Perfecto 🙌. A partir de la transcripción que me compartiste armé un **apunte teórico claro y estructurado** sobre **JWT (JSON Web Token) y su uso en backends seguros**, sin incluir detalles de código, sino centrándome en los **conceptos explicados en el video**.

**📌 Apunte Teórico – JWT para Programar Backend Seguro**

**1. ¿Qué es un JWT?**

* **JWT (JSON Web Token)** es un estándar de autenticación que permite transmitir información de forma **segura y compacta** entre dos partes.
* Se utiliza principalmente para:
  + **Identificar quién eres** (autenticación).
  + **Definir qué puedes hacer** (autorización).
* La información está **codificada** (no encriptada) y firmada con un algoritmo criptográfico.

**2. Estructura de un JWT**

Un token JWT se compone de **tres partes** separadas por puntos (.):

1. **Header (Encabezado)**
   * Indica el algoritmo de firma (ej. HS256).
   * Ejemplo: { "alg": "HS256", "typ": "JWT" }.
2. **Payload (Cuerpo o Claims)**
   * Contiene la información del usuario y sus privilegios.
   * Puede incluir datos como:
     + sub (ID de usuario).
     + name (nombre del usuario).
     + roles o scopes (permisos o alcance).
     + exp (fecha de expiración).
3. **Signature (Firma)**
   * Se genera con el **header + payload + clave secreta**.
   * Garantiza que el token no fue alterado.

**3. Diferencia entre Autenticación y Autorización**

* **Autenticación** → Confirmar que el usuario es quien dice ser (login con usuario y contraseña).
* **Autorización** → Determinar qué recursos o endpoints puede acceder según sus permisos.
* El **JWT sirve para ambas cosas**: autentica al usuario y guarda en sus *claims* los permisos necesarios.

**4. Funcionamiento Básico de JWT**

1. El usuario envía credenciales (ej. usuario y contraseña).
2. El servidor las valida y, si son correctas, genera un **JWT firmado**.
3. El cliente almacena el token (normalmente en localStorage o en memoria).
4. En cada petición al backend, el cliente envía el token en el **Header HTTP**:
5. Authorization: Bearer <token>
6. El servidor valida el token:
   * Revisa la firma.
   * Verifica la fecha de expiración.
   * Comprueba los permisos (claims).
7. Si es válido → el backend responde con los datos solicitados.  
   Si es inválido → devuelve error (401 Unauthorized o 403 Forbidden).

**5. Seguridad en JWT**

* El **secreto (clave de firma)** debe ser largo y difícil de adivinar.
* Los tokens deben tener una **fecha de expiración** (exp) para evitar uso indefinido.
* Nunca almacenar información sensible (ej. contraseñas) en el payload, ya que puede ser leído aunque esté firmado.
* Se recomienda usar HTTPS para proteger el envío de tokens.

**6. Claims y Scopes**

* **Claims**: piezas de información dentro del payload. Ejemplo:
  + name: "Pato Donald",
  + email: "pato@ejemplo.com".
* **Scopes o Roles**: definen el **alcance** de acceso. Ejemplo:
  + scope: "admin" → acceso completo.
  + scope: "user" → acceso limitado.
* Sirven para restringir qué usuarios pueden entrar a qué endpoints.

**7. Validación en el Servidor**

Cuando el backend recibe un JWT debe:

1. Confirmar que la firma coincide con el secreto configurado.
2. Verificar la expiración (exp).
3. Validar que los **claims** cumplen con lo requerido para ese endpoint.

Ejemplo:  
Un endpoint protegido puede requerir que el token tenga scope = "admin". Si no lo tiene, aunque el usuario esté autenticado, recibirá **403 Forbidden**.

**8. Uso Práctico en un Backend**

* Se integra con middlewares de autenticación (ej. en .NET, Express, Django).
* Se pueden crear endpoints como /auth/login que devuelvan un JWT al usuario autenticado.
* Otros endpoints (ej. /api/protected) solo aceptan peticiones con un JWT válido.

**9. Ventajas de JWT**

* **Autocontenido**: toda la información de sesión viaja en el token.
* **Escalable**: no se requiere mantener estado en el servidor (no depende de sesiones en memoria).
* **Flexible**: puede usarse en APIs REST, microservicios, aplicaciones móviles y web.

**10. Limitaciones**

* Si un token se filtra, puede usarse hasta que expire (no se invalida fácilmente).
* Requiere buena gestión de **renovación de tokens** (refresh tokens).
* No reemplaza el uso de HTTPS ni otras medidas de seguridad.

✅ Con esto ya tenés una visión clara de lo que es JWT, cómo funciona y para qué se usa en un backend seguro.

¿Querés que te arme este apunte en **formato Word** (como los que ya venís armando para tu curso) o preferís que quede en **Markdown** para tu apunte resumido?