de encontrar medidas explícitas de los pesos entre posiciones.

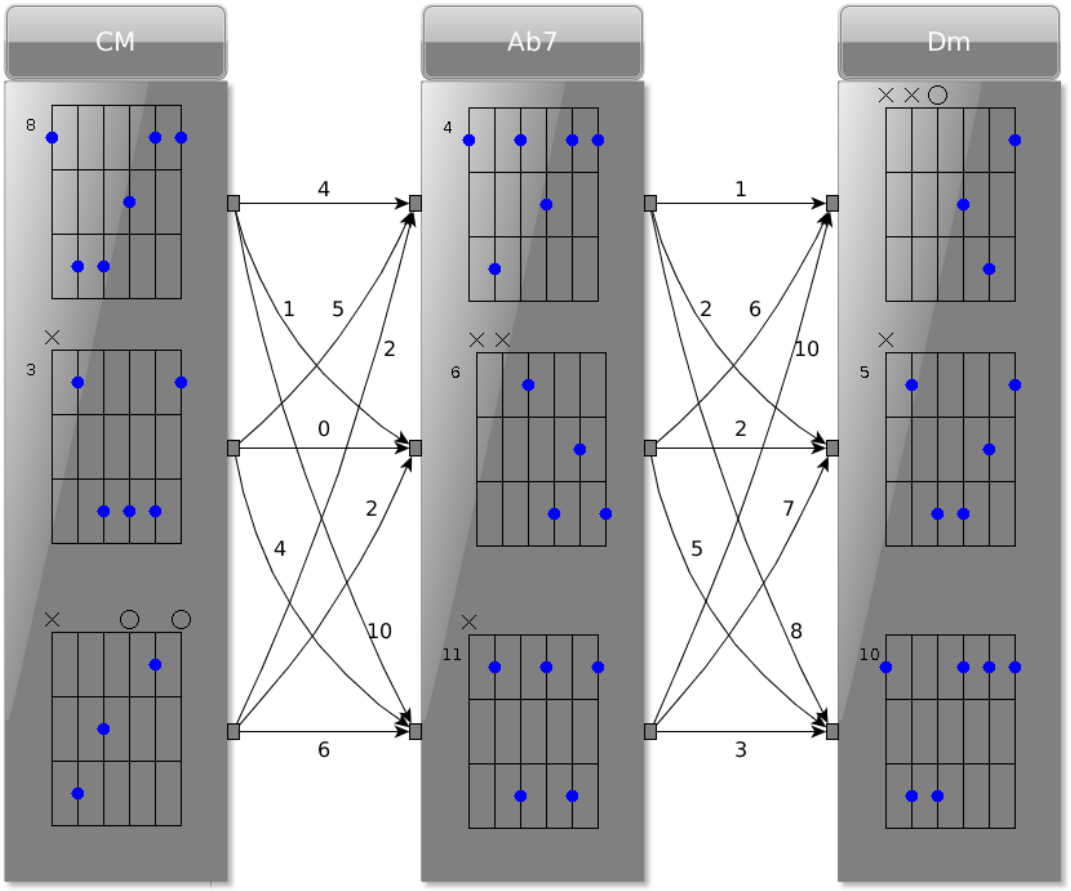


Figura 5: Grafo multietapa

2.2.5 Razonamiento basado en casos (CBR)

Case Based Reasoning es un proceso que soluciona nuevos problemas basándose en problemas ya resueltos. *CBR* es un tipo de sistema experto y, desde la perspectiva de la inteligencia artificial, es un sistema que intenta emular el comportamiento humano, experto en la materia donde se desarrolló el sistema.

Se utiliza una gran base de conocimiento realizada por expertos. El problema es descrito con datos necesarios para que el sistema los utilice y en base al conocimiento experto, devolver la secuencia solucionada al usuario.

Ventajas y desventajas:

- **Ventaja:** Existe un conocimiento implícito en los casos. No es necesario *a priori* diseñar distancias entre posiciones.
- **Desventaja:** Al utilizar casos entre dos posiciones se comportaría como un algoritmo voraz y no se puede asegurar una solución óptima. Muy dependiente de los casos definidos por expertos.

En la sección 3.1 (Razonamiento basado en casos) se explica en profundidad esta técnica.

De estas dos técnicas descritas anteriormente, se seleccionó *CBR* porque resultó ser la más adecuada de aplicar, la otra técnica no se aplicó porque las distancias entre posiciones, aristas, no son triviales, la selección de medidas de distancias se presentaba *a priori* como un problema complejo, sobre el cual no se encontró en trabajos previos. Además, la adecuada relación de estas medidas es crítica para obtener una solución adecuada.

2.2.6 Desarrollo del prototipo

El 21 de Febrero del 2014 fue la primera reunión con mi tutor, donde se comenzó a analizar el problema principal del prototipo. Cuando no nos encontrábamos de reunión, me dedicaba a investigar por Internet sobre problemas similares y, al mismo tiempo, buscaba herramientas que podían ser útiles para el desarrollo del prototipo. Posteriormente, se decidió realizar una reunión cada dos semanas, para discutir sobre cómo se estaba desarrollando el proyecto y qué cambios había que ir haciendo.

El proceso de desarrollo, ha tenido varias fases.

La primera, y la más importante, se ha basado en analizar la problemática con mi tutor y buscar qué técnicas eran las más adecuadas para solucionar los diversos problemas que suponía realizar un proyecto de estas características.

La segunda, desarrollar la aplicación de manera ágil. Cada dos semanas, se realizaba con el tutor una reunión, donde se comprobaba que las funcionalidades añadidas tenían un correcto funcionamiento al que se planteó en las reuniones anteriores y él me encomendaba nuevas tareas para desarrollar durante las dos semanas posteriores.

Una de las técnicas que ha sido utilizada durante el proceso de desarrollo fue *TDD*, *Test-Driven Development*. Primero se seleccionaba la funcionalidad que se iba a desarrollar, segundo se implementaba los tests y, por último, se desarrollaba esa funcionalidad hasta que pasara todos los tests y funcionase de la manera que se requería.

Tercera, seguir analizando el problema y comprobar, con mi tutor, que las funcionalidades desarrolladas eran las adecuadas, si daba el caso que no eran apropiadas, se volvía a analizar la característica/problemática para buscar y desarrollar soluciones alternativas.

2.3 Especificaciones y herramientas

2.3.1 Especificaciones

Las siguientes especificaciones del prototipo, explican el conjunto de características y funcionalidades que se reunieron durante la fase de análisis. Deben cumplirse cada uno de ellos para que el resultado que proporcione el prototipo sea satisfactorio, sencillo de entender y que el sistema sea fácil de utilizar.

1. **Interfaz gráfico**

- El sistema proporciona una interfaz visual, donde un usuario, tanto amateur como profesional del dominio, deberá entender sin problemas el uso de sus componentes y acciones tras utilizar el sistema poco menos de un minuto.
- El usuario al insertar una posición que no cumple con los parámetros configurables, le saldrá una ventana indicando el problema y como solucionarlo.

2. **Gestión del conjunto de acordes**

- El usuario, seleccionará el primer acorde de la secuencia y verá la posición dibujada en la interfaz.
- A la hora de añadir o quitar acordes de la secuencia, el usuario tendrá la posibilidad de hacerlo de manera intuitiva, mediante botones bien diferenciados en colores y en etiquetado. Si la secuencia tiene una cantidad inmensa de acordes, el usuario podrá eliminarlos de la secuencia en un solo paso sin perder tiempo.

- El usuario visualizará la secuencia de acordes que quiere insertar al sistema en una tabla, podrá interactuar con ella y modificar sus valores de manera sencilla mediante cajas de selección.
- El usuario podrá tener tantas ventanas de resultados abiertas como desee. El prototipo no deberá estar limitado en la cantidad de ventanas abiertas.

3. Resultado de la secuencia de acordes

- El resultado proporcionado por el sistema podrá ser visualizado por el usuario mediante una tabla e imágenes de las posiciones para que de un vistazo pueda posicionarlos en su guitarra.
- El resultado podrá ser exportado a *PDF* para que el usuario pueda visualizarlo en cualquier dispositivo sin problemas de formatos.
- El usuario podrá utilizar la acción de exportar el conjunto de acordes a *XML* para que pueda, en futuras ocasiones, importarlo al sistema para gestionar y ejecutar la secuencia tantas veces como crea necesaria.

4. Imágenes de posiciones dinámicas

- El sistema deberá crear, si no existieran, las imágenes de posiciones de acordes de manera automática. Si en un futuro el sistema soporta nuevas posiciones de acordes y nuevas familias, éste deberá crear las imágenes sin interacción humana.
- El sistema deberá almacenar *posiciones genéricas*, posiciones que se pueden ejecutar en cualquier lugar de la guitarra, no tiene asociado número de traste.

5. Sonido en las posiciones

- El usuario podrá pulsar sobre las imágenes de las posiciones para hacer sonar el acorde. El sistema deberá soportar nuevas posiciones de acordes y nuevas familias para que el usuario al pulsarlas pueda escucharlas sin problemas.

6. Casos

- El sistema deberá albergar un conjunto amplio de casos para que el subsistema *CBR* funcione de manera correcta.

- El usuario, podrá ver los casos de manera gráfica y con la suficiente información para entender cada caso. El usuario, tendrá que poseer un nivel básico de conocimiento sobre teoría musical.

2.3.2 Herramientas

Para el desarrollo del prototipo, se han utilizado varias herramientas para llegar a los objetivos especificados.

2.3.2.1 Java

Java[12] ha sido el lenguaje principal de este trabajo, versión *v1.7.0_21,* por dos motivos importantes, primero porque se ha empleado un framework *CBR* basado en *Java* llamado *jColibri* y, segundo, por el paradigma de programación orientada a objetos, usado en el desarrollo del prototipo.

Las ventajas de utilizar este lenguaje es la inmensa cantidad que existe de documentación y librerías, tanto oficiales como de terceros. Es multiplataforma, por lo que podemos hacer uso de los proyectos en casi cualquier sistema operativo, dándonos total libertad a la hora de utilizar el prototipo en diferentes plataformas.

Además, se han incorporado varias librerías de *Java* para diseñar la Interfaz gráfica, aplicar sonido a los acordes, exportar el resultado a *PDF* y exportar e importar en formato *XML* la secuencia de acordes, testear la aplicación mediante test unitarios, entre otras tareas de desarrollo.

A continuación, se detallan brevemente las librerías utilizadas.

2.3.2.1.1 AWT y Swing

AWT[13](*Abstract Window Toolkit*) y *Swing* son librerías de *Java* que han sido útiles para el diseño de la interfaz gráfica del prototipo. Se ha hecho uso de sus componentes como botones, tablas, etiquetas. Se ha tenido en cuenta la usabilidad de la interfaz, para que al usuario le resulte intuitiva y cómoda. Para el diseño de la interfaz se han utilizado patrones de diseño centrados en el usuario, como colores bien definidos, iconos representativos, interfaces minimalistas y los elementos necesarios en cada vista.

2.3.2.1.2 JFugue

JFugue[14] es una librería *open-source* para *Java*, que ayuda a abstraer las complejidades del *MIDI (Musical Instrument Digital Interface)*. Tiene utilidades que

permiten especificar acordes, instrumentos, notas, reproducir música en tiempo de ejecución, guardar y abrir archivos *MIDI*, entre otras funcionalidades.

El prototipo sólo ha utilizado de esta librería, concretamente las características de *MusicString*[15], para hacer sonar las posiciones de los acordes al pulsar sobre ellos cuando están dibujados en la interfaz.

2.3.2.1.3 iText

iText[16] es una librería *open-source* para *Java* que ayuda a crear y manipular documentos *PDF*. El prototipo hace uso de sus características para exportar los resultados a ese formato.

2.3.2.1.4 DOM

DOM[17], *Document Object Model*, es un paquete de *Java* para definir estructuras de documentos *HTML* y *XML*, para su posterior manipulación.

El prototipo lo utiliza para construir y manipular la secuencia de acordes.

2.3.2.1.5 Properties

Representa un conjunto de propiedades persistentes en un fichero y utiliza la filosofía de *clave-valor* para su gestión.

Se han utilizado dos archivos *.properties*, uno para tener bajo control los mensajes de error y de información, y otro para que el usuario pueda configurar varios parámetros del prototipo, como el número de traste límite del mástil de la guitarra, la afinación y los pesos de los atributos del *CBR*.

2.3.2.1.6 JUnit

Para la realización de los test del prototipo, se ha utilizado la librería *JUnit* para implementar los test unitarios. Se han empleado *stubs* y *mocks* para probar métodos que tienen dependencias externas y para los tests de la capa *DAO*, *Data Access Object*, se ha tenido en cuenta el uso de una base de datos de pre-producción. (No utiliza la base de datos de producción para realizar los tests)

La metodología utilizada para los tests, ha consistido en pruebas de caja blanca, ya que se seleccionaba valores de entrada para examinar cada uno de los posibles flujos de ejecución y se comprobaba que devolvían los valores de salida adecuados.

2.3.2.2 Sqlite

Sqlite[18] es un sistema de gestión de bases de datos relacional. Se ha utilizado este tipo de base de datos porque realiza llamadas simples a subrutinas y funciones, reduciendo la latencia cuando se acceden a los datos, consiguiendo ser muy ágil.

Uno de los inconvenientes, es que no utiliza ningún usuario para acceder a la BD, por consiguiente no tiene ninguna seguridad en los datos, ya que puede acceder cualquiera. Esta BD se suele utilizar en la fase de desarrollo y no cuando la aplicación cambia a producción. Pero en este prototipo no habría ningún problema, ya que no almacena datos sensibles por lo que no haría falta aplicar la *Ley orgánica de Protección de Datos (LOPD).*

2.3.2.3 jColibri

jColibri[19] es un *framework open-source* para el desarrollo de aplicaciones basadas en *CBR, Case Based Reasoning*. Define una estructura clara para diseñar sistemas *CBR*, con la ventaja de ser extensible y reutilizable para diferentes dominios. *jColibri* puede ser utilizado tanto para prototipos experimentales, como para aplicaciones que son desplegadas en escenarios reales.

Esta es la principal herramienta de desarrollo del prototipo. Se ha elegido por su facilidad de uso y por su amplia documentación[20]. La versión utilizada de *jColibri* es la *v.2.3* y se ha utilizado *ColibriStudio v1.0.0*, un entorno de desarrollo basado en *Eclipse* que ayuda a generar el código suficiente para hacer uso de las librerías de *jColibri*.

A continuación se explican algunas de sus características.

- **Capa de persistencia:** Los casos se pueden guardar en varios medios, como bases de datos, archivos de texto, archivos *ARFF*.
- **Organización de los casos:** Los casos se pueden guardar en distintas estructuras, como listas lineales, árboles, mapas, entre otros.
- **Métodos de recuperación:** *jColibri* utiliza *k Nearest Neighbor,* un algoritmo de clasificación supervisada que determina la probabilidad *a posteriori* de que un elemento *x* pertenezca a una clase C_j, a partir de la información proporcionada por un conjunto de atributos. Utiliza funciones de similitud globales para comparar atributos.

- **Editor:** *jColibri* dispone de un editor sencillo que abstrae la dificultad para definir los casos, las medidas de los atributos, así como la estructura de como se guardan los casos.

2.3.2.4 Git y Bitbucket

Para el sistema de control de versiones, *CVS*, se ha utilizado *Git*[21]. Es una herramienta muy útil, que sirve para conocer en todo momento en qué estado se encuentra el proyecto y tener un control sobre las características que se añaden y/o modifican. Por otro lado, al ser un sistema de control de versiones distribuido, otorga la confianza de añadir y/o modificar alguna característica al repositorio local, con lo que se comprueba, antes de subir los cambios al repositorio remoto, que no exista ningún error.

Para almacenar los cambios del repositorio se ha utilizado *Bitbucket*[22], un servidor basado en web, que me permitió compartir el proyecto con el tutor, para que pudiera observar en todo momento qué funcionalidades/cambios se iban realizando en el prototipo durante la fase de desarrollo.

