fidélitas UNIVERSIDAD

Ingeniería en Sistemas de Computación

Proyecto Final **Curso:** Programación Avanzada
Prof:
Código curso: SC-601

1. Especificación

Proyecto programado grupal a desarrollar para su entrega y exposición en Semana 14/15.

Los grupos deben estar conformados por un mínimo de 3 estudiantes y un máximo de 4. Los problemas entre compañeros del grupo deben solucionarse a lo interno de cada grupo.

2. Código

El código debe entregarlo durante la primera hora de clase en la semana 14, después de esto no se recibirán proyectos, y perderá la calificación de esta evaluación. Bajo ninguna condición existen prórrogas.

Un sólo integrante por grupo subirá al campus virtual un único archivo .zip con su proyecto exportado de .NET.

El archivo debe llevar como nombre la siguiente convención:

SC-601-PA-G[numero_de_grupo]-[inicial_del_dia_de_la_clase]

Ejemplo: SC-601-PA-G1-L.zip

Dentro de ese archivo Zip, agregue un archivo de texto con el nombre **readme.txt (puede ser un readme.md también)** donde indique:

1. Integrantes finales del grupo. A los que se les asignará la nota del proyecto
2. Enlace del repositorio si lo subió en GitHub o en algún otro.
3. Especificación básica del proyecto:
    a. Arquitectura del proyecto (tipos de proyectos que utilizo y contiene el programa)
    b. Libraries o paquetes de nuget utilizados
    c. Principios de SOLID y patrones de diseño utilizados

Nota Final de Proyecto
El proyecto será calificado según la rúbrica que se presenta en el programa del curso.

Se evaluarán los temas según hayan sido vistos en clase. Si cumple con los requisitos especificados se asignan todos los puntos del tema, sino se descuentan según se incumpla.

La funcionalidad principal debe ser programada por los estudiantes desde cero. El código de cada grupo debe ser desarrollado por cada grupo por separado. Puede incluir librerías o paquetes de **nuget** pero debe especificarlo en el *readme*.

fidélitas UNIVERSIDAD

Ingeniería en Sistemas de Computación

Proyecto Final **Curso:** Programación Avanzada
Prof:
Código curso: SC-601

Consideraciones generales:

- Se debe realizar las validaciones que se consideren necesarias como no repetir Ids.
- Debe utilizar de forma adecuada los tipos de variables, métodos o de arreglos según cada necesidad.
- Debe escribir el código respetando las convenciones de ***naming, coding*** y *de* ***style*** para nombres de clases, atributos, métodos y con una correcta indentación.
- Debe contar con un menú que permita la selección de las opciones que vayan a ser presentadas, en el caso de los reportes, presenta un submenú.

3. Enunciado

Desarrollo de un sistema de ejecución de un *Queue* de tareas

La empresa desarrolladora de software a la medida (SaM) necesita desarrollar un sistema que permitirá la ejecución **automatizada** de tareas en una cola de forma sincrónica. Las tareas pueden estar relacionadas con procesamiento de datos, envío de correos electrónicos, generación de reportes, ejecución de scripts etc. El sistema manejará el procesamiento de una tarea a la vez y priorizará la ejecución en función de las necesidades del usuario.

Especificaciones Funcionales

a) Gestión de Tareas

- Los usuarios podrán crear tareas, asignarles prioridad (alta, media, baja) y definir una fecha de ejecución.
- Cada tarea tendrá estados como "Pendiente", "En Proceso", "Finalizada", “Fallida".
- El sistema permitirá la re-ejecución manual de tareas que hayan fallado.
- Debe ejecutar primero todas las tareas de mayor prioridad:
    - Si una tarea de menor prioridad es agregada deberá esperar a que las de mayor prioridad se ejecuten.
    - Si una tarea de mayor prioridad es agregada debe pasar a un lugar mas alto en el ***queue*** para ser ejecutada apenas termine la tarea en ejecución.

b) Queue de Tareas

- Las tareas se almacenarán en una cola con prioridades.
- El sistema procesará las tareas de forma FIFO (First In, First Out) dentro de cada nivel de prioridad.

