Programación 4

EXAMEN JULIO 2024

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su número de examen
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el examen

Problema 1 (30 puntos)

Se le ha contratado para incorporarse en el diseño de una aplicación de gestión de tareas para una organización de desarrollo de software. La aplicación cuenta con usuarios, los cuales se registran mediante un nickname (que los identifica) y un puesto (Desarrollador, Tester o PM). Todo usuario puede ser asignado a un proyecto, pero solo los usuarios que además sean administradores pueden crear los proyectos. Los proyectos tienen un nombre, una fecha de inicio y una descripción. Cada proyecto está compuesto por un conjunto de tareas específicas del proyecto, las cuales define el administrador al momento de crearlo. Cada tarea cuenta con un estado que describe el avance realizado en la tarea como un porcentaje de los tickets completados, además de la fecha de inicio del trabajo sobre la misma y un nombre. Las tareas son asignadas a los usuarios, quienes deben completar todos los tickets que componen la tarea. Cada ticket posee una descripción. La aplicación debe brindar la posibilidad a los usuarios de realizar notas sobre las tareas a las cuales están asignados. Cada nota tendrá un campo de texto y una fecha de creación.

Además, se cuenta con la descripción del siguiente Caso de Uso:

Caso de Uso	Alta de proyecto
Actor	Administrador
Descripción	El caso de uso comienza cuando se desea crear un nuevo proyecto en el sistema. Para ello, el Administrador ingresa el nombre del proyecto, la fecha de inicio y una descripción del proyecto. Luego, el Sistema lista el nickname de todos los usuarios registrados, y el Administrador selecciona los usuarios que serán asignados al proyecto. Posteriormente, el Administrador define un conjunto de tareas iniciales (sin tickets) para el proyecto, ingresando su nombre. Cada tarea puede ser asignada a uno o varios usuarios seleccionados previamente. Una vez ingresados todos los datos, el Sistema muestra un resumen del proyecto con todas las tareas y usuarios asignados. El Administrador tiene la opción de confirmar la creación del proyecto o cancelar la operación. Si el Administrador confirma, el Sistema crea el nuevo proyecto con los datos ingresados. Si cancela, el Sistema descarta la información y no se crea el proyecto.

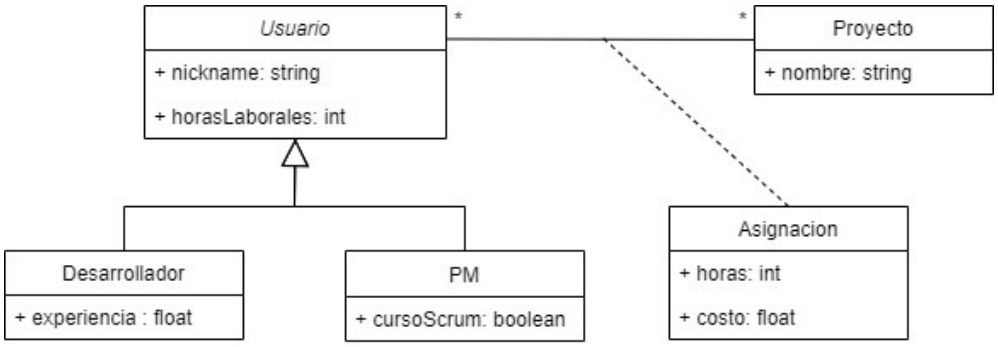
Se pide:

a) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.

b) Realizar un Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso. Indique el uso de memoria del Sistema y de datatypes, si corresponde.