Capítulo 3: Solución al problema aplicando un sistema experto

Este capítulo, ilustra el sistema principal del prototipo y explica el sistema utilizado, así como su aplicación para la solución del problema, con ejemplos.

Por último, se detallan otras características del proyecto, como otras funcionalidades e interfaces utilizadas.

3.1 Razonamiento basado en casos

El razonamiento basado en casos, *CBR*, en inglés, es una técnica que resuelve nuevos problemas mediante soluciones ya resueltas (Soluciones similares). La idea principal es la de utilizar las experiencias previas para los nuevos problemas, basándose en el concepto de similitud.

3.1.1 Casos

Los casos representan historias completas, es decir, un problema y su solución. El *CBR* las utiliza para resolver los nuevos problemas.

Para la descripción y solución de un caso se definen unos atributos, el *CBR* los usa para conocer a qué caso conocido se asemeja más un nuevo problema.

Los atributos aparte de su valor, utiliza unos pesos para indicar su importancia entre el conjunto de atributos de la descripción.

Por ejemplo, los atributos de la descripción de un taller de coches podrían ser, *modelo del coche*, *año de fabricación*, *cilindros del motor*, *caballos* y, para la solución, *diagnóstico y reparación.* Por otro lado, se puede añadir una explicación o comentario sobre la calidad de la solución, para hacer de él un caso más completo, útil e informativo.

Existen varias maneras de representar los casos, dependiendo del dominio del sistema y cómo queramos estructurarlos. Por ejemplo, se puede representar su estructura en un grupo de Clases (representaciones orientadas a objetos).

El conjunto de los casos, se denomina *casos base*, dicho con otras palabras, una base de datos de casos. Si en la base de datos disponemos de muchos casos, entonces aumentará la probabilidad de que se encuentre una solución adecuada al problema que se esté analizando.

3.1.2 Similitud

CBR utiliza una función de similitud entre casos, basada en medidas locales para comparar los atributos de los casos.

La similitud es la función esencial utilizada en la fase *Retrieval*, para la recuperación del caso más similar, véase la sección 3.1.3 (Fases).

La medida de similitud, es el elemento central para desplazarse por el espacio de posibles soluciones.

Hay varias formas de representar la similitud. La más utilizada define pesos por cada atributo de la descripción del caso, normalmente estos pesos están comprendidos entre 0 y 1. Los pesos indican la importancia relativa de cada atributo en la función de similitud.

Existen dos tipos de similitud: local y global.

La similitud local, entre atributos, se calcula de varias formas, dependiendo del tipo de atributo a comparar. Pueden ser: numéricos, cadenas, multivaluados.

La similitud global, entre casos, combina los resultados de las similitudes locales, aplicando una función de distancia euclidiana, de Minkowski, o la media, por ejemplo. Una función de similitud típica puede ser la siguiente.

$$SIM(C1, C2) = \sqrt{\sum_{i=1}^{n} v_i^2 \times w_i}$$

- **C1:** Descripción del caso 1.
- **C2:** Descripción del caso 2.
- **n:** Número total de atributos.
- $\mathbf{v_i}$: Valor del atributo i-ésimo.
- $\mathbf{w_i}$: Peso normalizado del i-ésimo atributo.
    - $w_i \frac{p_j}{\sum_{i=1}^{n} p_i}$
    - $\mathbf{p_i}$**:** Peso del atributo
    - **n:** Número total de atributos

3.1.3 Fases

CBR consta de 4 fases, llamadas *Retrieve* (Recuperar), *Reuse* (Reutilizar), *Revise* (Revisar) y *Retain* (Conservar), como se muestra en la *Figura 6*.

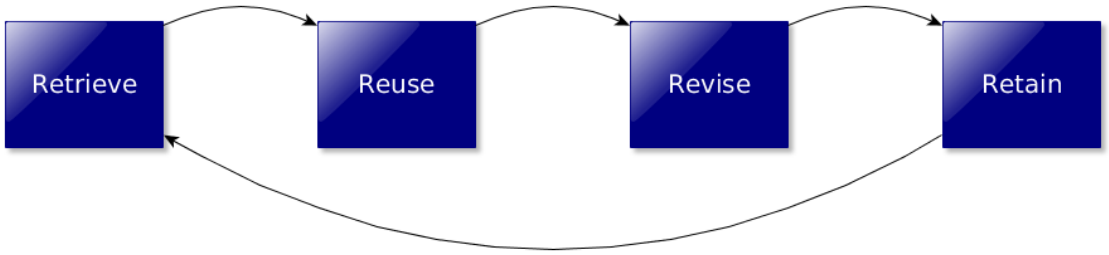


Figura 6: Fases CBR

A continuación se describe cada una de estas fases.

3.1.3.1 Retrieve

La primera fase, recupera el caso más similar entre los *casos base*. Antes de aplicar la fase *Retrieve* debe generarse una consulta con un problema específico, normalmente denominado *query*. Una *query* no es más que un problema con una descripción específica, donde cada atributo tiene su valor. El algoritmo *k Nearest Neighbor* utiliza los pesos de cada atributo para realizar los cálculos necesarios y obtener los casos más similares a la *query*.

3.1.3.2 Reuse

La segunda fase, utiliza el caso más similar hallado en la fase *Retrieve* para que, posteriormente, sea adaptado como solución.

Si el nuevo problema, *query*, es exactamente igual que el caso más similar, la adaptación del caso se haría de forma sencilla, asociando la vieja solución como solución de la *query*. Sin embargo, es raro utilizar una solución tal y como se registró en los *casos base*. Eso sucede, si la *query* no es muy diferente en los aspectos esenciales al caso más similar hallado en la fase *Retrieve*. Entonces, la recomendación es usar la solución sin adaptar.

La adaptación en un sistema *CBR* depende del dominio del problema, quiere decir que puede haber distintas formas de utilizar un caso para adaptarlo al nuevo problema.

Por ejemplo, tenemos un gimnasio y un sistema que asigna un plan de ejercicios a los clientes. Uno de ellos tiene 25 años, pesa 75 Kg, varón y tiene una pequeña lesión en la articulación de la muñeca derecha. En la base de datos de casos del sistema hay un caso similar, en la solución del caso aparece un ejercicio que el cliente no puede realizar por su lesión, por lo que el sistema debe adaptarla para que ese ejercicio no aparezca o sea cambiado por otro que pueda realizar el cliente sin problemas.

3.1.3.3 Revise

La tercera fase, evalúa la aplicabilidad de la solución propuesta (la *query* con la solución adaptada) en un contexto real. Se comprueba que la solución es válida y que soluciona el problema que se planteó.

3.1.3.4 Retain

La cuarta fase, y última del ciclo *CBR*, también conocida como *fase de aprendizaje*, se encarga de retener el caso adaptado y evaluado en los *casos base*. Si la revisión es satisfactoria, se guarda como un nuevo caso, si no fuera así, también se guarda en la base de conocimiento para que el sistema conozca soluciones erróneas para no producirlas en el futuro, así se permite una mejora en la calidad de las soluciones.

Realmente esta fase es la encargada de que la base de conocimiento crezca y evolucione.

3.1.4 Ejemplo

En el siguiente ejemplo, imaginamos un sistema meteorológico que nos pronostique el porcentaje de precipitación. Para hacer el ejemplo más sencillo, la base de casos tiene solamente dos casos, descritos en la *Tabla 2* y *Tabla 3*.

CASO 1		
Descripción		**Solución**
Velocidad del viento (Km/h)	12 Km/h	- 0% de probabilidad de precipitaciones
Presión atmosférica (mb)	1012 mb	
Humedad (%)	64 %	
Temperatura (ºC)	27 ºC	

Tabla 2: Caso 1 de ejemplo

CASO 2		
Descripción		**Solución**
Velocidad del viento (Km/h)	5 Km/h	- 78% de probabilidad de precipitaciones
Presión atmosférica (mb)	998 mb	
Humedad (%)	86 %	
Temperatura (ºC)	13 ºC	

Tabla 3: Caso 2 de ejemplo

Ahora, se definen los pesos para cada atributo, *Tabla 4*. Los pesos son fundamentales, por lo que sería recomendable que los definiera personas expertas del dominio.

Atributos	Pesos
***a1*: Velocidad del viento**	0.3
***a2*: Presión atmosférica**	0.8
***a3*: Humedad**	0.6
***a4*: Temperatura**	0.5
	Total=2.2

Tabla 4: Pesos de los atributos de ejemplo

Por último, se decide la similitud local para cada atributo. Al ser numéricos, aprovechamos la misma medida para todos los atributos, su valor oscila entre 0 y 1 e indica la similitud entre los valores de un atributo de la *query* y de un caso. Cuando el valor es igual o está cerca a 1 indica un alto grado de similitud.

Para los atributos ***a1, a2*** y ***a4***:

$$\begin{cases} Caso.a_i \geq query.a_i \rightarrow SIMLOCAL(Caso.a_i, query.a_i) = \dfrac{query.a_i}{Caso.a_i} \\ Caso.a_i < query.a_i \rightarrow SIMLOCAL(Caso.a_i, query.a_i) = \dfrac{Caso.a_i}{query.a_i} \end{cases}$$

Para el atributo ***a3***:

$$SIMLOCAL(Caso.a_i, query.a_i) = \frac{|Caso.a_i - query.a_i|}{100}$$

En la *Tabla 5*, tenemos la *query* que ha entrado al sistema.

query1	
Descripción	
Velocidad del viento (Km/h)	5 Km/h
Presión atmosférica (mb)	998 mb
Humedad (%)	86 %
Temperatura (ºC)	13 ºC

Tabla 5: query1 de ejemplo

En la fase *Retrieve*, se compara la *query1* con todos los casos de la base de datos de casos. De esta forma, se puede observar que el problema indicado por la *query1* ya existe en la base de datos (es el mismo que el Caso 1), por lo que el sistema no adapta la solución y no va aprender el caso, ya que se encuentra en los *casos base* y se ha decidido que no se repita ningún caso.

Ahora tenemos la siguiente consulta, *query2*, *Tabla 6*, donde observamos que el problema nunca ha sido solucionado y por lo tanto no existe en la base de casos.

query2	
Descripción	
Velocidad del viento (Km/h)	23 Km/h
Presión atmosférica (mb)	1002 mb
Humedad (%)	54 %
Temperatura (ºC)	18 ºC

Tabla 6: query2 de ejemplo

En la fase *Retrieve*, se realizan los cálculos con todos los *casos base*. En la *Tabla 7* tenemos la información de los cálculos de las similitudes locales.

Atributos	simlocal(query2, Caso 1)	simlocal(query2, Caso 2)
Velocidad del viento	12/23=0.52	5/23=0.21
Presión atmosférica	1002/1012=0.99	998/1002=0.99
Humedad	(|54-64|)/100= 0.1	(|54-84|)/100= 0.32
Temperatura	18/27=0.66	13/18=0.72

Tabla 7: Cálculos de las similitudes locales de ejemplo

Se calcula las similitudes globales utilizando las similitudes locales.

$$SIM(query2, Caso\ 1) = \sqrt{0.52^2 \times \frac{0.3}{2.2} + 0.99^2 \times \frac{0.8}{2.2} + 0.1^2 \times \frac{0.6}{2.2} + 0.66^2 \times \frac{0.5}{2.2}} = 0.7035$$

$$SIM(query2, Caso\ 2) = \sqrt{0.21^2 \times \frac{0.3}{2.2} + 0.99^2 \times \frac{0.8}{2.2} + 0.32^2 \times \frac{0.6}{2.2} + 0.72^2 \times \frac{0.5}{2.2}} = 0.7128$$

Por el resultado, se observa que el Caso 2 es más similar a la *query2*, por lo que en la fase *Reuse* se comprueba si es necesario adaptar la solución o asociar directamente la solución del Caso 2 al nuevo caso.

Por lo tanto, y para hacerlo más sencillo, se asocia la solución con el nuevo caso, *Tabla 8*.

CASO 3		
Descripción		**Solución**
Velocidad del viento (Km/h)	23 Km/h	- 78% de probabilidad de precipitaciones
Presión atmosférica (mb)	1002 mb	
Humedad (%)	54 %	
Temperatura (ºC)	18 ºC	

Tabla 8: Caso 3 de ejemplo

En la fase *Revise* se revisa si la adaptación y solución, para el nuevo caso, es correcta para un ámbito real. En este ejemplo damos por hecho que es correcto.

Por último, en la fase *Retain* se aprende el caso, añadiéndolo en la base de datos de casos.

3.2 Aplicación del CBR

A la hora de aplicar el sistema *CBR* al problema, se decidió cambiar la fase *Reuse* por *Revise*. Se realizó esa modificación en el flujo para que el sistema tuviera menos carga evitando la fase de adaptación, ya que si al revisar los datos no son correctos volvería a la fase anterior, *Retrieve*, para seleccionar el siguiente caso más similar.

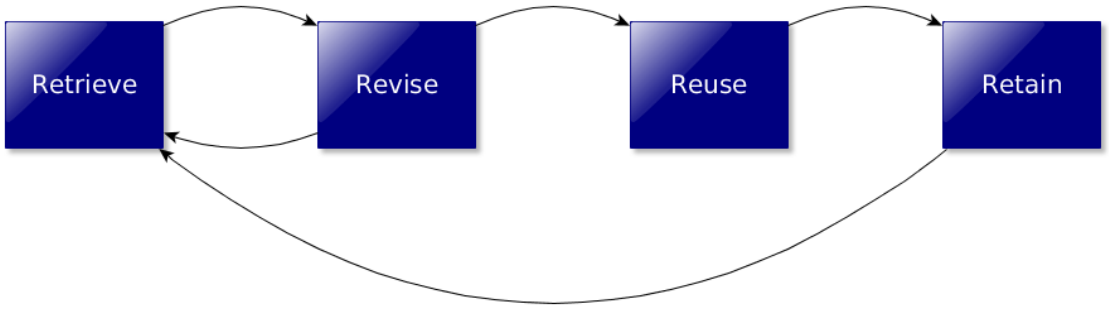


Figura 7: Fases CBR del prototipo

No hay ningún problema en cambiar una fase por otra, *CBR* te da total libertad para adaptar el sistema a las necesidades del proyecto.

En la *Figura 7* se puede observar el flujo de *CBR* que utiliza el prototipo.

El *CBR* actúa en el prototipo para resolver las transiciones entre dos acordes consecutivos de la secuencia de entrada sin tener en cuenta otro contexto, comportándose como un algoritmo voraz. Quiere decir, que resuelve los problemas de dos en dos, y la solución del caso sirve para definir el siguiente y así hasta terminar la secuencia. Se puede ver un ejemplo en la *Figura 8*.

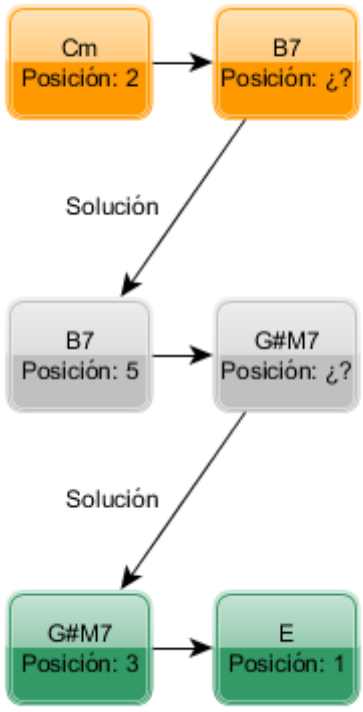


Figura 8: Transiciones en la secuencia de acordes

3.2.1 Representación de casos

La representación de un caso, ha sido una de las tareas más complejas de la fase de análisis, es fundamental en sistemas *CBR* y no tiene que cambiar durante la fase de desarrollo. Por ese motivo, se le ha dedicado el tiempo necesario para definirla de forma correcta. Un caso representa la transición entre dos acordes en la guitarra y se ha definido para que sea independiente de la fundamental de los acordes.

