Universitat Oberta de Catalunya

uoc.edu

MusicNet

Jonathan Sierra Hernández
Grado de ingeniería informática
Desarrollo web

Gregorio Robles Martínez
Santi Caballe Llobet

06/2020

FICHA DEL TREBAJO FINAL

Título del trabajo:	*MusicNet*
Nombre del autor:	*Jonathan Sierra Hernández*
Nombre del consultor:	*Gregorio Robles Martínez*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega (mm/aaaa):	*06/2020*
Titulación o programa:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo web*
Idioma del trabajo:	*Castellano*
Palabras clave	*Música, formación online, e-learning*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de la aplicación, metodología, resultados y conclusiones del Trabajo.*

El gran desconocimiento al entrar en el mundo de la música hace que muchas personas se equivoquen, no encuentren el material que necesiten en ese momento e incluso lleguen a la frustración y fracaso por no tener los métodos pedagógicos necesarios.

La finalidad de este proyecto nace de este contexto para solucionar dichos problemas y poder mejor los procesos musicales didácticos y llevar al usuario hacia el éxito musical sin frustraciones.

Se ha utilizado una metodología de proyecto en cascada definida por las fases de Inicialización, Planificación, Ejecución, Seguimiento y control y Cierre.

Los resultados obtenidos han sido exitosos, hemos podido observar que para el alumno es mucho más fácil estudiar de forma correcta llevándolo al éxito asegurado.

Por lo tanto, queda comprobado que la problemática detectada en las primeras fases del proyecto se puedo solucionar con nuestro proyecto.

Abstract (in English, 250 words or less):

The great ignorance when entering the world of music makes many people

make mistakes, don’t find the materials what they need at that time and even reach frustation and failure in order to not having the necessary pedagogical methods.

The purpose of this project is born from this context to solve these problems and better be able to the didactic musical processes and lead the user towards musical success without frustrations.

A cascade method project methodology defined by the Initialization, Planning, Execution, Follow-up and Control and Closing phases has been used.

The results obtained have been successful, we have been able to observe that for the student it is much easier to study correctly, leading to assured success.

Therefore, it is proven that the problems detected in the early phases of the project can be solved with our project.

Índice

1. Introducción

1.1 Contexto y justificación del trabajo

MusicNet nace de la falta de material didáctico centralizado y organizado a través de Internet. Por ejemplo, en plataformas como Youtube o Vimeo encontramos miles de videos relacionados con la música. De todos ellos, solo unos centenares son dedicados a la pedagogía y, además, no presentan ninguna conexión u organización lo que hace que el estudiante tenga una muy alta probabilidad de aprender de una forma incorrecta llevándolo al fracaso. Por ello, en MusicNet el estudiante podrá buscar cursos de alta calidad que ofrecen varias herramientas y medios para ayudar al estudiante a llevar a cabo su objetivo de forma exitosa. A través de la plataforma online, se podrá acceder des de cualquier parte del mundo a vídeo aulas acompañadas de herramientas musicales y documentación.

1.2 Objetivos del trabajo

El objetivo principal es la creación de una plataforma de cursos musicales organizados según estilo, instrumento y dificultad. A partir de este, derivan los siguientes:

- Tienda online de cursos.

- Cursos basados en videos transmitidos en Streaming.

- Cursos que pueden o no ofrecer tablaturas y partituras en formato pdf.

- Cursos que pueden o no ofrecer herramientas rítmicas como metrónomos, bases rítmicas o bases de acompañamiento.

- Cursos que pueden o no ofrecer interacción con el mismo profesor del curso o con otros alumnos.

- Profesores y alumnos tendrán un perfil de usuario. Los profesores podrán dar aulas individuales o en grupos y los alumnos podrán acceder a ellas.

- Plataforma diseñada para ser usada des de cualquier dispositivo (ordenador, Tablet o móvil).

- Plataforma diseñada en distintos idiomas.

- Cursos que pueden o no estar en más de un idioma.

1.3 Enfoque y método seguido

En el mercado podemos encontrar bastantes plataformas e-learning open source por lo que una posible estrategia seria la elección de una de estas

plataformas y modificarla o configurarla para conseguir nuestro objetivo. Algunos ejemplos de estas plataformas son Moodel, Chamilo y LMS en WordPress. Cada una de ellas presenta diferentes problemas:

- Moodel: la usabilidad no es la esperada.
- Chamilo: el soporte de la comunidad no es el suficiente.
- LMS en WordPress: No ofrece actividades didácticas colaborativas.

Entre todos los productos e-learning, no hay ninguna que se ajuste realmente a todas nuestras necesidades así que es necesario el desarrollo de un nuevo producto capaz de cumplir todos los requisitos. Para ello, se utiliza una metodología en cascada con las fases de inicialización, planificación, ejecución, seguimiento y control y cierre.

1.4 Planificación del Trabajo

Planificación temporal

Nombre de la tarea	Duración	Inicio	Fin
Planificación	16 días	20/02/2020	06/03/2020
Instalación y configuración de herramientas	10 días	07/03/2020	17/03/2020
Especificación	16 días	18/03/2020	03/04/2020
Diseño	16 días	04/04/2020	20/04/2020
Implementación y testeo	41 días	21/04/2020	01/06/2020

1.5 Breve sumario de los productos obtenidos

Se obtiene un único producto dividido en diferentes componentes: FrontEnd, BackEnd y BBDD. El FrontEnd y el Backend serán desplegados en un servidor y la base de datos instalada y configurada en un motor de base de datos con. Todos ellos correctamente integrados forman el producto final.

1.6 Breve descripción de los otros capítulos de la memoria

En primer lugar, se presentará el entorno utilizado en MusicNet mostrando las tecnologías y patrones de arquitectura. Seguidamente se hará una evaluación de los riesgos para después mostrar las diferentes tareas de trabajo (planificación, instalación y configuración de herramientas, especificación, diseño e implementación). Después de estas secciones se presentan las conclusiones del proyecto y, finalmente, se muestra un glosario con los términos más usados, así como una bibliografía con las URLs utilizadas en alguna fase.

2. Entorno del proyecto

2.1 Tecnologías y patrones que se van a utilizar

- Java JDK [1]

Ya que el backend del proyecto está basado en el lenguaje de programación Java, Java Development Kit (JDK) es totalmente necesario. El JDK es una implementación de una de las plataformas conocidas como Java Platform Standard Edition, Java Platform Enterprise Edition o Java Platform Micro Edition lanzada por Oracle Corporation en forma de un producto binario dirigido a desarrolladores de Java en Solaris, Linux, macOS o Windows. Este también incluye una JVM privada y algunos otros recursos para finalizar el desarrollo de una aplicación Java. Desde la creación de la plataforma Java, ha sido el kit de desarrollo de software (SDK) más utilizado. El 17 de noviembre de 2006, Sun anunció que lanzaría el JDK bajo la Licencia pública general (GPL) de GNU, convirtiéndolo así en software libre. Esto sucedió el 8 de mayo de 2007, cuando Sun contribuyó con el código fuente al OpenJDK.

El JDK también viene con Java Runtime Environment (está separado del JDK y tiene contenidos adicionales). Consiste en una máquina virtual Java con las bibliotecas de clases del entorno de producción y con bibliotecas adicionales que solo son útiles para los desarrolladores (como las bibliotecas de internacionalización y las bibliotecas IDL).

- Eclipse [2]

Un IDE (Integrated Development Environment) nos facilita mucho el desarrollo de aplicaciones a partir de un tamaño medio. Habitualmente permite, entre otras cosas: gestionar proyectos en las que participan varias (o muchas) clases, navegar fácilmente por las clases Java y sus métodos, depurar errores, etc (todo ello puede resultar en una reducción importante del tiempo de desarrollo). Si no se utiliza ningún IDE de desarrollo el tiempo usado para la implementación del código fuente será demasiado alto como para desarrollar el proyecto dentro de las datas establecidas.

Hay bastantes IDEs para Java. De todos ellos, hay unos cuantos que son gratuitos; y de este subconjunto hay dos que son de código abierto: Netbeans (www.netbeans.org) y Eclipse (www.eclipse.org). Tras evaluar estos últimos IDEs, se cree que Eclipse resulta mucho más confortable y cómodo de usar; y proporciona todas las prestaciones que serían exigibles a un buen IDE. Eclipse está desarrollado en su totalidad en Java. Por lo tanto, está accesible para cualquier plataforma que soporte Java.

- Github [3]

Es obvio que la necesidad de un repositorio y control de versiones es totalmente necesario para el desarrollo de un proyecto de esta envergadura. Por lo tanto, se decide que se utilizara la plataforma GitHub ya que el software

que sustenta el sitio es el propio Git y, además, añade las funcionalidades necesarias.

Se puede acceder y manipular proyectos en GitHub utilizando la interfaz de línea de comandos estándar de Git. GitHub también permite a los usuarios registrados y no registrados navegar por los distintos repositorios públicos de la plataforma. Además, se han creado múltiples clientes de escritorio y complementos de Git.

También proporciona funciones similares a las redes sociales, como feeds, seguidores, wikis (utilizando el software wiki llamado Gollum) y un gráfico de red social para mostrar cómo los desarrolladores trabajan en sus versiones ("forks") de un repositorio.

- La aplicación será desarrollada bajo el patrón de diseño MVC.

Entre todos los patrones de diseño el que más se ajusta para los requisitos de este proyecto es el patrón de diseño Modelo-vista-controlador (MVC) el cual es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación, de su representación y del módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

El modelo es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

El controlador responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos). Se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

La vista presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario).

- Spring [4]

Como se ha comentado anteriormente, Java es el lenguaje escogido para desarrollar el BackEnd de la aplicación. Dentro de Java existen diferentes frameworks que nos ayudan a encarar la implementación y desarrollo, se debe escoger un framework que aporte soluciones para nuestros retos técnicos.

Spring es conocido por ofrecer funcionalidades para crear aplicaciones de negocio complejas y por ofrecer los modelos de programación que históricamente han sido dominantes en la industria. Además, introduce funcionalidades que eran poco familiares en prácticas de programación que hoy en día son básicas, incluso más allá de la Plataforma Java. Todas estas características están basadas en las mejores prácticas y estándares de la industria, haciéndolo válido por muchos dominios Java.

Spring puede ser considerado como una colección de entornos más pequeños, o entornos dentro de entornos. La mayoría de estos están diseñados para trabajar independientemente unos de otros, aunque se supone que juntos, mejoran las funcionalidades. Estos módulos están divididos en bloques de construcción típicos de aplicaciones complejas:

- Contenedor de Inversión de Control: configuración de componentes de aplicación y gestión del ciclo de vida de objetos Java.
- Entorno de Programación orientada a aspectos (AOP): habilita la implementación de rutinas transversales.
- Entorno de acceso a los datos: proporciona el acceso a bases de datos relacionales, sobre la plataforma Java utilizando JDBC y herramientas de mapeo objeto - relacional (JPA, Hibernate, etc).
- Entorno de gestión de transacciones: armonización de diferentes APIs de gestión de transacciones.
- Entorno modelo-vista-controlador: basado en HTTP y Servlet permiten usar el patrón MVC.
- Entorno de acceso remoto: importación y exportación de objetos java usando RPC sobre redes informáticas que soportan protocolos basados en HTTP, tales como RMI, CORBA, y servicios web (REST).
- Entorno de autenticación y personalización: orquestación configurativa de autenticación y procesos de autorización que soportan muchos estándares de la industria.
- Entorno de mensajería: registro configurable de objetos que reciben mensajes desde colas vía JMS, mejora el envío de mensajes sobre APIs y soporte para el protocolo AMQP (RabbitMQ, Kafka Broker ...).
- Entorno de testeo: soporta clases para la creación de unidades de testeo y pruebas de integración.

- Hibernate [5]

Para que Spring pueda interactuar con la base de datos se necesita una herramienta de mapeo objeto-relacional (ORM) facilitando el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Hibernate es un software libre que se utilizará para MusicNet, distribuido bajo los términos de la licencia GNU LGPL

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto, se permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO, convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

- Boostrap [6]

Para que el desarrollo de nuestra aplicación llegue a un producto real en un corto periodo de tiempo se necesite el uso de tecnologías que permitan hacer el desarrollo lo antes posible. Boostrap es un framework para su uso en la parte FrontEnd para proyectos como el que nos ocupa.

Además, nuestra aplicación será accedida des de todo tipo de dispositivos los cuales tienen diferentes tamaños. Boostrap ofrece diferentes mecanismos para poder crear sitios web responsive o adaptables lo que significa que cuando nuestra aplicación sea accedida des de un móvil, ésta será adaptada a la pantalla del dispositivo; si la aplicación es accedida des de una tablet, ésta será adaptada a la pantalla del dispositivo y así para cualquier aparato.

Este framework hace uso de las tecnologías HTML, CSS y JavaScript donde el desarrollador tan solo tendrá que descargarse Boostrap, instalarlo en su proyecto y comenzar a hacer uso de éste. Se podrá crear proyectos profesionales y adaptables de rápidamente.

- HTML5 [7]

HTML 5 (HyperText Markup Language, versión 5) es la quinta revisión del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML 5 que deberá servirse con sintaxis XML (application/xhtml+xml). Esta es la primera

vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014.

Nuestras necesidades se ajustan a la variante más clásica o HTML5. Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se recomienda al usuario común actualizar su navegador a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML 5.

- CCS3 [8]

CSS (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Sera usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML5.

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes. Con el uso de este se mejorará la accesibilidad del documento, se tendrá más flexibilidad y control en la especificación de características presentacionales, se permitirá que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css y se reducirá la complejidad y la repetición de código en la estructura del documento.

La separación del formato y el contenido hace posible presentar el mismo documento marcado en diferentes estilos para diferentes métodos de renderizado, como en pantalla, en impresión, en voz (mediante un navegador de voz o un lector de pantalla), y dispositivos táctiles basados en el sistema Braille. También se puede mostrar una página web de manera diferente dependiendo del tamaño de la pantalla o tipo de dispositivo.

La especificación CSS describe un esquema prioritario para determinar qué reglas de estilo se aplican si más de una regla coincide para un elemento en particular. Estas reglas son aplicadas con un sistema llamado de cascada, de modo que las prioridades son calculadas y asignadas a las reglas, así que los resultados son predecibles.

- Javascript [9]

El FrontEnd utilizará JavaScript (o JS) como lenguaje de programación. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Sera utilizado del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a

funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX.

- REST [10]

Un servicio web (en inglés, web service o web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en diferentes lenguajes de programación, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Para el caso que nos ocupa, el servicio web que más se adapta a nuestras necesidades son los servicios REST ya que se ganara en adaptabilidad, interoperabilidad, escalabilidad y mantenimiento.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en nuestro proyecto será usado en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

- Thymeleaf [11]

La creación de un sitio web dinámico tiene que acceder a datos que varían con el tiempo y, según estos, el contenido mostrado al usuario será de una forma o de otra. Los usuarios podrán ser los propios generadores de contenido, otras veces será el mismo administrador… Lo que realmente importa es que dentro del diseño MVC, las vistas generaran contenido dinámico.

La plataforma necesita una biblioteca Java que implemente un motor de plantilla HTML5 (también extensible a otros formatos). Thymeleaf es la solución más trabajada y conocida a nuestros requisitos. Éste se acopla muy bien para trabajar en la capa vista del MVC de aplicaciones web, pero puede procesar cualquier archivo XML, incluso en entornos desconectados.

El objetivo principal de Thymeleaf es permitir la creación de plantillas de una manera elegante y un código bien formateado. Sus dialectos *Standard* y *SpringStandard* permiten crear potentes plantillas naturales que se pueden visualizar correctamente en los navegadores de Internet, por lo que también funcionan como prototipos estáticos.

- Postgresql [12]

Durante las siguientes fases de especificación y diseño, se comprobará que los datos deberán ser guardados en un sistema relacional. Como tal, se escogerá un motor de sistema de baso de datos libre, es decir, que no hayamos de pagar

una licencia. De los diferentes que hay en el mercado, escogemos PostgreSQL por las características expuestas aquí abajo.

PostgreSQL es una base de datos relacional, distribuida bajo licencia BSD y con su código fuente disponible libremente. Es el motor de bases de datos de código abierto más potente del momento y en sus últimas versiones empieza a no tener que envidiar nada a otras bases de datos comerciales.

Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. En los últimos años se han concentrado mucho en la velocidad de proceso y en características demandadas en el mundo empresarial

- Youtube API [13]

Un sistema de almacenamiento y gestión de videos como el que se implementará tiene un gran coste tanto en términos de desarrollo como en términos monetarios. Ofrecer un servicio de tales características a decenas, centenas o tal vez millares de usuarios tendrían un gran coste financiero sin tener asegurado un retorno inmediato.

Existen servicios web que nos ayudan a poder realizar un sistema de video potente sin necesidad de gastar absolutamente nada, dos de estos servicios que encontramos en el mercado son Youtube y Vimeo. Del primero Google tiene una documentación muy extensa, así como librerías en varios lenguajes de programación (entre ellos Java). Por otro lado, cada servicio consumido del api tiene un coste y cada usuario tiene una cuota asignada. Vimeo también tiene una cuota parecida a la de Google aunque solo ofrece librerías ya desarrolladas en PHP, Python y JavaScript. Por ello se elige Youtube para este proyecto, por la facilidad de integración con Spring.

- Zoom [14]

Durante la definición del proyecto queda claro que será necesario una funcionalidad la cual permita la comunicación entre diferentes usuarios (de profesor a alumno o de profesor a varios alumnos en tiempo real). De todas las tecnologías mencionadas hasta este punto, no se ajusta exactamente a nuestras necesidades y es por ello que se decide utilizar Zoom.

En el mercado se puede encontrar diferentes plataformas de video conferencias (Google Hangouts, Jitsi, Skype, ooVoo, Microsoft Teams, etc). Algunas de ellas no permiten la integración de en otras aplicaciones, otras si lo permiten. De las que lo permiten, Zoom ofrece un sistema robusto, así como una API la cual puede ser usada por otras aplicaciones externas. La escalabilidad de Zoom permite que sea fácilmente integrada con nuestra aplicación.

- Stripe Java [15]

Al usar el patrón MVC basado en Rest, la plataforma de cursos va a ser un proyecto con una alta escalabilidad, de bajo acoplamiento y de una integración sencilla con otros proyectos. Ya que se tiene dicha arquitectura, la forma más fácil y potente de implementar las funcionalidades para permitir la venta y compra dentro de MusicNet es usando otros proyectos que nos ofrezcan estas posibilidades. Para el caso que nos ocupa se elige Stripe Java. Stripe Java es una librería que permite la integración de las funcionalidades de venta y compra dentro de nuestro sitio web de una forma segura y autentificada.

3. Evaluación

3.1 Evaluación de riesgos

El mayor riesgo es la planificación de un proyecto donde los requisitos sean demasiado complejos. Desconozco el alcance real de varios objetivos como, por ejemplo, que tan difícil debe ser crear una interfaz de la aplicación para dispositivos móviles o cuanto tiempo me llevara aprender Boostrap, Spring o Hibérnate.

Es posible que algunos de los objetivos no puedan ser implementados según las tecnologías usadas. Por ejemplo, ¿es posible crear un sistema de streaming con las tecnologías arriba mencionadas?

4. Tareas

4.1 Planificación

Ver apartado 1.4.

4.2 Instalación y configuración de herramientas

Ver anexo.

4.3 Especificación

Enunciado del proyecto

El proyecto “MusicNet” consiste en desarrollar un espacio digital que permite a cualquier persona de cualquier lugar del mundo acceder a diferentes cursos musicales de diferentes niveles i de alta calidad por un módico precio y, el sitio web será accesible des de cualquier dispositivo (ordenador, teléfono, tableta, etc).

En “MusicNet” tendremos registrados tanto alumnos como profesores. De los alumnos nos interesa su nombre, primer apellido, segundo apellido que es opcional, dirección de correo electrónico, dirección postal que es opcional, número teléfono que es opcional y la contraseña. Así, los alumnos tendrán una contraseña, que se utilizara conjuntamente con la dirección de correo electrónico para acceder al sistema.

De los profesores nos interesa exactamente la misma información. Además, sabremos cuál es su estilo o estilos musicales, cuantos años tiene de experiencia en cada estilo y la cuenta bancaria donde se depositará el dinero ganado.

La plataforma tendrá cursos musicales. Para cada curso sabremos su descripción, a que estilo pertenece, a que instrumento está dedicado, el grado de dificultad, la aproximación del total de horas que el alumno ha de dedicar y si el profesor y los demás alumnos ofrecen contacto vía e-mail.

Cada curso tendrá material didáctico y herramientas de estudio. El material didáctico será básicamente uno o varios videos multimedia transmitidos via Streaming, un posible texto con explicaciones y podrán ofrecer acceso a las tablaturas y partituras. Las herramientas de estudio serán metrónomos con diferentes velocidades, pistas de acompañamiento y pistas de compás.

Los profesores podrán crear tantos cursos como deseen y quedarán disponibles para ser comprados. Ellos decidirán si el curso estará disponible en más de un lenguaje (ellos son los creadores de contenido). Los alumnos podrán comprar uno o más cursos y podrán acceder a ellos de forma indeterminada.

Los profesores podrán acceder a un sistema de videoconferencia des del cual podrán ofrecer clases en tiempo real. Los alumnos que lo deseen podrán realizar el pago correspondiente y acceder a la videoconferencia donde podrán acompañar la clase realizando comentarios o preguntas a través de un sistema de mensajería.

La plataforma está diseñada para ser accesible des cualquier dispositivo por lo que debe estar completamente disponible para teléfonos móviles, tabletas y ordenadores. Además, toda la plataforma está disponible en diferentes idiomas.

En definitiva, el proyecto “MusicNet”, según los requisitos anteriores, se desarrollará con 4 componentes principales:

- Administración del sistema: Autentificación y operaciones que puede hacer el administrador del sistema como la gestión de herramientas o usuarios.
- Cursos: Todo el conjunto de operaciones que tanto alumnos como profesores pueden hacer en relación a los cursos.
- Aulas: Todo el conjunto de operaciones que tanto alumnos como profesores pueden hacer en relación a las aulas en tiempo real.

- Perfil: Conjunto de operaciones asociadas al perfil de cada usuario, entre otros, la modificación de la información personal.

4.4 Diseño de la base de datos

1) Recogida y análisis de requisitos

El enunciado del proyecto del apartado anterior, es el resultado de la toma y análisis de requisitos el cual usaremos tanto para el diseño de la base de datos como para el diseño de la aplicación.

2) Diseño conceptual [16]

Se crea un esquema conceptual de alto nivel e independiente a partir de los requisitos, las especificaciones y las restricciones.

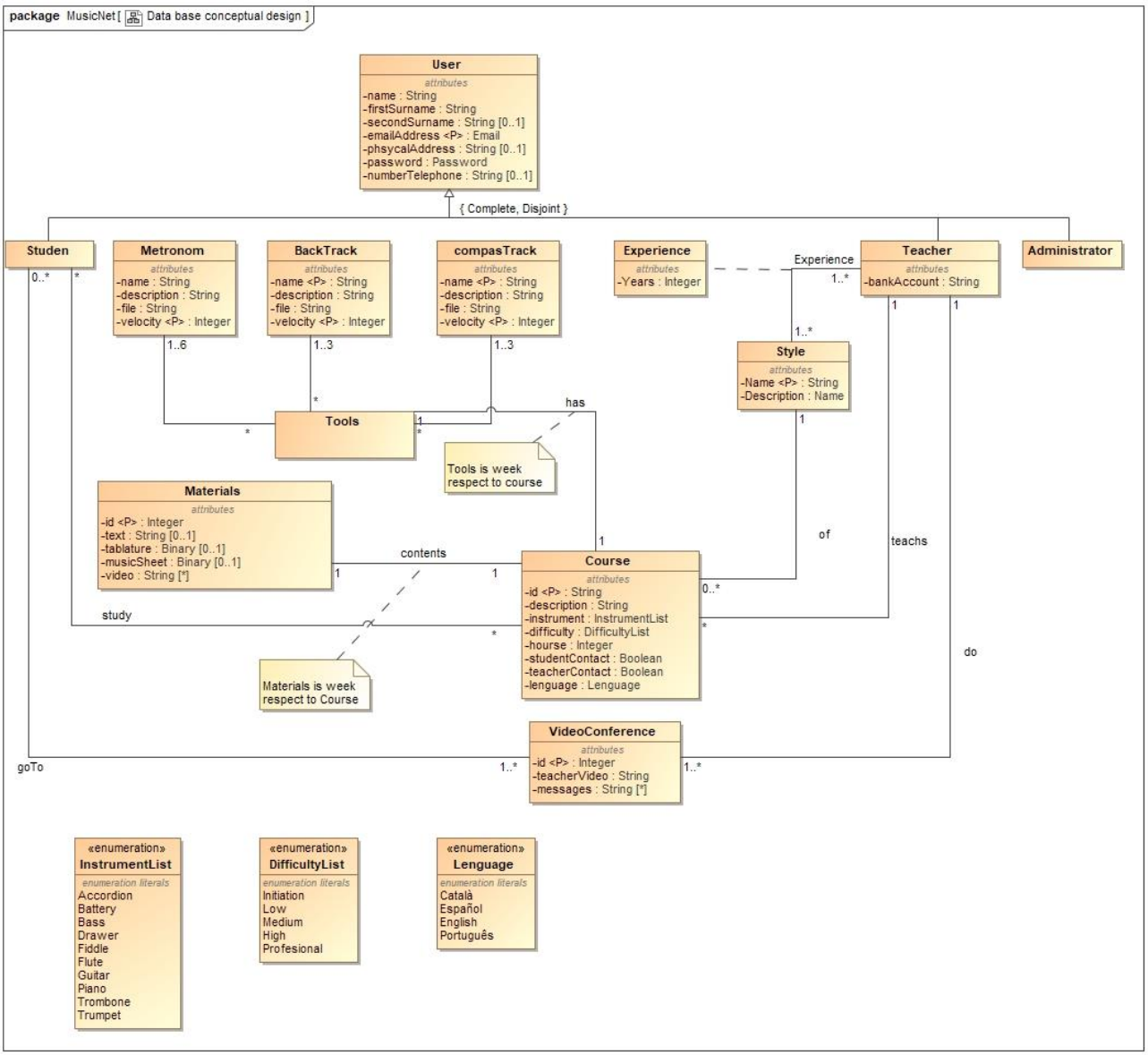


Figura 1: Diseño conceptual de la base de datos

3) Diseño lógico [17]

Se obtiene un esquema lógico a partir del esquema conceptual del punto 2. El esquema lógico dependerá de una base de datos relacional, pero será independiente a la implementación concreta del sistema de gestión de la base de datos (SGBD).

*User (**name**, **firstName**, secondName, emailAddress, physicalAddress, **password**, numberTelephone)*

*Student (**name**, **firstName**, secondName, emailAddress, physicalAddress, **password**, numberTelephone)*
{ emailAddress } is foreign key to User

*Teacher (**name**, **firstName**, secondName, emailAddress, physicalAddress, **password**, numberTelephone, **bankAccount**)*
{ emailAddress } is foreign key to User

*Administrator (**name**, **firstName**, secondName, emailAddress, physicalAddress, **password**, numberTelephone)*
{ emailAddress } is foreign key to User

*VideoConference (teacherVideo, **emailAddressTeacher**)*
{ emailAddressTeacher } is foreign key to Teacher

Messages (idVideoConference, message)
{ idVideoConference } is foreign key to VideoConference

GoTo (emailAddressStudent, idVideoConference)
{ emailAddressStudent } is foreign key to Student
{ idVideoConference } is foreign key to VideoConference

Study (emailAddressStudent, courseId)
{ emailAddressStudent } is foreign key to Student
{ courseId } is foreign key to Course

*Course (id, **description**, **instrumentName**, **difficulty**, **hours**, **studentContact**, **teacherContact**, **lenguage**, **emailAddressTeacher**, **styleName**)*
{ emailAddressTeacher } is foreign key to Teacher
{ styleName } is foreign key to Style
{ instrumentName } is foreign key to Instrument
{ dificulty } is foreign key to Difficulty
{ lenguage } is foreign key to Lenguage

*Style (name, **description**)*

*Instrument (name, **description**)*

Difficulty (name)

Lenguage(name)

*Experience (styleName, emailAddressTeacher, **years**)*
{ styleName } is foreign key to Style
{ emailAddressTeacher } is foreign key to Teacher

Materials (id, courseId, text, tablature, musicSheet)
{ courseID } is foreign key to Course

VideosMaterials (idMaterials, video)
{ idMaterials} is foreign key to Materials

*Tools (icourseID, metronom1, **metronom2**, **metronom3**, **metronom4**, **metronom5**, **metronom6**, backTrack1Name, backTrack1Velocity, **backTrack2Name**, **backTrack2Velocity**, **backTrack3Name**, **backTrack3Velocity**, compasTrack1Name, compasTrack1Velocity, **compasTrack2Name**, **compasTrack2Velocity**, **compasTrack3Name**, **compasTrack3Velocity**)*

{ courseID } is foreign key to Course
{ metronom1 } is foreign key to Metronom
{ metronom2 } is foreign key to Metronom
{ metronom3 } is foreign key to Metronom
{ metronom4 } is foreign key to Metronom
{ metronom5 } is foreign key to Metronom
{ metronom6 } is foreign key to Metronom
{ backTrack1Name, backTrack1Velocity } is foreign key to BackTrack
{ backTrack1Name, backTrack1Velocity } is foreign key to BackTrack
{ backTrack1Name, backTrack1Velocity } is foreign key to BackTrack
{ compasTrack1Name, compasTrack1Velocity } is foreign key to CompasTrack
{ compasTrack2Name, compasTrack2Velocity } is foreign key to CompasTrack
{ compasTrack3Name, compasTrack3Velocity } is foreign key to CompasTrack

Metronom (name, description, file, velocity)

BackTrack (name, description, file, velocity)

CompasTrack (name, description, file, velocity)

Normalización

Utilizaremos la teoría de la normalización para garantizar la separación de conceptos y la ausencia de redundancia para evitar las anomalías de actualización.

· Esta en la primera forma normal (1FN) porque ningún atributo de la relación es por sí mismo una relación, ni descomponible ni con multiplicidad de valores. Los atributos son atómicos.

· Esta en la segunda forma normal (2FN) porque está en la primera forma normal y todo atributo que no forma parte de una clave candidata depende completamente de todas las claves candidatas de la relación.

· Esta en la tercera forma normal (3FN) porque está en la segunda forma normal y ningún atributo que no forma parte de una clave candidata depende del conjunto de atributos que contiene alguno que no forma parte de la clave candidata.

· Esta en formal normal de Boyce-Cood (FNBC) porque está en la tercera forma normal y los determinantes de todas las dependencias que presenta la relación son claves candidatas.

4) Diseño físico [18]

Para el diseño físico de la base de datos se ha elegido el motor de base de datos PostgreSQL. A partir de la normalización del modelo relacional y el

estándar SQL pasamos al diseño físico (ver scripts de creación de la base de datos).

5) Implementación y optimización [19]

Ver anexo.

4.5 Esquema invariante del punto de vista de la información [20]

A partir del enunciado anterior podemos concluir el esquema invariante del punto de vista de la información, según el modelo de referencia par al procesamiento abierto i distribuido (RM-ODP):

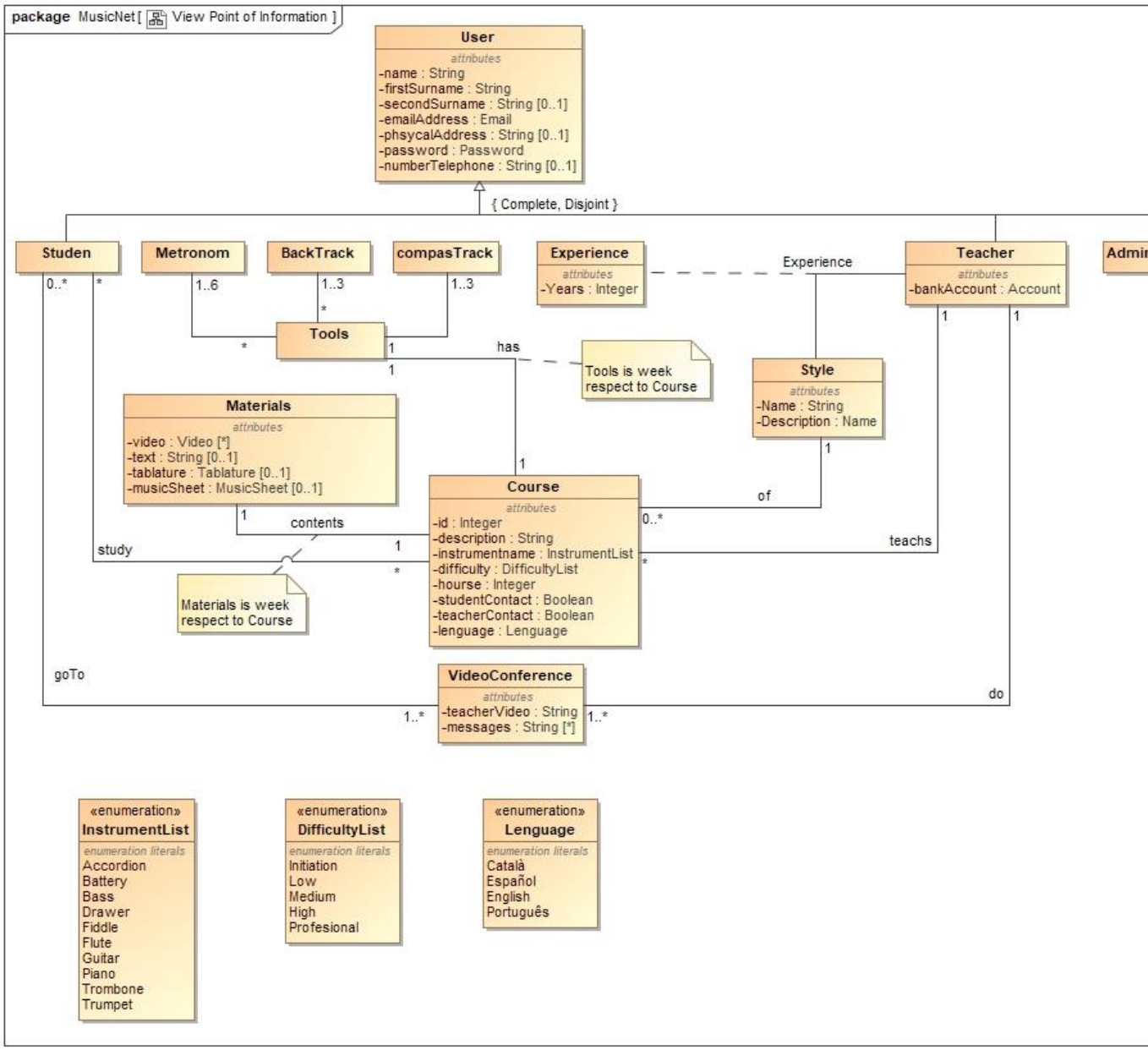


Figura 2: Diseño des del punto de vista de la información

4.6 Diagrama de casos de uso [20]

A continuación, se listan las funcionalidades que en una primera fase ha de ofrecer la aplicación MusicNet. Para cada una de las funcionalidades se especifica los usuarios que hacen uso de ellas: Usuario no registrado (U), Profesor (P), Alumno (A) y Administrador (D):

· Identificarse en el sistema (login)(P,A,D)
· Salir del sistema (logout)(P,A,D)
· Añadir, modificar, borrar o listar estilos musicales (P, D)
· Añadir, modificar, borrar o listar instrumentos musicales (P, D)
· Añadir, modificar, borrar o listar metrónomos, pistas de acompañamiento o pista de compás (P, D)
. Comprar curso (A)
· Vender curso (P, D)
· Añadir, modificar, borrar o listar cursos (P, D)
· Buscar curso (U,P,A,D)
· Realizar aula online en tiempo real (P)
· Asistir a aula online en tiempo real (A, D)
· Buscar profesor por estilo musical (U, P, A, D)
· Modificar los datos personales (P,A,D)
· Registrarse en el sistema (U)

La imagen siguiente muestra el diagrama de casos de uso de las funcionalidades requeridas agrupando los casos de uso en paquetes sobre la base de la funcionalidad que describen.