Problema 2 (35 puntos)

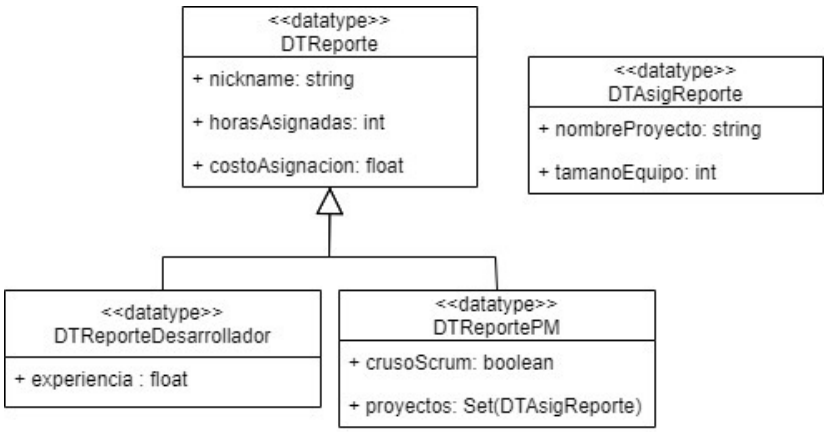
La figura muestra el Modelo de Dominio realizado para un sistema de gestión de proyectos de software. Los usuarios se identifican por un nickname y poseen la cantidad de horas laborales máximas que pueden estar asignados a los proyectos. Se dividen en dos tipos, desarrolladores, de los cuales se conoce la cantidad de años de experiencia, y project manager (PM), de los que se necesita llevar registro si tienen el curso aprobado de scrum master o no. De los proyectos se conoce su nombre, que los identifica. Se desea poder registrar para cada asignación de cada usuario a cada proyecto la cantidad de horas asignadas al proyecto, así como el costo que le insume a la empresa esa asignación. Se sabe que un usuario puede estar asignado a más de un proyecto al mismo tiempo, pero no superando su cantidad máxima de horas laborales que tiene permitida.



Se definieron las siguientes operaciones del sistema:

`AgregarAsignacion(usr:String, proy:String, horas:int, costo:float):boolean`	
Descripción	Crea una nueva asignación de un usuario a un proyecto, indicando true si pudo realizar la asignación o false en caso de que se superen las horas permitidas.
Parámetros	– `usr`: nickname del usuario – `proy`: nombre del proyecto – `horas`: cantidad de horas a asignar al proyecto – `costo`: importe del gasto que insume esa asignación
Precondiciones	– Existe un `Usuario u` con `nickname usr` – Existe un `Proyecto p` con `nombre proy` – El `Usuario u` no está asignado al `Proyecto p`
Postcondiciones	– Si la suma del atributo `horas` de las instancias `a` de `Asignacion` relacionadas al `Usuario u` más el valor de `horas` pasado por parámetro es mayor a `u.horasLaborales` se retorna `false`. – En caso contrario se crea una instancia de `Asignacion` entre la instancia de `Usuario u` y la instancia de `Proyecto p` con los valores de atributos `horas` y `costo` igual a los pasados por parámetro, y se retorna `true`.

`GenerarReporte():Set(DTReporte)`	
Descripción	Retorna una colección de la información de todos los usuarios del sistema con las horas y costos para cada uno. Para los PMs además se retorna los proyectos en los que están y el tamaño de sus equipos (cantidad de asignaciones)
Parámetros	No tiene
Precondiciones	No tiene
Postcondiciones	– Para cada instancia de `Usuario u` se genera un data value `d` de tipo `DTReporte` tal que `d.nickname = u.nickname`, `d.horasAsignadas` = suma de `a.horas` y `d.costoAsignacion` = suma de `a.costo`, siendo `a` cada una de las instancias de `Asignacion` asociadas con `u` – Si el usuario es `Desarrollador`, `d.experiencia = u.experiencia` – Si el usuario es `PM`, `d.cursoScrum = u.cursoScrum` y `proyectos` es igual al conjunto de data values `r` de tipo `DTAsigReporte` tal que para cada `Asignacion a` relacionada con `u` se toma `r.nombreProyecto = p.nombre`, donde `p` es el `Proyecto` relacionado con `a` y `r.tamanoEquipo` = cantidad de instancias de `Asignacion` asociadas con `p`



Se pide:

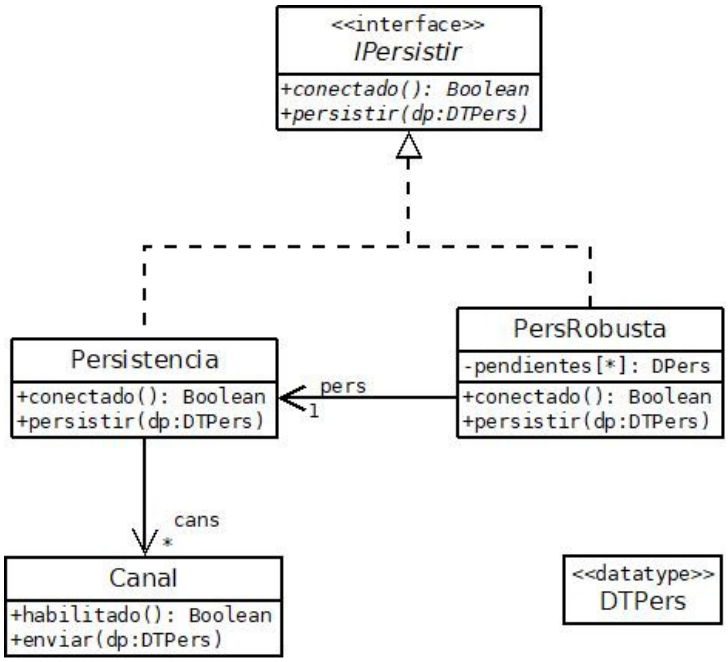
a) Realizar los Diagramas de Comunicación correspondientes a las operaciones especificadas, incluyendo visibilidades. Tener en cuenta que solo puede haber un controlador, en el cual solo existen colecciones de Usuarios y Proyectos.

b) Realizar el Diagrama de Clases de Diseño (DCD) resultante. No incluya getters ni setters. No incluya fábrica ni interfases.

<u>Problema 3</u> (35 puntos)

Se está construyendo un módulo de persistencia de datos para aplicaciones en la nube, cuyo diseño se muestra en la figura de la siguiente página. Las aplicaciones acceden al servicio de persistencia a través de la interface `IPersistir`, cuyas operaciones tienen los siguientes comportamientos:

- `conectado`: Devuelve `TRUE` si hay conexión, y por lo tanto es seguro utilizar la operación `persistir`. Devuelve `FALSE` si no hay conexión; en ese caso el comportamiento de la operación `persistir` es indefinido.
- `persistir`: Persiste un dato de tipo `DTPers`, siempre que haya conexión.



Una primera realización de `IPersistir` es la clase `Persistencia`, cuyas operaciones tienen los siguientes métodos:

- `conectado`: Devuelve `TRUE` si alguno de los canales de la colección `cans` está habilitado; `FALSE` en caso contrario.
- `persistir`: Persiste el dato a través del primer canal habilitado que se encuentre en la colección `cans`, utilizando la operación `enviar`.

Además, la clase `Persistencia` tiene un constructor que recibe un entero que indica la cantidad de canales que se deben crear y almacenar en la colección `cans`. Asuma que cuenta con un constructor por defecto en la clase `Canal`.

Por otra parte, se cuenta con una realización adicional de `IPersistir`, denominada `PersRobusta`, que utiliza una instancia de `Persistencia` y simula que siempre hay conexión. Los métodos en este caso son:

- `conectado`: Siempre devuelve `TRUE`.
- `persistir`: Si la instancia `pers` está conectada, entonces persiste los datos utilizando dicha instancia. En caso contrario, almacena los datos en la colección `pendientes` y la próxima vez que se invoque y `pers` esté conectada, persiste todos los datos pendientes y vacía la colección.

  El constructor por defecto de la clase `PersRobusta` debe crear la instancia `pers` de tipo `Persistencia` con una cantidad de canales igual a `NCAN`.

Se pide:

a) Nombrar el patrón de diseño utilizado en la solución, identificando las clases e interfaces participantes y sus roles.
b) Implementar (.h y .cpp) la interface `IPersistir` y las clases `Persistencia` y `PersRobusta`. Incluir los constructores y destructores necesarios.

Observaciones**:** Utilice colecciones paramétricas (contenedores STL). No incluya directivas al precompilador (#include, etc).