Como se puede ver en la *Figura 9*, el caso está definido por dos acordes y tiene un total de 6 atributos. Del atributo 1 al 4 representa la descripción del caso y del 5 al 6 la solución.

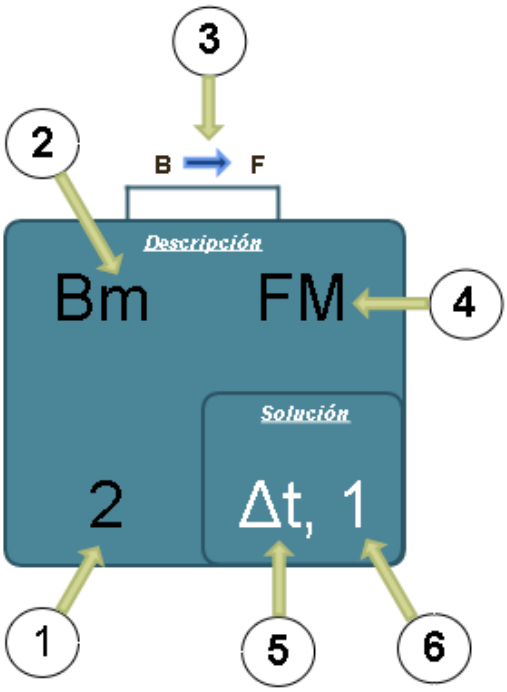


Figura 9: Representación de un caso

A continuación se explica cada uno de ellos.

Descripción

1. **F_C_Type (*First Chord Type*):** La posición del primer acorde, representado mediante una clave que indica qué posición es, dentro de la BD de posiciones.
2. **F_C_Family (*First Chord Family*):** La familia del primer acorde. (Mayor, menor, etc)
3. **S_C_Interval (*Second Chord Interval*):** El intervalo de la fundamental del primer acorde a la fundamental del segundo en semitonos. No se almacenan las fundamentales de los acordes, sino el intervalo entre ellas. El rango es de [0, 6], porque si se calculan las distancias entre fundamentales, la distancia máxima es de 6 semitonos. Para una mayor aclaración, véase la sección 3.2.2.2. (Similitud entre intervalos)

4. **S_C_Family (*Second Chord Family*):** Familia del segundo acorde.

Solución

5. **FS_Delta_Fret (*First & Second Delta Fret*):** Distancia en trastes, entre el primer y el segundo acorde. Si la distancia es negativa, indica que hay que desplazarse hacia arriba del mástil, si es 0 no hay que desplazarse y si es positivo hay que desplazarse hacia abajo del mástil.
6. **S_C_Type (*Second Chord Type*):** La posición del segundo acorde representado mediante una clave que indica qué posición es, dentro de la BD de posiciones.

Hay que tener en cuenta, que la descripción y solución tienen la posición del primer y segundo acorde. Por ello, es necesario conocer en qué traste se ejecutan las dos posiciones, por lo que se han creado cuatro atributos que no se encuentran definidos en el caso, *atributos derivados*. A partir de estos, se calculan *S_C_Interval* y *FS_Delta_Fret*.

Los *atributos derivados* son:

- ***F_C_Root (First Chord Root)*:** La fundamental del primer acorde.
- ***S_C_Root (Second Chord Root)*:** La fundamental del segundo acorde.
- ***F_C_Fret* (*Frist Chord Fret*):** Número de traste de la posición del primer acorde.
- ***S_C_Fret* (*Second Chord Fret*):** Número de traste de la posición del segundo acorde.

Entre las fundamentales y las posiciones de los acordes, se calcula el número de traste para cada posición.

También, sirven para calcular el atributo *S_C_Interval*, necesario para las similitudes locales.

3.2.2 Similitudes locales y Similitud global

Las similitudes locales y global son necesarias para que el sistema pueda realizar los cálculos necesarios para comprobar qué grado de similitud existe entre dos casos o un caso y una *query*.

Para calcular la similitud global, se ha utilizado la distancia euclídea, véase la sección 3.1.2. (Similitud)

$$SIM(C1, C2) = \sqrt{\sum_{i=1}^{n} v_i^2 \times w_i}$$

A continuación se detallan las medidas de similitud locales.

3.2.2.1 Similitud entre familias

Las familias se representan con una cadena de texto, por ejemplo *M*, *M7*, *7* y m7. Al no poder realizar cálculos matemáticos con estos valores, se ha definido una *matriz de similitud entre familias*, *Tabla 9*.

	M	**m**	**M7**	**m7**	**7**
M	1	0.2	0.8	0	0.2
m	0.2	1	0	0.8	0
M7	0.8	0	1	0	0.2
m7	0	0.8	0	1	0
7	0.2	0	0.2	0	1

Tabla 9: Matriz de similitud entre familias

Esta matriz es definida con 4 pesos *0*, *0.2*, *0.8 y 1*. Para obtenerlos, se compara un intervalo de una familia con todas las demás.

A continuación, se puede ver en la *Tabla 10* la distancia en semitonos desde la fundamental para cada familia.

	3 (Tercera)	5 (Quinta)	7 (Séptima)
M	4	7	-
m	3	7	-
M7	4	7	11
M7	3	7	10
7	4	7	10

Tabla 10: Número de semitonos desde la fundamental para cada familia

Las comparaciones, siguen cuatro requisitos, a continuación se explican cada uno de ellos.

- **Primer requisito** (Valor=1): Las familias comparadas son las mismas, por lo que sus intervalos son iguales.
- **Segundo requisito** (Valor=0.8): Las familias comparadas, se distinguen porque una de ellas tiene un intervalo más respecto a la otra familia y los intervalos que comparten no sufren ninguna alteración. Por ejemplo, el intervalo de los Mayores (*M*) Fundamental, 4 y 7 y el intervalo de Mayor Séptima (*M7*) Fundamental, 4, 7 y **11**.
- **Tercer requisito** (Valor=0.2): Las familias comparadas tienen los mismos intervalos, pero en alguno de ellos sufre alguna alteración. Por ejemplo, el intervalo de los Mayores (*M*) Fundamental, **4** y 7 y el intervalo de los Menores (*m*) Fundamental, **3** y 7.
- **Cuarto requisito** (Valor=0): Las familias comparadas, se distinguen porque una de ellas tiene una nota más y algunos de sus intervalos que comparten sufre alguna alteración. Por ejemplo, el intervalo de los Mayores (*M*) Fundamental, **4** y 7 y el intervalo de Menor Séptima (*m7*) Fundamental, **3**, 7 y **10**.

3.2.2.2 Similitud entre intervalos

La similitud local para los intervalos, es una medida que oscila entre 0 y 1.

$$SIMINTERVALO(Caso.a_i, query.a_i) = MaxST - \frac{|query.a_i - Caso.a_i|}{MaxST}$$

El valor de la constante *MaxST* es 6, porque la distancia más corta entre dos notas dentro de una octava se encuentra entre 0 y 6. Por ejemplo, tenemos las siguientes notas: Si(*B*) y Fa(*F*). Al calcular las distancias obtenemos que de Si(*B*) a Fa(*F*) el resultado es de 6 semitonos y de Fa(*F*) a Si(*B*) es de 6 semitonos, en este caso son los mismos valores por lo que da igual cuál escoger. Veamos otro ejemplo, ahora tenemos Re(*D*) y Sol(*G*), al calcular sus distancias obtenemos que de Re(*D*) a Sol(*G*) el resultado es de 5 semitonos y de Sol(*G*) a Re(*D*) es de 7 semitonos, entre los dos valores seleccionamos el menor de ellos, que es 5.

A nivel teórico musical, las distancias se calculan posicionándose sobre la nota en el pentagrama y contando los semitonos hasta encontrar la siguiente nota que se está buscando. Hay dos distancias, una calcula los semitonos yendo hacia arriba del pentagrama y la otra hacia abajo, en la *Figura 10* se puede ver un ejemplo de distancias entre Sol(*G*) y Mi(*E*).

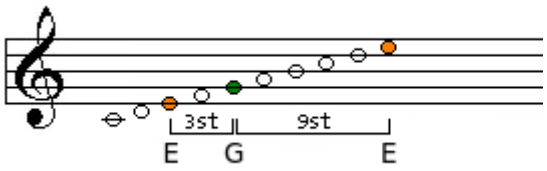


Figura 10: Distancias en semitonos entre Sol(G) y Mi(E)

A continuación vamos a realizar un ejemplo.

Tenemos dos casos. El Caso1 tiene como valor 2, el Caso2 el valor de 3 y la *query* tiene el valor de 1.

Realizamos los cálculos:

$$SIMINTERVAL(Caso1, query) = \frac{6 - |1 - 2|}{6} = 0.83$$

$$SIMINTERVAL(Caso2, query) = \frac{6 - |1 - 3|}{6} = 0.66$$

Se puede observar que el intervalo del Caso1 es más similar al intervalo de la *query*, ya que el intervalo de la *query* está más cerca al del Caso1.

3.2.2.3 Similitud entre tipos

El cálculo de la similitud local para las posiciones, se realiza de una manera más simple que las medidas locales anteriores.

- Cuando las posiciones son idénticas, es decir tienen la misma familia y es el mismo tipo dentro de esa familia, el resultado que devuelve es 1.
- Si las posiciones son de la misma familia, pero son posiciones distintas, devuelve 0.5.
- Cuando las posiciones son de distinta familia devuelve 0.

3.2.3 Adaptación de la solución

Para adaptar la solución, es necesario la descripción de la *query*, la solución del caso más similar y los atributos *F_C_Fret* y *S_C_Fret*.

La adaptación consiste en incorporar *S_C_Type* y calcular *FS_Delta_Fret*.

Los pasos a seguir para la adaptación son los siguientes:

1. **Asociar *S_C_*Type al *query:*** Se asocia a la *query* la posición del segundo acorde, el valor del atributo *S_C_Type* del caso más similar.
2. **Calcular F_*C_Fret* y *S_C_Fret:*** Se calculan los trastes, donde se posicionan los acordes sobre el mástil, del primer y segundo acorde.
3. **Calcular *FS_Delta_Fret:*** Por último, se calcula *FS_Delta_Fret*, restando el traste del segundo acorde con el traste del primero, *S_C_Fret - F_C_Fret*, para hallar la distancia, número de traste que lo separa, entre el primer y segundo acorde.

3.2.4 Ejemplo

A continuación se va a realizar un ejemplo real del comportamiento del sistema *CBR* que se ha aplicado al prototipo.

Los pesos que se han definido para el ejemplo se pueden observar en la *Tabla 11*.

Atributos	Pesos
F_C_Type	0.20
F_C_Family	0.70
S_C_Interval	0.20
S_C_Family	1
	Total=2.1

Tabla 11: Pesos de los atributos de ejemplo

En la base de datos de casos vamos a tener sólo dos casos, los cuales utilizan la información de la *Tabla 12* y *Tabla 13*. La nomenclatura que aparece en el atributo *F_C_Type* y *S_C_Type*¸ es el formato que utiliza las posiciones para diferenciarse dentro de la base de datos de posiciones, por ejemplo: t_M_3, quiere decir que la familia es *M* (Mayor) y el número de posición es 3. Puedes ver cada una de ellas en el Anexo 2 (Diccionario de posiciones).

CASO 1			
Descripción		**Solución**	
F_C_Type	t_M_3	**FS_Delta_Fret**	0
F_C_Family	M	**S_C_Type**	t_m_4
S_C_Interval	5		
S_C_Family	m		

Tabla 12: Caso 1 - Aplicación CBR

CASO 2			
Descripción		**Solución**	
F_C_Type	t_M_3	**FS_Delta_Fret**	2
F_C_Family	M	**S_C_Type**	t_m_1
S_C_Interval	1		
S_C_Family	m		

Tabla 13: Caso 2 - Aplicación CBR

En la *Tabla 14*, tenemos la *query* que ha entrado al sistema.

query	
Descripción	
F_C_Type	t_M_1
F_C_Family	M
S_C_Interval	¿?
S_C_Family	m
F_C_Root	A
S_C_Root	C#

Tabla 14: query - Aplicación CBR

Como se puede observar, el atributo *S_C_Interval* es desconocido hasta que se calcule el intervalo entre las fundamentales de los dos acordes, *F_C_Root* y *S_C_Root*.

1. Se calcula el atributo *S_C_Interval* con las notas *F_C_Root* y *S_C_Root*, el resultado que nos da es 4, que es la distancia más corta entre *A* y *C#*.
2. En el siguiente paso, el sistema calcula el traste del primer acorde, se utiliza para ello la posición (*F_C_Type*) y la nota fundamental (*F_C_Root*).
   El resultado que obtenemos para *F_C_Fret* es 5. Tenemos el acorde A con el tipo t_M_1 y se coloca en el traste 5.
3. Se realiza la fase *Retrieve* para calcular las similitudes, *Tabla 15*.

Atributos	simlocal(query, Caso 1)	simlocal(query, Caso 2)
F_C_Type	SIMTYPE(query, Caso1) = 0.5	SIMTYPE(query, Caso2) = 0.5
F_C_Family	SIMFAMILY(query, Caso1) = 1	SIMFAMILY(query, Caso1) = 1
S_C_Interval	SIMINTERVAL(query, Caso1) = (6 - |4 - 5|)/6 = 0.83	SIMINTERVAL(query, Caso2) = (6 - |4 - 1|)/6 = 0.5
S_C_Family	SIMFAMILY(query, Caso1) = 1	SIMFAMILY(query, Caso1) = 1

Tabla 15: Cálculos de las similitudes locales - Aplicación CBR

Al tener las similitudes locales, se calcula la similitud global.

$$SIM(query, Caso\ 1) = \sqrt{0.5^2 \times \frac{0.2}{2.1} + 1^2 \times \frac{0.7}{2.1} + 0.83^2 \times \frac{0.2}{2.1} + 1^2 \times \frac{1}{2.1}} = 0.9481$$

$$SIM(query, Caso\ 2) = \sqrt{0.5^2 \times \frac{0.2}{2.1} + 1^2 \times \frac{0.7}{2.1} + 0.5^2 \times \frac{0.2}{2.1} + 1^2 \times \frac{1}{2.1}} = 0.9259$$

El resultado, en la fase *Retrieve*, del caso similar es el Caso1.

4. Ahora se realiza la fase *Revise*.
   Antes de hacer las comprobaciones, el sistema calcula el traste para colocar el segundo acorde sobre el mástil, utiliza para ello la posición (*S_C_Type*) y la nota fundamental (*S_C_Root*).
   El resultado que obtenemos para *S_C_Fret* es 9. Ahora tenemos que el segundo acorde C#m se coloca en el traste 9 y utiliza la posición t_m_4.

   Para que el Caso1 sea el candidato apropiado y que el sistema pueda utilizar su solución para la *query*, tiene que cumplir los siguientes requisitos:

   1. Se comprueba que la familia del segundo acorde, *S_C_Family*, del Caso 1 y de la *query*, sean la misma. No tiene sentido dar una solución, donde la familia no corresponde, se estaría entregando una solución errónea. Si se diera el caso, se escogería el siguiente caso más similar.
   2. Por último, se comprueba que los acordes son ejecutados en lugares correctos del mástil, puede darse el caso que una de las notas del acorde a la hora de pulsarla sobre el mástil, sobresalga de este, por lo tanto sería una posición y una solución incorrecta. Algunas posiciones sólo se pueden utilizar en el primer traste de la guitarra, ya que son imposibles de ejecutar en otro sitio del mástil.

   En este ejemplo, el Caso1 cumple con los dos requisitos, por lo que las comprobaciones en la fase *Revise* son correctas y dan paso a la adaptación.

5. Después de que el sistema ha revisado y seleccionado el caso, se ha de adaptar la solución en la fase *Reuse*.
   En esta fase, se calcula el atributo *FS_Delta_Fret*, restando los trastes del segundo acorde, con el primero.
   FS_Delta_Fret = *S_C_Fret* – *F_C_Fret* = 9 - 5 = 4.

