FACULTAD DE VALLADOLID

Escuela de Ingeniería Informática

Diseño, Integración y Adaptación de Software

Práctica de Laboratorio - Grupo 5

Sergio Esteban Pellejero - Alejandro Martínez Andrés - Jose Ignacio Hernández Velsaco

Índice

Índice de figuras

1. Introducción

En esta práctica de laboratorio tenemos que modelar un sistema software para una compañía de música. Siendo esto una muy breve descripción del sistema, la compañía precisa de un software capaz de mantener una página web en la que el usuario a grandes rasgos pueda: descargar música, organizar la música que descarga, suscribirse a diferentes grupos de interés, comprar diferentes suscripciones a la plataforma, etc. Esta empresa también pone a disposición de todo el público, máquina grabadoras de CD'S en diferentes puntos de la ciudad, para que todas aquellas personas que quieran puedan comprar música en formato CD de manera personalizada.

Dicho esto, los diferentes diagramas que tenemos que realizar para modelar el sistema en cuestión, son:

- **Modelo de dominio**: describiendo las principales partes que componen el sistema y de las cuales guardamos algún tipo de información, importante para la empresa.
- **Diagrama de secuencia - Análisis - 1**: describiendo el caso de uso *Realizar Pedido Merchandaising*.
- **Diagrama de secuencia - Análisis - 2**: describiendo el caso de uso *Descarga de Álbum*.
- **Diagrama de clases**: gracias al modelo de dominio y después de introducir algunas operaciones, a partir de los diagramas de secuencias, tenemos el diagrama de clases.
- **Propuesta de arquitectura del sistema**: en este apartado tenemos que decir que patrones arquitectónicos van a componer nuestro sistema y porqué hemos decidido usar esos y no otros. Este apartado estará acompañado de:
  - Diagrama de paquetes: donde se ve la estructuración lógica del sistema.
  - Diagrama de despliegue: donde se puede ver la estructura física del sistema.
- **Diagrama de secuencia - Diseño - 1**: describiendo el caso de uso *Crear Disco* para una máquina vending. Este diagrama irá acompañado de:
  - Diagrama de clases - Diseño: apropiado para ese caso de uso, y representando esa parte del sistema.

2. Modelo de Dominio del sistema

El modelo de dominio de nuestro sistema se puede encontrar en la **Figura 2**. Debido a lo grande que es, se encuentra en formato apaisado en la siguiente página.

Este modelo describe las principales entidades que nuestro sistema tiene que tener en cuenta y de las cuales vamos a almacenar algún tipo de información relevante para el mismo.

Los usuarios registrados y las canciones son las dos clases más importantes y básicas que tenemos, y a partir de ellas podemos dar sentido al sistema e implementar funcionalidades.

