Exercicis de Model de Domini

Disseny de software

Contingut

1. Diari

En un diari es desitja automatitzar el procés de producció, escriptura i publicació de notícies. S'ha decidit que la única persona que utilitza el programa que es desitja construir serà la secretària del cap de redacció. El procés al que es desitja donar suport és el següent:

- Diàriament, a primera hora del matí es reuneix el consell de redacció i decideix quines notícies van a aparèixer en l'edició d'aquest dia.

- La Redactora Cap comunica a la secretària de redacció quines són les notícies que es van incloure en l'edició d'aquest dia i qui és el redactor que es va a encarregar de redactar cada notícia. Com els titulars de cada notícia encara no es coneixen (l'ha de fixar el redactor de la notícia) a cada notícia se li assignarà un titular intern que ens permetrà saber a quina notícia ens referim.

- La Secretària crea una nova edició del diari (indicant la data actual), introdueix en l'edició del dia les notícies i introdueix en el sistema, per a cada notícia, el redactor que es farà càrrec. Assumim que els redactors no canviaran i es poden identificar per el seu D.N.I. De cada redactor hem de emmagatzemar el seu nom i cognom.

- La Secretària sol·licita al sistema la creació de les fulles de treball, una per cada redactor, en les quals apareixerà la llista de notícies que li han estat assignades. Per a cada notícia, en el llistat apareixerà el titular intern i l'identificador de notícia, que serà únic per cada noticia. També, apareixeran com a camps per emplenar, el titular real de la notícia, l'antetítol i el text de la notícia.

- La Secretària repartirà cada fulla de treball al redactor corresponent.

- La Secretària rebrà dels redactors les fulles de treball amb totes les notícies emplenades.

- La Secretària introduirà en el sistema el titular, antetítol i text de cada notícia.

- La Secretària obtindrà un llistat amb tota la informació de totes les notícies de l'edició i l'hi lliurarà a la Cap de Redacció. En el llistat, apareixerà, com a camp a emplenar, el número de pàgina al que es desitja assignar la notícia.

- El Cap de Redacció indicarà el número de pàgines de l'edició del diari (habitualment 8) i emplenarà sobre el llistat els números de pàgina que corresponen a cada notícia, així com la posició en la pàgina en cas de considerar-ho oportú (a dalt, a baix, dreta, esquerra). Una vegada fet això retornarà al llistat omplert a la Secretària.

- La Secretària introduirà en el sistema la informació de paginació.

- La Secretària tancarà l'edició i a partir d'aquest moment, l'edició passarà a dependre d'impremta.

Es sol·licita:
1. Realitzar el model de domini.
2. Modificar el model de domini tenint en compte que una notícia pot aparèixer a diverses pàgines. En aquest cas, considerarem que la notícia es divideix en diversos fragments. Cada fragment només pot aparèixer en una pàgina.

2. Lloguer de Videojocs

L'empresa de lloguer de videojocs (EnjoyGames) té un local d'atenció al públic on estan exposades les caràtules dels videojocs més demandats i les últimes novetats, encara que també existeixen llistats en paper de tots els títols que es podran llogar. Quan un client sol·licita un títol, es comproven si hi ha exemplars lliures i si no hi ha problemes per exemplars no retornats es realitza el lloguer, quedant constància de la data de lloguer i la data màxima de lliurament; de manera que quan el client retorni l'exemplar es podrà comprovar si se li ha d'imposar una sanció. Cada client pot sol·licitar una relació dels videojocs que ha llogat prèviament.

Es sol·licita construir el model de domini utilitzant UML i completa-ho amb una descripció que faciliti la seva comprensió.

3. Professors

Un alumne assisteix a cursos. Els cursos estan impartits per un únic professor. L'alumne no pot repetir el mateix curs, però pot assistir a més d'un curs. El professor pot impartir diferents cursos i repetir un mateix curs en diverses ocasions. Perquè un curs s'imparteixi ha d'haver-hi un mínim de 10 alumnes i un màxim de 50. Com a registre del curs es guarda la data de començament, la data de finalització i la nota de l'alumne.

Es sol·licita construir el model de domini utilitzant UML i completant-ho amb una descripció que faciliti la seva comprensió.

4. La Lliga de Futbol

Ens sol·liciten un programa per gestionar la informació sobre els resultats de la lliga de futbol. De les diferents categories (de les quals sabem el nom, com a primera divisió, segona divisió, etc.) ens guardarem els resultats de totes les jornades. Cada jornada s'identifica per un número de jornada i en ella es disputen diversos partits, dels quals es coneix la data del partit, l'hora d'inici, l'estadi, la capacitat de l'estadi, el número de gols de l'equip visitant i número de gols de l'equip local.

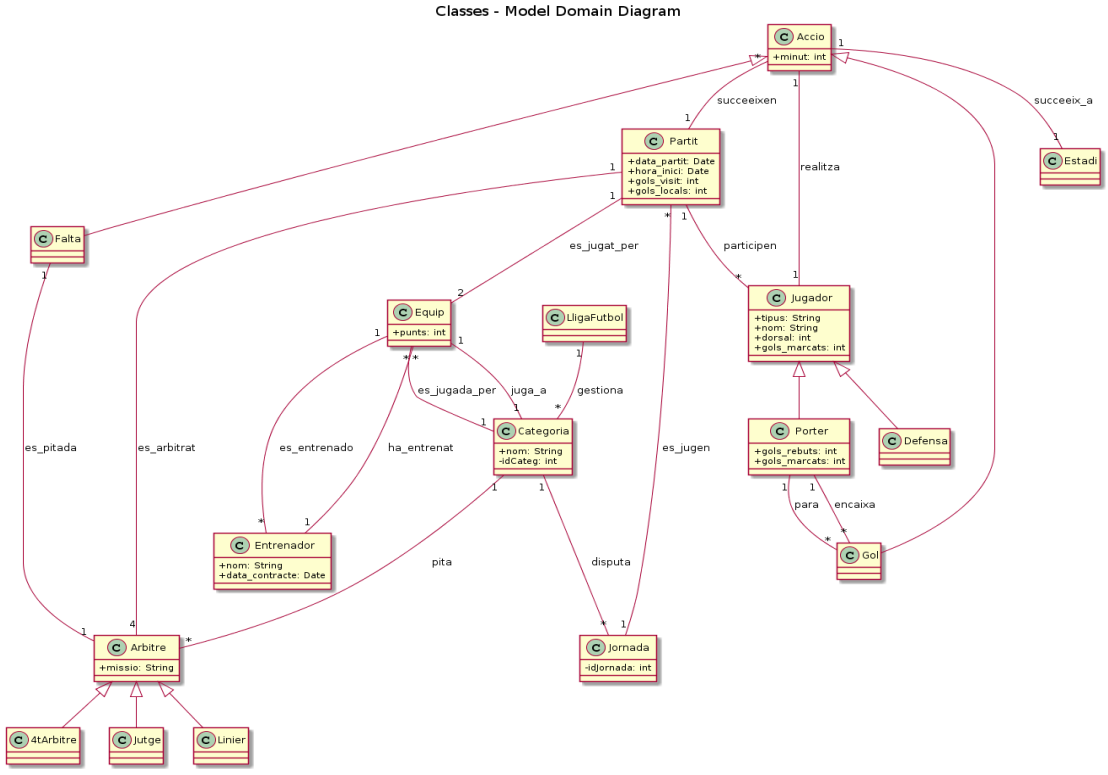
En cada partit participen 4 àrbitres (on cadascun té una missió, 2 fan de liniers, 1 de àrbitre i un altre de quart àrbitre), un equip local i 3 un equip visitant. Els àrbitres i els equips només poden xiular o jugar en una categoria alhora. Durant un partit succeeixen diverses accions com a gols, faltes, corners, etc., de cadascuna d'elles es vol conèixer el minut en què va ocórrer i el jugador que la va realitzar. De les faltes es vol saber el àrbitre que la va xiular. En un partit, l'equip guanyador suma 3 punts, el perdedor 0 punts i un empat suma 1 punt per a cada equip. Un equip està format per jugadors i pot tenir diversos entrenadors.