6. Por último, se pone en marcha la fase *Retain*, donde se guarda el nuevo caso con la solución adaptada, *Tabla 16.*

CASO 3			
Descripción		**Solución**	
F_C_Type	t_M_1	**FS_Delta_Fret**	4
F_C_Family	M	**S_C_Type**	t_m_4
S_C_Interval	4		
S_C_Family	m		

Tabla 16: Caso 3 - Aplicación CBR

Con esto se termina el ejemplo, pero hay que explicar un último detalle. Qué sucede cuando el usuario inserta una secuencia de acordes para solucionar, no sólo hay una *query* sino varias.

Vamos a explicarlo con un pequeño ejemplo:

Se da una secuencia de seis acordes (*Figura 11)*, tenemos una primera *query* llamada *query1*, está representada por una descripción y las notas fundamentales del primer y segundo acorde. Cuando se soluciona la *query1* pasa a ser un caso con descripción y solución, entonces para la *query2* el sistema construye su descripción a partir de los atributos de la solución de la *query1* y la familia del siguiente acorde a solucionar.

También se asigna a *F_C_Root* la fundamental del segundo acorde de la solución de la *query1* y a *S_C_Root* la fundamental del siguiente acorde a solucionar.

Y se repite este mismo ciclo hasta terminar de procesar todos los acordes de la secuencia.

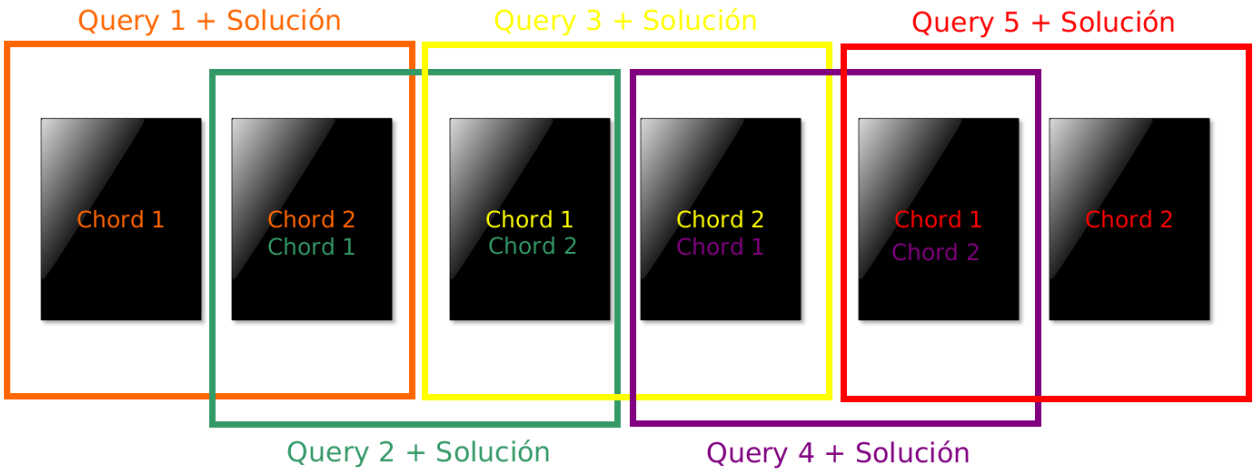


Figura 11: Secuencia de querys

3.3 Diseño del prototipo

Para la interfaz del prototipo, se ha tenido en cuenta un diseño simple y eficiente.

Se han utilizando colores y etiquetados apropiados para diferenciar las funcionalidades, los datos se han organizado en tablas para su cómoda visualización, imágenes de los acordes para visualizar las posiciones de manera sencilla, pulsar con el ratón sobre las imágenes para poder escucharlas, poder recuperar una secuencia de acordes anteriormente guardada y convertir el resultado de la secuencia de acordes en formato *PDF*.

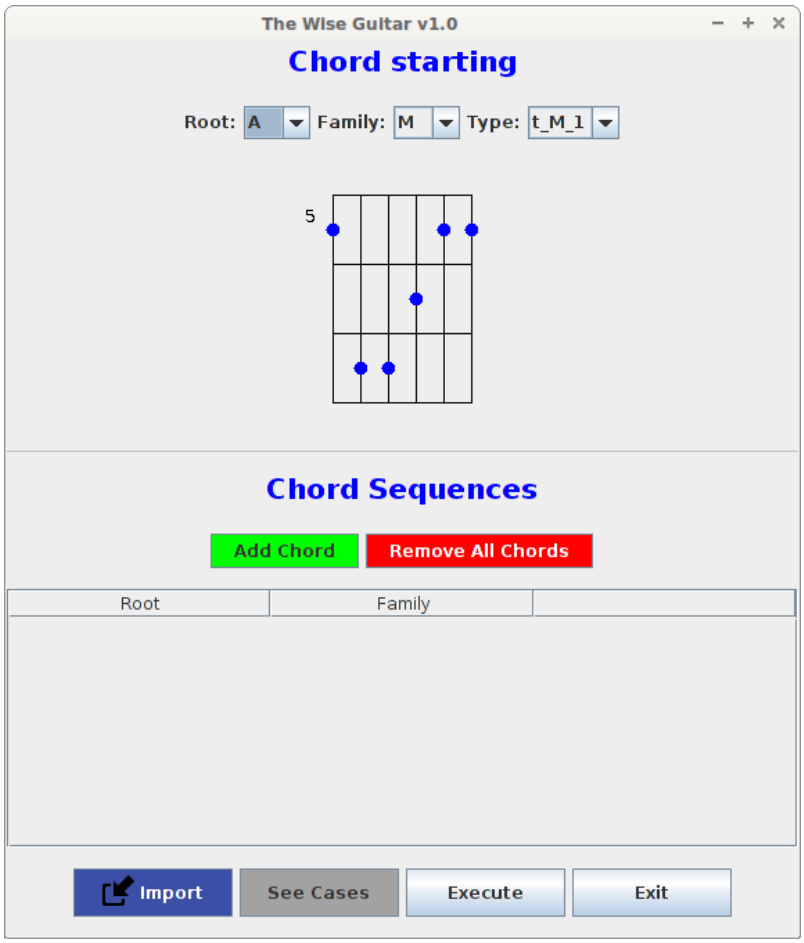


Figura 12: Interfaz: Pantalla principal

Cuando iniciamos *The Wise Guitar*, la primera pantalla que se muestra es la de la *Figura 12*, donde el usuario puede interactuar con el sistema.

Se puede pulsar sobre *Add Chord*, para ir añadiendo acordes en la secuencia, *Figura 13.*

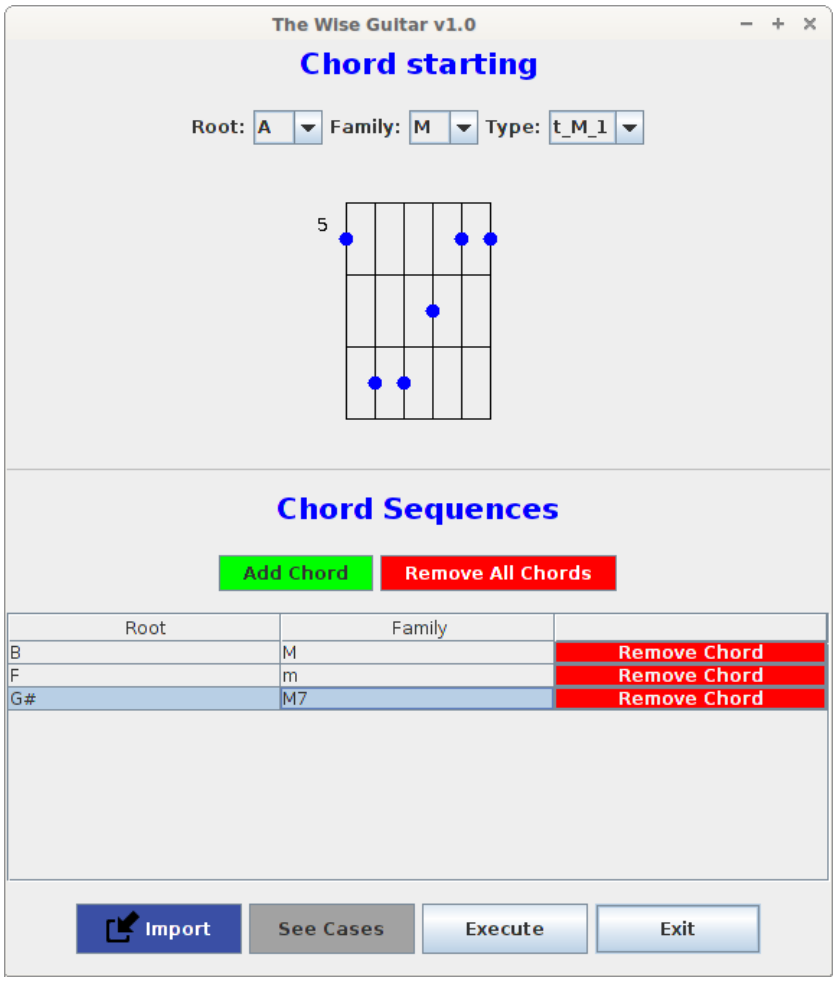


Figura 13: Interfaz: Pantalla principal con secuencias de acordes

Cuando pulsamos sobre *Execute* ejecuta, con los datos insertados, el sistema para que nos devuelva la secuencia de acordes más óptima, *Figura 14.*

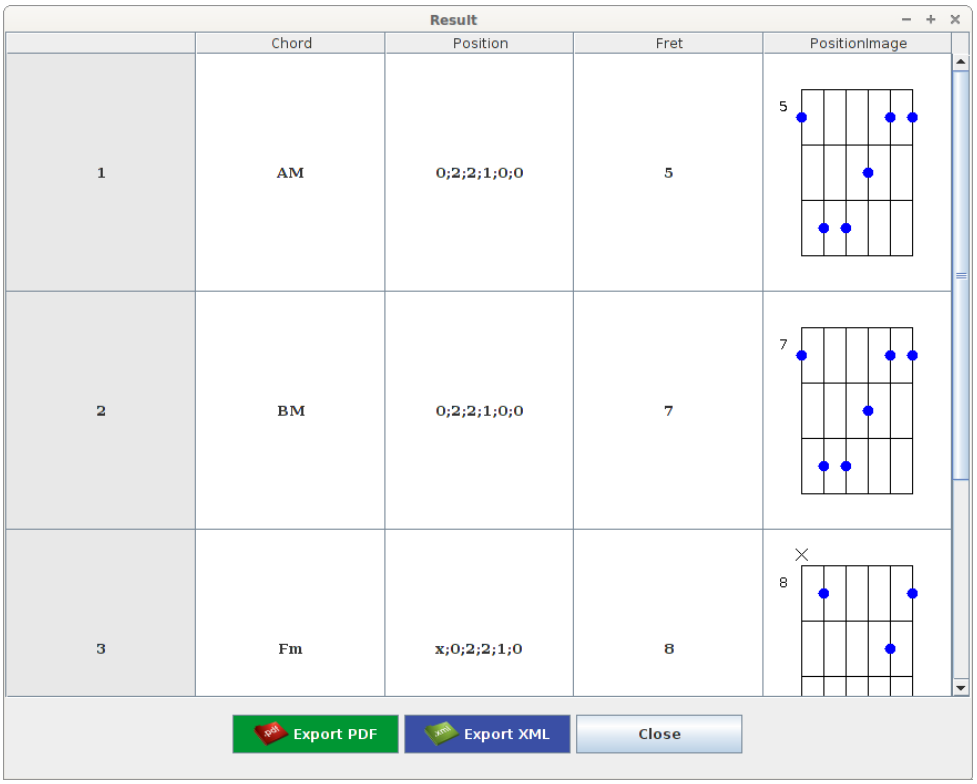


Figura 14: Interfaz: Resultado

Guardamos el resultado en formato *PDF*, véase la figura *Figura 15.*

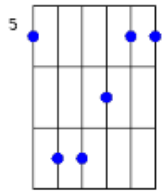
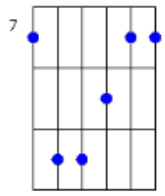
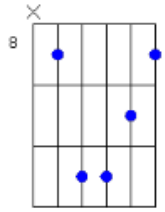
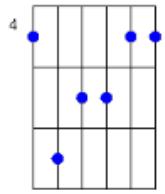
1	AM	0;2;2;1;0;0	5	
2	BM	0;2;2;1;0;0	7	
3	Fm	x;0;2;2;1;0	8	
4	AbM7/G#M7	0;2;1;1;0;0	4	

Figura 15: Interfaz: Resultado en PDF

Guardamos la secuencia de acordes en *XML*, véase la *Figura 16.*

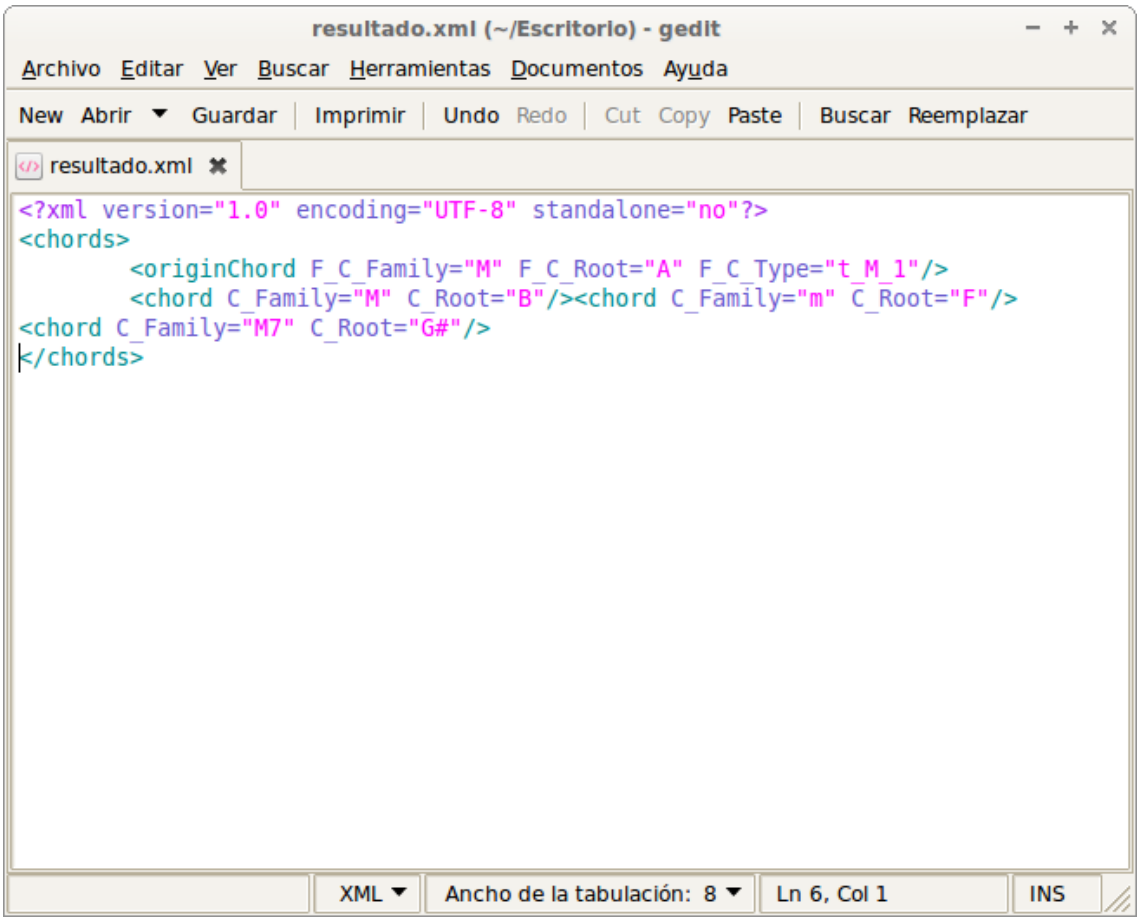


Figura 16: Interfaz: Resultado en XML

Seleccionamos la acción *Import*, para importar una secuencia de acordes anteriormente guardada, *Figura 17*.

