

NOTA:  = Contenido nuevo reentrega 04/07/2021

Proyecto 2º Semestre



SmartPiano

Grupo		S2-Projecte_B9 - Piano
ALUMNOS		
Nombres	Apellidos	Login
Arnau	Metaute Carrillo	arnau.metaute@students.salle.url.edu
Gabriel	Vallarta Angulo	gabriel.vallarta@students.salle.url.edu
Wesley	Lucas Mas	wesley.lucas@students.salle.url.edu
Marina	Ortega Picazo	marina.ortega@students.salle.url.edu
Pau	Pons Clotet	pau.pons@students.salle.url.edu

ÍNDICE

Resumen de las especificaciones del proyecto	3
Diseño de la interfície gráfica	4
Pantalla de bienvenida	4
Pantalla de 'Registro' y 'Inicio de sesión'	6
Pantalla del menú principal	11
Pantalla del menú lateral	13
Pantalla de 'Jugar'	16
Pantalla de 'Librería de listas de reproducción'	19
<i>Pantalla de listas de reproducción</i>	21
<i>Pantalla de canciones</i>	23
Pantalla de 'Entrenamiento'	26
Pantalla de 'Ajustes de controles'	27
Pantalla de 'Estadísticas'	29
Diagrama de clases	32
Metodología de desarrollo	37
SPRINT 1	37
SPRINT 2	38
SPRINT 3	40
SPRINT 4	42
Tiempo de dedicación	45
Conclusiones	47
Bibliografía consultada	49

1. Resumen de las especificaciones del proyecto

En esta ocasión, nos encontramos ante un proyecto de una magnitud más grande en comparación a la del 1er semestre, donde nuestro objetivo es crear un piano virtual completo a nivel no solo de opciones de modos de juego, sino también de usuario.

Es decir, el objetivo es crear una aplicación donde nosotros como usuarios, tenemos una cuenta donde tenemos registrado nuestras canciones tanto creadas por nosotros como aquellas que han sido sacadas de internet a través de webscrapping. Además de esto, con o sin canciones registradas a nuestra cuenta, podemos tocar el piano ya sea de forma libre o con motivo de entrenamiento con los controles que el usuario desee asignar a las teclas de su ordenador/portátil y mouse.

En general, este es el panorama en el que nos encontramos como grupo de desarrolladores de estudiantes de 2º año de carrera. A primera vista puede parecer asequible esta tarea, pero si indagamos más en las especificaciones que se nos piden llevar a cabo, podemos encontrar una serie de opciones extra que son imprescindibles a la hora de implementar dentro del proyecto.

Aparte de implementar un formulario de registro/inicio de sesión para el usuario, detrás de esta capa se ha de implementar previamente una base de datos que haga conexión con el proyecto en Git, y además permita llevar a cabo lo que se denominan las funciones CRUD (Create, Read, Update and Delete) para el desarrollo del proyecto al trabajar paralelamente con la base de datos de MySQL que se ha de implementar.

Aparte de la base de datos, y volviendo a los formularios de registro/inicio de sesión de los usuarios, también hemos de tener en cuenta los posibles errores que pueden haber al introducir un correo electrónico o contraseña erróneos según una serie de criterios establecidos en el enunciado original.

Al igual que el usuario tiene un perfil establecido desde el momento en el que se crea la cuenta, implica también que tiene su propia lista de canciones tanto públicas que comparte con otros usuarios, como privadas suyas restringidas a él mismo, los cuales puede reproducirlos o practicar con ellos a través del visualizador de notas del modo 'Play'.

Y por último, otro aspecto a destacar es que el usuario puede visualizar su actividad del uso que le da a la aplicación a través de dos gráficas distintas según las horas del día (eje x):

- El número de canciones reproducidas
- El número de minutos reproducidos en total

Dando así pues, no solo un acceso a una forma de entretenimiento que es el piano, sino también un acceso a la visualización del uso de tiempo que se le dedica a la aplicación en general.

2. Diseño de la interficie gráfica

En cuanto al diseño de la interficie gráfica general de la aplicación, en nuestro caso hemos optado por hacer uso en incorporar una gran variedad de layouts agrupados unos con otros con tal de que la información quede ordenada y estructurada lo mejor posible, desde BordeLayouts hasta GridLayouts, BoxLayouts, etc... e incluso aplicar LayeredPanes para el diseño del piano.

En los siguientes subapartados se entrará más en detalle en cada una de las capturas de pantallas diseñadas proporcionando no solo las capturas de mock-ups, sino también resultados finales y la planificación/colocación de los layouts y componentes ATW/Swing en cada pantalla.

NOTA: Las pantallas de indicación de layouts y componentes ATW/Swing han sido escritos sobre las capturas de mock-ups y no sobre los resultados finales como tal. Eso es porque tanto los mock-ups como los resultados finales siguen el mismo layout conceptualizado al principio del desarrollo, y además, durante el proceso de escritura de la memoria, aún había pantallas en proceso de recibir cambios estéticos, por lo tanto, no podíamos trabajar sobre ellos, y únicamente sobre los mock-ups.

Y otra cosa a mencionar es que algunas de las capturas fueron sacadas de Discord desde la pantalla de otro miembro del equipo. Por lo tanto, no tener en cuenta por favor algunos de los elementos como las flechas de cambio de usuario.

2.1. Pantalla de bienvenida

Como se puede observar a continuación, no ha habido mucho cambio de estética entre el diseño inicial conceptualizado y el final, además de que la única interacción presente en esta pantalla es la de presionar la tecla 'Espacio' para acceder a la pantalla de registro/inicio de sesión del usuario.

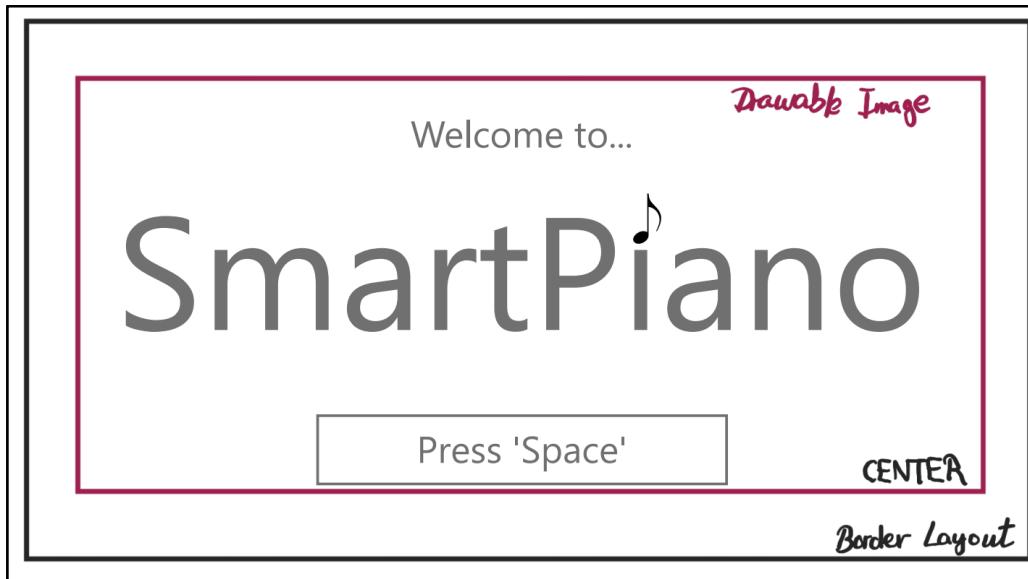
Captura(s) del **mock-up**:



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:



Captura(s) de los **componentes AWT/SWING y layouts** utilizados:



2.2. Pantalla de 'Registro' y 'Inicio de sesión'

Al igual que en la pantalla de 'Welcome', tampoco ha habido mucha diferencia entre la versión inicial conceptualizada y el resultado final, salvo quizás las medidas de anchura y altura de los campos de introducción de información. Pero aparte de eso, tanto la estética como la colocación de componentes es la misma.

Una cosa que sí es nueva que no estaba incorporada antes, es la adición de un checkbox de mostrar la información de las contraseñas de los usuarios tanto en el registro como inicio de sesión. Y del número de interacciones presentes sólo hay tanto los campos de información, como los checkboxes, botón de proceder y los de volver atrás de los mensajes de error.

Captura(s) de los **mock-up's** no solo de los formularios en sí, sino de los pop-up's que muestran los errores:

SmartPiano

— Register —

Username

Email

Password

Confirm password

— Login —

Username

Password

→

SmartPiano

Registration error!

The email must follow the following format:
emailfield@domain.com

The password must contain 8 or more characters.

The password must contain characters from at least two different character classes (upper- and lower-case letters, letters and symbols, letters and numbers, etc.):
e.g. p4ss_wOrd

The password must be composed of characters in the Roman alphabet or symbols on the US keyboard.

i.e. 長生きするウリレックス
أول ، إلواز ، برجي ، اجيبار الدورة
are not allowed

←

— Register —

Username

Email

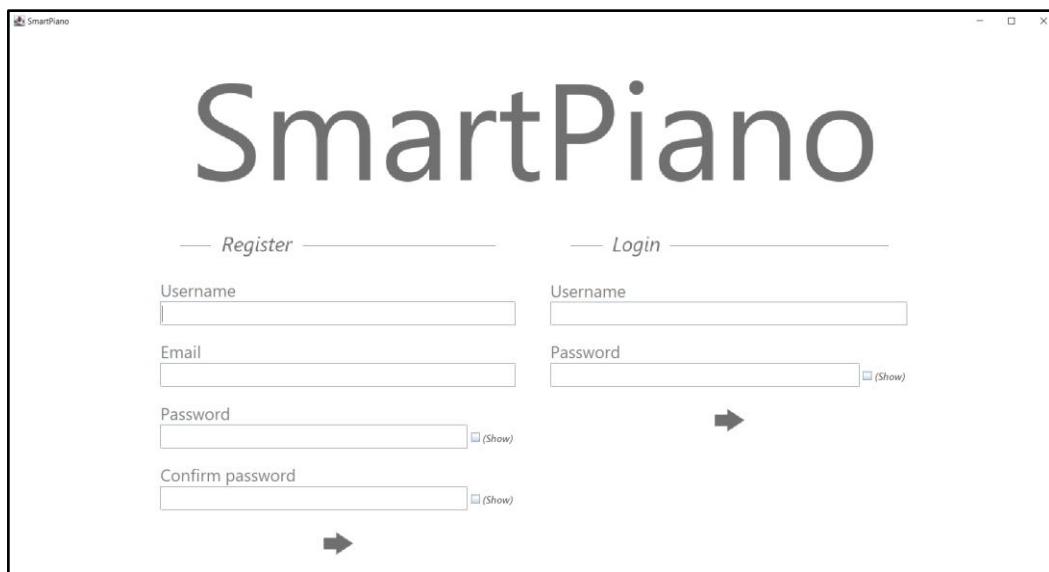
Password

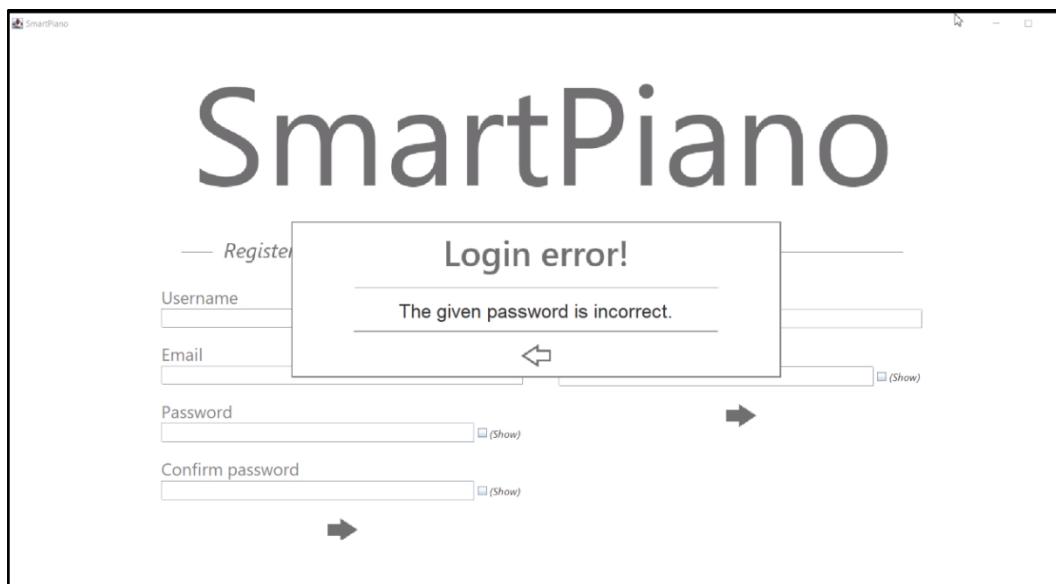
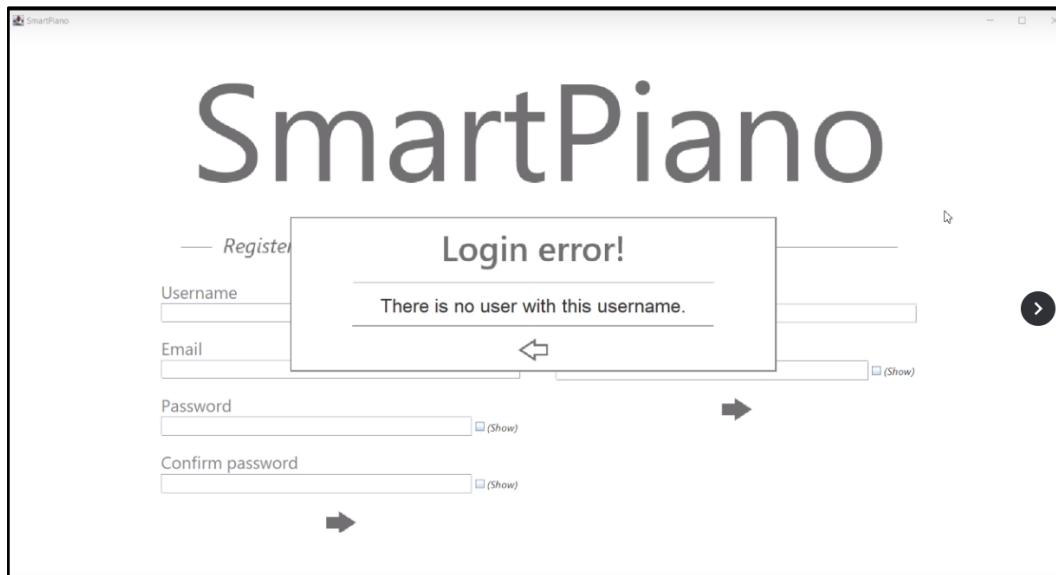
Confirm password

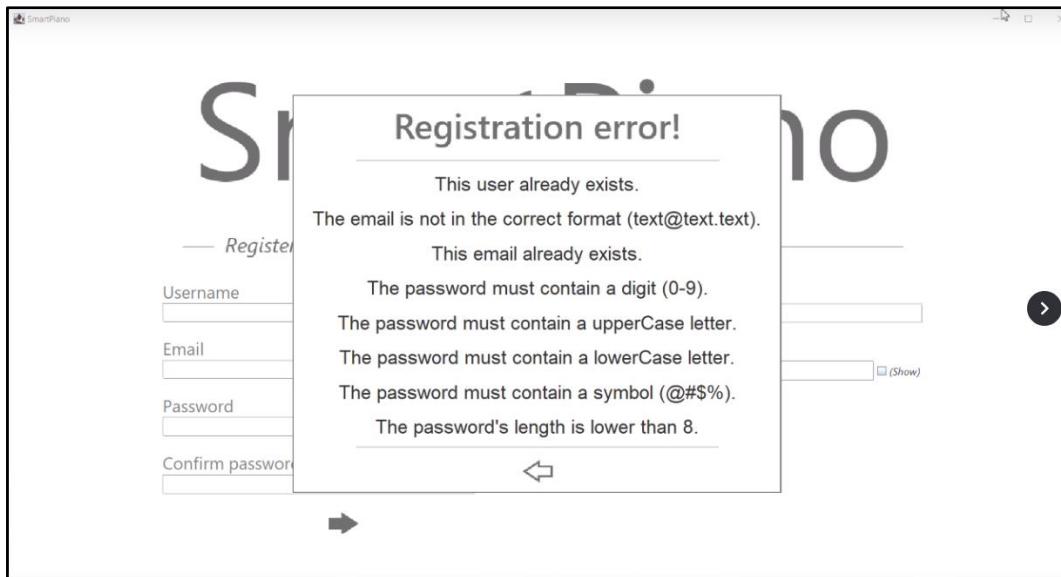
→



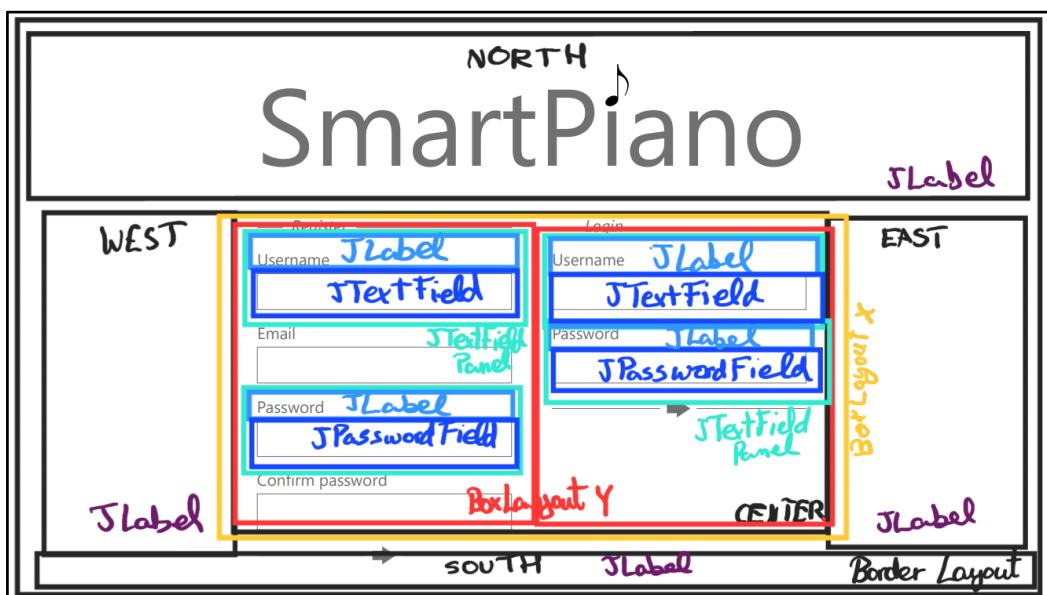
Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:

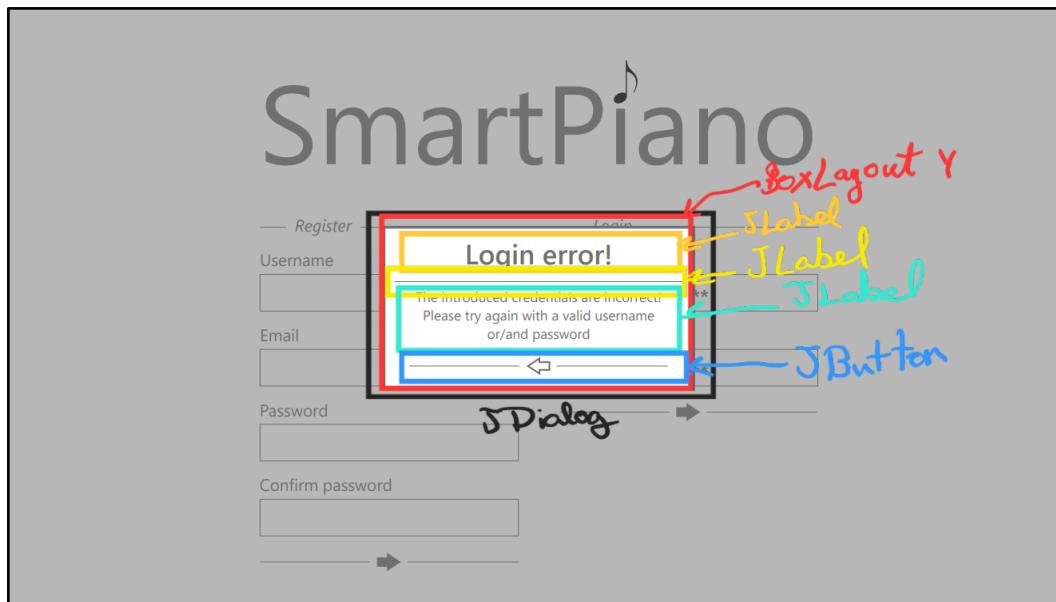
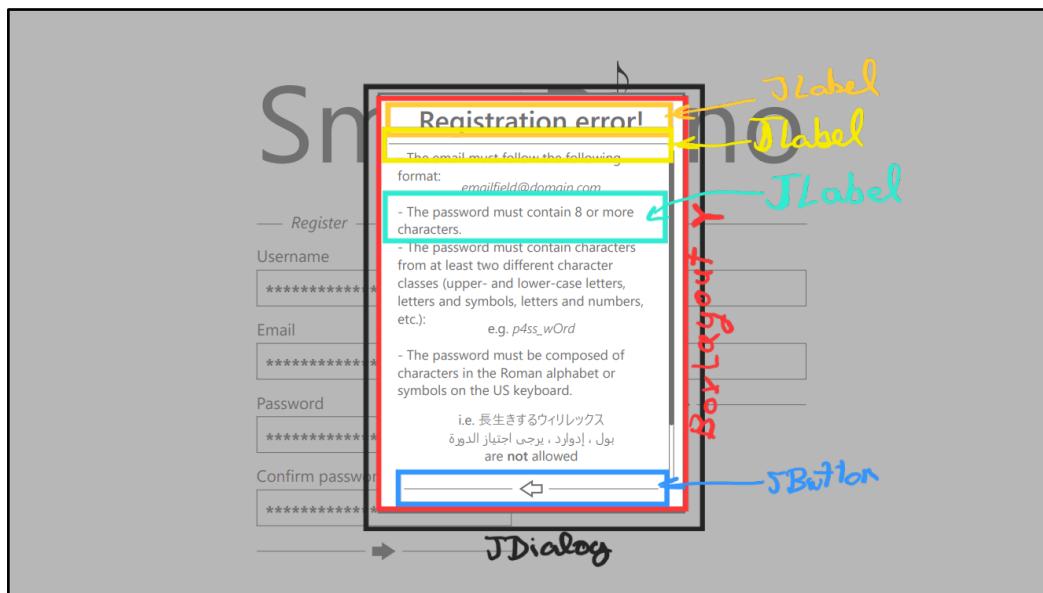






Captura(s) de los **componentes AWT/SWING y layouts** utilizados:

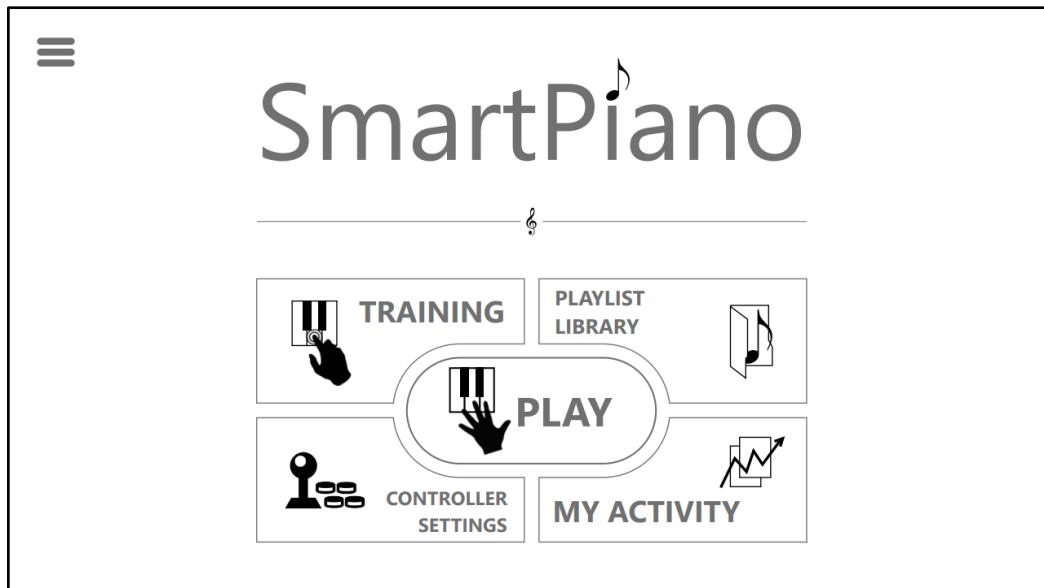




2.3. Pantalla del menú principal

Estéticamente no cambió mucho la versión inicial conceptualizada a la versión final. Y a nivel de interacciones, solo hay que tener una interacción a lo largo de todas las pantallas que vienen a continuación, que es el panel del hamburguer del side menu, y un grid “imaginario”, donde según donde se posicione el cursor, te llevará a una de las distintas secciones del programa.

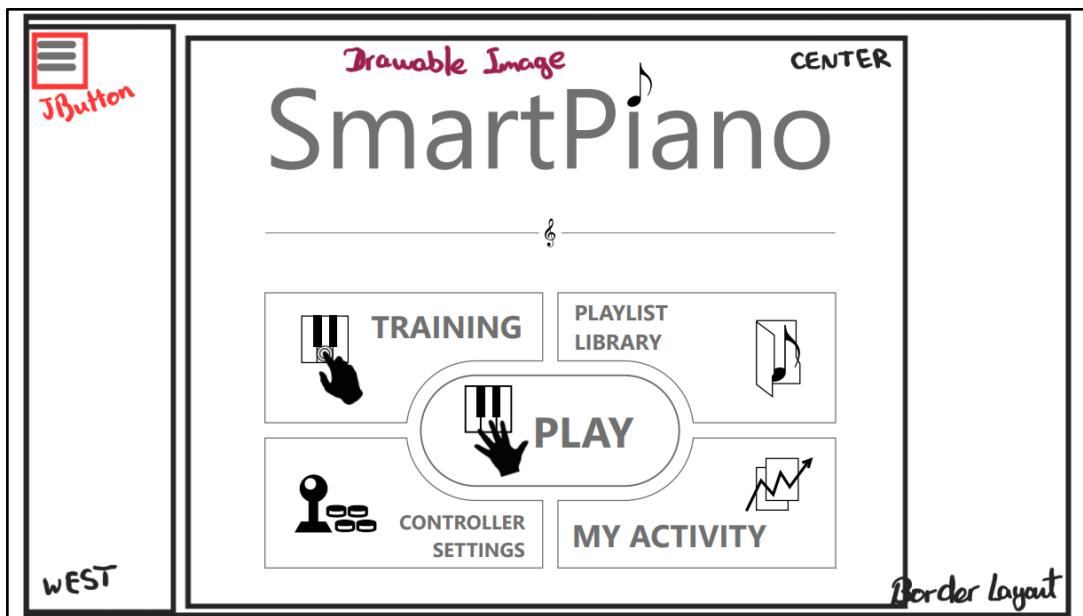
Captura(s) del **mock-up**:



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:



Captura(s) de los **componentes AWT/SWING y layouts** utilizados:



2.4. Pantalla del menú lateral

Presente en todas y cada una de las pantallas del programa. Donde según en qué sección nos encontremos, nos aparecerá un número de paneles adicional a los del menú principal y un orden distinto y aparición de paneles.

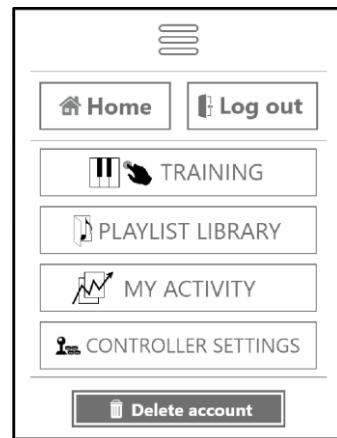
A nivel estético y de código si cabe remarcar que teníamos la intención principal de hacer el sidemenu una pantalla extra con una capa de opacidad negra detrás y el menú saliendo por el lado izquierdo. Pero debido a las complicaciones que tenía llevar a cabo eso, lo acabamos pasando a un formato de pop-up, que era más favorable que dedicarle su propia pantalla, y además el botón de cierre del pop-up se encuentra en el centro y no a la izquierda.

Captura(s) de los **mock-up's** de los diferentes menús laterales que se pueden encontrar según la sección en la que te encuentres:



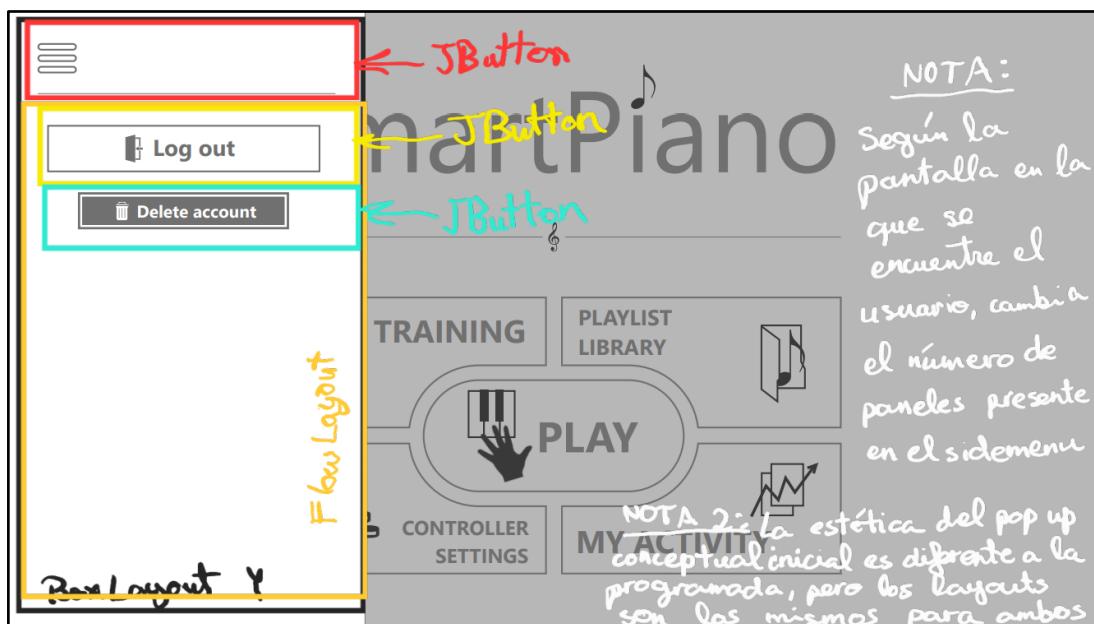
Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:





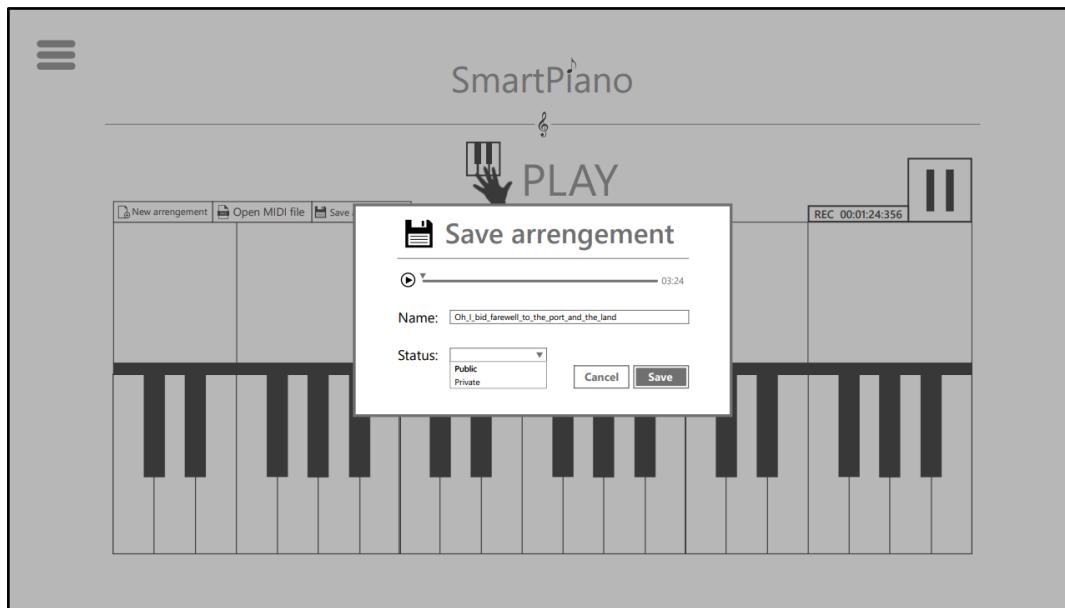
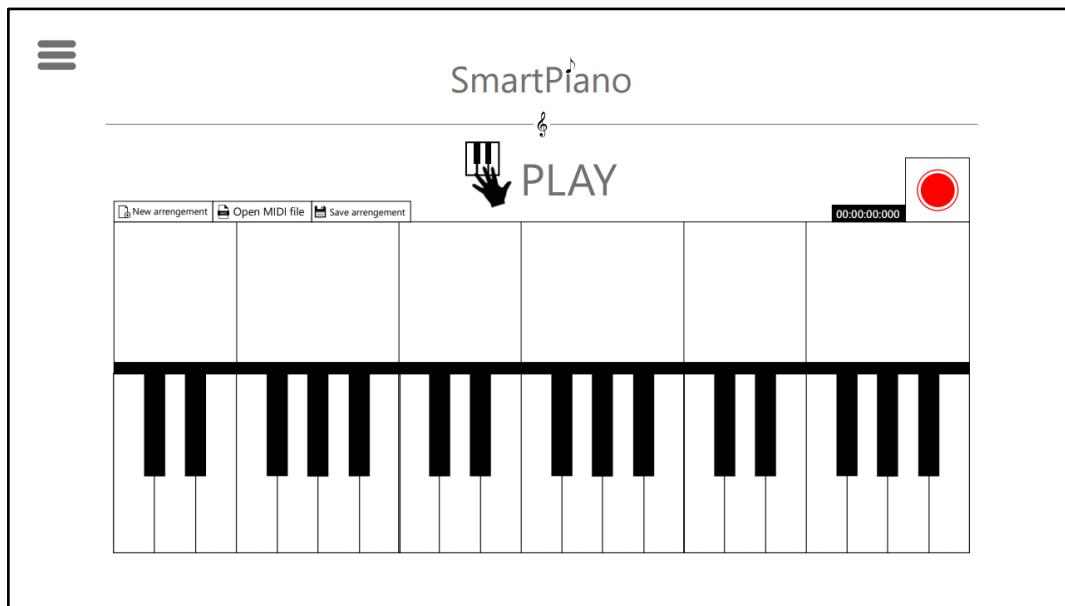
Layout que se utiliza en el resto de pantallas salvo el menú principal