Durant la lliga ,un jugador roman en el mateix equip. D'un entrenador sabem el nom, la data en què va ser contractat i els equips on anteriorment ha entrenat. Un jugador pot ser defensa, lateral, porter, etc., cadascun té un nom, un dorsal assignat i el recompte dels gols marcats en la lliga, en el cas de ser un porter també es coneix el total de gols rebuts i aturats.

Es sol·licita construir el model de domini utilitzant UML i completant-ho amb una descripció que faciliti la seva comprensió. Indiqueu els tipus dels atributs en el model de domini.

Solució:

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Arbitre <|-- 4tArbitre
Arbitre <|-- Jutge
Arbitre <|-- Linier
Jugador <|-- Defensa
Jugador <|-- Porter
Accio <|-- Gol
Accio<|-- Falta

LligaFutbol "1" -- "*" Categoria: gestiona
Equip "1"--- "*" Entrenador : es_entrenado
Entrenador "1"---"*" Equip: ha_entrenat
Categoria "1" --- "*" Arbitre : pita
Categoria "1" --- "*" Jornada : disputa
Categoria "1" -- "*" Equip : es_jugada_per
Equip "1" -- "1" Categoria : juga_a
Jornada "1" -- "*" Partit : es_jugen
Partit "1" -- "4" Arbitre : es_arbitrat
Falta "1" -- "1" Arbitre : es_pitada
Partit "1" -- "2" Equip : es_jugat_per
Partit "1" -- "*" Jugador : participen
Jugador "1" -- "1" Accio : realitza
Accio "*" -- "1" Partit : succeeixen
Accio "1" -- "1" Estadi : succeeix_a
Porter "1" -- "*" Gol: encaixa
Porter "1" -- "*" Gol: para
class LligaFutbol
class Entrenador{
    + nom: String
    + data_contracte: Date
}
class Categoria{
    + nom: String
    - idCateg: int
}
class Jornada{
    - idJornada: int
}
class Arbitre{
    + missio: String
}
class Equip{
```



5. Campionat de Formula1

La Federació Internacional d'Automobilisme (FIA) ens sol·licita una aplicació per la gestió del campionat mundial automobilístic, conegut com la Formula 1 o F1.
El campionat de Formula 1 està format per diferents escuderies (també denominades equips) que participen en totes les carreres del campionat.

Una escuderia té un nom, un país d'origen, un número de punts, un conjunt de mecànics i dos pilots: el primer pilot i el segon pilot que corren en totes les carreres del campionat1. Els mecànics es divideixen en enginyers mecànics cap, mecànics ordinaris i mecànics especialistes. Els enginyers mecànics cap, s'encarreguen de la gestió d'un monoplaça i la resta de mecànics s'encarreguen de la posta a punt i reparació dels monoplaça. Tant dels mecànics com dels pilots es coneix el seu nom, any de naixement i país on va néixer. Cada pilot té el seu propi enginyer mecànics cap i un comptador de punts. Una escuderia té un o diversos patrocinadors que paguen una quantitat de diners en concepte de publicitat en el monoplaça.

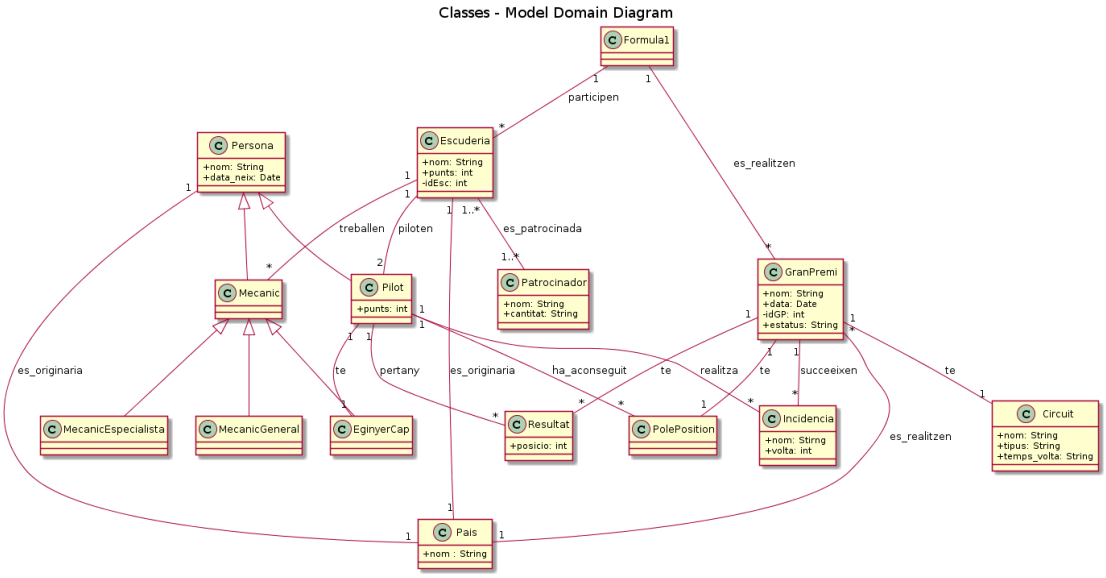
Els patrocinadors poden pagar publicitat en diverses escuderies. Cada carrera té un nom, el país sortida, el circuit i un estat de la carrera que pot ser pendent, realitzada, acabada. Amb l'estat pendent s'indica que el gran premi no s'ha realitzat, amb realitzada s'indica que la prova s'ha realitzat però no s'han actualitzat els punts als guanyadors i amb l'estat acabada s'indica que la prova ha finalitzat completament i els punts s'han actualitzat tant a les escuderies com als pilots.

Els circuits de les carreres poden ser urbans o permanents. Tots els circuits tenen un nom i el temps de la volta més ràpida de tota la seva història. En cada carrera es guarda la informació dels resultats de classificació de les incidències esdevingudes durant la celebració de la mateixa. La classificació només guarda el pilot i la posició final en la qual ha quedat.
Les incidències han d'indicar el nom de la incidència, quin pilot l'ha comès i la volta en la qual ha succeït.

Es sol·licita construir el model de domini utilitzant UML i completant-ho amb una descripció que faciliti la seva comprensió. Indiqueu els tipus dels atributs en el model de domini.

Solució

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Persona <|-- Mecanic
Persona <|-- Pilot
Mecanic <|-- MecanicGeneral
Mecanic <|-- EginyerCap
Mecanic <|-- MecanicEspecialista

Formula1 "1" -- "*" Escuderia: participen
Formula1 "1"--- "*" GranPremi : es_realitzen
GranPremi "*"---"1" Pais: es_realitzen
Escuderia "1" --- "1" Pais : es_originaria
Persona "1" --- "1" Pais : es_originaria
Escuderia "1..*" -- "1..*" Patrocinador : es_patrocinada
Escuderia "1" -- "*" Mecanic : treballen
Escuderia "1" -- "2" Pilot : piloten
Pilot "1" -- "1" EginyerCap : te
Pilot "1" -- "*" PolePosition : ha_aconseguit
Pilot "1" -- "*" Resultat : pertany
Pilot "1" -- "*" Incidencia : realitza
GranPremi "1" -- "1" PolePosition : te
GranPremi "1" -- "*" Resultat : te
GranPremi "1" -- "*" Incidencia : succeeixen
GranPremi "1" -- "1" Circuit: te

class Formula1
class Pais{
    + nom : String
}
class Patrocinador{
    + nom: String
    + cantitat: String
}
class Escuderia{
    + nom: String
    + punts: int
    - idEsc: int
}
class GranPremi{
    + nom: String
    + data: Date
    - idGP: int
    + estatus: String
}
class Circuit{
    + nom: String
    + tipus: String
    + temps_volta: String
}
class PolePosition
class Persona{
    + nom: String
    + data_neix: Date
}
class Pilot{
    + punts: int
}
class Mecanic
class Resultat{
    + posicio: int
}
class Incidencia{
    + nom: Stirng
    + volta: int
}
class MecanicGeneral
class MecanicEspecialista
class EginyerCap
```



6. Eleccions Municipals

L'empresa UBSoft ens ha demanat fer una aplicació pilot per gestionar els vots i els resultats de les eleccions municipals en el país. Encara que siguin eleccions municipals, es volen fer estadístiques de vots a nivell autonòmic.

