Listado de Problemas

Diseño de Software

Enginyeria Informàtica

Universitat de Barcelona

Curso 2013/2014

1. Problemas de modelo de dominio

1.1. Periódico

Dificultad estimada: Fácil

En un diario se desea automatizar el proceso de producción, escritura y publicación de noticias. Se ha decidido que la única persona que utilizará el programa que se desea construir será la secretaria del jefa de redacción. El proceso al que se desea dar soporte es el siguiente:

1. Diariamente, a primera hora de la mañana se reúne el consejo de redacción y decide qué noticias van a aparecer en la edición de ese día.

2. La Redactora Jefe comunica a la secretaria de redacción cuáles son las noticias que se van incluir en la edición de ese día y quién es el redactor que se va a encargar de redactar cada noticia. Como los titulares de cada noticia aún no se conocen (lo debe fijar el redactor de la noticia) a cada noticia se le asignará un titular interno que nos permitirá saber a que noticia nos referimos.

3. La Secretaria crea una nueva edición del diario (indicando la fecha actual), introduce en la edición del día las noticias e introduce en el sistema, para cada noticia, el redactor que se hará cargo. Asumimos que los redactores no cambiarán y se pueden identificar por su D.N.I. De cada redactor se deberá almacenar también su nombre y apellidos.

4. La Secretaria solicita al sistema la creación de las hojas de trabajo, una por cada redactor, en las que aparecerá la lista de noticias que le han sido asignadas. Para cada noticia, en el listado aparecerá el titular interno y el identificador de noticia, que será único para cada notícia. Además, aparecerán como campos para rellenar, el titular real de la noticia, el antetítulo y el texto de la noticia.

5. La Secretaria repartirá cada hoja de trabajo al redactor correspondiente.

6. La Secretaria recibirá de los redactores las hojas de trabajo con todas las noticias rellenadas.

7. La Secretaria introducirá en el sistema el titular, antetítulo y texto de cada noticia.

8. La Secretaria obtendrá un listado con toda la información de todas las noticias de la edición y se lo entregará a la Jefa de Redacción. En el listado, aparecerá, como campo a rellenar, el número de página al que se desea asignar la noticia.

9. El Jefe de Redacción indicará el número de páginas de la edición del diario (habitualmente 8) y rellenará sobre el listado los números de página que corresponden a cada noticia, así como la posición en la página caso considerarlo oportuno (arriba, abajo, derecha, izquierda). Una vez hecho esto retornará el listado relleno a la Secretaria.

10. La Secretaria introducirá en el sistema la información de paginación.

11. La Secretaria cerrará la edición y a partir de ese momento, la edición pasará a depender de imprenta.

Se solicita:

1. Realizar el modelo de dominio.
2. Modificar el modelo de dominio teniendo en cuenta que una noticia puede aparecer el varias páginas. En ese caso, consideraremos que la noticia se divide en varios fragmentos. Cada fragmento sólo puede aparecer en una página.

1.2. Alquiler de videojuegos

Dificultad estimada: Fàcil

La empresa de alquiler de videojuegos (EnjoyGames) tiene un local de atención al público donde están expuestas las carátulas de los videojuegos más demandados y las últimas novedades, aunque también existen listados en papel de todos los títulos que se podrían alquilar. Cuando un cliente solicita un título, se comprueban si hay ejemplares libres y si no hay problemas por ejemplares no devueltos se realiza el alquiler, quedando constancia de la fecha de alquiler y la fecha máxima de entrega; de forma que cuando el cliente devuelva el ejemplar se podrá comprobar si se le tiene que imponer una sanción. Cada cliente puede solicitar una relación de los videojuegos que ha alquilado previamente.

Se solicita construir el modelo de dominio utilizando UML y completándolo con una descripción que facilite su comprensión.

1.3. Profesores

Dificultad estimada: Fàcil

Un alumno asiste cursos. Los cursos están impartidos por un único profesor. El alumno no puede repetir el mismo curso, pero puede asistir a más de un curso. El profesor puede impartir diferentes cursos y repetir un mismo curso en varias ocasiones. Para que un curso se imparta debe haber un mínimo de 10 alumnos y un máximo de 50. Como registro del curso se guarda la fecha de comienzo, la fecha de finalización y la nota del alumno.