Captura(s) de los **componentes AWT/SWING y layouts** utilizados:

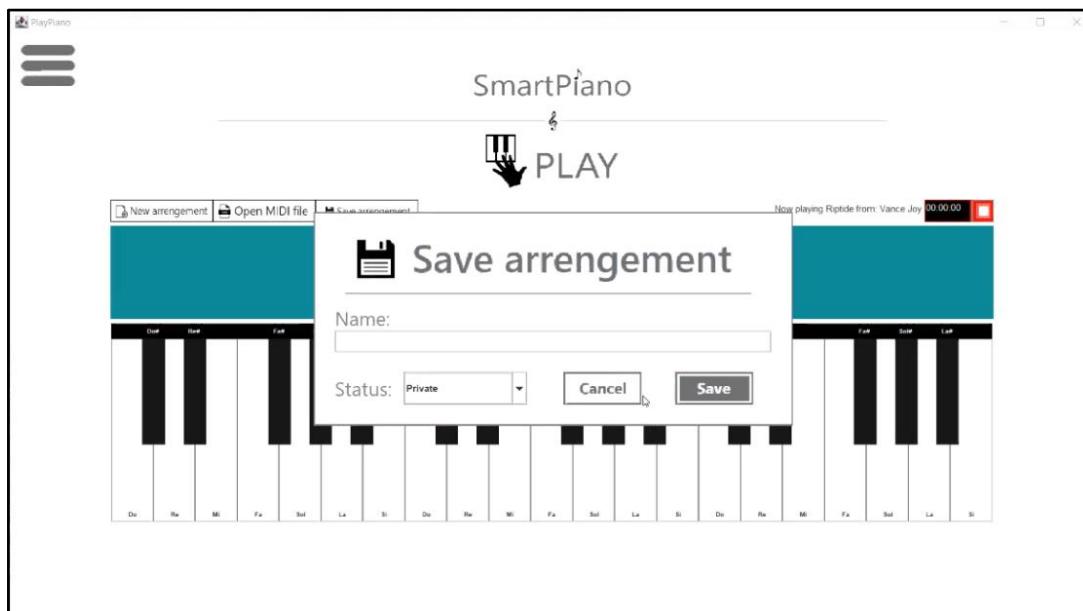
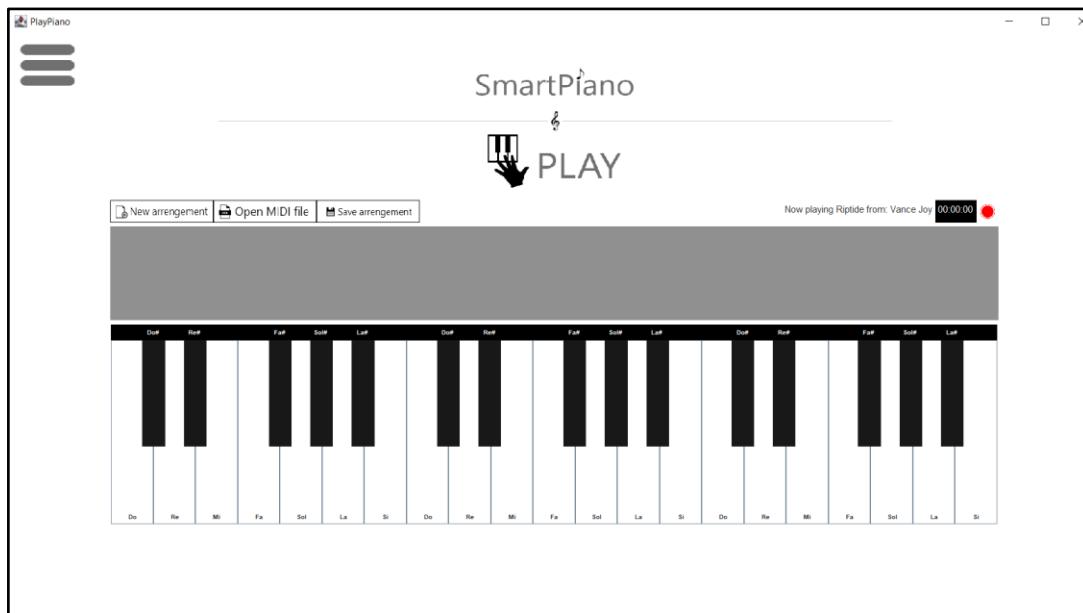


2.5. Pantalla de 'Jugar'

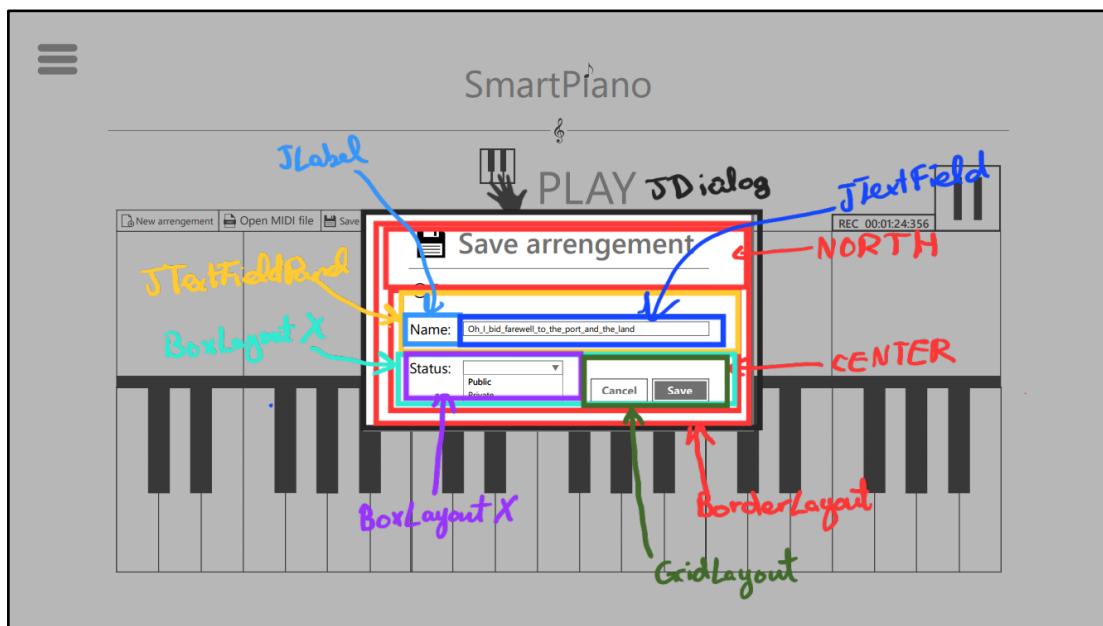
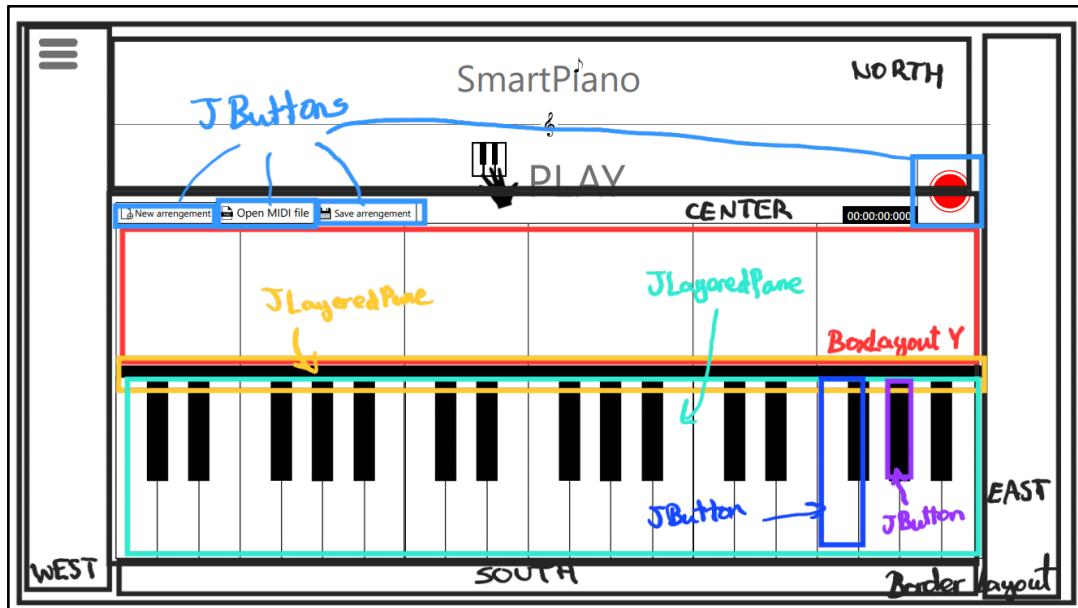
Captura(s) del **mock-up** de la pantalla de 'play' y las diferentes interacciones visuales que ocurren al interactuar con las opciones que se nos ofrecen como 'Open MIDI file', 'Save arrangement', 'Record' y 'Pause', además de ver las notas caer:



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:



Captura(s) de los **componentes AWT/SWING y layouts** utilizados:



2.6. Pantalla de 'Librería de listas de reproducción'

Para esta sección de **mock-up's**, hay dos pantallas diferentes que se caracterizan cada una por los elementos que almacenan, siendo este caso una las listas de reproducción mientras que la otra, sería de pistas musicales correspondientes a dicha lista de reproducción de la que proviene.

A nivel de estética, ninguna difiere de la conceptualización inicial a la del resultado final, salvo el botón de webscrapping, la cual tenía la función de que al interactuar con esta, saldría un pop up pidiendo donde guardar las canciones, pero al final lo cambiamos por un botón de ‘refresh’ donde el único propósito que tiene es que al darle, la playlist pública se actualiza con nuevas canciones. O así, o pasando cada 5 minutos. Depende. Pero en principio el cambio estético es ese, junto también la barra de progreso de escuchar música, que nos llevaba más tiempo de lo normal implementarla y decidimos no incorporar, junto con el botón de ‘Press Me’ que sustituye el botón de visualización de MIDIs (que ni en principio existía).

Actualización Reentrega:

Para la reentrega hemos añadido los siguientes elementos:

- **Pantalla de listas de reproducción**

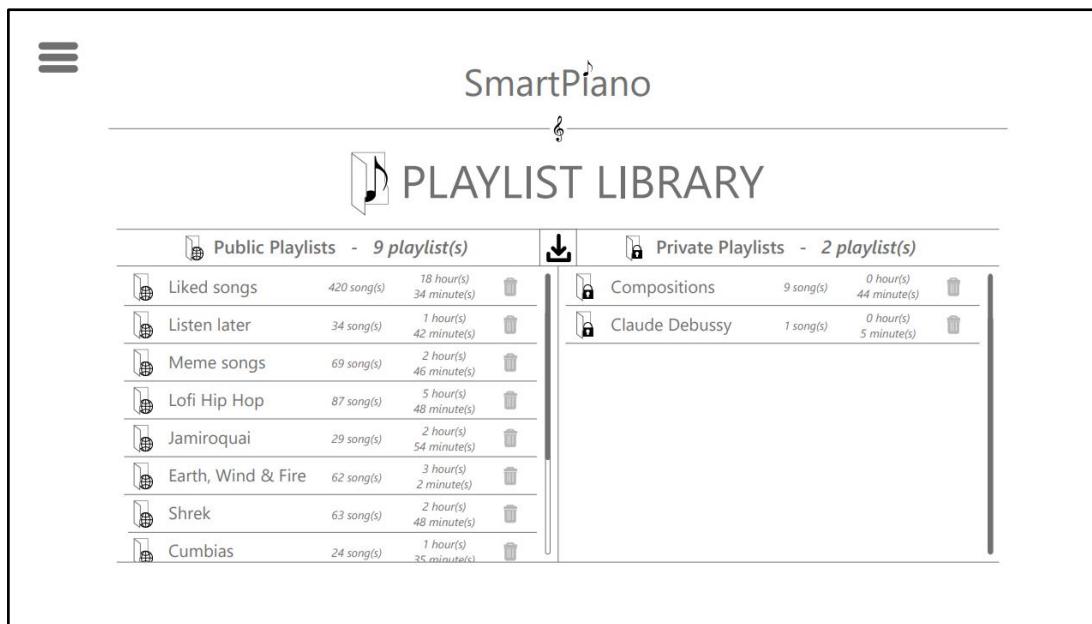
Un nuevo componente, un JButton en este caso, ejerciendo una de las funciones que nos faltó por añadir en la entrega de mayo, y es la función de crear una nueva carpeta con un nombre personalizado asignado por el usuario y un estado de ‘público’ o ‘privado’. Este botón se encuentra al lado del botón de webscrapping que hay en el centro entre ambos títulos de las listas de tanto pública como privada.

- **Pantalla de canciones**

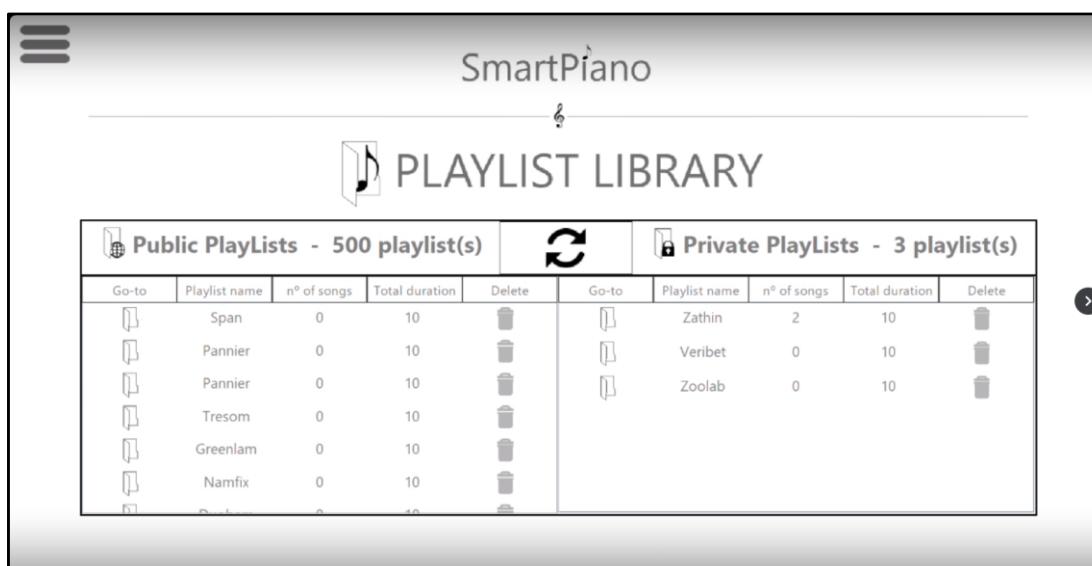
Hemos añadido la funcionalidad que faltaba por incorporar de mover una canción de una lista de canciones a otra. Esta funcionalidad se mantiene básica, donde el usuario al clicar sobre dicha opción solo se le pide que introduzca en un campo de información el nombre de la lista música de su interés donde quiere mover la canción. Una vez introducido el nombre y clicado sobre el JButton ‘Move’, el programa se encarga de comprobar si dicha playlist existe o no. En el caso de que no exista, el programa avisa con un pop up explicitando que no exista dicha playlist con ese nombre y que por favor introduzca el nombre de una playlist válida. En el caso opuesto, si el nombre de la playlist es válida, el programa cumple su función y mueve la canción a la playlist especificada.

