**🔑 Authentication (Autenticación)**

* **Qué es:** Es el proceso de **verificar la identidad de un usuario o sistema**.
* **Pregunta que responde:** *¿Quién sos?*
* **Ejemplo:** Ingresar tu usuario y contraseña, usar tu huella digital, o un código enviado por SMS.
* **Resultado:** El sistema sabe *qué persona/entidad está intentando entrar*.

**🛂 Authorization (Autorización)**

* **Qué es:** Es el proceso de **verificar qué acciones o recursos tiene permitido hacer un usuario autenticado**.
* **Pregunta que responde:** *¿Qué podés hacer?*
* **Ejemplo:** Un usuario “entrenador” puede ver rutinas, pero **no puede eliminar usuarios**; en cambio, un “administrador” sí tiene ese permiso.
* **Resultado:** El sistema limita o habilita funciones según el rol o permisos.

👉 En resumen:

* **Authentication = identidad**
* **Authorization = permisos**

**📌 Resumen Registro y Login con Hash + JWT**

**🔹 Registro (Register)**

* El usuario manda username + password al endpoint de registro.
* **No se guarda la contraseña en texto plano** → se convierte en un **hash** usando PasswordHasher.
* El hash se guarda en la base de datos.
* Así, aunque roben la BD, **no se puede recuperar la contraseña original**.

👉 Hash = **proteger contraseñas** en la BD.

**🔹 Login (Inicio de sesión)**

* El usuario manda username + password al endpoint de login.
* El sistema busca el usuario en la BD por username.
* Se compara el **password ingresado** con el **hash guardado** usando PasswordHasher.VerifyHashedPassword.
* Si coincide ✅ → el usuario es válido.
* Entonces se genera un **JWT** con:
  + **Claims** (datos básicos como IdUsuario, Username, Rol).
  + **Issuer/Audience** (quién lo emite y quién lo consume).
  + **Expiración** (ej: 1 hora).
  + **Firma** con la clave secreta (AppSettings:Token).
* El JWT se devuelve al cliente (frontend/Postman).
* En cada request protegida, el cliente manda el token en el header:
* Authorization: Bearer <token>
* El middleware valida el token (firma, expiración, claims) antes de entrar al controlador.

👉 JWT = **credencial digital** para identificar y autorizar al usuario en cada request.

**🔹 Diferencia clave**

* **Hash** → solo para la contraseña (seguridad en la BD).
* **JWT** → para autenticar/autorizAR después del login (seguridad en la API).

**🔹 Esquema paso a paso**

1. **Register** → password → hash → se guarda en BD.
2. **Login** → password ingresado → comparar con hash en BD.
3. Si ok → generar JWT firmado con clave secreta.
4. Cliente usa JWT en cada request (Authorization: Bearer).
5. Middleware valida JWT antes de acceder a los endpoints.

✅ Con esto en tu apunte ya tenés la idea clara:

* **Hash = proteger password**.
* **JWT = carnet digital firmado para autenticación/autorizar acceso**.