Se solicita construir el modelo de dominio utilizando UML y completándolo con una descripción que facilite su comprensión.

1.4. La liga de fútbol

Dificultad estimada: Media

Nos solicitan un programa para gestionar la información sobre los resultados de la liga de fútbol. De las diferentes categorías (de las que sabemos el nombre, como *primera división*, *segunda división*, etc.) nos guardaremos los resultados de todas las jornadas. Cada jornada se identifica por un número de jornada y en ella se disputan varios partidos, de los que se conoce la fecha del partido, la hora de inicio, el estadio, la capacidad del estadio, el número de goles del equipo visitante y número de goles del equipo local.

En cada partido participan 4 árbitros (donde cada uno tiene una misión, 2 hacen de linieres, 1 de árbitro y otro de cuarto árbitro), un equipo local y

un equipo visitante. Los árbitros y los equipos sólo pueden pitar o jugar en una categoría a la vez. Durante un partido suceden varias acciones como goles, faltas, corners, etc., de cada una de ellas se quiere conocer el minuto en que ocurrió y el jugador que la realizó. De las faltas se quiere saber el árbitro que la pitó. En un partido, el equipo ganador suma 3 puntos, el perdedor 0 puntos y un empate suma 1 punto para cada equipo. Un equipo esta formado por jugadores y puede tener varios entrenadores.

Durante la liga un jugador permanece en el mismo equipo. De un entrenador sabemos el nombre, la fecha en que fue contratado y los equipos donde anteriormente ha entrenado. Un jugador puede ser defensa, lateral, portero, etc., cada uno tiene un nombre, un dorsal asignado y el recuento de los goles marcados en la liga, en el caso de ser un portero además se conoce el total de goles recibidos y parados.

Se solicita construir el modelo de dominio utilizando UML y completándolo con una descripción que facilite su comprensión. Indicad los tipos de los atributos en el modelo de dominio.

1.5. Campeonato Fórmula 1

Dificultad estimada: Media

La Federación Internacional de Automovilismo (FIA) nos solicita una aplicación para la gestión del campeonato mundial automovilístico, conocido como la Fórmula 1 o F1. El campeonato de Fórmula 1 está formado por diferentes escuderías (también denominadas equipos) que participan en todas las carreras del campeonato.

Una escudería tiene un nombre, un país de origen, un número de puntos, un conjunto de mecánicos y dos pilotos: el primer piloto y el segundo piloto que corren en todas las carreras del campeonato1. Los mecánicos se dividen en ingenieros mecánicos jefe, mecánicos ordinarios y mecánicos especialistas. Los ingenieros mecánico jefe se encargan de la gestión de un monoplaza y el resto de mecánicos se encargan de la puesta a punto y reparación de los monoplazas. Tanto de los mecánicos como de los pilotos se conoce su nombre, año de nacimiento y país donde nació. Cada piloto tiene su propio ingeniero mecánico jefe y un contador de puntos. Una escudería tiene uno o varios patrocinadores que pagan una cantidad de dinero en concepto de publicidad en el monoplaza. Los patrocinadores pueden pagar publicidad en varias escuderías.

Cada carrera tiene un nombre, el país donde se celebra, la fecha, la pole position3 que indica qué piloto va a ser el primero en la parrilla de salida, el circuito y un estado de la carrera que puede ser pendiente, realizada, acabada. Con el estado pendiente se indica que el gran premio no se ha realizado, con realizada se indica que la prueba se ha realizado pero no se han actualizado los puntos a los ganadores y con el estado acabada se indica que la prueba ha finalizado completamente y los puntos se han actualizado tanto a las escuderías como a los pilotos. Los circuitos de las carreras pueden ser urbanos o permanentes. Todos los circuitos tienen un nombre y el tiempo de la vuelta más rápida de toda su historia. En cada carrera se guarda la información de los resultados de clasificación y de las incidencias acontecidas durante la celebración de la misma. La clasificación sólo guarda el piloto y la posición final en la que ha quedado. Las incidencias deben indicar el nombre de la incidencia, qué piloto la ha cometido y la vuelta en la que ha sucedido.