(**Imágenes nuevas** de referencia a continuación en la siguiente página)

2.6.1. Pantalla de listas de reproducción



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:



SmartPiano

PLAYLIST LIBRARY

Public PlayLists - 498 playlist(s)				Private PlayLists - 2 playlist(s)					
Go-to	Playlist name	nº of songs	Total duration	Delete	Go-to	Playlist name	nº of songs	Total duration	Delete
	Cloud songs	41	10			Veribet	4	10	
	Span	0	10			Zoolab	8	10	
	Pannier	0	10						
	Tresom	0	10						
	Greenlam	0	10						
	Kanlam	0	10						
	T... T...	0	10						

SmartPiano

PLAYLIST LIBRARY

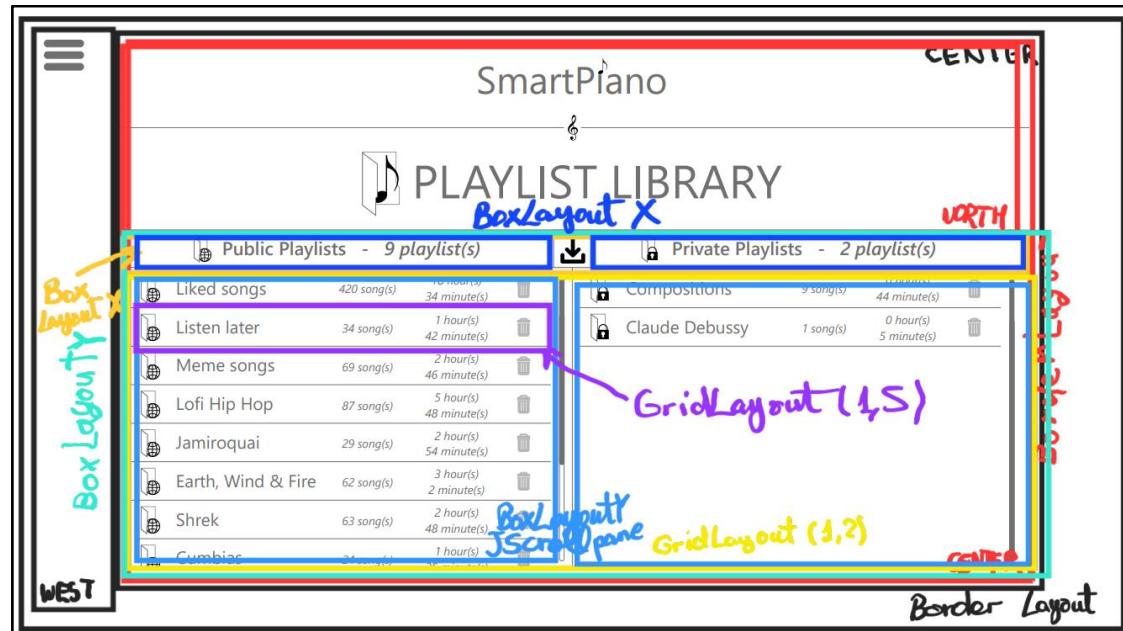
Public PlayLists - 498 playlist(s)				PlayLists - 2 playlist(s)			
Go-to	Playlist name	nº of song	Total duration	% of songs	Total duration	Delete	
	Cloud songs	41	10	4	10		
	Span	0	10	8	10		
	Pannier	0	10				
	Tresom	0	10				
	Greenlam	0	10				
	Kanlam	0	10				
	T... T...	0	10				

Create new playlist

Name:

Status:

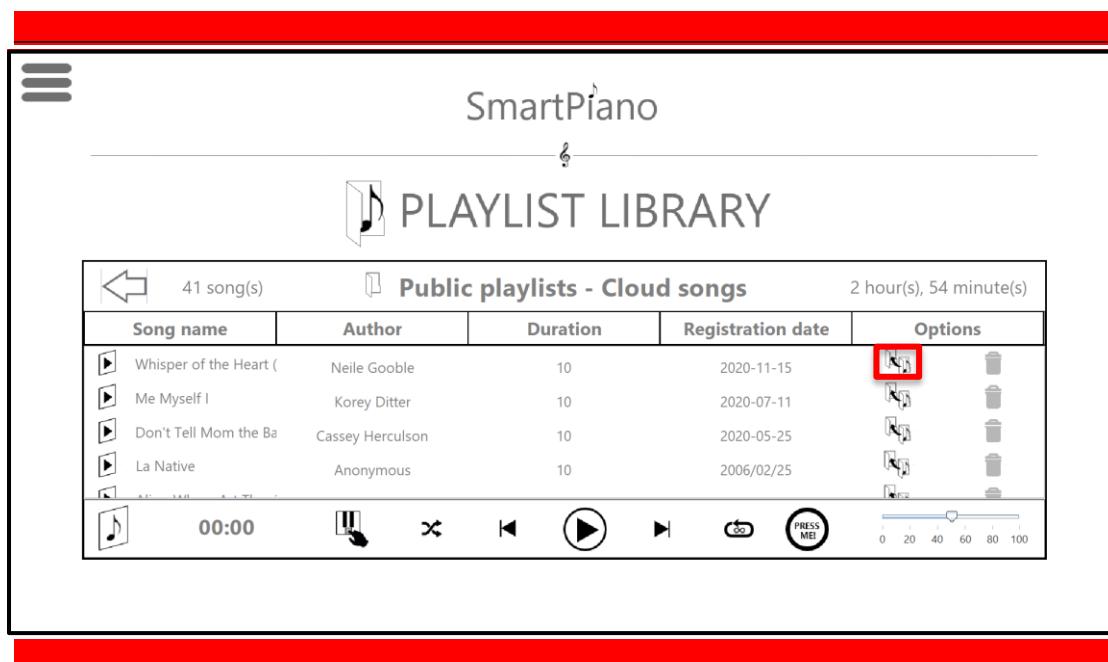
Captura(s) de los **componentes AWT/SWING y layouts** utilizados:



2.6.2. Pantalla de canciones

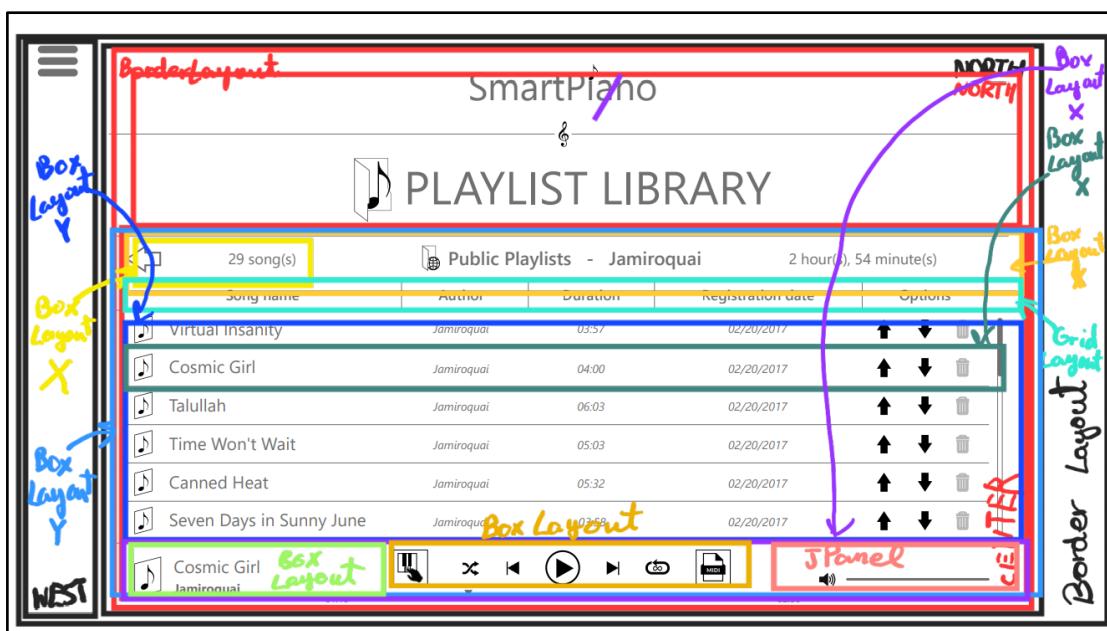


Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:





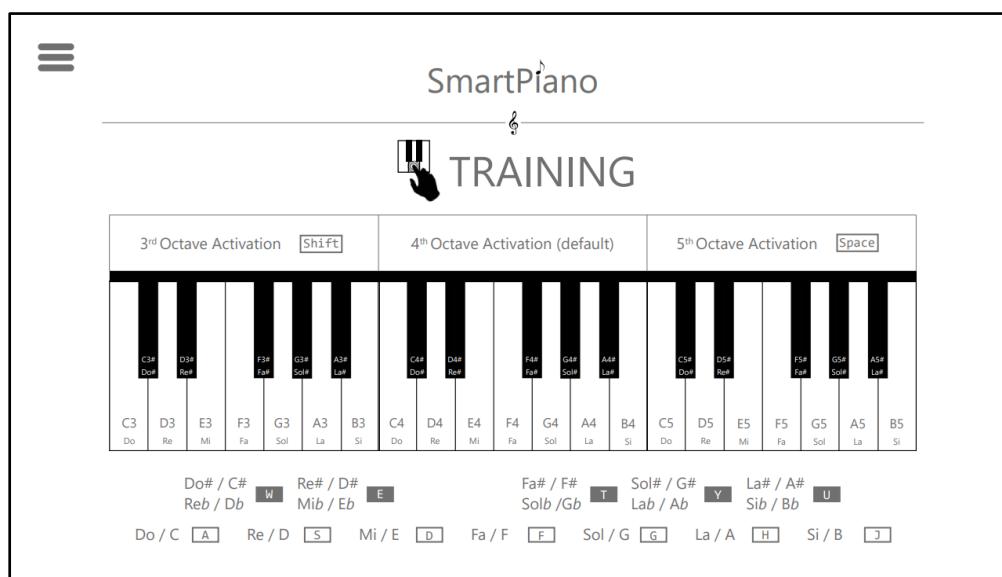
Captura(s) de los **componentes AWT/SWING** y layouts utilizados:



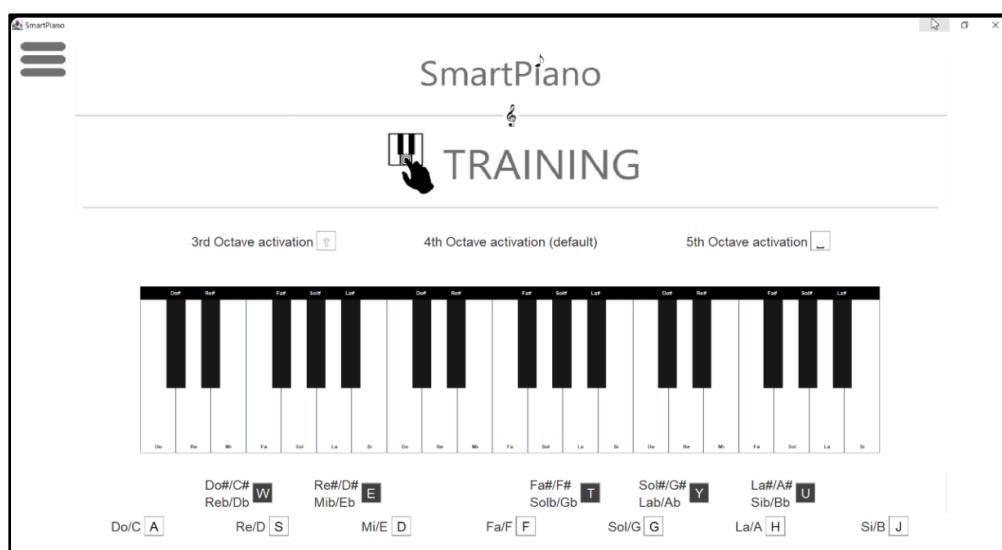
2.7. Pantalla de 'Entrenamiento'

En general se mantiene fiel a nivel de estética el resultado final comparado con la conceptualización inicial. El único detalle pequeño que cambia es la colocación de la indicación de las notas en las teclas negras, que en vez de ponerlas en la punta inferior de las teclas, las hemos colocado en el punto superior debido a complicaciones a nivel de código.

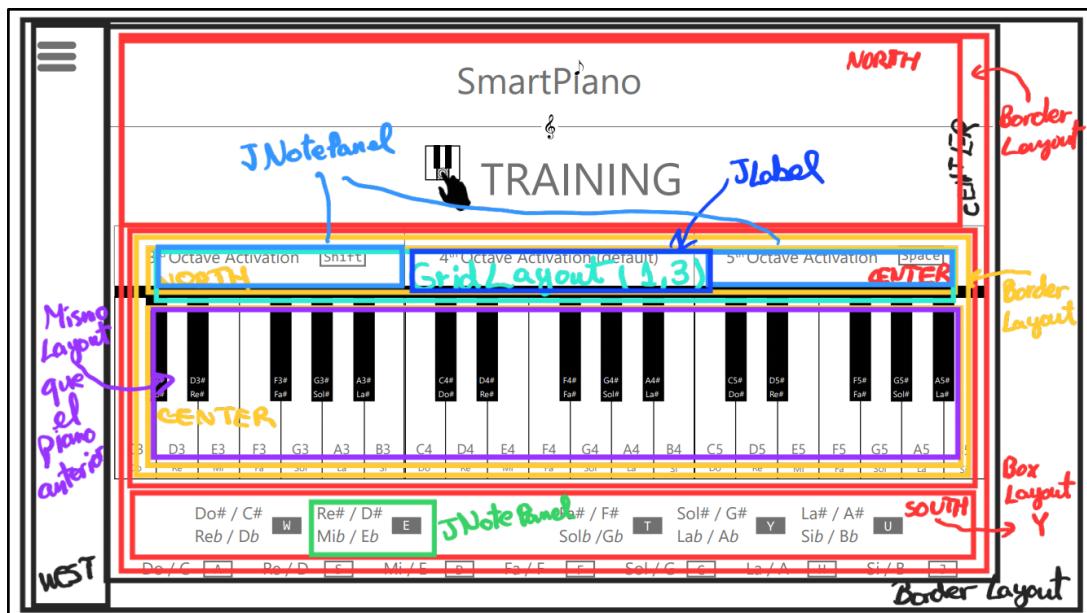
Captura(s) del **mock-up** de la pantalla de entrenamiento que muestra no solo los controles asignados a según qué teclas, sino también las teclas del piano con su nombre de referencia para poder guiar al usuario en todo momento:



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:



Captura(s) de los **componentes AWT/SWING y layouts** utilizados:

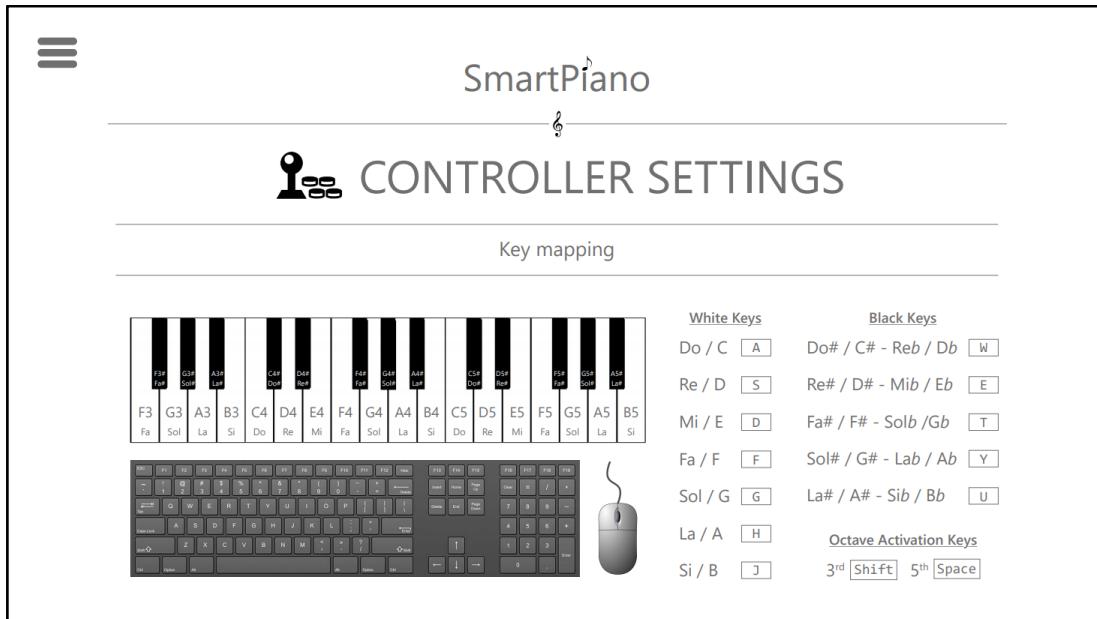


2.8. Pantalla de 'Ajustes de controles'

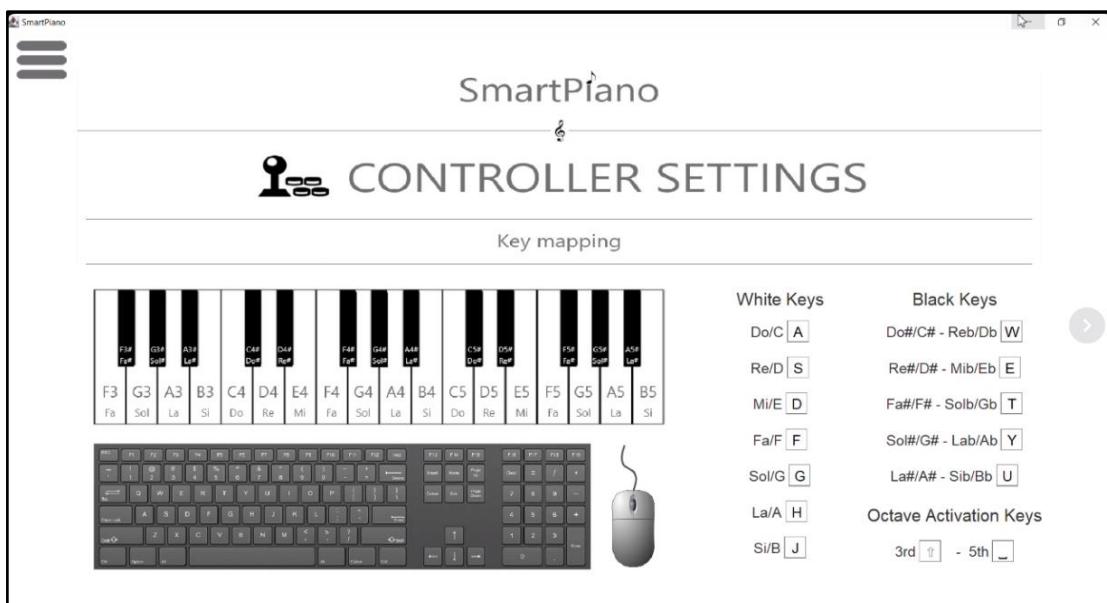
Al igual que antes, seguimos igual de fieles con la pantalla de 'Ajustes de controles', donde no hay ni un pequeño detalle en el que se diferencie de la versión original.

Sobre todo, como mera mención, la lógica seguida para la asignación de botones de teclas ha sido llevado a cabo de un Hashmap que nos agiliza y facilita más el proceso de asignación de teclas.

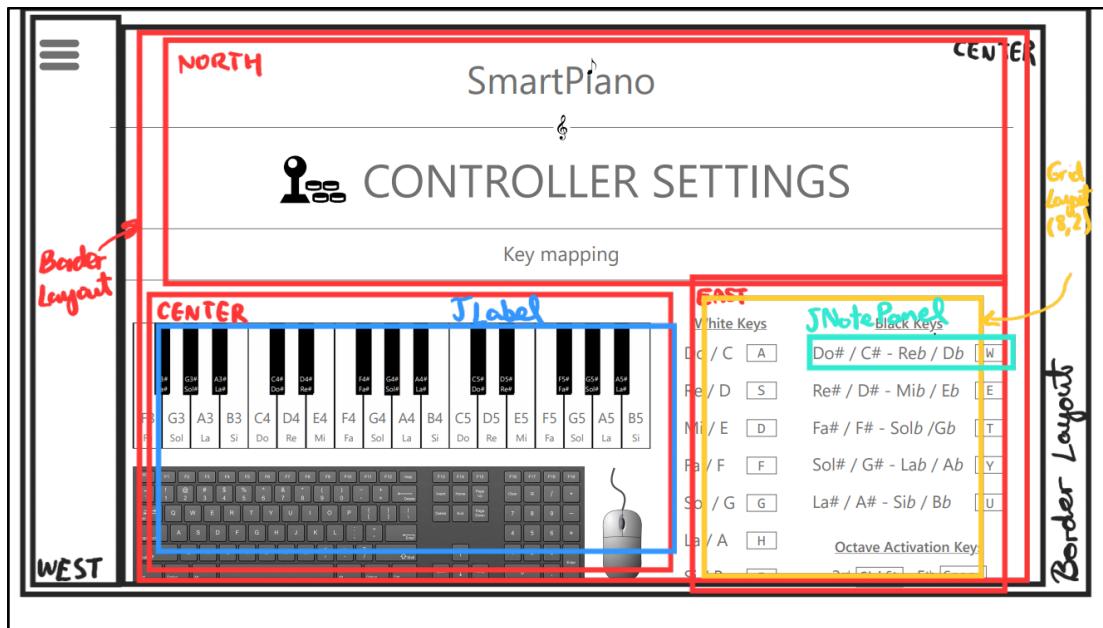
Captura(s) del **mock-up** de la pantalla de ajuste de controles donde se muestra una imagen del teclado del piano como la del ordenador que el usuario usa, junto con el ratón. Aparte de eso, también se muestra el listado de teclas blancas y negras que el usuario puede asignarles un botón, junto con las teclas de activación de las octavas:



Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:

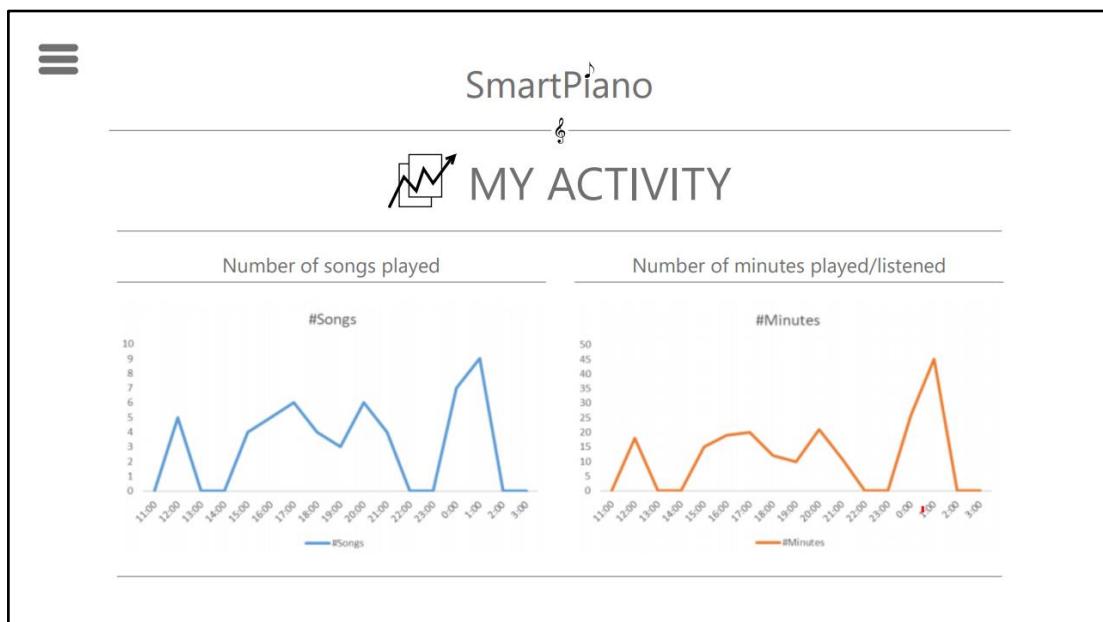


Captura(s) de los **componentes AWT/SWING y layouts** utilizados:

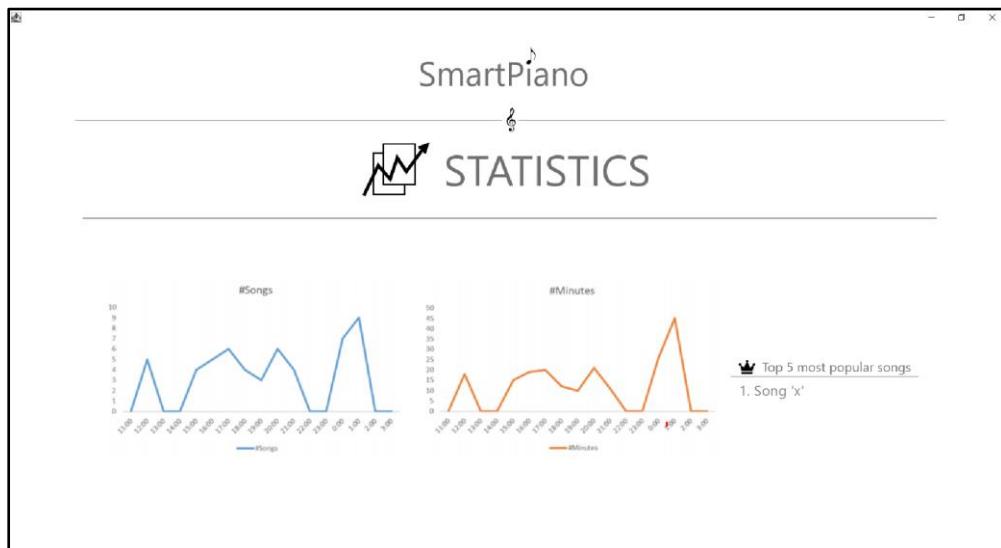


2.9. Pantalla de 'Estadísticas'

Captura(s) del **mock-up** de la pantalla 'Estadísticas' donde se puede visualizar la evolución no solo de las canciones escuchadas en base a la hora del día, sino también el número de minutos escuchados en base a la hora del día.

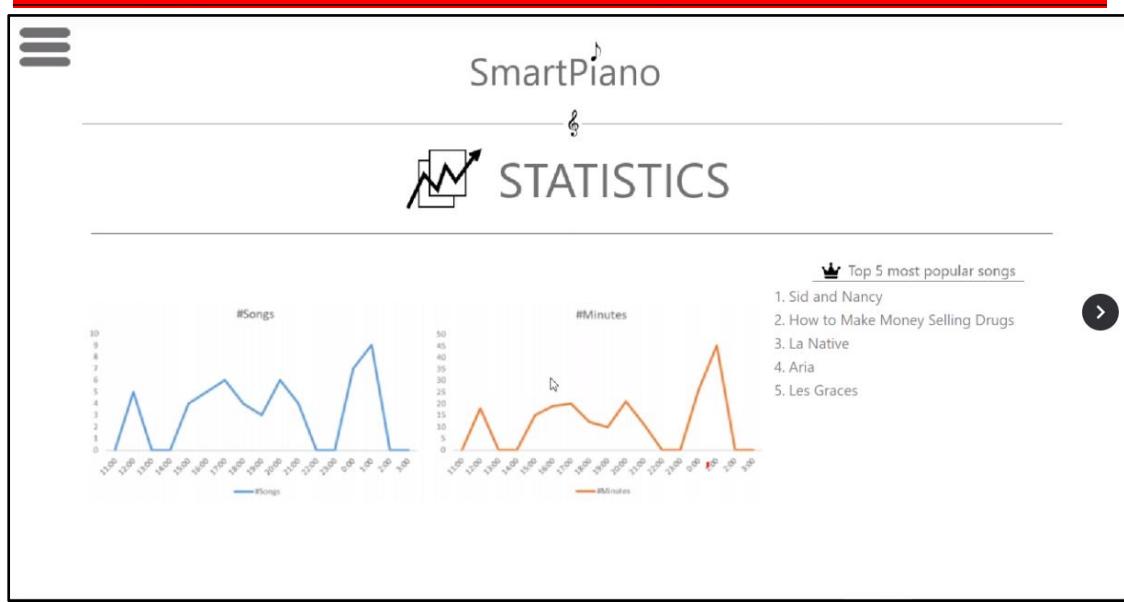


Captura(s) del **resultado final** del aspecto de la(s) captura(s) de pantalla:

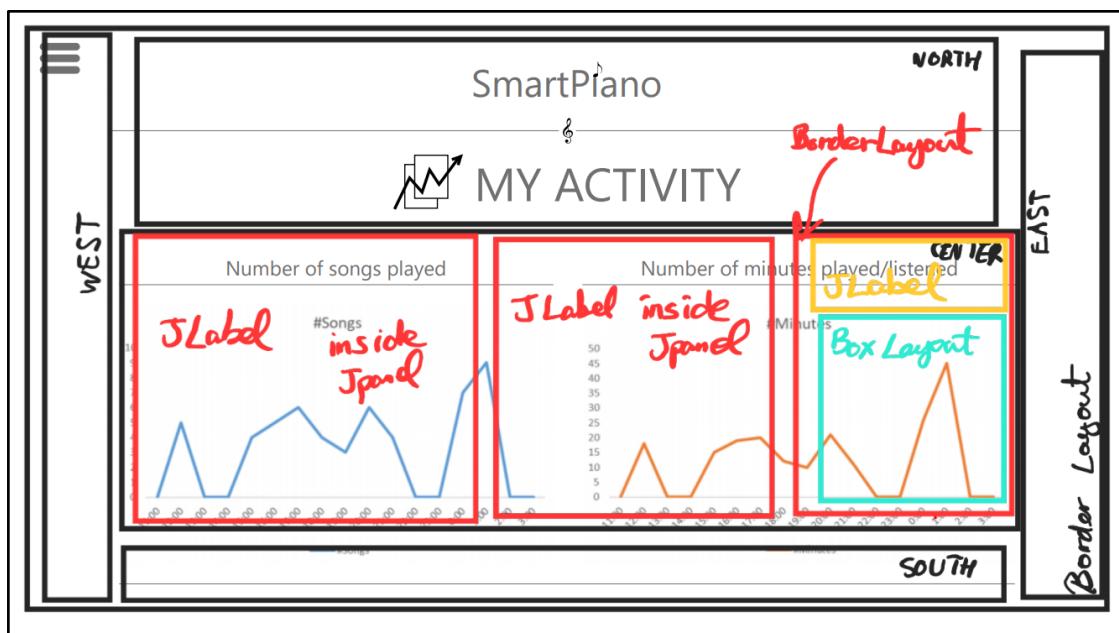


Actualización Reentrega:

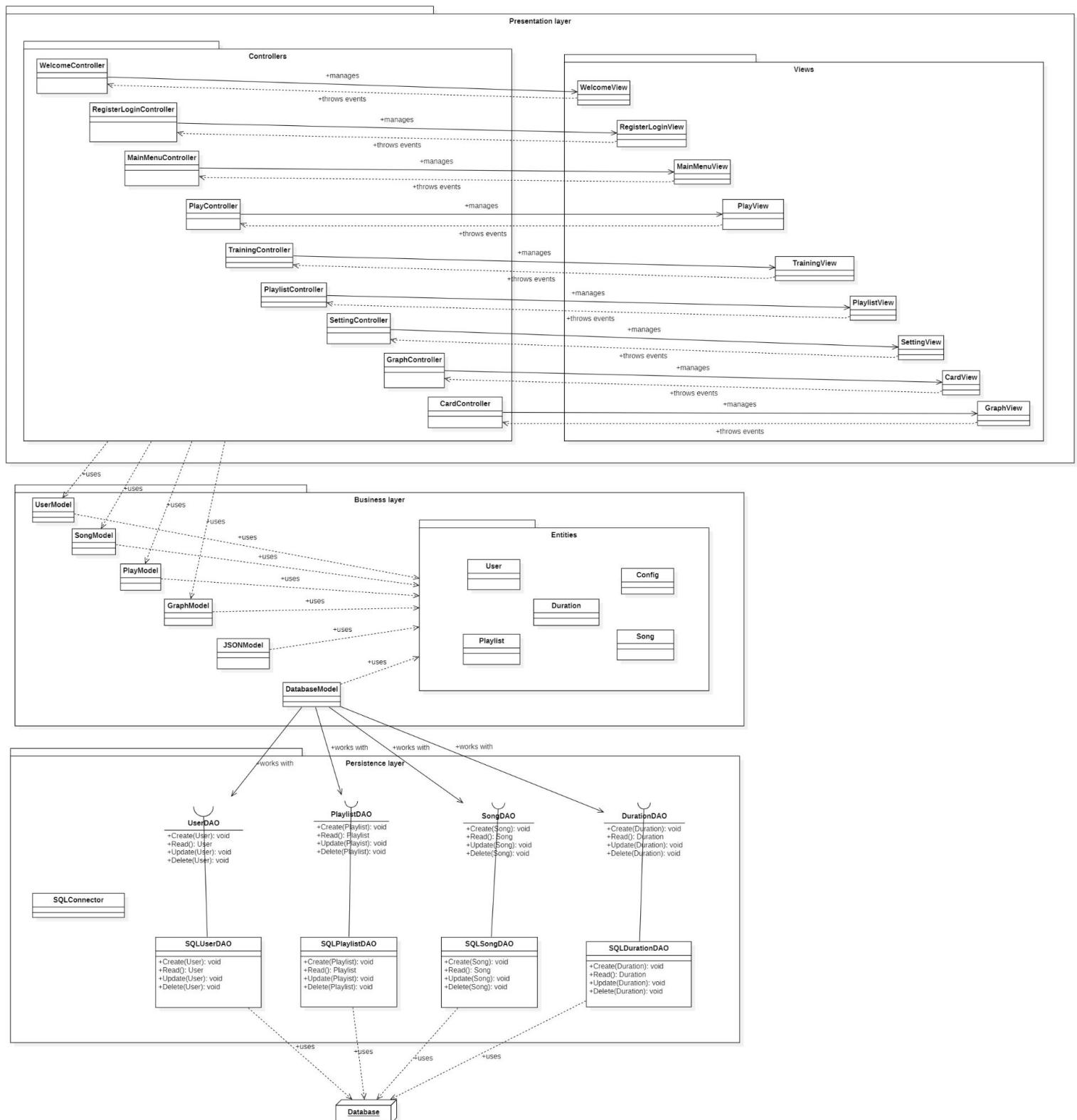
En esta ocasión, hemos aprovechado para intentar implementar como mínimo el ranking de canciones según el nivel de popularidad, y en general hacerlo funcional según está explicitado en la rúbrica del proyecto haciendo las conexiones necesarias con la base de datos para listar dichas canciones, entre más cosas.



