Puntos de la Rúbrica Mejorados.

1. **Git:**
    a. **En el repositorio hay un proyecto de su IDE favorito, con los directorios bin, src, data y docs.**
    En docs se encuentra 1 documento con los requerimientos funcionales y en el mismo documento las imágenes del diseño de diagrama de clases en alta resolución (no borrosas).

    Se han puesto las imágenes del diseño junto en el mismo PDF en el que se encuentran los requerimientos.

    b. **El usuario del estudiante tiene al menos 10 commits con al menos 1 hora de diferencia entre ellos en el repositorio remoto (GitHub o GitLab) (1.5 si tiene 1 commit, 3.0 si tiene 2 commits, 5.0 si tiene los 3 commits).**

    Más de 280 commits implementados en el repositorio remoto en GitHub del programa con diferencias de 1+ hora.

2. **Requerimientos Funcionales:**
    a. **Especifica correctamente todos y cada uno los requerimientos funcionales de la solución a desarrollar.**
    NO utiliza tablas con campos de entrada, nombre, salida, ni similares para especificar cada requerimiento. Los requerimientos si tienen un código que los identifica RF1, RF2, etc, pero luego sigue, para cada uno, la redacción de al menos un párrafo describiendo el requerimiento. El requerimiento debe iniciar con un verbo en infinitivo indicando cuál es la funcionalidad que el sistema debe estar en la capacidad de hacer.

    Se especifican cada uno de los requerimientos funcionales del programa. No se utilizan tablas. Además, cada requerimiento tiene su respectiva codificación seguido del párrafo describiéndolo. Inicia con verbo en infinitivo

3. **Diagrama de Clases:**
    a. **El diagrama de clases del modelo respeta la sintaxis UML, las clases (sus atributos, constantes y métodos) se encuentran bien definidos, las relaciones establecidas son las apropiadas y tienen la dirección y forma correcta de acuerdo a su tipo (las relaciones de asociación tienen cardinalidad correcta y nombre). Define apropiadamente cada paquete.**

    Cada paquete se nombró adecuadamente y sus respectivas clases están correctamente definidas. Cada clase se relaciona con su correspondiente de acuerdo con el tipo de relación que tienen (asociación, dependencia o herencia). Finalmente, se respeta la cardinalidad y nombres de estas relaciones.

b. **El diagrama de clases incluye correctamente definidas las clases, métodos, atributos y constantes del modelo, satisfaciendo las necesidades planteadas en el enunciado.**

   El paquete “model” se ha completado, permitiendo así que se pueda visualizar las relaciones que existen entre las clases de este y, también, que se determinen los atributos y métodos de cada clase. Así mismo, siguen las necesidades planteadas en el enunciado.

c. **El diagrama de clases del modelo presenta un diseño apropiado que incluye herencia.**

   Dentro del paquete “model” existe una relación de herencia o generalización en la cual la clase “User” hereda de “Employee”.

d. **El diagrama de clases de la GUI presenta una adecuada distribución de responsabilidades y correcta notación.**

   Se usan el número de clases controladoras necesarias para dividir las responsabilidades de manera adecuada

4. **Implementación:**
   a. **El programa permite el registro y actualización de ingredientes, tipos, productos, empleados, clientes y usuarios.**

      Ahora se pueden gestionar los mencionados objetos en el modelo.

   b. **El programa permite el registro pedidos y posterior actualización de su estado.**

      Ahora se pueden registrar pedidos y, posteriormente, actualizar su estado.

   c. **El programa serializa y deserializa de toda la información del modelo en archivos de forma automática cuando se agrega nueva información y cuando el programa inicia, respectivamente.**

      El programa ahora está en la capacidad de guardar y cargar, a través de los archivos serializados, toda la información del programa de manera automática tras realizar cambios en el mismo

   d. **El programa exporta el archivo csv con las características especificadas en el enunciado.**

      El programa está en la capacidad de exportar los pedidos con el formato y características determinadas por el enunciado.

e. **El programa permite generar los reportes especificados en el enunciado: a) Empleados consolidando pedidos y, b) Productos consolidando número de veces que se pidió. Ambos en un rango de fechas indicadas por el usuario del programa.**

   El programa genera los reportes de empleados y productos con los requerimientos enunciados.

f. **El programa tiene una opción que hace búsqueda binaria de un cliente dado un nombre e indica el tiempo que tardó la búsqueda. Los clientes son insertados en ese orden siempre para mantener la lista ordenada.**

   Ahora el programa utiliza búsqueda binaria para buscar un cliente. Además, se muestra en pantalla el tiempo que tardó. Siempre mantiene ordenada la lista.

g. **El programa implementa al menos 2 algoritmos de ordenamiento de los 3 vistos en clase (burbuja, selección e inserción). Hace por lo menos 1 ordenamiento utilizando Comparable y 1 ordenamiento utilizando Comparator, utilizando en ambos casos el sort de Collections o de Arrays.**

   Se usan dos (2) algoritmos de ordenamiento: selección para ordenar pedidos por orden de fecha, e inserción para agregar ordenadamente a los clientes. Por otro lado, se usa Comparator para realizar la búsqueda binaria de cliente en la sección de pedidos.

h. **El programa permite importar datos de un archivo csv con información de 1) clientes, otro con información de 2) productos, otro con información de 3) pedidos. El proyecto tiene en el directorio data cada archivo csv con al menos 1000 datos c/u.**

   El programa ahora está en la capacidad de importar un archivo csv de clientes, productos y pedidos. Esto se prueba con los archivos de prueba.

i. **Correcta implementación de la GUI, respetando el diseño y la apropiada distribución de responsabilidades.**

   La GUI se ha implementado correctamente distribuyendo las responsabilidades en más clases controladoras.

j. **El programa funciona correctamente. NO TIENE WARNINGS. Se puede ejecutar y se pueden utilizar las funcionalidades tal como se describe en el enunciado**

El programa ahora si se ejecuta sin problemas y se pueden usar cada una de las funcionalidades como menciona el enunciado. No tiene ningún warning.

5. **Bonus:**
    a. **Utiliza correctamente Enum para collecciones de elementos fijos que no cambian (p.e. estado del pedido, estado de un objeto, pero no tamaño de un producto)**

       Se usa Enum para elementos fijos tales como el Estado de un Pedido.

    b. **Utiliza al menos un hilo correctamente para que la fecha y hora se actualice permanentemente. Si no se actualiza permentemente o no usa hilos, no se gana este bonus.**

       Se muestra correctamente la fecha y la hora actualizada. Además, se usa hilos para realizar esta tarea.