UML: Diagramas de casos de uso y de clases

Entornos de Desarrollo

GRUPO 6
Elvira Medina Velilla
María Moreno Rodríguez
Álvaro Tercero Nieves
Tatiana Villa Ema

Índice

Pág.

Propuesta del equipo

Tras haber valorado la propuesta de cada miembro del equipo, la entrega final que hemos considerado presentar sería la siguiente:

Requerimiento 1

Diagrama Casos de Uso

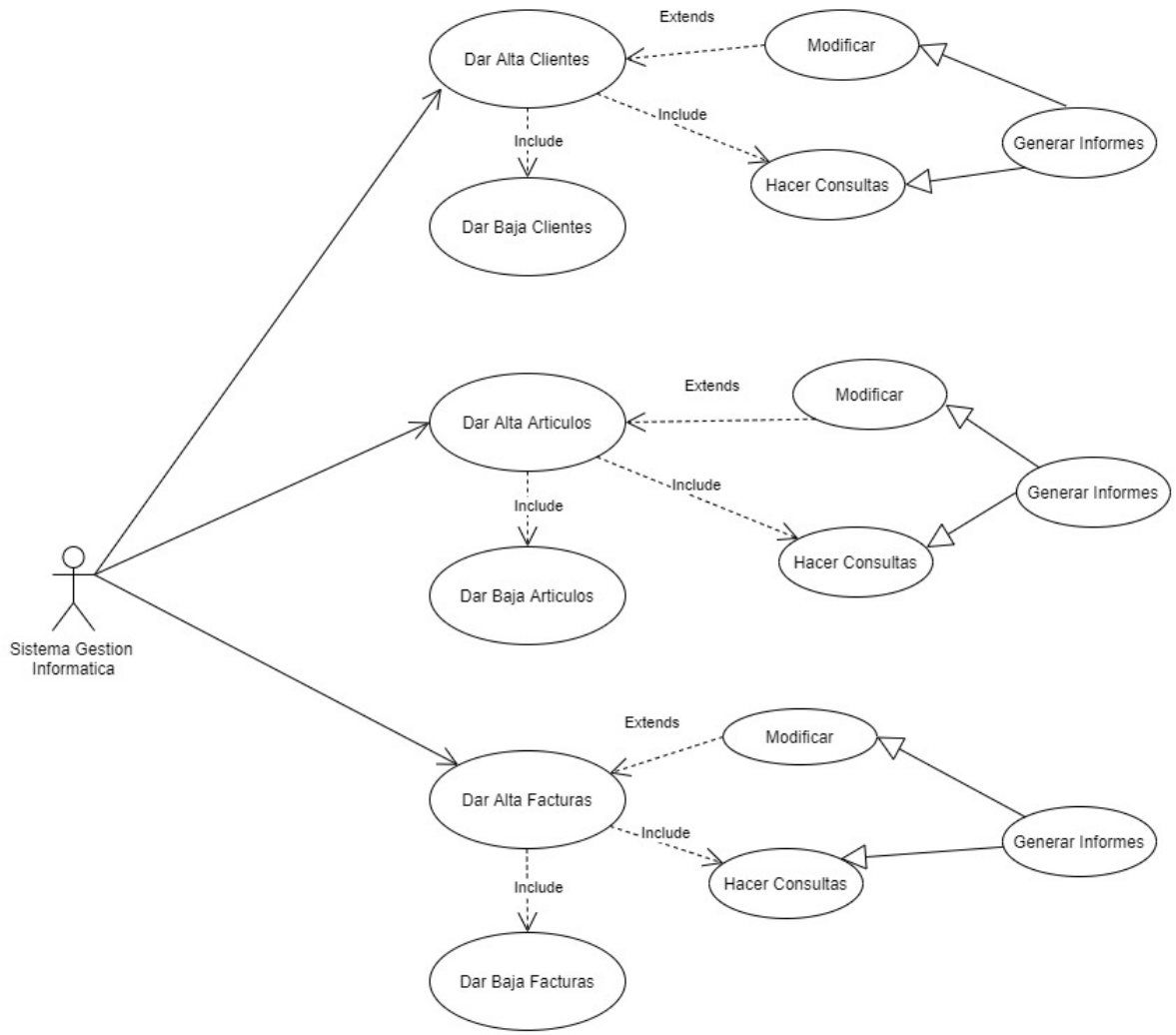
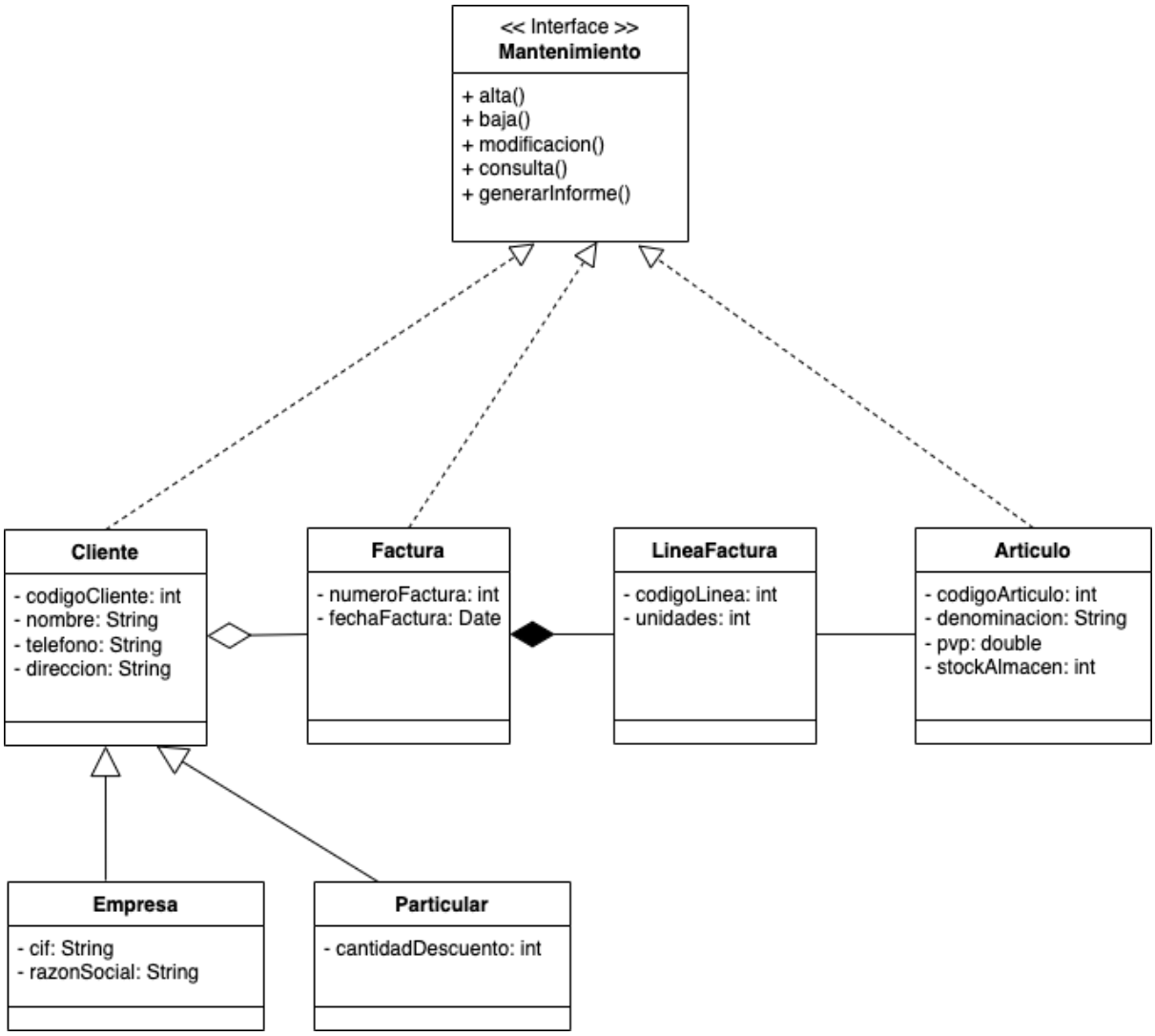