Se solicita construir el modelo de dominio utilizando UML y completándolo con una descripción que facilite su comprensión. Indicad los tipos de los atributos en el modelo de dominio.

1.6. Elecciones municipales

Dificultad estimada: Alta

La empresa *UBSoft* nos ha pedido hacer una aplicación piloto para gestionar los votos y los resultados de las elecciones municipales en el país. Aunque sean elecciones municipales, se quieren hacer estadísticas de votos a nivel autonómico. Por este motivo, la aplicación considera las diferentes comunidades autónomas y los municipios que pertenecen a cada una de ellas (se ha decidido omitir provincias y cualquier otro tipo de división territorial).

Cada municipio dispone de diferentes mesas electorales, cada una se distingue según el número que la identifica. Además, cada mesa electoral tiene su censo electoral. Este censo electoral esta formado por ciudadanos de los cuales sabemos su nombre, DNI, edad, profesión y la dirección donde residen en el municipio. Un ciudadano sólo puede ir a votar a la mesa electoral que tiene asignada. En la mesa electoral participan tres ciudadanos, dos vocales y un presidente, ninguno de ellos puede ser candidato en una lista electoral.

Un ciudadano solo puede votar si consta en el censo electoral y sólo puede votar una vez. Su voto puede ir dirigido a un partido político de los que se presentan en el municipio (en nuestra aplicación se considera que no hay votos en blanco o votos nulos, todos los votos van dirigidos a un partido político). Los partidos políticos pueden ser diferentes dependiendo del municipio y en cada municipio presentan una lista de candidatos diferente. Los candidatos son ciudadanos que residen en el mismo municipio donde se presentan y sólo pueden aparecer como candidatos de un partido.

Cuando un ciudadano va a ejercer su derecho al voto, el presidente introduce el DNI del votante en el sistema y éste verifica si puede votar o no. A la vez, el primer vocal se encarga de hacer, externamente al sistema, la misma comprobación en la copia impresa del censo que esta en su poder. Después de comprobar que el ciudadano está en el censo, el sistema muestra una pantalla con los partidos políticos a los que puede votar. En ese momento, el presidente deja seleccionar al ciudadano uno de los partidos que hay en la pantalla. Cuando el ciudadano ya ha votado, sólo falta realizar el apunte de voto. Un apunte de voto consiste en apuntar el nombre y el DNI del ciudadano que ha votado. El segundo vocal hace un apunte de voto externo en una hoja de papel y el sistema automáticamente hace un apunte de voto interno. El sistema no guarda la relación entre el voto y el ciudadano, ya que el voto es secreto. Sólo aumenta el número de votos que corresponda.

Se solicita construir el modelo de dominio utilizando UML y completándolo con una descripción que facilite su comprensión. Indicad los tipos de los atributos en el modelo de dominio.

2. Problemas completos

2.1. Lista de tareas

Dificultad estimada: Fácil
Se nos propone el desarrollo de un gestor de tareas. El objetivo del mismo es mantener una lista de tareas por realizar. De cada tarea se almacenará una breve descripción que explica lo que se debe hacer. El sistema debe permitirnos observar las tareas pendientes, añadir una nueva tarea y eliminar una tarea ya existente. Diseñar el sistema sabiendo que el usuario utiliza una interficie de tipo consola

1. Analizar el modelo de dominio.
2. Diseñar los diagramas de secuencia de sistema.
3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.
4. Diseñar el diagrama de clases de la aplicación.

2.2. Lista de tareas (II)

Dificultad estimada: Fácil
Se nos pide ampliar el sistema del ejercicio anterior para que incluya además las siguientes funcionalidades. Cada tarea tendrá asociada una fecha de inicio y de final. Además, ahora podremos marcar una tarea como realizada en lugar de tener que borrarla (aunque se mantiene la posibilidad de borrarlas). El nuevo sistema también permitirá asociar una prioridad a cada tarea, que podrá tener los valores {baja,media,alta}. Se exige poder modificar cada uno de las carácterísiticas de una tarea.

