Universidad Autónoma de Yucatán
Facultad de Matemáticas
LIS
Asignatura: Construcción de Software
Proyecto Final

Instrucciones: Desarrolla el siguiente programa de software aplicando correctamente los estándares y prácticas de construcción de software abordados en la asignatura:

1. Requerimientos Funcionales Detallados

A. Gestión de Tareas

1. **CRUD de Tareas (primer commit a la rama development):**
    - **Crear tarea**:
        - Campos: id (auto-generado), título (requerido), descripción, fechaVencimiento (LocalDate), prioridad (ENUM: ALTA, MEDIA, BAJA), estado (ENUM: PENDIENTE, EN_PROGRESO, COMPLETADA).
        - Validación: Título no vacío, fecha futura o hoy.
    - **Listar tareas**:
        - Mostrar en formato tabla por consola (usar System.out o librería como Apache Commons Text para bordes).
        - Opciones de ordenamiento: por fechaVencimiento o prioridad.
    - **Actualizar tarea**:
        - Editar cualquier campo (excepto id).
    - **Eliminar tarea**:
        - Confirmación por consola (Sí/No).
2. **Búsqueda/Filtrado (segundo commit a la rama development):**
    - Filtrar por: estado, prioridad, fechaVencimiento.
    - Buscar por palabra clave en título o descripción (case-insensitive).

B. Persistencia de Datos:

3. **Backup en JSON (tercer commit a la rama development):**
    - Usar Jackson o Gson para almacenar las tareas.

C. Interfaz de Usuario (cuarto commit a la rama development)

4. **JavaFX/Swing:**
    - Pantalla simple con lista de tareas y botones para CRUD.

D. Pruebas Automatizadas

5. **Cobertura >80% con JUnit 5:**
    - Pruebas para:
        - Lógica de validación (ej.: fecha inválida).

- Persistencia (ej.: guardar/recuperar tarea).

C. Documentación

6. **README.md:**
    - Cómo compilar/ejecutar (usar Maven/Gradle).
    - Diagrama de clases UML básico.

---

2. Stack Tecnológico

- **Lenguaje:** Java 11+.
- **JSON:** Jackson.
- **Pruebas:** JUnit 5 + Mockito (para dependencias).
- **Control de Versiones:** Git + GitHub.
- **Build Tool:** Maven o Gradle.

Puntos a evaluar:

1. Modelado del diseño de la aplicación. (5 pts.)
    - Utilizar una herramienta para realizar el diagrama de clases de la aplicación.
    - Formato de entrega: Imagen, la cual será incluida como un apartado del documento integrador.

2. Checklist de las prácticas de construcción utilizadas (5 pts.).
    - Elaborar una lista de todas las prácticas de construcción de software abordadas en la asignatura, y posteriormente realizar un checklist de las prácticas aplicadas en el proyecto.
    - Formato de entrega: Archivo de texto, el cual deberá incluirse como un apartado del documento integrador.

3. Legibilidad y calidad del código construido (40 pts.).
    - Desarrollar la aplicación de software.
    - Formato de entrega: Se debe entregar el código de la aplicación.

4. Pruebas de Aceptación (30 pts.)
    - Se realizarán las pruebas funcionales de la aplicación con base en las especificaciones del proyecto.

5. Empleo de herramientas de construcción.
    a. Junit (10 pts.)
        - Utilizar la herramienta JUnit en la refactorización del código del proyecto, e incluir las evidencias correspondientes.
    b. Documentación de código (5 pts.)
    c. Herramienta de medición de la calidad del Código (5 pts.)

Consideraciones importantes:

1. Utilizar el paradigma de Programación Orientada a Objetos.
2. Utilizar el patrón de diseño MVC.
3. Programar en el lenguaje Java.

4. Utilizar el estándar de codificación de Java.
5. Aplicar los lineamientos y técnicas de construcción de software abordados en la asignatura.
6. Emplear la herramienta Git en el desarrollo de la aplicación utilizando GitFlow.

Formato de entrega final.- Un archivo comprimido que incluya lo siguiente:

1. Documento integrador.- Un archivo de texto con el siguiente contenido:
    a. Portada.
    b. Modelado del diseño de la aplicación.
    c. Checklist de las prácticas de construcción.
2. Proyecto en Java de la aplicación de software. Incluir las pruebas en JUnit.
3. Documentación del código.