Diagrama de clases



Requerimiento 2

Diagrama Casos de Uso

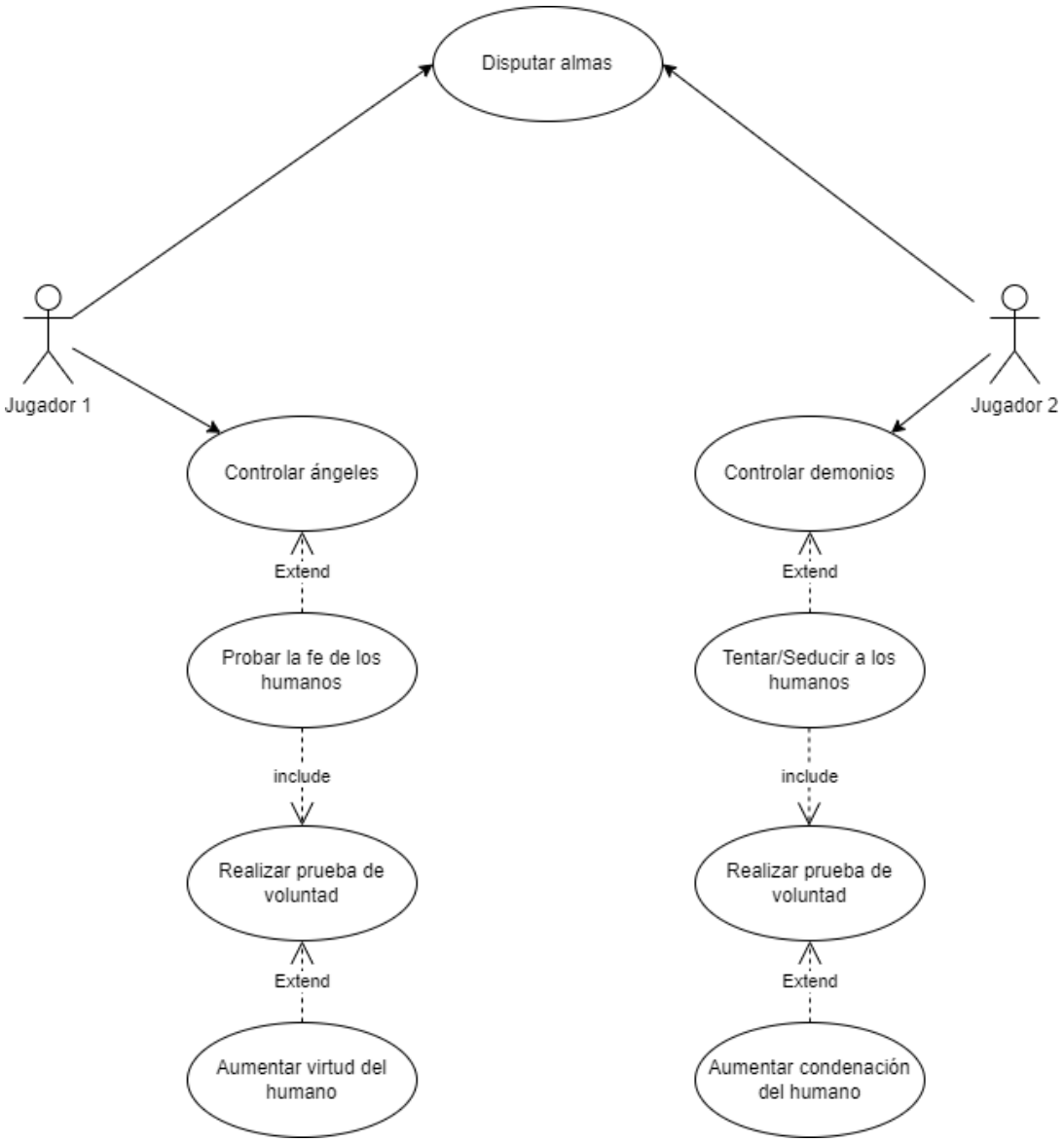
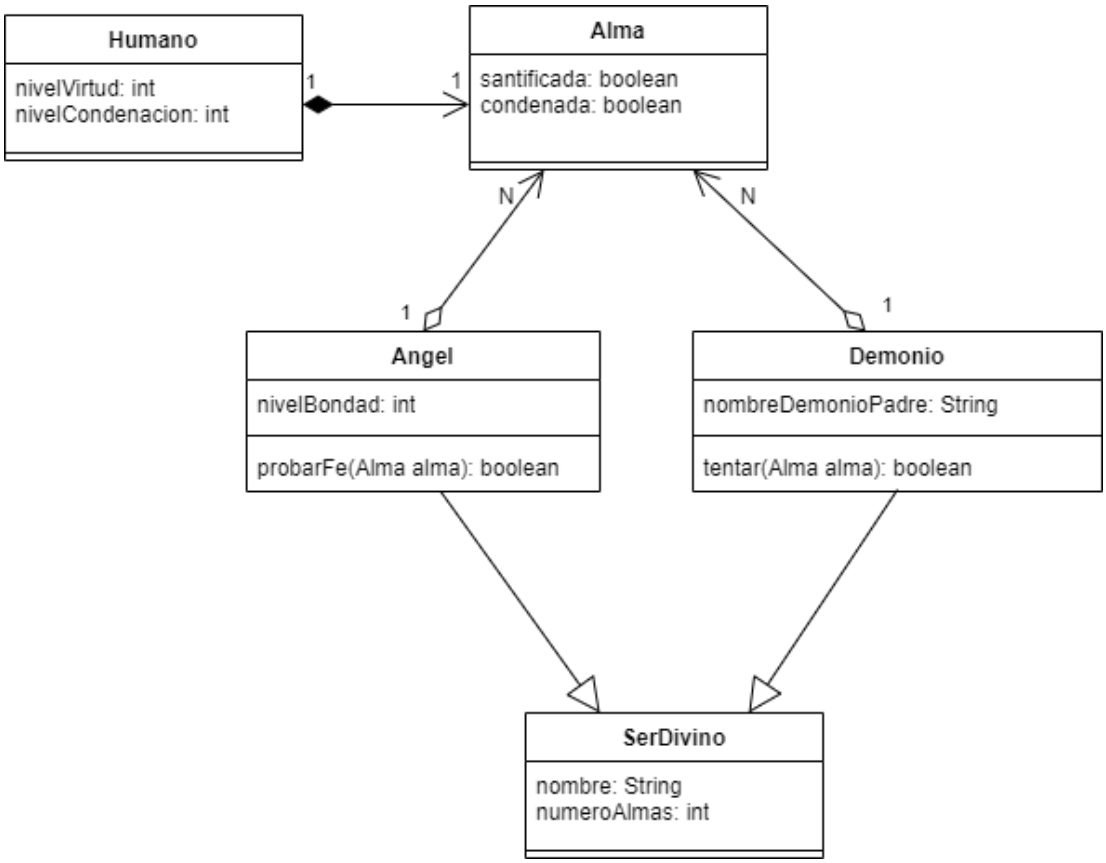


Diagrama de clases



Desarrollo en equipo

Para el desarrollo de esta actividad, hemos realizado una propuesta por cada miembro del equipo. Debido a la naturaleza de los requisitos, hemos decidido que cada miembro desarrollase su propia versión sin realizar ninguna consulta al resto, para que el resultado de cada uno fuese diferente.

En esta fase, cada persona ha tenido dudas en algún aspecto, ya que algunos enunciados podrían interpretarse de diferentes maneras. Al poner en común nuestro trabajo individual, cada uno ha explicado su punto de vista sobre estos aspectos ambiguos. Por ejemplo, en el Requerimiento 2 hemos debatido sobre las almas de los Humanos y su posible santificación o condenación.

El proyecto final presentado está compuesto por los siguientes aportes de cada miembro del equipo:

- El diagrama de casos de uso del Requerimiento 1 está realizado por : Elvira Medina.
- El diagrama de clases del Requerimiento 1 está realizado por: Álvaro Tercero.
- El diagrama de casos de uso del Requerimiento 2 está realizado por : María Moreno.
- El diagrama de clases del Requerimiento 2 está realizado por: María Moreno.

Propuesta de Elvira

Requerimiento 1

Diagrama de Casos.

