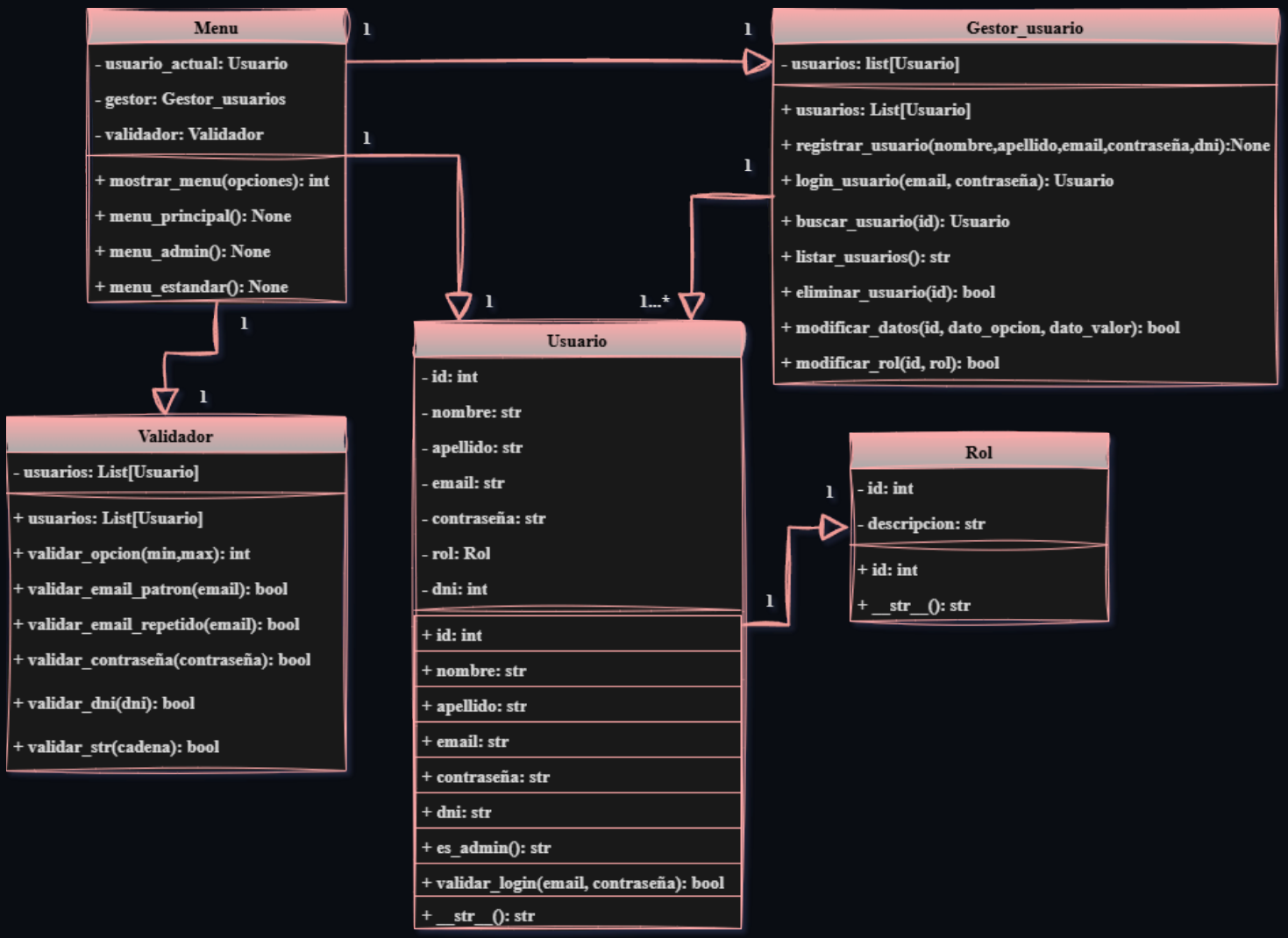

Objetivo del diagrama de clases

La creación de este diagrama tiene la intención de guiar y estructurar el modelo de POO que será utilizado para crear un sistema de consola en Python. En él se presentan diferentes clases con sus respectivos atributos, métodos y relaciones. El sistema de consola está basado en un programa para la gestión de usuarios de la tienda de ropa Merced basados en su rol.

Clases y sus responsabilidades

Menu

- Responsabilidad: actúa como interfaz entre el usuario y el sistema. Controla el flujo del programa según el rol del usuario actual.
- Atributos:

    - usuario_actual: instancia de Usuario que está actualmente logueado.
    - gestor: instancia de Gestor_usuario, encargada de la lógica de datos.
    - validador: instancia de Validador, utilizada para validar los input del usuario.
- Métodos:
    - mostrar_menu(opciones): muestra dinámicamente un menú según el rol.
    - menu_principal(): lógica principal del sistema.
    - menu_admin(): opciones específicas para usuarios administradores.
    - menu_estandar(): opciones específicas para usuarios estándar.

