	**METODOLOGÍA DE LA PROGRAMACIÓN** **DEVS PROJECT** **EMC**	

1) Diseñar un programa modular para gestionar contactos. El programa debe permitir al usuario realizar las siguientes operaciones:

- **Agregar un contacto:** El usuario debe poder ingresar el nombre, número de teléfono y correo electrónico del contacto a agregar. Validar que el teléfono no se repita.

- *Ejemplo:*

  María, 984888999, maria@gmail.com

- **Antes de guardar un contacto:** debe validarse que tenga un formato válido. Esto significa que debe tener una "@" y la misma no puede estar ni al inicio ni al final de la cadena de caracteres. Además, los dominios comunes tampoco son válidos, su longitud no puede ser menor a 7 caracteres.
- **También debe validar que el nombre no posea caracteres numéricos ni especiales**; considere solo letras, espacios y **(* - . /)**

- **Buscar un contacto:** El usuario debe poder buscar un contacto por su nombre. Si el contacto existe, se debe imprimir por pantalla el número de teléfono y su correo. Si no existe, debe mostrarse un mensaje de que el contacto no se encontró. Para esto, se leerá el contenido del fichero en una lista.

1) Diseñar un programa modular para gestionar tareas. El programa debe permitir al usuario realizar las siguientes operaciones:

Agregar una tarea:
El usuario debe poder ingresar la fecha en formato **dd-mm-yyyy**, la hora en formato **hh:mm**, y la descripción de la tarea a agregar. También debe asignarse el estado de la tarea, cuyo valor por defecto será **"Registrada"**, y un número de tarea generado automáticamente de forma secuencial.

La información debe almacenarse en un archivo llamado **tareas.txt**. Cada tarea se debe guardar en una línea diferente, donde cada atributo estará separado por **punto y coma.**

Ejemplo del archivo:
1; 10-05-2024, 10:30; Enviar instrucciones a 3 clientes; Registrada
2; 10-05-2024, 12:00; Consulta médico; Registrada
3; 11-05-2024, 14:00; Reunión con coordinadores; Completada

Validaciones:
Antes de guardar la tarea, deben realizarse las siguientes validaciones:

- **Fecha y hora:** Asegurarse de que la fecha tenga un formato válido dd-mm-yyyy y que la hora esté en el formato hh:mm. Además, validar que el valor de la fecha y la hora se encuentre dentro de rangos posibles.
- **Descripción:** La descripción de la tarea no debe estar vacía.

Buscar una tarea:
El usuario debe poder buscar una tarea por su número de tarea (ID). Si la tarea existe, se debe imprimir por pantalla su descripción, fecha, hora y estado. Si no existe, debe mostrarse un mensaje de que la tarea no se encontró. Para realizar esta búsqueda, se leerá el contenido del archivo **tareas.txt** y se procesará en una lista.

Consideraciones:

- La solución entregada debe estar correctamente **modularizada,** siguiendo las características de **Clean Code.**

- El archivo **tareas.txt** no existe al inicio del programa y se debe crear al agregar la primera tarea.

2) Dada la siguiente lista de números:

6	**1**	**9**	**20**	**1**	**15**	**19**	**25**	**40**

Realice lo siguiente:

- Explique el funcionamiento del método **selección.**

- Ordene en forma manual la lista en forma ascendente, utilizando el método selección. Debe mostrar el detalle de los pasos que se realizan hasta que la lista quede ordenada.

MODELOS GENERADOS POR IA

1)Diseñar un programa modular para gestionar productos en un inventario. El programa debe permitir al usuario realizar las siguientes operaciones:

Agregar un producto:

- El usuario debe ingresar el nombre del producto, el precio y la cantidad en stock. Además, debe registrarse automáticamente un identificador único para cada producto, generado secuencialmente. El estado del producto por defecto será "Disponible".

- La información debe almacenarse en un archivo llamado productos.txt, donde cada producto se guardará en una línea, y cada atributo estará separado por comas.

Ejemplo del archivo:
1, Laptop, 1500.00, 5, Disponible
2, Teclado, 25.00, 12, Disponible

Validaciones:
Antes de guardar el producto, deben realizarse las siguientes

- **Nombre:** No debe estar vacío y debe contener solo letras, espacios y números.
- **Precio:** Debe ser un número decimal positivo.
- **Cantidad:** Debe ser un número entero no negativo.

Buscar un producto:
El usuario debe poder buscar un producto por su nombre. Si el producto existe, se deben mostrar sus detalles (precio, cantidad en stock, y estado). Si no existe, debe mostrarse un mensaje indicando que no se encontró.

Consideraciones:

- El archivo productos.txt no existe al inicio del programa.
- La solución debe estar correctamente modularizada y seguir principios de Clean Code.

2) Dada la siguiente lista de números:

5	2	54	21	50	1	15	37	27

Ordene en forma manual la lista en forma ascendente, utilizando el método de **burbuja**. Debe mostrar el detalle de los pasos que se realizan hasta que la lista quede ordenada

1)Diseñar un programa modular para gestionar productos en un inventario. El programa debe permitir al usuario realizar las siguientes operaciones:

Leer productos:

- El programa debe leer la lista de productos desde un archivo llamado productos.txt y almacenarlos en una lista. Cada línea del archivo contendrá la información del producto separada por comas.