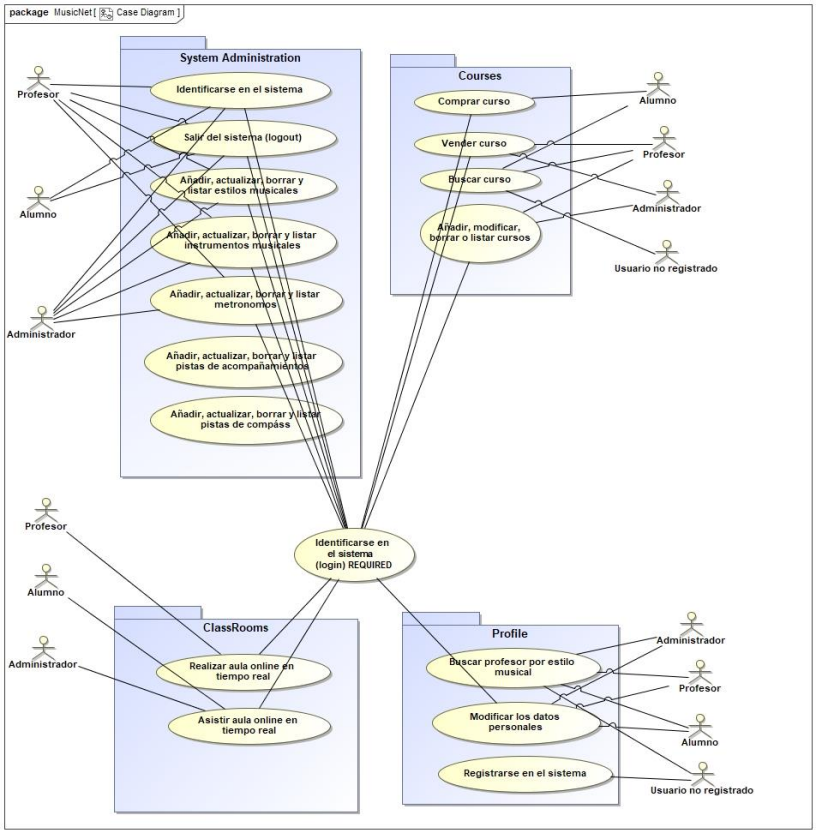


Figura 3: Diagrama de casos de uso

4.7 Punto de vista de la computación [20]

Se representa el diseño interno de la funcionalidad de la aplicación utilizando diagramas de componentes dentro de una arquitectura en capas:

- Cada capa a nivel lógico se representa como un paquete. Dentro de cada paquete, se identifican los componentes lógicos.
- Se describen las interficies de todos los componentes utilizando interficies UML donde se muestra la firma de los servicios que ofrece cada componente.

Por motivos de legibilidad del diagrama las interficies se muestran colapsadas. En los gráficos Presentation, Business y Integration se muestran con todas sus definiciones.

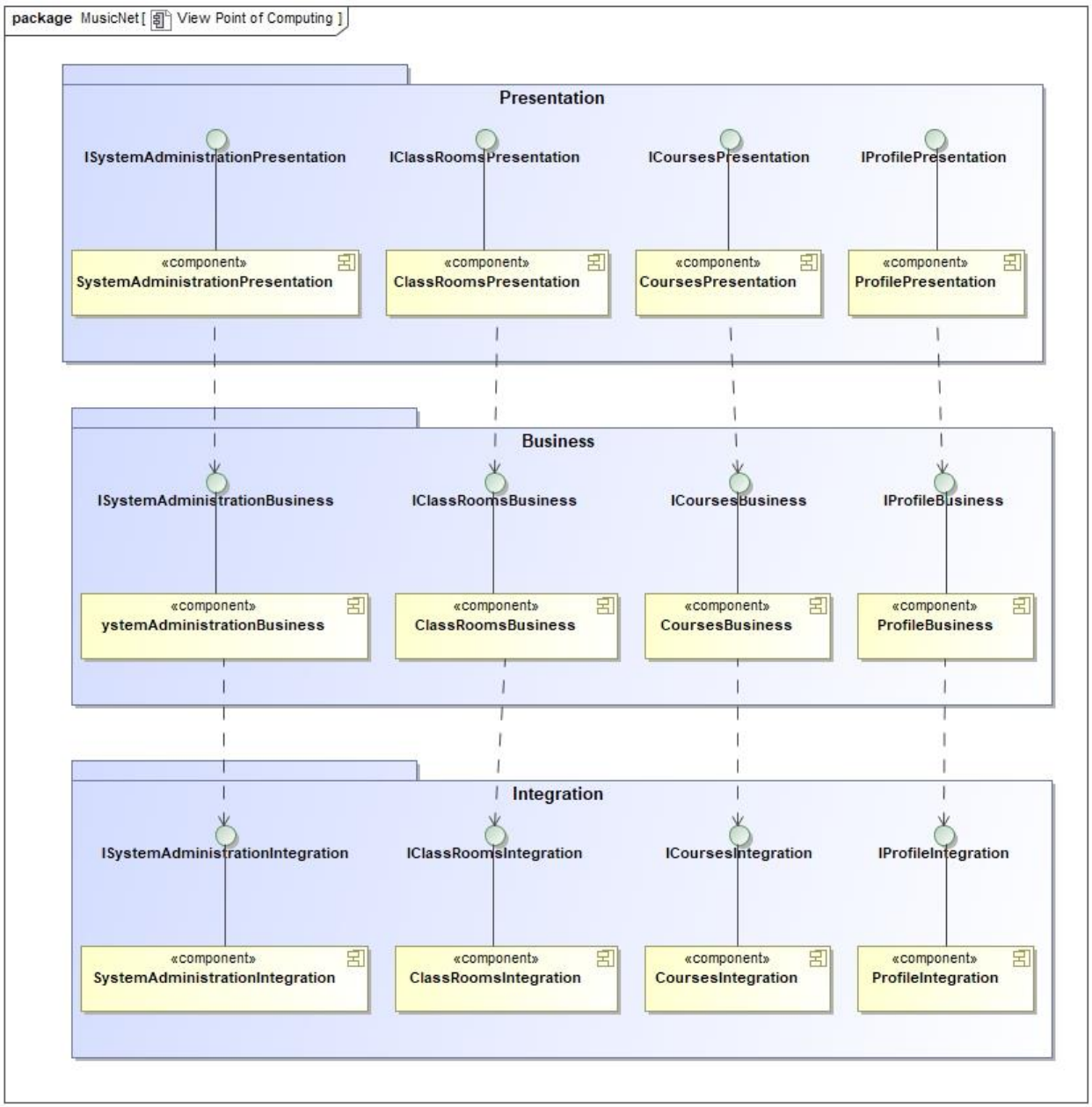


Figura 4: Diagrama des del punto de vista de la computación

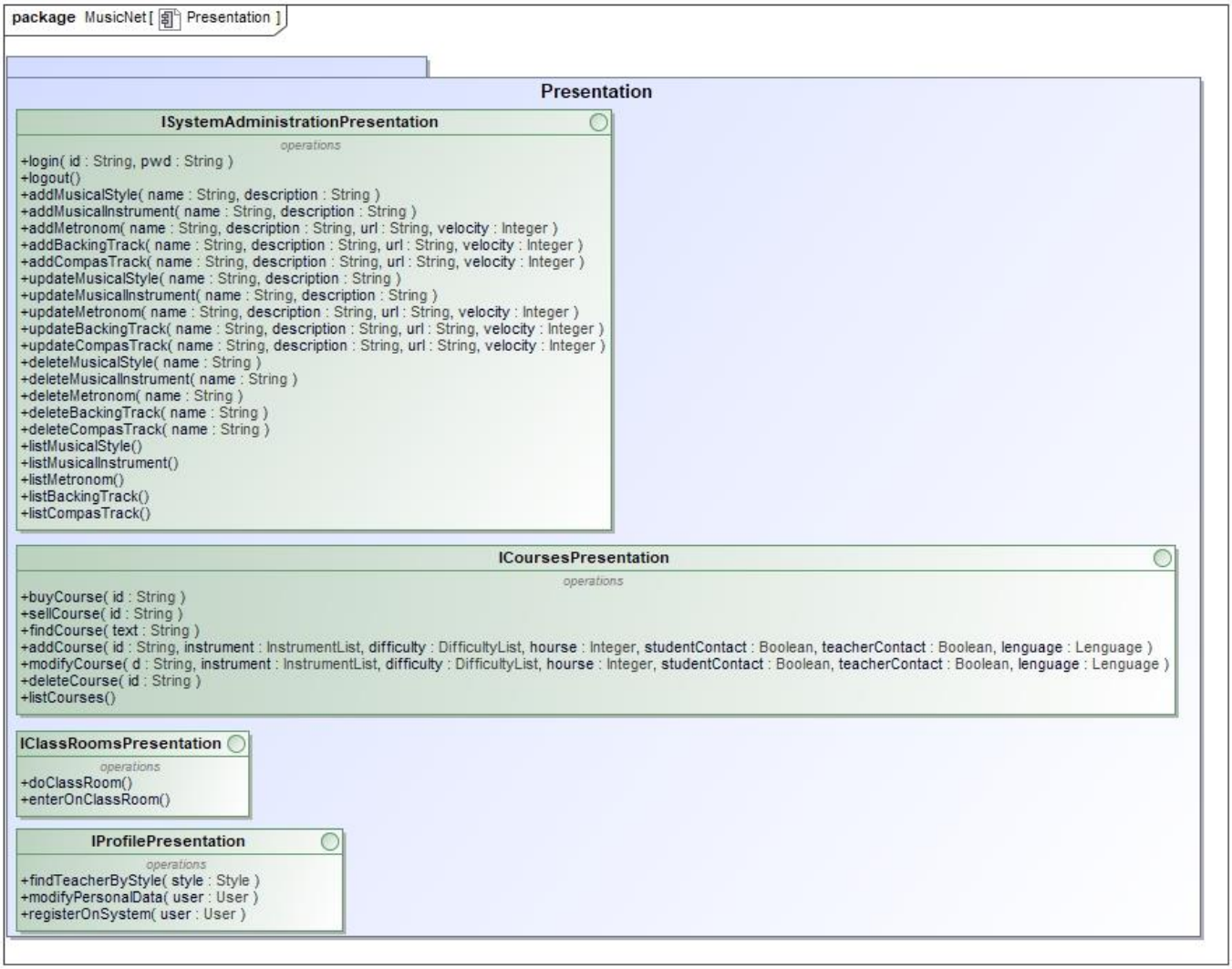


Figura 5: Diagrama de la capa de presentación

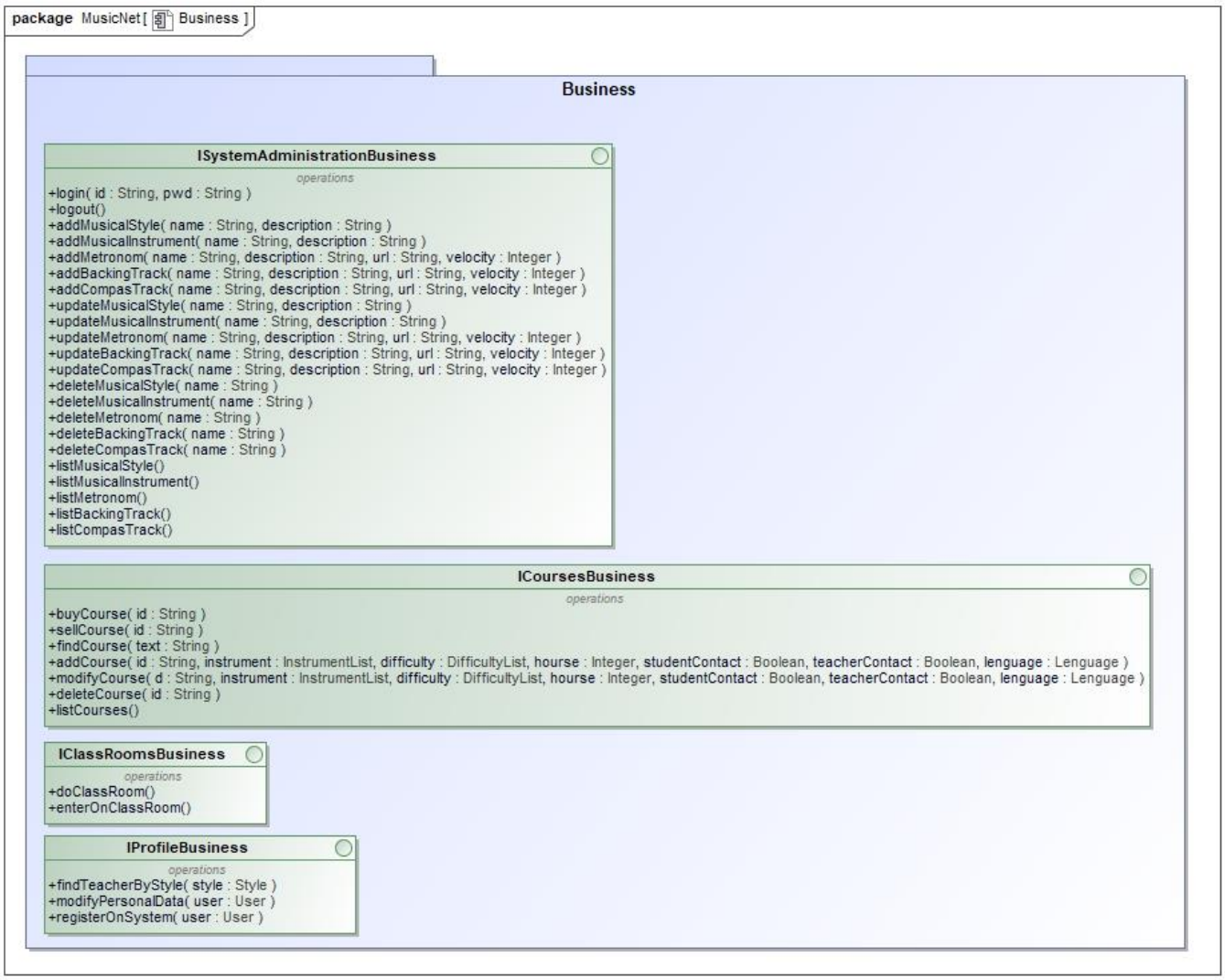


Figura 6: Diagrama de la capa de negocio

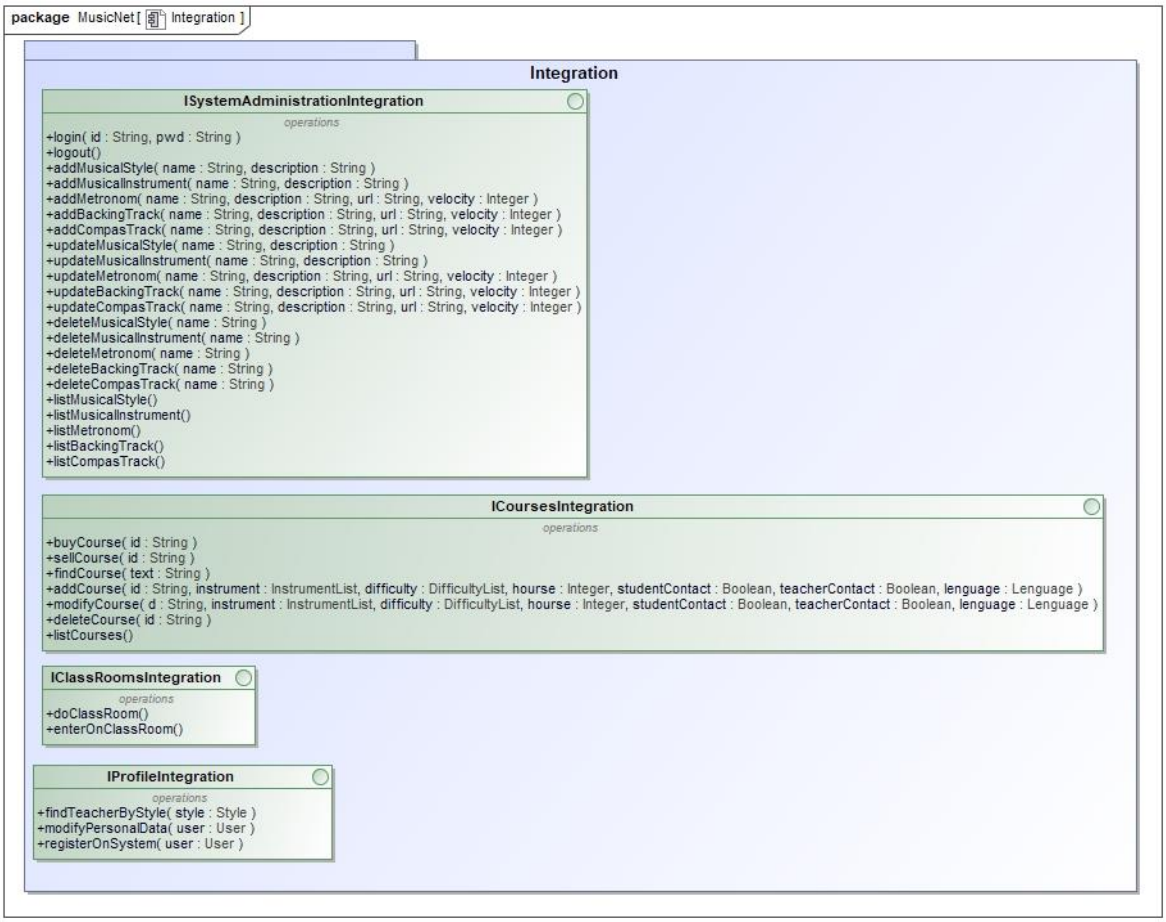


Figura 7: Diagrama de la capa de integración

4.8 Diseño [20]

Primeramente, se afinan los componentes del punto de vista de la computación para obtener un diagrama de componentes de programario.

Capa de presentación

Habrá un único controlador llamado FrontController para todas las operaciones de cada componente.

El controlador será responsable de coordinar la interacción entre las vistas y las acciones, de tal forma que la implementación estará desacoplada.

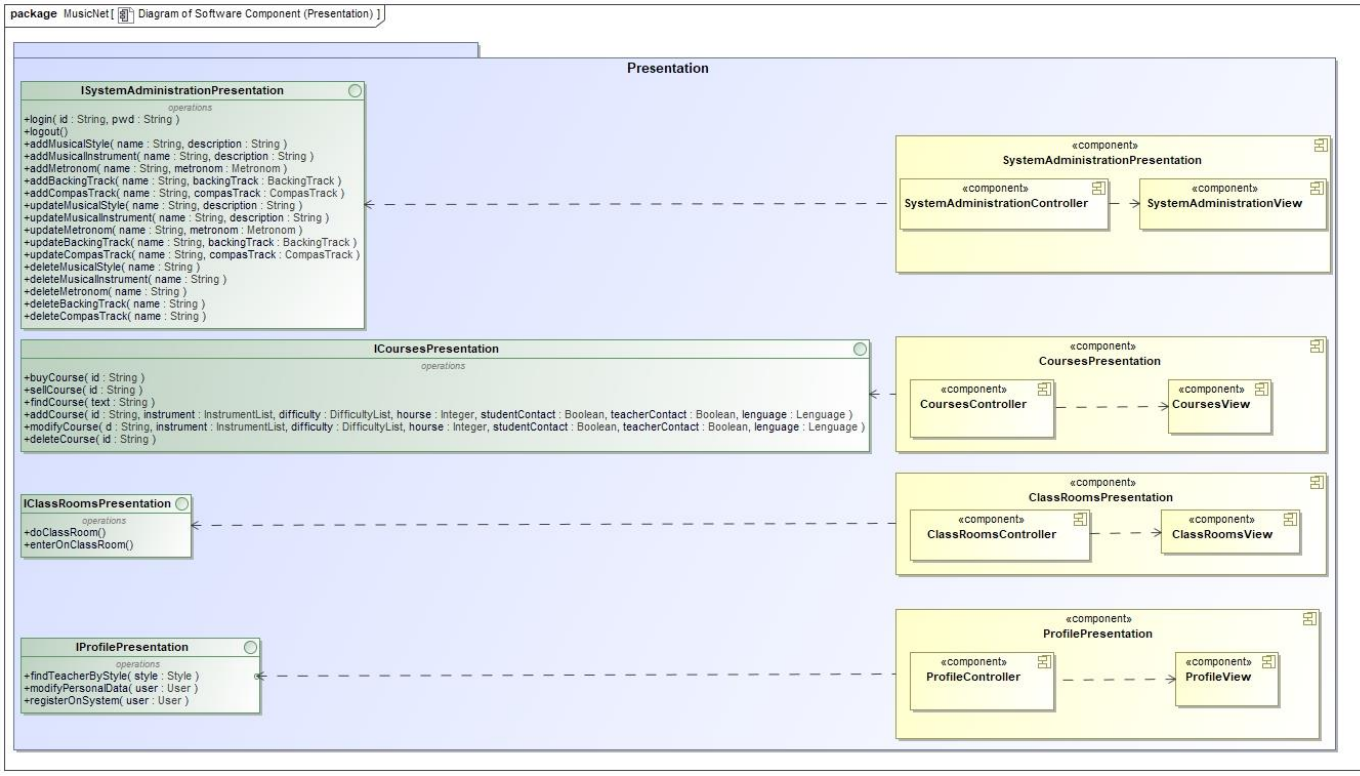


Figura 8: Diagrama de componentes (capa de presentación)

A continuación, se muestra una vista detallada del refinamiento de cada componente:

Módulo de administración

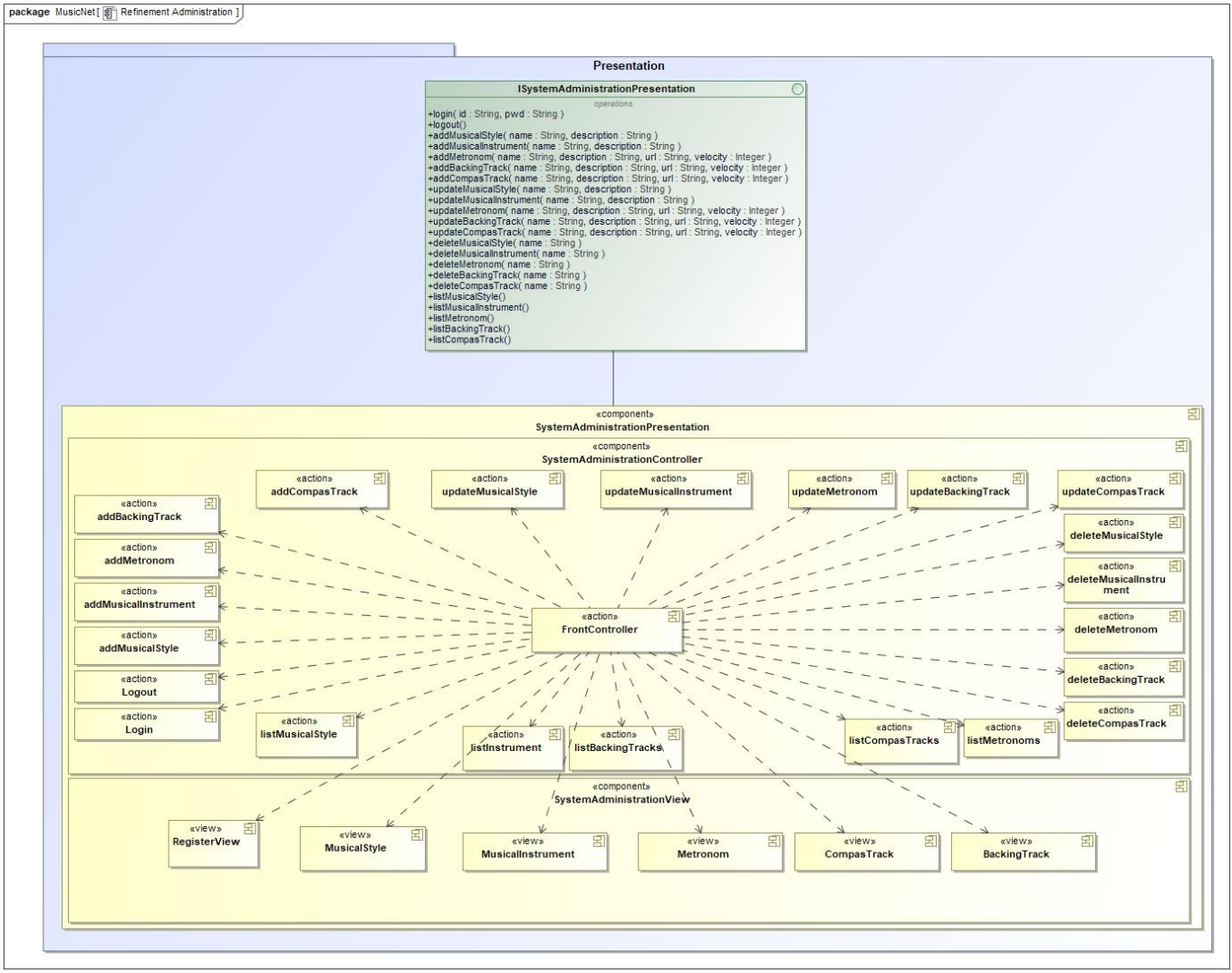


Figura 9: Diagrama de refinamiento (módulo administración)

Módulo de cursos

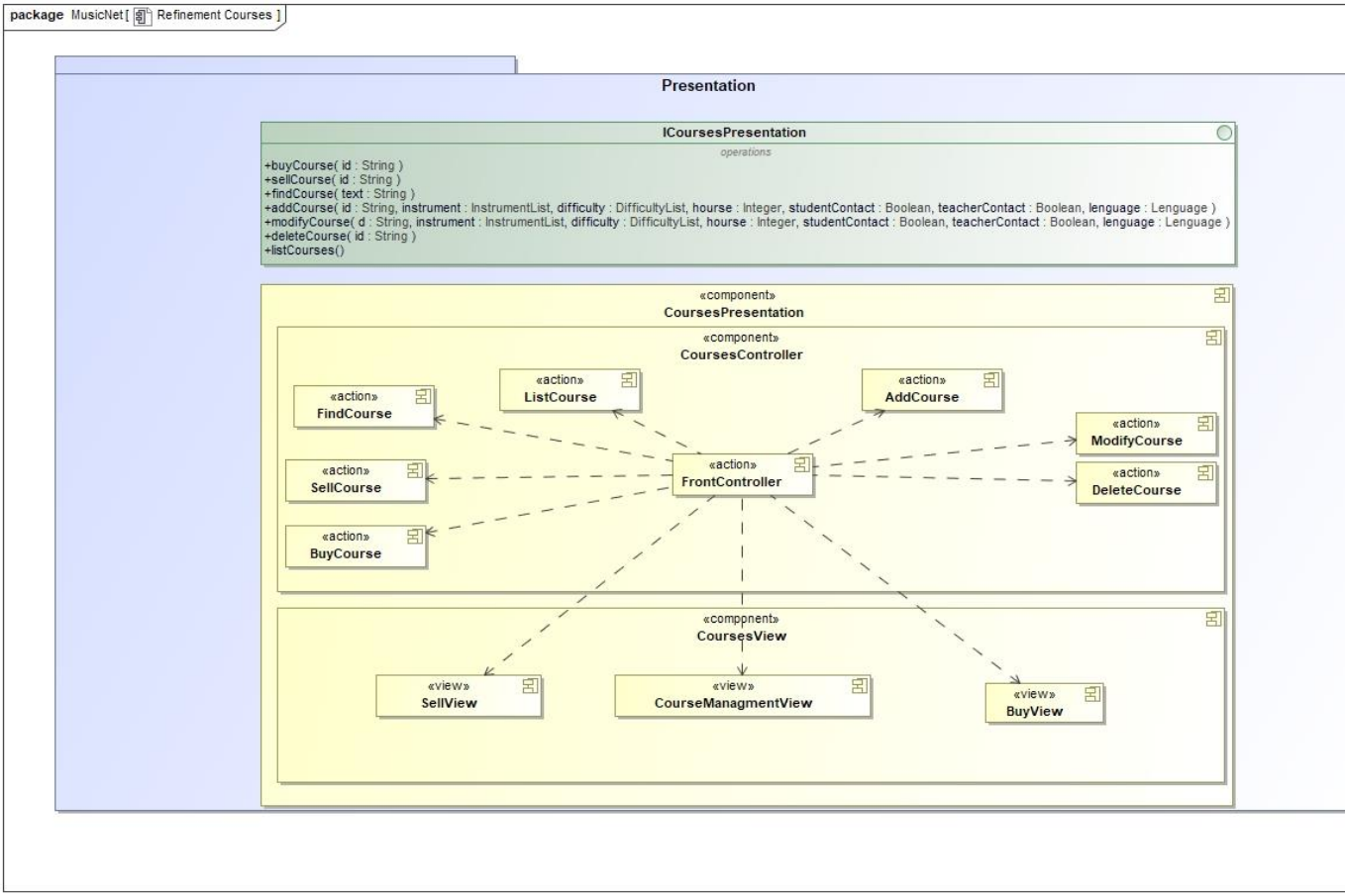


Figura 10: Diagrama de refinamiento (módulo cursos)

Módulo de clases

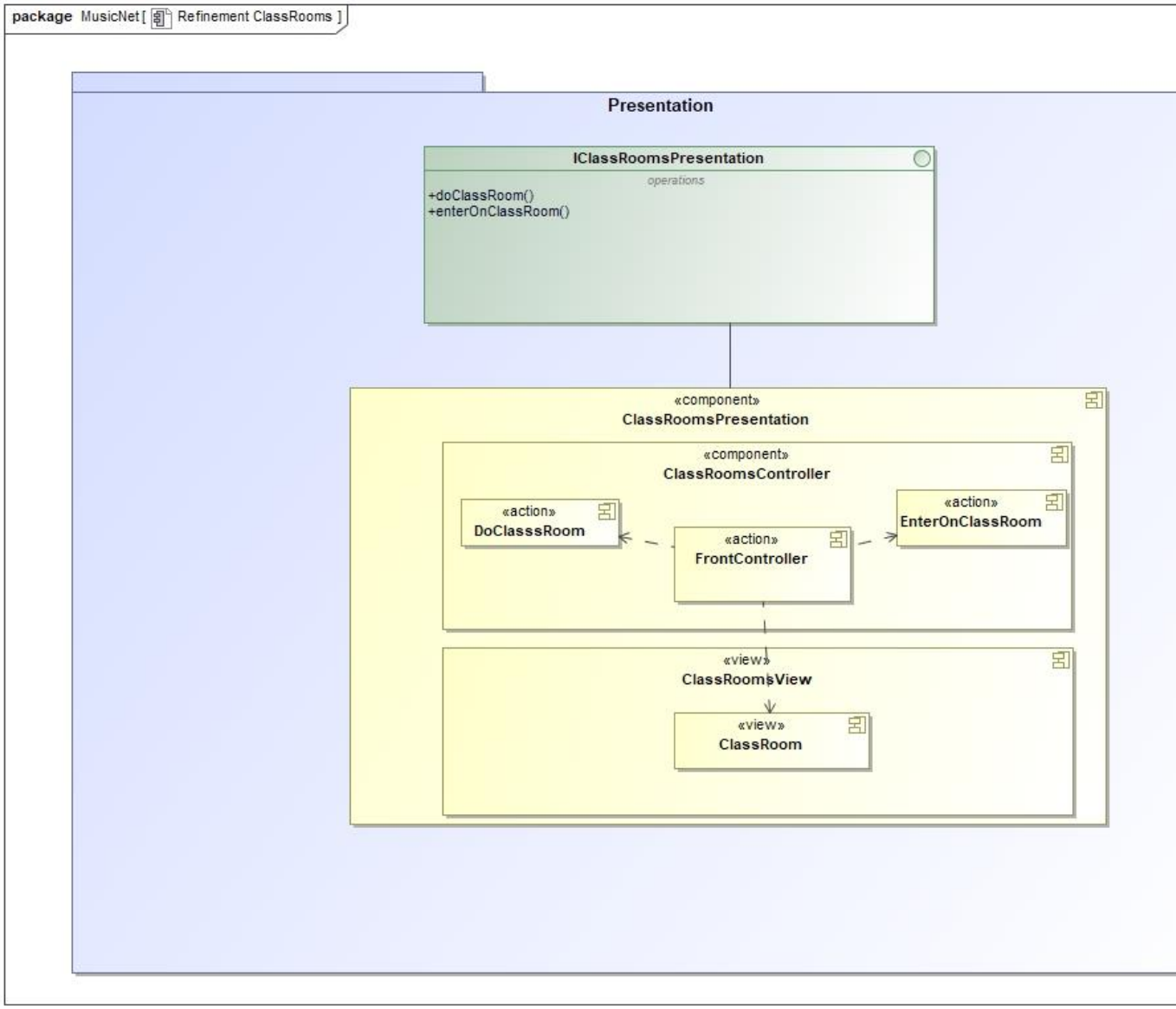


Figura 11: Diagrama de refinamiento (módulo clases)

Módulo del perfil

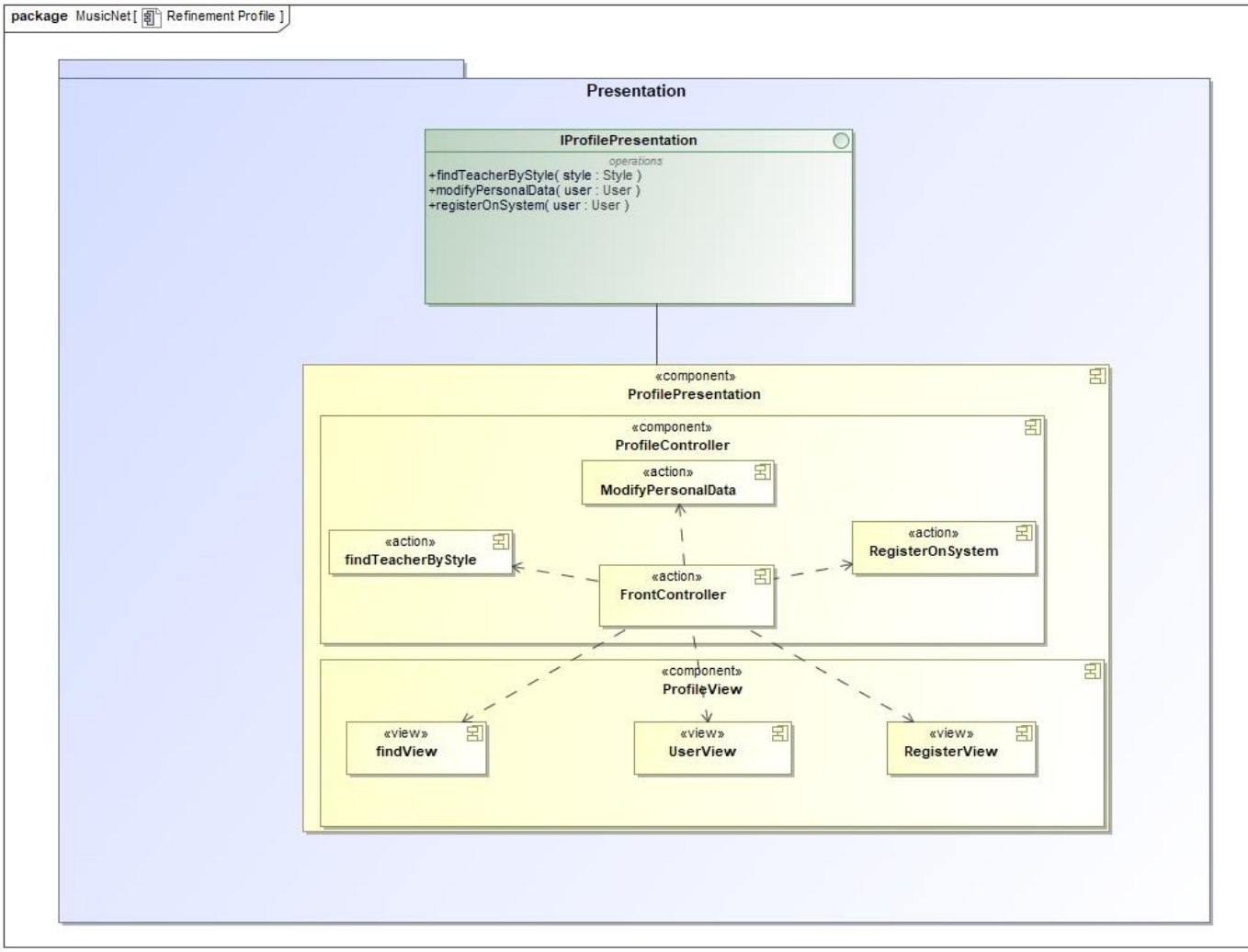


Figura 12: Diagrama de refinamiento (módulo perfil)