En diagrama de casos de requerimiento 1, entiendo como actor principal el sistema de gestión Informática . Que se relaciona de forma directa con los casos de uso de "**Dar Alta Clientes"**, "**Dar Alta Artículo"** y "**Dar Alta Facturas".** Estos casos de uso tienen relación *include*, con el caso de uso **"Dar de baja"**. Porque necesariamente para poder dar de baja tienes previamente que haber dado de alta,esto ocurre tanto al cliente, al artículo, como a la factura.

Según avanza el texto del requerimiento, nos encontramos los casos de uso **"Modificar"** y **"Hacer Consultas"** cada uno en sus diferentes "ramas" clientes, artículos y facturas.

La relación que mantiene el caso "Dar de Alta" con el "Hacer consultas" es a través de una flecha con líneas discontinuas *include*, entendiendo que para hacer una consulta se tiene que realizar la alta. La flecha ira apuntando en dirección a e "Hacer Consultas". Se da igual para clientes, artículos y factura, mismos casos de uso y mismas relaciones.

Para la relación " Dar de Alta" con el caso de uso "Modificar" lo planteo a través de una flecha con líneas discontinuas *extend*s, pero esta vez, la flecha ira apuntando a "Dar de alta" ya que entendemos que del dar un alta podemos hacer o no una modificación.

A continuación tenemos otro caso de uso que seria el "Generar informes" que se relacionaría tanto con "Modificar", como con "Hacer Consultas" mediante una flecha con punta blanca. Lo entendemos como una herencia, porque de las consultas y las modificaciones, heredaríamos el generar esos informes , la punta de la flecha apuntaría a "Hacer Consultas" y a "Modificar". Se daría igual para los Clientes, Artículos y Facturas.

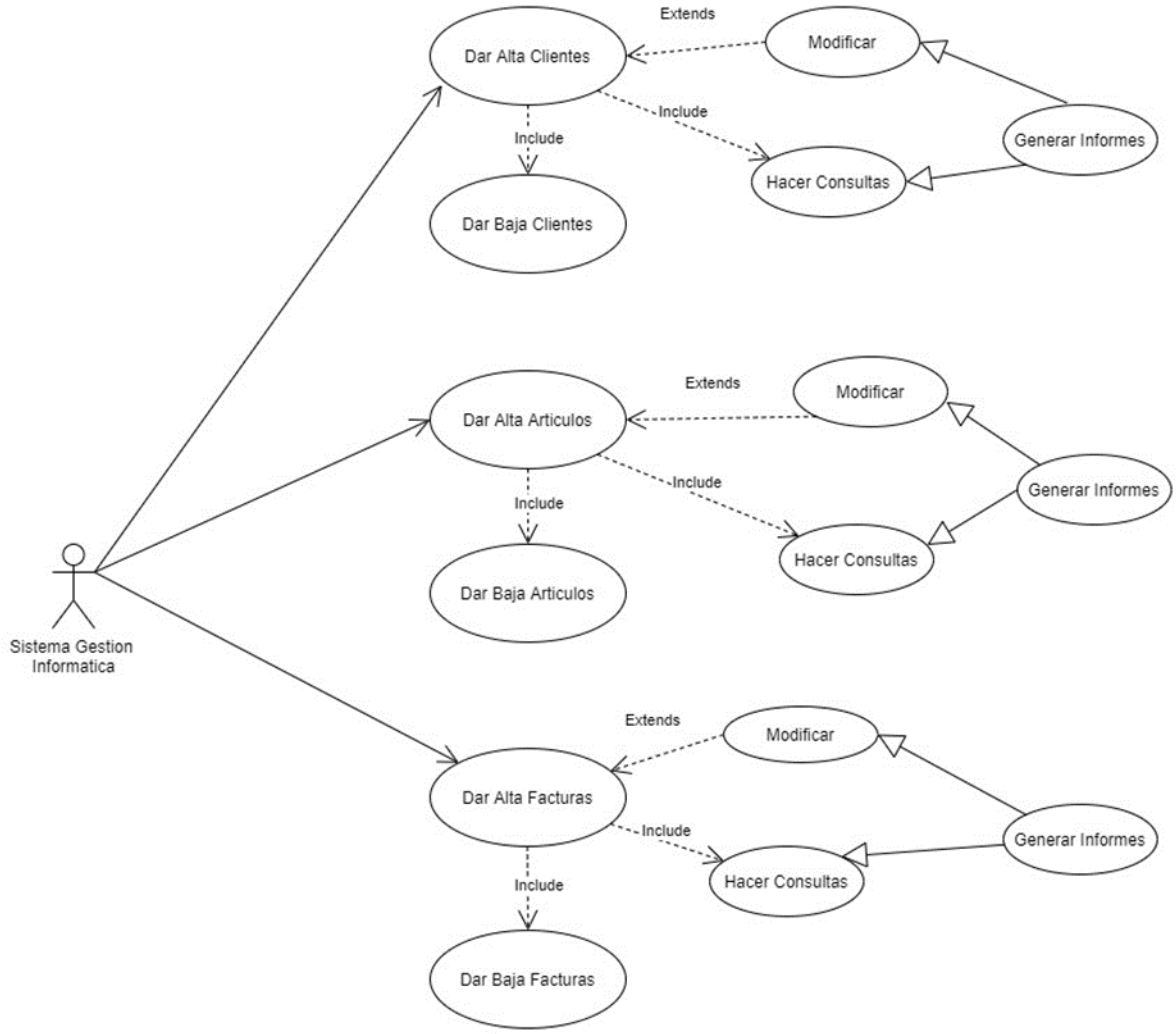


Diagrama de Clases

En el diagrama de clases del primer requerimiento he creado las **Clases Cliente, Particular, Empresa, Artículo, Factura y Líneas Factura** teniendo diferentes relaciones entre sí.

A continuación paso a detallar cuales han sido las relaciones entre cada una de ellas.

Tanto la Clase Empresa como la Clase Particular, están relacionadas las dos con la clase Clientes, por medio de una flecha con líneas discontinuas, representado la dependencia que hay entre cada una de ellas, con la Clase Cliente.

La Clase Empresa esta formada por dos atributos, uno seria el cif y otro la razonSocial, a su vez la Clase particular esta formada por el atributo cantidadDesuento.

La Clase Clientes, tiene relación con la Clase Facturas, mediante una flecha negra con un rombo negro, que indica una dependencia fuerte, ya que creo que no tiene sentido la existencia de la clase factura, si no vamos a tener determinados clientes que generen esas facturas.

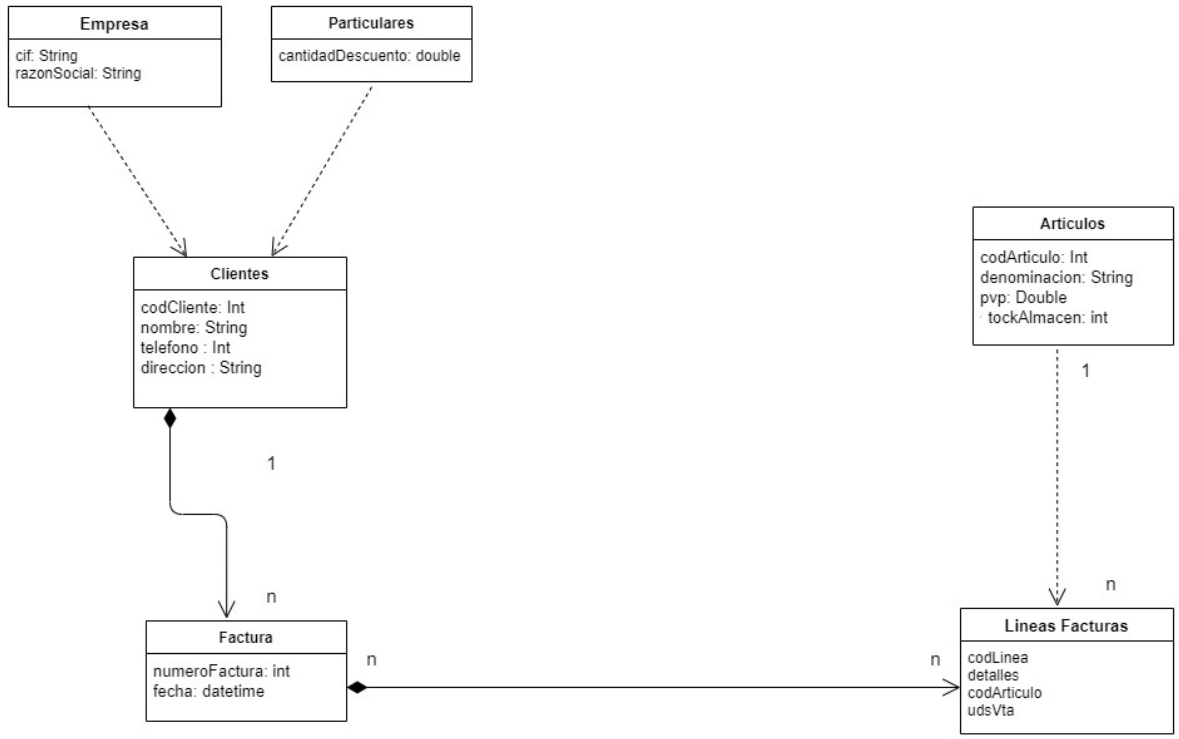
La Clase cliente esta formada por los atributos codCliente, nombre, teléfono y dirección .