Cuando se le solicite al sistema la lista de tareas pendientes, presentará únicamente las tareas para las cuales la fecha actual es superior a la fecha de inicio y que no han sido marcadas como realizadas. Las tareas aparecerán ordenadas de la siguiente forma. En primer lugar aparecerán aquellas tareas para las cuales la fecha de final ya haya pasado. Estas aparecerán marcadas con un *. A continuación aparecerán el resto de tareas, ordenadas por prioridad.

También podremos solicitar una lista de tareas realizadas. En ella aparecerá, por orden de fecha en la que se marcaron como realizadas, todas las tareas realizadas.

1. Ampliar el modelo de dominio.
2. Diseñar los nuevos diagramas de secuencia de sistema y modificar los antiguos si fuera necesario.
3. Diseñar un diagrama de secuencia de cada uno de los nuevos eventos de sistema que aparezcan en el diagrama de secuencia de sistema y modificar los antiguos si fuera necesario.
4. Ampliar el diagrama de clases de la aplicación

2.3. Lista de tareas (III)

Dificultad estimada: Media

Se nos pide ampliar el sistema del ejercicio anterior para convertirlo en un gestor de tareas para un jefe de proyecto. El jefe de proyecto tiene varios trabajadores a su cargo. El jefe de proyecto se encarga de asignar cada tarea a un trabajador. Cuando la tarea se termina, el jefe de proyecto la marca como realizada e incluye una calificación {mal,regular,bien,excelente} indicando el nivel de satisfacción con el trabajo.

El jefe aún no conoce bien a los trabajadores, con lo que es frecuente que el trabajador rechace la tarea por no saber hacerla. En ese caso, el jefe vuelve a asignarla a otro trabajador. Cada tarea se anota con una estimación de las horas de trabajo que debe llevar. Al empezar el día, el jefe necesita una hoja de tareas para cada trabajador, para tener en cuenta la carga de cada trabajador al asignar las tareas y para darle una copia para que sepa lo que tiene que hacer. El formato de la hoja es el mismo que se comentó en el ejercicio anterior. A final de mes, el jefe necesita tener un informe para cada trabajador, en el que aparezcan las tareas que ha realizado y el total que suman las horas que en su momento se estimaron para cada tarea. Deberán aparecer también en el resumen del informe cuantas tareas se puntuaron mal, regular, etc. Así el jefe puede evaluar quién es el trabajador más eficiente.

1. Ampliar el modelo de dominio.
2. Diseñar los nuevos diagramas de secuencia de sistema y modificar los antiguos si fuera necesario.
3. Diseñar un diagrama de secuencia de cada uno de los nuevos eventos de sistema que aparezcan en el diagrama de secuencia de sistema y modificar los antiguos si fuera necesario.
4. Ampliar el diagrama de clases de la aplicación

2.4. Lista de tareas polimórfica

Dificultad estimada: Intermedio

Se nos propone el desarrollo de un gestor de tareas. El objetivo del mismo es mantener una lista de tareas por realizar. De cada tarea se almacenará una breve descripción que explica lo que se debe hacer. El sistema debe permitirnos observar las tareas pendientes, añadir una nueva tarea y eliminar una tarea ya existente. Nuestro usuario, que es un buen programador, quiere poder distinguir entre tres tipos de tareas: tareas de análisis, tareas de diseño y tareas de implementación. Para las tareas de análisis nuestro usuario quiere guardar el directorio donde se encuentra el modelo de dominio a desarrollar. Para las tareas de diseño, se quiere guardar el número de diagramas a desarrollar asociados conesa tarea. Para las tareas de implementación, se quiere almacenar el número de lineas de código estimadas. Diseñar el sistema sabiendo que el usuario utiliza una interficie de tipo consola. Estudiar las diferencias de diseño si se asume que en el futuro se incorporarán nuevos tipos de tarea o no.

