Obligatorio 1 de Diseño de Aplicaciones

El avance de la tecnología ha generado una rápida evolución en las herramientas de colaboración disponibles para que los equipos puedan trabajar en conjunto. Herramientas para gestionar documentos compartidos, archivos sincronizados y presentaciones han estado presentes desde hace más de una década. Sin embargo, Ud se ha dado cuenta que existe un espacio no aprovechado en el mercado, de cambiar el mecanismo de escritura y carga de archivos y cuadros de texto.

Luego de mucho trabajo, ha conseguido que la empresa CleanCoder apoye su proyecto, y lo contrate para diseñar un sistema que permita a sus equipos colaborar en un documento en formato pizarrón, en el cual se podrán colgar tanto cuadros de imágenes cómo cuadros de texto, permitiendo a múltiples usuarios trabajar sobre el mismo pizarrón a la vez.

El sistema deberá soportar dos tipos de usuarios; administrador y colaborador, teniendo en cuenta que todo usuario administrador es también un colaborador.

Requerimientos Administradores:

ABM usuarios: De cada usuario se conoce su nombre, apellido, email, fecha de nacimiento y contraseña. Un administrador puede generar una contraseña por defecto o reiniciar la contraseña del usuario.

ABM equipo: De cada equipo se conoce su nombre, la fecha de creación, una breve descripción de sus tareas y la cantidad máxima de usuarios que pueden pertenecer al mismo. En todo momento, un equipo debe tener mínimo un usuario.

Informes:

- **Pizarrones creados por equipo:** Se desea obtener una rápida vista de la cantidad de pizarrones creados por equipo, filtrando por fecha de creación y equipos. Se deberá mostrar: equipo creador, fecha de creación, fecha de última modificación, cantidad de elementos en el pizarrón.
- **Comentarios resueltos por usuario:** mostrará los datos de comentarios resueltos por usuario, filtrando por fecha de creación de comentario, de resolución del mismo, de usuario creador y del usuario que lo resuelve. Los datos a mostrar son: fecha de creación de comentario, usuario creador, usuario que resuelve, pizarrón al que pertenece el elemento y fecha de resuelto el comentario.

Requerimientos Genéricos (Tanto colaboradores como administradores):

ABM pizarrones: de cada pizarrón se conoce su nombre, el equipo al que pertenece, una breve descripción, un ancho y un alto. Todos los usuarios del equipo al que pertenece pueden ver y modificar el pizarrón, pero solo el creador o un administrador pueden eliminarlo.

Visualizar un pizarrón: A partir de la lista de pizarrones de un equipo, se podrá seleccionar uno y abrirlo. Esto resultará en una ventana de la cual se podrán colgar elementos, moverlos y eliminarlos, como se describe en el

siguiente punto. Al agregar un elemento a un pizarrón, este debe aparecer al instante, y no al abrir y cerrar la ventana.

Agregar un elemento a un pizarrón: Existen dos tipos de elementos; cuadro de texto e imagen. Un elemento tiene un ancho, una altura, una lista de comentarios y un punto de origen (basado en su esquina superior izquierda), relativo al punto superior izquierdo del pizarrón. Para colocar el elemento en su posición, se realizará arrastrando el elemento a lo largo del pizarrón. El tamaño inicial del el elemento puede estar definido por defecto, y permitir al usuario agrandar o achicarlo a gusto.

- Cuadro de texto: tiene un contenido, el cual puede tener distinto tamaño y tipo de letra.
- Imagen: Puede cargarse un archivo jpg, png, gif o jpeg.

Modificar un elemento de un pizarrón: Desde la ventana de visualización de un pizarrón, se podrá modificar los elementos que lo componen. Esta modificación puede ser tanto de su ubicación en el pizarrón, tamaño o contenido. Para realizar la modificación, el mecanismo que deberá ofrecerse al usuario es visual. Esto quiere decir que para modificar la posición de un elemento en la pizarra podrán seleccionar y arrastrar un elemento en la misma. Para modificar el tamaño de un elemento, deberán seleccionar uno de sus bordes o esquinas y demarcar el tamaño final.

Agregar un comentario a un elemento: Cada usuario que participe del equipo podrá comentar sobre el pizarrón, para fomentar la colaboración. Una vez que el contenido del comentario haya sido tomado en cuenta, los usuarios involucrados en el pizarrón podrán marcar como resuelto el comentario. De los comentarios se conoce la fecha de creación, fecha de resolución, usuario creador y usuario que lo resuelve. Un comentario solo puede estar asociado a un elemento.