Capa de negocio

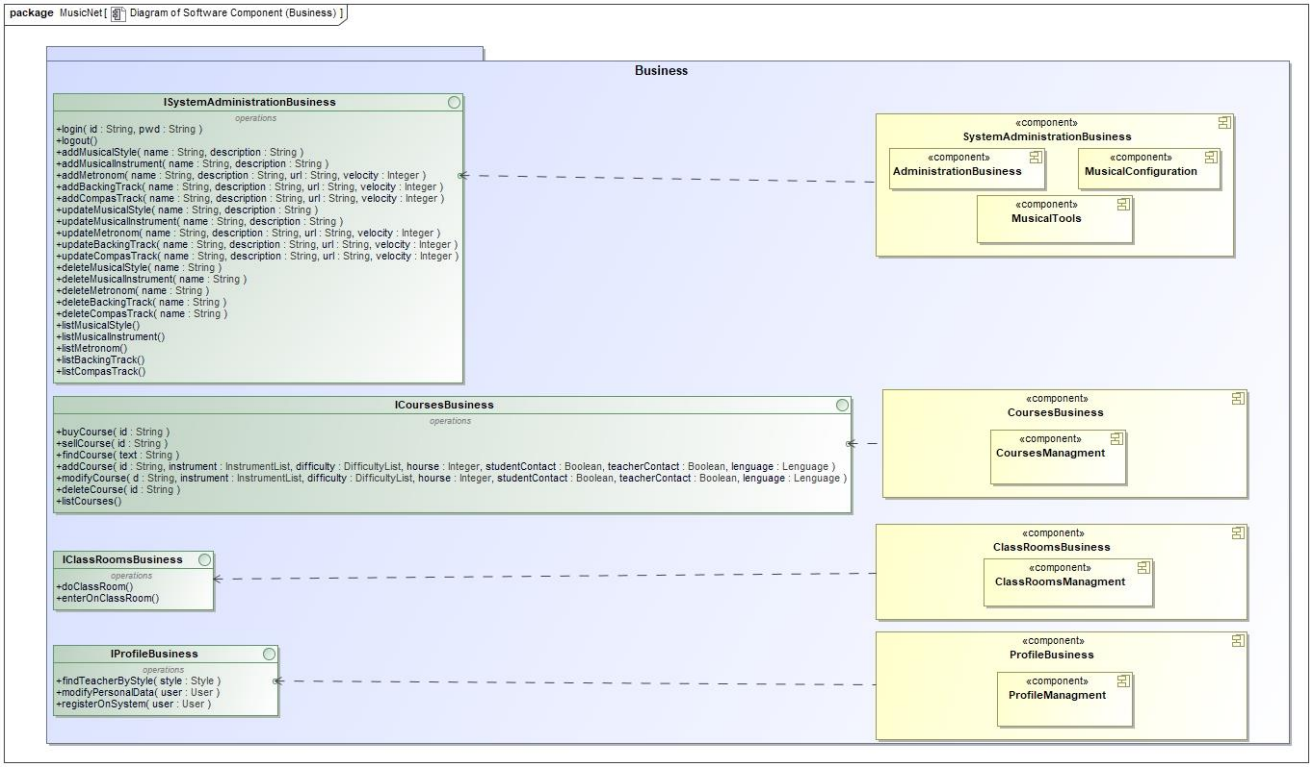


Figura 13: Diagrama de componentes (capa de negocio)

Capa de integración

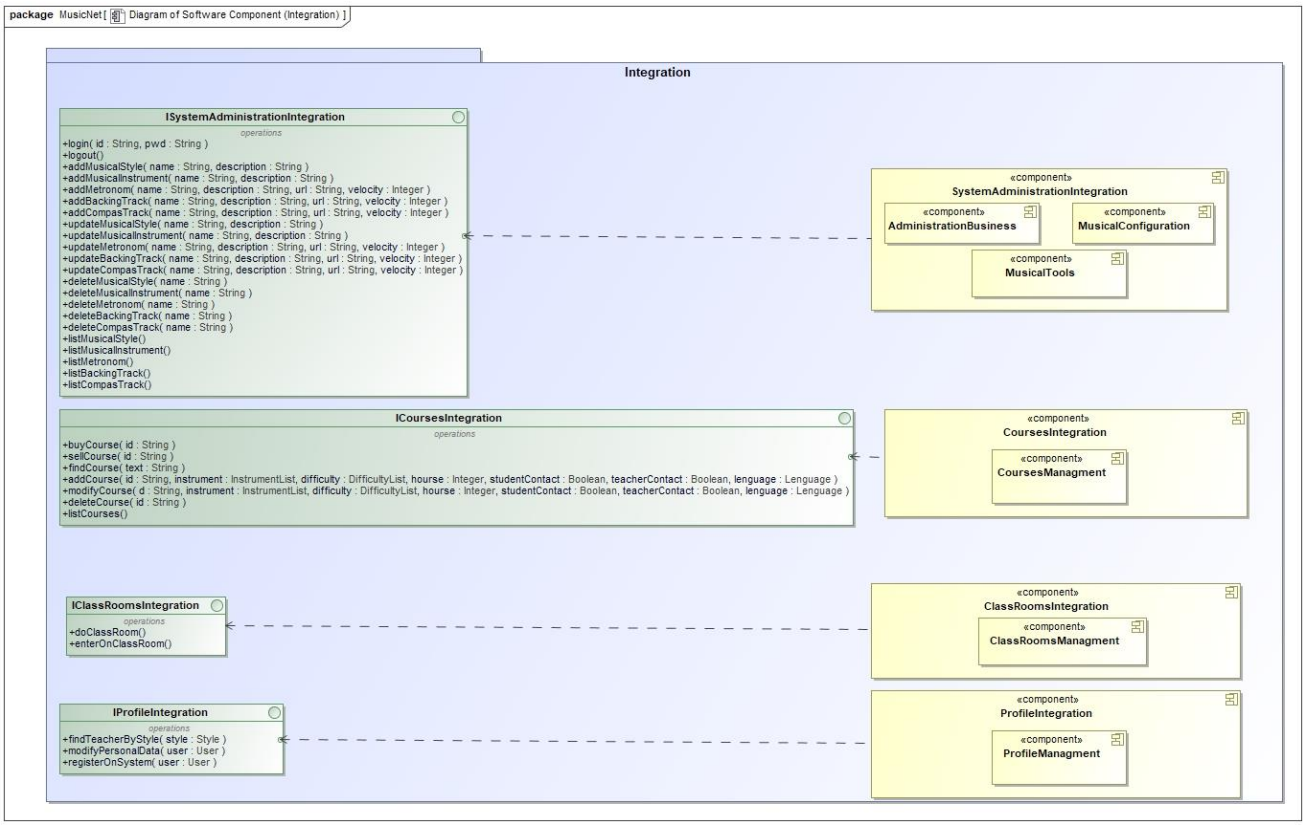


Figura 14: Diagrama de componentes (capa de integración)

Ahora, se aplican las decisiones de diseño oportunas considerando que la tecnología para la implementación será el framework de Java llamado Spring Boot (se aplica el perfil Spring). Además, también se consideran los siguientes puntos:

- Utilizar Spring Boot como la tecnología de componentes distribuida.
- Los datos de los clientes se guardarán en una base de datos relacional.
- El acceso a la capa de negocio puede ser remota o local.
- El acceso a la capa de integración (persistencia) será local.

Capa de presentación

· ”DispatcherServlet” es el responsable de dirigir las solicitudes http hacia un controlador o constructor.

· Se usarán Spring beans con las anotaciones específicas para que actúen como controladores o constructores.

· Para las vistas, se usa Boostrap con Thyemeleaf.

Modulo administración

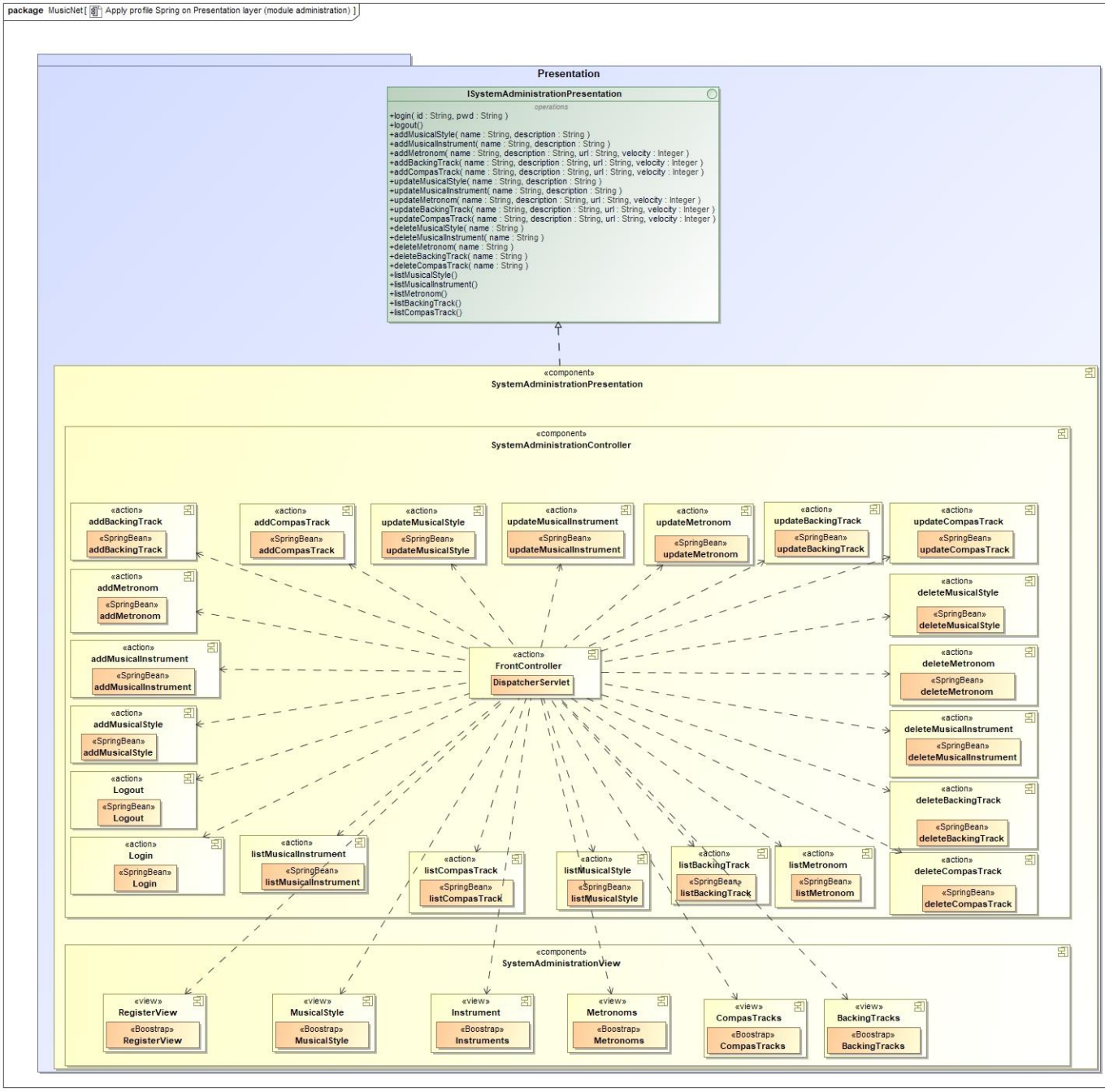


Figura 15: Perfil Spring en la capa de presentación (módulo administración)

Modulo cursos

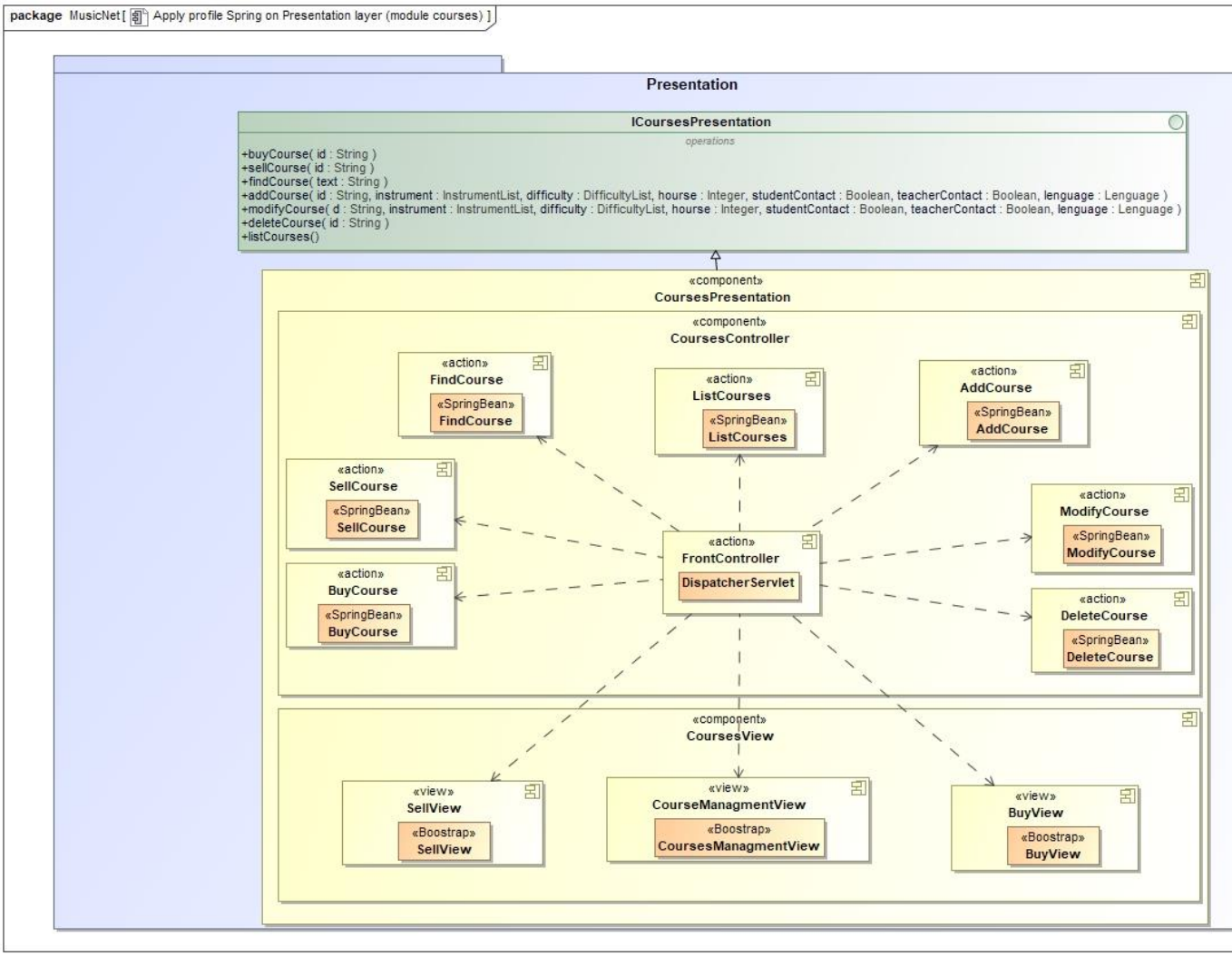


Figura 16: Perfil Spring en la capa de presentación (módulo cursos)

Modulo clases

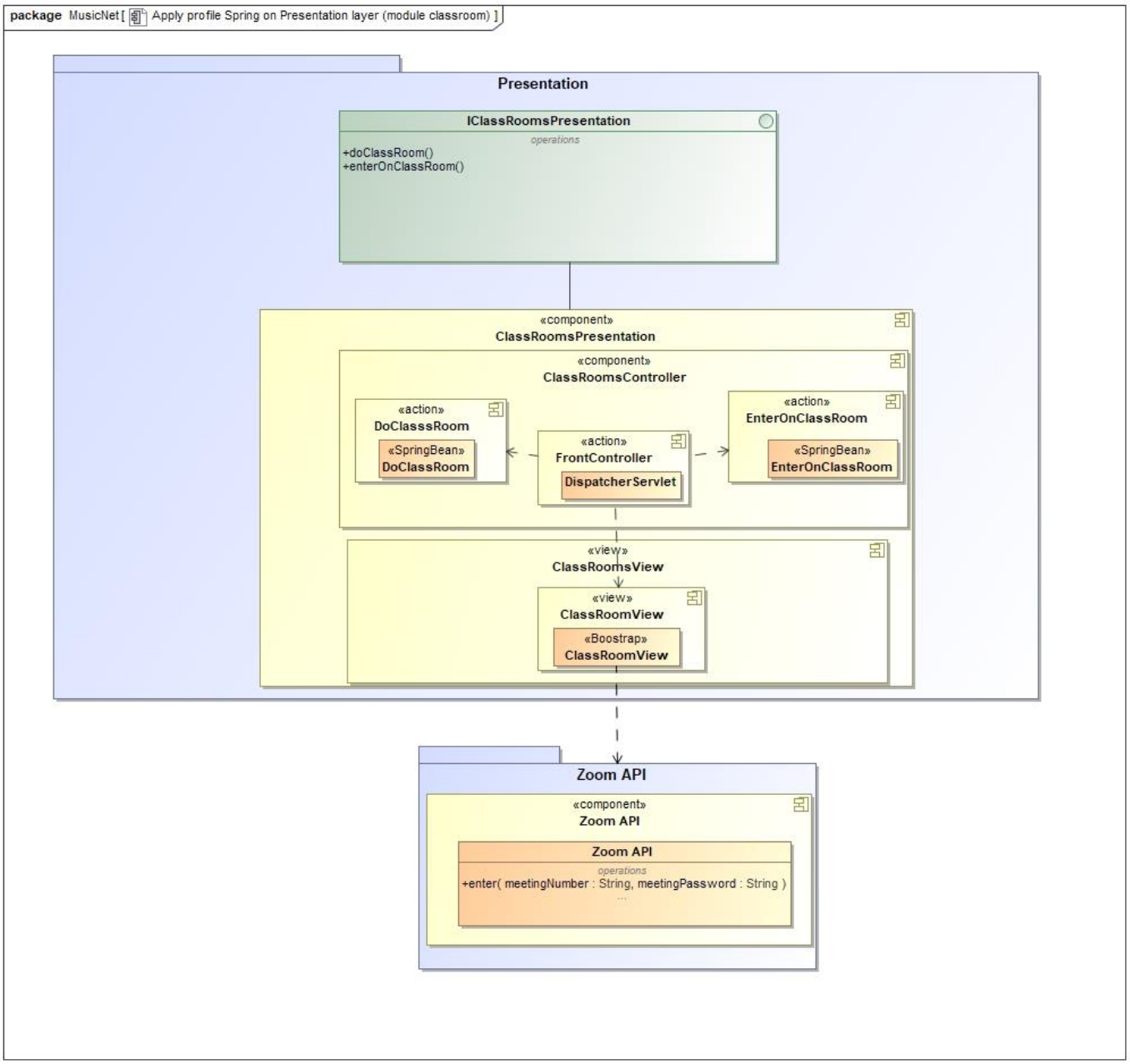


Figura 17: Perfil Spring en la capa de presentación (módulo clases)

Módulo perfil

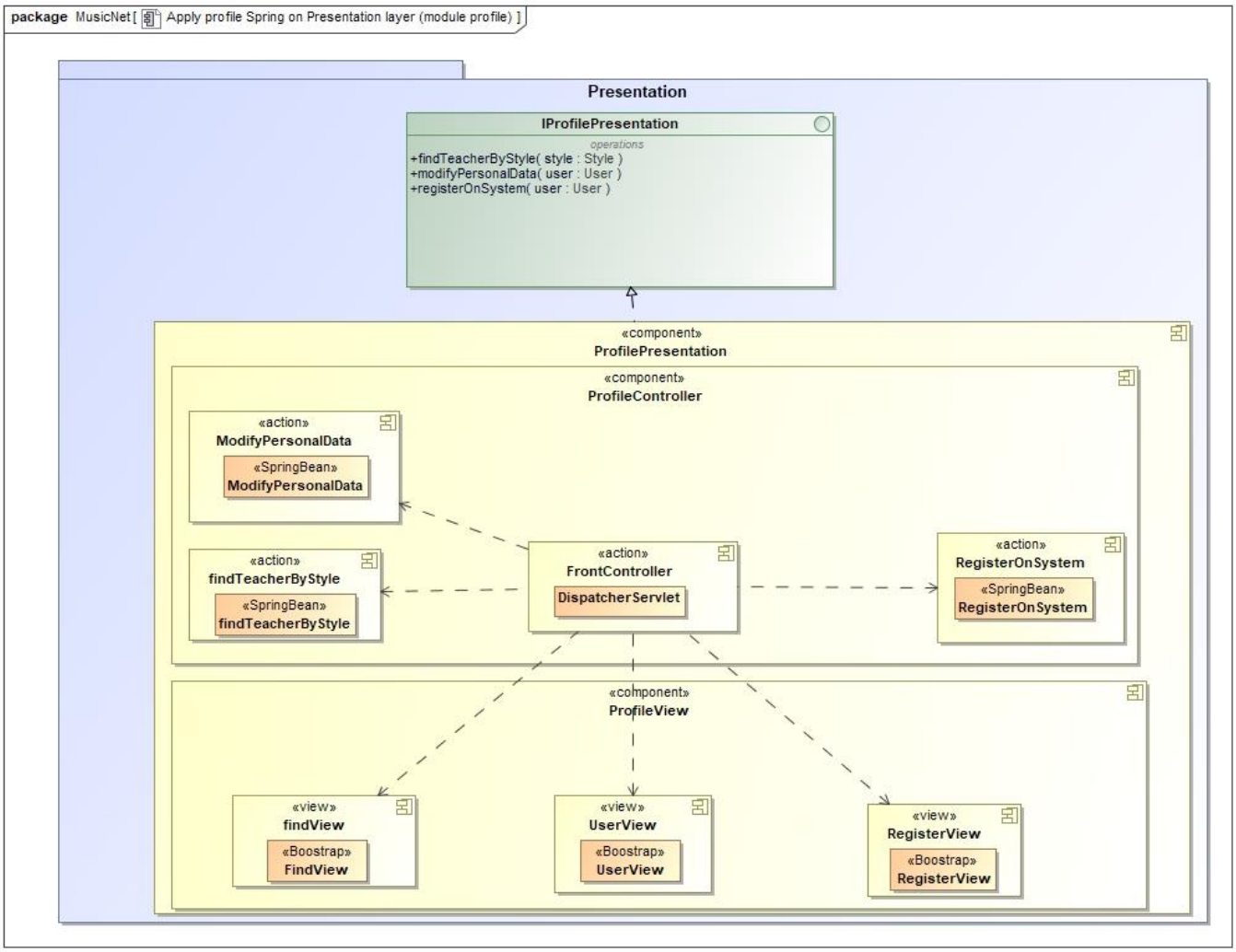


Figura 18: Perfil Spring en la capa de presentación (módulo perfil)

Capa de negocio

· Se usarán Spring beans con las anotaciones específicas para que actúen como servicios.

· Se aplica el patrón “Facade”.

Módulo administración

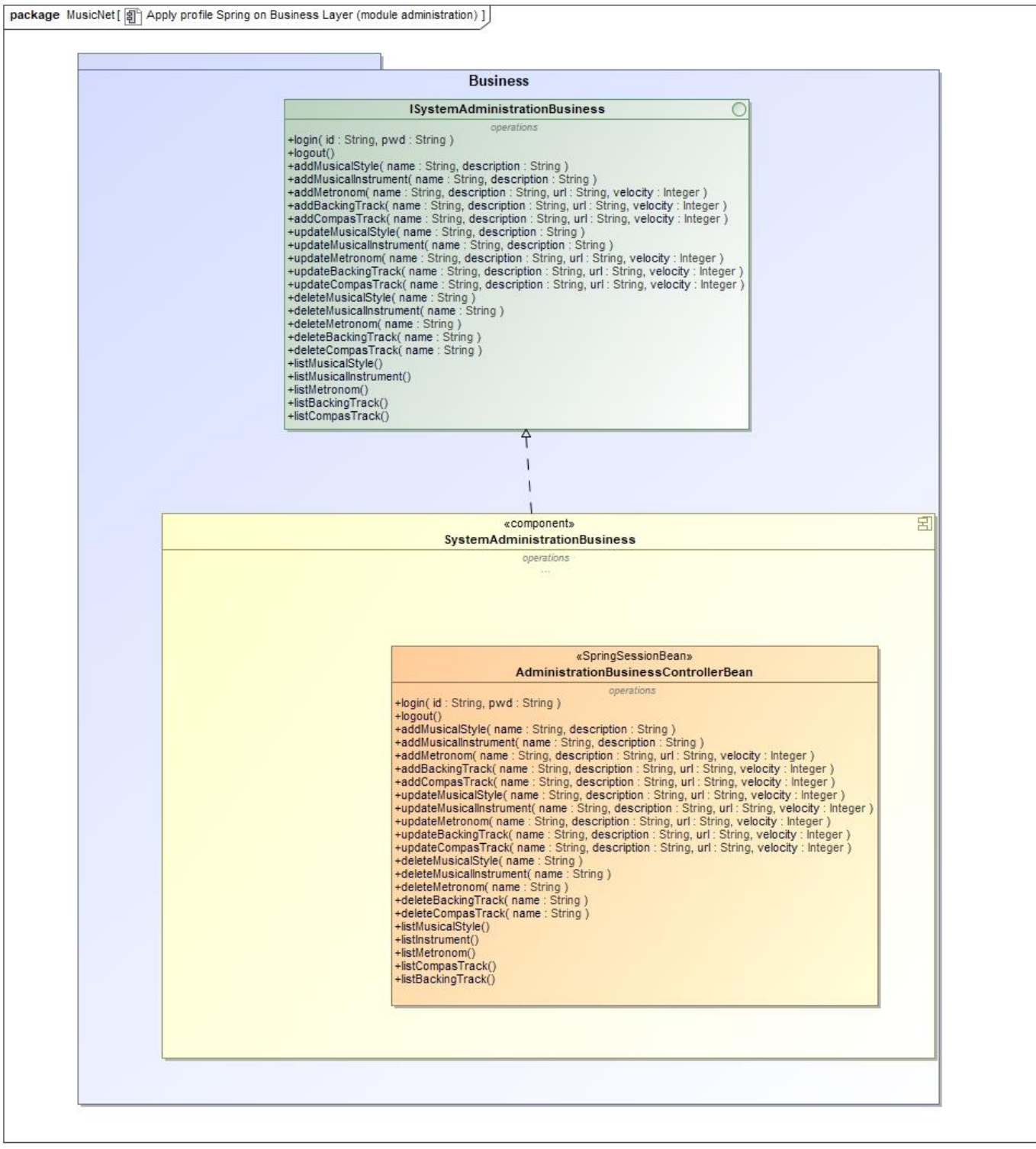


Figura 19: Perfil Spring en la capa de negocio (módulo administración)

Módulo cursos

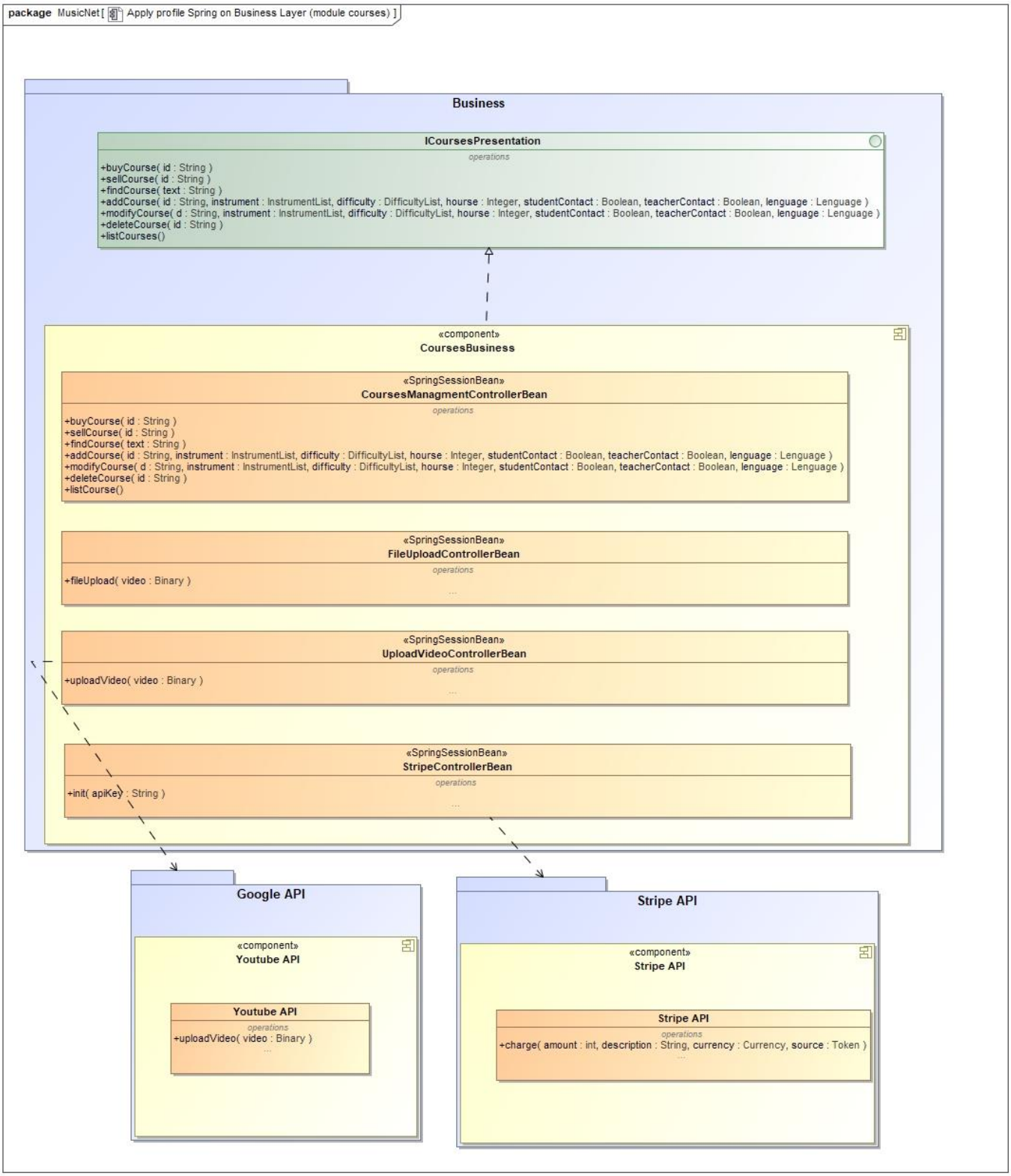


Figura 20: Perfil Spring en la capa de negocio (módulo cursos)

Módulo clases

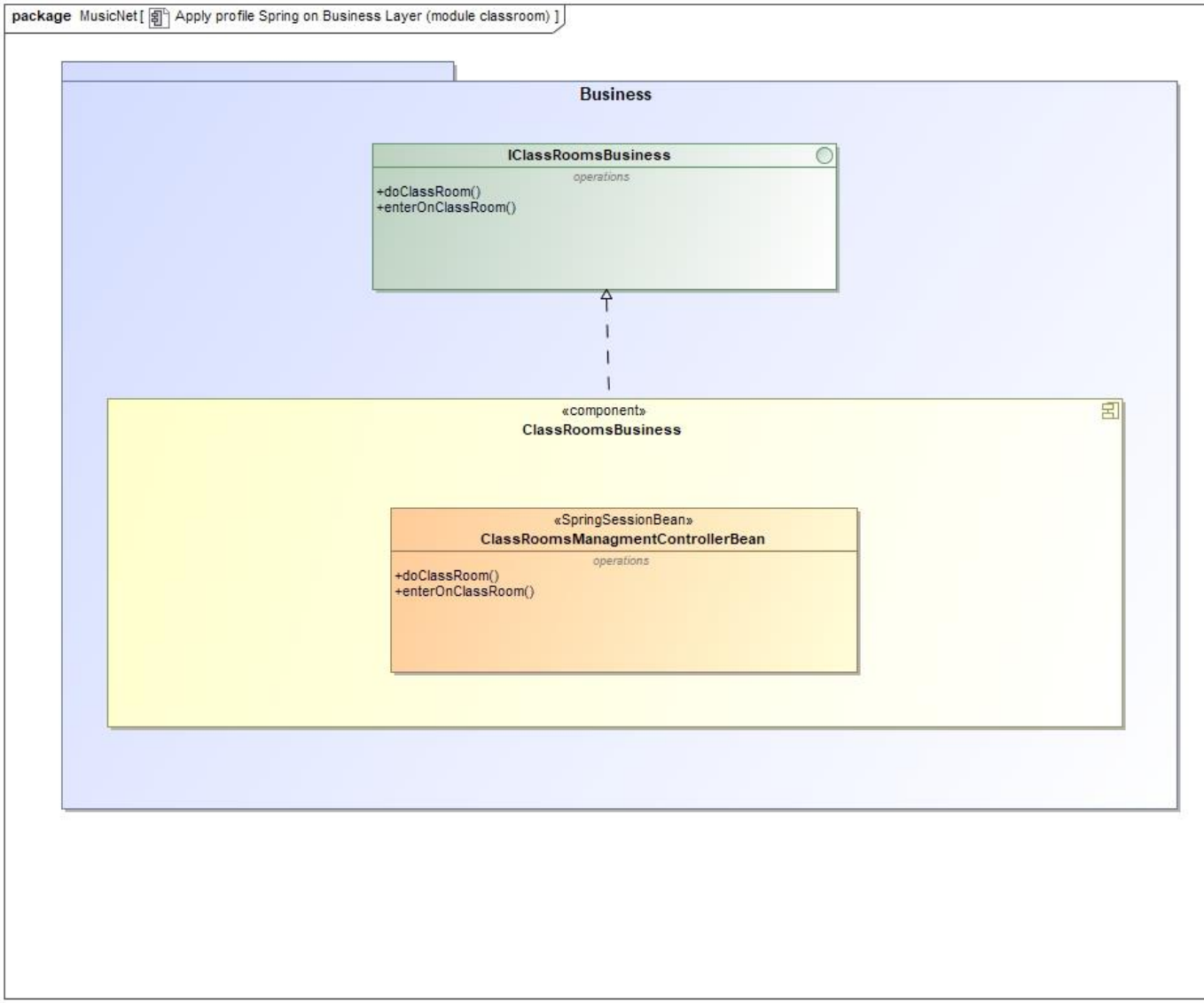


Figura 21: Perfil Spring en la capa de negocio (módulo clases)