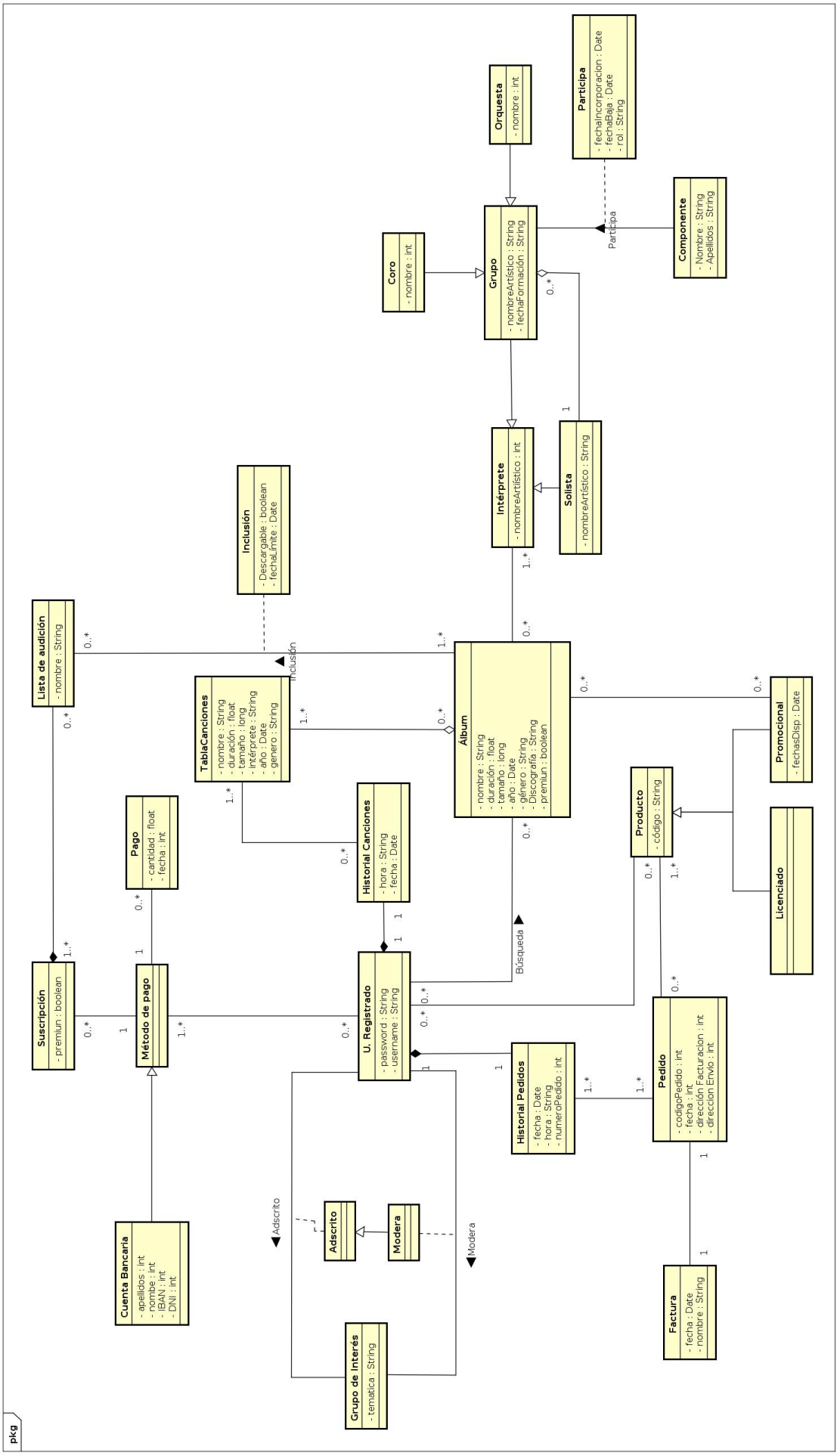

Figura 1: Modelo de Dominio del sistema

3. Realizaciones de casos de usos - Diagramas Secuencia

En esta sección, primero vamos a presentar los diagramas de secuencia asociados a cada una de las realizaciones de los diferentes casos de uso, acompañados de una breve descripción. Después obtendremos el diagrama de clases junto con las operaciones extraídas de los diagramas de secuencia.

3.1. Diagrama de secuencia Análisis 1 - Realizar Pedido Merchandising

La descripción de este caso de uso la podemos encontrar en el enunciado, por lo que simplemente la vamos a resumir. El sistema tiene que ser capaz de permitir a los usuarios registrados adquirir productos promocionales o licenciados vendidos por la empresa. Este diagrama de secuencia se puede ver en la **Figura 2**. Por temas de tamaño lo mostramos en una única hoja en apaisado para su mejor visualización.

3.2. Diagrama de secuencia Análisis 2 - Descargar Álbum

La descripción de este caso de uso la podemos encontrar en el enunciado de la práctica, por lo que simplemente la vamos a resumir. El sistema tiene que ser capaz de permitir a los usuarios registrados descargarse un álbum de música. Este diagrama de secuencia se puede ver en la **Figura 3**. Por temas de tamaño lo mostramos en una única hoja en apaisado para su mejor visualización.

3.3. Diagrama de clases - Análisis

Este diagrama de clases, es como el modelo de dominio del sistema, pero con más información. Aquí se añade el comportamiento de cada una de las clases que tenemos en nuestros diagramas de secuencias. Podemos encontrar este diagrama en la **Figura 4**. Como en los anteriores casos, en apaisado se ve bien si se gira el PDF hacia la derecha.