La Clase Facturas a su vez se relaciona con la Clase Líneas de Factura, entendiendo esa relación, como dependencia fuerte, al no tener sentido la Clase Líneas de Factura, sin la Clase facturas de la que depende. Esta relación se representara con una flecha con un rombo negro. La Clase Facturas tiene los siguientes atributos: nuúmeroFactura y la fecha.

La Clase de Líneas de factura a su vez se relaciona con una flecha de línea discontinua con la Clase artículos, entendiendo que hay entre ellas una relación de dependencia. La flecha va dirigida hacia la Clase Artículos ya que podemos instanciar artículos dentro de las líneas de factura.

La Clase Líneas de factura contiene los siguientes atributos: codLínea, detalles, codArtículo, udsVta.

En cuanto a los atributos de la Clase Artículos son los siguientes: codArtículo, denominación, pvp y StockAlmacen.



Requerimiento 2

Diagrama casos de uso.

En las primeras líneas del requerimiento dos, denominado “La batalla de las almas” nos encontramos con los dos actores principales en el diagrama de clases que van a ser el **“Jugador 1”** y el **“Jugador 2”,** ambos actores se relacionan de forma directa con los dos primeros casos de usos que nos encontramos el **“Controlar Ángeles”** y **“Controlar a demonios”.** Partiendo de esos dos casos de usos llegamos al siguiente que sería **“Probar la voluntad”,** a este caso se llega por medio de una flecha de

líneas discontinuas *include* que nos indica que para llegar a la prueba de Voluntad tenemos o que controlar al demonio o al ángel.

Probar la Voluntad se relaciona, con una flecha de líneas discontinuas *include*, con el caso **"Probar la fe"**. Si probar la fe tiene éxito, heredamos el aumento de virtud. La relación entree el caso Probar la fe y **"Tener éxito"** es por medio de una relación *extend* entendiendo que al realizar la prueba de fe se puede tener éxito. En el caso de tener éxito heredaremos el **"Aumentar la virtud"**. (Generalización)

Por otra parte, tenemos la prueba de voluntad que se relaciona con el caso la "Probar Tentacion/Seduccion" (Podríamos tomarlo como dos casos de uso, pero lo vamos a tratar como uno) también por medio de una línea discontinúa *include*, entendiendo que hay q hacer la prueba de voluntad para poder hacer la prueba de tentar/seducir. Esta prueba puede "Sucumbir" (otro caso de uso) por medio de una relación *extends*, ya que puede que suceda o puede que no.

A su vez, en el caso de sucumbir, heredaremos el aumento de **"Condenacion"**.

En caso de tener éxito, o en el caso de sucumbir, llegaremos al objetivo del juego que es el **"Disputar el alma humana".**

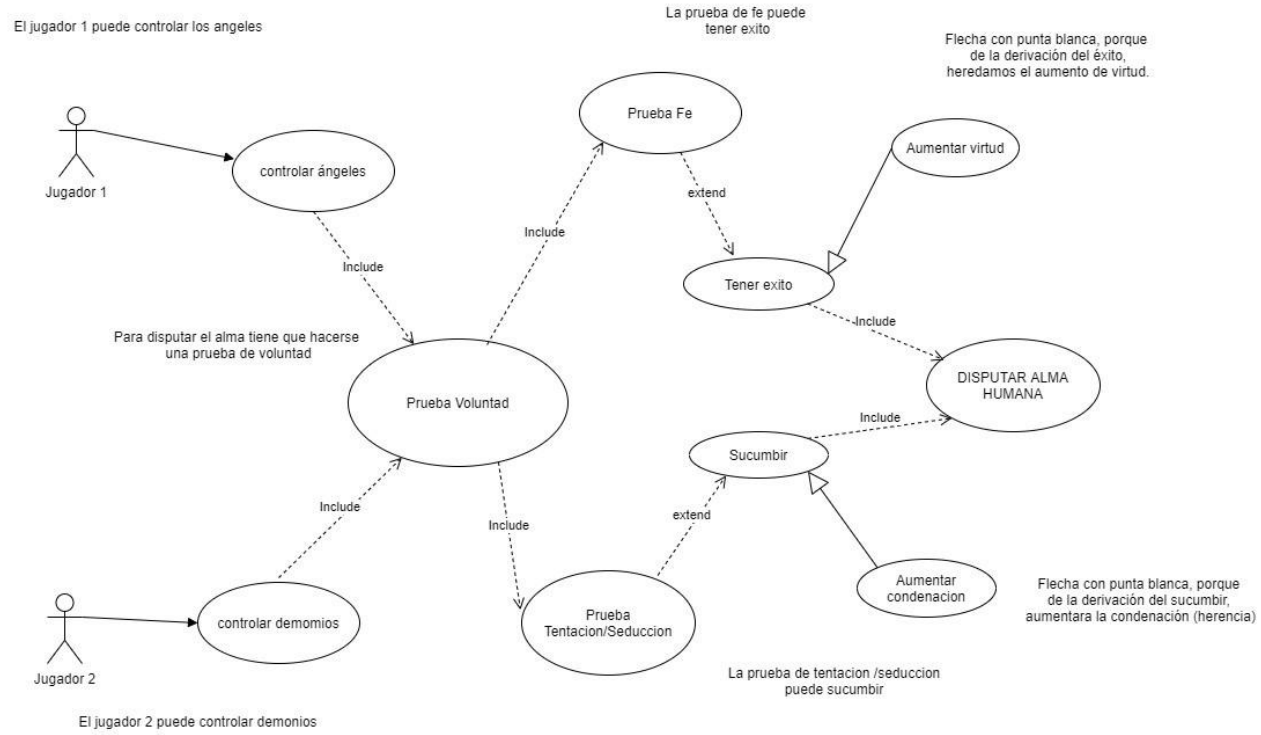


Diagrama de Clases de Clases

Tenemos las siguientes Clases: **Jugadores, Humanos, SerDivinoAngel y SerDivino Demonio**. La relación entre estas clases es la siguiente:

La Clase ser humano se relaciona con tanto con la clase SerDivinoÁngel como con la Clase SerDivinoDemonio mediante una línea bidireccional, que nos indica que tanto la Clase SerDivinoAngel como la Clase SerDivinoDemonio tienen una relación de uno a uno con la Clase Humanos.