Módulo perfil

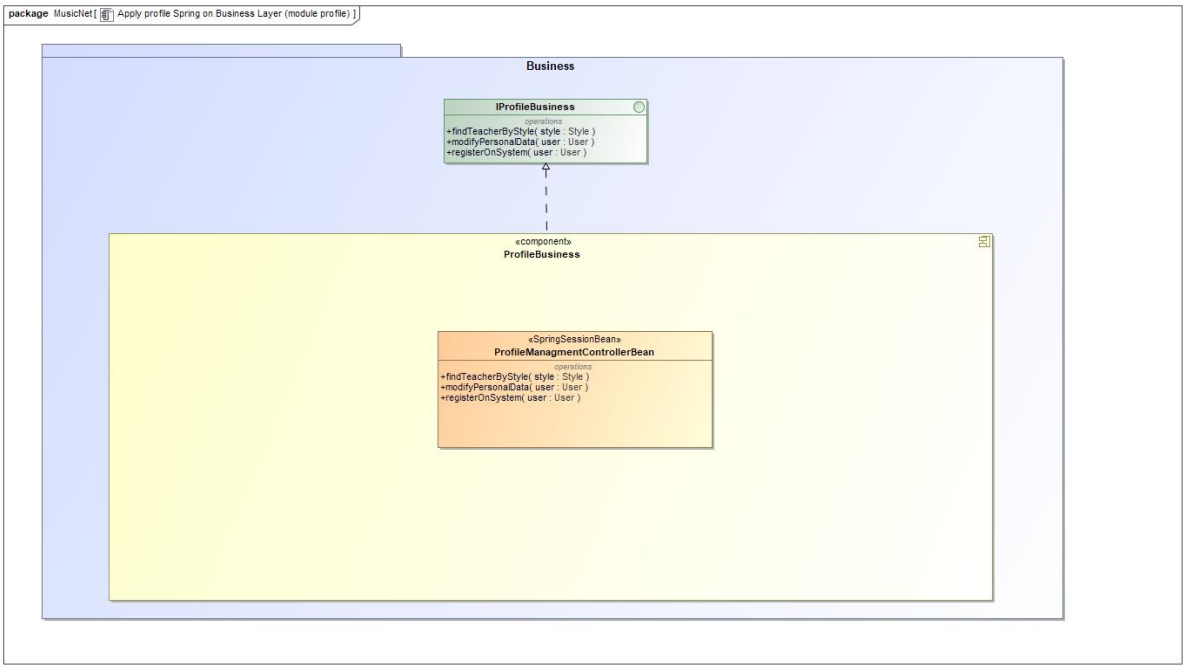


Figura 22: Perfil Spring en la capa de negocio (módulo perfil)

Capa de integración

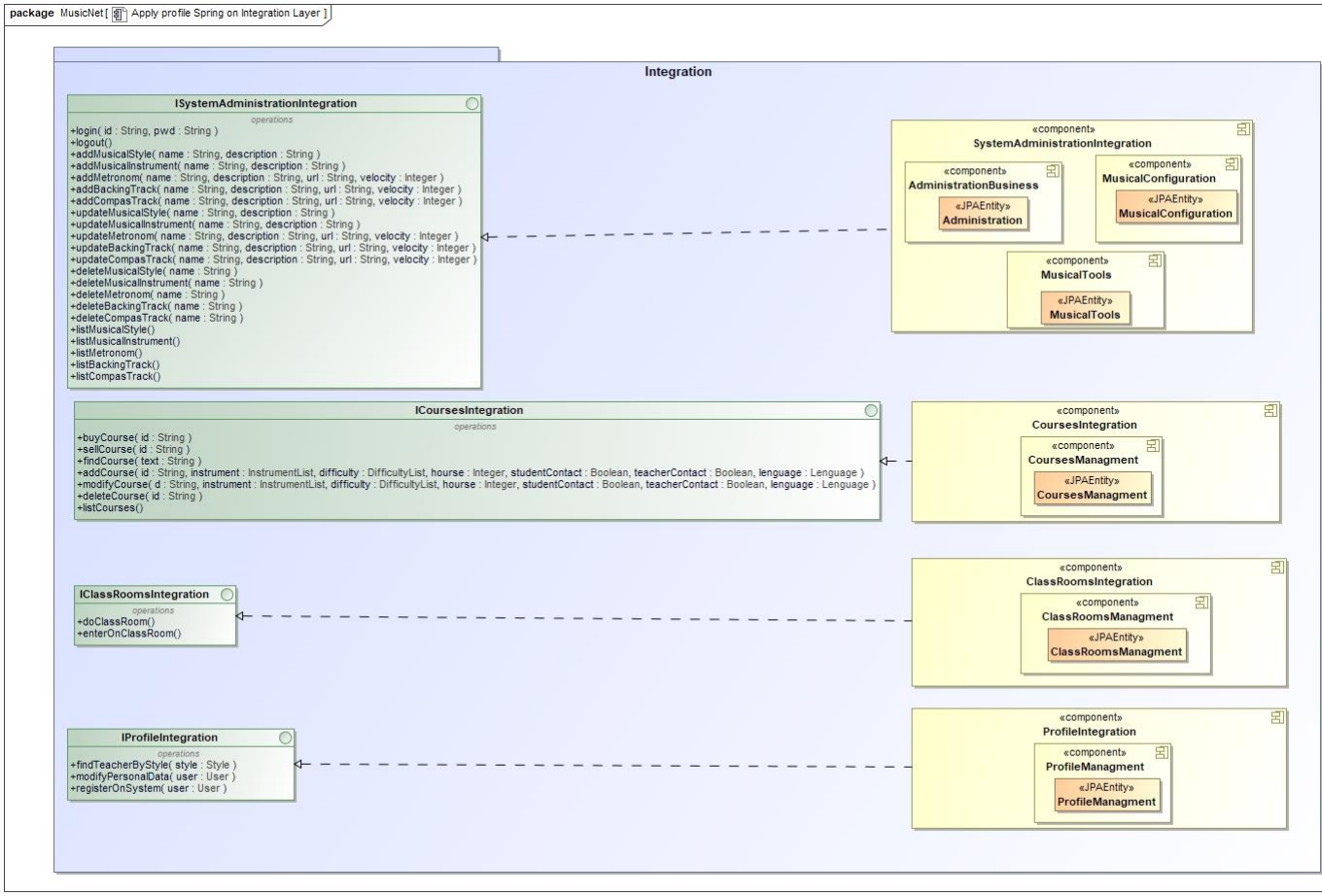


Figura 23: Perfil Spring en la capa de integración

4.9 Implementación

Código fuente

Se puede ver el código fuente en https://github.com/GitHubUserWorld/MusicNet

Vistas

Por otro lado, en cada una de las vistas se encontrará el menú de navegación que se puede ver en las siguientes imágenes y que se explicará en detalle.

Vista inicial: se encuentra una explicación breve de las diferentes funcionalidades del producto (gestión de cursos, gestión de alumnos y profesores, etc) de una forma llamativa para el usuario.

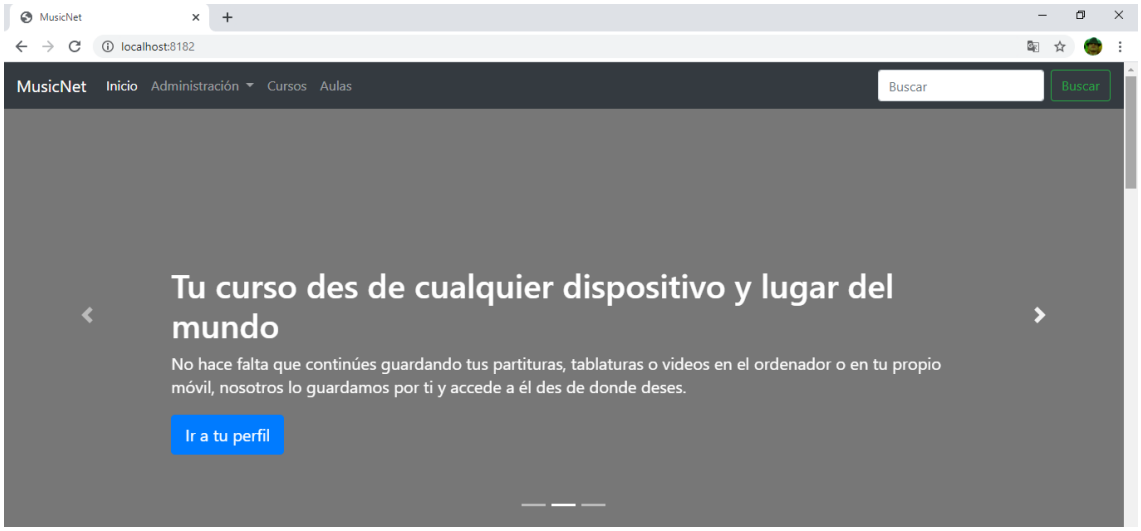


Figura 24: Vista inicial

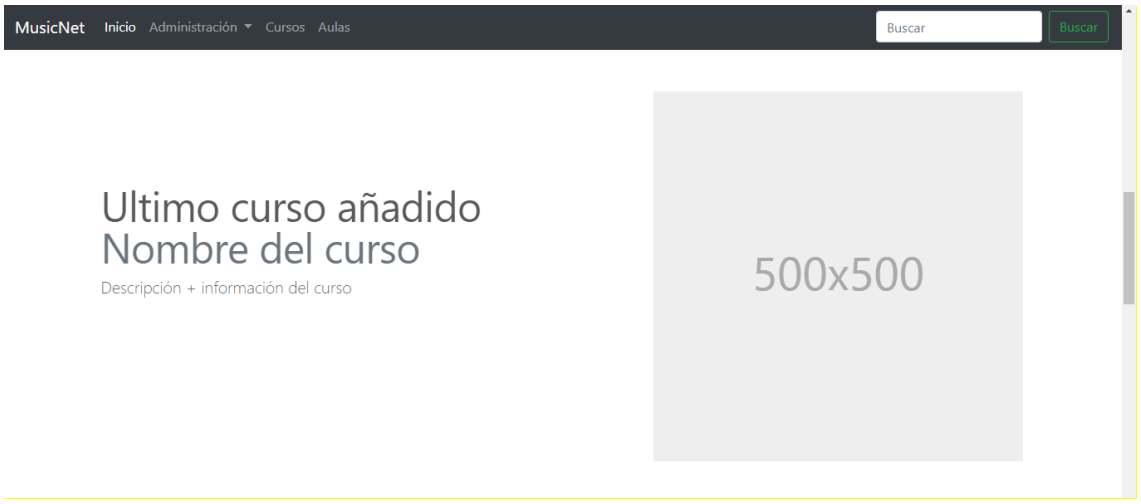


Figura 25: Vista inicial

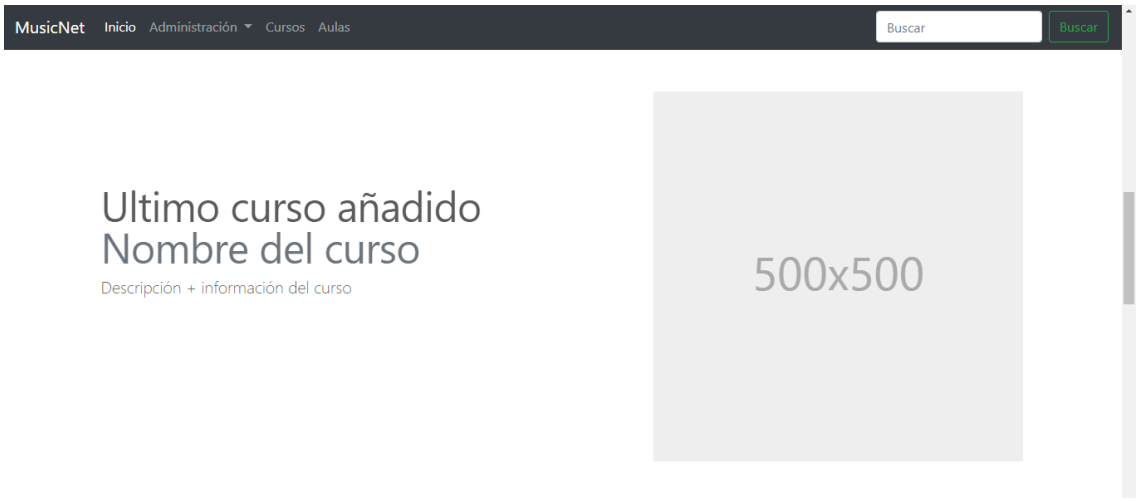


Figura 26: Vista inicial

Vista administración: en el menú Administración se encuentran todas las funcionalidades relacionadas con el módulo administración. Al clicar sobre él, se abre otro submenú des del cual se puede acceder a la gestión de dicho módulo. Por ejemplo, dentro de la gestión de estilos musicales se encontrará una pestaña con la lista de todos ellos, otra pestaña para añadir un nuevo estilo, otra para modificar cualquier estilo y una última para borrarlos. Esta plantilla será la misma para todo el módulo administración (estilos musicales, instrumentos, metrónomos, bases de compas y bases de acompañamiento).