1. Analizar el modelo de dominio.
2. Diseñar los diagramas de secuencia de sistema.

3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.

4. Diseñar el diagrama de clases de la aplicación.

2.5. El jardinero

Dificultad estimada: Media

Nuestro informático se ofrece para realizar un sistema que ayude en el día a día a un amigo suyo que es jardinero. Su compañero le comenta que su principal problema es mantener la información de las diferentes enfermedades que atacan a las plantas y su tratamiento. Cada especie de planta (bien sea un árbol, un arbusto o una planta de interior) sufre habitualmente una serie de enfermedades típicas que se pueden caracterizar por una serie de síntomas (hojas secas, presencia de un cierto ácaro, etc.). Suele ser el caso que el mismo síntoma aparece en varias enfermedades. Una misma enfermedad puede afectar a diversas especies de plantas, como en el caso de las enfermedades causadas por hongos (roya, mildiu, etc.). Nuestro amigo quisiera poder dar de alta en el sistema las especies de plantas con las que habitualmente trabaja, así como sus enfermedades típicas y los tratamientos de éstas. Además desearía que el sistema le ayudase a encontrar las posibles enfermedades en una especie de planta que presenta un cierto síntoma.

1. Analizar el modelo de dominio.

2. Diseñar los diagramas de secuencia de sistema.

3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.

4. Diseñar el diagrama de clases de la aplicación.

2.6. El jardinero II

Dificultad estimada: Difícil

Nuestro amigo ha quedado encantado con el sistema del ejercicio anterior y nos ha solicitado una ampliación. Quiere utilizar el sistema para gestionar a sus clientes y los tratamientos de sus plantas. El tratamiento de una enfermedad se compone de varias visitas, espaciadas en el tiempo, en cada una de las cuales se efectúa una acción curativa (fumigación, poda, abono). Nuestro amigo deberá introducir el tratamiento para cada enfermedad.

Habitualmente visita a un cliente al día siguiente de que éste le llama. Una vez realizada esa visita, nuestro amigo quiere llegar a casa y dar de alta cada una de las plantas enfermas del cliente y que el sistema le avise de futuras visitas. Nuestro amigo quiere que el sistema le proporcione, al inicio del día, una lista con las visitas que debe realizar ese día (incluyendo nombre, dirección y teléfono del cliente) y las acciones curativas que debe realizar en cada una de ellas. Al finalizar el día, el jardinero debe marcar como realizadas las visitas que haya hecho. Aquellas que no haya realizado se acumularán para el día siguiente.

1. Ampliar el modelo de dominio.

2. Diseñar los nuevos diagramas de secuencia de sistema y modificar los antiguos si fuera necesario.

3. Diseñar un diagrama de secuencia de cada uno de los nuevos eventos de sistema que aparezcan en el diagrama de secuencia de sistema y modificar los antiguos si fuera necesario.

4. Ampliar el diagrama de clases de la aplicación

2.7. Gestión del historial académico

Dificultad estimada: Difícil

Se nos propone informatizar la gestión de historiales académicos de la Facultat de Matemàtiques. En la Facultat se imparten, como ya sabeis, varias titulaciones, entre las que figuran ETIS y la licenciatura de matemáticas. Cada una de estas titulaciones tiene un plan de estudio en el que aparecen las asignaturas que se deben cursar para obtener la titulación. Para cada asignatura, existirán, en general uno o varios grupos de teoría, aunque lo habitual es que solo haya uno. Los grupos de teoría se subdividen a su vez en varios grupos para la realización de las clases prácticas. Cada grupo de teoría o de prácticas es impartido por un único profesor. A cada grupo de teoría se le asigna un nombre de letra mayúscula y a cada grupo de prácticas un nombre de letra minúscula.

El sistema que se desea tener debe ser gestionado por el personal de secretaría y el profesorado. Los profesores deben disponer de una interficie web en la cual puedan puntuar a los alumnos de su grupo. Esta interficie únicamente se utilizará para la introducción de las notas finales de la asignatura en cada convocatoria y no para ninguna nota parcial. Cada semestre que un alumno matricula una asignatura, tiene derecho a ser evaluado en dos convocatorias. Las notas sólo pueden tener los valores: No presentado, Suspendido, Suficiente, Notable, Sobresaliente, Matrícula de Honor. Para cada alumno, el profesor responsable de la introducción de las notas es el profesor de teoría de la asignatura.