Per aquest motiu, l'aplicació considera les diferents comunitats autònomes i els municipis que pertanyen a cadascuna d'elles (s'ha decidit ometre províncies i qualsevol altre tipus de divisió territorial). Cada municipi disposa de diferents taules electorals, cadascuna es distingeix segons el número que la identifica. A més, cada taula electoral té el seu cens electoral. Aquest cens electoral aquesta format per ciutadans dels quals sabem el seu nom, DNI, edat, professió i la direcció on resideixen en el municipi. Un ciutadà només pot anar a votar a la taula electoral que té assignada. En la taula electoral participen tres ciutadans, dues vocals i un president, cap d'ells pot ser candidat en una llista electoral. Un ciutadà solament pot votar si consta en el cens electoral i només pot votar una vegada. El seu vot pot anar dirigit a un partit polític dels quals es presenten en el municipi (en la nostra aplicació es considera que no hi ha vots en blanc o vots nuls, tots els vots van dirigits a un partit polític).

Els partits polítics poden ser diferents depenent del municipi i en cada municipi presenten una llista de candidats diferent. Els candidats són ciutadans que resideixen en el mateix municipi on es presenten i només poden aparèixer com a candidats d'un partit.

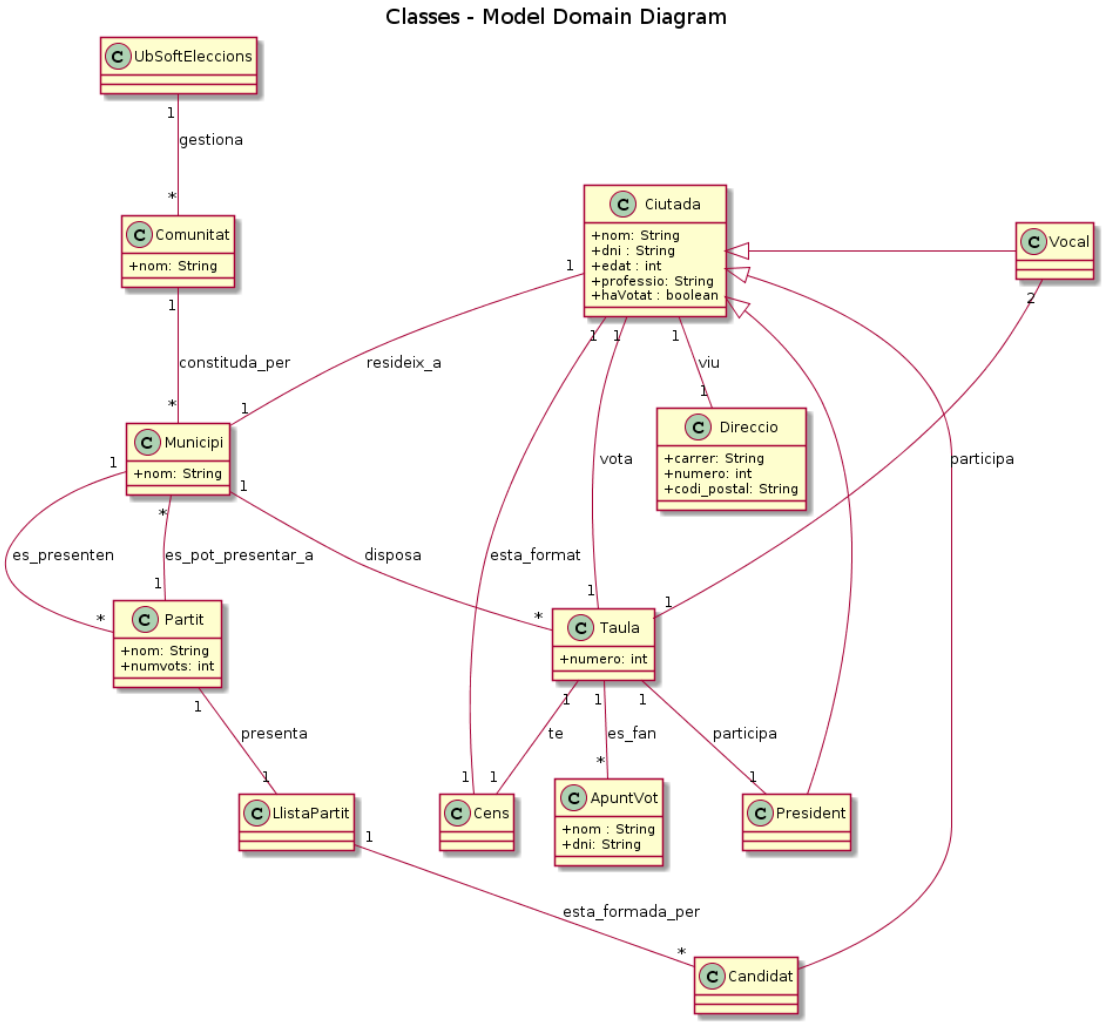
Quan un ciutadà va a exercir el seu dret al vot, el president introdueix el DNI del votant en el sistema i aquest verifica si pot votar o no. Alhora, el primer vocal s'encarrega de fer, externament al sistema, la mateixa comprovació en la còpia impresa del cens que aquesta en el seu poder.

Després de comprovar que el ciutadà esta en el cens, el sistema mostra una pantalla amb els partits polítics als quals pot votar. En aquest moment, el president deixa seleccionar al ciutadà un dels partits que hi ha en la pantalla. Quan el ciutadà ja ha votat, només falta realitzar l'anotació de vot. Una anotació de vot consisteix a apuntar el nom i el DNI del ciutadà que ha votat. El segon vocal fa una anotació de vot extern en una fulla de paper i el sistema automàticament fa una anotació de vot intern. El sistema no guarda la relació entre el vot i el ciutadà, ja que el vot és secret. Només augmenta el número de vots que correspongui.

Es sol·licita construir el model de domini utilitzant UML i completant-ho amb una descripció que faciliti la seva comprensió. Indiqueu els tipus dels atributs en el model de domini.

Solució

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Ciutada <|-- Candidat
Ciutada <|-- President
Ciutada <|- Vocal
UbSoftEleccions "1" -- "*" Comunitat: gestiona
Comunitat "1" -- "*" Municipi : constituda_per
Municipi "1" -- "*" Partit : es_presenten
Partit "1" -- "*" Municipi : es_pot_presentar_a
Ciutada "1" -- "1" Municipi : resideix_a
Municipi "1" -- "*" Taula : disposa
Partit "1" -- "1" LlistaPartit : presenta
LlistaPartit "1" -- "*" Candidat : esta_formada_per
Ciutada "1" -- "1" Taula : vota
Ciutada "1" -- "1" Cens : esta_format
Ciutada "1" -- "1" Direccio : viu
Taula "1" -- "1" Cens : te
Taula "1" -- "2" Vocal : participa
Taula "1" -- "1" President : participa
Taula "1" -- "*" ApuntVot : es_fan


class UbSoftEleccions
class Comunitat{
    + nom: String
}
class Partit{
    + nom: String
    + numvots: int
}
class Municipi{
    + nom: String
}
class LlistaPartit{