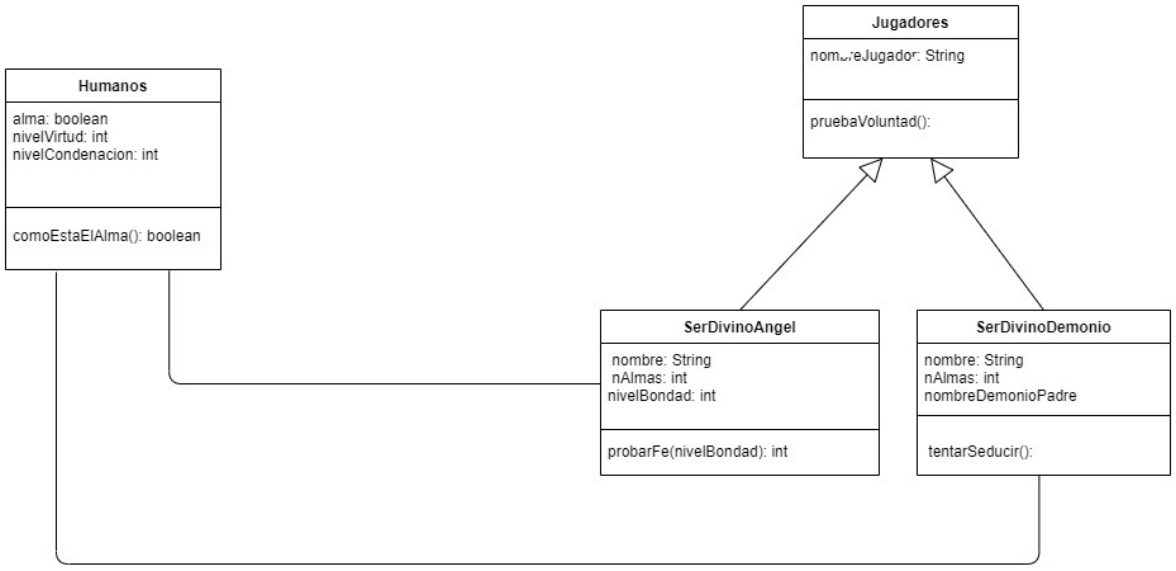
La Clase Humanos está compuesta por los siguientes atributos : alma, nivelVirtud y nivelCondencion. También tiene el método estadoAlma(): Boolean que nos indica si el alma es santificada (True) o Condenada (False)

Por otro lado, de la Clase jugador heredan tanto la Clase SerDivinoDemonio, como la Clase SerDivinoAlngel, está herencia la representamos con un flecha con punta blanca. Esta relación da a entender que un jugador es un ángel o un demonio.

La Clase Jugador está formada por el atributo nombreJugador y por el método pruebaVoluntad().

La Clase SerDivinoAngel tiene los siguientes atributos: nombre, nAlmas y nivelBondad(), también tenemos el método probarFe(nivelBondad) en el que le pasamos el nivel de bondad como parámetro y nos da la respuesta a la prueba de fe.

En la Clase SerDivinoDemonio tenemos los siguientes atributos: nombre, nAlmas y nombreDemonioPadre. Esta clase también contiene el método tentarSeducir()

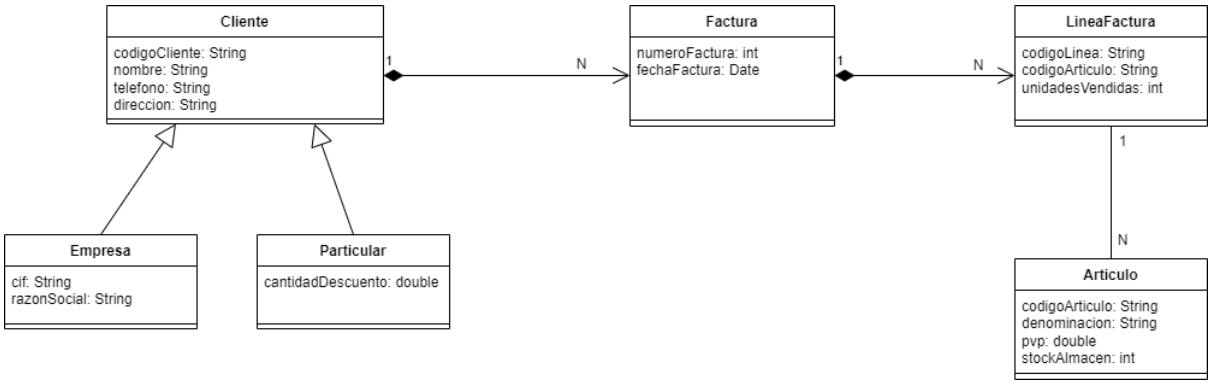


Propuesta de María

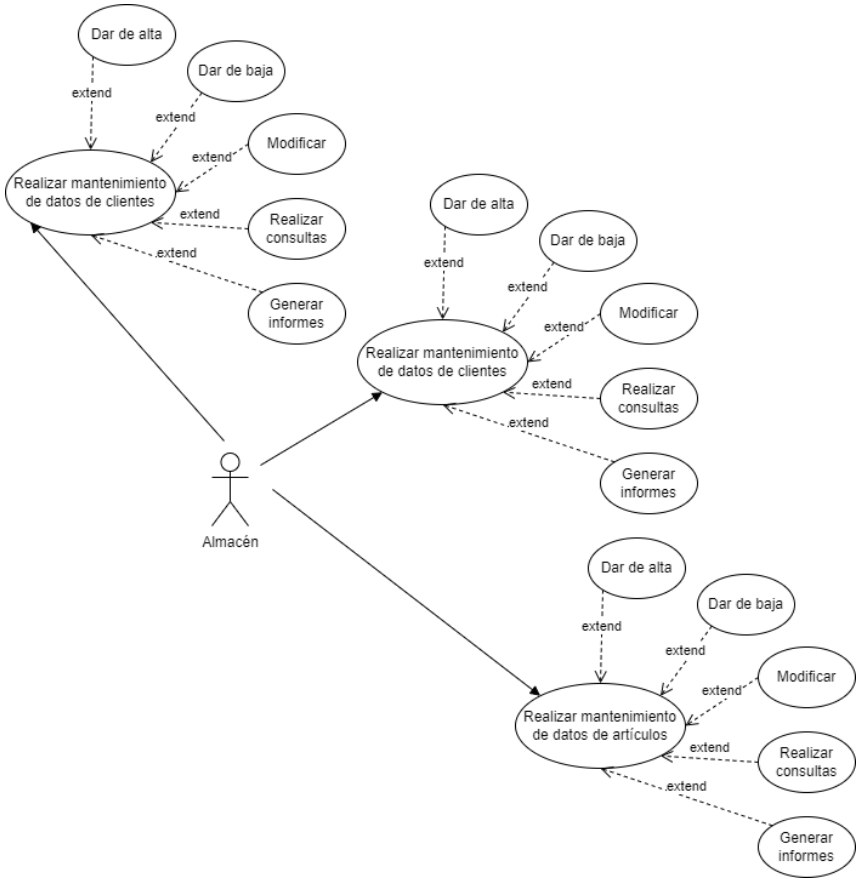
Requerimiento 1

Para realizar el diagrama de clases del requerimiento 1, he identificado 5 clases:

- La clase Cliente de la que heredan la clase Empresa y la clase Particular. La clase Cliente yo he entendido que posee una relación de dependencia fuerte con la clase Factura, puesto que considero que para ser cliente, tienes que haber realizado al menos una compra y, por tanto, poseer la correspondiente factura asociada a la misma.
- La clase Factura que se relaciona con la clase LíneaFactura mediante una dependencia fuerte, ya que cada factura deberá tener entre una y muchas líneas de factura.
- La clase LineaFactura, que se relaciona con la clase Artículo, mediante una relación de asociación.