Ejemplo del archivo:

ID, Nombre, Categoría, Marca, Año de lanzamiento, Precio, Estado
ABC123, Laptop, Laptop, MarcaX, 2021, 1500.00, D
DEF456, Smartphone, Smartphone, MarcaY, 2022, 800.00, A

Buscar productos:

- **Por nombre:** El usuario podrá buscar un producto por su nombre utilizando el método de búsqueda binaria. Si el producto es encontrado, se debe mostrar toda la información. Si no, se debe devolver un mensaje indicando que el producto no existe.
- **Por categoría:** El usuario podrá buscar productos por su categoría. La búsqueda debe soportar ocurrencias parciales, es decir, si se ingresa "Lap", debe devolver todos los productos de la categoría Laptop (sin distinguir entre mayúsculas y minúsculas).

Validaciones:

Antes de realizar las operaciones, deben realizarse las siguientes validaciones:

- **ID:** Debe ser un String de 6 caracteres sin espacios, formado por las primeras tres letras del nombre del producto y tres dígitos consecutivos.
- **Nombre:** Debe ser un String que describa el producto.
- **Categoría:** Debe ser una de las siguientes opciones: Laptop, Smartphone, Accesorios, Audio.
- **Marca:** Debe ser un String que indique el fabricante del producto.
- **Año de lanzamiento:** Debe ser un número en el rango [2010, 2024].
- **Precio:** Debe ser un número decimal positivo.
- **Estado:** Debe ser uno de los siguientes: D (Disponible), A (Agotado), P (En Pedido).

Consideraciones:

- El archivo productos.txt no existe al inicio del programa.
- La solución debe estar correctamente modularizada y seguir principios de Clean Code

2) Dada la siguiente lista de nombres:

Jorge	Ana	Bar	Ester	Alex	Pedro	Carlos	Gabriel	Pepe

Ordene en forma manual la lista en forma ascendente, utilizando el método de **Inserción.** Debe mostrar el detalle de los pasos que se realizan hasta que la lista quede ordenada.

NIVEL TRYHARD

1) Diseñar un programa modular para gestionar un catálogo de sitios web. El programa debe permitir al usuario realizar las siguientes operaciones:

Agregar sitio web:

El usuario podrá agregar un nuevo sitio web ingresando la siguiente información:

- **ID:** Debe ser un String de 6 caracteres sin espacios, formado por las primeras letras del nombre del sitio y un número consecutivo.
- **Nombre:** Debe ser un String que describa el sitio.
- **URL:** Debe ser un String que indique la dirección del sitio, siguiendo el formato correcto.
- **Categoría:** Debe ser uno de los siguientes: Buscador, Enciclopedia, Redes Sociales, E-commerce.
- **Año de creación:** Debe ser un número en el rango [1990, 2024].
- **Estado:** Debe ser uno de los siguientes: (Disponible), (Cerrado).

Leer sitios web:

- El programa debe leer la lista de sitios web desde un archivo llamado sitios.txt y almacenarlos en una lista. Cada línea del archivo contendrá la información del sitio web separada por guiones (-).

Ejemplo del archivo:

ID - Nombre - URL - Categoría - Año de creación - Estado
S001 - Google - https://www.google.com - Buscador - 1998 - Disponible
S002 - Wikipedia - https://www.wikipedia.org - Enciclopedia - 2001 - Cerrado

Buscar sitios web:

- **Por nombre:** El usuario podrá buscar un sitio web por su nombre utilizando el método de búsqueda binaria. Si el sitio es encontrado, se debe mostrar toda la información. Si no, se debe devolver un mensaje indicando que el sitio no existe.
- **Por categoría:** El usuario podrá buscar sitios web por su categoría. La búsqueda debe soportar ocurrencias parciales, es decir, si se ingresa "Bus", debe devolver todos los sitios de la categoría Buscador (sin distinguir entre mayúsculas y minúsculas).

Listar sitios web:

- **Por año de creación:** El usuario podrá listar todos los sitios web creados en un año específico. El listado debe estar ordenado de menor a mayor estado (disponible primero).
- **Por rango de estado:** El usuario podrá listar los sitios web cuyo estado se encuentre dentro de un rango definido por el usuario (Disponible y Cerrado).

Modificación de sitios web:

- **Modificar estado:** El programa debe permitir modificar el estado de los sitios web cuyo estado es "A" (Agotado), cambiándolo a "D" (Disponible).
- **Modificar información del sitio:** El usuario podrá modificar la información de un sitio existente (nombre, URL, categoría, año de creación o estado) ingresando su ID.

Eliminación de sitios web:

- **Eliminar sitio web:** El programa debe permitir al usuario eliminar un sitio web de la lista ingresando su ID. Si el sitio es encontrado y eliminado, se debe mostrar un mensaje de confirmación. Si no, se debe informar que el sitio no existe.

Validaciones:

Antes de realizar las operaciones, deben realizarse las siguientes validaciones:

- **ID:** Debe ser un String de 6 caracteres sin espacios, formado por las primeras letras del nombre del sitio y un número consecutivo.
- **Nombre:** Debe ser un String que describa el sitio.
- **URL:** Debe ser un String que indique la dirección del sitio, siguiendo el formato correcto.
- **Categoría:** Debe ser uno de los siguientes: Buscador, Enciclopedia, Redes Sociales, E-commerce.
- **Año de creación:** Debe ser un número en el rango [1990, 2024].
- **Estado:** Debe ser uno de los siguientes: D (Disponible), C (Cerrado).

Consideraciones:

- El archivo sitios.txt no existe al inicio del programa.
- La solución debe estar correctamente modularizada y seguir principios de Clean Code.

2) Dada las siguientes listas:

5	2	54	21	50	1	15	37	27

Jorge	Ana	Bar	Ester	Alex	Pedro	Carlos	Gabriel	Pepe

Ordene en forma manual la lista en forma ascendente y descendente, utilizando **todos los métodos de ordenación.** Debe mostrar el detalle de los pasos que se realizan hasta que la lista quede ordenada.