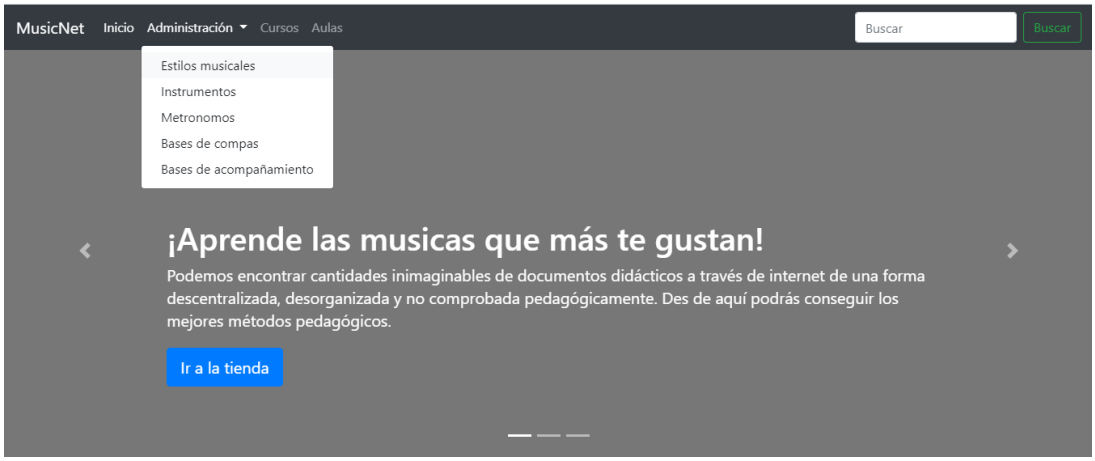


Figura 27: menú administración

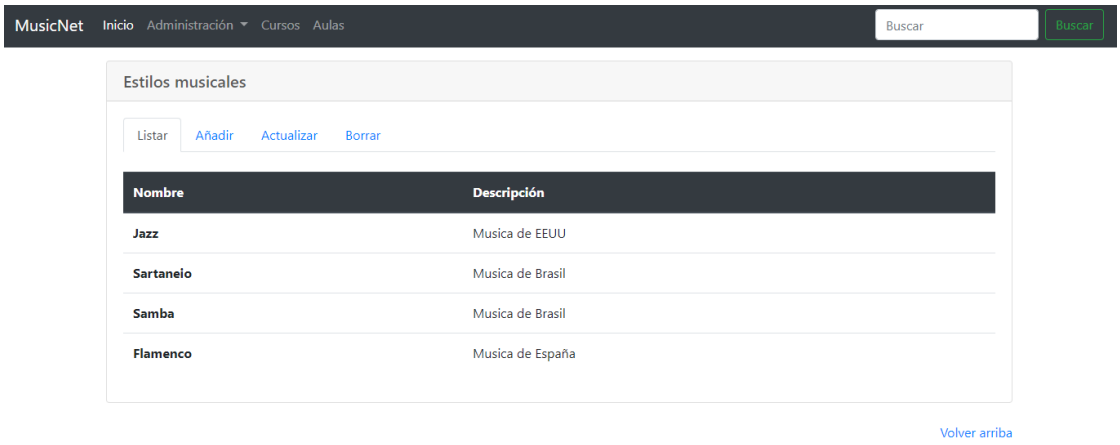


Figura 28: listado de estilos musicales

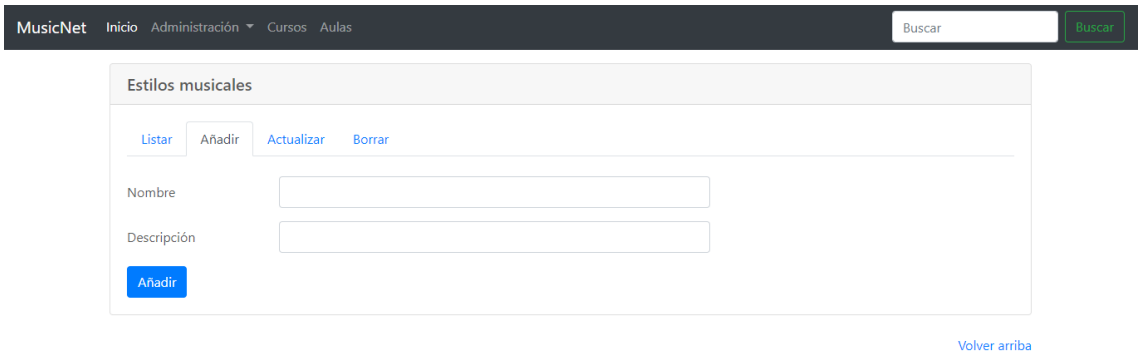


Figura 29: añadir de estilo musical

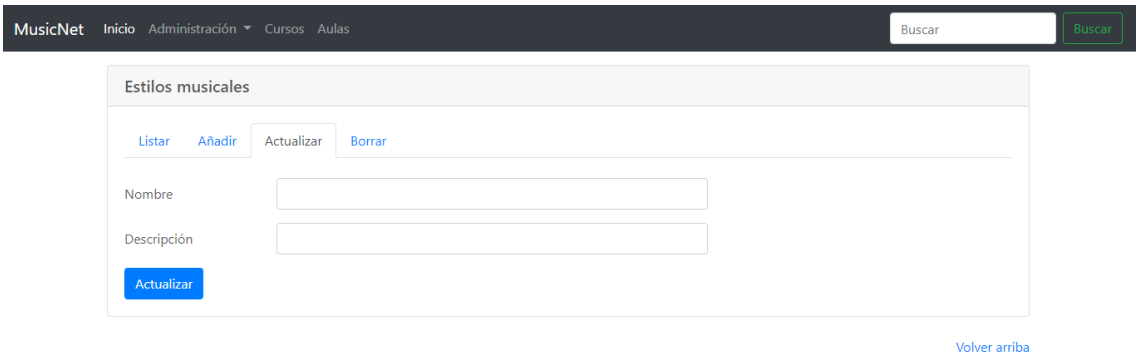


Figura 30: Actualizar estilo musical

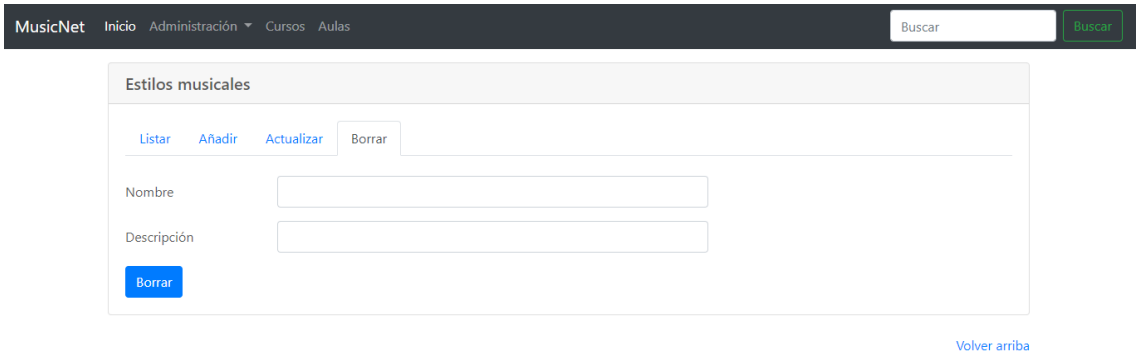


Figura 31: Borrar estilo musical

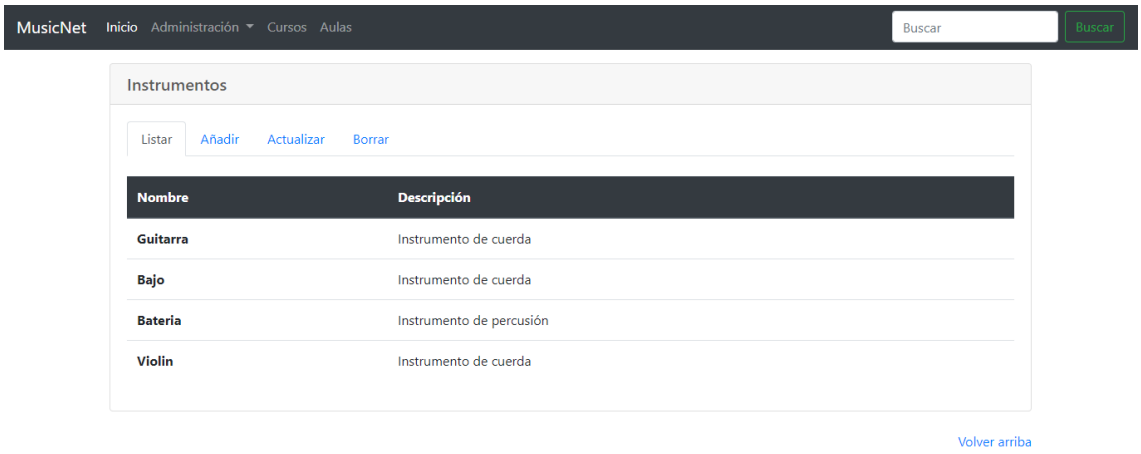


Figura 32: listado de instrumentos

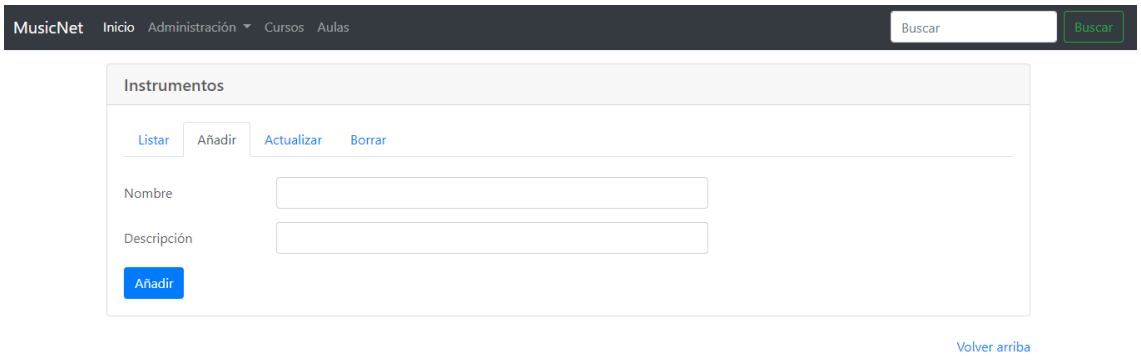


Figura 33: añadir instrumento

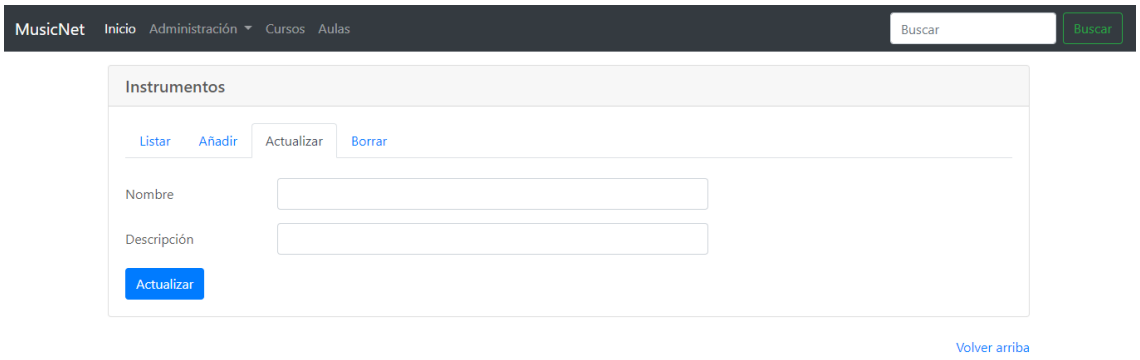


Figura 34: modificar instrumento

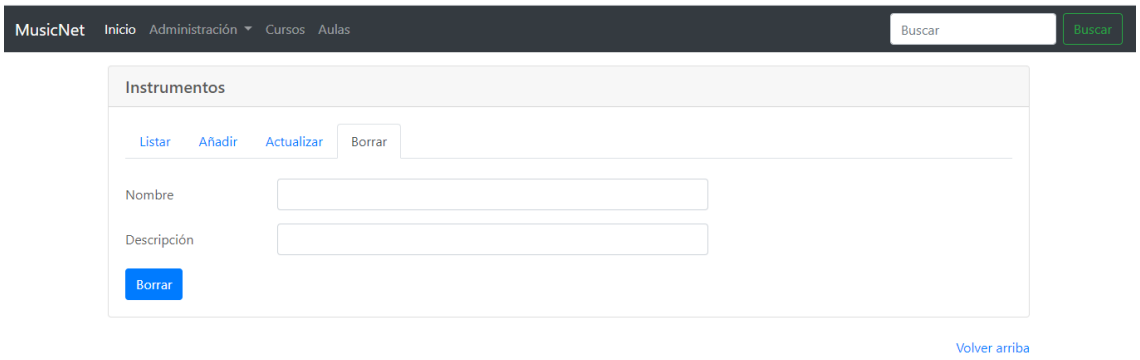


Figura 35: borrar instrumento

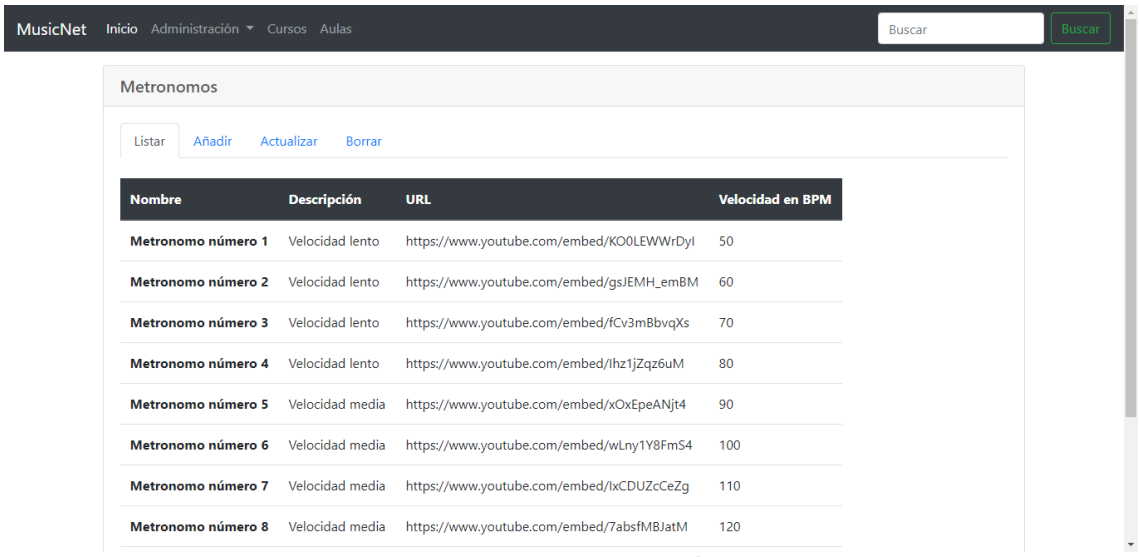


Figura 36: listado de metrónomos

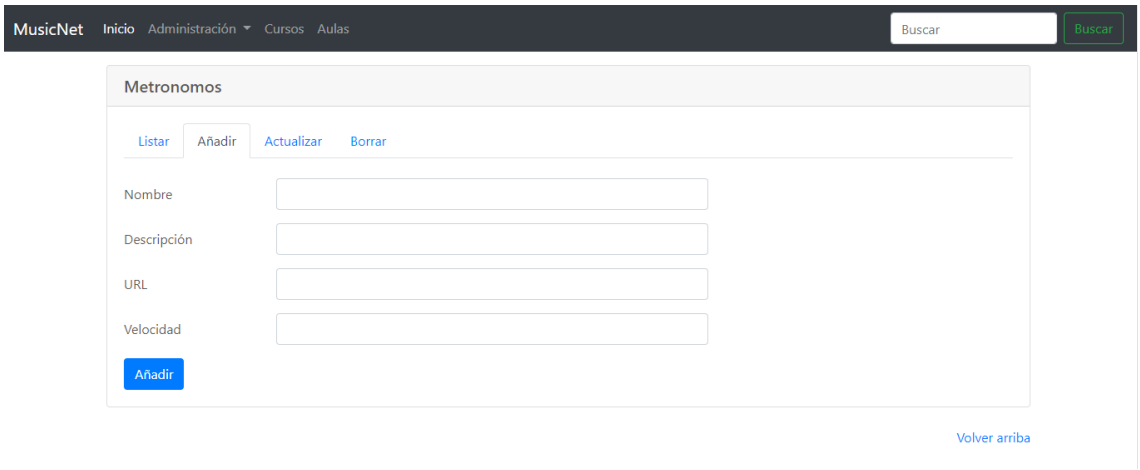


Figura 37: añadir metrónomo

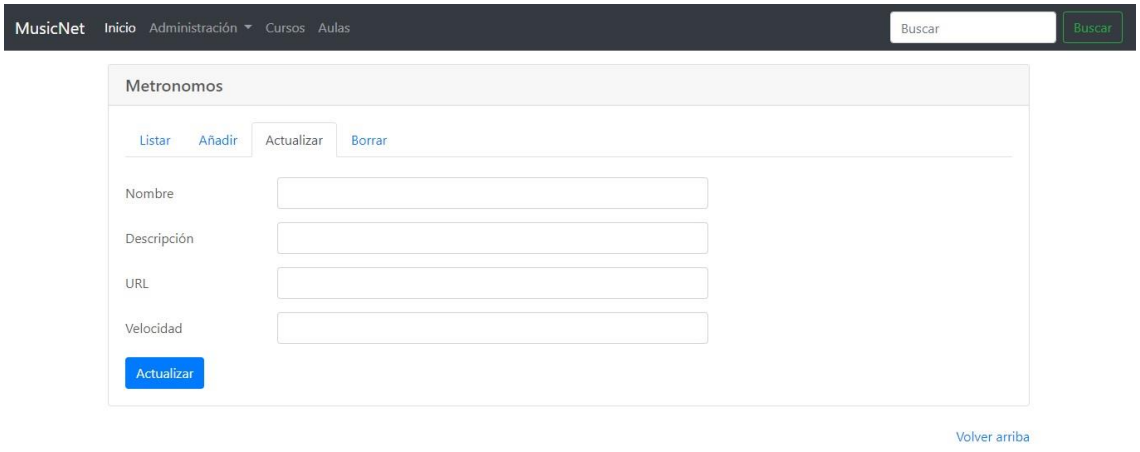


Figura 38: modificar metrónomo

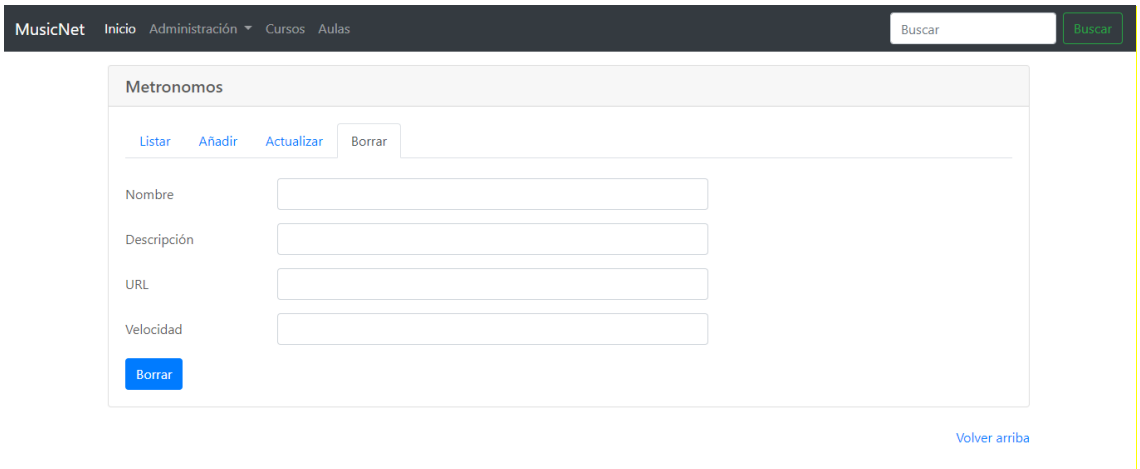


Figura 39: borrar metrónomo

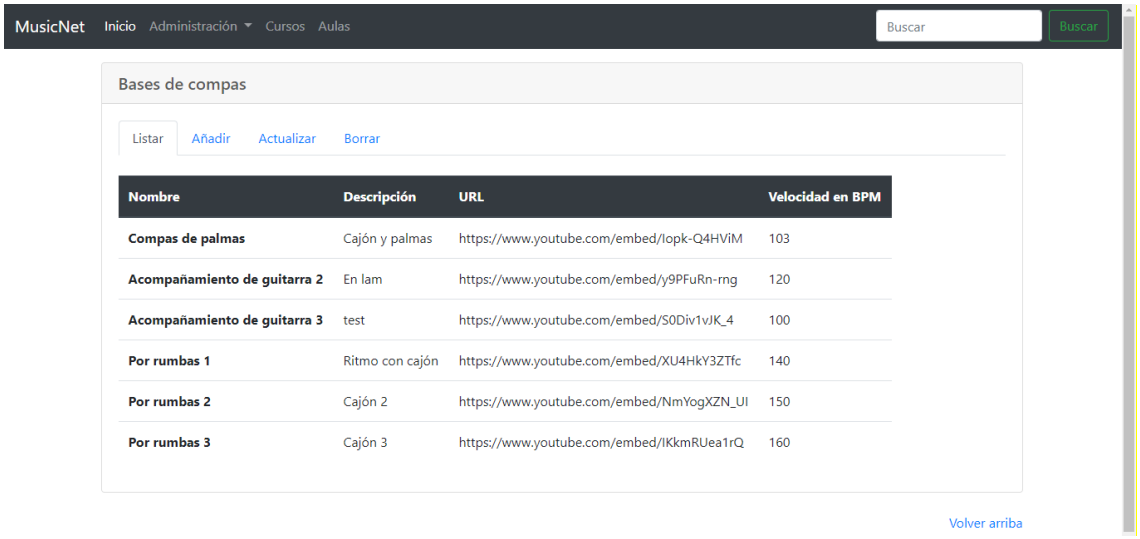


Figura 40: listado de bases de compas

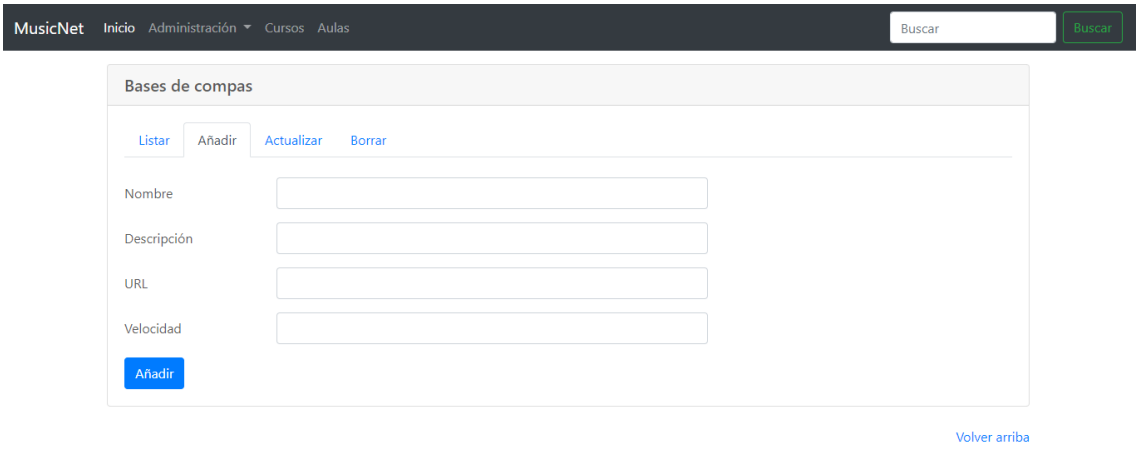


Figura 41: actualizar base de compás

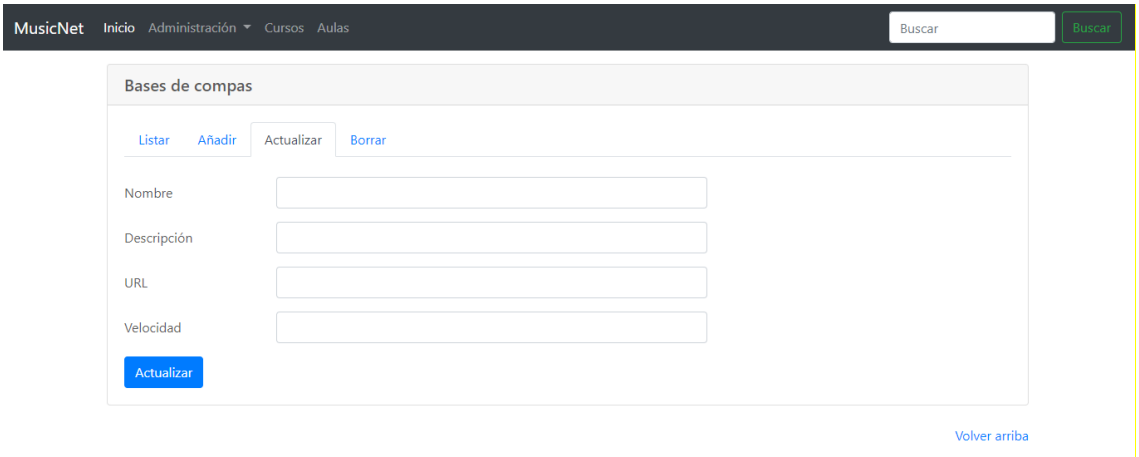


Figura 42: modificar base de compás

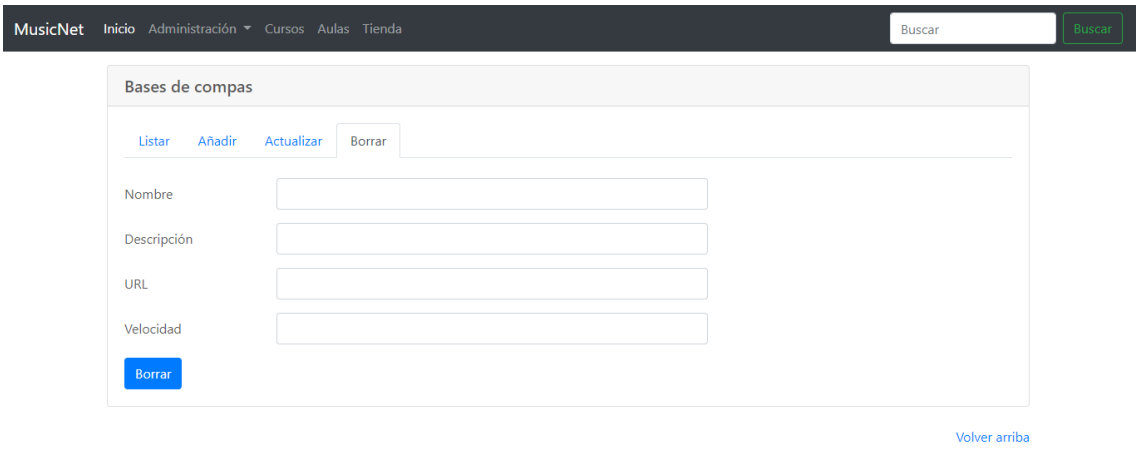


Figura 43: borrar base de compás

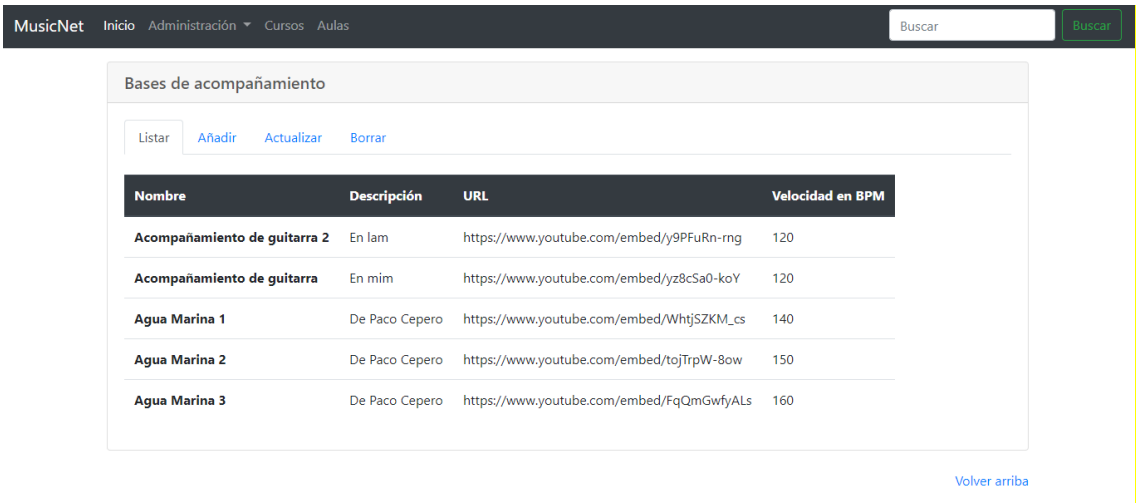


Figura 44: listar bases de acompañamiento

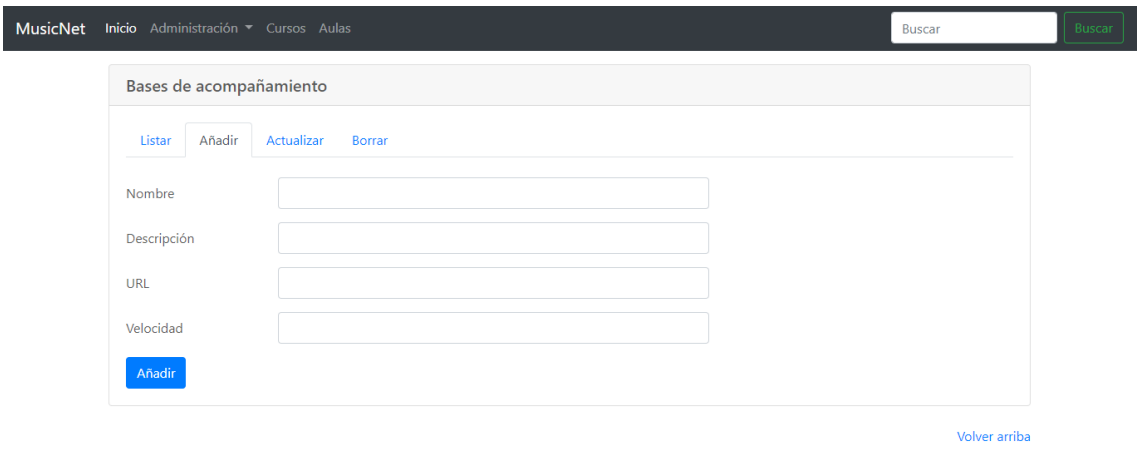


Figura 45: añadir base de acompañamiento

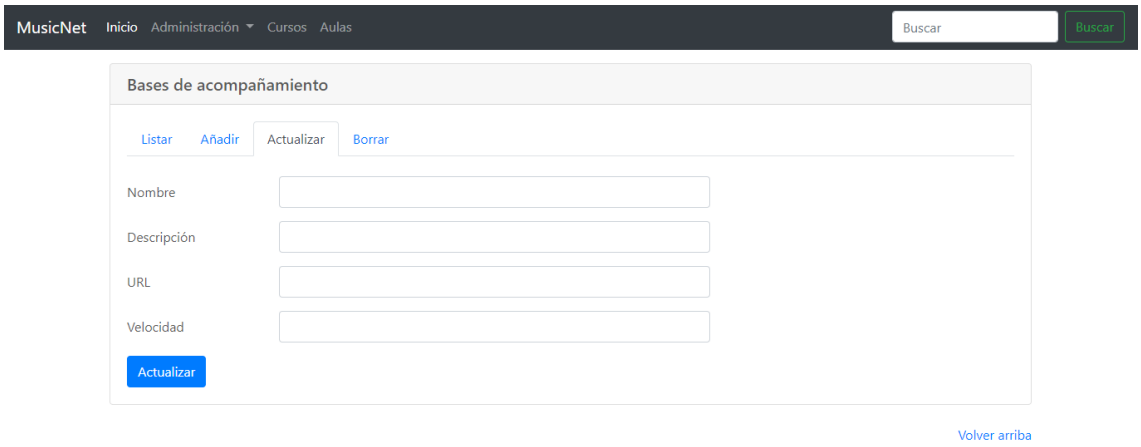


Figura 46: modificar base de acompañamiento

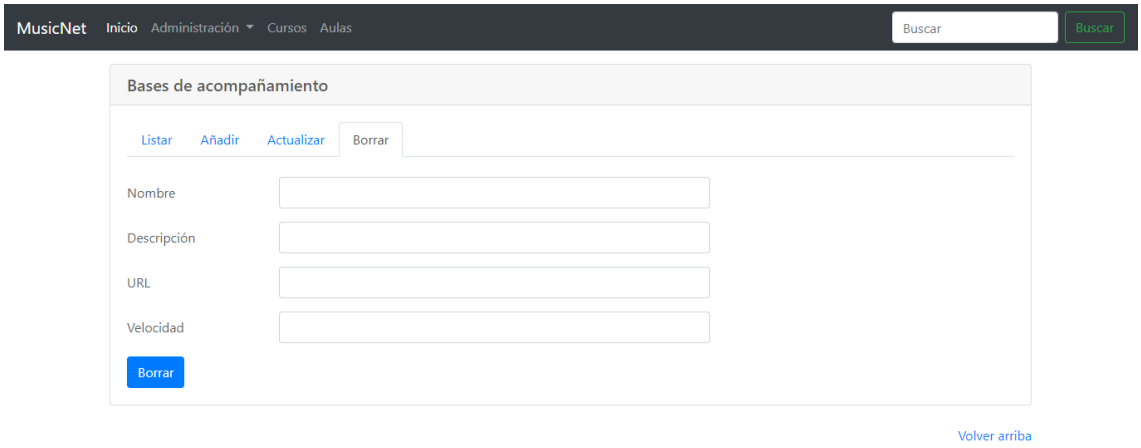


Figura 47: borrar base de acompañamiento

Vista cursos: contiene la gestión de cursos musicales. Al igual que el menú administración, éste esta dividido en cuatro pestañas des de las cuales se podrán listar los cursos, abrir uno de ellos, añadir el curso con sus respectivos videos y herramientas, actualizarlos y borrarlos.

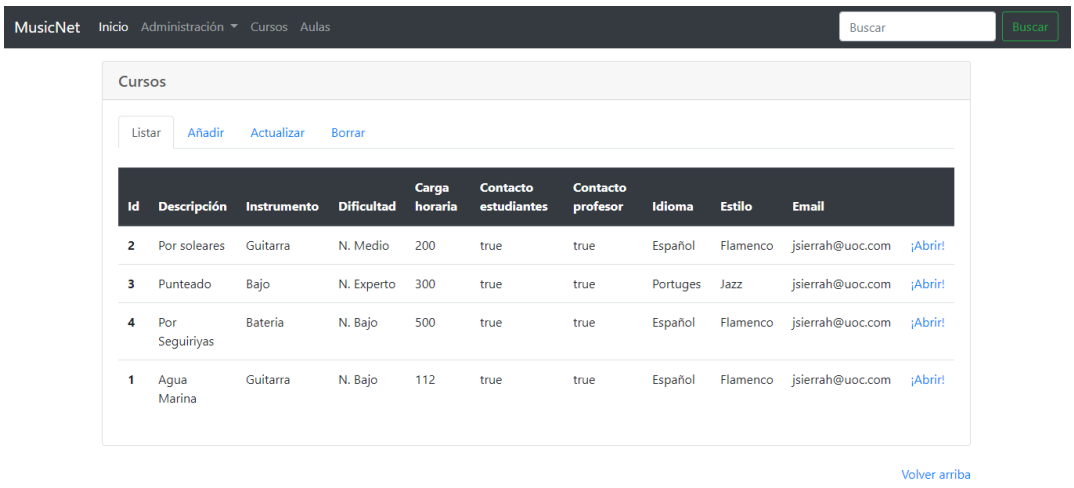


Figura 48: lista de cursos

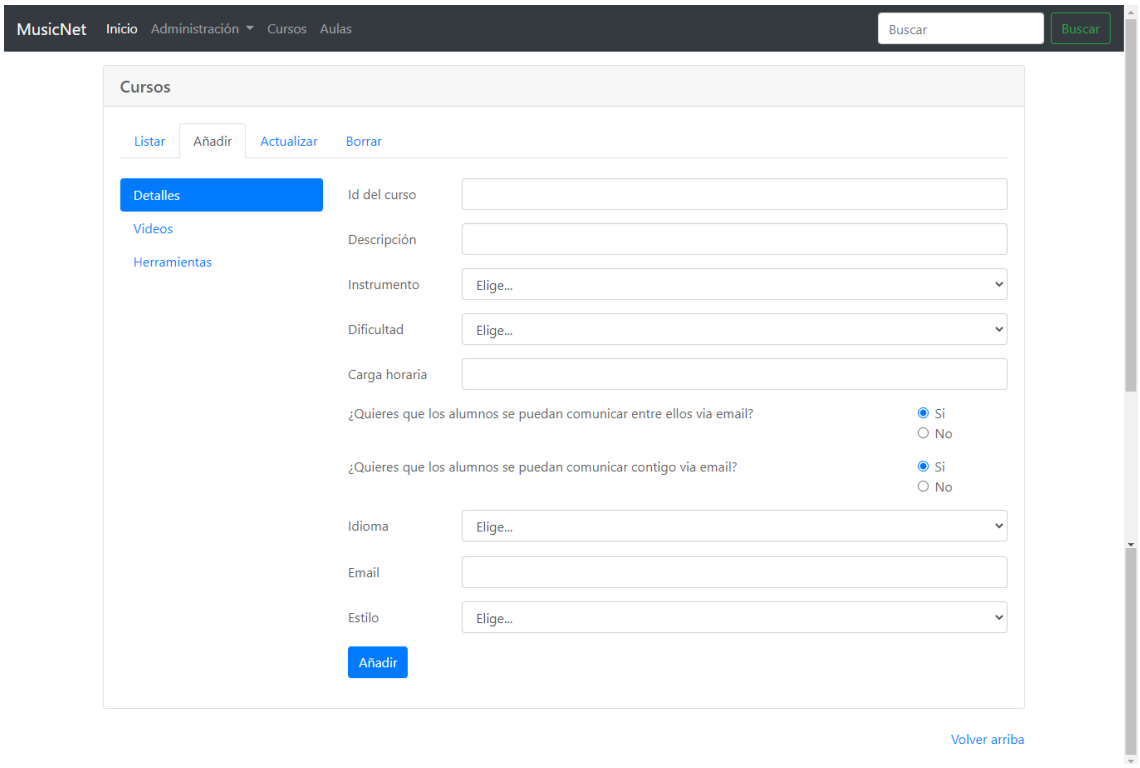


Figura 49: añadir detalles del curso

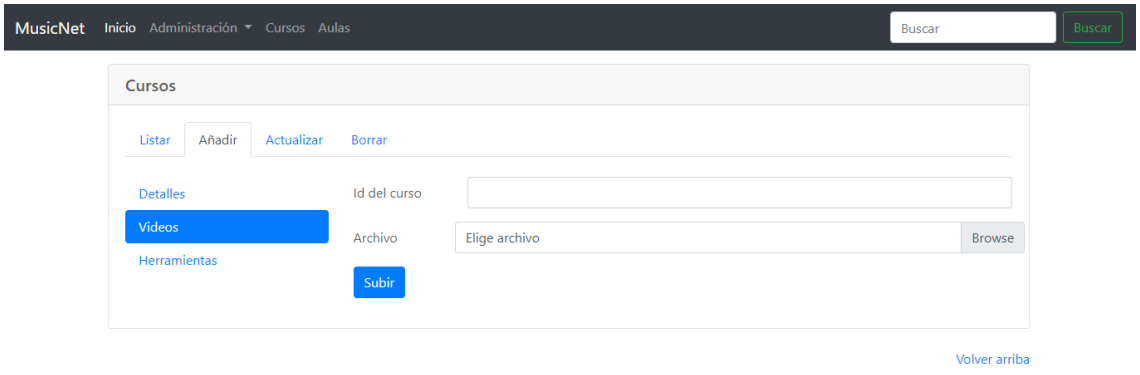


Figura 50: añadir video a un curso

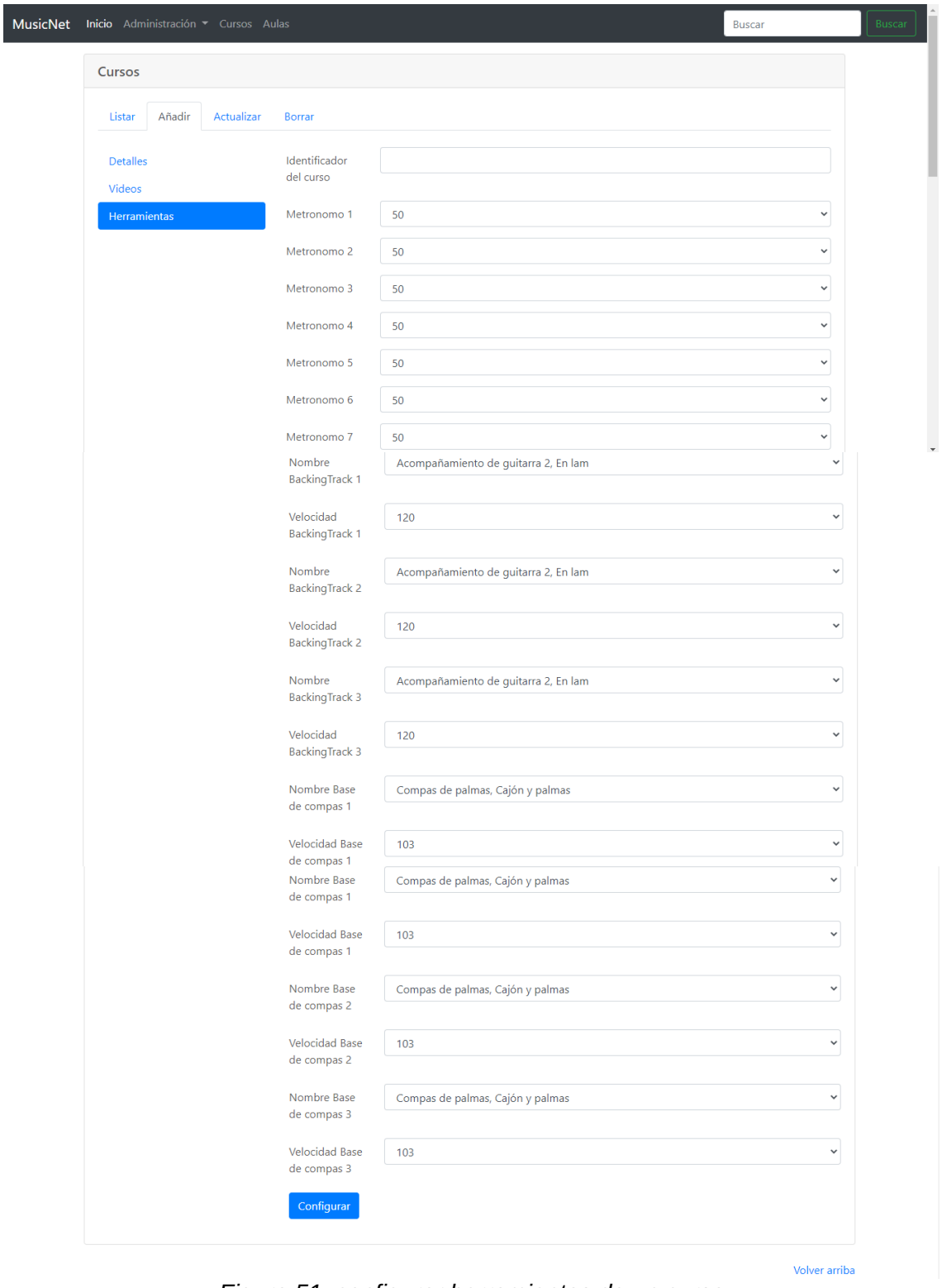


Figura 51: configurar herramientas de un curso

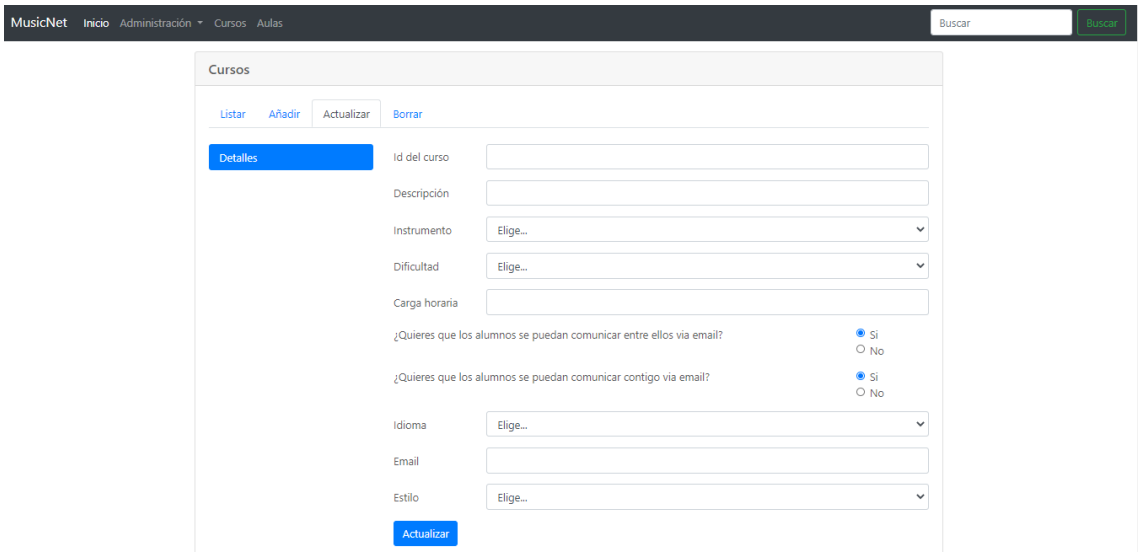


Figura 52: actualizar detalles del curso

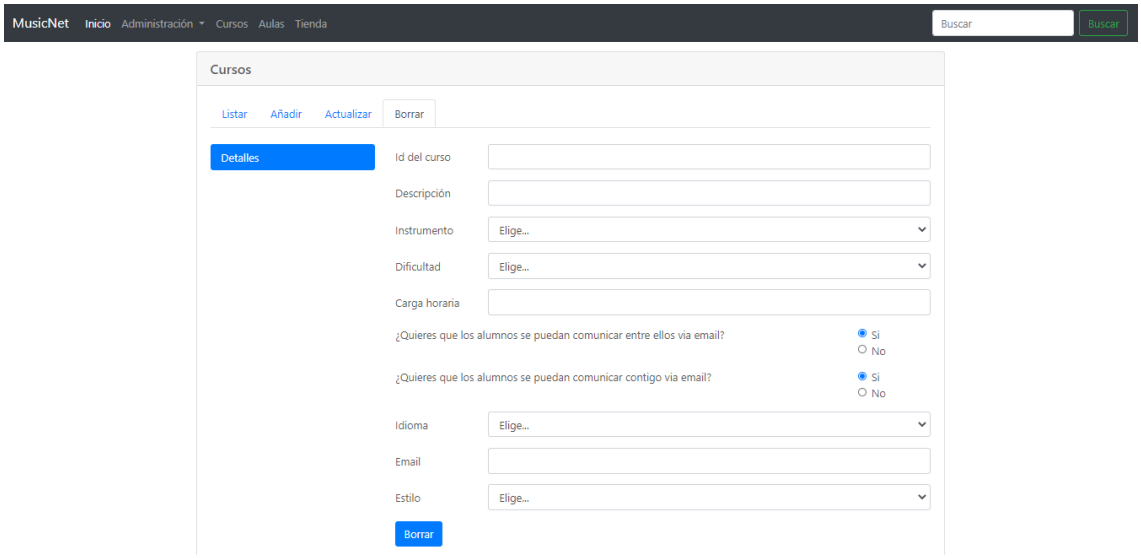


Figura 53: borrar detalles del curso

En la vista detallada de cada curso se puede encontrar los videos asignados al curso, así como las herramientas (metrónomos, bases de compas y bases de acompañamiento). Todas ellas están alojadas en Youtube, MusicNet inserta estos videos en el código HTML.

