UNIVERSIDAD POLITECNICA DE VALENCIA

Prácticas de Laboratorio

Entrega 1

Ingeniería del Software

ETS Ingeniería Informática
DSIC – UPV

Alcance de la entrega

En la entrega 1 el alumno presentará el trabajo de implementación desarrollado en las últimas sesiones de prácticas sobre el caso de estudio ***Gestión de Actividades Deportivas***:

- Práctica 2: Diseño de Clases.
- Práctica 3: Entity Framework.

En concreto, el alumno deberá demostrar que ha implementado:

- La capa de negocio para todas las clases del modelo de diseño presentado en el boletín de la práctica 2.

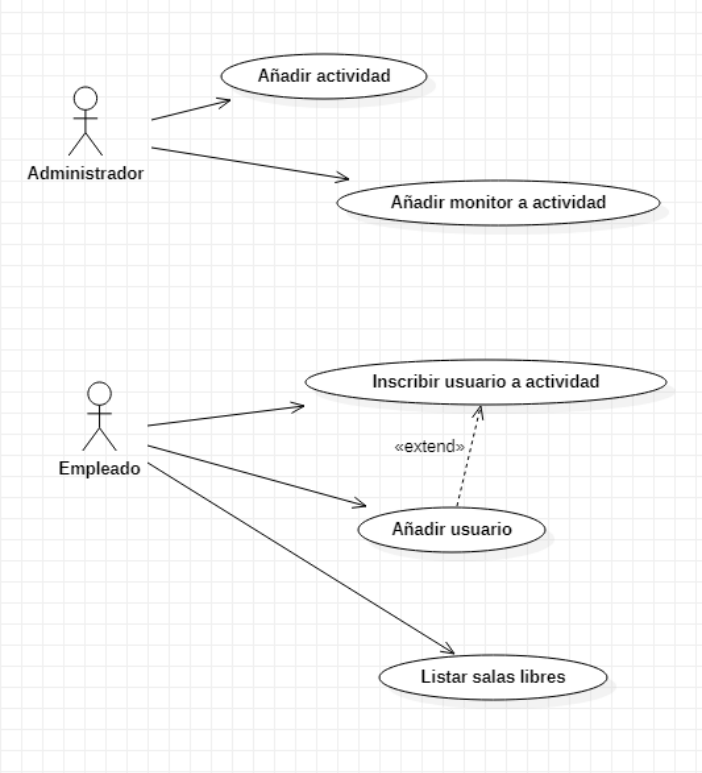
- La implementación completa de las capas de negocio y persistencia, siguiendo las guías de los respectivos boletines, para los siguientes casos de uso (tal y como se indicó en los boletines de la práctica 2 y 3, y los seminarios vistos en clase de teoría):

  - **Dar de alta una actividad** (actor: Administrador)
  - **Añadir monitor a una actividad** (actor: Administrador)
  - **Inscribir un usuario en una actividad** (actor: Empleado)
  - **Añadir usuario** (actor: Empleado)
  - **Listar las salas libres** (actor: Empleado)

  NOTA: la funcionalidad de estos casos de uso habrá que implementarlos en las sesiones 5, 6 y 7. La entrega se realizará en la sesión 8 de laboratorio, mire el calendario que se encuentra en Poliformat en Prácticas\calendario de prácticas.png

Descripción de los casos de uso a desarrollar

A continuación, se detalla la descripción en forma de plantilla de los casos de uso que habrá que implementar. Para esta entrega **solo** se pide la capa de negocio y persistencia. **No se pide la capa de interfaz gráfica**, que será objeto de la siguiente entrega, así como tampoco será necesario implementar una aplicación funcional de consola que interactúe con la capa de servicios.



En la implementación utilizaremos un único actor que podrá realizar todos los casos de uso.

ID	1
Caso de Uso	**Añadir actividad**
Actores	Administrador
Propósito	Añadir una nueva actividad
Descripción	1. El usuario introduce la información específica de la actividad que se va a impartir (fecha de comienzo, de finalización, días de la semana, salas, precios, etc.) 2. El sistema crea la nueva actividad.
Precond	--
Postcond	La actividad es creada y almacenada
Extensión síncrona	En 1, si la actividad se programa en fechas en las que las salas están ocupadas, el sistema mostrará un error y el empleado podrá cambiar los datos. En 1, el sistema comprueba que la fecha de comienzo es anterior a la fecha de finalización y que la fecha de comienzo es mayor que la fecha actual. Si no se cumple escribe un mensaje de error y pide al usuario que introduzca los datos de nuevo