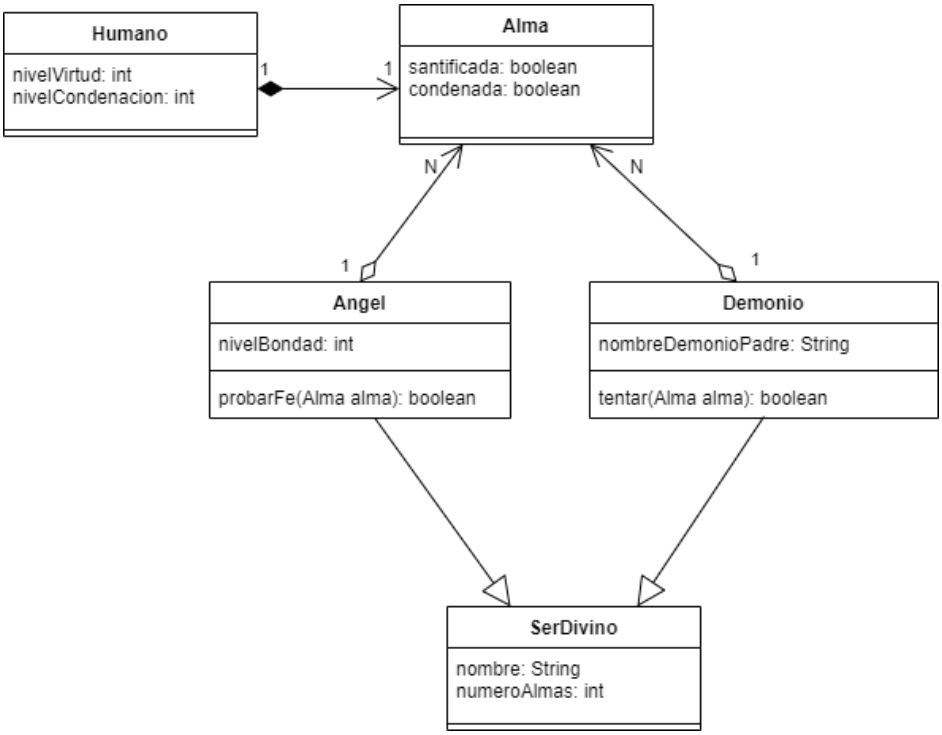
Para realizar el diagrama de casos de uso del requerimiento 1, he determinado que el actor (al que he denominado almacén) puede llevar a cabo 3 posibles casos de uso: realizar mantenimiento de datos de clientes, realizar mantenimiento de datos de facturas y realizar mantenimiento de datos de artículos. En los tres supuestos, se pueden realizar las siguientes acciones derivadas de los casos de uso principales: dar de alta, dar de baja, modificar, realizar consultas y generar informes. He determinado que la relación en todos los casos es de tipo extend puesto que son acciones optativas que derivan del caso de uso inicial.



Requerimiento 2

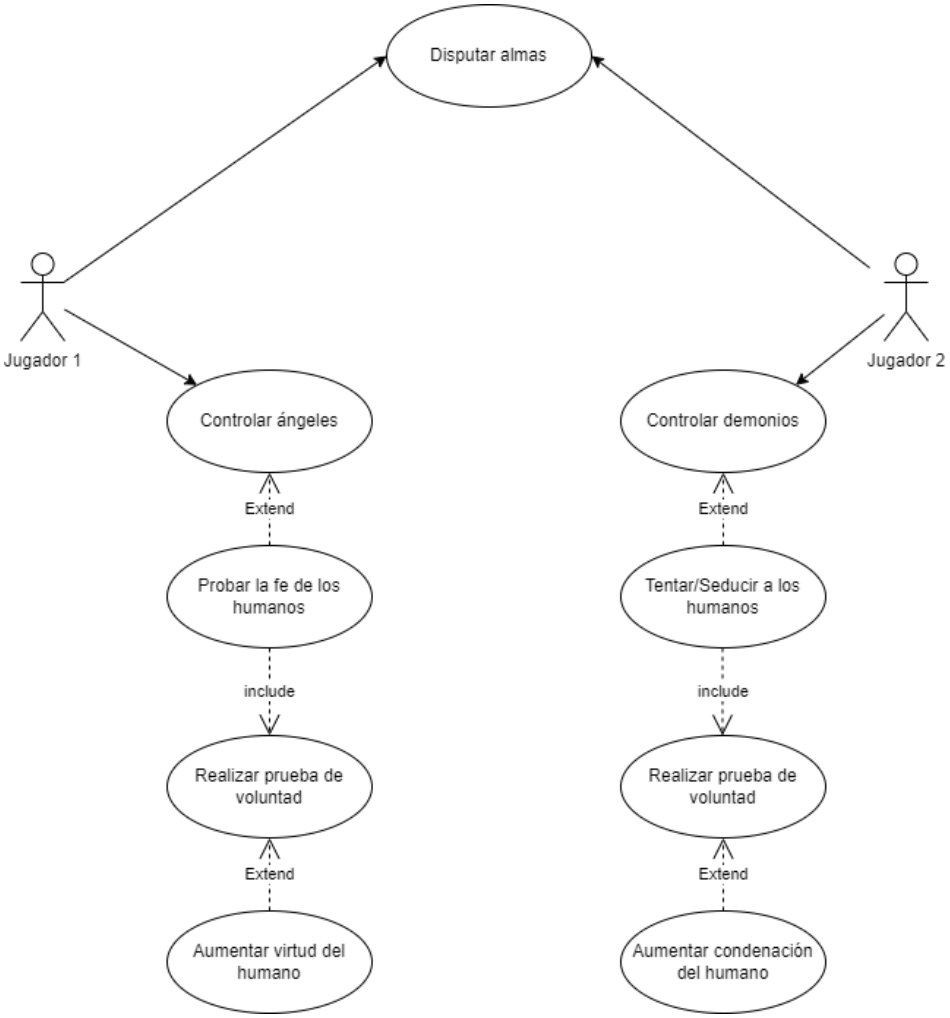
En este requerimiento, para realizar el diagrama de clases, yo he identificado 5 clases:

- La clase Humano, que se relaciona con Alma mediante dependencia fuerte, ya que considero que un humano tiene que tener un alma.
- La clase Alma, que se relaciona a su vez con las clases Angel y Demonio mediante dependencia débil, ya que según el enunciado, los ángeles y los demonios se deben disputar las almas y podrán o no obtenerlas. Este hecho, el que se disputen almas y no humanos como tal, así como que tengan el atributo numeroAlmas, el cual lo heredan de la clase SerDivino, es por lo que les he relacionado directamente con la clase Alma y no con la clase Humano.
- La clase Angel posee el método probarFe, al que le pasas un alma y según haya tenido éxito devuelve true o, en caso contrario, devuelve false.
- La clase Demonio, posee el método tentar, la que le pasas un alma y según haya tenido éxito devuelve true o, en caso contrario, devuelve false.
- La clase SerDivino, de la que heredan tanto la clase Angel, como la clase Demonio.



En este requerimiento, para realizar el diagrama de casos de usos yo he determinado que existen 2 actores, el jugador 1 y el jugador 2. Ambos pueden llevar a cabo el caso de uso disputar almas, puesto que es el objetivo principal del juego para ambos jugadores. Uno de los jugadores podrá controlar a los ángeles, estos podrán probar la fe de los humanos, y como esto podrá o no ocurrir, considero que la relación es de tipo extend. Para probar la fe de los humanos, se debe pasar una prueba de voluntad, por lo que en este caso la relación será de tipo include ya que es requisito obligado. Si todo este proceso es exitoso, esto resultará en un aumento de la virtud de los humanos; así que de nuevo, al ser esto una posibilidad, la relación sería de tipo extend.