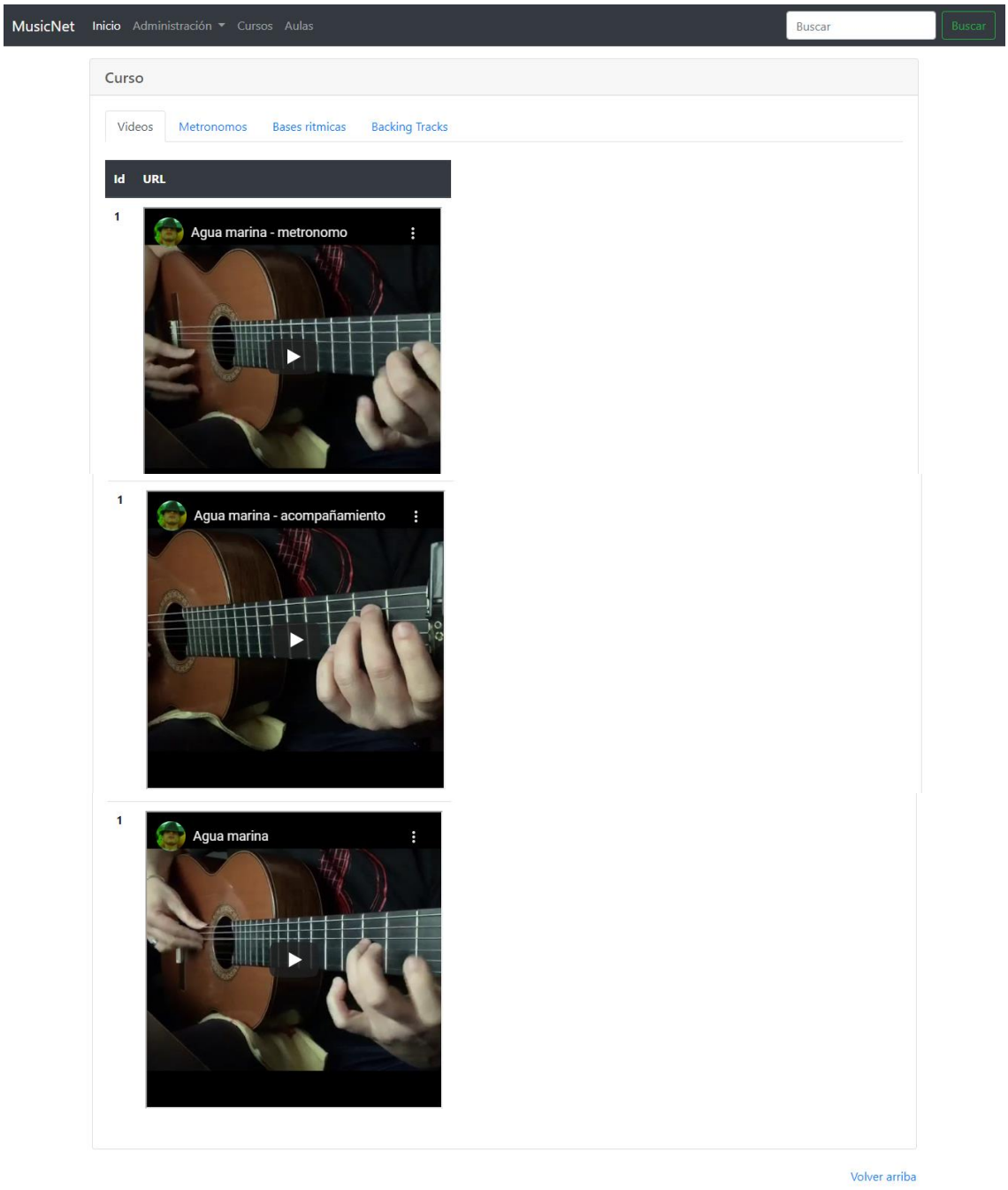


Figura 54: Ejemplo de videos de un curso

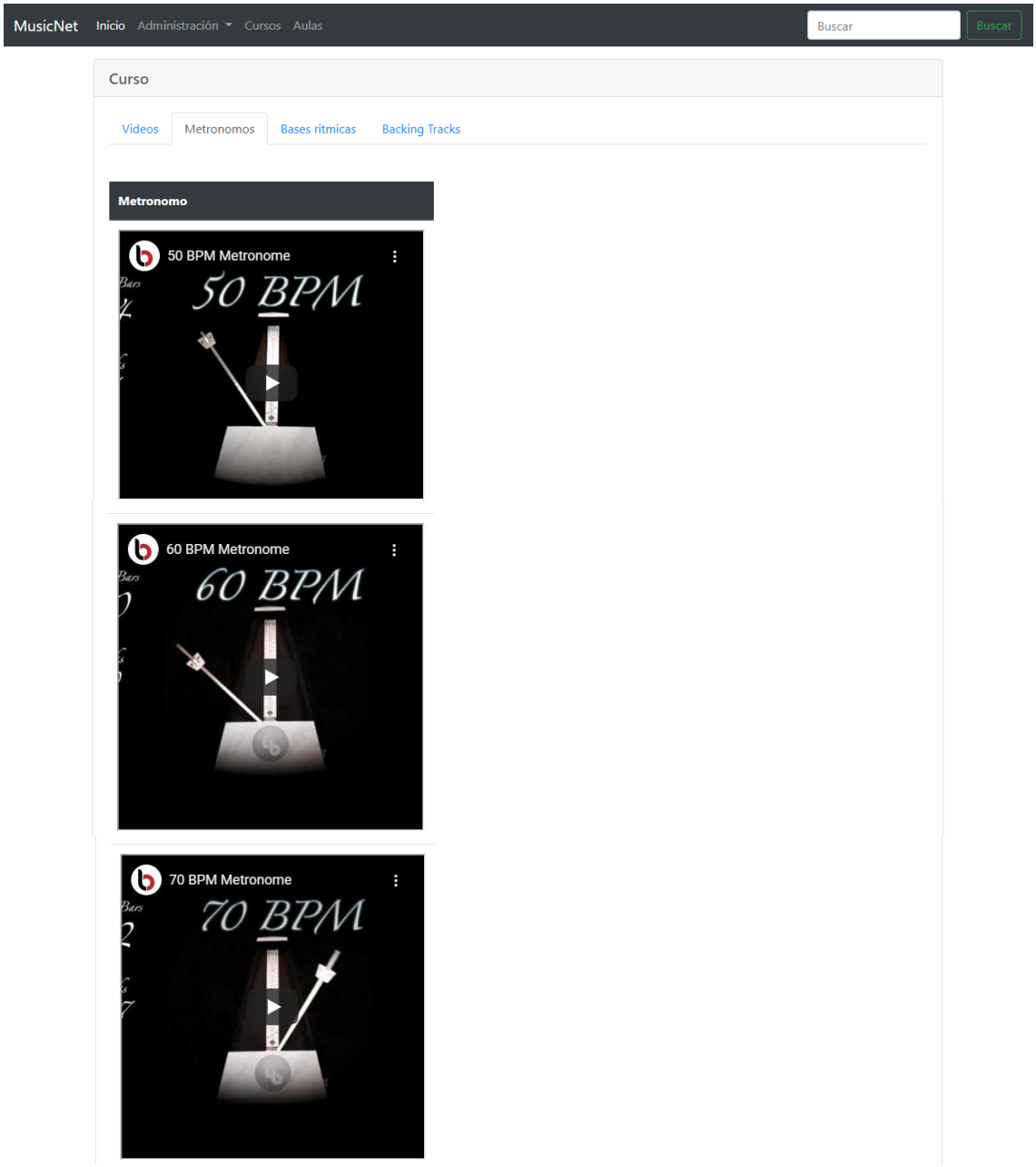


Figura 55: Ejemplo de metrónomos de un curso

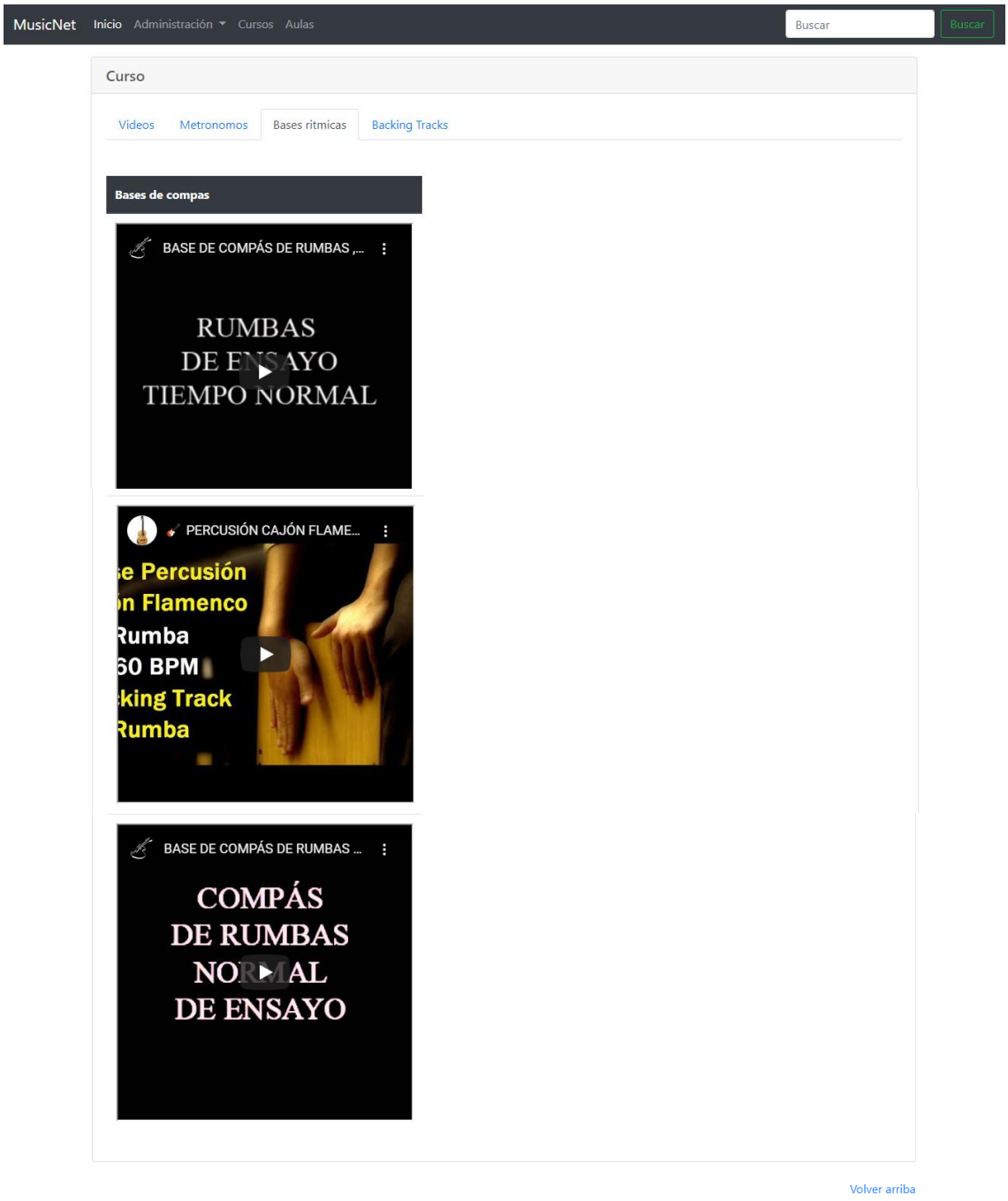


Figura 56: Ejemplo de bases de compas de un curso

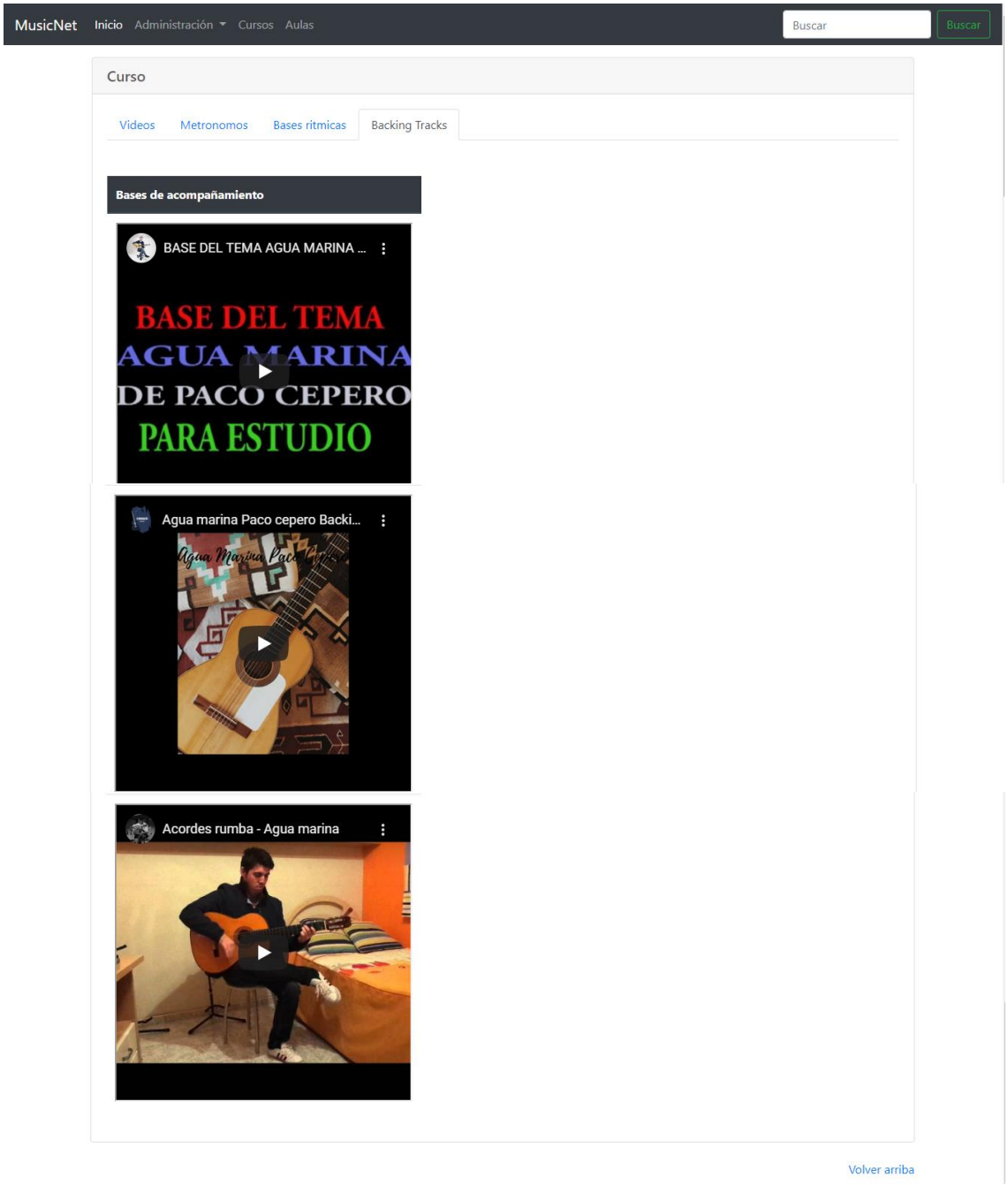


Figura 57: Ejemplo de bases de acompañamiento de un curso

Vista aulas: Des del menú de navegación se puede acceder al módulo de clases el cual permite la comunicación con otras personas. En este caso, al acceder se solicitará el identificador del aula y la contraseña (en el caso de ser el profesor se accederá como host y en el caso de ser alumno como asistente o ayudante):

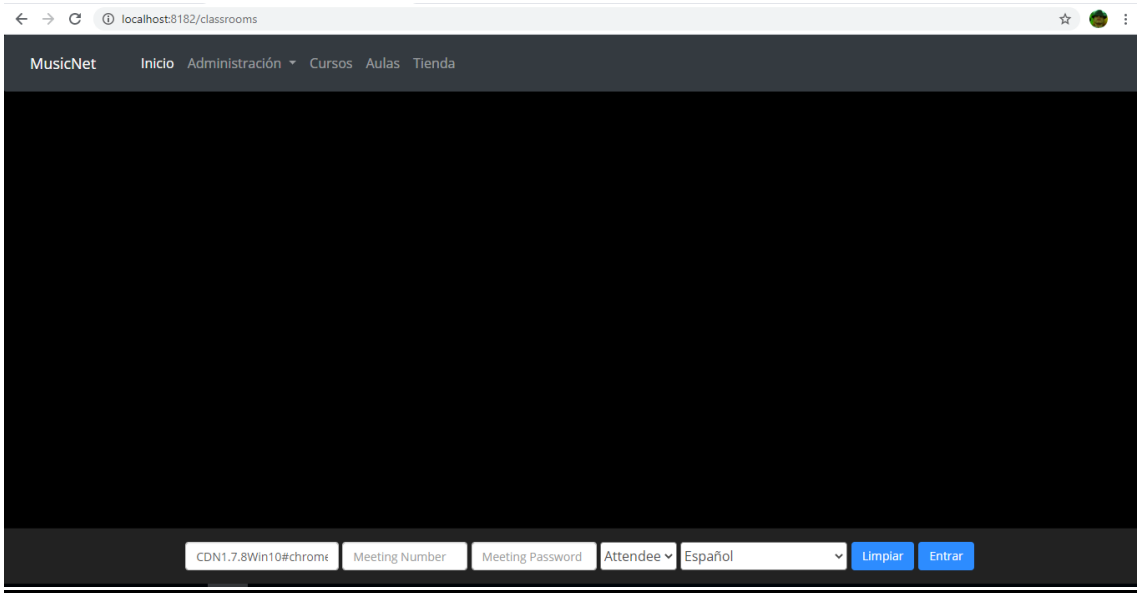


Figura 58: Sistema de videoconferencia

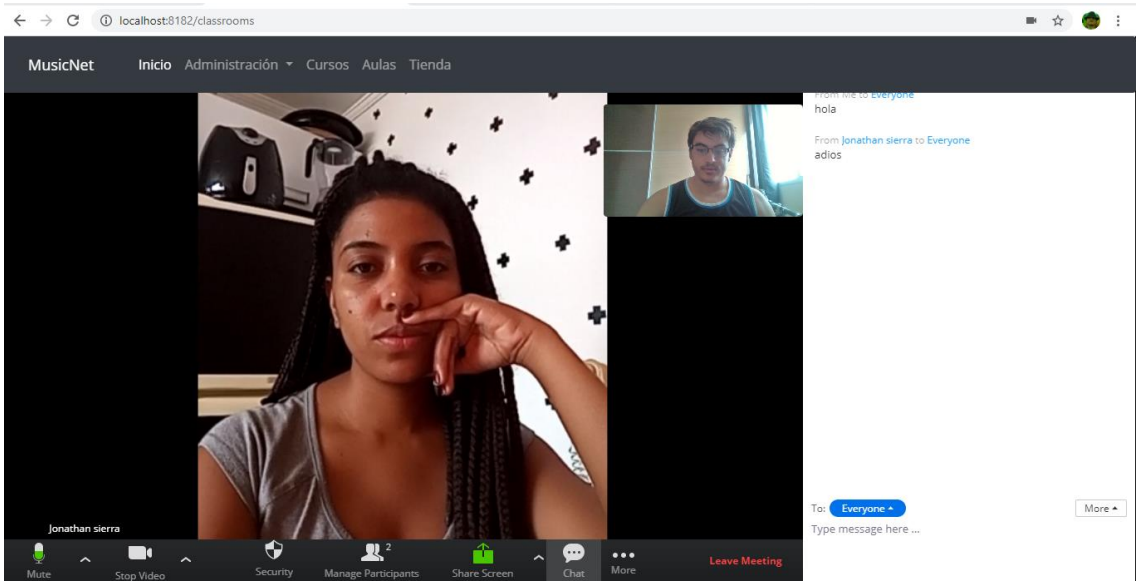


Figura 59: Ejemplo de videoconferencia

Vista tienda: Se obtiene un listado de todos los cursos con sus respectivos detalles. Des de esa misma vista, se puede comprar cualquiera de ellos introduciendo los datos de la tarjeta de crédito.

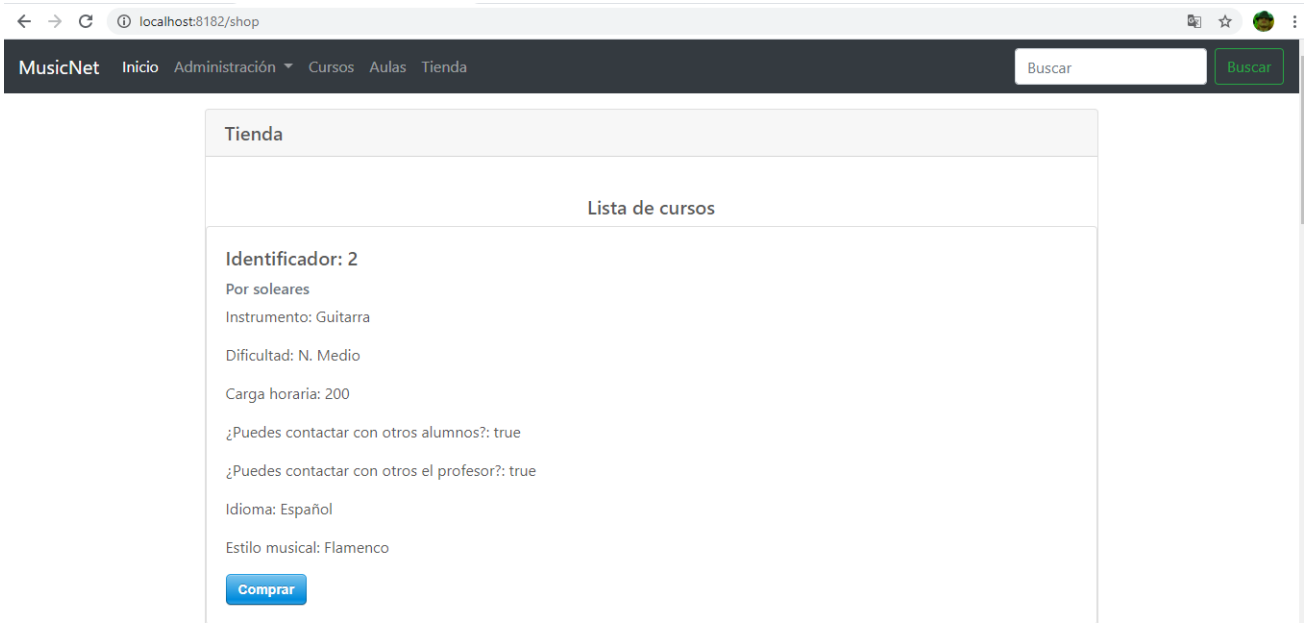


Figura 60: Venta de cursos

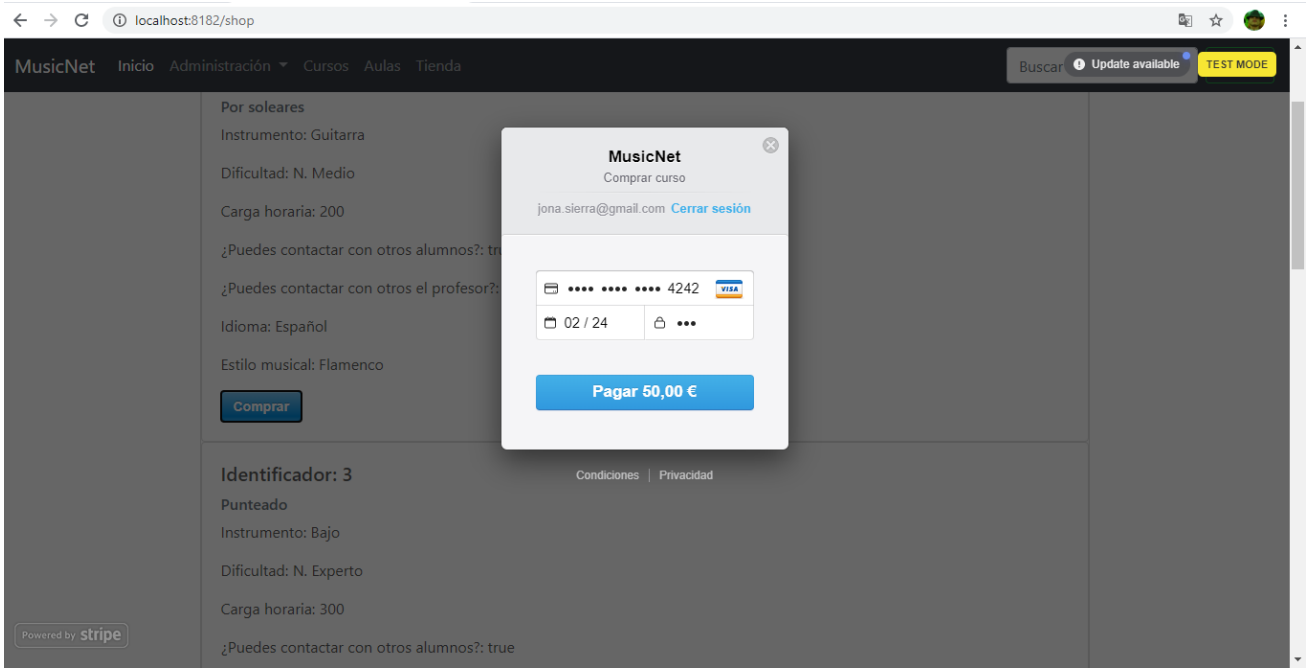


Figura 61: Venta de un curso

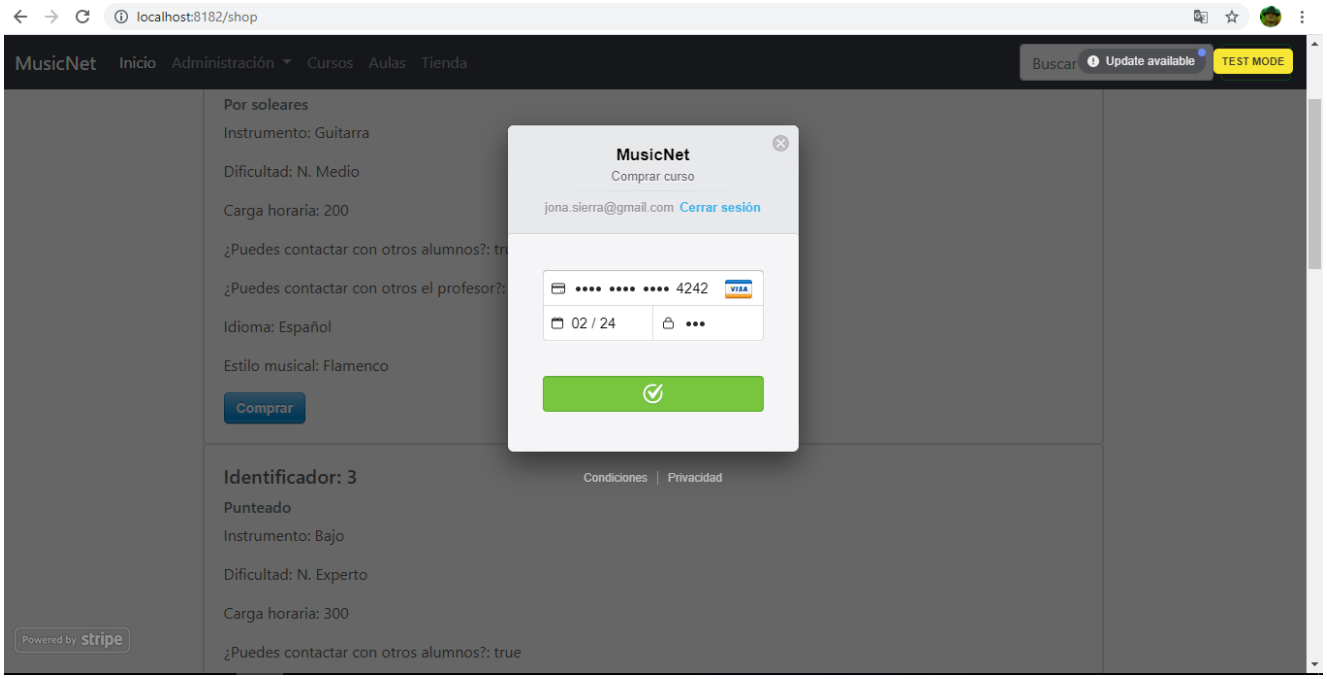


Figura 62: Compra de un curso

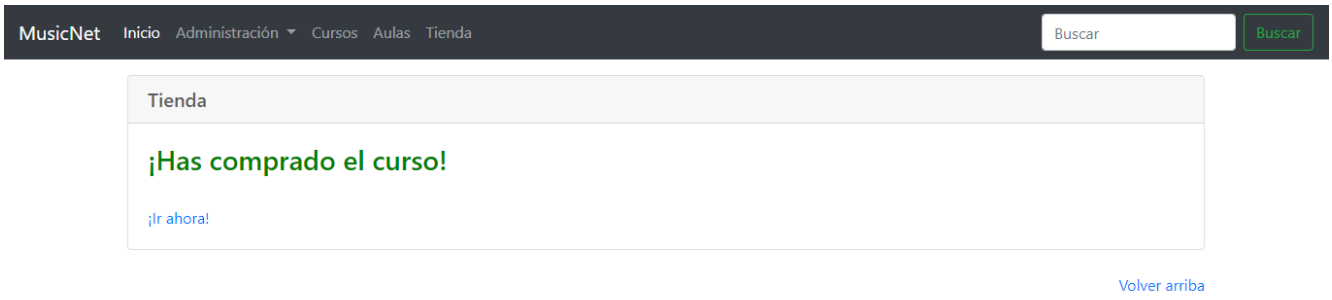


Figura 63: Ejemplo de compra

Recordar que todas las vistas del sistema son adaptables, es decir, todas y cada una de las vistas del sistema se presentaran de una forma u otra dependiendo del tamaño de la pantalla que se esté usando en el momento de acceder a ella. Por ejemplo, el menú de navegación para dispositivos con pantallas pequeñas (teléfonos móviles) será de la siguiente forma:

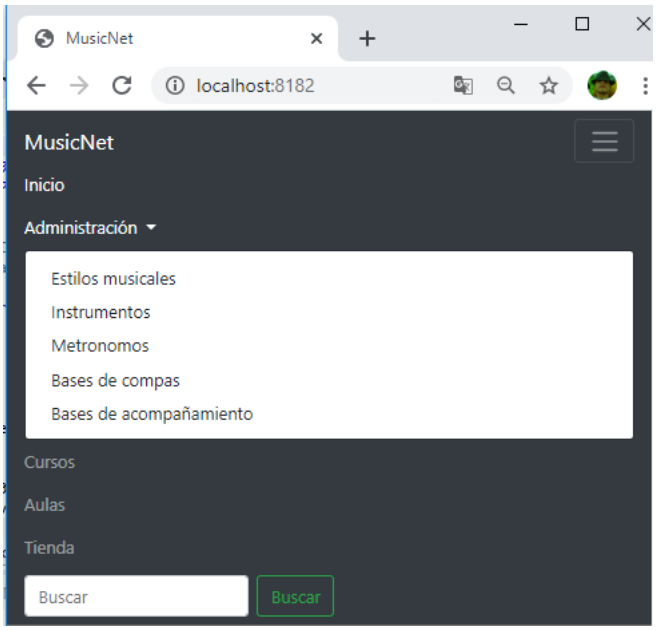


Figura 64: Menú de navegación para pantallas pequeñas

Así para todas las vistas, por ejemplo, la página de cursos queda de la siguiente forma para dispositivos con pantalla pequeña (tabla adaptable):

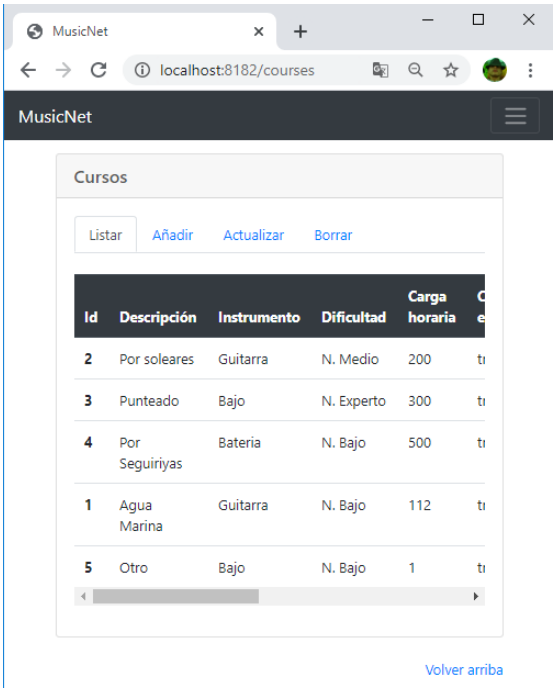


Figura 65: página cursos para pantallas pequeñas

Api de MusicNet

Para cada recurso se mostrará la ruta para acceder al recurso, los métodos que soporta, que tipo de solicitud HTTP es necesario usar, los parámetros que admite y una descripción (para entender exactamente los parámetros, ver la especificación del modelo).

- Recurso raiz

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/	Ningúno	Get	Ningúno	Devuelve la vista inicial de MusicNet

- Recurso courses (cursos didácticos)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/courses	findAll	Get	Ningúno	Devuelve una vista con la lista de cursos
/courses	save	Post	Objeto Course	Devuelve una vista con la lista de cursos después de crear un curso
/courses	save	Put	Objeto Course	Devuelve una vista con la lista de cursos después de actualizar un curso
/courses	delete	Delete	Objeto Course	Devuelve una vista con la lista de cursos después de borrar un curso
/courses/tools	save	Post	Objeto Tools	Devuelve una vista después de asignar herramientas para un curso

- Recurso musicalStyle (estilos musicales)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**

/musicalstyle	findAll	Get	Ningúno	Devuelve una vista con la lista de estilos musicales
/musicalstyle	save	Post	Objeto Style	Devuelve una vista con la lista de estilos musicales después de crear un estilo musical
/musicalstyle	save	Put	Objeto Style	Devuelve una vista con la lista de estilos musicales después de actualizar un estilo musical
/musicalstyle	delete	Delete	Objeto Style	Devuelve una vista con la lista de estilos musicales después de borrar un estilo musical

- Recurso instruments (instrumentos musicales)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/instruments	findAll	Get	Ningúno	Devuelve una vista con la lista de instrumentos musicales
/instruments	save	Post	Objeto Instruments	Devuelve una vista con la lista de instrumentos musicales después de crear un instrumento
/instruments	save	Put	Objeto Instruments	Devuelve una vista con la lista de instrumentos musicales después de actualizar un

				instrumento
/instruments	delete	Delete	Objeto Instruments	Devuelve una vista con la lista de instrumentos musicales después de borrar un instrumento

- Recurso metronoms (metrónomos musicales)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/metronoms	findAll	Get	Ningúno	Devuelve una vista con la lista de metrónomos
/metronoms	save	Post	Objeto Metronom	Devuelve una vista con la lista de metrónomos después de crear un nuevo metrónomo
/metronoms	save	Put	Objeto Metronom	Devuelve una vista con la lista de metrónomos después de modificar un metrónomo
/metronoms	delete	Delete	Objeto Metronom	Devuelve una vista con la lista de metrónomos después de borrar un metrónomo

- Recurso compastrack (bases de compas)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/compastracks	findAll	Get	Ningúno	Devuelve una vista con la lista de bases de compas

/compastracks	save	Post	Objeto CompasTrack	Devuelve una vista con la lista de bases de compas después de crear una base de compas
/compastracks	save	Put	Objeto CompasTrack	Devuelve una vista con la lista de bases de compas después de modificar una base de compas
/compastracks	delete	Delete	Objeto CompasTrack	Devuelve una vista con la lista de bases de compas después de borrar una base de compas

- Recurso backingtracks (bases de acompañamiento)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/backingtracks	findAll	Get	Ningúno	Devuelve una vista con la lista de bases de acompañamiento
/backingtracks	save	Post	Objeto BackingTrack	Devuelve una vista con la lista de bases de acompañamiento después de crear una base de acompañamiento
/backingtracks	save	Put	Objeto BackingTrack	Devuelve una vista con la lista de bases de acompañamiento después de modificar una base de acompañamiento
/backingtracks	delete	Delete	Objeto	Devuelve una

			BackingTrack	vista con la lista de bases de acompañamiento después de borrar una base de acompañamiento

- Recurso files

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/files	store	Post	Multipart file y Objeto VideoMaterials	Sube el archivo a la plataforma Youtube, almacena el id del video y devuelve una vista con la lista de cursos

- Recurso classrooms (aulas)

Ruta	**Métodos**	**Solicitud HTTP**	**Parámetros**	**Descripción**
/shop	Ningúno	Get	Ningúno	Devuelve una vista con el contenido de todos los cursos que están a la venta
/charge	charge	Post	Identificador del curso y Objeto ChargeRequest	Devuelve una vista donde se puede ver si el curso ha sido comprado o a tenido algún error.

5. Conclusiones

El método utilizado para que los alumnos puedan aprender cualquier temática musical es un método comprobado científicamente y que, por lo tanto, podemos ayudar a todos los usuarios a sentirse realizados.

El haber diseñado la API de MusicNet consumible por otras aplicaciones y, además, integrar las APIs de Youtube, Zoom y Stripe en este proyectou, me ha demostrado que las arquitecturas RESTful son un tipo de arquitecturas escalables y fáciles de integrar.

Las lecciones aprendidas durante este proyecto han sido varias y se han dado durante distantes fases del mismo. Por ejemplo, al acabar me siento capaz de afrontar otros proyectos de desarrollo web con diferentes características que MusicNet y abordarlos con éxito (independientemente de la tecnología empleada). El haber utilizado varias tecnologías y patrones de diseño que hasta ahora nunca había usado, me hace pensar que estoy preparado para afrontar problemas nunca antes planteados.

El no ser un gurú en algún asunto implica que en el momento de buscar una solución a un problema específico se consuma más tiempo y recursos de lo normal. Por ejemplo, en las primeras fases del proyecto se intentó desarrollar un sistema de streaming propio para el sistema de video. En el momento de la implementación quedó comprobado que no era viable para este proyecto, forzando a volver a las fases de diseño para modificar los diagramas iniciales. En definitiva, al no conocer una tecnología en profundidad se debe intentar ser ágil entre las fases de diseño e implementación pues las fases de diseño han variado durante todo este proyecto.

No se han alcanzado todos los objetivos planteados inicialmente, estos son la creación de la plataforma MusicNet con soporte para varios idiomas, permitir que los usuarios sean registrados en la plataforma y subir o descargar partituras y tablaturas. Todos los objetivos no alcanzados se deben a la falta de tiempo, es decir, el semestre se divide en cuatro meses y de todos ellos la implementación ha ocupado 41 días. Un proyecto de estas características necesitaría de varios desarrolladores trabajando conjuntamente para poder alcanzar todos los objetivos en ese periodo.

El seguimiento de la planificación y la metodología a lo largo del producto ha sido satisfactorio. Se ha seguido toda la planificación como se definió durante los primeros días y, además, se ha podido comprobar que los periodos establecidos para cada tarea se han ajustado a las expectativas iniciales. Se ha de mencionar que he sufrido una enfermedad tropical llamada Dengue que me ha desajustado 10 días de la planificación inicial durante las tareas de diseño y implantación. Para solucionar ese desajuste, dedique más horas de lo normal en los siguientes días después de superar la enfermedad.

Durante las siguientes líneas de trabajo, se alcanzarán todos aquellos objetivos no alcanzados. El orden a seguir para no tener problemas de dependencias debería ser el siguiente: agregar tablaturas y partituras, registro de usuarios y soporte para distintos idiomas tanto en la plataforma como en los cursos.

6. Bibliografia

[1] Web: https://www.oracle.com/ Data: 20/02/2020
[2] Web: https://www.eclipse.org/ Data: 20/02/2020
[3] Web: https://github.com/ Data: 20/02/2020
[4] Web: https://spring.io/ Data: 27/02/2020
[5] Web: http://hibernate.org/ Data: 27/02/2020
[6] Web: https://getbootstrap.com/ Data: 16/05/2020
[7] Web: https://ca.wikipedia.org/wiki/HTML_5 Data: 10/05/2020
[8] Web: https://en.wikipedia.org/wiki/CCS3 Data: 10/05/2020
[9] Web: https://www.javascript.com/ Data: 31/03/2020
[10] Web: https://pt.wikipedia.org/wiki/REST Data: 20/02/2020
[11] Web: https://www.thymeleaf.org/ Data: 01/04/2020
[12] Web: https://www.postgresql.org/ Data: 27/02/2020
[13] Web: https://developers.google.com/youtube/v3 Data: 16/05/2020
[14] Web: https://www.zoom.us/ Data: 06/06/2020
[15] Web: https://stripe.com/ Data: 31/02/2020
[16] Autor: Jordi Casas Roma, Titulo: Diseño conceptual de la base de datos, Fuente: FUOC, Data: 05/04/2020
[17] Autor: Xavier Burgués Illa, Titulo: Diseño lógico de la base de datos, Fuente: FUOC, Data: 05/04/2020
[18] Autores: Blai Cabré i Segarra, Jordi Casas Roma, Dolors Costal Costa, Pere Juanola Juanola, Àngels Rius Gavidia y Ramón Segret i Sala; Libro: Diseño físico de la base de datos, Fuente: Fuoc, Data: 25/04/2020
[19] Autores: Joan Anton Pérez Braña y Santiago Ortega Carazo, Libro: Procesamiento de consultas y vistas, Fuente: FUOC, Data: 25/04/2020
[20] Autor: Vicenç Font i Sagristà, Libro: Caso practico de estudio. Diseño, Fuente: FUOC, Data: 06/04/2020

7. Annexo

7.1 Manual de Instalación

7.1.1 Java JDK

Primero es necesario descargarse el software des de https://www.oracle.com/java/technologies/javase-downloads.html Para este caso en concreto, hemos utilizado ***Java Platform, S.E.: Java SE 8u221: Download JDK***, la versión de 64 bits para Windows.

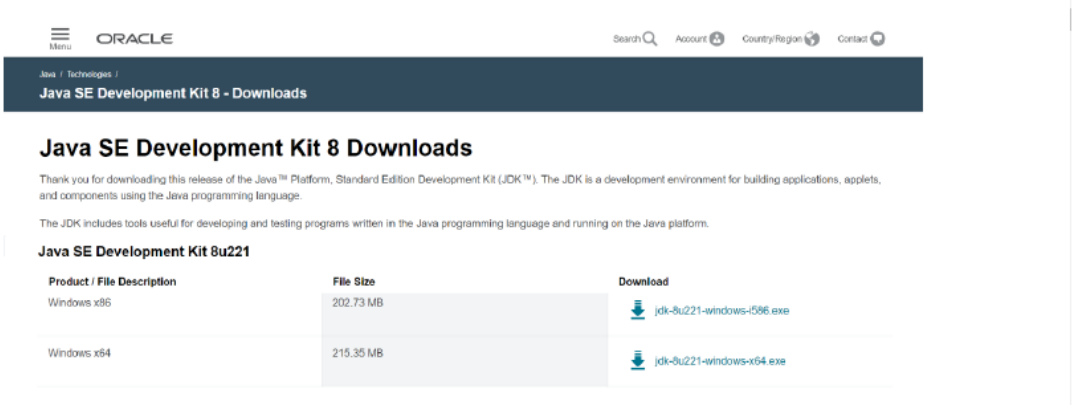


Figura 66: Descarga JDK

Para instalarlo, se da doble clic sobre "*jdk-8u221-windows-i586.exe*". Luego aceptamos la licencia, en la pantalla "Custom Setup" no modificamos nada y aceptamos. Finalmente pulamos "Finish" y reiniciamos el ordenador tal y como el instalador necesita.