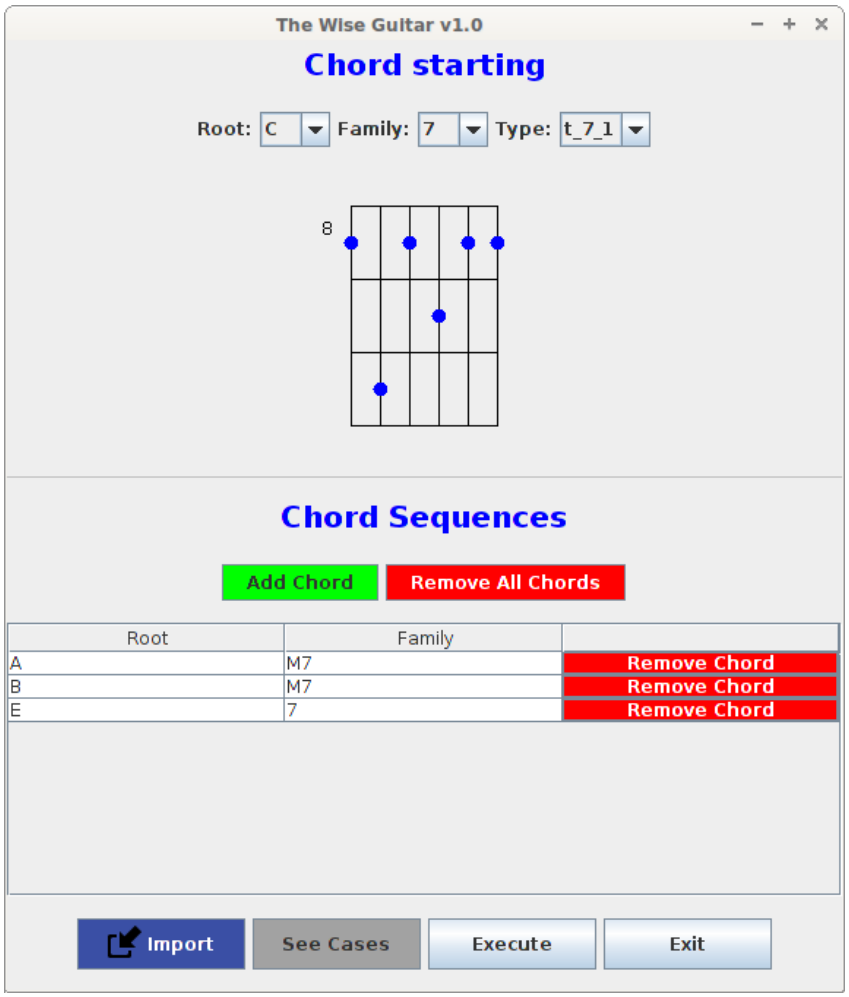


Figura 17: Interfaz: Secuencia de acordes importada

Pulsamos sobre *See Cases*, para ver de manera visual todos los casos que alberga la base de casos, *Figura 18.*

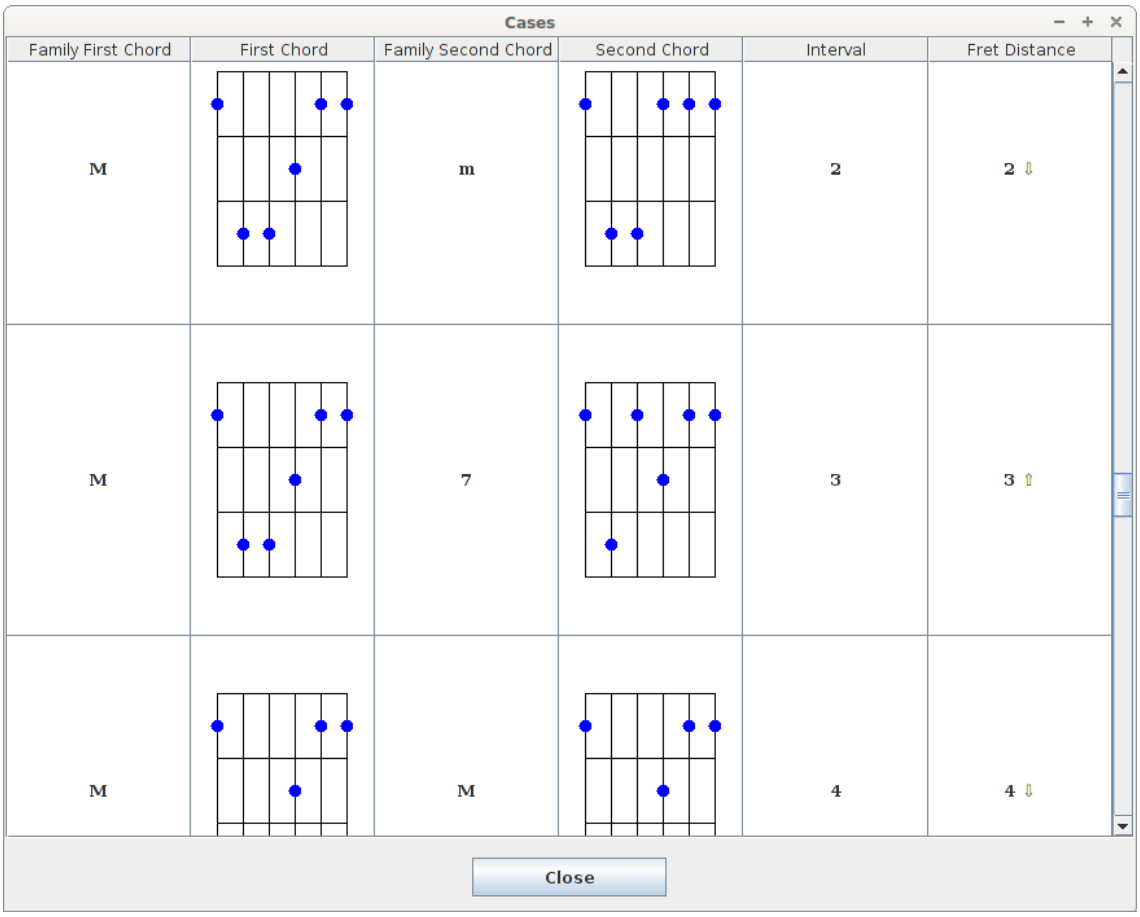


Figura 18: Interfaz: Ver todos los casos del caso base

Una vez que se ha expuesto las vistas de la interfaz del prototipo, hay que explicar cómo se interconecta con el sistema y cómo funciona el flujo del prototipo.

Para ello, véase la *Figura 19*, donde se puede ver las interacciones entre componentes.

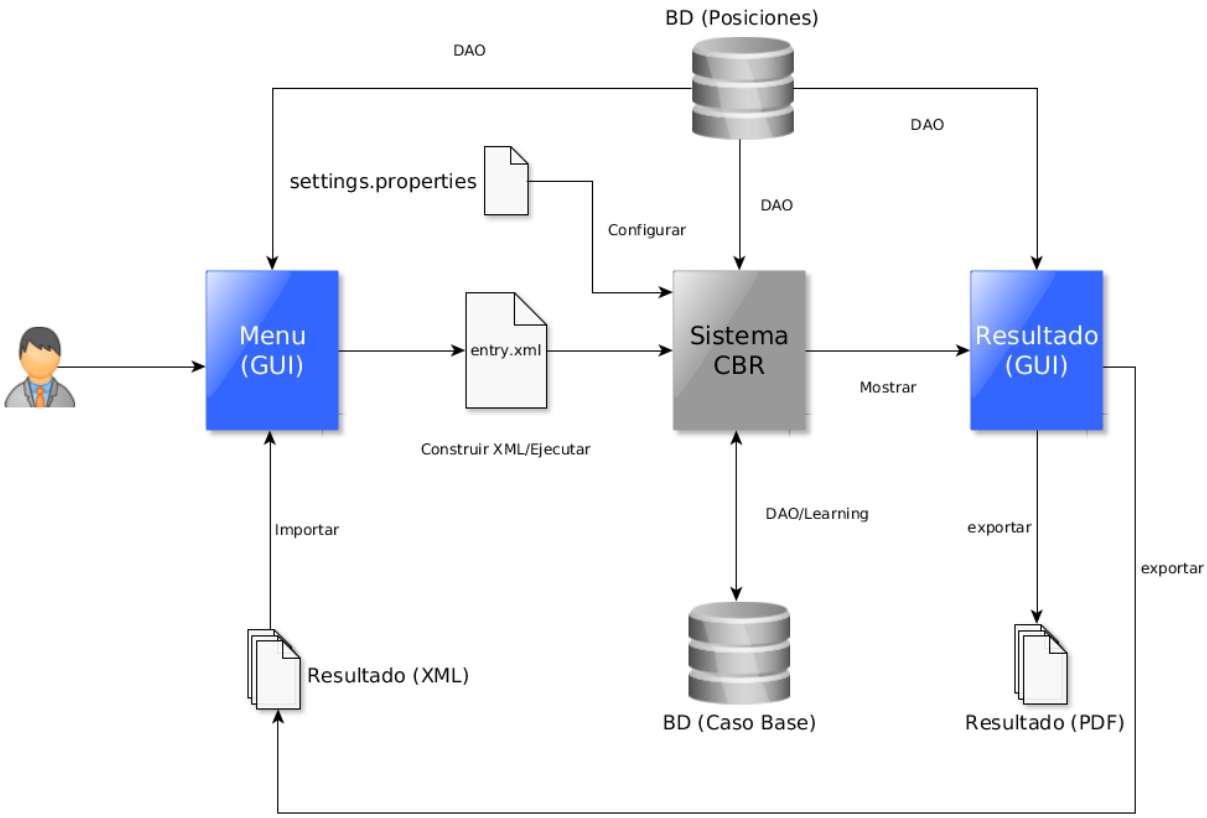


Figura 19: Diagrama del sistema

El usuario accede al menú, *Menu (GUI)*, donde puede insertar su propia secuencia o importar secuencias guardadas con anterioridad, *Resultado (XML)*. Los datos de las posiciones son cargados mediante clases *DAO*, para cargar los datos de la base de datos de posiciones utiliza las clases *DAOType*, *DAOFamilyChord* y *DAOType_FamilyChord*.

Al ejecutar la secuencia, el prototipo crea un *XML* de entrada llamado *entry.xml*. Este archivo es necesario, para que el *Sistema CBR* pueda recoger los datos de la secuencia de acordes para aplicar *CBR*. *El Sistema CBR*, utiliza la BD de las posiciones para reunir la información necesaria de las posiciones de los acordes y la BD de los casos, *BD (Caso Base)*, para reunir los datos de los casos. También, el sistema *CBR* hace uso de un archivo de configuración, *settings.properties*, que reúne parámetros necesarios para su aplicabilidad. Toda esta información es necesaria para poder realizar los cálculos para resolver el problema. Si es preciso aprender uno o más casos los guarda en la *BD (Caso Base)*.

Por último, se muestra la secuencia de acordes con las posiciones resultantes en la vista *Resultado (GUI)*. En ella se puede exportar el resultado en *PDF* y/o *XML*.

Capítulo 4: Evaluación del prototipo

En este último capítulo, se ha realizado la evaluación del sistema para comprobar su comportamiento y fiabilidad.

Y se ha concluido el proyecto con posibles trabajos futuros y una conclusión sobre este.

4.1 Evaluación

Para evaluar el prototipo, se ha elegido un total de 5 canciones, *Tabla 17*, con sus secuencias, y dos usuarios que han propuesto su propia solución basándose en las posiciones de acordes que hay almacenados en la BD de posiciones. Las pruebas comparan las secuencias aportadas por los usuarios y las del prototipo, y analizan la fiabilidad de *The Wise Guitar*.

Se han utilizado unas medidas de *esfuerzo* para estudiar los resultados aportados por los usuarios y los del prototipo y comprobar cuán útiles pueden llegar a ser los resultados que nos ofrece el sistema.

Tenemos que tener en cuenta que los resultados pueden variar, porque la base de conocimiento ha sido realizada por el creador del prototipo, guitarrista no profesional. Si hubiéramos obtenido una base de datos de casos realizados por auténticos profesionales, las pruebas habrían facilitado información más fiable.

Las medidas utilizadas son:

- **Medida 1**: Número medio de trastes de desplazamiento entre dos acordes consecutivos
- **Medida 2**: Número medio de dedos que cambian de cuerda entre dos acordes.
- **Medida 3**: Número medio del peso de cada posición del acorde. Para esta, se le da un peso de dificultad a una posición. Para hallar ese peso, se suman los dedos que se utilizan en el mástil y el número de trastes necesarios para representar la posición del acorde. Por ejemplo, tenemos Cm y el tipo de la *Figura 20*. Se puede observar que utiliza 3 trastes, del traste 8 al 10, y un total de 3 dedos, el primero se coloca en el traste 8 haciendo cejilla y en el traste 10 dos dedos. El peso de dificultad del acorde será la suma de los trastes y dedos, en este caso 3+3=6.

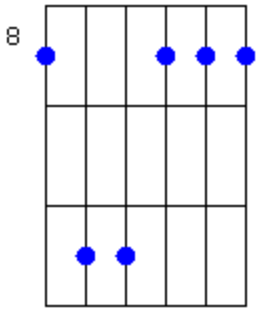


Figura 20: Posición del acorde Cm

Con esta evaluación se intenta averiguar en qué aspectos (desplazamiento por el mástil, cambios de dedos o dificultad de los acordes) de las soluciones del sistema difieren más de las soluciones propuestas por los dos usuarios. A mayor valor de las medidas, mayor esfuerzo, por lo que el sistema funcionará mejor cuanto menor sean estos valores.

Listado de canciones		
Nº de canción	**Título**	**Grupo/Artista**
1	Stairway to heaven	Led Zeppelin
2	Nothing else matter	Metallica
3	Ave Maria	Franz Schubert
4	Tune up	Miles Davis
5	On a little bamboo bridge	Louis Armstrong

Tabla 17: Listado de canciones utilizadas en las pruebas

El sistema aprende nuevos casos por cada prueba que se realiza, por lo que se ha decidido borrar los casos aprendidos por cada ejecución antes de realizar la siguiente prueba, para estar con igualdad de condiciones.

De una misma canción, cada usuario empieza por un tipo de acorde que le parezca más cómodo.

De las 10 canciones que en un principio se iban a evaluar, se han seleccionado 5 de ellas, porque eran las únicas donde los usuarios empezaban por la misma posición de acorde y, así, poder comparar las soluciones de ambos usuarios con la misma solución del sistema.

Tenemos configurado el sistema para que el límite de las posiciones no exceda el traste 10, no tiene sentido realizar acordes en zonas cercanas al puente de la guitarra porque son más incómodos de realizar.

Las tablas de las evaluaciones contienen la secuencia de la canción, las posiciones dadas por cada usuario y el resultado del prototipo. Además, se rellenan de color verde las posiciones coincidentes de los usuarios con las del prototipo. Se usan las siguientes abreviaturas:

- **Sec.**: Secuencia de acordes de la canción.
- **G1**: Secuencia de posiciones del usuario 1.
- **G2**: Secuencia de posiciones del usuario 2.
- **P**: Secuencia de posiciones del prototipo.

1. Stairway to heaven

Secuencias de acordes de la canción: C D F Am C G Am F Am C Am7

En la *Tabla 18* se observan los resultados, tanto de los dos usuarios como las del prototipo.

Sec.	**C**	**D**	**F**	**Am**	**C**	**G**	**Am**	**F**	**Am**	**C**	**Am7**
G1	4	5	1	2	4	3	2	1	2	4	2
G2	4	2	1	2	4	3	2	1	2	4	2
P	4	2	2	1	1	1	1	2	1	1	2

Tabla 18: Test1: Stairway to heaven. Resultados

Al realizar la evaluación, obtenemos los datos de la *Tabla 19*. La *Figura 21* muestran los datos de la evaluación en una gráfica de barras para su mejor análisis.

