Práctica de Clean Architecture.

1 Estructura base para el proyecto:

a) Cree un nuevo proyecto de tipo Web API con nombre de proyecto “Presentation” y nombre de solución “MyFirstWebApi”.

b) Agregue tres proyectos más a la solución, que sean del tipo Biblioteca de Clases, y llámelos “Application”, “Domain”, e “Infrastructure”.

c) Referencie los proyectos entre sí respetando la jerarquía de capas propuesta por Clean Architecture.

2. La capa Domain debe contener lo siguiente:

a) una carpeta llamada “Entities”, con una clase llamada “User” que tenga las siguientes propiedades:
- un Id de tipo entero,
- un Name de tipo string.
- un LastName de tipo string.
- un Email de tipo string.
- un Password de tipo string.
- un CreationDate de tipo DateTime.

b) Una carpeta llamada “Interfaces” con una interfaz llamada “IUserRepository” que contenga la firma del siguiente método:
- List<User> GetAll();

3. La capa Infrastructure debe contener lo siguiente:
- Una clase llamada “UserRepository” que implemente la interfaz IUserRepository que se encuentra en la capa Domain (debe implementar el metodo de dicha interfaz), y que además contenga una propiedad estática llamada “users” del tipo List<User>. (Debe contener dos usuarios hardcodeados).

4. La capa Application debe contener lo siguiente:
a) Una carpeta llamada Models que contenga un UserDTO con las siguientes propiedades:
- un Id de tipo entero,
- un FullName de tipo string.
- un Email de tipo string.
- un CreationDate de tipo DateTime.

b) Una carpeta llamada Interfaces con una interfaz llamada “IUserService” que contenga la siguiente firma de método:
- List<User> GetAll();

c) Una carpeta llamada Services que contenga un clase llamada UserService, que a su vez implemente la interfaz IUserService y que por supuesto implemente el método de dicha interfaz.

5. La capa Presentation debe contener en su carpeta Controllers el controlador UsersController, y dentro deberá contener un endpoint de tipo

GET también llamado Get(), que retorne la lista completa de usuarios que existen.

6. Cada clase debe recibir su dependencia inyectada por constructor, y cada dependencia debe ser del tipo de una interfaz para logar un mayor desacople.
   (Asegurarse de configurar los servicios necesarios para IUserService y IUserRepository en el contenedor de servicios de la clase Program.cs. Para ambos se debe indicar el tiempo de vida Scoped).

   (Nota: El flujo de ejecución debe quedar así:
   UserController => UserService => UserRepository)