ID	2
Caso de Uso	**Añadir monitor a una actividad**
Actores	Administrador
Propósito	Asignar un monitor a una actividad
Descripción	1. El sistema mostrará todas las actividades. 2. El usuario seleccionará una actividad. 3. El sistema mostrará la información detallada de la actividad incluyendo si la misma ya tiene un monitor asignado. 4. El sistema mostrará todos los monitores que pueden impartir dicha actividad atendiendo a las posibles restricciones horarias que puedan existir. 5. El usuario seleccionará el monitor que lo impartirá. 6. El sistema asignará el monitor a la actividad.
Precond	--
Postcond	La asignación es creada y almacenada en el sistema
Extensionesíncronas	Si en 4 no se encuentran monitores disponibles el sistema mostrará un mensaje de error y el caso de uso terminará.

Un monitor no puede impartir una actividad si en el lapso de tiempo que dura ésta su horario solapa con otros horarios de actividades que ya tiene asignadas. Una actividad tiene la siguiente información temporal:

- StartDate: el día que empieza la primera sesión de la actividad.
  Ejemplo: "08/02/2021"
- FinishDate: el día de la última sesión de la actividad.
  Ejemplo: "12/03/2021"
- ActivityDays: los días de la semana que se imparte la actividad.
  Ejemplo: lunes, miércoles y viernes (escritos en inglés)
- StartHour: hora a la que empieza una sesión de la actividad.
  Ejemplo: “9:30”
- Duration: tiempo que dura cada sesión de la actividad.
  Ejemplo: “0:45”

ID	3
Caso de Uso	**Inscribir un usuario en una actividad**
Actores	Empleado
Propósito	Inscribir un usuario en alguna de las actividades previstas a realizar
Descripción	1. El sistema muestra la lista de actividades cuya fecha de comienzo sea posterior a la actual o se estén impartiendo en la actualidad. 2. El empleado escoge una de las actividades. 3. El sistema solicita el dni del usuario. 4. El empleado introduce el dni. 5. El sistema busca el usuario por dni y muestra sus datos. Calcula el precio de la primera cuota y la muestra. 6. El empleado confirma la inscripción del usuario a la actividad. 7. El sistema almacena la inscripción y cobra la primera cuota.
Precond	-
Postcond	Una nueva inscripción queda registrada en el sistema, junto al primer pago.
Extensión síncrona	En 5, si el usuario no es encontrado, se preguntará si debe dar de alta un nuevo usuario y llamará en caso afirmativo al caso de uso "añadir usuario". En 5, si el usuario ya está asignado a esa actividad el sistema informa del error.

*Nota: en nuestro modelo, el dni es la propiedad Id de Person

ID	4
Caso de Uso	**Añadir usuario**
Actores	Empleado
Propósito	Añadir un nuevo usuario
Descripción	1. El empleado introduce la información específica de un nuevo usuario (nombre, dirección, cuenta bancaria, dni, fecha de nacimiento, si está jubilado, etc.) 2. El sistema crea el nuevo usuario.
Precond	--
Postcond	El usuario es creado y almacenado
Extensión síncrona	En 1, si el usuario existe (dni), el sistema mostrará un error y el empleado podrá cambiar los datos.

*Nota: en nuestro modelo, el dni es la propiedad Id de Person

ID	5
Caso de Uso	**Listar las salas libres**
Actores	Empleado
Propósito	Proporcionar información de las salas que quedan libres una semana concreta
Descripción	1. El sistema pide la fecha del lunes de la semana para la que va a hacer el cálculo. 2. El sistema calcula las salas libres que habrá en cada franja horaria*, generando una tabla como la que se muestra a continuación.
Precond	-
Postcond	
Extensión síncrona	

* Franja horaria: tramos de 45 minutos que comienzan con la hora de apertura del gimnasio.

A continuación, se muestra un **ejemplo** de tabla del listado:

Gimnasio 1 - Semana del 16/11/2020

	Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**
08:00-08:45	6	6	6	6	6	6
08:45-09:30	6	6	6	6	6	6
09:30-10:15	3	6	3	6	3	2
10:15-11:00	0	3	0	3	0	2
11:00-11:45	6	6	6	6	6	2
11:45-12:30	6	6	6	6	6	2
12:30-13:15	6	6	6	6	6	2
13:15-14:00	6	6	6	6	6	2
…	…	…	…	…	…	…

Detalles de la capa lógica

En las sesiones de práctica anteriores se ha construido la parte de la funcionalidad que tiene que ver con el diseño de las clases y su almacenamiento persistente.

