Obligatorio 1 de Diseño de Aplicaciones

Un colegio lo contrata a usted y su equipo para desarrollar un sistema del tipo *Enterprise Resource Planner* (ERP) con el objetivo de gestionar y manejar de forma más eficiente las operaciones del colegio.

Un ERP es un sistema cuyo objetivo es permitir la administración de distintas áreas de una organización y normalmente está compuesto por distintos módulos que ayudan a manejar las distintas áreas del sistema.

A usted se le encarga desarrollar la plataforma ERP *extensible*, así como también X módulos que deberá implementar sobre la plataforma. Por extensible se entiende que debe ser trivial agregar un nuevo módulo a la plataforma, sin que ello implique alterar el código principal de la misma.

A continuación, se describen los módulos que deberá desarrollar:

1. Gestión de Materias
    a. Alta, Baja y Modificación de cada Materia. Se conoce su nombre, un código (alfanumérico), los alumnos que participan en ella, los docentes que la dictan.
2. Gestión Docente
    a. Alta, Baja y Modificación de docentes. De cada docente se conocerá su Nombre, Apellido y la(s) materia(s) que dicta.
    b. Listado de docentes, mostrando nombre, apellido y materias que dicta.
3. Gestión de Alumnos
    a. Alta, Baja y Modificación de alumno. De cada alumno se conoce su número de estudiante, su cédula de identidad, su nombre, apellido, y las materias a las que está inscripto
    b. Listado de alumnos, mostrando su nombre, apellido, número de estudiante. Al seleccionar un estudiante debe mostrar las materias a las que está inscripto.
4. Gestión de Camionetas
    a. El colegio ofrece un servicio de camionetas para recoger a los alumnos del colegio. El módulo permite:
        i. Alta, Baja y Modificación de Camionetas.
        ii. Determinar cuántas camionetas hay a disposición, la capacidad de cada una, y agrega a la información de los alumnos dos coordenadas *x* e *y*, decimales ambas que es la posición relativa a la escuela que se encuentra en el 0,0.
    b. Además, este módulo permite calcular las rutas que cada camioneta debe hacer para completar su capacidad recorriendo la menor distancia[1] posible. (Las camionetas siempre parten desde la escuela).
    c. Como es la primera versión, asuma que primero deben salir las camionetas de mayor capacidad.

De cada módulo se conoce su Nombre, una descripción, el menú principal del módulo y las acciones que permite realizar (cada acción es una opción del menú), que abre una pantalla específica.

[1] La distancia se calculará como la distancia entre dos puntos, sin considerar calles ni posibilidad de que haya calles flechadas

GRUPOS DE 3 ESTUDIANTES

Los grupos de 3 estudiantes deberán implementar además el siguiente Módulo:

1. Gestión de Actividades
    a. Alta, baja y modificación de Actividades, de las que se conoce el Nombre, el Id, la fecha y el costo
    b. Participantes (estudiantes)
2. Gestión de Pagos de Cuotas y Actividades
    a. Agregar a la información de un alumno las cuotas que el estudiante pagó (se consideran 12 por año)
    b. Cada vez que un alumno “paga” una cuota (proceso manual que se hace a través de este módulo), se le registra esa cuota pagada. De la cuota se conoce el mes y el año a la que corresponden (ejemplo 1 (Enero), 2016)
    c. Además de las cuotas, se desarrollan actividades a lo largo del año, por lo que se puede anotar un estudiante a esa actividad al pagarla.

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** (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 (incluyendo documentación).

Inmediatamente luego de la primera entrega se le debe dar acceso al repositorio a un integrante de otro grupo (grupo a asignar por los docentes el día siguiente a la entrega) para que pueda realizar el informe correspondiente a la corrección cruzada.

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).
8. 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).

Para esto, cada grupo dará acceso a su repositorio a los docentes y a un integrante del otro grupo.

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)	5

	Completitud de la documentación Prolijidad de la documentación	
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 restarán 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:	12-09-2017	
Plazo máximo de entrega obligatorio:	17-10-2017	Puntaje mín. / máx. Obligatorio 1: 10 / 20 puntos
Plazo máximo de entrega informe:	25-10-2017	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.