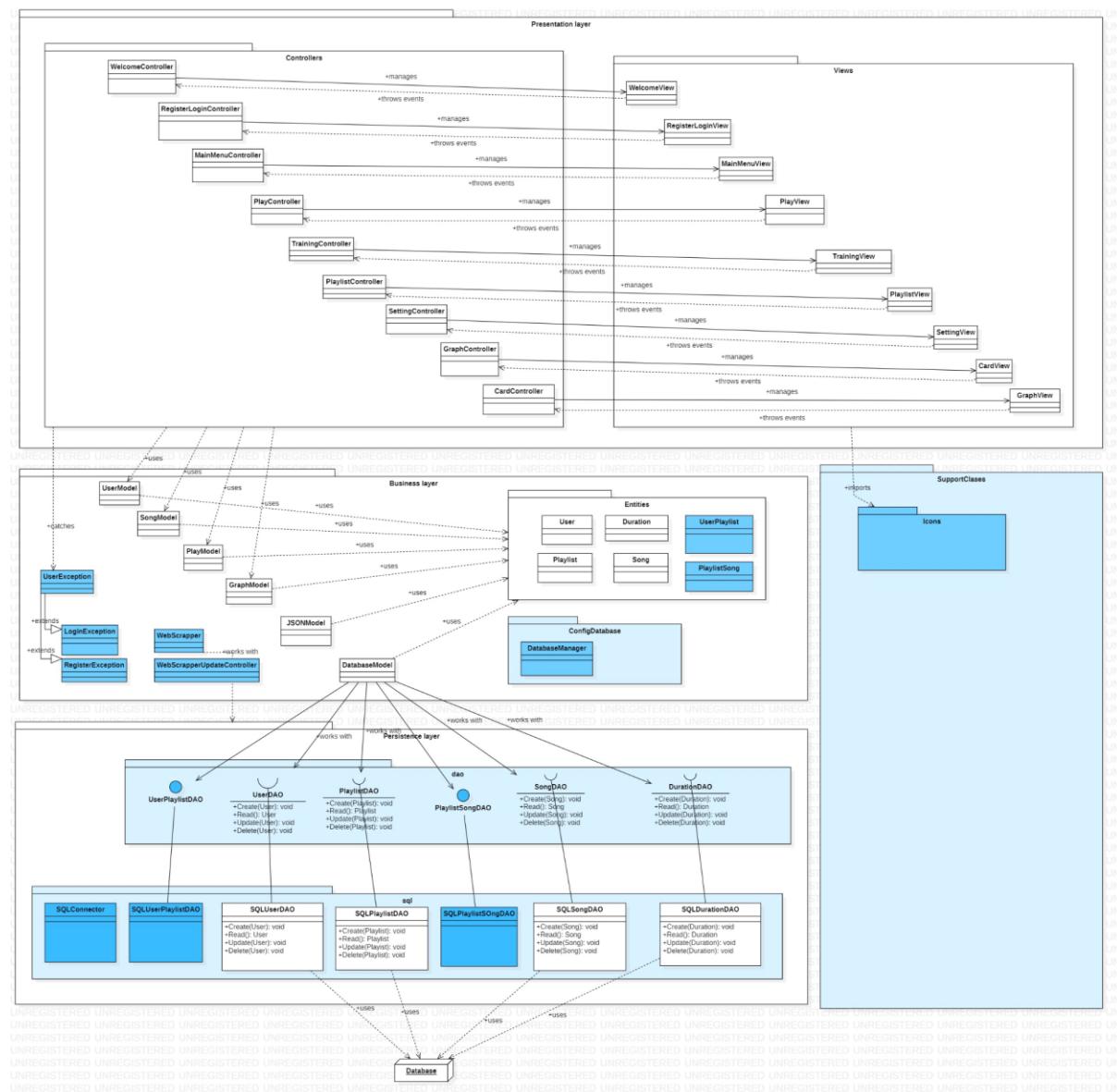
Captura(s) de los **componentes AWT/SWING y layouts** utilizados:



3. Diagrama de clases



Nueva versión



Como se puede observar, ha habido una evolución respecto el UML planteado al inicio del proyecto durante el primer semestre. Para empezar, durante el desarrollo del Swing, como teníamos que aprovechar componentes agrupados en layouts para ser reutilizados en diversas views, hemos creado clases de ‘soporte’ que nos permite reutilizar esos mismos componentes una y otra vez aprovechándose desde clases llamadas como por ejemplo ‘JPasswordFieldText’, ‘SongPanel’ para la generación de canciones dentro de las listas de playlists, ‘PlaylistPanel’ para generar playlists públicas y privadas del usuario, etc... y así seguido.

Aparte de la adición de la carpeta de clases de soporte, también han habido otras adiciones necesarias como el paquete de DAOs, los diferentes Exceptions como el User, Login y Register, una entidad extra de UserPlaylist en la carpeta de Entities, etc... En la capa de Persistence por ejemplo también está dividido en dos directorios llamados 'sql' y 'dao's. Y por último, como aspecto remarcable a destacar, es la adición de las clases WebScrapper y WebScrapperUpdateController que apunta al package de persistence layer por medio del Database Manager para conservar la abstracción.

Las clases de Exceptions que fueron utilizadas a nivel de los registros y logins han sido añadidas a nivel de Business en paralelo a los modelos encargados de gestionar dichas actividades.

En general, la estructura MVC se ha mantenido la misma que en el primer diseño conceptual inicial del Sprint 1, y la única novedad ha sido la adición de nuevas clases, orden general de directorios con sus clases correspondientes, etc... Este ha sido el cambio significativo y evolución del UML.

Actualización Reentrega:

En esta ocasión, al ser una reentrega, hemos añadido un montón de cosas que no se encontraban originalmente en la versión de la entrega de mayo, ya sean nuevos packages, nuevas clases, relaciones entre clases nuevas y ya existentes, etc...

- **SupportClasses**

Se han añadido las siguientes nuevas clases:

- CreatePlaylist: Pop up de la creación de una playlist nueva.
- DeletePlaylistPanel: Pop up donde se pregunta al usuario si está seguro de querer eliminar la playlist deseada.
- DeleteSongPanel: Pop up donde se pregunta al usuario si está seguro de querer eliminar la pista musical deseada.
- MoveError: Pop up donde se notifica al usuario de la inexistencia de una playlist cuyo nombre se especificó en el campo de información, no existe.
- MoveSong: Pop up donde se pregunta al usuario el nombre de la playlist a donde quiere mover la pista musical deseada.

- **Business layer -> ConfigDatabase**

- Se ha añadido un nuevo package con el nombre de GestionJSON donde dentro de este se encuentra no solo el fichero .json de la configuración inicial del programa en sí, sino la clase que se encarga de gestionar la lógica detrás de la extracción de datos del .json, llamada 'Config'.
- El DatabaseModel pasa a llamarse DatabaseManager por un tema de nomenclatura más que nada, que era incorrecto previamente.

- Eliminación del JSONModel porque hemos reestructurado el código de nuevo y metido la lógica del JSONModel en otras clases más adientes que respetan los conceptos de abstracción/encapsulación y MVC/Layered Architecture.

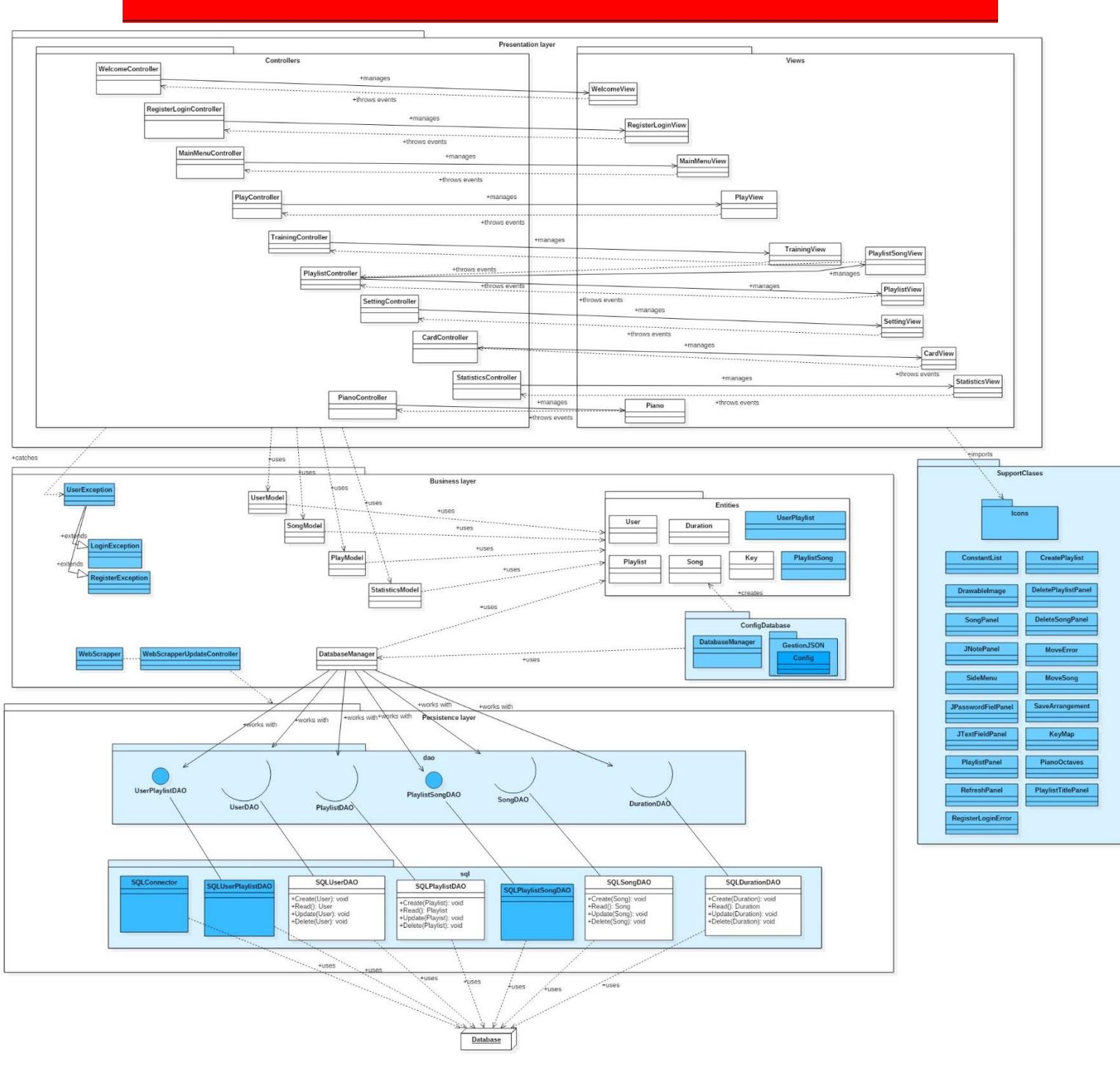
- **Business layer -> Entities**

- Se ha añadido la clase Key dentro del package. Esta clase se encarga de la estética y lógica del funcionamiento de las notas que compone el piano en sí.

- **Presentation layer -> Controllers & Views**

- Se han añadido la vista y controller del Piano, llamados respectivamente 'Piano' (view) y 'PianoController', que se encargan de todo aquello relacionado con la estética del piano y el buen funcionamiento de los componentes que lo forman.
- Se ha añadido la clase PlaylistSongView al Controller, relacionado con el controller de PlaylistController, ya que tanto PlaylistView como PlaylistSongView funcionan a través del mismo controller al venir cada uno de una pantalla a la otra en la misma sección.

(Mirar diagrama nuevo siguiente página)



4. Metodología de desarrollo

SPRINT 1	
Arnau Metaute Carrillo	Durante el primer sprint, mis tareas consistieron en crear el modelo UML sobre el que se basaría luego el proyecto y la base de datos, y la creación y el set-up del proyecto en git (bitbucket). Para hacer la comprobación de que estaba todo bien escrito, se subieron ficheros de prueba para confirmar que no hubiera errores. Una vez acabado, empecé a mirar en profundidad Swing, pues sería una de las personas que trabajaría en el front-end.
Gabriel Vallarta Angulo	Comencé el proyecto por ayudar a montar la estructura básica de la Base de Datos que se encargaría de almacenar toda la información de nuestras entidades. Con la ayuda del esquema inicial de Marina, he creado el script de creación e importación de datos temporales con los que pudieramos comenzar a hacer testeos de funcionalidad de la base y mejor planteamiento del uso de los datos del juego. Este csv global para la inserción de datos lo realicé desde Mackaroo. Una vez hecho esto, comencé a mirar el tema del registro y login de nuestros usuarios. Comenzando por la lógica de cada validación que podríamos llegar a necesitar para ambos procesos.
Wesley Lucas Mas	Encargado de realizar los mock-ups de todo el proyecto que se han observado previamente en el apartado 4 de la memoria. Un total de 10 capturas de pantalla que conforman la estructura de la interficie gráfica del programa, combinándolo con aproximadamente 15 elementos gráficos interactivos tales como pop-ups de interacción, submenús, botones interactivos, etc... Todo este apartado llevado a cabo a través de un software especializado en el diseño de wireframes/mock-ups con el nombre de Adobe Xd. Y por último, pero de menor importancia que el punto anterior, ha dado apoyo a Marina Ortega Picazo con el planteamiento del modelo conceptual/relacional de la base de datos a implementar en este proyecto.
Marina Ortega Picazo	Realización del modelo conceptual y relacional de la base de datos junto con el soporte de Wesito. Además me encargué de pasar a MySQL el Script de la creación de las tablas hecho por Gabriel ya que dicho Script se encontraba en lenguaje SQLite. También hice el Configuration file a Json y su consiguiente lectura en Java, concretamente en la clase Config.
Pau Pons Clotet	Mi participación en el principio del proyecto estuvo muy afectada por graves problemas familiares los cuales ya lo he hablado con

	Pol (el tutor de la asignatura) y Joan Claudi (mi tutor personal). Por el momento no pude contribuir demasiado pero sí que seguía el trabajo y progreso de mis compañeros para no quedarme atrás y poder incorporarme tan pronto mis circunstancias personales/familiares me lo permitieran.
--	--