- Podrás implementar una lógica de procesamiento que espere por lo menos 30 segundos antes de iniciar la siguiente tarea.

c) Ejecución de Tareas

- Un "Worker" o "Job" en segundo plano gestionará la ejecución de las tareas desde la cola.
- Las tareas que se ejecutan pueden tener tiempos de ejecución variados, por lo que debe esperar hasta que la tarea falle o finalice para continuar con la siguiente.
- Para evitar el bloqueo de la interfaz mediante un Request interminable puede responder mediante un ActionResult que ha agregado una tarea a la cola para terminar su ejecución con un **StatusCode** de **200 OK**, y posteriormente las tareas se podrán visualizar en el panel de monitoreo.
- Una vez que una tarea ha sido iniciada esta no puede ser ni eliminada ni modificada.

d) Monitoreo y Notificación

- Los usuarios podrán ver un panel con el estado actual de las tareas en cola (pendiente, en ejecución, completada).
- Envío de notificaciones (como correos electrónicos o un detalle que explique por que falló) cuando las tareas finalicen o fallen.
- Logs detallados para monitorear el resultado de la ejecución de cada tarea y que guarden un historial de las tareas ejecutadas.

Especificaciones Técnicas

a) Tecnologías y Frameworks

- Framework: .NET Framework 4.8 (MVC)
- Base de datos: SQL Server para almacenar las tareas y sus estados.
- Entity Framework
- ASP.NET Razor

b) Automatización

- Debe utilizar código propio para la ejecución automatizada de las tareas.

Controladores (Controllers)

a) TaskController:

- Acciones para crear, editar, eliminar tareas.
- Listar tareas en función de su estado (pendiente, en ejecución, finalizada).

b) QueueController:

- Gestión del trabajo en cola.
- Ver el estado de la cola y el historial de ejecución de tareas.

d) Vistas (Views)

- Queue de Tareas:
  - Mostrar las tareas NO ejecutadas agrupadas por prioridad (y por fecha).
  - Botón de "Reintentar" para tareas fallidas.
- Crear, Editar, Eliminar Tareas
- Historial
  - Opción de ver detalles del log de cada tarea.
- Dashboard
  - Detalle con la cantidad de tareas ejecutadas por estado

4. Presentación y Demostración

La presentación y demostración debe realizarse en semana 14, según la asignación del profesor, deben presentarse en ambas semanas o de lo contrario no se aceptará la exposición.

Deberá durar 15-20 minutos en total. La exposición se hará según la asignación realizada en semana 13 en conjunto con el profesor.

Los estudiantes deberán presentar en forma oral, sin hacer uso de presentación:

1. Introducción:
   a. Presentar a todos los integrantes del grupo
   b. Presentar la temática del proyecto
2. Demostración del proyecto:
   a. Deberán presentar la demostración de la funcionalidad de cada uno de los elementos solicitados en el proyecto. El profesor puede solicitar realizar alguna funcionalidad específica con el fin de validar la completitud de los requerimientos antes diversos datos de entrada.
3. Retos y lecciones
   a. Comparta con la clase los retos y lecciones de realizar la visión de las funcionalidades de su proyecto en cuanto a decisiones de diseño, discusiones de implementación y apartado de seguridad, etc.

5. Adicionales

- No se solicita trabajo escrito.
- Si tiene un problema con un compañero de grupo, comuníquemelo lo antes posible.

fidélitas
UNIVERSIDAD

Ingeniería en Sistemas de Computación

Proyecto Final **Curso:** Programación Avanzada
Prof:
Código curso: SC-601

- La asistencia al día de la exposición es obligatoria. No se puede reponer la exposición, solo se puede justificar la ausencia.
- El estudiante deberá mostrar dominio completo de la aplicación desarrollada.
- No puede retirarse de la clase hasta que expongan todos los grupos.
- Debe respetar los temas desarrollados en las lecciones.
- Si se comprueba la intervención de personas ajenas al grupo en la programación del proyecto, éste será anulado y será causa para la pérdida del curso. Se aplicará la misma sanción en caso de que se compruebe que existe copia parcial o total de códigos de Internet.

«El pesimista ve dificultades en cada oportunidad. El optimista ve oportunidades en cada dificultad» —Winston Churchill