A continuación, se definirán las variables de entorno, las cuales se definen como variables de sistema o de usuario (todas las variables de este entorno deberán ser definidas del mismo tipo). Para ello:

1) Pulsar sobre "Mi PC" y seleccionar propiedades.
2) Seleccionar la pestaña "Opciones avanzadas" y seguidamente en "Variables de entorno".
3) Crear en variables del sistema la variable "JAVA_HOME". Como valor, se ha de poner el directorio donde esta Java, por ejemplo, C:\Program files (x86)\Java\jdk1.8.0_221

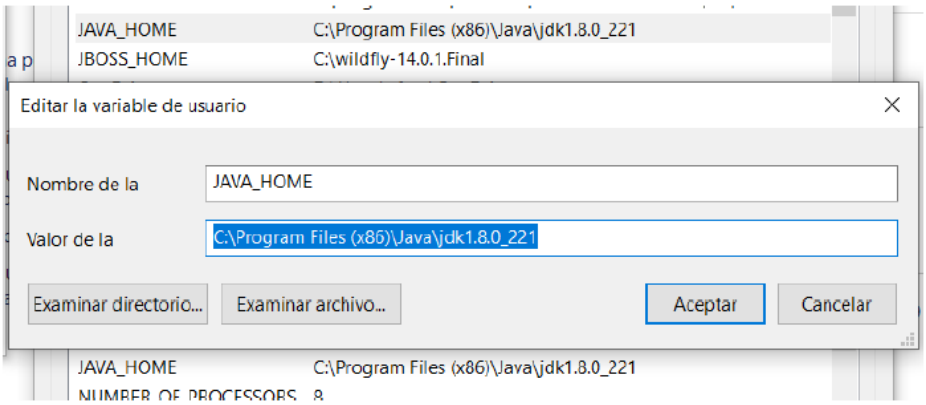


Figura 67: Variables de entorno

4) Buscar la variable "PATH" y modificarla, se deberá dejar el mismo valor que ya tiene pero añadiendo al final de ella *";%\JAVA_HOME%\BIN".*
5) Buscar la variable "CLASSPATH" en variables del sistema y de usuario (si no existen se han de crear). Se le ha de dar el valor **";%JAVA_HOME%\lib\tools.jar;."**
6) Por último, cerrar el cuadro de dialogo de variables de entorno, "Aceptar" y luego "Aplicar"

Para comprobar el correcto funcionamiento, sobre la consola del sistema escribir “Java”. Si se obtiene un mensaje parecido a la siguiente imagen, la instalación y configuración del JDK es correcta:

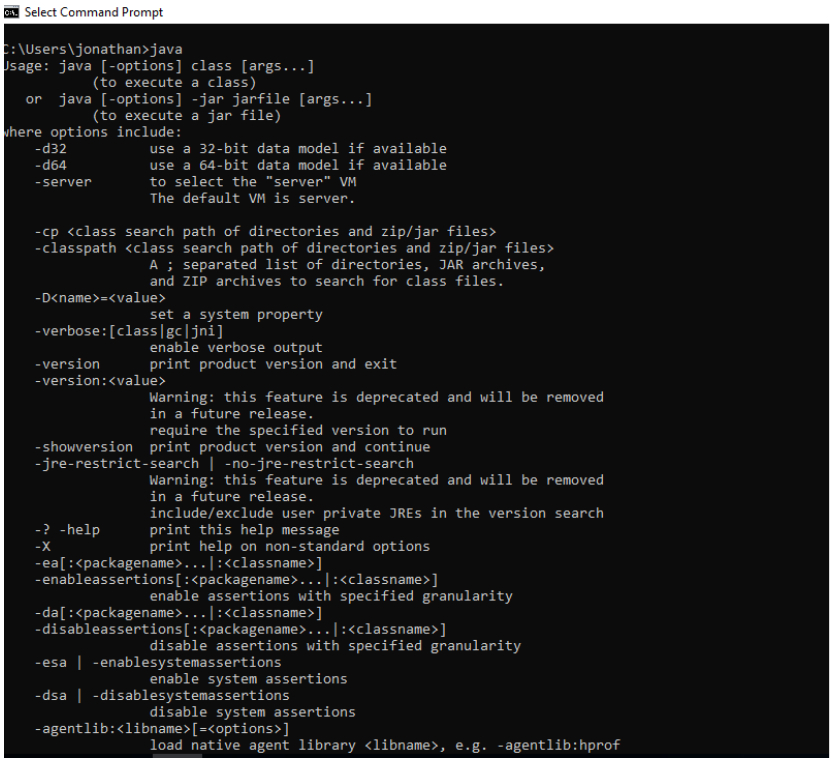


Figura 68: Ejecución de java

7.1.2 PostgreSQL

En la página oficial www.postgresql.org se encuentra el repositorio para descargar la versión necesaria del motor de la base de datos. En este caso, se ha usado la versión 10.10-1 para Windows de 64 bits.

Una vez descargado el ejecutable, hacer doble clic sobre él. En la siguiente interfaz, aceptar todas las opciones por defecto como muestra la siguiente imagen:

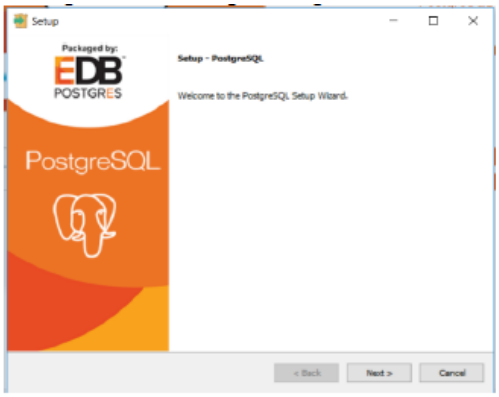


Figura 69: Instalación de PostgreSQL

Cuando finalice la instalación, se deberá configurar para poder utilizar la base de datos en nuestro proyecto. Existe una interfaz web para administración llamada pgAdmin4 (buscar ese mismo nombre en cortana). Para conectarse a la base de datos, se debe hacer como en la siguiente imagen:

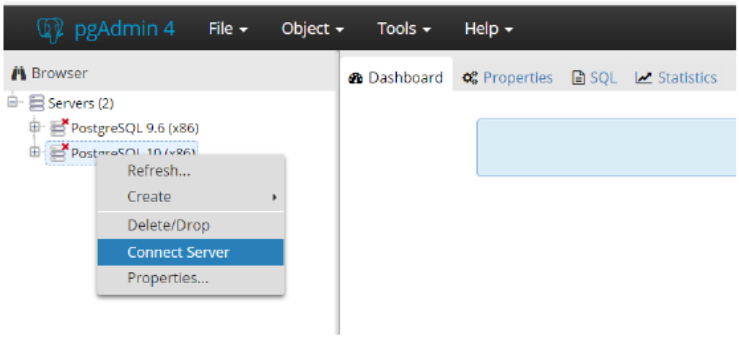


Figura 70: Conexión a PostgreSQL

Luego se crea un nuevo rol de entrada.

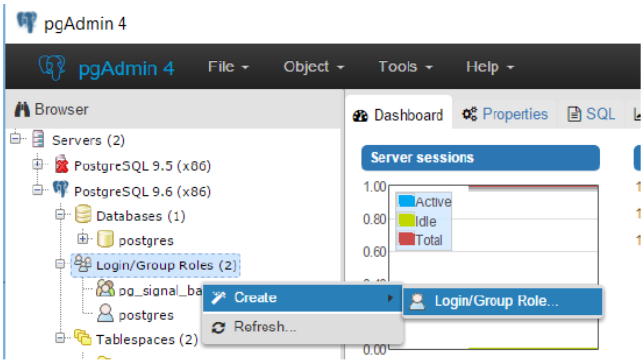


Figura 71: Crear rol

Se configura un nombre de rol y una contraseña (USER y PASSWORD en nuestro caso).

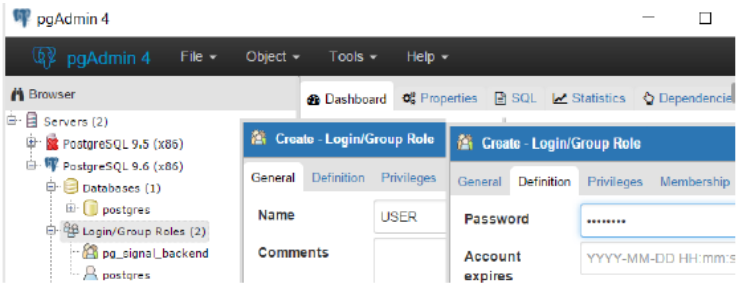


Figura 72: configurar usuario y contraseña

Se le dan los siguientes permisos:

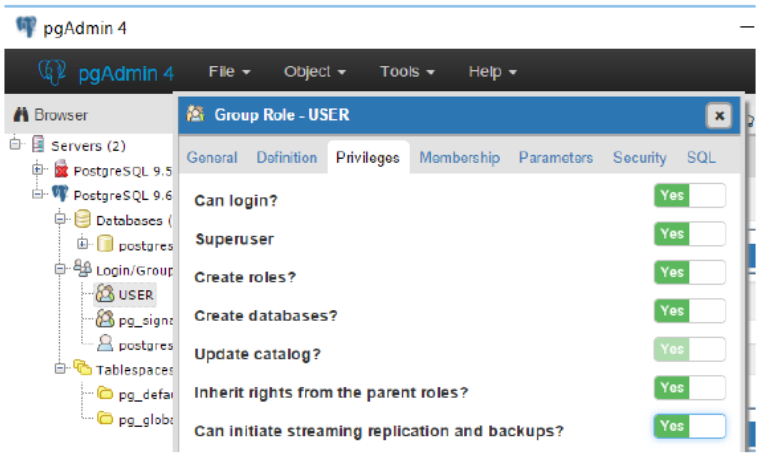


Figura 73: Asignar permisos

Y finalmente se crea un esquema, en este caso es llamado musicnet. Como owner se configura el usuario USER con privilegios ALL.

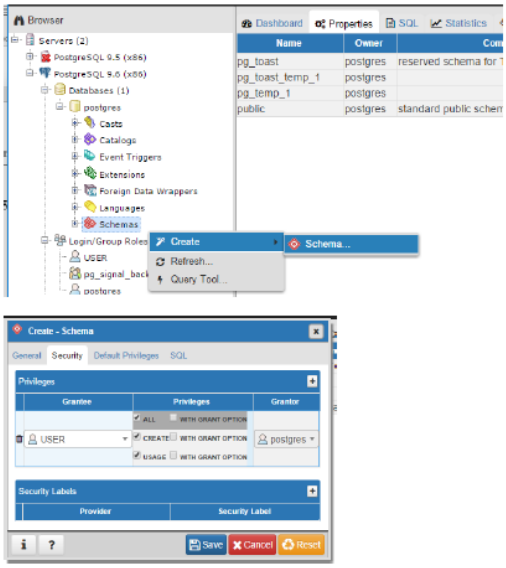


Figura 74: Crear esquema

7.1.3 Eclipse

Al igual que con otros aplicativos, primeramente, debemos descargar el software de la página oficial. En este caso se encuentra en www.eclipse.org/downloads. Se ha utilizado la versión 2020-03 (4.15.0) para Windows de 64 bits. Una vez descargado el instalador, seguir las instrucciones de instalación sin modificar ningún valor (aceptar la configuración por defecto) hasta realizar con éxito el proceso.

Al abrir Eclipse, se mostrará una ventana como la siguiente (Se recomienda dejar el workspace por defecto):

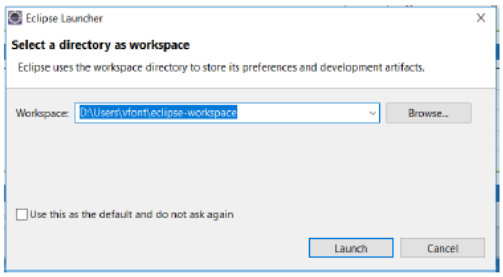


Figura 75: Ejemplo de workspace

7.1.4 GitHub

Para el repositorio no será necesario instalar este en local, tan solo se deberá acceder a www.github.com y registrarse en la plataforma.

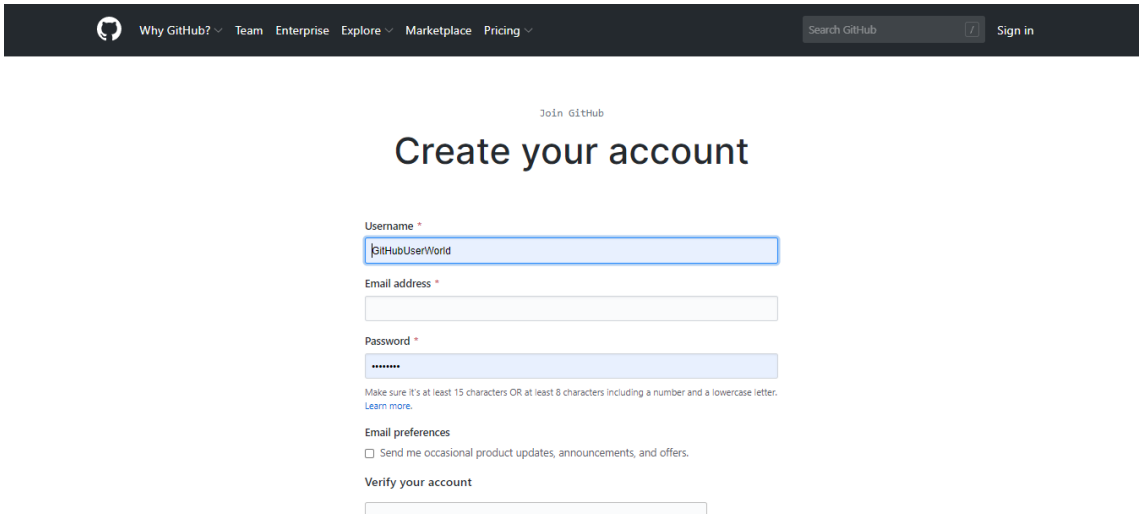


Figura 76: Crear cuenta GitHub

Después de introducir todos los datos que la plataforma, se recibirá un email para validar la creación de la cuenta. Después de validar la cuenta y acceder a ella, pulsamos en “New” para crear un repositorio:

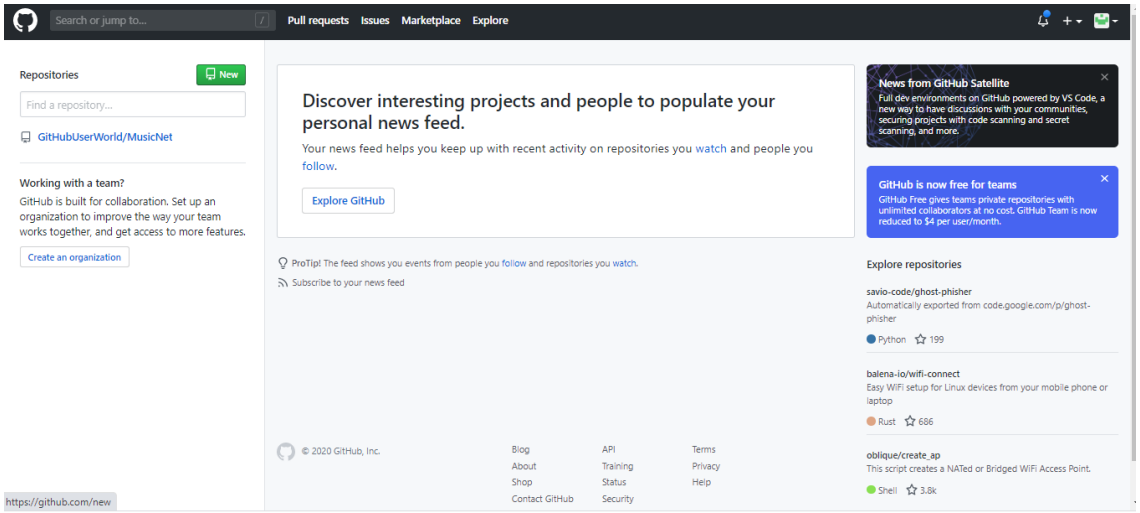


Figura 77: Ejemplo de interfaz GitHub

Luego se introduce el nombre, la descripción, se configura como repositorio público o privado y se inicializa con un archivo README.

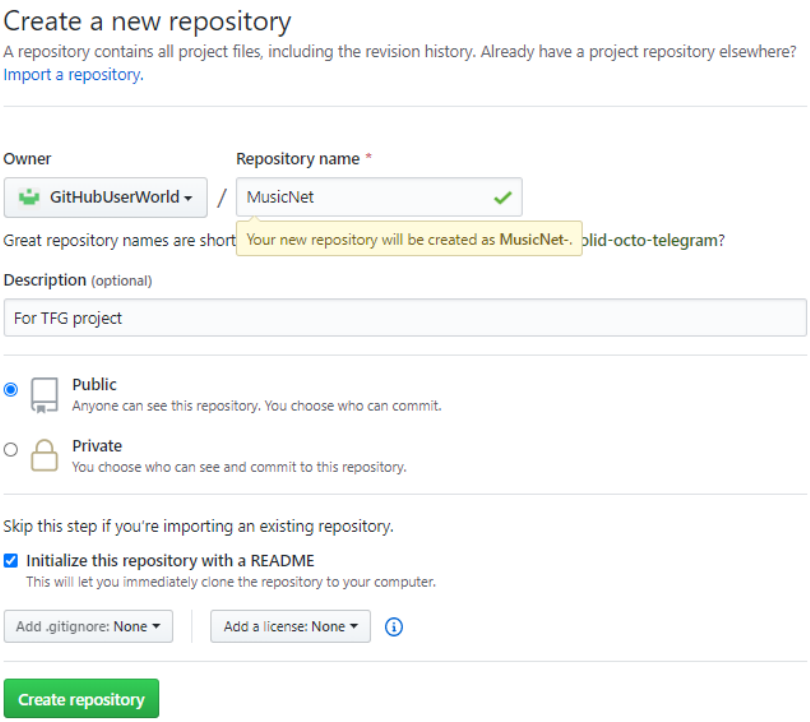


Figura 78: Crear repositorio

Después de pulsar “Create repository” el repositorio estará listo para usar.

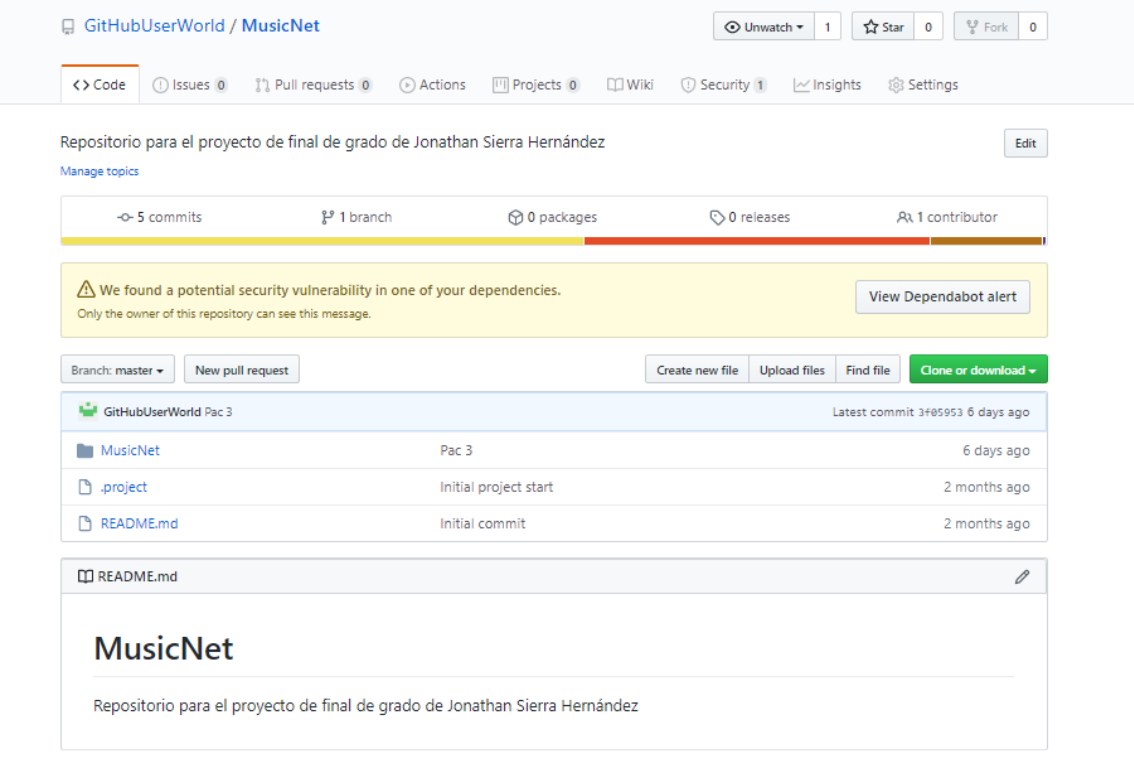


Figura 79: Ejemplo de repositorio

7.1.5 Spring

En este punto se tendrá todo lo necesario para crear la estructura base del proyecto. Para ello, se puede hacer des del plugin Spring Tools o des de la web https://start.spring.io/ En este caso, vamos a mostrar como hacerlo des de la pagina web de spring así que una vez dentro de ella, se seleccionan las siguientes opciones:

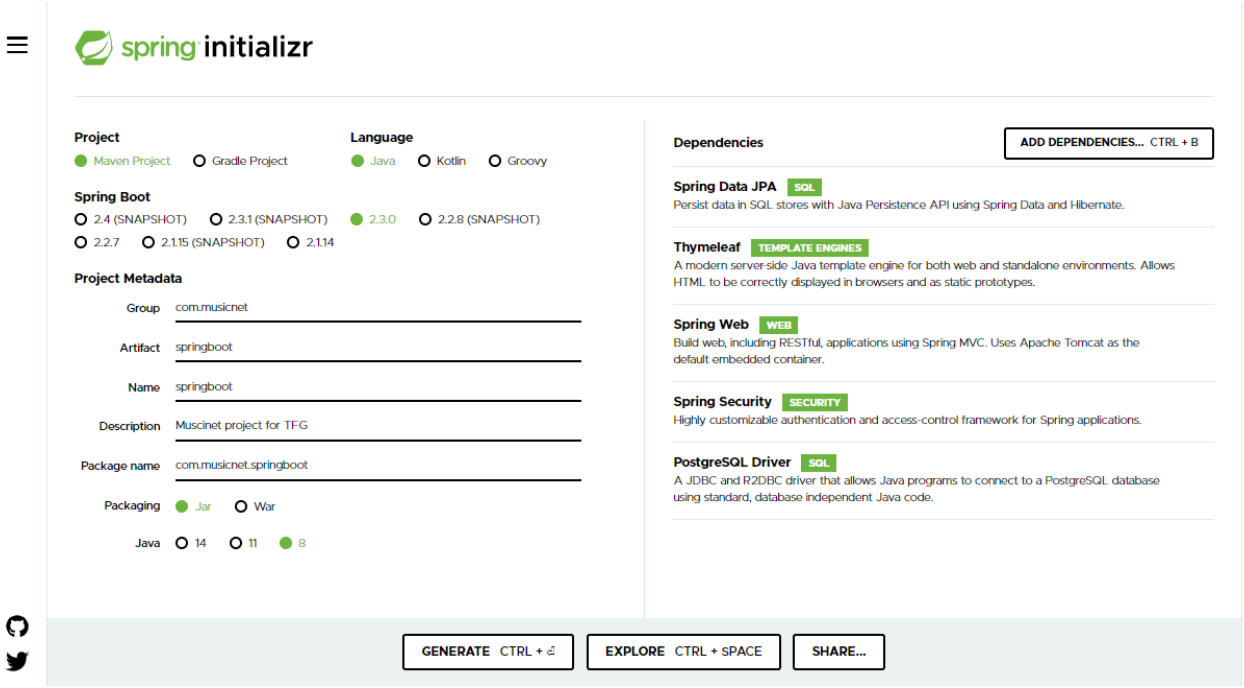


Figura 80: Interfaz de Start Spring

- Se utiliza Maven para gestionar este proyecto.
- Se utiliza Java como lenguaje de programación.
- Se elige la versión 2.3.0 de Spring
- Se define el grupo com.musicnet, artefacto springboot, nombre springboot, se añade una descripción y el nombre de paquete.
- Se escoge Jar como packaging.
- La versión de Java utiliza es la 8.
- Las dependencias utilizadas son las dependencias básicas de nuestro proyecto. Más adelante se añadirán otras (básicamente las necesarias para la integración con la API de Youtube). Se utiliza la dependencia PostgreSQL como driver para poder interactuar con la base de datos, Spring Data JPA permite utilizar la persistencia orientada a objetos en Java, Spring Web nos ofrecerá las librerías para el desarrollo de la API Rest de MusicNet. Spring Security definirá los permisos y accesos a las diferentes URIs y Thymeleaf permitirá el uso de las plantillas dinámicas.

Una vez generado y descargado el proyecto base, se debe descomprimir el .zip y seguidamente importar el directorio dentro de eclipse.

7.1.6 Boostrap

Primeramente, se deberá descargar Boostrap de la sección descargas del sitio web oficial (www.boostrap.com). Allí se encontrarán diferentes formas de usar/descargar el framework (descargar código fuente, gestionarlo des de npm, RubyGems, etc), en nuestro caso descargaremos los css y js compilados versión 4.5.0.

Dentro del zip descargado se encuentran dos directorios, uno llamado css y otro llamado js. Cada uno de ellos contiene a su vez varios archivos, pero para nuestro proyecto solo será necesario el archivo boostrap.css y boostrap.bundle.js.

Estos dos archivos son suficientes para trabajar con boostrap pero en MusicNet, además, se utiliza el css de Carousel. Para obtenerlo, se puede descargar en la sección “Examples” del sitio web oficial. Allí se encontrará el directorio Carousel el cual contiene el archivo carousel.css y index.html.

Finalmente, para poder implementar estos estilos deberemos poner el archivo index.html dentro del directorio /src/main/resources/static de Eclipse. Además, se añade los dos estilos de css dentro de /src/main/resources/static/css y el archivo javascript dentro de /src/main/resources/static/js.

Una vez descargado y instalado boostrap, se podrá acceder des de cualquier vista a los estilos css y funciones de javascript.

7.1.7 Youtube API

Google API y más concretamente Youtube API, nos ofrece un amplio abanico de posibilidades y funcionalidades muy potentes. En MusicNet básicamente se necesita una funcionalidad que implementa la subida de videos a la plataforma de Streaming Youtube.

El primer paso para el uso de Youtube API es crear nuestro proyecto en Google APIs y, además, registrar la aplicación que tiene intención de consumir la API. En https://console.developers.google.com/ se encuentra el menú de administración y des de “Nuevo proyecto” lo podremos crear introduciendo el nombre:

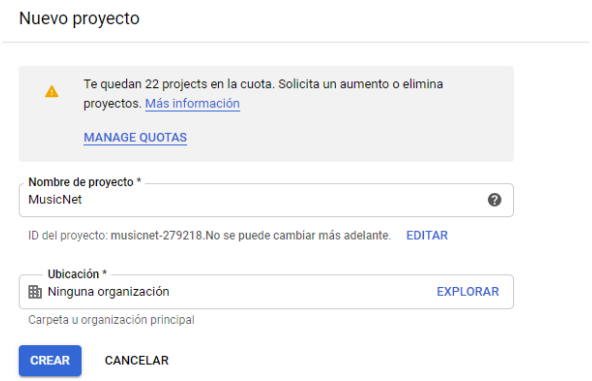


Figura 81: Crear proyecto en Google console

Seguidamente se ha de crear una “Clave de API” des de el menú “Credenciales” -> “Crear Credenciales”:

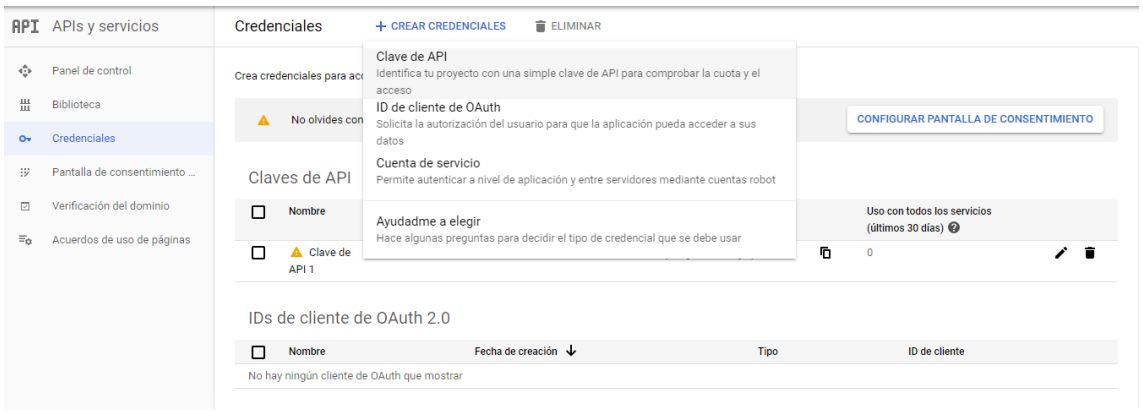


Figura 82: Configuración de credenciales

Para el caso que nos ocupa, no restringiremos el acceso a la clave:

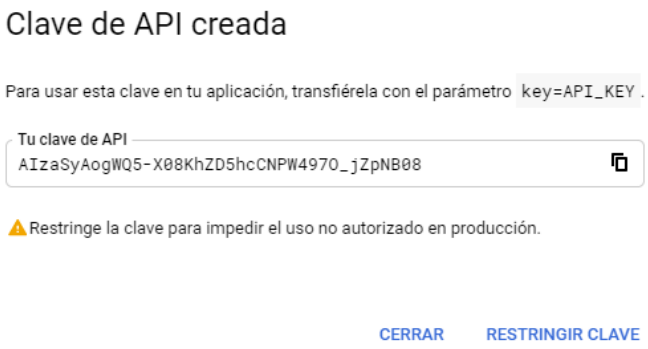


Figura 83: Ejemplo de claves creadas

Des del mismo menú des del que se ha creado la clave de API, se creará el ID de cliente de OAuth.

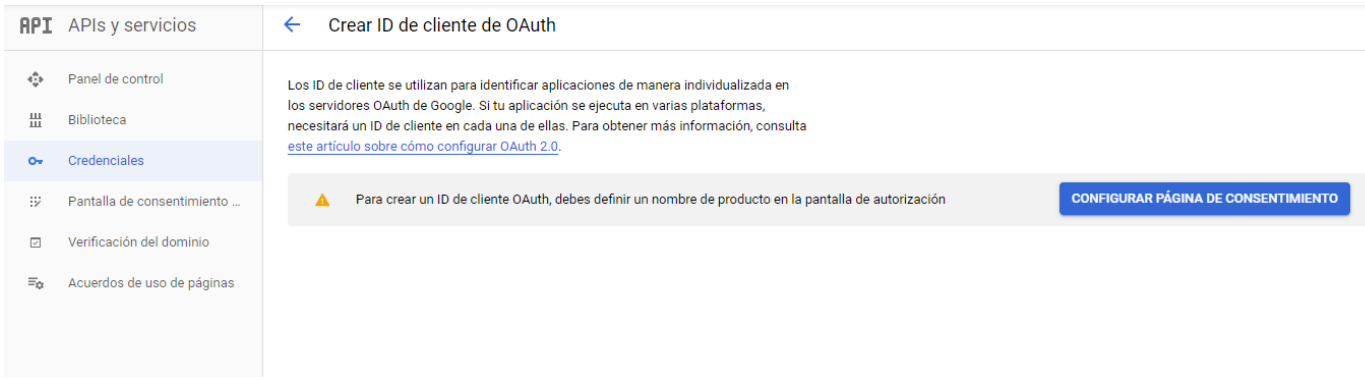


Figura 84: Crear id de cliente

Se pulsa en “Configurar página de consentimiento” y seguidamente en “Crear” habiendo marcado “Externos” como tipo de usuario:

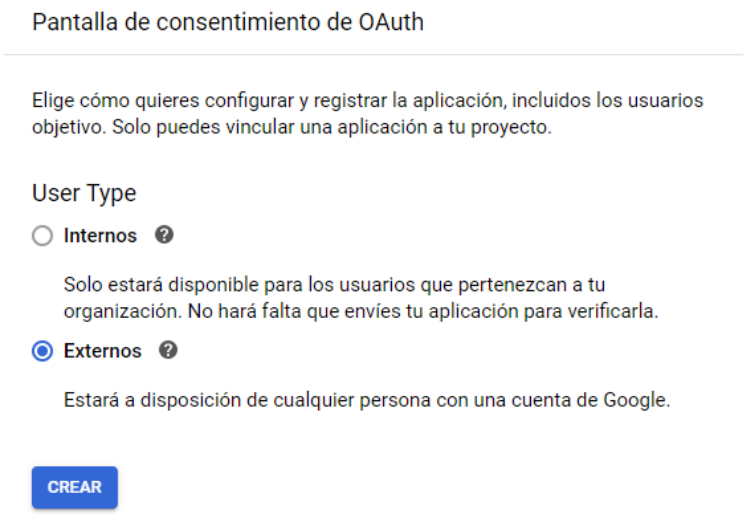
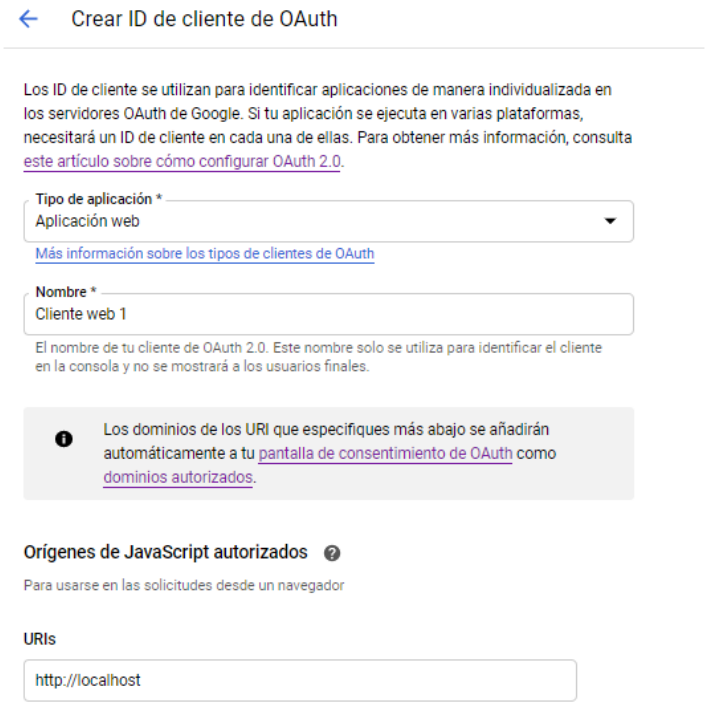


Figura 85: Pantalla de consentimiento

En la siguiente pantalla Google solicita bastante información, pero básicamente nosotros tan solo introduciremos el nombre de nuestra aplicación. Una vez creado el ID del cliente, se configurará accediendo de nuevo a “Credenciales” -> “Crear credenciales” -> “ID del cliente OAuth”. Al acceder esta segunda vez, Google Console solicitará más información sobre la aplicación:



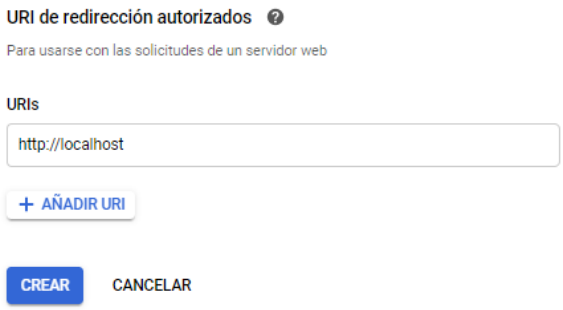


Figura 86: Configurando aplicación

Y finalmente se obtendrán las claves de acceso para ese id del cliente.

Figura 87: Ejemplo de clave

Los desarrolladores de Youtube ofrecen unos ejemplos de como usar su API en https://github.com/youtube/api-samples. Para nuestro proyecto, se ha utilizado youtube/api-samples/java ya que el BackEnd de MusicNet es Java.