	G1		G2		P
Medida 1	0.8		1.3		3.4
Medida 2	2.6		2.6		1.4
Medida 3	5.5		5.6		6.8

Tabla 19: Test1: Evaluación

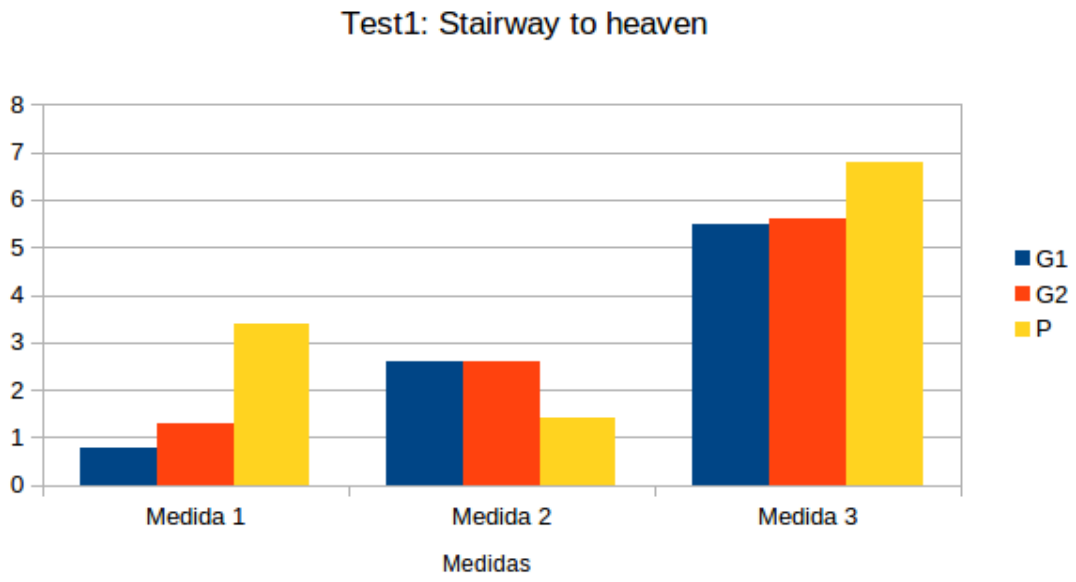


Figura 21: Test1: Gráfica

2. Nothing else matter

Secuencias de acordes de la canción: Em D C G B7 Em D C A D Em.

En esta secuencia ambos guitarristas aportan la misma solución, en la *Tabla 20* se observan esos datos.

Sec.	Em	D	C	G	B7	Em	D	C	A	D	Em
G1	1	5	4	3	2	1	5	4	2	5	1
G2	1	5	4	3	2	1	5	4	2	5	1
P	1	2	2	1	1	1	2	2	2	2	2

Tabla 20: Test2: Nothing else matter. Resultados

En la *Tabla 21* y la *Figura 22* se muestran los resultados.

	G1		G2		P
Medida 1	0.6		0.6		3.5
Medida 2	2.5		2.5		1.6
Medida 3	4.9		4.9		6.1

Tabla 21: Test2: Evaluación

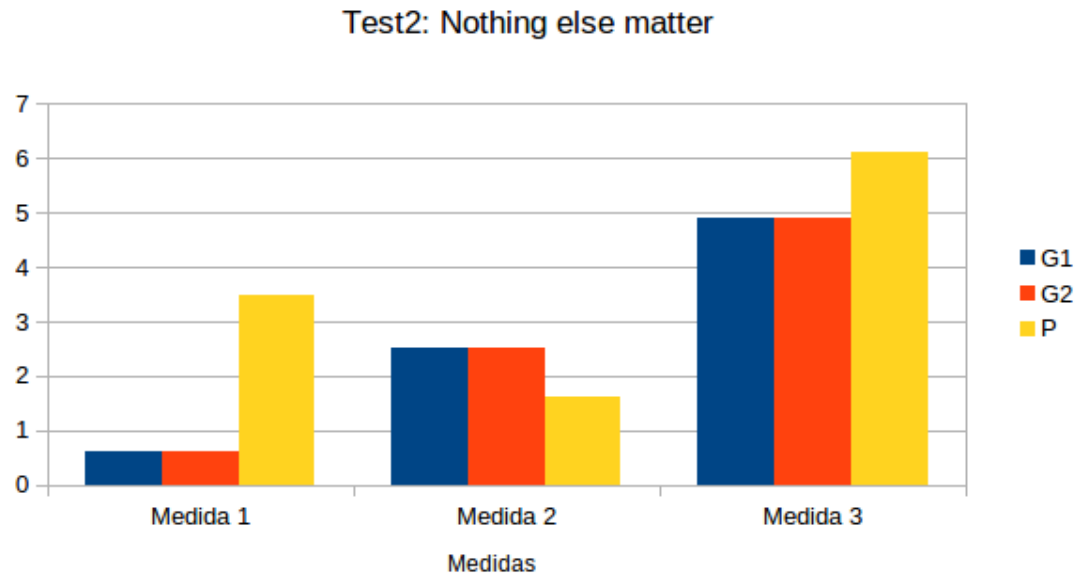


Figura 22: Test2: Gráfica

3. Ave Maria

Secuencias de acordes de la canción: Bb Gm F7 Gm Eb F7 Bb Bb A7 F# Gm F G7 D F C7 F F D7 Cm Eb7M G F F7 Bb Gm F F7

Ambos usuarios y el sistema coinciden en la mayor parte de posiciones propuestas, lo cual se refleja en la similitud entre los valores de las tres medidas de esfuerzo. En la *Tabla 22* se observa esas coincidencias.

Sec.	Bb	Gm	F7	Gm	Eb	F7	Bb	Bb	A7	F#	Gm	F	G7	D	F	C7	F	F	D7	Cm	Eb7M	G	F	F7	Bb	Gm	F	F7
G1	2	1	1	1	4	1	2	2	2	1	1	1	1	5	1	3	1	1	4	2	5	1	1	1	2	1	1	1
G2	2	1	1	1	2	1	2	2	2	1	1	1	1	5	6	2	6	6	4	2	2	3	1	1	2	1	1	1
P	2	1	2	1	2	2	2	2	2	1	1	1	1	2	2	1	5	1	2	1	6	3	1	1	2	1	1	1

Tabla 22: Test3: Ave Maria. Resultados

En la *Tabla 23* y la *Figura 23 se* muestran los resultados de la tercera evaluación.

	G1		G2		P
Medida 1	1.3		1.7		2.6
Medida 2	1.7		1.9		1.7
Medida 3	6.4		6.2		6.4

Tabla 23: Test3: Evaluación

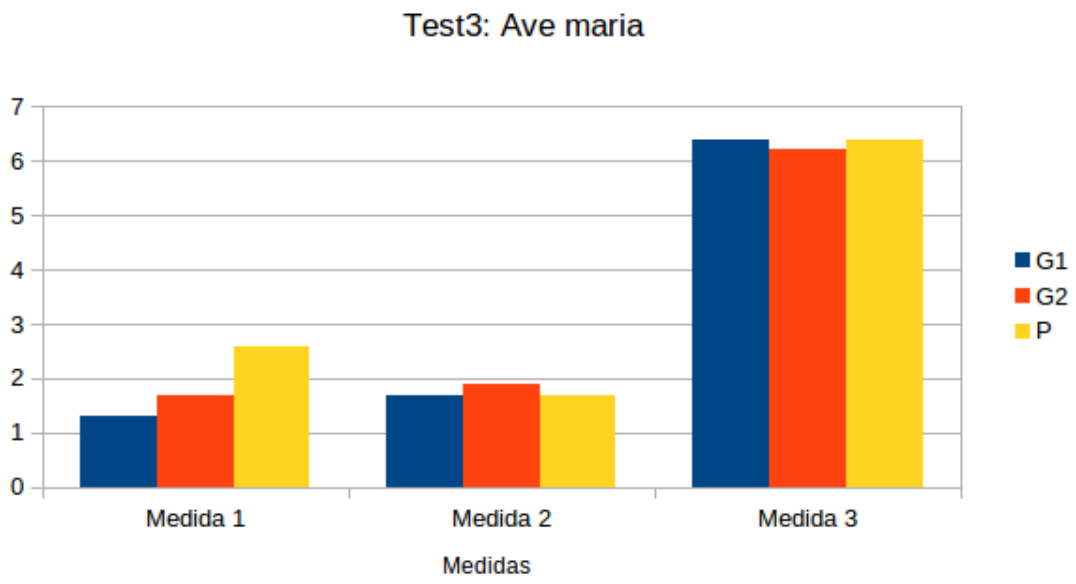


Figura 23: Test3: Gráfica

4. Tune up

Secuencias de acordes de la canción: Em7 A7 D7M G7 C7M Em7 A7 Bb7M Cm7 F7 Bb7M Eb7M

Ambos guitarristas coinciden con el sistema en aproximadamente un 50%. Se puede ver los resultados de las secuencias en la *Tabla 24*.

Sec.	Em7	A7	D7M	G7	C7M	Em7	A7	Bb7M	Cm7	F7	Bb7M	Eb7M
G1	4	2	6	1	2	4	2	2	2	1	2	5
G2	4	2	6	1	5	4	2	2	2	1	2	6
P	4	2	2	1	2	2	2	2	4	2	2	5

Tabla 24: Test4: Tune up. Resultados

Véase la *Tabla 25* y la *Figura 24* para ver los resultados.

	G1		G2		P
Medida 1	1.1		0.8		3.1
Medida 2	1.8		1.8		1.8
Medida 3	5.2		4.5		5.7

Tabla 25: Test4: Evaluación

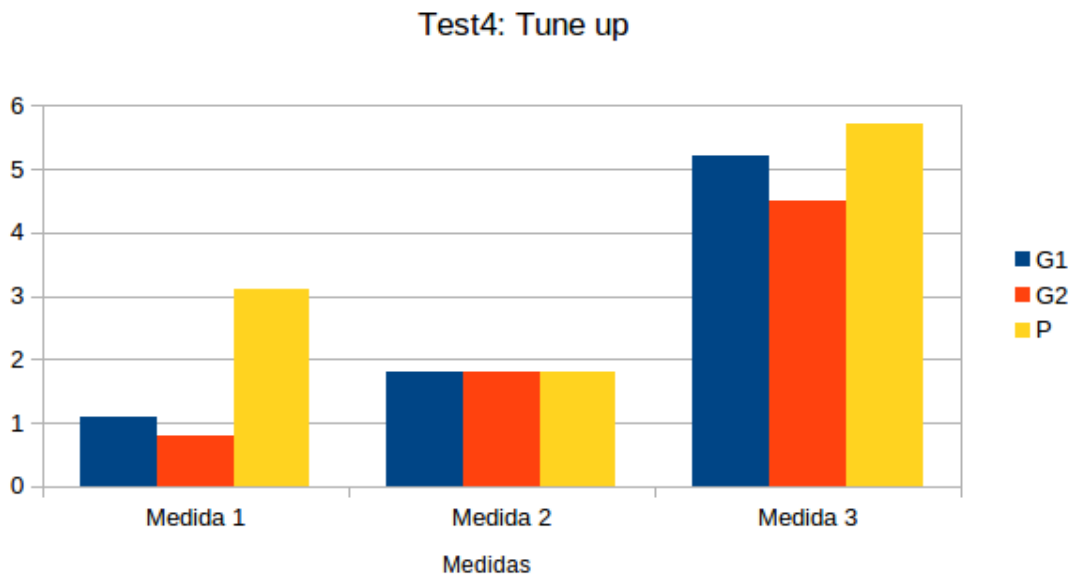


Figura 24: Test4: Gráfica

5. On a Little bamboo bridge

Secuencias de acordes de la canción: F# Bb7 Eb7 Ab7 C#7 F# B Bm F#

De nuevo, la reelección de posiciones de los guitarristas coincide en aproximadamente un 50% con la del sistema. Véase la *Tabla 26*.

Sec.	F#	Bb7	Eb7	Ab7	C#7	F#	B	Bm	F#
G1	1	2	2	1	2	1	2	2	1
G2	1	1	2	1	2	1	2	2	1
P	1	1	4	1	2	5	1	1	1

Tabla 26: Test5: On a little bamboo bridge. Resultados

Véase la *Tabla 27 y* la *Figura 25* donde se encuentran los resultados de la evaluación.

	G1		G2		P
Medida 1	1.2		1		2.5
Medida 2	2		2		2.2
Medida 3	6.5		6.5		6.6

Tabla 27: Test5: Evaluación

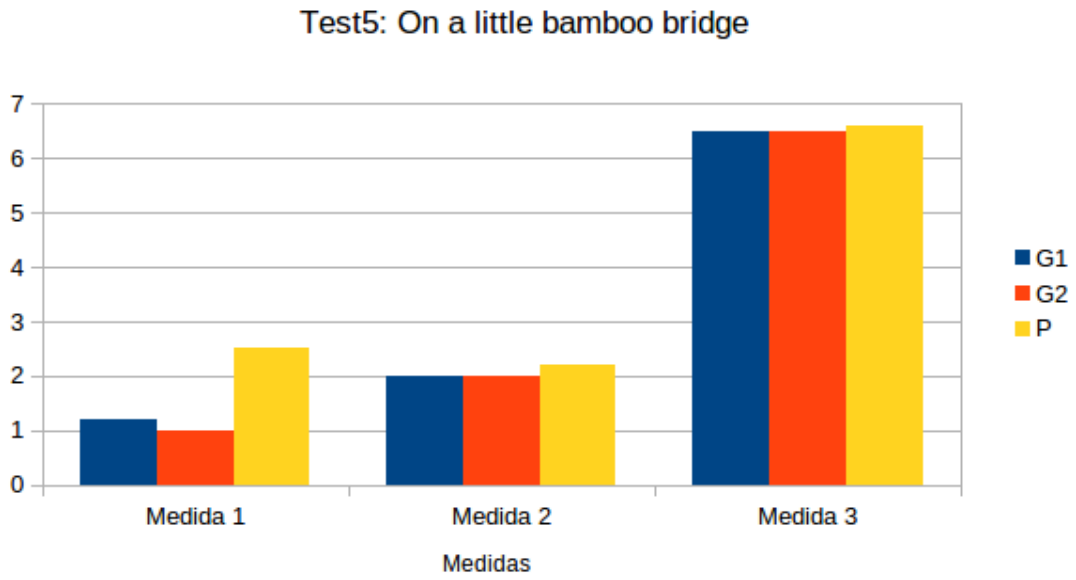


Figura 25: Test5: Gráfica

Evaluación final

Para finalizar la evaluación, se ha realizado un análisis con los datos obtenidos de las evaluaciones de las 5 canciones.

Se ha hallado las medias de cada medida para cada usuario y para el sistema, para analizar la distancia obtenida por el prototipo respecto a los usuarios. (*Tabla 28)*

$$\mu_{gk}^{j} = \sum_{i=1}^{N} m_j(g_k)/N$$

- **m_j:** Número de medida.
- **N:** Número total de temas.
- **g_k:** Usuario k (k=1 ó k=2) ó sistema (**p**).

	Medida 1	**Medida 2**	**Medida 3**
G1	$\mu_{g1}^{1} = 1$	$\mu_{g1}^{2} = 2.12$	$\mu_{g1}^{3} = 5.7$
G2	$\mu_{g2}^{1} = 1.08$	$\mu_{g2}^{2} = 2.16$	$\mu_{g2}^{3} = 5.54$
P	$\mu_{p}^{1} = 3.02$	$\mu_{p}^{2} = 1.74$	$\mu_{p}^{3} = 6.32$

Tabla 28: Evaluación final. Resultados

Como primera impresión, destaca que los valores medios son muy similares entre ambos guitarristas.