}
class Ciutada{
    + nom: String
    + dni : String
    + edat : int
    + professio: String
    + haVotat : boolean
}
class Taula{
    + numero: int
}
class ApuntVot{
    + nom : String
    + dni: String
}
class Candidat
class President
class Vocal
class Cens
class Direccio{
    + carrer: String
    + numero: int
    + codi_postal: String
}
```



7. Llista de tasques

Se'ns proposa el desenvolupament d'un gestor de tasques. L'objectiu del mateix és mantenir una llista de tasques per realitzar. De cada tasca s'emmagatzemarà una breu descripció que explica el que s'ha de fer. El sistema ha de permetre'ns observar les tasques pendents, afegir una nova tasca i eliminar una tasca ja existent. Dissenyar el sistema sabent que l'usuari utilitza una interfície de tipus consola.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

8. Llista de Tasques II

Se'ns demana ampliar el sistema de l'exercici anterior perquè inclogui a més les següents funcionalitats. Cada tasca tindrà associada una data d'inici i de final. A més, ara podrem marcar una tasca com realitzada en lloc d'haver d'esborrar-la (encara que es manté la possibilitat d'esborrar-les). El nou sistema també permetrà associar una prioritat a cada tasca, que podrà tenir els valors {baixa, mitjana, alta}. S'exigeix poder modificar cadascun de les característiques d'una tasca.

Quan se li sol·liciti al sistema la llista de tasques pendents, presentarà únicament les tasques per les quals la data actual és superior a la data d'inici i que no han estat marcades com realitzades. Les tasques apareixeran ordenades de la següent forma. En primer lloc apareixeran aquelles tasques per les quals la data de final ja hagi passat. Aquestes apareixeran marcades amb un *. A continuació apareixeran la resta de tasques, ordenades per prioritat.
També podrem sol·licitar una llista de tasques realitzades. En ella apareixerà, per ordre de data en la qual es van marcar com realitzades, totes les tasques realitzades.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

9. Llista de Tasques III

Se'ns demana ampliar el sistema de l'exercici anterior per convertir-ho en un gestor de tasques per a un cap de projecte. El cap de projecte té diversos treballadors al seu càrrec. El cap de projecte s'encarrega d'assignar cada tasca a un treballador. Quan la tasca s'acaba, el cap de projecte la marca com realitzada i inclou una qualificació {malament, regular, bé, excel·lent}indicant el nivell de satisfacció amb el treball. El cap encara no coneix bé als treballadors, amb el que és freqüent que el treballador rebutgi la tasca per no saber fer-la. En aquest cas, el cap torna a assignar-la a un altre treballador. Cada tasca s'anota amb una estimació de les hores de treball que ha de portar. En començar el dia, el cap necessita una fulla de tasques per a cada treballador, per tenir en compte la càrrega de cada treballador en assignar les tasques i per donar-li una còpia perquè sàpiga el que ha de fer. El format de la fulla és el mateix que es comenta en l'exercici anterior. A final de mes, el cap necessita tenir un informe per a cada treballador, en el qual apareguin les tasques que ha realitzat i el total que sumen les hores que al seu moment es van estimar per a cada tasca. Han aparèixer també en el resum de l'informe quantes tasques es van puntuar malament, regular, etc. Així el cap pot avaluar qui és el treballador més eficient

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

10. Llista de tasques polimòrfica

Se'ns proposa el desenvolupament d'un gestor de tasques. L'objectiu del mateix és mantenir una llista de tasques per realitzar. De cada tasca s'emmagatzemarà una breu descripció que explica el que s'ha de fer. El sistema ha de permetre'ns observar les tasques pendents, afegir una nova tasca i eliminar una tasca ja existent. El nostre usuari, que és un bon programador, vol poder distingir entre tres tipus de tasques: tasques d'anàlisis, tasques de disseny i tasques d'implementació.
Per a les tasques d'anàlisis el nostre usuari vol guardar el directori on es troba el model de domini a desenvolupar. Per a les tasques de disseny, es vol guardar el número de diagrames a desenvolupar associats amb aquesta tasca. Per a les tasques d'implementació, es vol emmagatzemar el número de línies de codi benvolgudes. Dissenyar el sistema sabent que l'usuari utilitza una interfície de tipus consola. Estudiar les diferències de disseny si s'assumeix que en el futur s'incorporaran nous tipus de tasca o no.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

11. El jardiner

El nostre informàtic s'ofereix per realitzar un sistema que ajudi en el dia a dia a un amic seu que és jardiner. El seu company li comenta que el seu principal problema és mantenir la informacióon de les diferents malalties que ataquen a les plantes i el seu tractament. Cada espècie de planta (ben sigui un arbre, un arbust o una planta d'interior) sofreix habitualment una sèrie de malalties típiques que es poden caracteritzar per una sèrie de símptomes (fulles seques, presència d'un cert insecte, etc.). Sol ser el cas que el mateix símptoma apareix en diverses malalties. Una mateixa malaltia pot afectar a diverses espècies de plantes, com en el cas de les malalties causades per fongs (rosegui, míldiu, etc.). El nostre amic volgués poder donar d'alta en el sistema les espècies de plantes amb les quals habitualment treballa, així com les seves malalties típiques i els tractaments de aquestes. A més desitjaria el fet que el sistema li ajudés a trobar les possibles malalties en una espècie de planta que presenta un cert símptoma.

- Analitzar el model de domini.
- Dissenyar els diagrames de seqüència de sistema.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

12. El jardiner II

El nostre amic ha quedat encantat amb el sistema de l'exercici anterior i ens ha sol·licitat una ampliació. Vol utilitzar el sistema per gestionar als seus clients i els tractaments de les seves plantes. El tractament d'una malaltia es compon de diverses visites, espaiades en el temps, en cadascuna de les quals s'efectua una acció curativa (fumigació, poda, abonament). El nostre amic haurà introduir el tractament per a cada malaltia.

Habitualment visita a un client al dia següent que aquest el crida. Una vegada realitzada aquesta visita, el nostre amic vol arribar a casa i donar d'alta cadascuna de les plantes malaltes del client i que el sistema li avisi de futures visites. El nostre amic vol que el sistema li proporcioni, a l'inici del dia, una llista amb les visites que ha de realitzar aquest dia (incloent nom, direcció i telèfon del client) i les accions curatives que ha de realitzar en cadascuna d'elles. Al analitzar el dia, el jardiner ha de marcar com realitzades les visites que hagi fet. Aquelles que no hagi realitzat s'acumularan per al dia següent.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

13. Gestió del historial Academic

Se'ns proposa informatitzar la gestió d'historials acadèmics de la Facultat de Matemàtiques. En la Facultat s'imparteixen, com ja sabeu, diverses titulacions, entre les quals inauguren ETIS i la llicenciatura de matemàtiques.

Cadascuna d'aquestes titulacions té un pla d'estudi en el qual apareixen les assignatures que s'han de cursar per obtenir la titulació. Per a cada assignatura, existiran, en general un o diversos grups de teoria, encara que l'habitual és que solament hi hagi un. Els grups de teoria se subdivideixen al seu torn en diversos grups per la realització de les classes pràctiques. Cada grup de teoria o de pràctiques és impartit per un únic professor. A cada grup de teoria se li assigna un nom de lletra majúscula i a cada grup de pràctiques un nom de lletra minúscula.

El sistema que es desitja tenir ha de ser gestionat pel personal de secretaria i el professorat. Els professors han de disposar d'una interfície web en la qual puguin puntuar als alumnes del seu grup. Aquesta interfície únicament s'utilitzarà para la introducció de les notes finals de l'assignatura en cada convocatòria i no per a cap nota parcial. Cada semestre que un alumne matricula una assignatura, té dret a ser avaluat en dues convocatòries. Les notes solo poden tenir els valors: No presentat, Suspès, suficient, Notable, Excel·lent, Matricula d'Honor. Per a cada alumne, el professor responsable de la introducció de les notes és el professor de teoria de l'assignatura. Cada any es jan un termini per a cada convocatòria. A partir de que finalitza el termini d'una convocatòria, les notes d'aquesta convocatòria ja no poden tornar a ser modificades pel professor. En Secretaria se encarreguen de la introducció dels terminis de cada convocatòria.