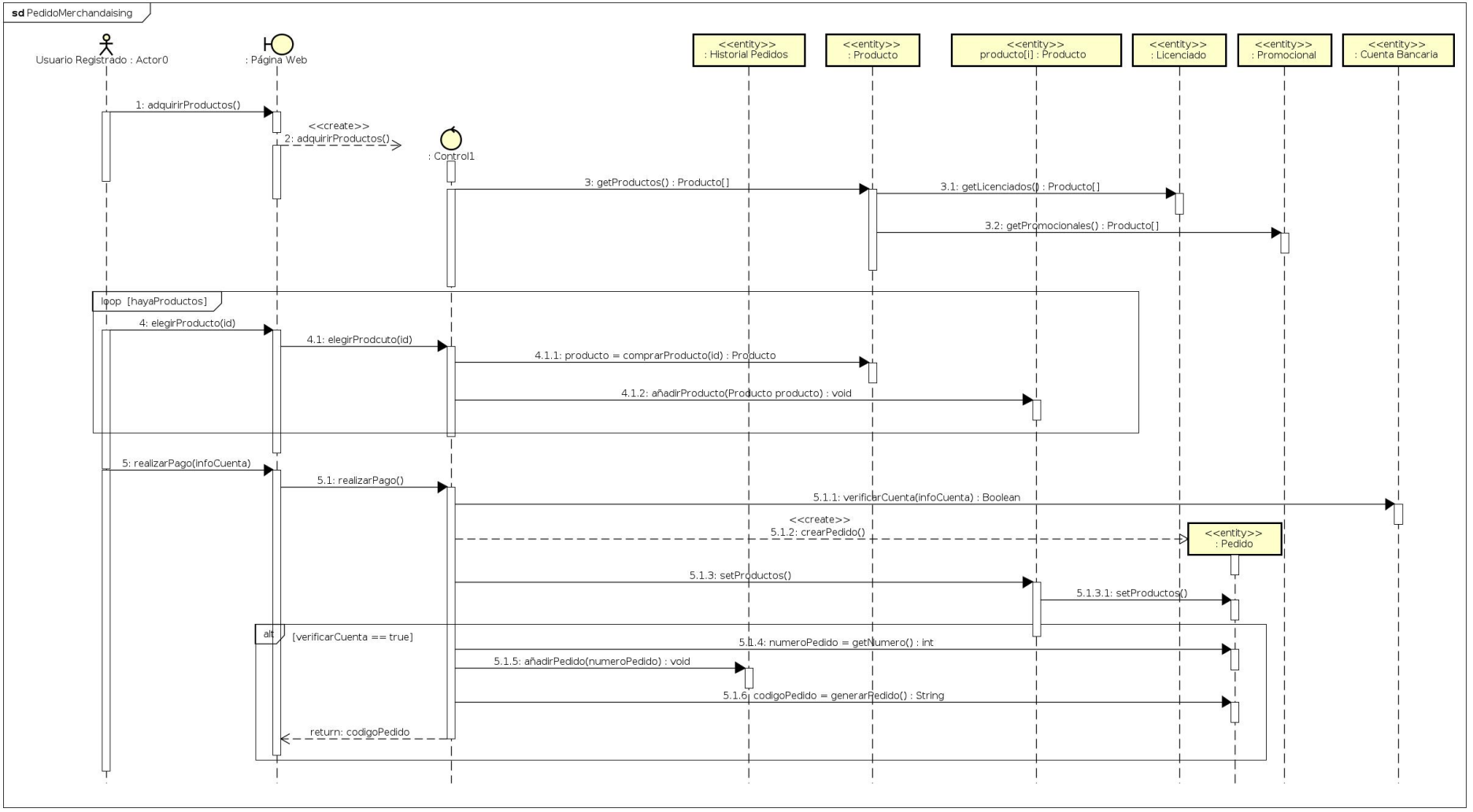

Figura 2: Diagrama de secuencia - Análisis - Realizar Pedido

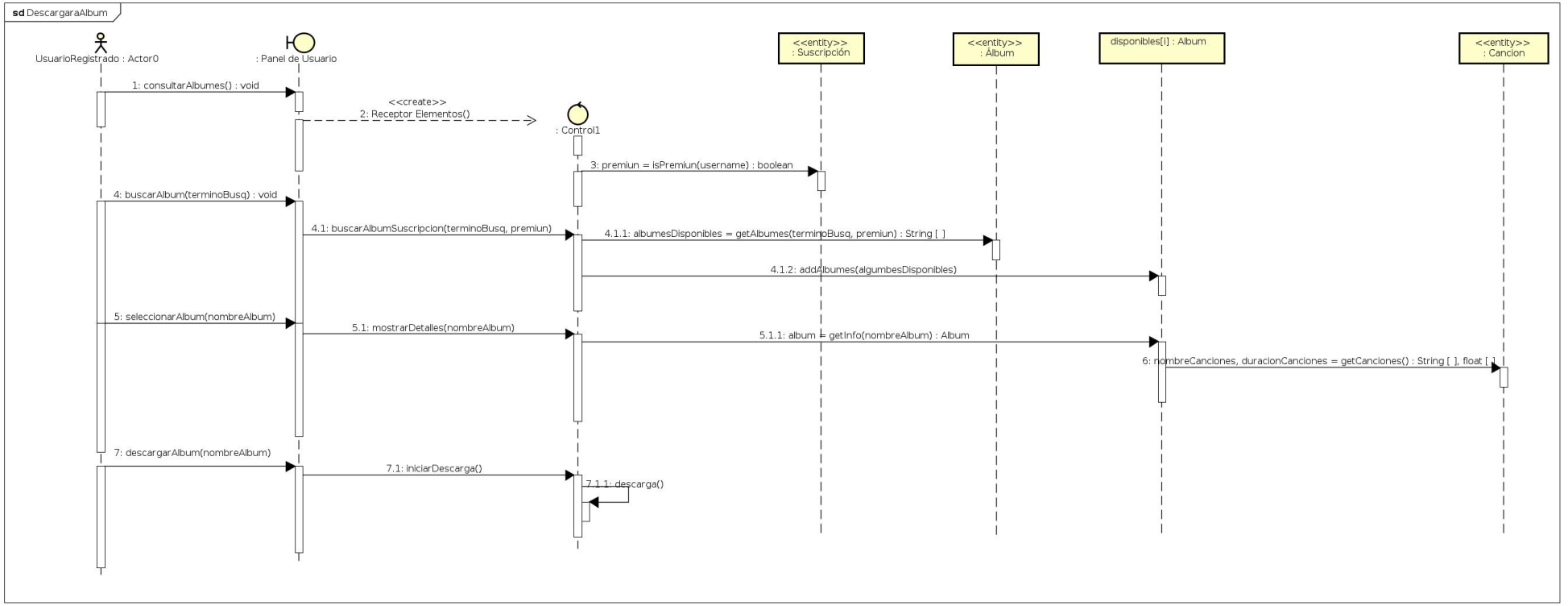

Figura 3: Diagrama de secuencia - Análisis - Descargar Álbum

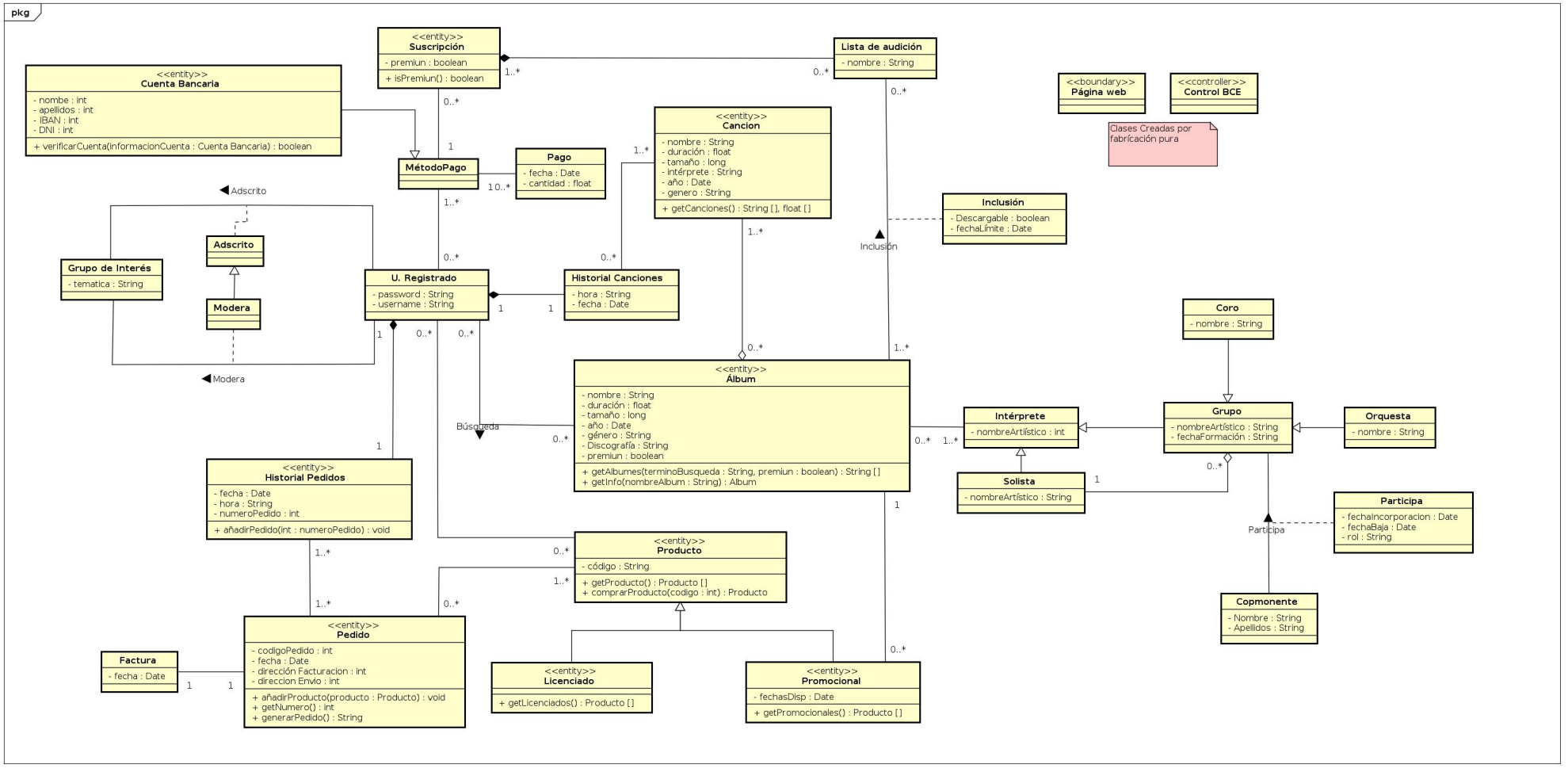

Figura 4: Diagrama de clases - Análisis

4. Propuesta de arquitectura del sistema

Aquí tenemos que describir los diferentes patrones que usa nuestro sistema para sustentar toda la funcionalidad pedida por los clientes, de manera que el sistema en cuestión sea más fácil de mantener, encontrar problemas y tenga unas bases sólidas en las que desarrollar mejoras futuras sin tener un alto impacto en el sistema actual.