Se observan dos aspectos importantes:

Uno de ellos, está relacionado con la segunda medida (cambio de dedos), en el que se observa en los resultados, en particulares en el test1 y test2, una reducción de la dificultad a la hora de cambiar un acorde por otro por parte del sistema. En el contexto de los guitarristas, es importante utilizar lo menos posible el movimiento de dedos, para realizar el cambio entre acordes, eso significa que a la hora de ejecutar los acordes es más cómodo y rápido, lo cual es valorado a la hora de interpretar las secuencias de las canciones.

El segundo de ellos, no menos importante, es acerca de las medidas restantes, primera medida y tercera medida, donde se observa en todos los test realizados que han obtenido

unos resultados no tan positivos respecto a los de los guitarristas. La excepción se encuentra en el test5, que el resultado de la tercera medida comparado entre la secuencia aportada por los usuarios y el prototipo, se asemejan de manera indudable.

En la evaluación final se observa que aún falta por perfilar el sistema, para mejorar sus resultados. Con los datos obtenidos, véase la *Tabla 28* y la *Figura 26*, se analiza la distancia entre las secuencias de los usuarios y del prototipo, al realizarlo se observa que la secuencia de los usuarios se solapan, mostrando su indudable parecido, y del prototipo, difiere claramente en la primera medida. La idea es que el resultado del prototipo, en sus medidas, se encuentre lo más próximo a 0.

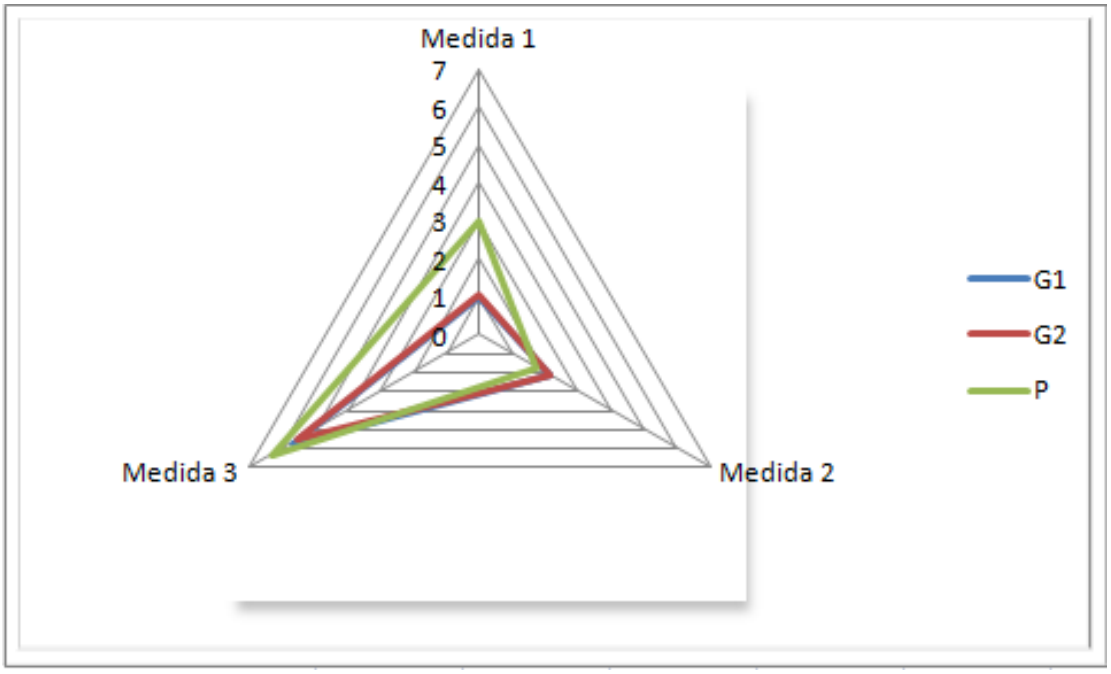


Figura 26: Evaluación final. Gráfica

Por último, debemos tener en cuenta, que nuestra base de datos de casos es de 66 casos y no expertos, por lo que si tuviera un millar de casos realizados por personas con altos conocimientos en el dominio, es idóneo en aplicaciones *CBR*, la evaluación podría llegar a ser más positiva, por lo que el prototipo podría mejorar los resultados aportados por los dos usuarios que se ofrecieron al experimento, o al menos aproximarse al de ellos, convirtiendo a *The Wise Guitar* en una herramienta útil.

4.2 Trabajos futuros

Uno de los posibles trabajos que se podrían realizar con el prototipo, sería centralizar la base de datos de los casos en un servidor externo. Así, al estar centralizada, accedería cualquier usuario desde sus dispositivos personales y el sistema aprendería de manera más rápida.

Los usuarios tendrían la misma información de casos, por esta razón el sistema albergaría una base de datos más completa que si la tuviera en local. Se permitiría al usuario que pudiera elegir sobre qué base de datos trabajar: la centralizada o la local.

Para mejorar la base de conocimiento, se podría dar la posibilidad al usuario de modificar el resultado del prototipo. Este propondría un cambio, modificando una posición de la secuencia resultante, el sistema calcularía de nuevo el resultado, aplicándolo a todas las subsecuencias que comparten el tipo y la familia del acorde cambiado y, por último, el sistema aprendería de esas modificaciones. Por ejemplo, tenemos el siguiente resultado: Cm (Tipo 1) - A7 (Tipo 2) - Dm (Tipo 1) - B7 (Tipo 2) - Em (Tipo 3). El usuario decide cambiar el tipo de A7 al Tipo 3, porque así le resulta más cómodo cuando cambia de Cm a A7. Entonces, el sistema debe de aplicar esa modificación para el acorde B7, ya que este se encuentra en el mismo caso, Dm (Tipo 1) B7 (Tipo 2), que Cm a A7 (mismas familias, posiciones y tipos). El sistema deberá cambiar el tipo de B7 a Tipo 3.

Otra posible aplicación, habría sido desarrollar el prototipo para dispositivos móviles en distintos sistemas operativos (*Android*, *iOS*, *Windows Phone*).

4.3 Conclusiones

Como se ha visto a lo largo de este trabajo, se ha conseguido realizar satisfactoriamente todos y cada uno de los objetivos que se marcaron en un principio e incluso algunos que iban emergiendo durante el desarrollo de este trabajo.

La importancia de obtener una secuencia de acordes que pueda ser ejecutada de manera cómoda para un usuario, tiene tanto valor a día de hoy como hace 50 años. Al estar en la era de la información, es altamente probable, que el usuario se desborde de la inmensa cantidad de posiciones de acordes que existen en cualquier medio (revistas, Internet,

libros). Este no podrá asimilar todas ellas y mucho menos conocer si está ejecutando una secuencia de acordes de manera cómoda y eficiente.

La evaluación del prototipo aquí implementado aporta indicios de que, con ciertas mejoras, puede convertirse en una herramienta útil como herramienta didáctica para guitarristas nóveles.

Es cierto, de ahí el nombre de *prototipo*, que es un mero experimento para comprobar su comportamiento y fiabilidad para una posible utilización en el futuro, conociendo sus actuales límites y que debe ser refinado para mejorar sus funcionalidades y resultados.

Académicamente hablando, ha sido un proyecto muy nutritivo a la hora de desarrollar y poder aplicar todos los conocimientos aprendidos en la Universidad. Lo que más he valorado, ha sido enfrentarme a nuevos problemas, donde he tenido que aplicar una actitud investigadora, tanto para analizar el problema principal, como para aprender por mi cuenta nuevas herramientas completamente nuevas. La Universidad me ha proporcionado una mentalidad de análisis y de aprendizaje.

Me gustaría concluir este trabajo, comentando que, con el hecho de haber realizado un proyecto de estas características uno siente que ha aportado una idea innovadora, que pueda ser aplicada a futuras ideas relacionadas con el mundo de la guitarra y la música. Cada vez es más complicado innovar, pero el ser humano no tiene límites en buscar y realizar nuevos retos o mejorar los ya existentes.

Por consiguiente animo al lector, que desarrolle, sin dudar, sus propios proyectos. Al fin y al cabo, es lo que nos hace avanzar como seres humanos.

Bibliografía y referencias

1. Lixing Wang & Wai-Hung Ip, *A Soft Case-based Reasoning System for Travelling Time Estimation*, http://tinyurl.com/ll6tmv8
2. Jaroslav Hodál & Jirí Dvorák, Using Case-Based Reasoning For Mobile Robot Path Planning, http://www.engineeringmechanics.cz/pdf/15_3_181.pdf
3. ISMIR, *The International Society For Music Information Retrieval*, http://www.ismir.net/
4. Takuya Yoshioka, Tetsuro Kitahara, Kazunori Komatani, Tetsuya Ogata & Hiroshi G. Okuno, *Automatic chord transcription with concurrent recognition of chord symbols and boundaries*, http://ismir2004.ismir.net/proceedings/p020-page-100-paper149.pdf
5. Yuichiro Yonebayashi, Hirokazu Kameoka & Shigeki Sagayama, *Automatic Decision of Piano Fingering Based on Hidden Markov Models*, www.aaai.org/Papers/IJCAI/2007/IJCAI07-469.pdf
6. Thomas Grapperon. *Chord!*. http://getchord.com/
7. Jim Cranwell, *Guitar Chords Generator*, http://www.danmansmusic.com/chords3/guitar.html
8. Ovelin Team, *Guitar Tuna*, http://ovelin.com/guitartuna/
9. Christopher Ladd, *ChordBank*, http://www.chordbank.com/
10. SongSterr, *SongSterr*, http://www.songsterr.com/
11. Garrobé i Marqui, Joan (1993). *El libro de los acordes para guitarra.* Barcelona: Hospitalet.
12. Oracle, *Java*, http://www.java.com
13. Oracle, *AWT*, http://docs.oracle.com/javase/1.5.0/docs/guide/awt/
14. David Koelle, *jFugue*, http://www.jfugue.org/
15. David Koelle, *Using the JFugue MusicString*, http://www.jfugue.org/jfugue-chapter2.pdf
16. Bruno Lowagie, *iText*, http://itextpdf.com/
17. W3C, *Document Object Model*, http://www.w3.org/DOM/
18. D. Richard Hipp, *Sqlite*, http://www.sqlite.org/
19. GAIA (Universidad Complutense de Madrid), *jColibri*, http://gaia.fdi.ucm.es/research/colibri/jcolibri

20. GAIA (Universidad Complutense de Madrid), *jColibri Tutorial*, http://gaia.fdi.ucm.es/files/people/juanan/jcolibri/downloads/tutorial.pdf
21. Linus Torvalds, *Git*, http://git-scm.com/
22. Atlassian, *Bitbucket*, https://bitbucket.org/
23. Ramon López de Mántaras Badia, *Case Based Reasoning*, http://grfia.dlsi.ua.es/repositori/grfia/otros/ramon_lopez_IntroCBR20062505.ppt
24. Pablo Baudí Canales, *The Wise Guitar*, https://bitbucket.org/pbc26/tfg
25. Evelyn Fix and J.L. Hodges, *Jr.* (1951), *An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges*, Gran Bretaña.
26. Michael M. Richter & Rosina O. Weber (2013), *Case Based Reasoning: A Textbook,* Alemania y Estados Unidos.

Anexos

Anexo1 (Base de datos)

- ### Base de datos de posiciones

A la hora de diseñar la base de datos de las posiciones y familias, sólo nos hizo falta representar tres entidades, véase la *Figura 27*.

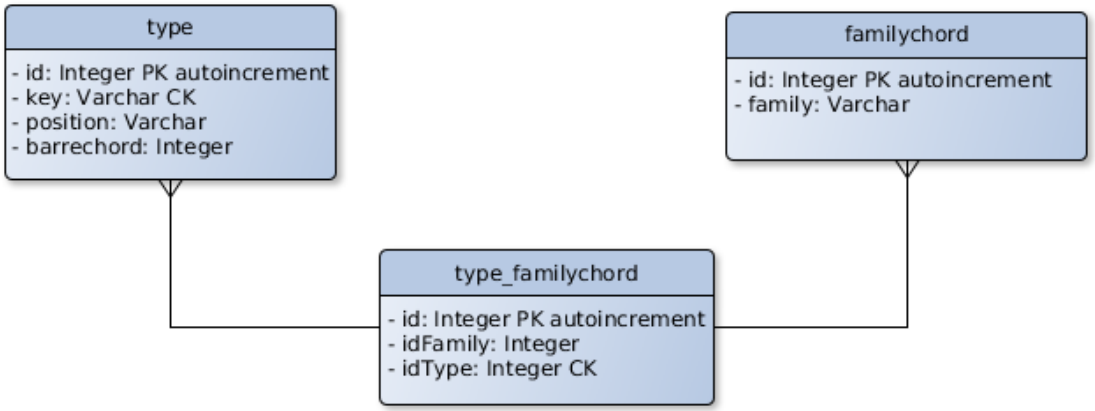


Figura 27: Diagrama BD de posiciones

Las entidades son, *type*, *familychord y type_familychord*, esta última se comporta como la relación muchos a muchos.

Al principio, se pensó en incluir entidades de utilidad para que albergase las notas musicales (*A, C, D#, B*), pero al final se decidió realizarlo a nivel de programación.

Al tener pocas entidades, su gestión se hace simple, con lo que conseguimos más rapidez a la hora de realizar alguna acción *CRUD (Create, Read, Update, Delete).*

Todas las entidades, hacen uso de un entero identificativo autoincremental para optimizar las consultas y permitir reconocer un registro como único.

A continuación vamos a explicar, para qué utilizamos cada atributo de cada entidad:

type

- **key:** Se puede definir como el identificativo de la posición, se representa como *t_Familia_X*, *t*: indica que es un tipo, *Familia*: la familia a la que pertenece ese

tipo y *X*: el número de posición. Como sabemos, hay más de un tipo por familia. Por ejemplo: *t_M_1*, *t_M_2*, *t_7_1*, *t_M7_1*.

- **position:** Representa textualmente la posición, de la siguiente manera *6º;5º;4º;3º;2º;1º*. Cada número, referencia la cuerda de la guitarra, siendo la *6º* la más gruesa hasta la *1º* la más fina. Aparte de contener números, también puede incluir una *x*, que indica que esa cuerda no se pulsa. Por ejemplo: *0;2;2;1;0;0*.
- **barrechord:** Indica si la posición utiliza cejilla. Cuando una posición usa cejilla, se puede ejecutar en cualquier parte del mástil. Aunque, en este proyecto, al indicar que una posición hace uso de esta, solamente se puede utilizar en una parte concreta del mástil y cerca del clavijero, dejando algunas notas al aire (Cuerdas que no se pulsan en el mástil, pero si deben de sonar)

type_familychord

- **idFamily:** Id del registro de *familychord*.
- **idType:** Id del registro de *type*.

familychord

- **family:** Una cadena que representa a la familia (*M, m, M7, 7, m7*).

- ### Base de datos de casos

La BD de los casos, está representado por un simple fichero de texto, configurado por el framework *jColibri* llamado *plainTextConnector.txt*. Cada caso se representa en una fila y para separar cada atributo del caso se utiliza el carácter *punto y coma (;)* como delimitador.

El orden de los atributos de un caso, en la base de datos de casos, es el siguiente:

idCaseDescription;F_C_Type;F_C_Family;S_C_Interval;S_C_Family;idCaseSolution; FS_Delta_Fret;S_C_Type

Descripción

- **idCaseDescription:** Identificativo de la descripción.
- **F_C_Type (First chord type):** La posición del primer acorde mediante la *key* del tipo.
- **F_C_Family (First chord family):** La familia del primer acorde.
- **S_C_Interval (Second Chord Interval):** El intervalo del primer al segundo acorde en semitonos.
- **S_C_Family (Second chord family):** Familia del segundo acorde.

