Obligatorio 2 de Diseño de Aplicaciones

Para la segunda entrega se debe tener en cuenta los siguientes cambios o agregados a la letra de la primera entrega.

Requerimientos Administradores:

Configuración sistema de puntajes: Se deberá proveer una pantalla para asignar puntaje a cada uno de los siguientes eventos:

- Crear pizarrón
- Eliminar un pizarrón
- Agregar un elemento
- Agregar un comentario
- Marcar un comentario como resuelto

Reiniciar sistema de puntos: Para cada equipo, se podrá reiniciar los puntajes de los usuarios para el equipo seleccionado. En tal caso, la suma de puntajes

Requerimientos Genéricos (Tanto colaboradores como administradores):

Puntos de experiencia: Con el fin de fomentar el trabajo en equipo, se agrega una tabla de puntajes por equipo y que mostrará, en cada equipo, la contribución que realiza cada usuario.
El puntaje de cada usuario se acumula a medida que realiza las acciones, según el valor configurado por el administrador. Cada usuario mantendrá un sistema de puntos para el equipo al que pertenece. Es decir: Si un usuario pertenece a más de un equipo, las acciones que realice sobre los elementos de uno de esos equipos no acumulan puntaje para el ranking del otro equipo.
Cada usuario deberá poder ver el ranking de cada equipo al que pertenezca.

Conexión de elementos: Se deberá poder agregar conexiones entre los elementos. Se entiende por conexión un elemento gráfico dibujado que conecte a dos elementos. Cada conexión tendrá dos elementos, un nombre, una descripción (opcional) y podrá tener direccionalidad, bidireccionalidad o ninguna de las dos. La direccionalidad deberá mostrarse con una flecha.

Persistencia de los datos
En esta nueva versión, se requiere que todos los datos del sistema sean persistidos en una base de datos. De esta manera, la siguiente vez que se ejecute la aplicación se comenzará con dichos datos cargados con el último estado guardado antes de cerrar la aplicación. Los datos se deben ir guardando en la base de datos durante la operativa de la solución, a medida que el usuario realiza las acciones se debe ir almacenando, y **no** con un botón que salve todo en un bloque o al cerrar la aplicación.

Aplicación a entregar

Se debe entregar una aplicación que contemple toda la funcionalidad descrita en este documento.
Se espera que la aplicación se entregue con una base de datos con datos de prueba, de manera de poder comenzar las pruebas sin tener que definir una cantidad de datos iniciales. Dichos datos de prueba deben estar adecuadamente especificados en la documentación entregada. Por lo que la fuente de datos de prueba **no** debe estar embebida en el código como en la primera entrega.

Instalación
El costo de instalación de la aplicación debe de ser mínimo y documentado adecuadamente.

Tecnologías y herramientas de desarrollo

- Microsoft Visual Studio .NET 2015 (lenguaje C#)
- Astah o cualquier otra herramienta UML
- Entity Framework 6.1.3

NOTA: La totalidad y detalle de los requisitos serán relevados a partir de consultas en el foro correspondiente en aulas. Para evitar complejidades innecesarias se realizaron simplificaciones al dominio del problema real.

Persistencia en base de datos

Toda la información contenida en el sistema debe ser persistida en una base de datos Microsoft SQL Server Express 2014. El diseño debe contemplar el modelado de una solución de persistencia adecuada para el problema utilizando Entity Framework (Code First).

Se espera que como parte de la entrega se incluya dos respaldos de la base de datos: uno vacío y otro con datos de prueba. Se recomienda entregar el archivo *.bak* y también el script *.sql* para ambas base de datos.
Es condición necesaria para obtener el puntaje mínimo del obligatorio que al menos una entidad del sistema pueda ser persistida.

Mantenibilidad (se mantiene lo solicitado en la primera entrega)

La propia empresa eventualmente hará cambios sobre el sistema, por lo que se requiera un alto grado de mantenibilidad, flexibilidad, calidad, claridad del código y documentación adecuada.
Por lo que el desarrollo de todo el obligatorio debe cumplir:

- ➔ Estar en un repositorio **Git**.
- ➔ Haber sido escrito utilizando **TDD** (desarrollo guiado por pruebas) lo que involucra otras dos prácticas: escribir las pruebas primero (Test First Development) y refactoring. De esta forma se utilizan las pruebas unitarias para dirigir el diseño.
- ➔ Cumplir los lineamientos de **Clean Code** (capítulos 1 al 10, y el 12), utilizando las técnicas y metodologías ágiles presentadas para crear código limpio.

Pruebas unitarias (se mantiene lo solicitado en la primera entrega)

Se requiere escribir los casos de prueba automatizados con Visual Studio Unit Tests, documentando y justificando las pruebas realizadas.

Como parte de la evaluación se va a revisar el nivel de cobertura de los test sobre el código entregado. Por lo que se debe entregar un reporte y un análisis de la cobertura de las pruebas.

Control de versiones (se mantiene lo solicitado en la primera entrega)

La gestión del código del obligatorio debe realizarse utilizando el repositorio Git de **Github** en la organización ORT-DA1 (github.com), apoyándose en el flujo de trabajo recomendado **GitFlow** (nvie.com/posts/a-successful-git-branching-model).