Los patrones que hemos utilizado de arquitectura de software son:

- **MVC**: para la parte de la presentación de la página web del sistema, en la que los usuarios hacen consultas de la música. Usamos el MVC porque las vistas que tenemos de las páginas web cambian constantemente, se añaden nuevas funcionalidades que cambian la vista, se modernizan las apariencias. Es por esto que el MVC nos proporciona una solución antes los frecuentes cambios de las vistas.

- **Capas**: el patrón capas utilizado para estructurar todo nuestro sistema y decidir donde se van a poner las diferentes partes del mismo. Utilizamos este patrón debido a que nos proporciona un alto nivel de independencia entre las diferentes capas. Podemos cambiar la capa de la vista si se precisa, si las utilidades cambian podemos tocar esa capa sin tener que cambiar el resto. A continuación describiremos cada una de las usadas en nuestro sistema:

  - Interfaz: que representa las diferentes vistas que hay de nuestro sistema. Aquí podemos encontrar tanto la página web como la máquina de discos, que en nuestro caso se llama máquina de vending. Dentro se encuentran tanto la vista como el controlador del MVC nombrado anteriormente.

  - Servicios: aquí es donde se encuentran los principales servicios ofrecidos tanto por la página web como por la máquina de vending.

  - Dominio: es donde se encuentra la información importante para nuestra página, solo de la página porque la máquina es un cliente rico, encapsula toda su funcionalidad y solo requiere los datos de la BBDD de la empresa.

  - Servicios técnicos / Utilidades: aquí introducimos algunos servicios o utilidades que son necesarios para que todo el sistema funcione de manera correcta y para el control en el acceso a los datos.

  - Almacenamiento de datos: en esta última capa localizamos nuestra base de datos, donde se almacena la información necesaria para el sistema de una manera persistente.

Esta estructura se corresponde al diagrama de paquetes que podemos encontrar en la **Figura 5**.