Eliminar un elemento de un pizarrón: se podrán eliminar elementos de los pizarrones, o pizarrones completos.
Imprimir un pizarrón a PDF o png. (Image, Bitmap): En caso de desearlo, se podrá generar un archivo pdf o png que muestre el pizarrón. **NOTA:** Se recomienda utilizar la librería iText para el manejo de pdf (https://github.com/itext/itextsharp)

Aplicación a entregar

Se debe entregar una aplicación que contemple toda la funcionalidad descrita en este documento. Para esta primera instancia la solución guardará toda la información en memoria, no siendo necesario el uso de base de datos (esto se implementará en la segunda entrega).

Además se espera que la aplicación contenga una opción en su menú principal que permita generar 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.

Usabilidad
La aplicación debe ser fácil de utilizar, intuitiva y atractiva.

Tecnologías y herramientas de desarrollo

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

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.

Mantenibilidad

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 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

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. Se debe documentar el análisis de requerimientos, construyendo un documento de especificación de requerimientos (ESRE mínimo), en el cual deben estar identificados y descritos todos los casos de uso, así como los requerimientos no funcionales. Se espera además un modelo de análisis del dominio (modelo conceptual).
3. Justificación de diseño, explicando los mecanismos generales y descripción de las principales decisiones de diseño tomadas.
4. Diagrama de paquetes.
5. Diagramas de clases: al menos uno por paquete.
6. Evidencia de Clean Code. Breve explicación de por qué entiende que su código se puede considerar limpio, y en caso de existir, aclaración de oportunidades de mejora o aspectos en los que cree que no cumple con el estándar.
7. 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.

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).

Entrega informe (corrección cruzada)

Al día siguiente de la entrega del obligatorio, la cátedra asignará a cada grupo un obligatorio de otro grupo, de forma que puedan realizar un análisis exhaustivo del mismo en un plazo de una semana (ver fechas de entregas al final del presente documento).

Este informe contendrá los siguientes puntos:

- ➔ Corrección de la funcionalidad pedida. Se debe ejecutar la aplicación (desde la carpeta con la aplicación compilada en release) y probar toda la funcionalidad marcando si fue implementada correctamente o no fue implementada o si fue implementada con errores (especificando los mismos).
- ➔ Impacto nueva funcionalidad. La cátedra enviará el detalle de una nueva funcionalidad deseada en el

sistema para que sea evaluado el impacto de realizar dicha implementación en la solución. No se deberá implementar los cambios explícitamente sino que documentar el impacto de los cambios (código que cambia, elimina, o agrega).

- ➔ Conformidad TDD. Se debe evaluar el proceso de desarrollo guiado por pruebas realizado por el grupo mediante el análisis de los commits en el repositorio, el código y las pruebas unitarias.
- ➔ Conformidad Clean Code. Se debe analizar el código de la solución para realizar un informe en cuanto al cumplimiento de los lineamiento de clean code.

Evaluación (25 puntos)

	Evaluación de	**Puntos**
Funcionalidad	Implementación de la funcionalidad pedida y calidad de la interfaz de usuario.	7
Diseño y documentación	Requerimientos funcionales y no funcionales Identificación de actores Diagrama de casos de uso Casos de uso con cursos alternativos / excepción Modelo conceptual Diagramas de paquetes Diagramas de clases Justificación adecuada de la solución y diseño Calidad del diseño Informe de Clean Code y pruebas Claridad de la documentación Organización de la documentación (debe tener un orden lógico y un índice) Completitud de la documentación Prolijidad de la documentación	5
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 Concordancia con el diseño	8
Informe	Corrección cruzada	5

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 restará puntos al trabajo.

Defensa

La defensa del trabajo intenta:

- ➔ Evaluar el conocimiento general de los integrantes del grupo sobre la solución propuesta. 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.

El mecanismo de defensa se determinará al momento de la entrega pudiendo ser el mismo escrito o en el laboratorio.

Información importante

Lectura de obligatorio:	03-04-2017	
Plazo máximo de entrega obligatorio:	15-05-2017	Puntaje mín. / máx. Obligatorio 1: 10 / 20 puntos
Plazo máximo de entrega informe:	23-05-2017	Puntaje mín. / máx. Informe: 0 / 5 puntos

Defensa: A definir por el docente

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.