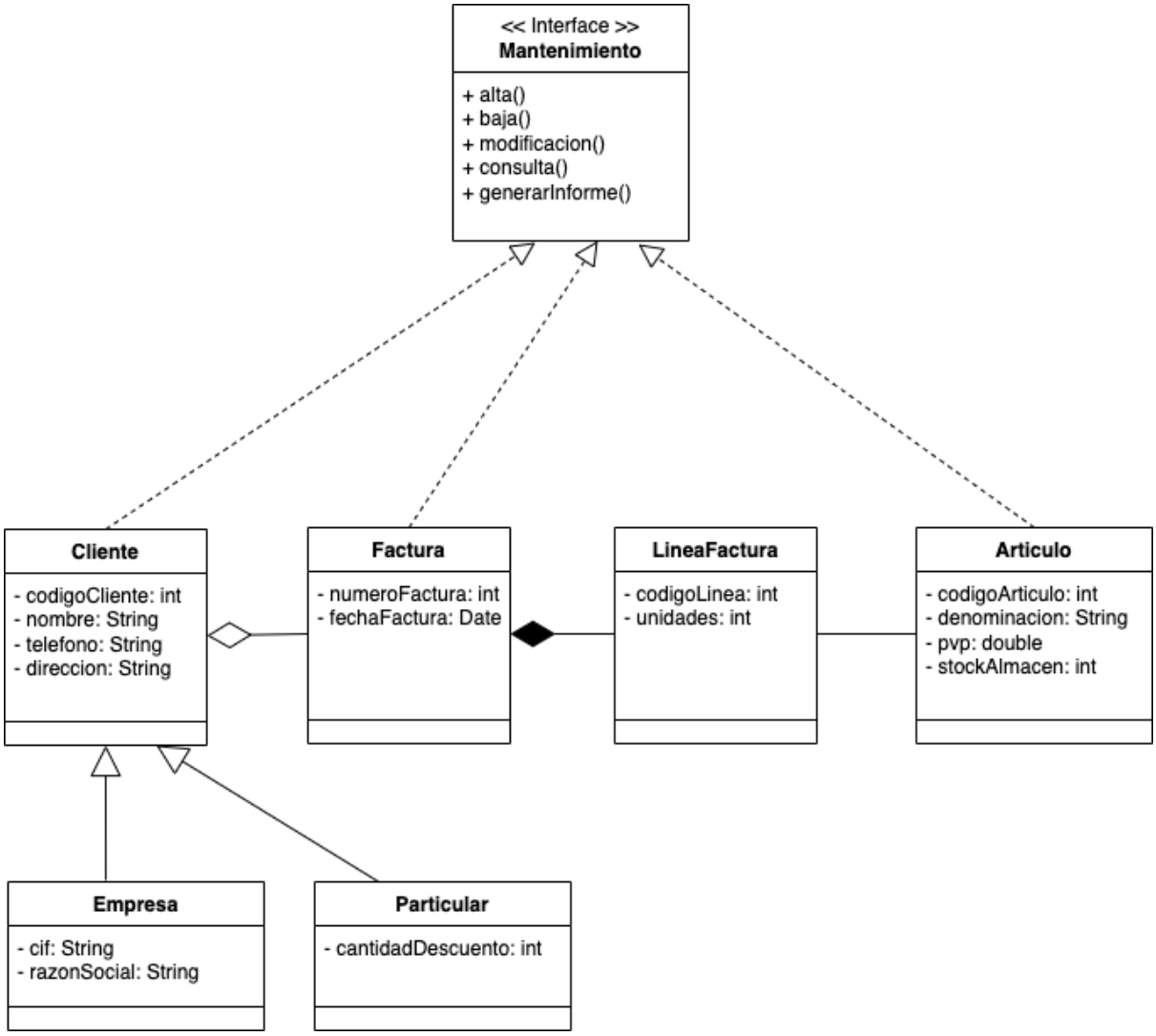
El otro jugador controlará a los demonios, estos pondrán tentar/seducir a los humanos, y como esto podrá o no ocurrir, considero que la relación es de tipo extend. Para seducir a los humanos, se deben pasar una prueba de voluntad, por lo que en este caso la relación será de tipo include ya que es requisito obligado. Si todo este proceso es exitoso, resultará en un aumento de la condenación de los humanos; así que al ser esto una posibilidad, la relación será de tipo extend.


Disputar almas
Jugador 1
Jugador 2
Controlar ángeles
Extend
Probar la fe de los humanos
include
Realizar prueba de voluntad
Extend
Aumentar virtud del humano
Controlar demonios
Extend
Tentar/Seducir a los humanos
include
Realizar prueba de voluntad
Extend
Aumentar condenación del humano

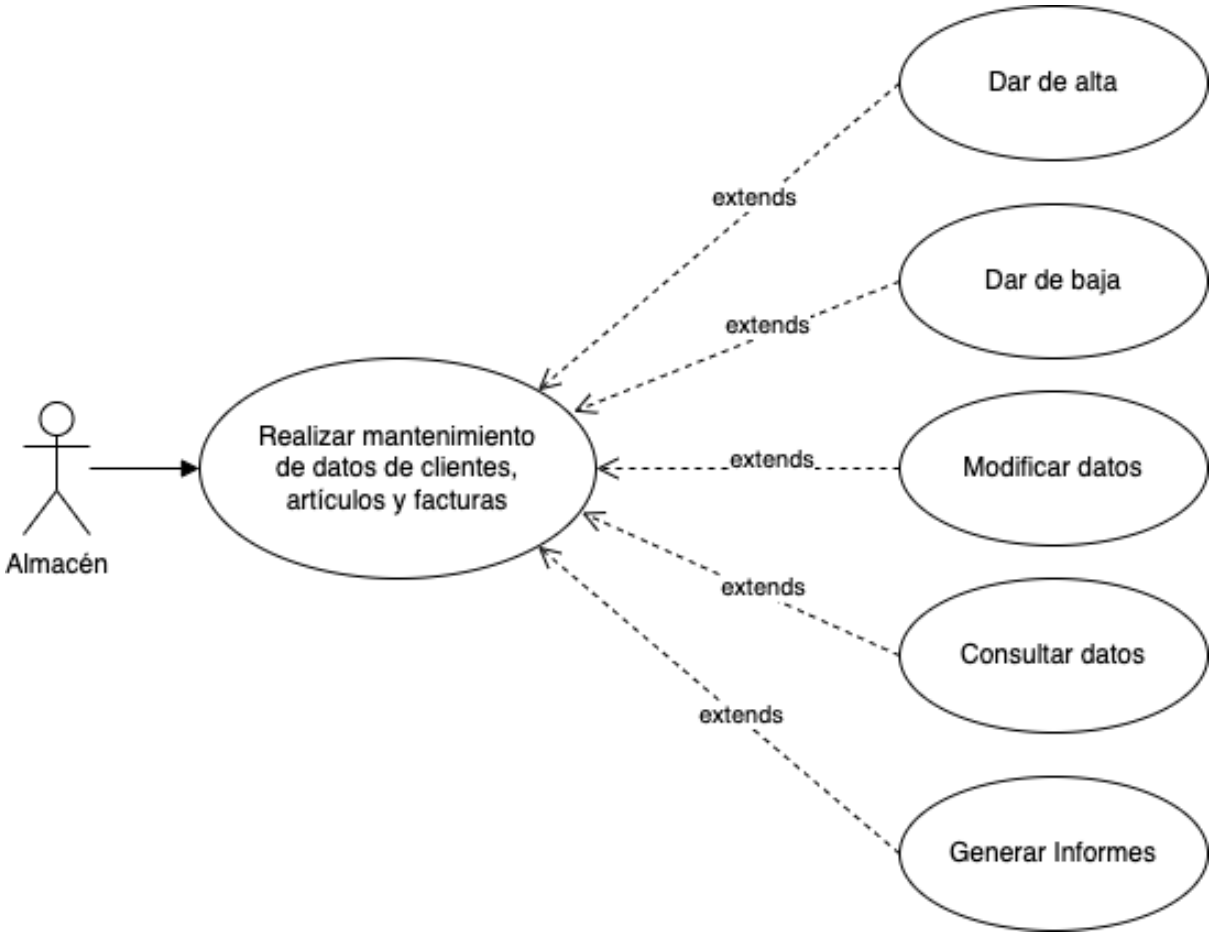
Propuesta de Álvaro

Requerimiento 1

En la realización del requerimiento 1 he tenido una visión muy clara sobre cómo quería hacer el diagrama de clases. He creado una interface que contiene los métodos de las clases Cliente, Factura y Artículo. Un cliente puede tener facturas o no, por lo que la relación es débil y se trata de una agregación. Pero una factura debe tener líneas de factura, no existiendo la segunda sin la primera, por lo que se trata de una composición. Cada línea de factura tiene un artículo, por lo que su relación sería una asociación. Además, las clases Empresa y Particular son tipos especiales de Cliente, por lo que la relación es de generalización.