Els alumnes poden sol·licitar a la secretaria dos tipus d'historial acadèmic: un oficial i un d'informatiu. En tots dos historials s'indica clarament el nom, cognoms i direcció de l'alumne. En l'historial informatiu apareixen les assignatures que l'alumne ha aprovat en la titulació així com la convocatòria en què les aprovo i amb quina nota. L'historial oficial inclou també la informació relativa als crèdits que suposa cada assignatura. En l'historial oficial apareixen totes les qualificacions obtingudes en les diferents convocatòries de cada assignatura, independentment de si s'han aprovat o no. L'historial oficial calcula també una nota mitjana que és una mitjana ponderada en funció dels crèdits de cada assignatura. Finalment, en l'historial oficial apareix si l'alumne ha obtingut ja el títol en la titulació. En cas contrari, apareix el numero de crèdits que ha de cursar encara per obtenir-ho.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

14. Club de Futbol

Ens veiem immersos en un projecte per la informatització d'un club de futbol. Es tracta d'una aplicació que permeti al director tècnic la gestion dels fitxatges del club.

El director tècnic té la necessitat d'emmagatzemar les negociacions que té obertes amb altres clubs per la cessió o el traspàs de jugadors. Aquestes negociacions habitualment funcionen de la següent manera: el director tècnic obre la negociació amb una trucada de telefono i ho anota en el sistema informàtic. A continuació, qualsevol interacció que tingui a veure amb aquesta negociació (sigui la recepció d'un fax, una trucada, un menjar de negocis) s'ha de registrar en el sistema. Aquestes interaccions poden ser realitzades pel director tècnic o per qualsevol dels seus assistents. El director desitja poder tenir, abans de qualsevol reunió relativa a un fitxatges, una llista amb totes les interaccions amb aquest club (incloent si el representant del club en aquesta interacció va ser ell personalment i/o un o varis dels seus assistents) en aquesta negociació.

A més, la política del director tècnic és mantenir un estricte control i previsió de caixa, per la qual cosa desitja que cada interacció s'introdueixi i actualitzi, si existeix: el valor de tancament ofert per part del nostre club, el valor de tancament ofert per part de l'altre club i el valor de tancament benvolgut, que és una valoració personal de la persona que ha fet la interacció sobre qual serà el valor de tancament de l'operació. Això solament s'emmagatzemarà para operacions de compra-venda, no per a intercanvis. També voldrà un llistat d'altres operacions de traspàs o cessió que s'hagin tancat amb aquest club. També desitja tenir un llistat de negociacions en les quals hagi aparegut algun dels jugadors amb els quals s'està negociant.
La política del director és que totes les negociacions siguin a dues bandes, és a dir, involucrar-se únicament ells i el club que vol comprar o vendre al jugador o jugadors. D'altra banda, donada la recessió al mercat, es prefereixen els canvis de jugadors a les compra/vendes, encara que aquestes ultimes han de ser suportades per l'aplicació.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

15. La pizzeria

Es tracta de dissenyar el sistema de recepció de comandes d'una pizzeria. Concretament, l'aplicació que s'ha de dissenyar és la que haurà utilitzar el tele operador que s'encarrega de rebre les comandes telefònics. El procés habitual és el següent:
El tele operador rep cridades. Per a cadascuna d'aquestes trucades, el tele operador ha de sol·licitar el telefono i la direcció en la qual es realitzar el lliurament. A continuació ha de sol·licitar els productes que es desitgen. Una vegada s'ha realitzat això la introducció de la comanda ha acabat i la nostra aplicació ha d'enviar la comanda al sistema de servei de comandes a través d'una connexió de xarxa.

Els productes oferts són pizzes, sandvitxos, amanides, begudes, postres y complements. Existeixen 3 tipus d'amanides: Hawaiana, Tonyina i Vegetal, cadascuna amb diferents preus. Existeixen 6 modalitats de pizzes: Barbacoa, Especial de la casa, Carbonara, Ibèrica, Hawaiana i 4 formatges. Cada pizza es pot sol·licitar en diferents mides o varietats : petita, mitjana, familiar, calzzone i base doble. El preu de la pizza serà diferent per a cada modalitat i mida. Les modalitats i els mides de pizza així com els tipus d'amanida poden variar amb certa freqüència i el sistema que es dissenya ha de suportar-ho sense canvis en el codi. La resta de productes es poden caracteritzar per nom i preu. Així "33 cl. Coca-cola: 1 Euro". Per a cada producte normal (que no sigui pizza ni amanida), el tele operador disposarà d'una tecla que directament afegeixi una unitat del producte a la comanda. Per a les pizzes, el tele operador premerà la tecla de pizza, a continuació escollirà amb el ratolí la modalitat de pizza i després seleccionarà també amb el ratolí el tamany o varietat. Per a les amanides, premerà la tecla d'amanida i després seleccionarà el tipus d'amanida amb el ratolí. No es permet al tele operador eliminar o modificar els productes afegits a la comanda.

El sistema té un únic cas d'ús que és el de captura de comanda. Se sol·licita:

- Realitzar el cas d'ús textual per a capturar la comanda.
- Analitzar el model de domini.
- Dissenyar amb el màxim detall possible, utilitzant diagrames de seqüència, l'esdeveniment o el conjunt d'esdeveniments de sistema necessaris per afegir una pizza a la comanda.

16. El videoclub

La UB té un videoclub (VideoClubUB) per als seus alumnes i desitja automatitzar la gestió de les pel·lícules i dels préstecs que es fan en ell. El videoclub disposa d'una sèrie de pel·lícules en la seva videoteca. Per a cada pel·lícula existeix com a mínim un exemplar encara que pot haver-hi més d'un.

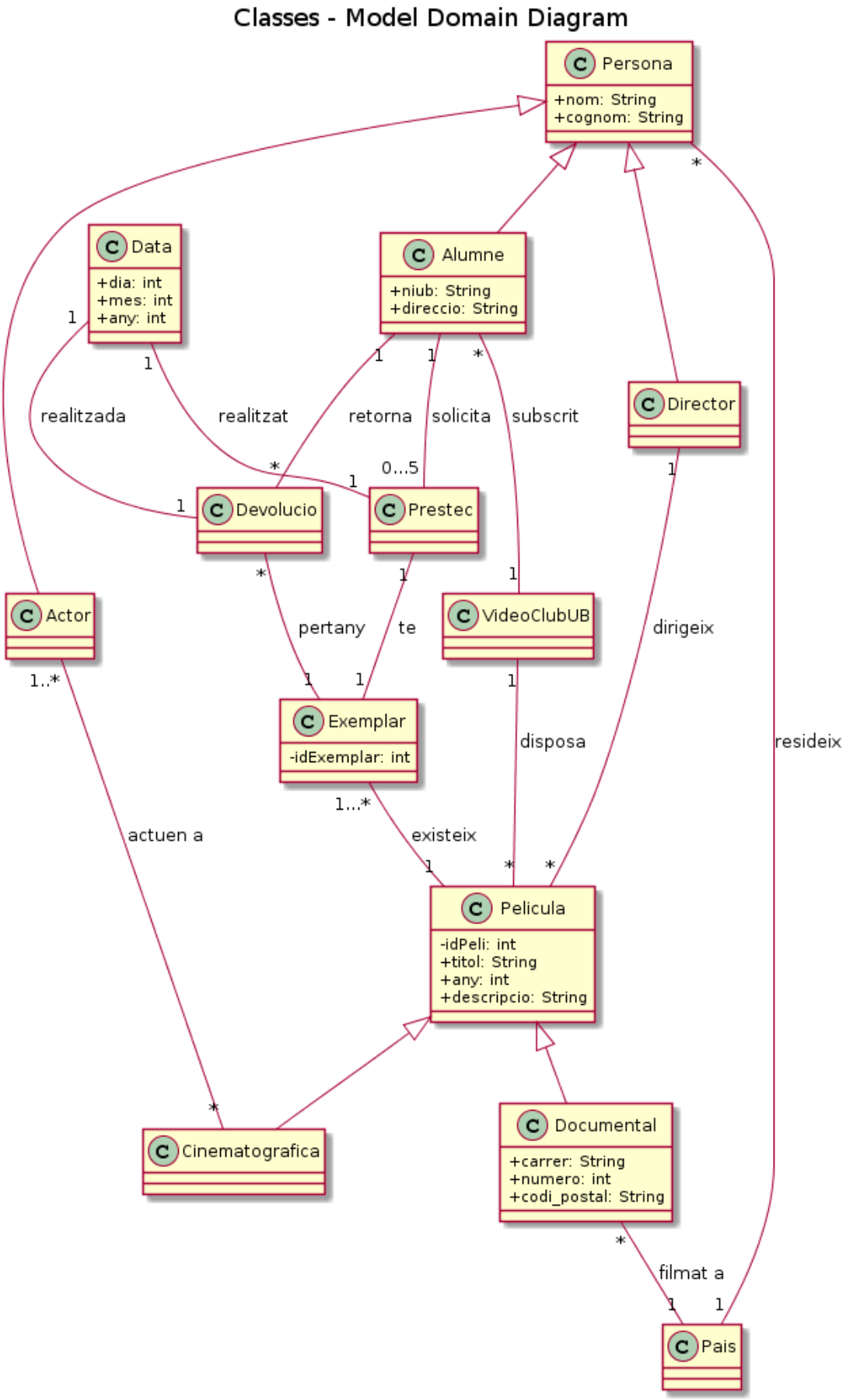
Cada pel·lícula té l'identificador de la pel·lícula que representa i cada exemplar té un identificador que ho identifica unívocament. Les pel·lícules poden ser documentals, o pel·lícules cinematogràfiques. En qualsevol cas, es guarda el títol, la durada, el director de la pel·lícula, l'any que es va realitzar i una breu descripció del contingut. Per als documentals es vol saber el país en el qual es va rodar. En el cas de pel·lícules cinematogràfiques, també emmagatzemem els actors que participen en ella. Tant dels actors com dels directors volem saber el seu nom i cognoms, i el seu país de naixement.

En el videoclub estan subscrits alumnes que sol·liciten en préstec pel·lícules. Aquests alumnes tenen el codi niub de la UB, un nom i cognoms, una adreça i un país de naixement. Per a cada alumne es coneix què préstecs té i quins préstecs ha retornat. Cada alumne pot tenir un màxim de 5 préstecs, en el qual s'indica la data en què es realitza el préstec i l'exemplar que es dóna en préstec. Cada devolució guarda la data en la qual es retorna el préstec i l'exemplar que s'ha retornat. Una data conté dia, mes i any. Una vegada retornat un préstec, aquest s'elimina i es guarda una nova devolució. L'aplicació a dissenyar haurà d'assumir que la introducció de les dades ja s'ha realitzat. L'objectiu de l'aplicació és permetre afegir nous préstecs, retornar préstecs i centrar-se en la gestió de diverses consultes.

- Analitzar el model de domini.
- Dissenyar un diagrama de seqüència de cadascun dels esdeveniments de sistema que apareguin en els diagrama de seqüència de sistema.
- Dissenyar el diagrama de classes de l'aplicació.

Solució

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Persona <|-- Director
Persona <|-- Actor
Persona <|-- Alumne
Pelicula <|-- Cinematografica
Pelicula <|-- Documental
Persona "*" -- "1" Pais: resideix
Alumne "*" -- "1" VideoClubUB : subscrit
Alumne "1" -- "0...5" Prestec : solicita
Alumne "1" -- "*" Devolucio : retorna
Data "1" -- "1" Prestec : realitzat
Data "1" -- "1" Devolucio : realitzada
VideoClubUB "1" -- "*" Pelicula : disposa
Prestec "1" -- "1" Exemplar : te
Devolucio "*" -- "1" Exemplar : pertany
Exemplar "1...*" -- "1" Pelicula : existeix
Actor "1..*" -- "*" Cinematografica : actuen a
Director "1" -- "*" Pelicula : dirigeix
Documental "*" -- "1" Pais : filmat a


class VideoClubUB
class Persona{
    + nom: String
    + cognom: String
}
class Pais{
}
class Director{
}
class Actor{
}
class Alumne{
    + niub: String
   + direccio: String
}
class Prestec{
}
class Exemplar{
    - idExemplar: int
}
class Pelicula{
    - idPeli: int
    + titol: String
    + any: int
    + descripcio: String
}
class Data{
    + dia: int
    + mes: int
    + any: int
}
class Devolucio
class Cinematografica
class Documental{
    + carrer: String
    + numero: int
    + codi_postal: String
}
```