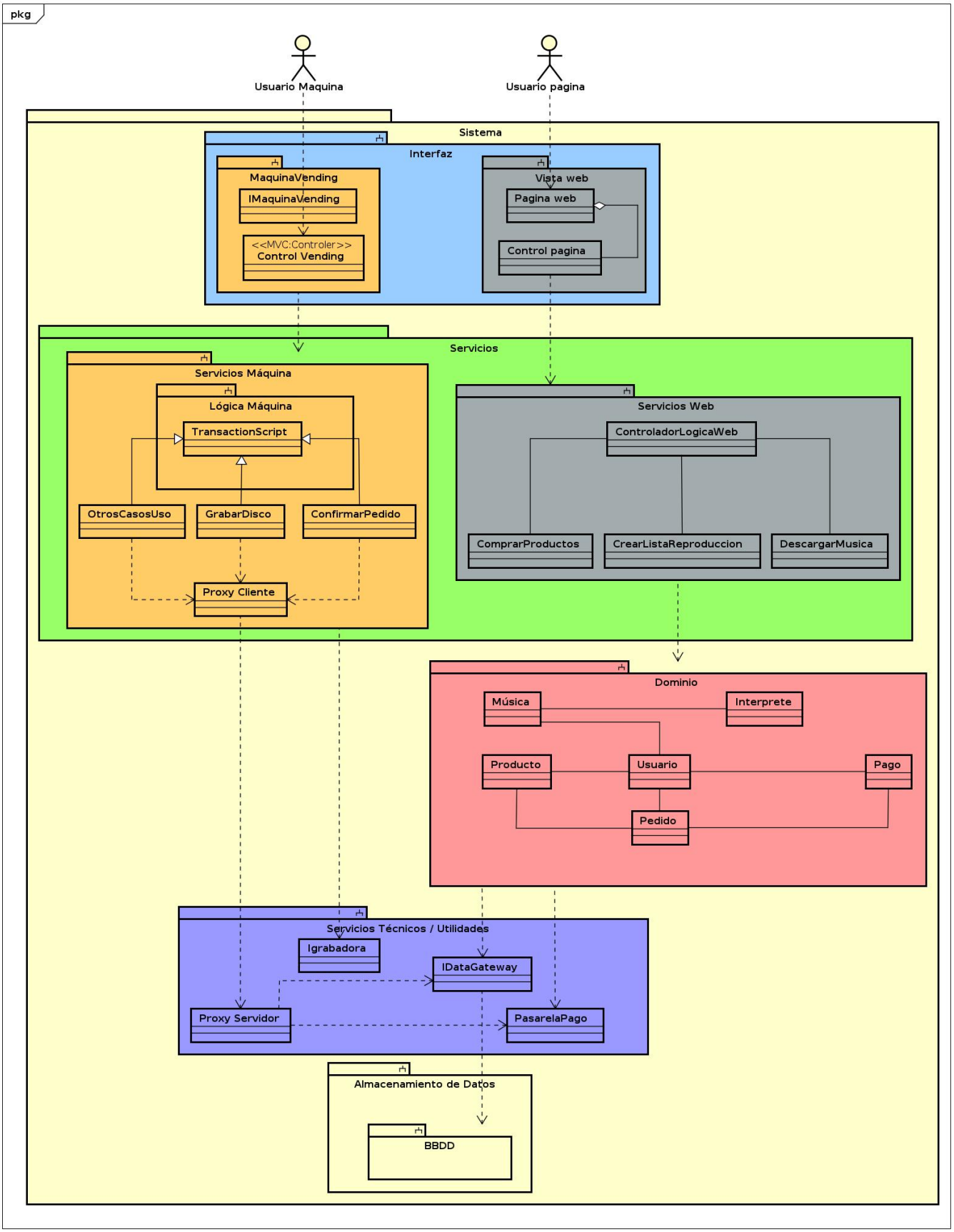

Figura 5: Diagrama de paquetes del sistema

A continuación presentamos un pequeño diagrama de despliegue para nuestra solución, en el que se puede ver la configuración aproximada del hardware necesario para desplegar el futuro sistema. Podemos ver este diagrama en la **Figura 6**.