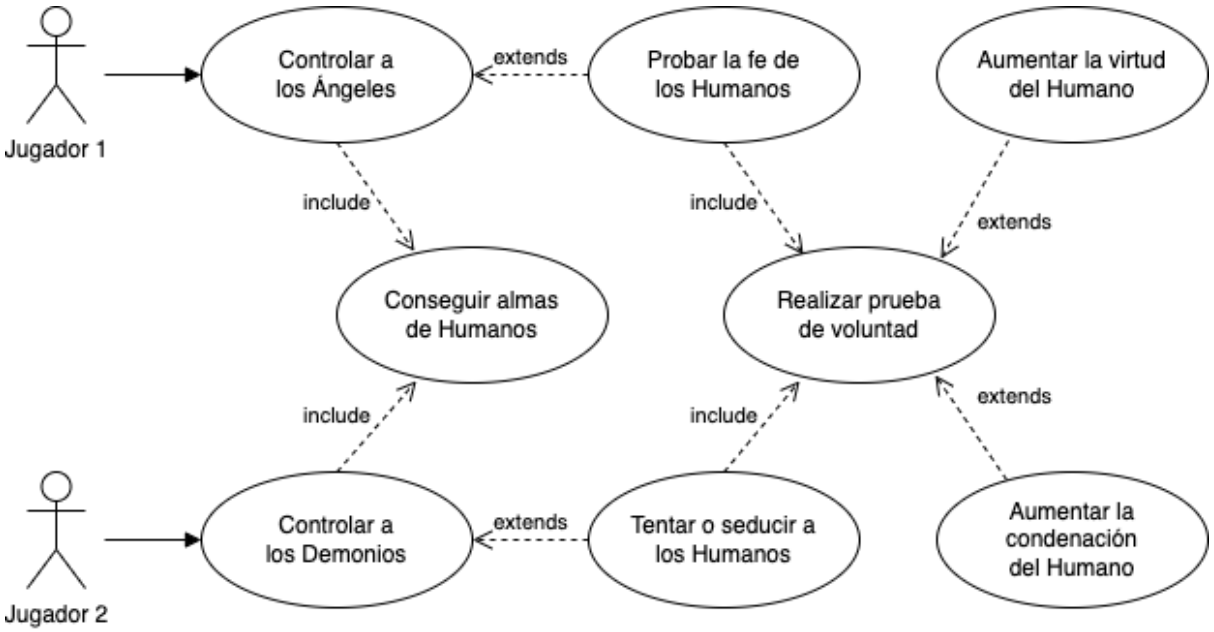


Sin embargo, en el diagrama de casos de uso apenas he encontrado información para su desarrollo. No se indica quién es el actor que utilizará el sistema de gestión informática, por lo que lo he denominado con un nombre genérico: almacén. El usuario realiza un mantenimiento de datos que afecta a tres clases: clientes, artículo y facturas. En cada una de ellas puede realizar cinco tareas: dar de alta, dar de baja, modificar datos, consultar datos y generar informes. Al ser optativas, ya que pueden ser utilizadas o no, se trata de posibilidades de casos de uso, utilizando la relación “extends”. He barajado dos opciones: o bien abrir tres ramas diferentes, una para cada clase, y enlazarlas con estas funciones, lo que daría lugar a un diagrama más lioso, o bien simplificarlo al agrupar las tres clases, ya que todas tienen las mismas acciones. Al final he optado por la segunda opción.



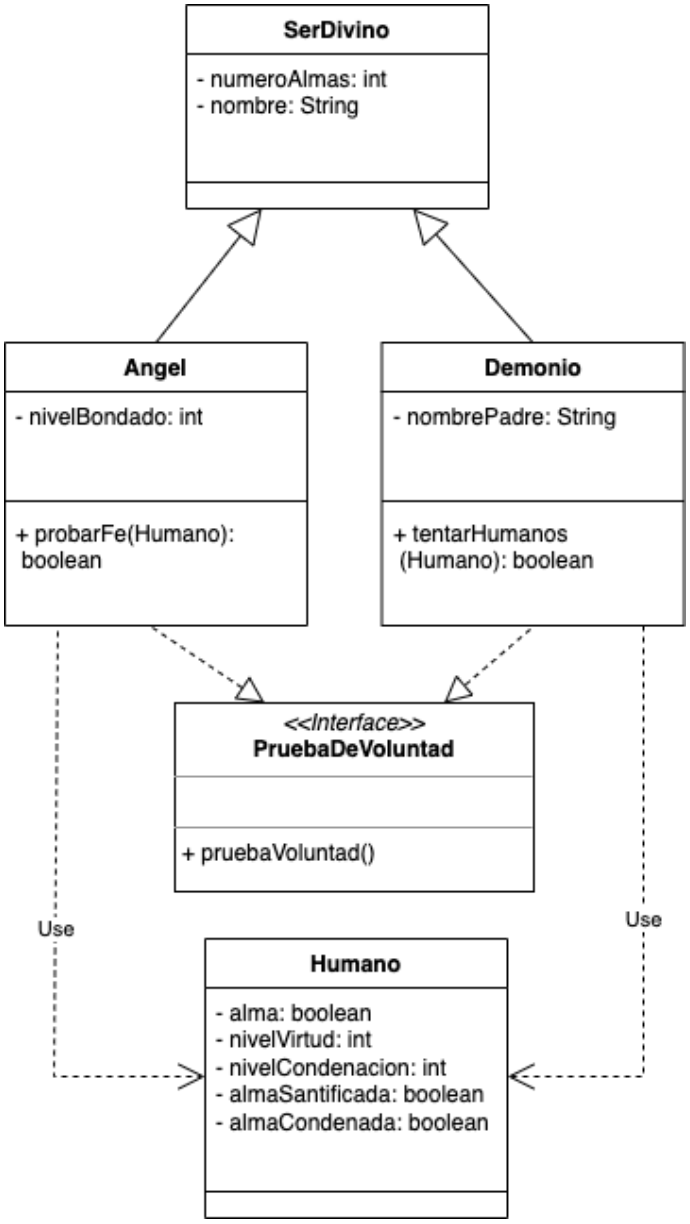
Requerimiento 2

En el Requerimiento 2 me ha pasado lo contrario: he tenido más claro el diagrama de casos de uso. En él he utilizado a dos actores: jugador 1 y jugador 2. Cada uno de ellos tiene una relación directa con dos casos de uso: controlar a los Ángeles o a los Demonios. Este control implica conseguir almas de Humanos, siendo obligatorio para el desarrollo del juego, por lo que su relación es de "include". En el control de Ángeles o Demonios hay dos casos respectivamente: probar la fe de los Humanos o tentarlos y seducirlos. Ambos son optativos, por lo que se relacionan con el anterior caso de uso con "extends". Pero si estos dos casos se realizan, tienen en común que deben realizar una prueba de voluntad, con una relación de "include". El resultado de dicha prueba da dos posibilidades: aumentar la virtud del Humano o aumentar su condenación, dos opciones que llevan una relación de "extends" al ser optativas.



Para elaborar el diagrama de clases del Requerimiento 2 he hecho una generalización de las clases Angel y Demonio en la clase SerDivino, ya que ambas comparten atributos. En Angel y en Demonio he utilizado dos métodos, respectivamente: probarFe y tentarHumanos. Cada uno de ellos utilizarían el método pruebaVoluntad, que implementan de la interface PruebaDeVoluntad. Tanto Angel como Demonio utilizan para sus métodos a Humano, por lo que su relación es de dependencia.

Con respecto a la clase Humano, he tenido ciertas dudas para su creación. En el enunciado se menciona que el Humano tendrá un alma. Por mi experiencia en otros videojuegos dudo sobre si el alma podría ser una propiedad intrínseca de este personaje, existiendo en él hasta que el mismo desaparezca, o si puede haber Humanos sin alma, como ocurre por ejemplo con los NPC de Dark Souls. Ante la duda, he añadido el atributo alma como boolean. Igualmente, había pensado en añadir el atributo estadoAlma como boolean, que verificase si su alma está santificada o condenada. Pero, aunque no se aclara en el enunciado, he pensado que sería posible que inicialmente el alma no estuviese ni santificada ni condenada. Por este motivo he incluido dos boolean: uno que verifique si está santificada (almaSantificada) y otro que verifique si está condenada (almaCondenada).


SerDivino
- numeroAlmas: int
- nombre: String
Angel
- nivelBondado: int
+ probarFe(Humano): boolean
Demonio
- nombrePadre: String
+ tentarHumanos (Humano): boolean
<<Interface>>
PruebaDeVoluntad
+ pruebaVoluntad()
Use
Use
Humano
- alma: boolean
- nivelVirtud: int
- nivelCondenacion: int
- almaSantificada: boolean
- almaCondenada: boolean

Propuesta de Tatiana

NO HAY PROPUESTA