Classes - Model Domain Diagram
Persona
+nom: String
+cognom: String
Data
+dia: int
+mes: int
+any: int
Alumne
+niub: String
+direccio: String
Director
Devolucio
Prestec
Actor
VideoClubUB
Exemplar
-idExemplar: int
Pelicula
-idPeli: int
+titol: String
+any: int
+descripcio: String
Cinematografica
Documental
+carrer: String
+numero: int
+codi_postal: String
Pais
realitzada
realitzat
retorna
solicita
subscrit
pertany
te
dirigeix
disposa
resideix
actuen a
existeix
filmat a
0...5
1..*
1...*

17. Companyia ferroviària

Se'ns sol·licita la creació d'un programa per la gestió d'una companyia de transport ferroviària. L'aplicació serà utilitzada per un operari per vendre bitllets de viatge, facturar mercaderies i realitzar la composició de vagons dels trens.

La companyia gestiona un únic tren diari que va de Barcelona a Bilbao i la seva tornada. No és previsible que la companyia passada a gestionar mas trens en el futur. Per a cada viatge, la companyia decideix que vagons compondran el tren en funció de la demanda.
La companyia disposa d'un magatzem de vagons en l'estació de Bilbao i un altre en la de Barcelona. La companyia usa aquests magatzems per deixar en ells els vagons si no necessita transportar-los en un determinat viatge.

Un tren es compon de vagons de passatgers, vagons de mercaderia i la locomotora. És possible que en el futur apareguin nous tipus de vagons i es vol minimitzar l'esforç necessari per estendre l'aplicació en aquest cas. Cada vagó de passatgers té una capacitat per 150 persones. Quan es realitza la venda d'un bitllet, s'assigna una posició (d'1 a 150) en un determinat vagó al client. Cada vagó de mercaderies pot contenir 697 m3. Quan es realitza la facturació d'una certa mercaderia , intervenen dos clients. El client principal (el que envia i paga) i el client receptor, que només rep. Pot donar-se el cas que el client principal i el client receptor siguin el mateix (per exemple quan algú viatja i decideix facturar el seu col·lecció de llibres al mateix tren). L'aplicació guardarà les dades habituals (nom, nif, direcció, telefono) dels clients. Per a la locomotora, cas de ser Dièsel, ha d'emmagatzemar-se el numero de litres de combustible, en cas contrari no.

- Identificar els casos d'ús i realitzar el diagrama de casos d'ús
- Construir el model de domini.
- Realitzar el disseny (diagrama de classes i diagrama de seqüència) de l'operació mostrarEstatDelTren() que mostra, per a cada tren, quin és l'estat de cadascun dels vagons que el composen. Per als vagons de passatgers, mostrar el número de seients lliures i el NIF i el nom de tots els clients que viatgen en aquest vagó. Per als vagons de mercaderies ha de mostrar la quantitat de m3 disponibles en el vagó, i per a cada paquet, la direcció d'origen i de destinació i el número de metres cúbics que ocupa. Per a la locomotora mostrar el número de litres de combustible. El disseny no ha d'assumir com precalculat el número de bitllets o el número de metres cúbics d'un vagó
- Passar a codi l'operació mostrarEstatDelTren del disseny anterior així com els mètodes més rellevants en els quals es recolza.
- Dissenyar (diagrama de classes i diagrama de seqüència) una operació mostrarTrensEnElsQueHeViatjatAmbLesMevesMercaderies(), en la qual donat un client, ens mostri els trens en els quals aquest client ha viatjat junt amb algun enviament de mercaderies del que ell mateix fos client principal.

18. Centre Excursionista

El Centre Excursionista de Folgeroles del Vallès ens sol·licita informatitzar la seva aplicació d'informació i cobrament de les excursions que organitzen.

El centre gestiona un conjunt d'excursions per als seus socis. Un soci té un nombre de soci que ho identifica inequívocament, a més, coneixem el seu nom, cognoms, any de naixement i la seva adreça completa, la qual inclou carrer, nombre, codi postal i població. De cada soci tenim les dades del seu compte bancari com són el nom de l'entitat bancària, el nombre de l'entitat bancària, el nombre de l'oficina i el nombre del seu compte corrent per poder fer el cobrament de les excursions.