Solución

- **idCaseSolution:** Identificativo de la solución.
- **FS_Delta_Fret (First&Second Delta Fret):** Distancia en trastes entre el primer y el segundo acorde. Si la distancia es negativa, indica que hay que desplazarse hacia arriba del mástil, si es 0 no hay que desplazarse y si es positivo hay que desplazarse hacia abajo del mástil.
- **S_C_Type (Second chord type):** La posición del segundo acorde mediante la *key* del tipo.

En la *Tabla 29*, se puede visualizar en formato texto todos los casos añadidos a mano en la BD. El sistema se basa en ellos para aprender cuando recibe nuevos problemas (*querys*). La BD de casos alberga un total de 66 casos.

idCase Description	F_C_Type	F_C_Family	S_C_Interval	S_C_Family	idCase Solution	FS_Delta_Fret	S_C_Type
1	t_M_4	M	1	M	1	-3	t_M_1
2	t_M_5	M	2	M	2	-3	t_M_3
3	t_M_1	M	4	M	3	4	t_M_1
4	t_M_2	M	3	M	4	3	t_M_2
5	t_m7_3	m7	3	m7	5	1	t_m7_4
6	t_m7_4	m7	5	m7	6	0	t_m7_5
7	t_m7_2	m7	2	m7	7	2	t_m7_2
8	t_m7_4	m7	4	m7	8	3	t_m7_2
9	t_7_3	7	4	7	9	4	t_7_3
10	t_7_4	7	3	7	10	1	t_7_1
11	t_7_1	7	2	7	11	0	t_7_4
12	t_7_5	7	3	7	12	3	t_7_2
13	t_m_2	m	1	m	13	1	t_m_2
14	t_m_2	m	2	m	14	3	t_m_1
15	t_m_1	m	3	m	15	-1	t_m_3
16	t_m_1	m	5	m	16	2	t_m_4
17	t_M_3	M	5	m	17	0	t_m_4
18	t_M7_2	M7	5	M7	18	0	t_M7_1
19	t_M7_5	M7	5	M7	19	0	t_M7_4
20	t_M7_3	M7	4	M7	20	-2	t_M7_5
21	t_7_1	7	2	m7	21	2	t_m7_4
22	t_7_2	7	0	m7	22	0	t_m7_2
23	t_7_5	7	0	m7	23	0	t_m7_2
24	t_m7_3	m7	5	7	24	0	t_7_2
25	t_7_1	7	5	M7	25	0	t_M7_2
26	t_M_1	M	0	7	26	0	t_7_1
27	t_M_1	M	1	7	27	-1	t_7_1
28	t_M_2	M	5	7	28	0	t_7_1
29	t_M_1	M	3	7	29	-3	t_7_1
30	t_M_2	M	5	7	30	-2	t_7_1
31	t_M_2	M	2	7	31	2	t_7_2
32	t_M7_5	M7	3	m7	32	0	t_m7_2
33	t_M7_5	M7	1	m7	33	2	t_m7_2
34	t_m_1	m	0	m7	34	0	t_m7_4
35	t_m_2	m	0	m7	35	0	t_m7_2
36	t_m7_1	m7	2	M	36	0	t_M_5
37	t_M_1	M	5	m7	37	0	t_m7_2
38	t_M_4	M	2	m7	38	-2	t_m7_4
39	t_M_2	M	3	m7	39	2	t_m7_4
40	t_m7_2	m7	5	M	40	0	t_M_5
41	t_M7_6	M7	2	m	41	0	t_m_1
42	t_M7_6	M7	5	M	42	0	t_M_2
43	t_7_4	7	5	m	43	0	t_m_2
44	t_m_2	m	0	7	44	0	t_7_2
45	t_7_4	7	2	m	45	0	t_m_1
46	t_m_1	m	4	7	46	-1	t_7_2
47	t_m_5	m	3	7	47	-1	t_7_4
48	t_m_1	m	2	M	48	-2	t_M_1
49	t_m_2	m	6	M	49	-1	t_M_1
50	t_M_2	M	5	m	50	0	t_m_1
51	t_m_1	m	5	M	51	0	t_M_2

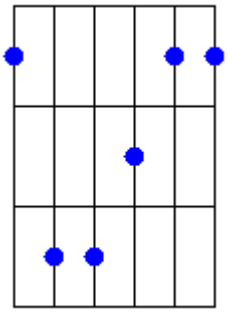
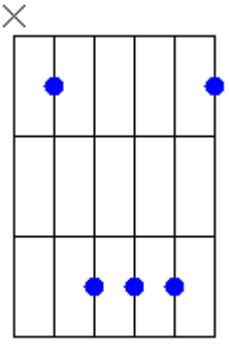
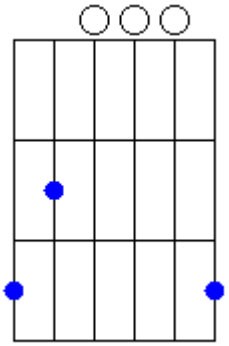
52	t_m_2	m	3	M	52	3	t_M_2
53	t_m_1	m	1	M	53	1	t_M_1
54	t_M_3	M	1	m	54	2	t_m_1
55	t_m_6	m	0	M	55	-2	t_M_5
56	t_m_5	m	2	M	56	2	t_M_6
57	t_m_1	m	5	M	57	2	t_M_7
58	t_M_1	M	0	m	58	0	t_m_1
59	t_M_4	M	6	m	59	-3	t_m_2
60	t_M_1	M	2	m	60	2	t_m_1
61	t_M_1	M	4	m	61	-1	t_m_2
62	t_M_2	M	2	m	62	-2	t_m_2
63	t_M_5	M	5	m	63	0	t_m_2
64	t_m_2	m	1	M	64	1	t_M_2
65	t_m_3	m	1	M	65	3	t_M_4
66	t_M_1	M	2	M7	66	0	t_M7_6

Tabla 29: Caso Base (Base de datos de casos)

Anexo2 (Diccionario de posiciones)

En las *Tabla 30, Tabla 31, Tabla 32, Tabla 33 y Tabla 34* se muestran todas las posiciones que se han utilizado en el prototipo.

- ### Acordes Mayores

Key	Position	Barrechord	Imagen
t_M_1	0;2;2;1;00	1	
t_M_2	x;0;2;2;2;0	1	
t_M_3	0;-1;-3;-3;-3;0	0	

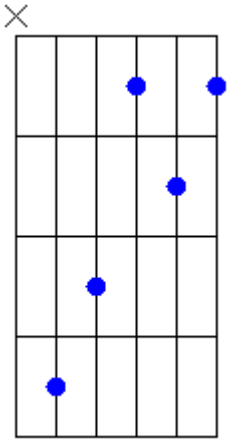
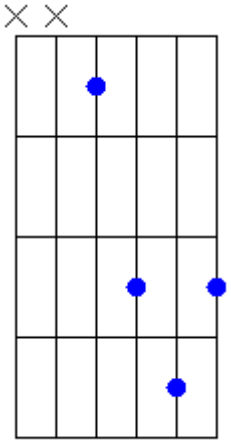
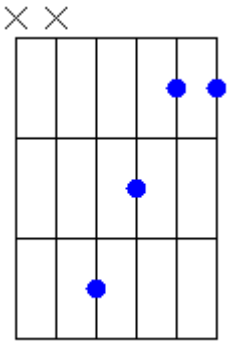
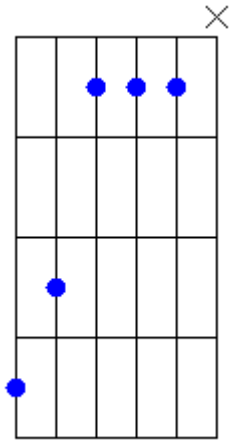
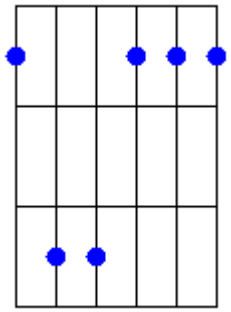
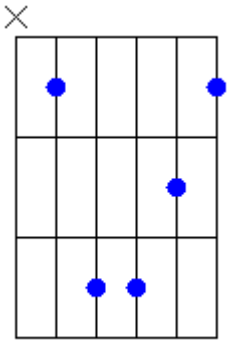
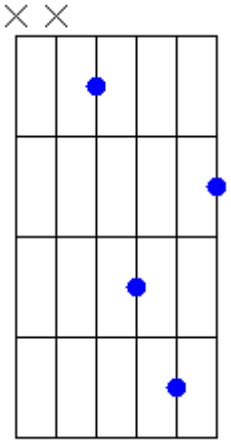
t_M_4	x;0;-1;-3;-2;-3	1	
t_M_5	x;x;0;2;3;2	1	
t_M_6	x;x;0;-1;-2;-2	1	
t_M_7	0;-1;-3;-3;-3;x	1	

Tabla 30: Acordes Mayores

- **Acordes menores**

Key	Position	Barrechord	Imagen
t_m_1	0;2;2;0;0;0	1	
t_m_2	x;0;2;2;1;0	1	
t_m_3	x;x;0;2;3;1	1	

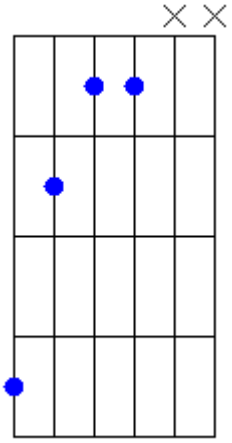
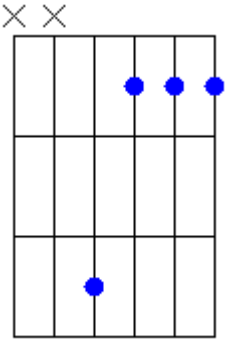
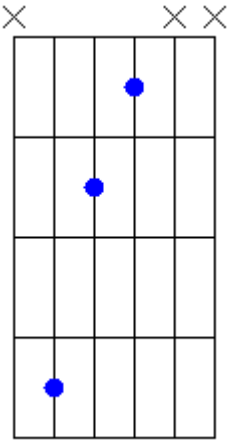
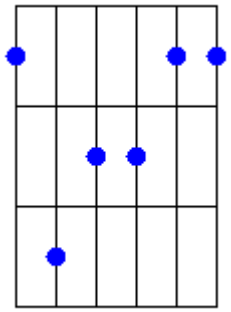
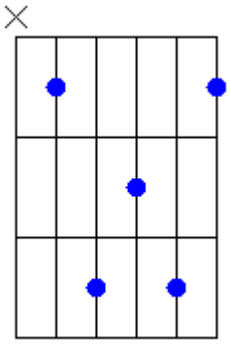
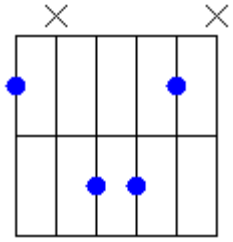
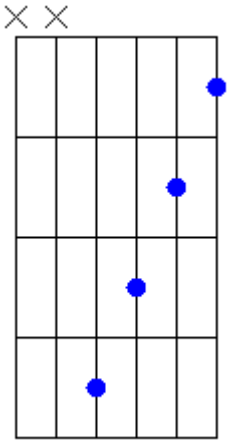
t_m_4	0;-2;-3;-3;x;x	1	
t_m_5	x;x;0;-2;-2;-2	1	
t_m_6	x;0;-2;-3;x;x	1	

Tabla 31: Acordes menores

- **Acordes de Mayor séptima**

Key	Position	Barrechord	Imagen
t_M7_1	0;2;1;1;0;0	1	
t_M7_2	x;0;2;1;2;0	1	
t_M7_3	0;x;1;1;0;x	1	
t_M7_4	x;x;0;-1;-2;-3	1	

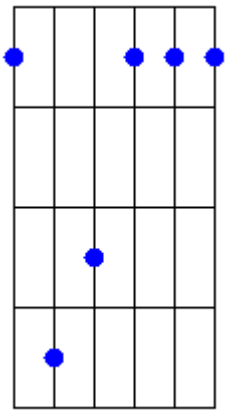
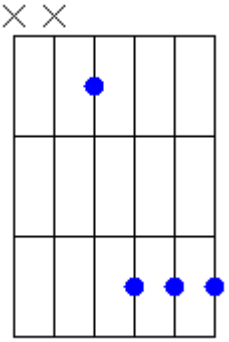
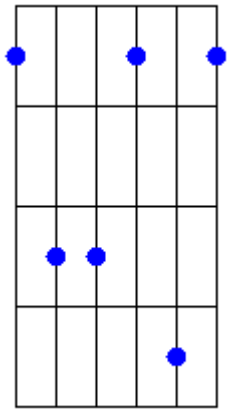
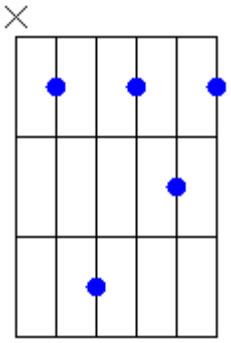
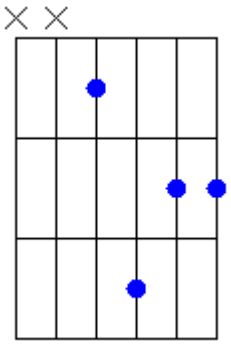
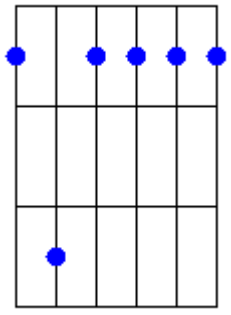
t_M7_5	-3;0;-1;-3;-3;-3	1	
t_M7_6	x;x;0;2;2;2	1	

Tabla 32: Acordes de Mayor Séptima

- **Acordes de menor séptima**

Key	Position	Barrechord	Imagen
t_m7_1	0;2;2;0;3;0	1	
t_m7_2	x;0;2;0;1;0	1	
t_m7_3	x;x;0;2;1;1	1	
t_m7_4	0;2;0;0;0;0	1	

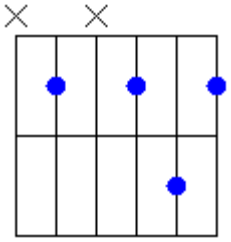
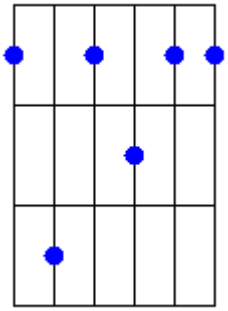
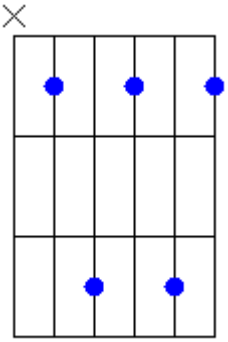
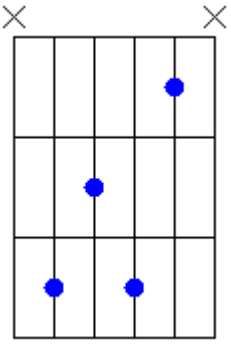
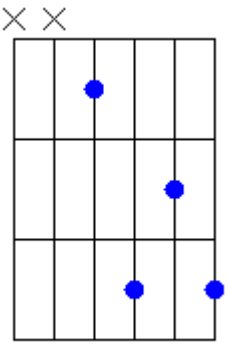
t_m7_5	x;0;x;0;1;0	1	

Tabla 33: Acordes de Menor Séptima

- **Acordes de séptima**

Key	Position	Barrechord	Imagen
t_7_1	0;2;0;1;0;0	1	
t_7_2	x;0;2;0;2;0	1	
t_7_3	x;0;-1;0;-2;x	1	
t_7_4	x;x;0;2;1;2	1	

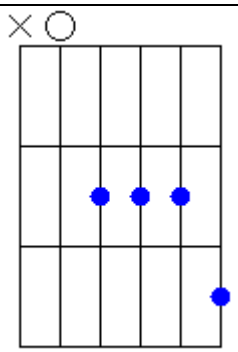
t_7_5	x;0;2;2;2;3	0	

Tabla 34: Acordes de Séptima