Básicamente se han seguido los siguientes pasos:

- Se han adaptado las clases UploadVideo.java y Auth.java a nuestros requisitos. En definitiva, se han importado estas clases (con sus respectivos paquetes) al proyecto de Eclipse y modificado para implementar la funcionalidad exacta y necesaria.
- Se han modificado las dependencias de Maven añadiendo todas las necesarias para implementar la API de Youtube (ver apartado 8.2 Dependencias).
- Se importan los archivos client_secrets.json y youtube.properties en src/main/resources y se modifican para que estos contengan la clave de la api de youtube asi como el id del cliente y su secreto.

7.1.8 Zoom API

Para poder explotar las funcionalidades de video conferencia que ofrece la API de Zoom se necesita estar registrado en su plataforma. Por lo tanto, el primer paso será registrarnos en https://marketplace.zoom.us/

Seguidamente, se debe registrar la aplicación que hará uso de la API de Zoom (en nuestro caso será MusicNet). Des del menú para desarrolladores, se debe seleccionar “JWT”:

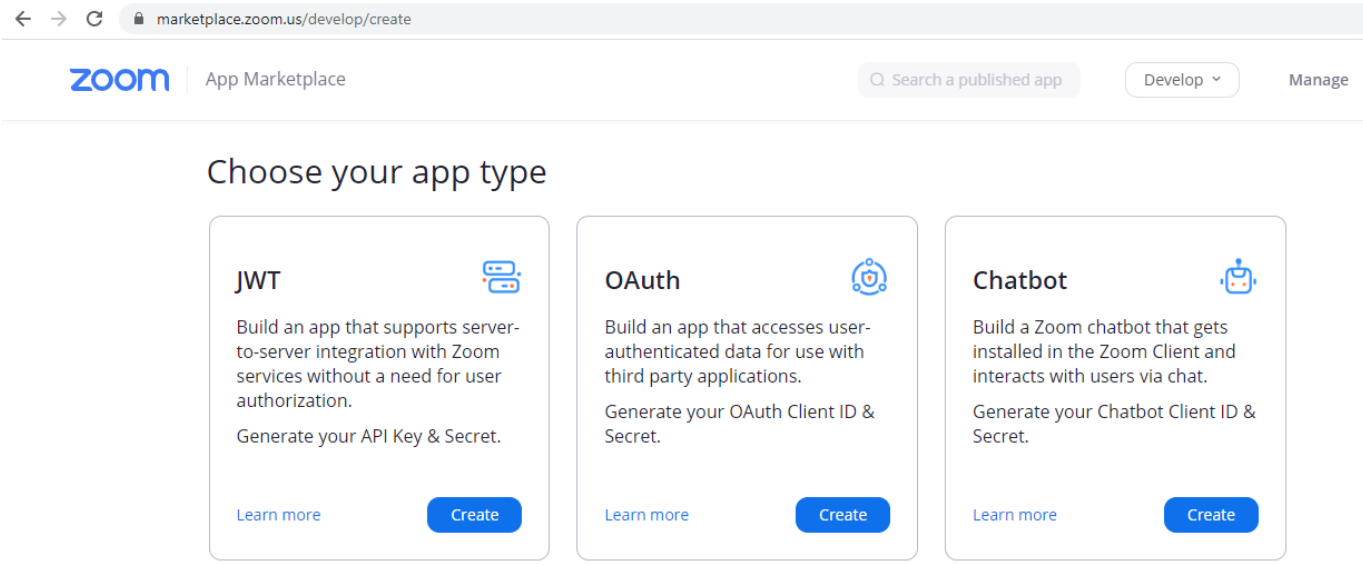


Figura 88: Interfaz Marketplace de zoom

Introducimos el nombre de la aplicación:

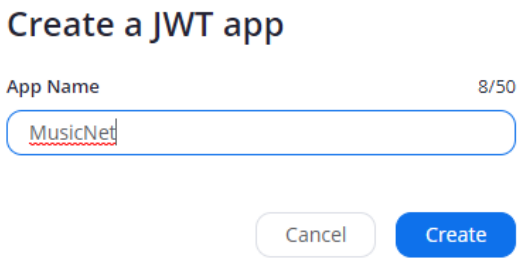


Figura 89: Creando JWT para la app de zoom

Luego se introduce otras informaciones como la dirección de correo:

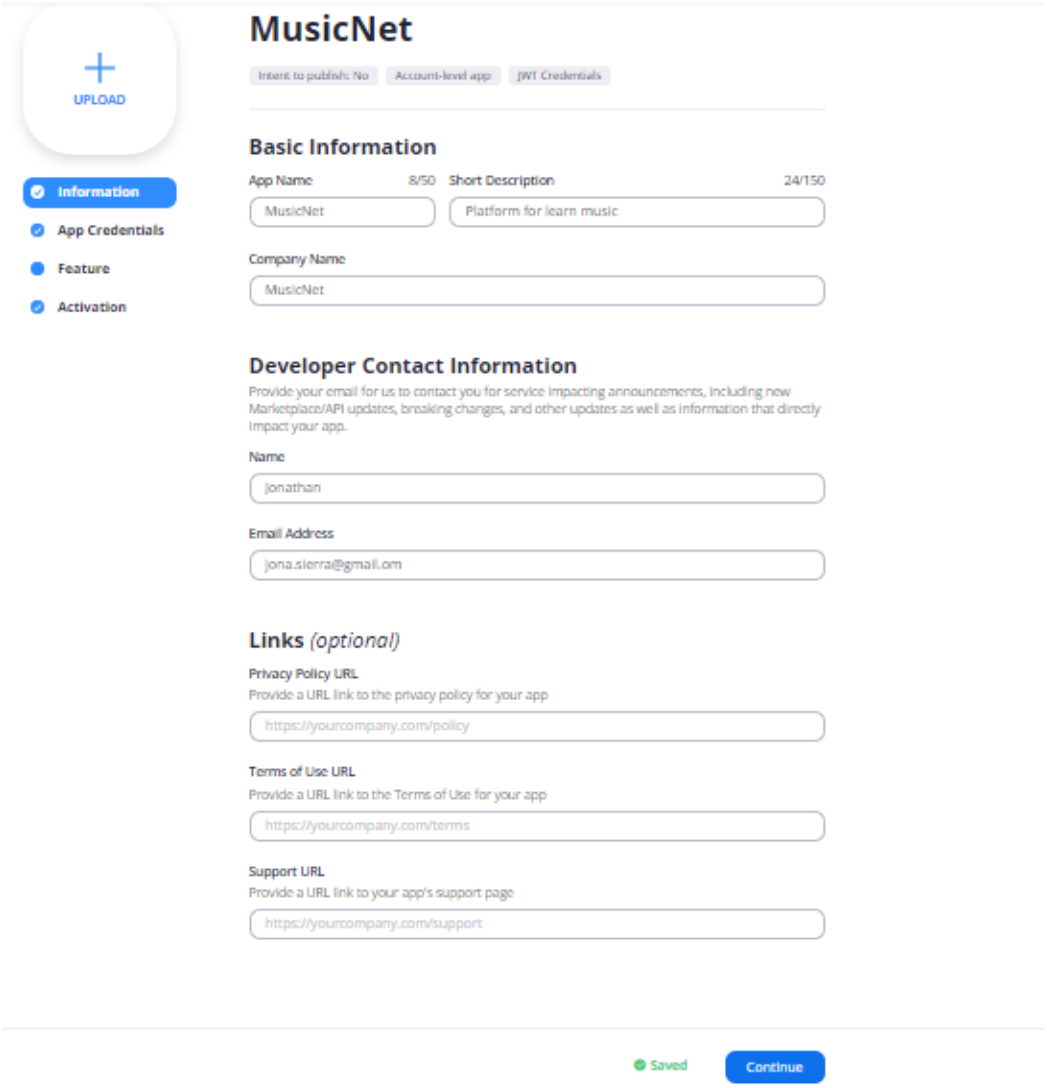


Figura 90: Añadiendo información sobre MusicNet

Se obtiene la clave que utilizaremos para usar la API de Zoom asi como el secreto:

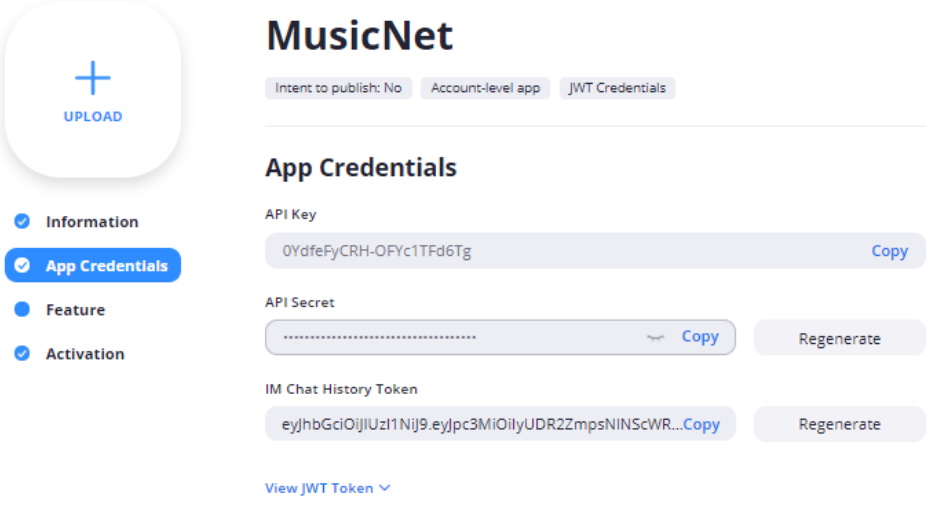


Figura 91: Ejemplo de credenciales

Aceptamos las “Features” que están por defecto:

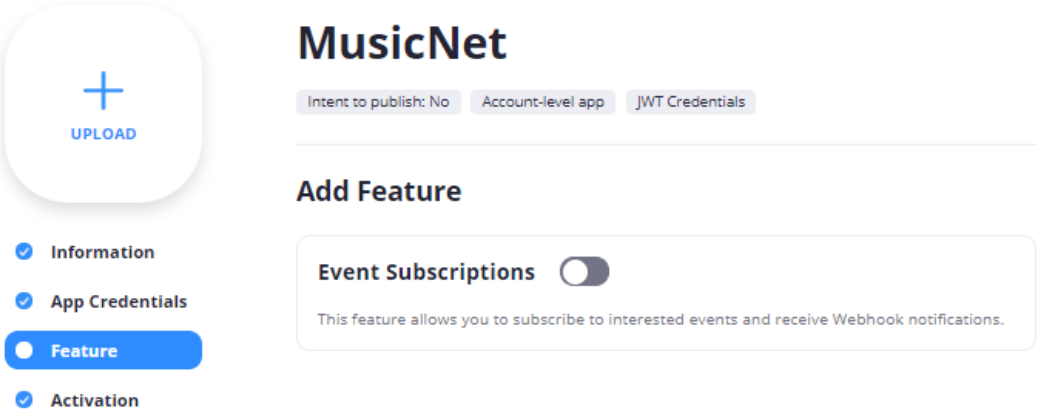


Figura 92: Ejemplo de caracterisiticas

Finalmente, tenemos la aplicación configurada para ser usada:

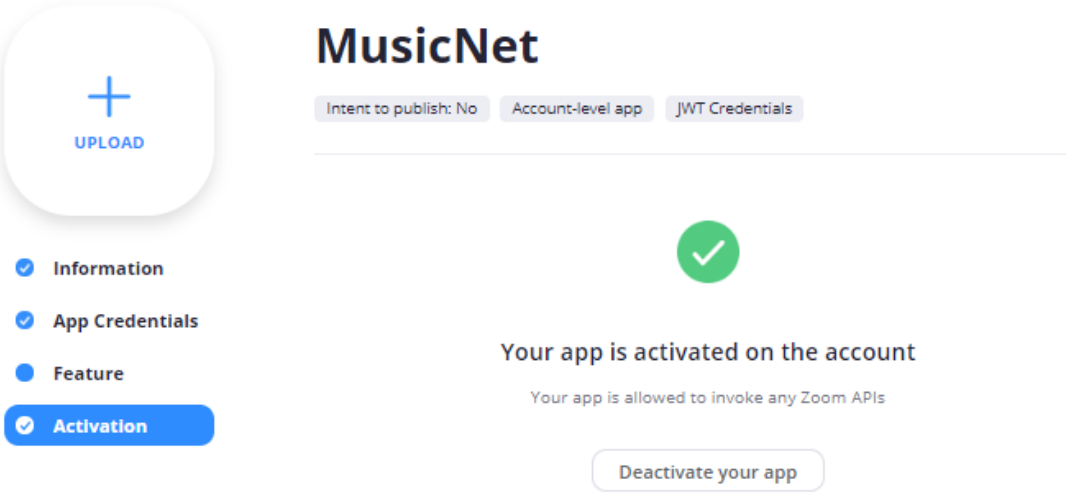


Figura 93: Ejemplo de activación

Por último, para que MusicNet pueda integrar zoom se hará a partir de un ejemplo. Es decir, en https://github.com/zoom/sample-app-web se puede encontrar un ejemplo del uso web de Zoom:

- Se descarga solo el contenido del directorio CDN.
- Se añade localhost.crt, localhost.key y package.json en el directorio src/main/resources/static de Eclipse.
- Se añaden los archivos javascript dentro de src/main/resources/static/js.
- Se modifica el archivo index.html para que sea la vista classroom.html.
- Se añaden las el identificador de la api y la correspondiente clave.

7.1.9 Stripe API

Para poder explotar las funcionalidades de compra y venta que ofrece la API de Stripe se necesita estar registrado en su plataforma. Por lo tanto, el primer paso será registrarnos en https://stripe.com/

Al igual que con las otras APIs utilizadas en MusicNet, se deberá generar el identificador de la API con su respectiva clave secreta. Para ello, después de registrarnos en la plataforma de Stripe, se tendrá a acceso a un panel de control. En el menú “Desarrolladores” se encuentra el apartado “Claves de api”. Des de allí, se podrán generar las claves de una forma muy sencilla y rápida (tan solo hay que introducir los datos que Stripe nos solicita los cuales son muy sencillos e intuitivos).

Después podremos ver estos datos des del panel de control:

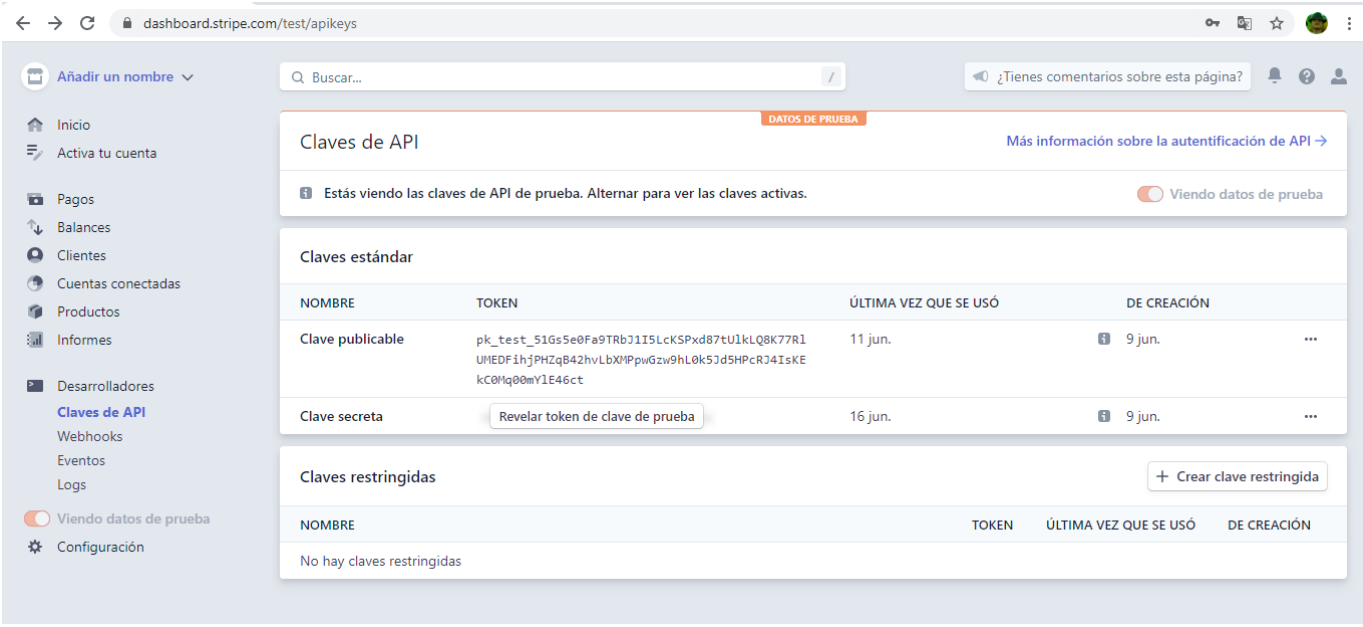


Figura 94: Panel de control

Para implementar la integración de Stripe en MusicNet configuraremos las dependencias necesarias en el archivo pom.xml (ver 7.2 Dependencias).

Para mayor claridad, en la siguiente imagen se muestra el flujo del proceso de compra venta:

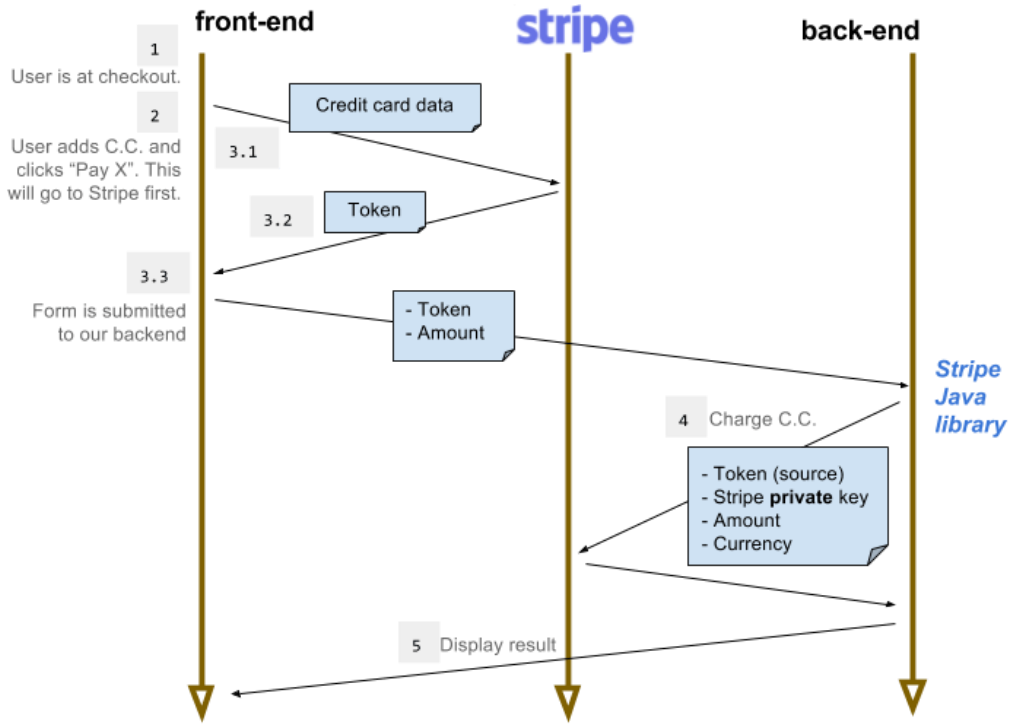


Figura 95: Flujo de comunicación en el uso de Stripe

A partir de toda esta información, solo hemos de codificar la parte FrontEnd y la parte BackEnd. Para la parte FrontEnd, se crea un formulario el cual envía los datos necesarios a Stripe y al BackEnd y para la parte BackEnd se codifica los controladores necesarios para la comunicación entre el FrontEnd y Stripe.

7.2 Dependencias

Maven es el gestor de dependencias de Java utilizado. En la raíz del proyecto dentro de eclipse se encuentra el archivo pom.xml el cual define y administra todas las dependencias de MusicNet. Es importante conocer este archivo:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.2.7.RELEASE</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.musicnet</groupId>
	<artifactId>musicnet</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>musicnet</name>
	<description>MusicNet is Spring project to develop the
TFG</description>
	<properties>
		<java.version>1.8</java.version>
```

```
            <project.youtube.version>v3-rev182-
1.22.0</project.youtube.version>
        <project.youtube.analytics.version>v1-rev63-
1.22.0</project.youtube.analytics.version>
        <project.youtube.reporting.version>v1-rev10-
1.22.0</project.youtube.reporting.version>
        <project.http.version>1.20.0</project.http.version>
        <project.oauth.version>1.20.0</project.oauth.version>
      </properties>
      <repositories>
        <repository>
            <id>google-api-services</id>
            <url>http://google-api-client-
libraries.appspot.com/mavenrepo</url>
        </repository>
    </repositories>
            <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-web-services</artifactId>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-web</artifactId>
            </dependency>
            <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
      </dependency>
      <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
      </dependency>
            <dependency>
                  <groupId>org.postgresql</groupId>
                  <artifactId>postgresql</artifactId>
                  <scope>runtime</scope>
            </dependency>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
                  <exclusions>
                        <exclusion>
                              <groupId>org.junit.vintage</groupId>
                              <artifactId>junit-vintage-engine</artifactId>
                        </exclusion>
                  </exclusions>
            </dependency>
       <!-- YouTube Data V3 support -->
        <dependency>
            <groupId>com.google.apis</groupId>
            <artifactId>google-api-services-youtube</artifactId>
            <version>${project.youtube.version}</version>
        </dependency>
```

```
        <!-- Required for any code that makes calls to the YouTube Analytics
API -->
        <dependency>
            <groupId>com.google.apis</groupId>
            <artifactId>google-api-services-youtubeAnalytics</artifactId>
            <version>${project.youtube.analytics.version}</version>
        </dependency>
        <!-- Required for any code that makes calls to the YouTube Reporting
API -->
        <dependency>
            <groupId>com.google.apis</groupId>
            <artifactId>google-api-services-youtubereporting</artifactId>
            <version>${project.youtube.reporting.version}</version>
        </dependency>
        <!-- This dependency is only used for the Topics API sample, which
requires the Jackson JSON parser -->
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-mapper-asl</artifactId>
            <version>1.9.4</version>
        </dependency>
        <dependency>
            <groupId>com.google.http-client</groupId>
            <artifactId>google-http-client-jackson2</artifactId>
            <version>${project.http.version}</version>
        </dependency>
        <dependency>
            <groupId>com.google.oauth-client</groupId>
            <artifactId>google-oauth-client-jetty</artifactId>
            <version>${project.oauth.version}</version>
        </dependency>
        <dependency>
            <groupId>com.google.collections</groupId>
            <artifactId>google-collections</artifactId>
            <version>1.0</version>
        </dependency>
        <dependency>
                    <groupId>com.stripe</groupId>
                    <artifactId>stripe-java</artifactId>
                    <version>4.2.0</version>
             </dependency>
             <dependency>
             <groupId>org.projectlombok</groupId>
             <artifactId>lombok</artifactId>
             </dependency>
      </dependencies>

      <build>
            <plugins>
                  <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                  </plugin>
            </plugins>
      </build>
</project>
```

En el se pueden ver todas las dependencias. Algunas de ellas están definidas automáticamente, por ejemplo, todas aquellas que se definieron en “Spring

Initializr". Otras, se han definido editando directamente el archivo pom.xml como son, por ejemplo, todas las dependencias de Google.

Durante el proceso "Build" del proyecto, Maven procesará el archivo y resolverá todas las dependencias. Las clases resultantes se guardarán en "Maven Dependencies" de Eclipse.

7.3 Estructura de directorios

La siguiente imagen muestra la estructura de directorios de MusicNet des de la interfaz de Eclipse:

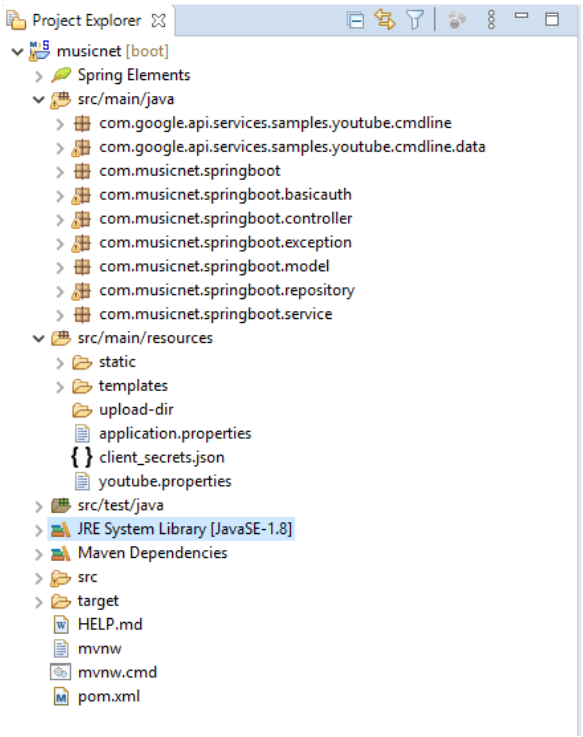


Figura 96: Estructura de directorios

- Dentro del directorio src/main/java se encuentran los paquetes de la aplicación:
  · com.google.api.services.samples.youtube.cmdline: contiene las clases java para la autorización del usuario y para almacenar las credenciales.
  · com.google.api.services.samples.youtube.cmdline.data: contiene las clases java necesarias para la subida de archivos a youtube
  · com.musicnet.springboot: contiene la clase main de Spring.
  · com.musicnet.springboot.basicauth: contiene la clase que implementa las reglas de seguridad de Spring.
  · com.musicnet.springboot.controller: contiene las clases que hacen de controladores (implementa la API de musicnet).
  · com.musicnet.springboot.exception: contiene las clases que implementan las excepciones de la aplicación.
  · com.musicnet.springboot.model: contiene las clases del modelo.

· com.musicnet.springboot.repository: contiene las clases que permiten tratar las clases del modelo como repositorios.
· com.musicnet.springboot.services: contiene las clases que implementan los servicios del sistema.

- Dentro del directorio src/main/resources se encuentra lo siguiente:
  · static: directorio que almacena los archivos html, css y javascript. Tambien guarda los archivos de configuración para la API de Zoom.
  · templates: directorio que almacena las plantillas thymeleaf que, a su vez, son las vistas del sistema.
  · upload-dir: directorio donde se almacenan los videos subidos por los usuarios.
  · application.properties: archivo de configuración de Spring.
  · client_secrets.json: archive para configurar las contraseñas de Youtube.
  · youtube.properties: archivo de configuración de la API de Youtube.

- Dentro del directorio src/test/java se encuentran los test de Spring.
- JRE System Library almacena las librerías JRE.
- Maven Dependencies almacena todas las librerías descargadas por Maven.
- Pom.xml es el archivo de configuración de Maven que gestiona las dependencias de Spring.

7.4 Implementación física de la base de datos

Script para la creación de las tablas de base de datos:

```
CREATE TABLE musicnet.user(
        name VARCHAR (10) NOT NULL,
        firstname VARCHAR (10) NOT NULL,
        secondname VARCHAR (10),
        emailaddress VARCHAR (50) NOT NULL,
        physicaladdress VARCHAR (50),
        password VARCHAR (10) NOT NULL,
        numbertelephone VARCHAR (10),
        PRIMARY KEY (emailaddress)
);

CREATE TABLE musicnet.student(
        name VARCHAR (10) NOT NULL,
        firstname VARCHAR (10) NOT NULL,
        secondname VARCHAR (10),
        emailaddress VARCHAR (50) NOT NULL,
        physicaladdress VARCHAR (50),
        password VARCHAR (10) NOT NULL,
        numbertelephone VARCHAR (10),
        PRIMARY KEY (emailaddress),
        FOREIGN KEY (emailaddress) REFERENCES musicnet.user (emailaddress)
);

CREATE TABLE musicnet.teacher(
        name VARCHAR (10) NOT NULL,
        firstname VARCHAR (10) NOT NULL,
        secondname VARCHAR (10),
        emailaddress VARCHAR (50) NOT NULL,
        physicaladdress VARCHAR (50),
```

```
        password VARCHAR (10) NOT NULL,
        numbertelephone VARCHAR (10),
        bankaccount VARCHAR (30) NOT NULL,
        PRIMARY KEY (emailaddress),
        FOREIGN KEY (emailaddress) REFERENCES musicnet.user (emailaddress)
);

CREATE TABLE musicnet.administrator(
        name VARCHAR (10) NOT NULL,
        firstname VARCHAR (10) NOT NULL,
        secondname VARCHAR (10),
        emailaddress VARCHAR (50) NOT NULL,
        physicaladdress VARCHAR (50),
        password VARCHAR (10) NOT NULL,
        numbertelephone VARCHAR (10),
        PRIMARY KEY (emailaddress),
        FOREIGN KEY (emailaddress) REFERENCES musicnet.user (emailaddress)
);

CREATE TABLE musicnet.videoconference(
        teachervideo VARCHAR (50) NOT NULL,
        emailaddressteacher VARCHAR (50) NOT NULL,
        PRIMARY KEY (teachervideo),
        FOREIGN KEY (emailaddressteacher) REFERENCES musicnet.teacher (emailaddress)
);

CREATE TABLE musicnet.messages(
        idvideoconference VARCHAR (50 NOT NULL,
        message TEXT,
        PRIMARY KEY (idvideoconference),
        FOREIGN KEY (idvideoconference) REFERENCES musicnet.videoconference (teachervideo)
);

CREATE TABLE musicnet.goto(
        emailaddressstudent VARCHAR (50) NOT NULL,
        idvideoconference VARCHAR (50 NOT NULL,
        PRIMARY KEY (emailaddressstudent, idvideoconference),
        FOREIGN KEY (emailaddressstudent) REFERENCES musicnet.student (emailaddress),
        FOREIGN KEY (idvideoconference) REFERENCES musicnet.videoconference (teachervideo)
);

CREATE TABLE musicnet.style(
        name VARCHAR (10) NOT NULL,
        description TEXT NOT NULL,
        PRIMARY KEY (name)
);

CREATE TABLE musicnet.instrument(
        name VARCHAR (10) NOT NULL,
        description TEXT NOT NULL,
        PRIMARY KEY (name)
);

CREATE TABLE musicnet.difficulty(
        name VARCHAR (10) NOT NULL,
        PRIMARY KEY (name)
);

CREATE TABLE musicnet.lenguage(
        name VARCHAR (10) NOT NULL,
        PRIMARY KEY (name)
);

CREATE TABLE musicnet.course(
        id SERIAL NOT NULL,
        description TEXT NOT NULL,
        instrumentname VARCHAR (10) NOT NULL,
```

```
        difficulty VARCHAR (10) NOT NULL,
        hours INTEGER NOT NULL,
        studentcontact BOOLEAN NOT NULL,
        teachercontact BOOLEAN NOT NULL,
        lenguage VARCHAR (10) NOT NULL,
        emailaddressteacher VARCHAR (50) NOT NULL,
        stylename VARCHAR (10) NOT NULL,
        PRIMARY KEY (id),
        FOREIGN KEY (emailaddressteacher) REFERENCES musicnet.teacher (emailaddress),
        FOREIGN KEY (stylename) REFERENCES musicnet.style (name),
        FOREIGN KEY (instrumentname) REFERENCES musicnet.instrument (name),
        FOREIGN KEY (difficulty) REFERENCES musicnet.difficulty (name),
        FOREIGN KEY (lenguage) REFERENCES musicnet.lenguage (name)
);

CREATE TABLE musicnet.study(
        emailaddressstudent VARCHAR (email) NOT NULL,
        courseid SERIAL NOT NULL,
        PRIMARY KEY (emailaddressstudent, courseid),
        FOREIGN KEY (emailaddressstudent) REFERENCES musicnet.student (emailaddress),
        FOREIGN KEY (courseid) REFERENCES musicnet.course (id)
);

CREATE TABLE musicnet.experience(
        emailaddressteacher VARCHAR (email) NOT NULL,
        stylename VARCHAR (10) NOT NULL,
        years INTEGER NOT NULL,
        PRIMARY KEY (emailaddressteacher, stylename),
        FOREIGN KEY (emailaddressteacher) REFERENCES musicnet.teacher (emailaddress),
        FOREIGN KEY (stylename) REFERENCES musicnet.style (name)
);

CREATE TABLE musicnet.materials(
        id SERIAL NOT NULL,
        courseid SERIAL NOT NULL,
        description TEXT NOT NULL,
        tablature BYTEA NOT NULL,
        musicsheet BYTEA NOT NULL,
        PRIMARY KEY (id, courseid),
        FOREIGN KEY (courseid) REFERENCES musicnet.course (id)
);

CREATE TABLE musicnet.videomaterials(
        idmaterials SERIAL NOT NULL,
        video VARCHAR (50) NOT NULL,
        PRIMARY KEY (video),
        FOREIGN KEY (idmaterials) REFERENCES musicnet.course (id)
);

CREATE TABLE musicnet.metronom(
        name VARCHAR (30) NOT NULL,
        description TEXT NOT NULL,
        file VARCHAR (50) NOT NULL,
        velocity INTEGER NOT NULL,
        PRIMARY KEY (velocity)
);

CREATE TABLE musicnet.backingtrack(
        name VARCHAR (30) NOT NULL,
        description TEXT NOT NULL,
        file BYTEA NOT NULL,
        velocity INTEGER NOT NULL,
        PRIMARY KEY (name, velocity)
);

CREATE TABLE musicnet.compastrack(
        name VARCHAR (30) NOT NULL,
```

```
        description TEXT NOT NULL,
        file VARCHAR (50) NOT NULL,
        velocity INTEGER NOT NULL,
        PRIMARY KEY (name, velocity)
);

CREATE TABLE musicnet.tools(
        courseid SERIAL NOT NULL,
        metronom1 INTEGER NOT NULL,
        metronom2 INTEGER,
        metronom3 INTEGER,
        metronom4 INTEGER,
        metronom5 INTEGER,
        metronom6 INTEGER,
        backingtrackname1 VARCHAR (30) NOT NULL,
        backingtrackvelocity1 INTEGER NOT NULL,
        backingtrackname2 VARCHAR (30),
        backingtrackvelocity2 INTEGER,
        backingtrackname3 VARCHAR (30),
        backingtrackvelocity3 INTEGER,
        compastrackname1 VARCHAR (30) NOT NULL,
        compastrackvelocity1 INTEGER NOT NULL,
        compastrackname2 VARCHAR (30),
        compastrackvelocity2 INTEGER,
        compastrackname3 VARCHAR (30),
        compastrackvelocity3 INTEGER,
        PRIMARY KEY (courseid),
        FOREIGN KEY (courseid) REFERENCES musicnet.course (id),
        FOREIGN KEY (metronom1) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (metronom2) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (metronom3) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (metronom4) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (metronom5) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (metronom6) REFERENCES musicnet.metronom (velocity),
        FOREIGN KEY (backingtrackname1, backingtrackvelocity1) REFERENCES
musicnet.backingtrack (name, velocity),
        FOREIGN KEY (backingtrackname2, backingtrackvelocity2) REFERENCES
musicnet.backingtrack (name, velocity),
        FOREIGN KEY (backingtrackname3, backingtrackvelocity3) REFERENCES
musicnet.backingtrack (name, velocity),
        FOREIGN KEY (compastrackname1, compastrackvelocity1) REFERENCES
musicnet.compastrack (name, velocity),
        FOREIGN KEY (compastrackname2, compastrackvelocity2) REFERENCES
musicnet.compastrack (name, velocity),
        FOREIGN KEY (compastrackname3, compastrackvelocity3) REFERENCES
musicnet.compastrack (name, velocity)
);
```