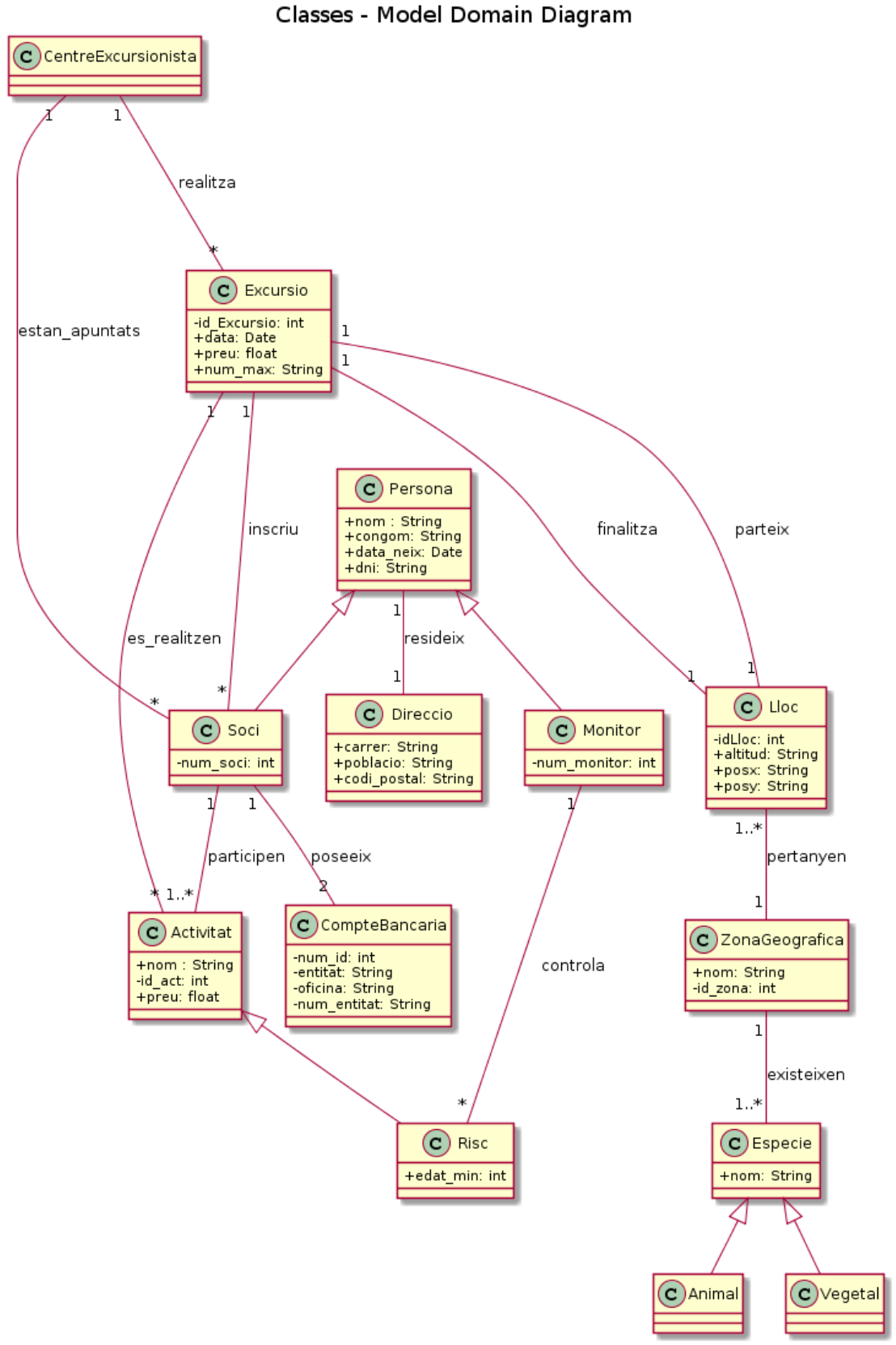
Cada excursió té el mes i l'any en la qual es va a realitzar, el preu basi de l'excursió (no inclou el cost de les activitats), el nombre màxim de socis que es poden apuntar a l'excursió, un lloc d'origen i un lloc de destinació. Per a cadascun dels llocs es desitja emmagatzemar la posició (posX) en quilòmetres, la posició (posY) en quilòmetres i l'altitud. Cada lloc pertany a una única zona geogràfica i a cada zona geogràfica existeixen una sèrie d'espècies (animals i vegetals) autòctones1 de la zona de les quals coneixem nom en llatí i el seu nom comú. Les espècies animals sabem si són animals perillosos i la seva alimentació (herbívor, carnívor, etc.) i per a les espècies vegetals sabem el tipus de fulla (caduca o perenne) que tenen. En cada excursió existeix la possibilitat de realitzar una sèrie d'activitats (visites a museus, bicicleta tot terreny, natació, etc.) de les quals sabem el seu nom i el seu preu. Especialment rellevants són les activitats de risc (barranquismo, descens en caiac, etc.). Totes les activitats de risc tenen una edat mínima per poder apuntar-se i a més hauran d'estar controlades per un monitor. Tots els monitors tenen un nombre de monitor que els dóna la federació de monitors d'activitats de risc.

L'aplicació a dissenyar ha d'assumir que la introducció de les dades de les excursions ja s'ha realitzat i centrar-se a gestionar l'accés a la informació disponible (diferents consultes que permetin trobar l'excursió ideal per a un cert soci), així com afegir i eliminar socis de cada excursió. Quant a les consultes, inicialment solament sera necessari una consulta per obtenir les excursions en les quals a la zona del seu lloc de destinació es pugui observar una certa espècie.

- Identificar els casos d'ús i realitzar el diagrama de casos d'ús
- Construir el model de domini.
- Realitzar el disseny (diagrama de classes i diagrama de seqüència) de l'operació que a partir d'un identificador d'espècie, ens mostra les excursions en les quals en el seu lloc de destinació es pot observar aquesta espècie.
- Passar a codi l'operació del disseny anterior així com els mètodes més rellevants en els quals es recolza.
- Realitzar el disseny (diagrama de seqüència i diagrama de classes si canvia pel que fa a l'anterior) que a partir d'un identificador de monitor mostri una excursió en la qual hi hagi una activitat que estigui controlada per ell.

Solució

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Activitat <|-- Risc
Persona <|-- Monitor
Persona <|-- Soci
Especie <|-- Animal
Especie <|-- Vegetal

CentreExcursionista "1" -- "*" Soci: estan_apuntats
CentreExcursionista "1"--- "*" Excursio : realitza
Excursio "1"---"*" Soci: inscriu
Excursio "1" --- "1" Lloc : parteix
Excursio "1" --- "1" Lloc : finalitza
Excursio "1" -- "*" Activitat : es_realitzen
Soci "1" -- "1..*" Activitat : participen
Soci "1" -- "2" CompteBancaria : poseeix
Persona "1" -- "1" Direccio : resideix
Lloc "1..*" -- "1" ZonaGeografica : pertanyen
ZonaGeografica "1" -- "1..*" Especie : existeixen
Monitor "1" --- "*" Risc : controla


class CentreExcursionista
class Persona{
    + nom : String
    + congom: String
    + data_neix: Date
    + dni: String
}
class Monitor{
    - num_monitor: int
}
class Direccio{
    + carrer: String
    + poblacio: String
    + codi_postal: String
}
class Soci{
    - num_soci: int
}
class Excursio{
    - id_Excursio: int
    + data: Date
    + preu: float
    + num_max: String
}
class Lloc{
    - idLloc: int
    + altitud: String
    + posx: String
    + posy: String
}
class ZonaGeografica{
    + nom: String
    - id_zona: int
}
class CompteBancaria{
    - num_id: int
    - entitat: String
    - oficina: String
    - num_entitat: String

}
class Activitat{
    + nom : String
    - id_act: int
    + preu: float
}
class Especie{
    + nom: String
}
class Risc{
    + edat_min: int
}
class Animal
class Vegetal