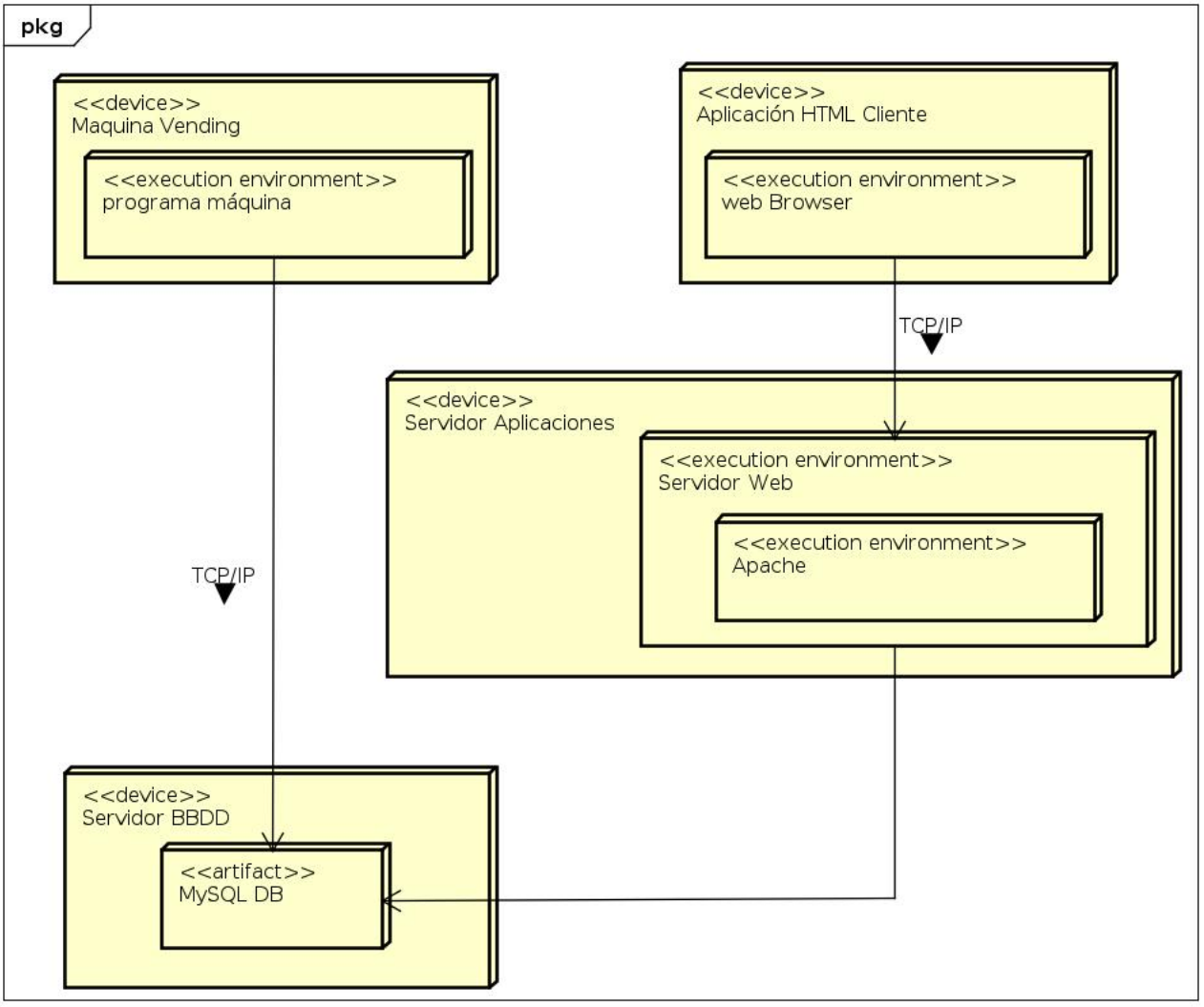

Figura 6: Diagrama de despliegue del sistema

5. Diagrama de secuencia Diseño - Grabar Disco

En este apartado se va a mostrar la realización del caso de uso - diseño de grabar disco. Para esto hemos creado un diagrama de secuencias y su diagrama de clases asociado (a este caso de uso no de todo el sistema).

Además, antes de presentar el diagrama de secuencia, vamos a introducir los patrones de diseño que hemos utilizado en esta parte:

- **Transaction Script**: hemos decidido usar este, porque la lógica de esta parte del

dominio es bastante sencilla. Al tener que diseñar solo una máquina que graba discos sacando las canciones de la BBDD, pensamos que un patrón para una lógica sencilla era suficiente. Es el encargado de invocar las diferentes operaciones que hace nuestra máquina para llevar a cabo la función de grabar un disco.

- **Proxy**: utilizamos el patrón proxy para el acceso a la BBDD. Evidentemente las máquinas no van a tener almacenado en el disco duro todas las canciones de la empresa, la idea es ir consultando el catálogo de la misma por partes que quepan en la memoria de la máquina y presentándolas al usuario. Para esta conexión a la BBDD, nosotros hemos aprovechado los proxys para que cada máquina no necesite saber dónde está el servidor de BBDD. Simplemente invoca a un intermediario que se encarga de encontrarla. Una vez que sabe donde conectarse, se procede al intercambio de archivos. Dde esta manera no solo ganamos en desacoplamiento, sino también en seguridad.

- **Adaptador**: el patrón adaptador lo hemos utilizado para la parte de los pagos. Nosotros disponemos de una método que nos proporciona la entidad bancaria contra el que realizar los pagos. La idea es que aunque ese método cambie, a nuestro sistema no le afecte o le afecte lo mínimo posible. Esto se hace gracias al patrón adaptador, que se conecta a ese método mediante una interfaz adecuada.

- **Row Data Gateway**: este patrón lo utilizamos para el acceso a los datos. El row data gateway lo que hace es crear un objeto por cada fila que solicitas a la BBDD. En principio, nosotros contábamos con traemos toda la BBDD, lo cual es una locura a nivel de espacio de almacenamiento y movimientos de datos. Pero la idea sería traer por ejemplo la información de las canciones por bloques manejables que quepan en memoria. Al igual que por ejemplo se hace en algunas redes sociales como Twitter, que por ejemplo te traen los ultimo 50 tweets, y si sigues bajando te da la opción de ver más, es entonces cuando hace la siguiente consulta a la BBDD y se trae los siguientes tweets. Nosotros hacemos lo mismo a la hora de mostrar la lista de las canciones al usuario.

Nuestro diagrama de secuencias de diseño lo podemos encontrar en el PDF adjunto a este informe. Esto es debido a que es un diagrama tan grande que no cabe en este PDF y no hemos logrado cambiar el tamaño de una única página en LaTeX. Así que hemos decidido darte otro PDF en tamaño A3 en el que sí que se ve el diagrama completo sin perder resolución.

5.1. Diagrama de clases - diseño

Este diagrama de clases de diseño, representa las diferentes clases que hemos utilizado en nuestro sistema y como interaccionan entre ellas. Este diagrama de clases lo podemos encontrar en la **Figura 7**.