Para este entregable se debe implementar el controlador (o proveedor de servicios) de la lógica de negocio. Dicho controlador deberá ofrecer a la capa superior (IU) acceso a la funcionalidad que ésta necesite.

El conjunto de los servicios ofrecidos a la IU debe estar definido en una interfaz llamada `IGestDepService`. Los métodos de dicha interfaz se implementarán en una clase llamada `GestDepService`. La interfaz y su implementación deben encontrarse en la carpeta `BusinessLogic/Services` y formar parte del espacio de nombres `GestDep.Services`. Todos los errores que puedan surgir en la ejecución de los métodos implementados se deberán reportar mediante excepciones.

Para reportar los errores generados en la capa lógica, en la carpeta `BusinessLogic/Services` se deberá crear una subclase de `Exception` llamada `ServiceException`. Cuando se creen objetos de esta clase para informar de algún error, la propiedad `Message` de dicha clase contendrá el mensaje de texto con el error generado.

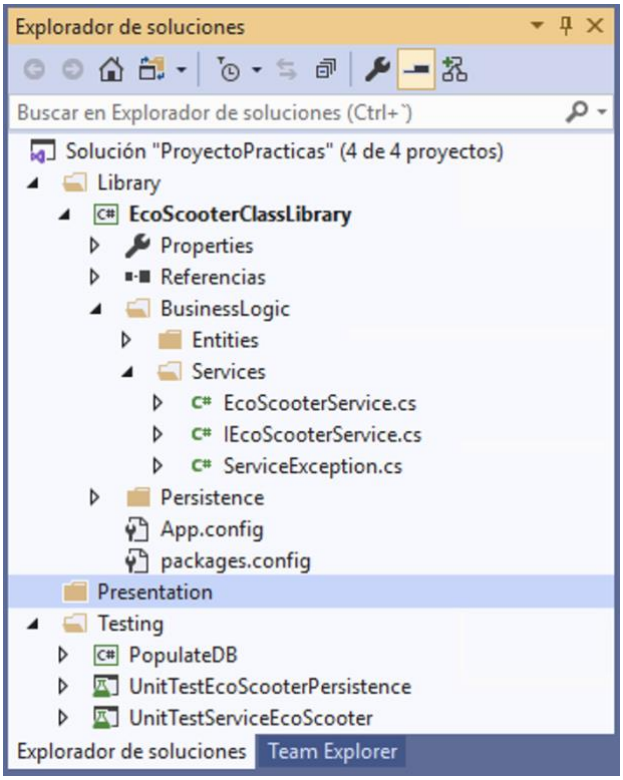
Por ejemplo, en el caso de estudio de referencia que hay en PoliformaT llamado *EcoScooter*, cuando se quiere añadir un usuario sería un error añadirlo si ya existe una con el mismo dni que la que se quiere crear. A continuación, se puede ver un fragmento de código del método correspondiente en el que antes de añadir una persona se comprueba que no exista una con el mismo dni. En caso contrario, se lanza una excepción.

```
public void RegisterUser(DateTime birthdate, string dni, …)
{
    […]
     if (ecoScooter.ExistsPerson(dni))
            throw new ServiceException("Person already exists.");
     ecoScooter.AddUser(birthDate, dni, …);
        saveChanges();
}
```

Con el objetivo de facilitar la identificación de los métodos a implementar en *GestDepService*, en PoliformaT se podrá descargar un

proyecto que contiene un programa de prueba de la lógica. En concreto, este programa asume que existe un proveedor de servicios (o controlador) que implementa ciertos métodos y los probará. Este programa contendrá las **pruebas mínimas** que debe superar el `GestDepService` implementado.

En la siguiente figura se puede ver la apariencia que tiene el explorador de soluciones para el ejemplo *EcoScooter* con los contenidos creados en la sesión. Destaca la interfaz `IEcoScooterService`, su implementación en una clase llamada `EcoScooterService` y la clase excepción.



Diseño de la interfaz de servicios

Para reducir la complejidad, la aplicación que se implementará utilizará únicamente un CityHall y un Gym, que contienen la información de las salas, los instructores, los usuarios, etc. Además, consideraremos que solo hay un actor que interactúa con el sistema.

El archivo `IGestDepService` proporcionado define un conjunto mínimo de servicios necesarios para poder implementar los casos de uso. Todos los servicios deben recibir y devolver tipos de datos simples: strings, doubles, integers, etcétera, con el fin de conseguir un

desacoplamiento entre las capas de lógica y presentación. De este modo, si se cambia algo de la capa de lógica, la de presentación no se verá afectada.