@enduml
```


Classes - Model Domain Diagram
CentreExcursionista
realitza
estan_apuntats
Excursio
-id_Excursio: int
+data: Date
+preu: float
+num_max: String
Persona
+nom : String
+congom: String
+data_neix: Date
+dni: String
inscriu
finalitza
parteix
es_realitzen
resideix
Soci
-num_soci: int
Direccio
+carrer: String
+poblacio: String
+codi_postal: String
Monitor
-num_monitor: int
Lloc
-idLloc: int
+altitud: String
+posx: String
+posy: String
participen
poseeix
pertanyen
Activitat
+nom : String
-id_act: int
+preu: float
CompteBancaria
-num_id: int
-entitat: String
-oficina: String
-num_entitat: String
controla
ZonaGeografica
+nom: String
-id_zona: int
existeixen
Risc
+edat_min: int
Especie
+nom: String
Animal
Vegetal

19. MusicaWeb

La universitat vol engegar un servei de música per Web que es dirà MusicaWebUB, per a això ens sol·licita una aplicació per a la gestió de la música que prefereixen els oients.

La música que es pot escoltar aquesta formada per diferents discos. Cada disc té un nom de grup o solista, un títol, un nombre de cançons, un any de publicació, la discogràfica que ho publica i un conjunt de cançons. Una discogràfica té un nom, un identificador i un any de creació. Les cançons per la seva banda tenen un nom, una durada (en minuts), un o diversos gèneres (com per exemple jazz, rock, pop, funky, etc.) i un o varis interprets, dels quals es vol distingir si són músics o cantants. Dels intèrprets es coneix el seu nom real, nom artístic i la seva data de naixement. En el cas dels músics es desitja conèixer què instrument és el que toquen (considerem que els músics només saben tocar un instrument). No es considera habitual que una persona sigui músic i cantant alhora, però pot ocórrer i el sistema ha de contemplar-ho d'alguna manera.

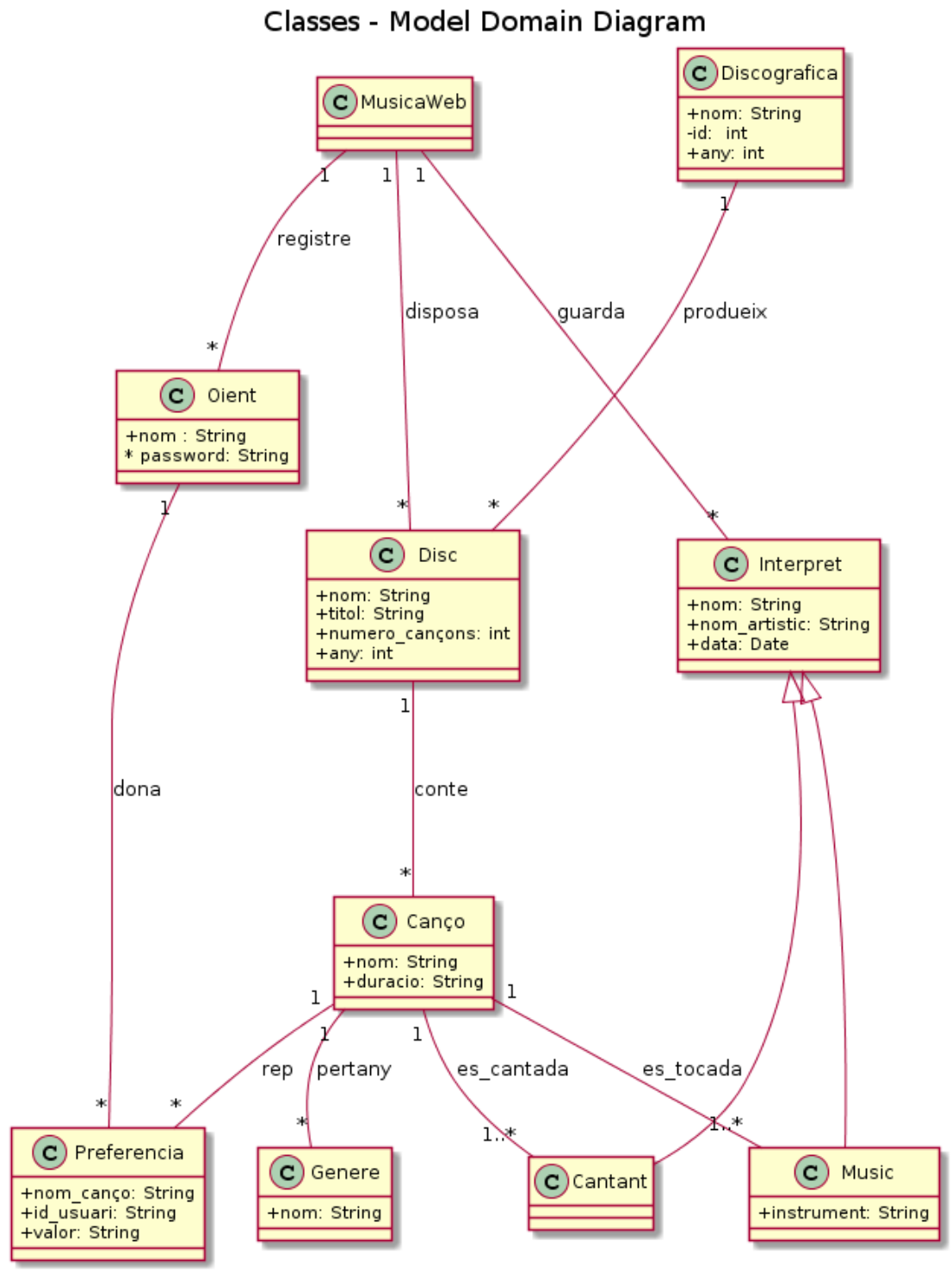
A més de la gestió de la música que es pot emetre, es vol saber quines són les preferències dels oïdors. Només els oïdors registrats prèviament poden indicar les seves preferències en el sistema, per al registre només han d'indicar un nom i un password. Una vegada registrat, un oient pot indicar les preferències de tantes cançons com desitgi, indicant el nom o identificador de la cançó, el seu identificador d'usuari i la seva valoració entre 1 i 5 (l'1 significa no m'agrada gens i el 5 significa m'agrada molt).

L'aplicació a dissenyar haurà d'assumir que la introducció de les dades ja s'ha realitzat. L'objectiu de l'aplicació és permetre afegir noves preferències per part dels usuaris i centrar-se en la gestió de diverses consultes, com per exemple:

- Mostrar el disc que ha rebut més preferències per part dels oïdors i el valor mitjana de les mateixes.
- Mostrar entre dos gèneres, com és el que més preferències ha introduït un oïdor donat.
- Mostrar totes les cançons que interpreta un cantant donat i al fet que disc pertanyen.
- Mostrar què cançó és la preferida pels oïdors.
- Mostrar si un oïdor ha valorat cançons d'un genero dau i quantes cançons d'aquest genero ha valorat.

Solució

```
@startuml
skinparam classAttributeIconSize 0
title Classes - Model Domain Diagram

Interpret <|-- Cantant
Interpret <|-- Music

MusicaWeb "1" -- "*" Oient: registre
MusicaWeb "1"--- "*" Disc : disposa
MusicaWeb "1"---"*" Interpret: guarda
Discografica "1" --- "*" Disc : produeix
Disc "1" --- "*" Canço : conte
Canço "1" -- "*" Genere : pertany
Canço "1" -- "1..*" Cantant : es_cantada
Canço "1" -- "1..*" Music : es_tocada
Canço "1" -- "*" Preferencia : rep
Oient "1" -- "*" Preferencia : dona


class MusicaWeb
class Oient{
    + nom : String
    * password: String
}
class Interpret{
    + nom: String
    + nom_artistic: String
    + data: Date
}
class Disc{
    + nom: String
    + titol: String
    + numero_cançons: int
    + any: int
}
class Discografica{
    + nom: String
    - id:  int
    + any: int
}
class Cantant
class Music{
    +instrument: String
}
class Canço{
    + nom: String
    + duracio: String
}
class Genere{
    + nom: String

}
class Preferencia{
    + nom_canço: String
    + id_usuari: String
    + valor: String
}
```


Classes - Model Domain Diagram
MusicaWeb
Discografica
+nom: String
-id: int
+any: int
registre
disposa
guarda
produeix
Oient
+nom : String
* password: String
Disc
+nom: String
+titol: String
+numero_cançons: int
+any: int
Interpret
+nom: String
+nom_artistic: String
+data: Date
dona
conte
Canço
+nom: String
+duracio: String
rep
pertany
es_cantada
es_tocada
Preferencia
+nom_canço: String
+id_usuari: String
+valor: String
Genere
+nom: String
Cantant
Music
+instrument: String