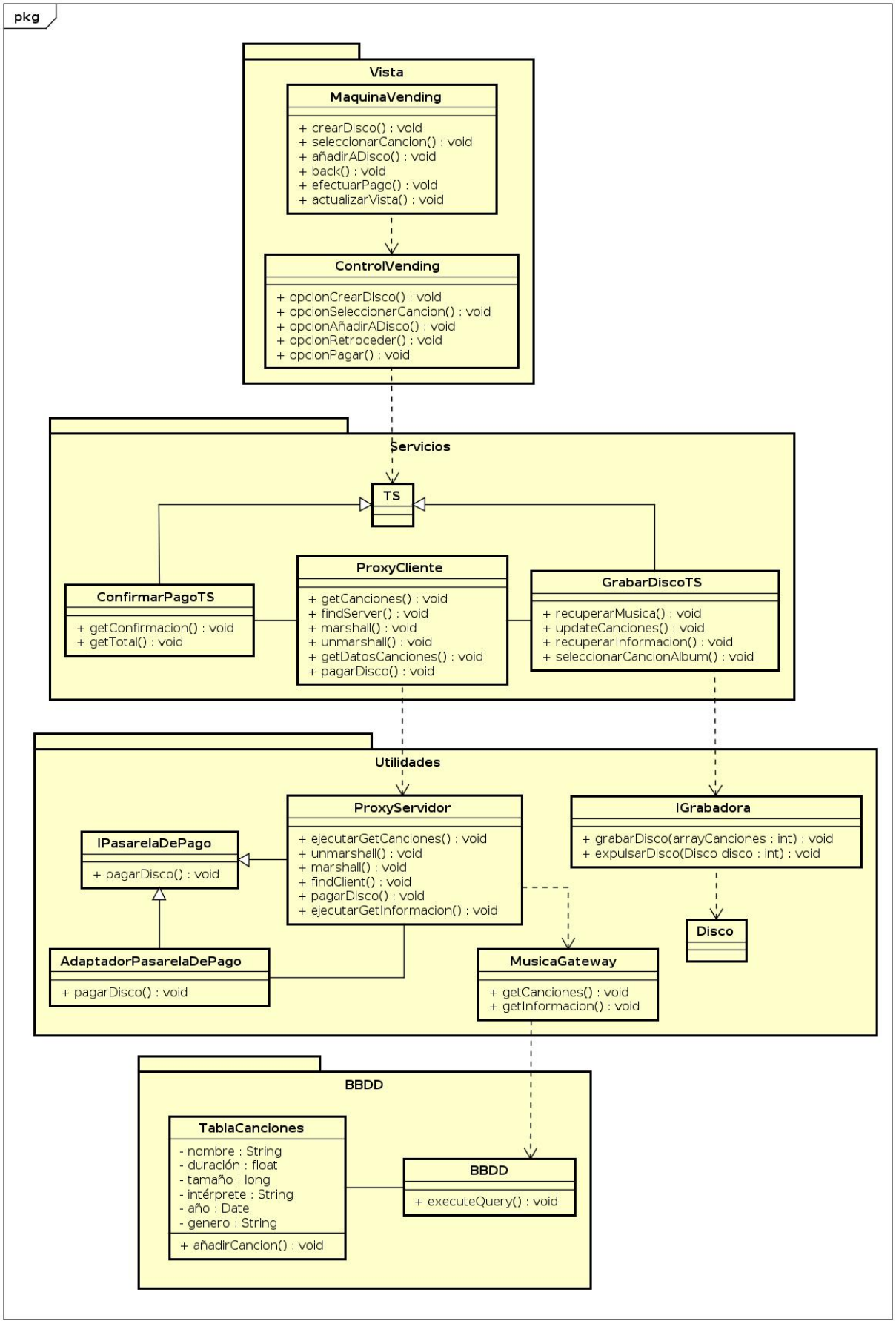

Figura 7: Diagrama de clases de Diseño (Caso uso grabar disco)

6. Mejoras futuras

Algunas mejoras futuras que podría incluir nuestro sistema son:

- **Patrón observador**: nosotros hemos hecho una especie de patrón observador, porque cada vez que tenemos que actualizar la vista, es el controlador el que se da cuenta de eso y se lo comunica a la vista. En futuras iteraciones del proyecto, podría implementarse el patrón observador completo. En el que hay una clase observer, a la que se suscriben la vista y el controlador y cuando algo cambia, es el observer el que le dice a vista que cambie.

- **Patrón compuesto**: este patrón lo que hace es tratar una serie de objetos de manera uniforme. Este patrón lo usaríamos para formar los álbumes, ya que pueden estar formado por simples canciones o por más álbumes. Para ese tratamiento recursivo de objetos, podemos usar perfecctamente un patrón compuesto. Así a la hora de seleccionar un álbum, podemos hacer de manera dinámica el tipo del álbum (si solo son canciones, si son otros álbumes mas canciones...). Así si en un futuro cambiá la definición de los álbumes, y se componen por más cosas (o menos), podríamos cambiarlo de manera rápida en nusetro diseño, simplemente tendríamos que poner o quitar más formas de composición.

- **Interfaces de paquetes**: podríamos definir también con claridad las interfaces de interacción entre los paquetes. Esto es así para que la interacción entre capas se haga mediante interfaces y con esto tendríamos un grado aún mayor de desacoplamiento. Podemos cambiar el contenido de la capa y adaptarlo para que use las interfaces que teníamos. De esta manera podríamos cambiar una capa sin que afecte absolutamente nada al resto de las capas.

- **Diagrama de despliegue**: mejorar el diagrama de despliegue especificando más componentes y subsistemas de nuestra empresa.

7. Bibliografía

[1] J.M Marqués. Diapositivas de la asignatura Diseño, Integración y Adaptación de Software.

[2] Martin Fowler. UML Destilled.

[3] Martin Fowler. Patterns of Enterprise Application Architecture.

[4] John Wiley. Pattern Oriented Osftware Architecture, A system of Patterns.

[5] Ian Sommerville. Ingeniería del Software.