🔐 JSON Web Token (JWT)

JWT is an open standard (RFC 7519) that defines a compact, self-contained way to securely transmit information between parties as a JSON object.

It consists of three parts:

- **Header** – contains the token type and signing algorithm.
- **Payload** – contains claims (user data such as ID, roles, permissions).
- **Signature** – ensures the token hasn't been tampered with.

✅ Common Use Cases

- **Authentication** – after login, the server issues a JWT; the client sends it with each request.
- **Authorization** – JWT includes roles/permissions to control resource access.
- **Secure data exchange** – trusted communication between APIs or microservices.

⚖️ Pros & Cons

Pros

- **Stateless** – the server does not store user sessions; each request is verified using the JWT itself → makes scaling easier.
- **Secure** – digitally signed (HMAC, RSA, etc.).
- **Portable** – works across different platforms and services.

Cons

- **Token revocation** – hard to invalidate before expiry if a token is leaked.
  - Mitigation: short token lifetimes + Refresh Tokens or server-side blacklist.
- **Bigger payloads** than simple session cookies.

🚀 Using JWT in Spring Boot

- Integrated with **Spring Security** to secure REST APIs.
- A custom **JWT filter** validates the token from the `Authorization` header on every request.
- On successful login, the server generates and signs a JWT, then the client uses it for subsequent requests.

🇸🇦 In Saudi Companies

- Used in telecom APIs (e.g., **Mobily APIs**) for partner integrations.
- Adopted in **National Single Sign-On (Nafath)** and many **digital wallets / fintech apps**.
- Common in mobile apps and government e-services for secure API authentication.