Usuario

- Responsabilidad: representa a cada persona registrada en el sistema
- Atributos privados:
    - id: identificador único.
    - nombre: dato personal del usuario.
    - apellido: dato personal del usuario.
    - email: dato de acceso del usuario.
    - contraseña: dato de acceso del usuario, deberá tener un mínimo de 6 caracteres, 1 letra y 1 número.
    - dni: dato personal del usuario.
    - rol: Rol
- Métodos:
    - es_admin(): verifica si el usuario tiene rol de administrador.
    - validar_login(email, contraseña): comprueba si las credenciales coinciden con alguna registrada.
    - __str__(): representación del usuario como texto.

Rol

- Responsabilidad: define el rol actual del usuario (admin o estándar) y por ende cuáles serán sus opciones de acceso.
- Atributos privados:
    - id: valor numerico que indica si el usuario es admin o estandar
    - descripcion: basado en el valor del id sera estandar o admin
- Métodos:
    - id:
    - __str__(): representación del rol como texto.

Gestor_usuario

- Responsabilidad: controla la gestión de datos de usuarios. Permite crear, modificar, buscar y eliminar (CRUD).
- Atributos:
    - usuarios: lista que contiene todas las instancias de Usuario.
- Métodos:
    - registrar_usuario(nombre,apellido,email,contraseña,dni): recibe todos los datos necesarios para crear un usuario con su respectiva clase.
    - login_usuario(email, contraseña): comprueba si las credenciales existen y son correctas, devuelve el usuario.
    - buscar_usuario(id): devuelve datos de un usuario específico basado en su id.

    - listar_usuarios(): devuelve un listado de todos los usuarios registrados.
    - eliminar_usuario(id): elimina a un usuario por su id.
    - modificar_datos(id, dato_opcion, dato_valor): permite actualizar algún dato del alumno a la vez (nombre, apellido, contraseña), recibe su id para localizar el cambio, el numero de la opción elegida que son equivalentes a cada dato que se puede modificar y el nuevo valor del dato.
    - modificar_rol(id, rol): cambia el rol de un usuario.

Validador

- Responsabilidad: realiza validaciones de input para verificar que se reciben datos con el formato esperado y si no están ya localizados como instancias en Usuario.
- Atributos:
    - usuarios: lista de objetos Usuario (para validar repetidos)
- Métodos:
    - validar_opcion(min, max): verifica que la opción ingresada esté dentro del rango válido.
    - validar_email_patron(email): comprueba si el formato del correo es válido.
    - validar_email_repetido(email): comprueba si el correo ya se encuentra en uso.
    - validar_contraseña(contraseña): válida que la contraseña tenga un mínimo de 6 caracteres, 1 letra y 1 número.
    - validar_dni(dni): verifica el formato y longitud del DNI, así mismo comprueba que no esté en uso.
    - validar_str(cadena): asegura que el campo no esté vacío ni contenga caracteres inválidos.

Relaciones entre las clases

- Menu a Usuario: es una relación 1 a 1 donde el menú mantiene el usuario activo.
- Usuario a Rol: es una relación 1 a 1 donde el usuario solo posee un tipo de rol.
- Menu a Gestor_usuario: es una relación 1 a 1 donde el menú utiliza los métodos del Gestor.
- Menu a Validador: es una relación 1 a 1 donde el menú utiliza los métodos del validador a la hora de modificar y crear los datos.
- Gestor_usuario a Usuario: es una relación 1 a 1 o muchos donde el Gestor gestiona todos los usuarios registrados en el sistema.

Observaciones finales

- No se crearon las clases Usuario_admin ni Usuario_estándar con una relación de herencia a la clase Usuario debido a: no presentan diferencias estructurales (comparten métodos y atributos), la implementación del atributo

rol que evitó la extensión innecesaria de código ya que permite distinguir y limitar el acceso de los usuarios según sus roles.

- Se respetan los principios de **encapsulamiento**, usando propiedades (`@property`) para acceder a atributos privados.
- El sistema utiliza setters para permitir la modificación controlada de atributos privados. Esto permite aplicar validaciones o lógica adicional antes de asignar un nuevo valor, respetando el principio de encapsulamiento.
- La clase Usuario actúa como modelo de datos, mientras que Gestor_usuario es el controlador lógico.
- Se incorporó la clase rol para separar responsabilidades, evitar el uso de strings directos y facilitar validaciones futuras.
- Menu representa la capa de interacción con el usuario y los métodos posibles diferirá según el rol del usuario actual.
- Las clases están conectadas por composición (uno contiene instancias de otro).