SPRINT 2	
Arnau Metaute Carrillo	<p>Para empezar el proyecto, me di cuenta de que una de las pantallas más importantes era el menú, dado que es el nexo de unión entre todas las diferentes partes del programa. No obstante, pronto me di cuenta de la primera gran dificultad del programa: según los mockups que hicimos, los botones del menú principal tenían formas personalizadas, por lo que no podíamos utilizar botones normales. Además, en cuanto se cambiaba el tamaño de la pantalla, la imagen desaparecía. Así pues, se creó la primera clase auxiliar del programa: la drawableImage, la cual permite que la imagen se escale con la pantalla. Para los botones, se hizo uso de un MouseListener, el cual, según la posición del mouse en la pantalla, al hacer clic, te llevaría a un sitio u otro. Esta función, además, se hizo escalable, por lo que funciona independientemente del tamaño de la ventana.</p> <p>La siguiente tarea fue la creación del menú lateral del programa. Después de probar diferentes combinaciones (cambiar el menú lateral, utilizar layered panes...) me di cuenta de que todos daban algún error, por lo que no servían. Después de comentarlo con el grupo, nos decantamos por una opción más viable, pero que cumplía la misma función: utilizar un JDialog personalizado.</p> <p>Una vez puesto el menú lateral, éste no servía de nada si no habían más pantallas pero, a su vez, estas no se verían sin un card view / controller que lo gestionase. De esta manera se creó una pantalla extra que sirviese como placeholder mientras las demás aún no estuviesen hechas, pero que permitía probar que el cardLayout funcionaba.</p> <p>Después de hacer el merge a dev para utilizarlo como base para el proyecto, empecé la siguiente pantalla: los settings de la aplicación. No obstante, durante este sprint no se hizo mucho más que hacer la estructura básica de la pantalla y la colocación de algunas imágenes.</p>
Gabriel Vallarta Angulo	Siguiendo con el planteamiento de los procesos y validaciones a tener en cuenta para la ejecución de un registro y un login he implementado completamente la parte del registro. El registro tuvo al menos cuatro iteraciones ya que cada vez encontraba una manera más óptima para validar la información introducida por los usuarios. Después de probar y de informarme más, decidí que sin duda las expresiones Regex serían el mejor camino para poder gestionar los datos. Cree variables finales en la clase del

	<p>Register y un par de funciones para gestión de datos.</p> <p>Una vez que tenía un sólido registro funcional me enfoque a implementar toda la parte de las Excepciones. Al principio no fue nada fácil ya que quería permitirle al usuario saber todos los errores que pudo haber tenido durante su registro o login y no solamente uno a la vez. Para esto, con la ayuda de mis clases de excepciones, cree una serie de errores que al ser cometidos serían concatenados hacia un mensaje general con todo error explicado de manera clara y en orden de aparición según los requisitos de contraseña del MIT sugeridas en el enunciado de la práctica.</p>
Wesley Lucas Mas	<p>Encargado de desarrollar a nivel de SWING la pantalla de registro del usuario, además de la de inicio de sesión con la referencia de los mock-ups en mano en todo momento, mientras que Gabriel Vallarta Angulo se encargó de desarrollar la lógica de dicha sección para luego más adelante poder unirlo.</p> <p>Aparte de lo mencionado, he intentado estar activo y receptivo en todo momento dando opiniones, apoyo/ayuda, críticas de mejoría hacia el resto de trabajo de mis compañeros.</p> <p>NOTA: A la hora de programar código, he ido trabajando en proyectos separados. Uno que era de prueba en local, y el otro el del grupo por Git. Es por esta razón que en lo que respecta al registro de actividades del grupo en Git, yo he hecho menos commits y pushes en comparación al resto de mis compañeros porque la mayor parte del tiempo lo pasaba trabajando en un proyecto aparte en local, en vez del grupal de Git.</p>
Marina Ortega Picazo	Población de las diferentes tablas mediante los csv hechos por Gabriel. Creación de las diferentes entidades de la base de datos en el proyecto en Java, el SQLConnector, SQLSong, SQLPlaylist, SQLPlaylistSong entre otros. Creación de métodos para extraer información, borrar o añadir a la base de datos.
Pau Pons Clotet	En este sprint por fin me he podido empezar a integrar en el grupo después de varias semanas de problemas familiares y personales que he mencionado con anterioridad. Se me encomendó por parte del grupo mi primera tarea, la de desarrollar e implementar la vista de la pantalla de welcome. Esta, debía de tener el nombre del programa y un botón indicando la tecla espaciadora para entrar a la pantalla de registro. Empecé a documentarme para la realización de dicha view, y empecé su implementación con un JFrame y conseguí implementar en mi ordenador la view, de manera que al pulsar la tecla correspondiente se activará el método para pasar a la siguiente view. No obstante, no conseguí implementarlo con clases separadas (view y controller) porque no conseguí hacer funcionar el registerController.

SPRINT 3	
Arnau Metaute Carrillo	<p>Para el tercer sprint, primero de todo acabé la tarea que empecé en el segundo sprint: hacer el panel de settings, es decir, relacioné cada una de las teclas del piano con una tecla del teclado del ordenador. Tal y como se tenía diseñado, se hizo que estas teclas pudieran ser totalmente personalizables. Para llevar a cabo todo este trabajo, se añadieron nuevas funciones y características que no se habían visto hasta el momento: el HashMap y el KeyListener, las cuales permitían detectar la tecla pulsada y almacenarla según una "key". Además también creamos una clase auxiliar que almacena todos los actionCommands que fuéramos creando a medida que hacíamos la práctica.</p> <p>La siguiente vista que se quiso hacer fue la de training. Dado que el piano no estaba creado aún, lo que se quiso fue que pudiese sonar al pulsar las teclas del teclado. Originalmente, se pensó en utilizar audios midi para hacer los sonidos del piano, aunque más tarde se cambiarían por sonidos hechos por el synthesizer de midis integrado en java. Para hacer el cambio de octavas, se creó otra clase auxiliar llamada PianoOctaves, que permitía hacer dicha distinción.</p> <p>Por último, una de las cosas relevantes que hice en el sprint fue limpieza de código. Éste no estaba nada optimizado, y las clases llegaban a ser demasiado largas para lo que deberían. Una de estas optimizaciones fue la separación del menú lateral en una clase a parte, además de clases menores que se iban repitiendo alrededor de la práctica, como la nombrada JNotePanel.</p>
Gabriel Vallarta Angulo	<p>En este sprint dejé de un lado la capa de lógica pura y dura del Business y decidí explorar el mundo del Swing. Me he puesto a la par de Wesley a idear la creación del piano para poder eventualmente crear la PlayView y hacer del piano algo ejecutable y funcional. Después de demasiado intentos y errores he conseguido crearlo a partir de la implementación de un JLayeredPane lo cual fue extremadamente útil ya que el problema más grande al que me enfrenté fue el hecho de no poder colocar botones por encima de otros por obvias razones, pero con la ayuda de las capas del JLayeredPane he conseguido colocar las notas negras (layer uno) por encima de las blancas (layer cero). Una vez que al piano ya le logré implementar dentro de su propia clase los métodos necesarios para generar todo el piano en función del número de octavas y teclas deseadas por nosotros vino la parte más visual del PlayView.</p> <p>Como ya lo he dicho, este piano estaba ya implementado y listo para ser puesto en su sitio dentro de la View, para ello he creado toda una estructura de Jpanels mediante la cual me logré apegar lo más posible al Mockup propuesto por nuestro diseñador de vistas Wesley.</p> <p>A pesar de todo el trabajo que me implicó desarrollar el piano,</p>

	<p>debo decir que la implementación de los textos de las notas en cada tecla y sus respectivos controllers fue también un reto interesante, aprendí bastante sobre textos en botones y cómo customizar los espacios de texto dentro de Swing, volví a utilizar el JLayeredPane, pero esta vez para nombrar las teclas negras del piano por la parte superior del piano.</p> <p>Ya que la parte visual estaba completa y acorde al diseño deseado, implementé toda la parte de los controllers a nivel de la clase piano para poder ser tocado mediante el clic del mouse y por parte del PlayView para poder ser tocado a través del teclado del ordenador, esta parte claramente con la ayuda de Arnau quien fue el que implementó esta parte originalmente para la funcionalidad del training de nuestro juego.</p>
Wesley Lucas Mas	<p>Este sprint al ser más largo en comparación a los otros debido a las fiestas de LaSalle, en ese período de tiempo de dos semanas ha estado encargado de los siguientes temas:</p> <ul style="list-style-type: none"> - Intento de desarrollo de la pantalla 'Play' del piano. Y digo 'intento' porque esta tarea nos la encomendamos yo y Gabriel Vallarta Angulo. En mi caso intenté desarrollar esta pantalla de tal forma que al redimensionar la ventana del programa, los elementos no se moverían y quedaría 'estáticamente' colocados en su sitio. El problema es que al tratar con las teclas del piano, era muy difícil mirases por donde lo mirases utilizando diferentes layouts y agrupaciones de estos. <p>Al final, quien acabó haciendo el mejor piano fue Gabriel, haciendo uso del LayeredPane, que no nos convencía mucho al principio por el asunto de que al redimensionar la pantalla la distribución de los elementos quizás acababa patas arriba.</p> <p>Pero cómo se veía bien y además los botones de las teclas eran funcionales, nos decantamos por seguir adelante con su versión, ya que estaba más funcional de lo que yo intenté desarrollar en su momento.</p> <ul style="list-style-type: none"> - Por otro lado, está el diseño de las pantallas de las listas de reproducción, tanto la pantalla de librería de carpetas como las de pistas musicales. <p>Una vez hecho eso, unimos la parte visual con la lógica tanto la de nivel de código como la de la base de datos para permitir la visualización de los paneles de carpetas de música como las de pistas musicales, además de añadir las interacciones de los botones del reproductor musical.</p> <ul style="list-style-type: none"> - Después, también me encargué de diseñar a nivel de SWING al igual que los puntos anteriores, una pequeña demo de lo que se vería en la pantalla de 'Estadísticas' del programa, donde se mostrarían las dos gráficas y el ranking Top-5 de pistas musicales. Al no haberlo implementado del todo, quisimos añadir algo.

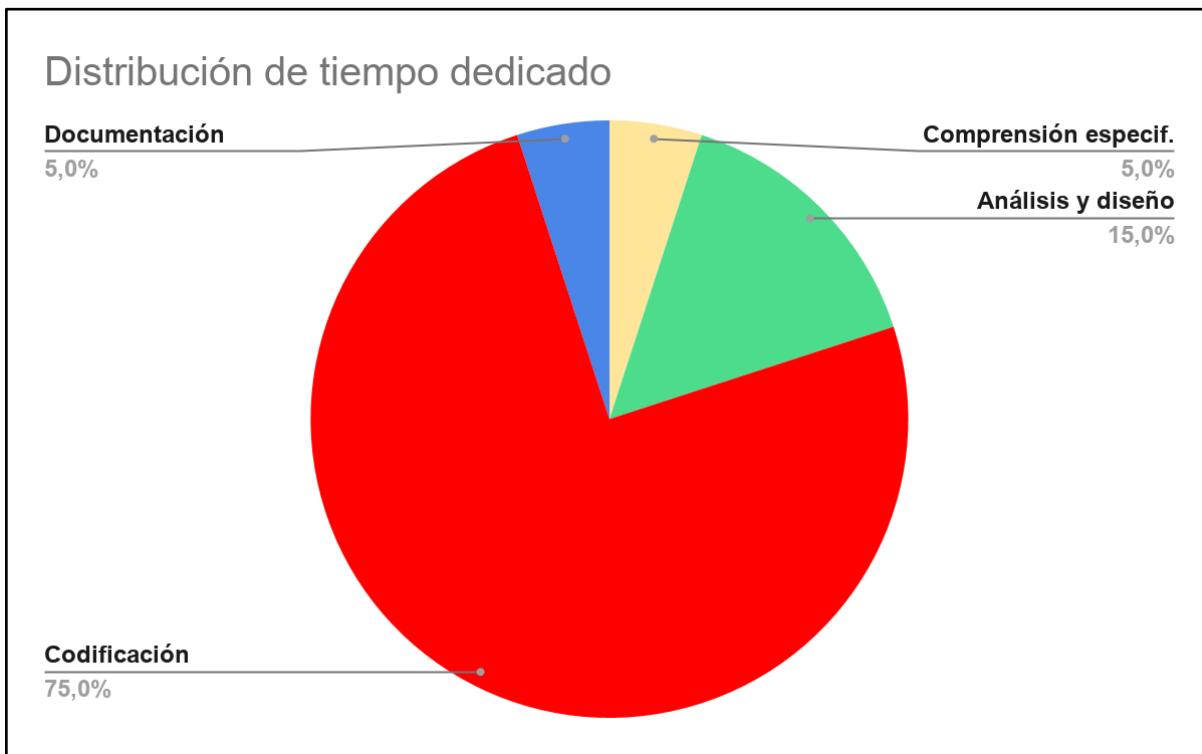
	<p>En general ha sido eso, aparte de participar activamente en el proyecto ayudando a mis compañeros en lo que necesitase además de reunirnos constantemente en persona en el día a día para hablar sobre cuál sería la siguiente tarea a realizar.</p> <p>NOTA: A la hora de programar código, he ido trabajando en proyectos separados. Uno que era de prueba en local, y el otro el del grupo por Git. Es por esta razón que en lo que respecta al registro de actividades del grupo en Git, yo he hecho menos commits y pushes en comparación al resto de mis compañeros porque la mayor parte del tiempo lo pasaba trabajando en un proyecto aparte en local, en vez del grupal de Git.</p>
Marina Ortega Picazo	Corrección de los errores previos como el funcionamiento del SQLConnector que no conseguía conectar con la base de datos o la adición errónea de las FK de las diferentes entidades. Creación de diferentes métodos como por ejemplo coger de la base de datos todas las playlist privadas del usuario o coger todas las playlist públicas en general.
Pau Pons Clotet	Finalmente, y gracias a la ayuda de Pol conseguí implementar (y aún más importante, entender el funcionamiento) de la implementación con clases separadas de view y controller, dónde la view se encarga de mostrar la ventana con sus contenidos, y el controller se encarga de detectar (o mejor dicho, "escuchar") la tecla correspondiente. Entre este sprint y el siguiente también empecé a documentarme para conseguir incorporar en el proyecto mi view.

SPRINT 4	
Arnaud Metaute Carrillo	<p>Lo primero que hice en este sprint fue relacionar la pantalla de register/login con la base de datos pues, aunque la lógica y la view estaban hechas desde hace tiempo, aún no se habían unido. Para eso tuve que modificar el código sustancialmente, ya que con lo que se tenía, hacer la conexión era difícil (pasar de exceptions a arrays de booleanos con los errores). También se crearon los JDIALOGS que contienen los errores que se han cometido al intentar hacer el login o el register.</p> <p>La siguiente tarea fue la de relacionar todas las pantallas entre sí, además de relacionar el piano (ya hecho) con la view de training, pues había diversas pantallas no conectadas, aunque fuesen importantes (como la pantalla de Welcome).</p> <p>La última tarea realizada para el cuarto sprint fue hacer funcional la pantalla de playlist, es decir, que se mostrasen las playlist correctamente según el usuario registrado (si eran playlists</p>

	<p>públicas o privadas), que al hacer clic salieran las canciones que le pertenecían, y que tuvieses la opción de utilizar la pantalla de canciones como un reproductor de música, implementando el botón de play/pausa, el botón de canción aleatoria, el botón de repetición, un botón de acción “secreta” y un slider para poder controlar el volumen de la canción que esté sonando. Además, se relacionó la pantalla de playlist con el web scrapping, haciendo que sea visualizable, a la vez que usable. Por último, se han estado relacionando las pantallas que faltaban a la aplicación, y haciendo una última limpieza del código.</p>
Gabriel Vallarta Angulo	<p>Para este último sprint el nivel de estrés aumentó de manera considerable, aunque a pesar de todo logré implementar desde cero y por completo la funcionalidad del Web Scrapping de la página web mutopianoproject a partir de una larga investigación sobre cómo navegar un código fuente y gestionar las etiquetas de la página para acceder a los campos que desea dentro de la web. Esta lectura la logré mediante el uso de la librería JSoup donde encontré clases extremadamente útiles como Element y sus herencias Document y Elements donde podía gestionar los bloques de información de la web a través de las etiquetas que mencionaba anteriormente. Una vez que descifré el patrón de las tablas publicadas en la url dada, implemente el switch encargado de coger la columna de aquellos campos que necesitaríamos para poder llenar la base de datos de manera automáticamente con un refresh y re-scrap cada x minutos. Una vez implementado esto con una conexión estable y funcional a la web me junté con Marina para definir ajustes finales en la Tabla Song para así poder tener una coherencia de importación entre la base de datos y la información obtenida del url. Probamos directamente con el proyecto y la base abiertas y la descarga y creación de los campos fue un éxito. Una vez que esto estaba solucionado, volví al PlayView para actualizar el feedback necesario para el usuario con respecto a sus recordings de canciones. Implementé esto desde el Controller del PlayView y con la ayuda de un Timer. Ya que esto funcionaba y el switch de botones no tenía errores, hice dos cosas. Primero, incluí el Pop Up de SaveArrangement de Wesley al código para que se ejecutará cada que el usuario decidiera dejar de grabar su canción. Después y por último he redactado mis partes de la memoria asignadas y el he redactado los comentarios para JavaDoc de todas las clases del proyecto, las más claramente explicadas más a detalle y las del resto del equipo con la estructura básica para que sea entendible por aquellos que lean el JavaDoc y lista para que añadan detalles personales que consideren.</p>
Wesley Lucas Mas	<p>En este último sprint como tal no me he centrado en una o dos tareas en concreto, sino en general en ayudar en lo que necesitasen mis compañeros en lo que llega a ser dar los últimos retoques al código y juntar código de apartados entre sí. Y si tuviese que mencionar algo más aparte de ayudar en general con</p>