Este conjunto de servicios se ha diseñado teniendo en cuenta los pasos descritos en los casos de uso. Para cada paso, el caso describe qué necesita el usuario y, para poder mostrar la información cuando sea necesario, la capa de presentación necesitará de un servicio de la capa lógica que se la proporcione. Para realizar una acción final (una inserción, una modificación, etcétera) se necesitará otro servicio que la lleve a cabo. Por tanto, de todo esto se puede deducir que cada caso de uso necesita más de un servicio.

Dentro de la clase `IGestDepService` se han definido regiones que agrupan los servicios por caso de uso. Por favor, revisadlos y leed la documentación proporcionada para cada servicio para entender cómo deben ser implementados.

Como ejemplo, podéis ver los servicios necesarios para implementar el caso de uso de añadir un monitor a una actividad en la tabla que se muestra a continuación. En este caso, necesitaremos cinco servicios, y todos ellos intercambian datos básicos con la capa de presentación.

CU2. Añadir monitor a actividad	
Paso del CU	*Servicio*
1. El sistema mostrará todas las actividades.	`GetAllActivitiesIds` `GetActivityDataFromId`
2. El administrador seleccionará una actividad.	
3. El sistema mostrará la información detallada de la actividad, incluyendo si ya tiene un monitor asignado.	`GetActivityDataFromId`
4. El sistema mostrará todos los monitores que puede impartir esta actividad, considerando las posibles restricciones horarias que puedan existir.	`GetAvailableInstructorsIds` `GetInstructorDataFromId`
5. El administrador seleccionará el monitor que impartirá la actividad.	
6. El sistema asignará el monitor a la actividad.	`AssignInstructorToActivity`

Consideraciones de la implementación de los servicios

Algunos de los métodos de los servicios deberán realizar comprobaciones como, por ejemplo, si un usuario ya existe o si hay monitores disponibles. La manera de realizar estas comprobaciones no es accediendo directamente al DAL. La información debe recuperarse desde las clases del modelo. Por ejemplo, en caso de querer saber si un usuario existe, se debe acceder a la colección de Person (People) de City Hall. Así, dentro de CityHall se tendrá que implementar un método que compruebe si un usuario de la colección de People que mantiene es igual al que se recibe para comparar.

Por otra parte, se debe tener en cuenta que, cuando se añade un nuevo elemento a la base de datos, el objeto debe respetar las relaciones indicadas en el modelo, de forma que, antes de la inserción, se debe añadir a cualquier lista o propiedad con la que esté relacionada de acuerdo con el modelo. Por ejemplo, cuando se crea un nuevo Payment, se debe añadir no solo a la lista que mantiene el Enrollment correspondiente, sino también a la lista de Payment que mantiene CityHall.

Por último, se debe recordar que cuando se modifica el estado de un objeto que ya existe en la base de datos, EntityFramework realiza todas las modificaciones necesarias en la base de datos para reflejar el estado actual del objeto después de llamar al método Commit del DAL a través del método ya implementado `SaveChanges()`. Por ejemplo, si modificáis la lista de Person de CityHall añadiendo una nueva instancia (es decir, una nueva persona), no es necesario hacer un insert explícito (`dal.Insert()` del objeto), ya que EntityFramework lo hará automáticamente. Sí es necesario, sin embargo, llamar al método `SaveChanges()`.

Fecha de entrega

La fecha de entrega será la correspondiente a la 8ª semana de prácticas en los laboratorios. **TODOS** los miembros del grupo **deben de acudir (obviamente de manera virtual) a la entrega** ya que pueden ser requeridos para responder preguntas sobre el proyecto.

Forma de entrega

Existirá una tarea en PoliformaT en la que cada grupo deberá subir el fichero comprimido del proyecto. También se realizará una operación de proteger sobre el repositorio de Team Services con el comentario "Entregable 1". El proyecto debe subirse a la tarea **ANTES** del comienzo de la sesión de evaluación.

Evaluación

En la sesión de laboratorio, el profesor evaluará a cada uno de los grupos comprobando que se ha implementado correctamente la funcionalidad solicitada y **haciendo preguntas a TODOS los miembros del grupo**. Además, **será obligatorio tener la cámara encendida durante la evaluación del entregable.** Se utilizará la rúbrica de evaluación adjunta para valorar la funcionalidad implementada.