Como clientes visuales desktop para el manejo del repositorio, pueden utilizar:

- ➔ Visual Studio with Git (msdn.microsoft.com/es-es/library/hh850437.aspx).
- ➔ SourceTree (www.atlassian.com/software/sourcetree) el cual incorpora GitFlow (www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow).

Al realizar la entrega se debe realizar un release en el repositorio con lo mismo entregado en bedelía.

Documentación

La documentación entregada debe ser en **un solo** documento impreso y digital, que contenga la siguiente información ordenada e indexada:

1. Descripción general del trabajo (1/2 carilla): si alguna funcionalidad no fue implementada o si hay algún error conocido (bug) deben ser descritos aquí.
2. Justificación de diseño, explicando los mecanismos generales y descripción de las principales decisiones de diseño tomadas.
3. Diagrama de paquetes.
4. Diagramas de clases: al menos uno por paquete.
5. Diagramas de interacción para especificar al menos uno por paquete.
6. Modelo de tablas de la estructura de la base de datos.
7. Evidencia de Clean Code.
8. Diseño de los casos de prueba funcionales generados y los casos de prueba concretos (con valores a ingresar, resultados esperados y resultado obtenido). Resultado de la ejecución de las pruebas. Evidencia del código de pruebas unitarias (en el repositorio), reporte de la herramienta de cobertura y análisis del resultado.
9. El documento debe cumplir con los siguientes elementos del Documento 302 (http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf) de la facultad.
    a. Capítulo 1, secciones 1.1 (Sin la leyenda), 1.5, 1.7, 1.8 y 1.9
    b. Capítulos 4, 5 y 9

10. Se debe actualizar todos los diagramas que corresponda, y agregar los que se considere necesario para mostrar los cambios realizados y justificar las decisiones tomadas. Para esta segunda entrega es necesario repetir la documentación de diseño de la primera entrega actualizada (no incluir la documentación de análisis).

Las condiciones de entrega serán evaluadas como si se le estuviese entregando a un cliente real: prolijidad, claridad, profesionalismo, etc.

La entrega debe ser en medio de almacenamiento CD o DVD (una sola copia como respaldo, ya que se tiene acceso al repositorio) y acceso a los docentes al repositorio Git utilizado por el grupo. Se debe incluir:

- ➔ Una carpeta con la aplicación **compilada en release**.
- ➔ Código fuente de la aplicación, incluyendo el proyecto que permita probar y ejecutar.
- ➔ Documentación impresa y digital (incluyendo modelado UML).
- ➔ Base de datos: entregar una base de datos vacía y otra con datos de prueba.

Evaluación (20 puntos)

	Evaluación de	**Puntos**
Demo al cliente	Cada equipo deberá realizar una demostración al cliente de su trabajo. Dentro de los aspectos que un cliente espera se encuentran: ● Ver la demo cuando él esté listo y no tener que esperar a que el equipo se apronte. ● Probar la solución y que la misma funcione sin problemas o con problemas mínimos, y de buena calidad de interfaz de usuario. ● Conocer las capacidades de cada uno de los integrantes del equipo, pudiendo preguntar a cualquier integrantes sobre la solución, su diseño, el código y sobre cómo fue construida, y así apreciar que fue un trabajo en equipo. Todos los integrantes deben conocer toda la solución. ● Verificar el aporte individual al trabajo por parte de cada uno de los integrantes del equipo y en función de los resultados, se podrán otorgar distintas notas a los integrantes del grupo. Se espera que cada uno de los integrantes haya participado en la codificación de parte significativa del obligatorio. Esta demostración al cliente hará las veces de **defensa del trabajo.** NOTA: El incorrecto funcionamiento de la instalación puede significar la no corrección de la funcionalidad. En el caso de defensa en el laboratorio, durante la defensa cada grupo contará con 15 minutos para la instalación de la aplicación. Luego de transcurridos los mismos se restan puntos al trabajo.	6
Diseño y documentación	Diagramas de paquetes Diagramas de clases Uso adecuado de los diagramas de interacción Justificación del diseño y patrones Uso adecuado de patrones de diseño Diagramas y justificación de diseño de la solución de persistencia Documentación y justificación de los cambios realizados a la primera entrega Calidad del diseño Modelo de tablas	7

	Informe de Clean Code y pruebas Claridad, completitud y prolijidad de la documentación Organización de la documentación (debe tener un orden lógico y un índice)	
Implementación	Desarrollo guiado por las pruebas (TDD) y técnicas de refactoreo de código Buenas prácticas de estilo y codificación y su impacto en la mantenibilidad (Clean Code) Correcto uso de las tecnologías Claridad del código Concordancia con el diseño Implementación de acceso a base de datos	7

Información importante

Lectura de obligatorio: 16-05-2017 Plazo máximo de entrega: 22-06-2017

Defensa: A definir por el docente Puntaje mínimo / máximo: 10 / 20 puntos

Los grupos de obligatorio se forman como máximo por 2 estudiantes. Todas las entregas se realizan en Bedelía (oficina 305) con boleta de entrega de obligatorio, y hasta las 20 hs del día de entrega.