Cada año se fijan un plazo para cada convocatoria. A partir de que finaliza el plazo de una convocatoria, las notas de esa convocatoria ya no pueden volver a ser modificadas por el profesor. En Secretaría se encargan de la introducción de los plazos de cada convocatoria.

Los alumnos pueden solicitar a la secretaría dos tipos de historial académico: uno oficial y uno informativo. En ambos historiales se indica claramente el nombre, apellidos y dirección del alumno. En el historial informativo aparecen las asignaturas que el alumno ha aprobado en la titulación así como la convocatoria en que las aprobó y con qué nota. El historial oficial incluye también la información relativa a los créditos que supone cada asignatura. En el historial oficial aparecen todas las calificaciones obtenidas en las diferentes convocatorias de cada asignatura, independientemente de si se han aprobado o no. El historial oficial calcula también una nota media que es una media ponderada en función de los créditos de cada asignatura. Finalmente, en el historial oficial aparece si el alumno ha obtenido ya el título en la titulación. En caso contrario, aparece el número de créditos que debe cursar todavía para obtenerlo.

1. Analizar el modelo de dominio.

2. Diseñar los diagramas de secuencia de sistema.

3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.

4. Diseñar el diagrama de clases de la aplicación.

2.8. El club de fútbol

Dificultad estimada: Difícil

Nos vemos inmersos en un proyecto para la informatización de un club de futbol. Se trata de una aplicación que permita al director técnico la gestión de los fichajes del club.

El director técnico tiene la necesidad de almacenar las negociaciones que tiene abiertas con otros clubs para la cesión o el traspaso de jugadores. Esas negociaciones habitualmente funcionan de la siguiente manera: el director técnico abre la negociación con una llamada de teléfono y lo anota en el sistema informático. A continuación, cualquier interacción que tenga que ver con esa negociación (sea la recepción de un fax, una llamada, una comida de negocios) se debe registrar en el sistema. Estas interacciones pueden ser realizadas por el director técnico o por cualquiera de sus asistentes. El director desea poder tener, antes de cualquier reunión relativa a un fichaje, una lista con todas las interacciones con ese club (incluyendo si el representante del club en esa interacción fue él personalmente y/o uno o varios de sus asistentes) en esa negociación. Además, la política del director técnico es mantener un estricto control y previsión de caja, por lo que desea que cada interacción se introduzca y actualice, si existe: el valor de cierre ofertado por parte de nuestro club, el valor de cierre ofertado por parte del otro club y el valor de cierre estimado, que es una valoración personal de la persona que ha hecho la interacción sobre cual será el valor de cierre de la operación. Esto solo se almacenará para operaciones de compra-venta, no para intercambios. También querrá un listado de otras operaciones de traspaso o cesión que se hayan cerrado con ese club. También desea tener un listado de negociaciones en las que haya aparecido alguno de los jugadores con los que se está negociando. La política del director es que todas las negociaciones sean a dos bandas, es decir, involucrarse únicamente ellos y el club que quiere comprar o vender al jugador o jugadores. Por otra parte, dada la recesión en el mercado, se prefieren los cambios de jugadores a las compra/ventas, aunque éstas últimas deben ser soportadas por la aplicación.

1. Analizar el modelo de dominio.

2. Diseñar los diagramas de secuencia de sistema.

3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.

4. Diseñar el diagrama de clases de la aplicación.

2.9. La pizzería

Dificultad estimada: Difícil

Se trata de diseñar el sistema de recepción de pedidos de una pizzería. Concretamente, la aplicación que se debe diseñar es la que deberá utilizar el teleoperador

que se encarga de recibir los pedidos telefónicos. El proceso habitual es el siguiente: El teleoperador recibe llamadas. Para cada una de esas llamadas, el teleoperador debe solicitar el teléfono y la dirección en la que se realizará la entrega. A continuación debe solicitar los productos que se desean. Una vez se ha realizado esto la introducción del pedido ha terminado y nuestra aplicación debe enviar el pedido al sistema de servicio de pedidos a través de una conexión de red.