	<p>todo, diría también que me he encargado de diseñar los últimos pop-ups emergentes a implementar en varias capturas, además de escribir el Java Code de todas aquellas clases relacionadas cercanamente con mi código además de las mías e ir redactando gran parte de la memoria para ir agilizando el proceso y que mis compañeros pudiesen acabar sus respectivas partes del trabajo. En general, mi rol ha sido ayudar a juntar código de varias de las cosas mencionadas por el resto de mis compañeros en este Sprint 4, además de implementar algunos elementos extras que faltaban, y redactar la memoria (a excepción de la Metodología de desarrollo, que eso les corresponde a mis compañeros).</p> <p>NOTA: A la hora de programar código, he ido trabajando en proyectos separados. Uno que era de prueba en local, y el otro el del grupo por Git. Es por esta razón que en lo que respecta al registro de actividades del grupo en Git, yo he hecho menos commits y pushes en comparación al resto de mis compañeros porque la mayor parte del tiempo lo pasaba trabajando en un proyecto aparte en local, en vez del grupal de Git.</p>
Marina Ortega Picazo	Realización de parte de la memoria. Creación de métodos que me han ido indicando mis compañeros que eran necesarios, como por ejemplo coger la siguiente canción en una playlist determinada. Como los números de las ID_Song en una playlist no son consecutivos el método nos retorna un array de songs que tratamos en la PlaylistView. Los métodos como borrar un usuario cuando el usuario así lo pide lo añadí a su sitio junto con el cambio de pantalla correspondiente entre otros.
Pau Pons Clotet	Incorporación de la view en el proyecto final. Para ello tuve que aprender el funcionamiento de bitbucket y de git, y con la ayuda de mis compañeros logré incorporar mi trabajo. Sin embargo, tuve que resolver varios problemas a los que tuve que recurrir a mis compañeros Arnau y Gabriel para que me ayudaran a entender por qué no funcionaba (cuando en mi proyecto sí que funcionaba). Resulta que era un problema con los focus y conseguí resolverlo. Por otra parte, también he empezado a documentarme sobre el uso e implementación. Posteriormente, empecé a mirarme el apartado 2.5 (descargar canciones mediante webscrapping), es decir, desde una input de una url conseguir un fichero midi. Sin embargo, debido a la alta carga de trabajo de las demás asignaturas he tenido que priorizar el trabajo y no he podido disponer del suficiente tiempo para documentarme y entender el funcionamiento para programar el webscraper. He acabado por limpiar mi código y contribuir en la memoria escrita además de ayudar a comentar funciones y métodos.

5. Tiempo de dedicación



En cuanto a una estimación de número de horas dedicado a cada uno de estos apartados, y según nuestra experiencia como grupo, lo interpretamos de la siguiente forma:

Distribución de tiempo dedicado al proyecto de 2º semestre Smart Piano	
Comprensión de las especificaciones	10 horas
Análisis y diseño	30 horas
Codificación	150 horas
Documentación	10 horas

En general, la parte que tomó más protagonismo fue obviamente la codificación, ya que es lo que requería más tiempo de dedicación y sobretodo coordinación entre los compañeros para ponernos de acuerdo con nuestras correspondientes partes de código.

Por otro lado en segundo lugar tenemos el análisis y diseño. Personalmente lo interpretamos mayoritariamente como la elaboración de los mock-ups, base de datos y diagrama de clases, donde nos llevó más tiempo de elaborar porque esos tres elementos conforman las bases del proyecto, y como tal, queríamos asegurarnos de que estábamos haciendo lo correcto.

Y por último, tenemos tanto la comprensión de las especificaciones del proyecto como la redacción de la documentación/memoria, que ambos tomaron el mismo tiempo de

realización, ya que en el caso de la comprensión de las especificaciones, teníamos que mirar bien con lupa esos detalles pequeños que entraban dentro de la funcionalidad general del proyecto, además de ir actualizando la memoria conforme íbamos finalizando los últimos detalles del proyecto a nivel de código, como por ejemplo las capturas de pantalla de las distintas interfaces gráficas de las pantallas programadas, que de vez en cuando hacíamos cambios constantes para alinear mejor los elementos u optimizar mejor las vistas cambiando layouts.

Actualización Reentrega:

Durante la etapa de desarrollo de la pre-reentrega, en general, se corrigieron el siguiente listado de apartados:

- Configuration JSON
No utilizábamos el timer del webscrapper viniendo del config.json, sino de otra clase que contenía ese parámetro. Ahora se ha corregido y se extrae el valor del timer directamente del config.json como debería de ser y como ya está explicitado en el enunciado del proyecto.
- Software Quality: Encapsulation & Abstraction
Hemos corregido un problema de encapsulación donde varias clases 'Support' tenían atributos asignados en público que luego dentro del layer de Business, la clase PlayModel utilizaba, provocando así pues el no haber respetado la encapsulación de la programación orientada a objetos. Además de varias clases más como la del WebScrapper también tenía un atributo en público por ejemplo, además de tener atributos package-visible en otras clases.
- Software Quality: Model View Controller & Layered Architecture
Corregidos varios problemas entre los cuales el que más destaca es el no uso de los controllers para trabajar a modo de puente entre las clases de las vistas y las de Business. Hemos ido clase por clase y reorganizado el código de tal forma que ahora se pasarán por el controller respectivo.
- Documentació: JavaDoc
Hemos añadido comentarios que faltaban por añadir, además de añadir de más en algunas clases donde solo había una línea de comentario JavaDoc.
- User Sign-up
Corregido un problema de registro donde cada vez que se hace un registro de un usuario ya existente como uno nuevo, no se detectaba si dicho usuario existía o no ya en la base de datos.
Ahora la comprobación de ese error la realiza de forma correcta.
- User Login

Añadida la funcionalidad de poder hacer login con el mail. Antes lo hacíamos solo con el nombre del usuario. Pero lo hemos corregido y ahora se permite hacer log in también con el mail de dicho usuario

- Playlist management
Añadidas las funcionalidades de crear y eliminar playlists, además de eliminar y mover pistas musicales de una playlist a otra.
- Songs management
Añadida la funcionalidad de eliminar canciones en las playlists.
- Top popular songs
Apartado implementado para la reentrega, donde antes lo único que teníamos hecho era la view con imágenes estáticas y poco más. Para esta ocasión, hemos hecho las conexiones que tocaban con la base de datos, actualizar los datos dentro de la base de datos cada vez que se reproduce una canción para tenerla en cuenta a nivel de popularidad, además de acceder a la base de datos para poder extraer las pistas musicales más populares para colgar en el ranking.

6. Conclusiones

A lo largo de este 2º y último semestre de Programación Orientada a Objetos hemos tratado con threads, Model View Controller/Layered Architecture y por último el apartado gráfico/librería gráfica de Java, ‘Java Swing’, que combinados junto con los conceptos aprendidos durante el 1er semestre, nos permiten realizar proyectos de gran escala como el que acabamos de realizar durante este semestre, el Smart Piano.

Una de las muchas cosas que se ha aprendido durante el desarrollo de este proyecto, es que por un lado requiere una revisión exhaustiva y trabajo constante de codificación para poder sacar adelante este tipo de proyectos de gran escala, además de la colaboración grupal y sobretodo comunicación eficaz entre los miembros del equipo.

Aparte de esta ‘lección de vida’, y entrando más en el proyecto en sí, comenzar diciendo que trabajar con una arquitectura de Model View Controller era difícil al principio porque estábamos acostumbrados a trabajar y ver ejemplos de Layered Architecture, que al final del día, una vez indagamos dentro del proyecto y comenzamos a trabajar, era un poco lioso, sobre todo a la hora de hacer el diagrama de clases. Pero una vez hemos aprendido su jerarquía y estructura ordenada que ofrece, poco a poco nos fuimos familiarizando con este modelo de arquitectura y descubrimos sus ventajas a la hora de trabajar con él, ya sea por su flexibilidad, rapidez de desarrollo, la habilidad de tocar diferentes partes del código y ver que no le afecta al resto de miembros del grupo que desarrollan su propia parte, etc...

Por otro lado, en cuanto a la base de datos, nos costó hacer el enlace entre esta y el proyecto en Git. Y después también estábamos indecisos a la hora de escoger qué sistema

gestor de base de datos con el que trabajar y en general cómo gestionar la base de datos dentro de un proyecto de código realizado con un lenguaje orientado a objetos como Java. Pero con el tiempo, y al igual que se comentó previamente en el Model View Architecture, fuimos poco a poco aprendiendo la lógica del enlace entre ambos mundos de las bases de datos y la orientación a objetos, y pudimos sacarlo adelante lo mejor que pudimos hacerlo.

Y por último, pero no el menos importante, la librería de desarrollo de interfaces gráficas, Java Swing. Una cosa que cabe destacar sobre esta parte en concreto es que pusimos un enfoque muy grande sobre el diseño de cada una de las pantallas que componen nuestro proyecto. Cada una de ellas contiene una gran variedad de layouts, además de composiciones formadas de unas con otras que nos permitía no solo expandir los horizontes con los diseños planteados desde el principio, sino además poder expandir/profundizar también nuestro conocimiento y habilidad de trabajo a la hora de lidiar con estos a nivel de código, como por ejemplo los diferentes métodos que ofrecen cada uno de los componentes disponibles que los hacen distintos al resto para así poder sacarles el máximo de provecho para darle más variedad a cada una de las pantallas. Esto, junto con los Controllers correspondientes de cada View, habían momentos como por ejemplo las dos pantallas de Playlists que nos lo ponían muy difícil, y podíamos haber optado por ejemplo por utilizar para el directorio de listas de reproducción y pistas musicales por ejemplo un JTree pegado a un panel y unos cuantos Labels, e íbamos aguantando con eso. Pero decidimos que no, y dimos el paso extra en incorporar diferentes componentes (JButtons, JSlider, JCheckboxes, JDiaogs, etc...) organizados en diferentes layouts (BorderLayout, GridLayout, BoxLayout, CardLayout, etc...) y así a lo largo del proyecto. Que de hecho, como anotación extra, otra de las pantallas que nos fue difícil de diseñar fue sin duda la del Play que contenía el piano. Directamente pusimos como base del instrumento un Layered Pane combinado con layouts como capas secundarias, y de ahí al Controller, también era complejo de lidiar por el tema de los sonidos de las notas, el tratamiento de ficheros MIDIs a través del web scraping, etc...

En general, aprendimos que la librería de Java Swing es una librería que ofrece mucho potencial, pero que a la larga acaba siendo un poco desfavorable a la hora de trabajar con esta por la cantidad de limitaciones que a veces nos ponía como obstáculos.

Dicho todo esto, este proyecto nos ha servido no solo para aprender conceptos nuevos relacionados con la programación web en sí, sino también nos ha servido como introducción a lo que nos viene en 3er curso de ingeniería, Proyectos Web I y Proyectos Web II, donde según hemos oído, se profundizan aún más los conceptos dados durante este 2º semestre de Programación Orientada a Objetos, solo que con la diferencia de que los trabajamos en otro lenguaje y entorno de trabajo que nos ofrece más flexibilidad y agilidad que la librería de Java Swing.

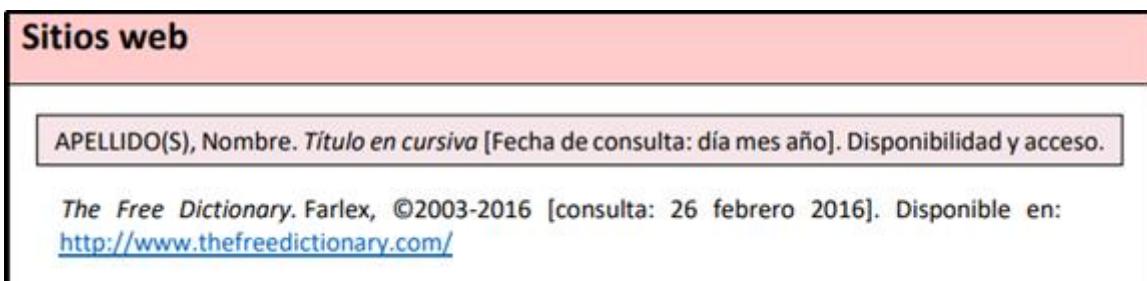
Actualización Reentrega:

Esta reentrega del proyecto nos ha servido no solo para añadir más cosas que nos faltaban previamente por añadir, además de corregir errores de código como conceptuales, sino que también nos ha servido para darnos cuenta de que aspectos como por ejemplo el no

respetar los conceptos de encapsulamiento, abstracción o el MVC/Layered Architecture, eran el corazón del proyecto (por decirlo de alguna forma) y lo vital que era importante tenerlos en cuenta, ya que entonces no estamos siguiendo los principios básicos de la orientación de objetos además a nivel de diseño de una aplicación. Estas semanas nos han servido para reevaluar el proyecto y reforzar nuestros conocimientos sobre la materia, de tal forma que de cara al curso siguiente en 3º de multimedia, podamos mantener estos conceptos en mente frescos y sin volver a cometer errores conceptuales como los vistos y comentados en la rúbrica de la entrega de mayo.

7. Bibliografía consultada

El formato de citación de la bibliografía según la norma ISO 690:2010 la hemos extraído de la siguiente página web como referencia. Específicamente página 4, apartado 'Sitios web'. En concreto seguimos el siguiente formato:



Microsoft. Prueba de concepto de la conexión a SQL mediante Java[10/04/2021] Disponible en:

<https://docs.microsoft.com/es-es/sql/connect/jdbc/step-3-proof-of-concept-connecting-to-sql-using-java?view=sql-server-ver15>

Inforux. Java: Practicando con BorderLayout [10/04/2021] Disponible en:

<https://inforux.wordpress.com/2009/01/20/java-practicando-con-borderlayout/#:~:text=BorderLayout%2C%20es%20un%20layout%20que,abajo%2C%20izquierda%20o%20a%20la%20derecha.>

Cuidiang. Ejemplo de BorderLayout, BoxLayout y FlowLayout [11/04/2021] Disponible en:

<http://www.chuidiang.org/java/layout/BorderLayout.php>

Oracle. Introduction to Event Listeners [14/04/2021] Disponible en:

<https://docs.oracle.com/javase/tutorial/uiswing/events/intro.html>

w3schools. Java Hashmap [20/04/2021] Disponible en:

https://www.w3schools.com/java/java_hashmap.asp

Oracle. How to use Layered Panes [20/04/2021] Disponible en:

<https://docs.oracle.com/javase/tutorial/uiswing/components/layeredpane.html>

Oracle. JLayeredPane [20/04/2021] Disponible en:

<https://docs.oracle.com/javase/7/docs/api/javax/swing/JLayeredPane.html>

Disco duro de roer. Eventos y Listeners en Java [20/04/2021] Disponible en:

<https://www.discoduroderoer.es/eventos-y-listeners-en-java/>

Oracle. How to use BoxLayout [20/04/2021] Disponible en:

<https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html>

Chuidiang. BoxLayout[20/04/2021] Disponible en:

<http://chuwiki.chuidiang.org/index.php?title=BoxLayout>

Oracle. CardLayout [23/04/2021] Disponible en:

<https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html>

Javatpoint. CardLayout [23/04/2021] Disponible en:

<https://www.javatpoint.com/CardLayout>

Inforux. Practicando con CardLayout [23/04/2021] Disponible en:

<https://inforux.wordpress.com/2009/03/19/java-practicando-con-cardlayout/>

MyJavaZone. Contenido CardLayout [23/04/2021] Disponible en:

<http://www.myjavazone.com/2011/02/manejadores-de-contenido-cardlayout.html>

Jsoup. jsoup: HTML Parser [14/05/2021] Disponible en:

<https://jsoup.org/>

Adictos al trabajo. Implementando un crawler sencillo con jsoup [15/05/2021] Disponible en:

<https://www.adictosaltrabajo.com/2019/05/14/implementando-un-crawler-sencillo-con-jsoup/>

Softtek. jsoup [15/05/2021] Disponible en:

<https://blog.softtek.com/es/jsoup>

StackOverflow. Java Append() [18/05/2021] Disponible en:

<https://stackoverflow.com/questions/23353926/java-append-method-string-vs-stringbuilder/23353940#23353940>

Javatpoint. Java Date to String [18/05/2021] Disponible en:

<https://www.javatpoint.com/java-date-to-string>