Los productos ofrecidos son pizzas, sandwitches, ensaladas, bebidas, postres y complementos. Existen 3 tipos de ensaladas: Hawaiana, Atún y Vegetal, cada una con diferentes precios. Existen 6 modalidades de pizzas: Barbacoa, Especial de la casa, Carbonara, Iberica, Hawaiana y 4 quesos. Cada pizza se puede solicitar en diferentes tamaños o variedades : pequeña, mediana, familiar, calzzone y base doble. El precio de la pizza será diferente para cada modalidad y tamaño. Las modalidades y los tamaños de pizza así como los tipos de ensalada pueden variar con cierta frecuencia y el sistema que se diseñe deberá soportarlo sin cambios en el código. El resto de productos se pueden caracterizar por nombre y precio. Así "Lata 33 cl. Coca-Cola: 1 Euro". Para cada producto normal (que no sea pizza ni ensalada), el teleoperador dispondrá de una tecla que directamente añade una unidad del producto al pedido. Para las pizzas, el teleoperador pulsará la tecla de pizza, a continuación eligirá con el ratón la modalidad de pizza y después seleccionará también con el ratón el tamaño o variedad. Para las ensaladas, pulsará la tecla de ensalada y después seleccionará el tipo de ensalada con el ratón. No se permite al teleoperador eliminar o modificar los productos añadidos al pedido.

El sistema tiene un único caso de uso que es el de captura de pedido. Se solicita:

1. Dibujar el diagrama de secuencia de sistema del caso de uso captura de pedido. Es importante que en todos los eventos de sistema aparezcan claramente los parámetros que recibe y devuelve el sistema y sus tipos.

2. Analizar el modelo de dominio.

3. Diseñar con el máximo detalle posible, utilizando diagramas de secuencia, el evento o el conjunto de eventos de sistema necesarios para añadir una pizza al pedido.

2.10. El videoclub

Dificultad estimada:
Se necesita construir un sistema software para un video club. En este videoclub:

- Los clientes tiene cuentas a las que pueden cargar los productos.
- Los producstos son o videos o DVDs.
- Podemos tener una o varias copiasde cada título en el catálogo.
- Los títulos se agrupan por temas
- Debemos mantener la información sobre las preferencias de los clientes
- Los clientes tienen que poder reservar títulos

- Los clientes tienen que poder alquilar y retornar títulos.
- Se debe mantener un histórico de alquileres realizados por los clientes.
- El gestor de la tienda es el único que puede añadir o eliminar títulos.
- A cada cliente se le dan dos tarjetas para alquilar productos.
- Cuando se devuelve un producto con retraso se multa al cliente.
- Los títulos perdidos se cargan en la cuenta del cliente que los alquiló.

1. Analizar el modelo de dominio.
2. Diseñar los diagramas de secuencia de sistema.
3. Diseñar un diagrama de secuencia de cada uno de los eventos de sistema que aparezcan en los diagrama de secuencia de sistema.
4. Diseñar el diagrama de clases de la aplicación.

2.11. Compañía ferroviaria

Dificultad estimada:
Se nos solicita la creación de un programa para la gestión de una compañía de transporte ferroviaria. La aplicación será usada por un operario para vender billetes de viaje, facturar mercancías y realizar la composición de vagones de los trenes. La compañía gestiona un único tren diario que va de Barcelona a Bilbao y su retorno. No es previsible que la compañía pase a gestionar más trenes en el futuro. Para cada viaje, la compañía decide que vagones compondrán el tren en función de la demanda. La compañía dispone de un almacén de vagones en la estación de Bilbao y otro en la de Barcelona. La compañía usa esos almacenes para dejar en ellos los vagones si no necesita transportarlos en un determinado viaje. Un tren se compone de vagones de pasajeros, vagones de mercancía y la locomotora. Es posible que en el futuro aparezcan nuevos tipos de vagones y se quiere minimizar el esfuerzo necesario para extender la aplicación en ese caso. Cada vagón de pasajeros tiene una capacidad para 150 personas. Cuando se realiza la venta de un billete, se asigna una posición (de 1 a 150) en un determinado vagón al cliente. Cada vagón de mercancias puede contener 697 m^3. Cuando se realiza la facturación de una cierta mercancía, intervienen dos clientes. El cliente principal (el que envia y paga) y el cliente receptor, que sólo recibe. Puede darse el caso de que el cliente principal y el cliente receptor sean el mismo (por ejemplo cuando alguien viaja y decide facturar su colección de libros en el mismo tren). La aplicación guardará los datos habituales (nombre, nif, dirección, telefono) de los clientes. Para la locomotora, caso de ser Diesel, debe almacenarse el número de litros de combustible, en caso contrario no.

1. Identificar los casos de uso y realizar el diagrama de casos de uso
2. Construir el modelo de dominio.
3. Realizar el diseño (diagrama de clases + diagrama de secuencia) de la operación mostrarEstadoTren() que muestra, para cada tren, cuál es el estado de cada uno de los vagones que lo componen. Para los vagones de

pasajeros, mostrará el número de asientos libres y el NIF y el nombre de todos los clientes que viajan en ese vagón. Para los vagones de mercancias deberá mostrar la cantidad de m^3 disponibles en el vagón, y para cada paquete, la dirección de origen y de destino y el número de metros cúbicos que ocupa. Para la locomotora mostrará el número de litros de combustible. El diseño no debe asumir como precalculado el número de billetes o el número de metros cúbicos de un vagón

4. Pasar a código la operación mostrarEstadoTren de el diseño anterior así como los métodos más relevantes en los que se apoya.

5. Diseñar (diagrama de clases + diagrama de secuencia) una operación mostrarTrenesEnLosQueHeViajadoConMisMercancias(), en la que dado un cliente, nos muestre los trenes en los que ese cliente ha viajado junto con algún envio de mercancias del que él mismo fuera cliente principal.

2.12. Centro excursionista

Dificultad estimada:
El *Centre Excursionista de Folgeroles del Vallès* nos solicita informatizar su aplicación de información de excursiones organizadas.

El centro gestiona un conjunto de excursiones para sus socios. Cada excursión tiene un lugar de origen y un lugar de destino. Para cada uno de ellos se desea almacenar la posición x en kilómetros, la posición y en kilómetros y la altitud. Cada lugar pertenece a una cierta zona geográfica y en cada zona geográfica existen una serie de especies (animales y vegetales) autóctonas de la zona.

En cada excursión existe la posibilidad de realizar una serie de actividades (visitas a museos, bicicleta de montaña, natación, etc). Especialmente relevantes son las actividades de riesgo (barranquismo, descenso en kayak, etc). Todas las actividades de riesgo deberán estar controladas por un monitor.

La aplicación a diseñar deberá asumir que la introducción de los datos de las excursiones ya se ha realizado y centrarse en gestionar el acceso a la información disponible (diferentes consultas que permitan encontrar la excursión ideal para un cierto socio), así como añadir y eliminar socios de cada excursión. En cuanto a las consultas, inicialmente solo será necesario una consulta para obtener las excursiones en las que en la zona de su lugar de destino se pueda observar una cierta especie.

1. Identificar los casos de uso y realizar el diagrama de casos de uso (diagrama de contexto)

2. Construir el modelo de dominio.

3. Realizar el diseño (diagrama de clases + diagrama de secuencia) de la operación que a partir de un identificador de especie, nos muestra las excursiones en las que en su lugar de destino se puede observar esa especie.

4. Pasar a código la operación del diseño anterior así como los métodos más relevantes en los que se apoya.

5. Realizar el diseño (diagrama de secuencia + diagrama de clases si cambia con respecto al anterior) que a partir de un identificador de monitor

muestre una excursión en la que haya una actividad que esté controlada por él.