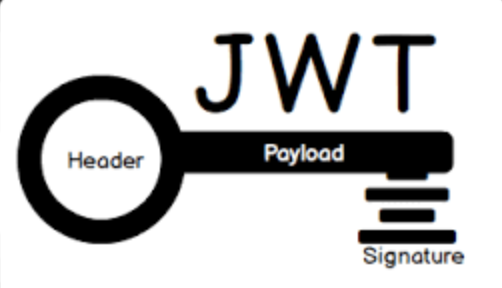
JSON

JSON Web Token is an open standard for securely transferring data within parties using a JSON object. JWT is used for stateless authentication mechanisms for users and providers, this means maintaining sessions on the client-side instead of storing sessions on the server.



Working Principle

**header** - encodeded with base64url

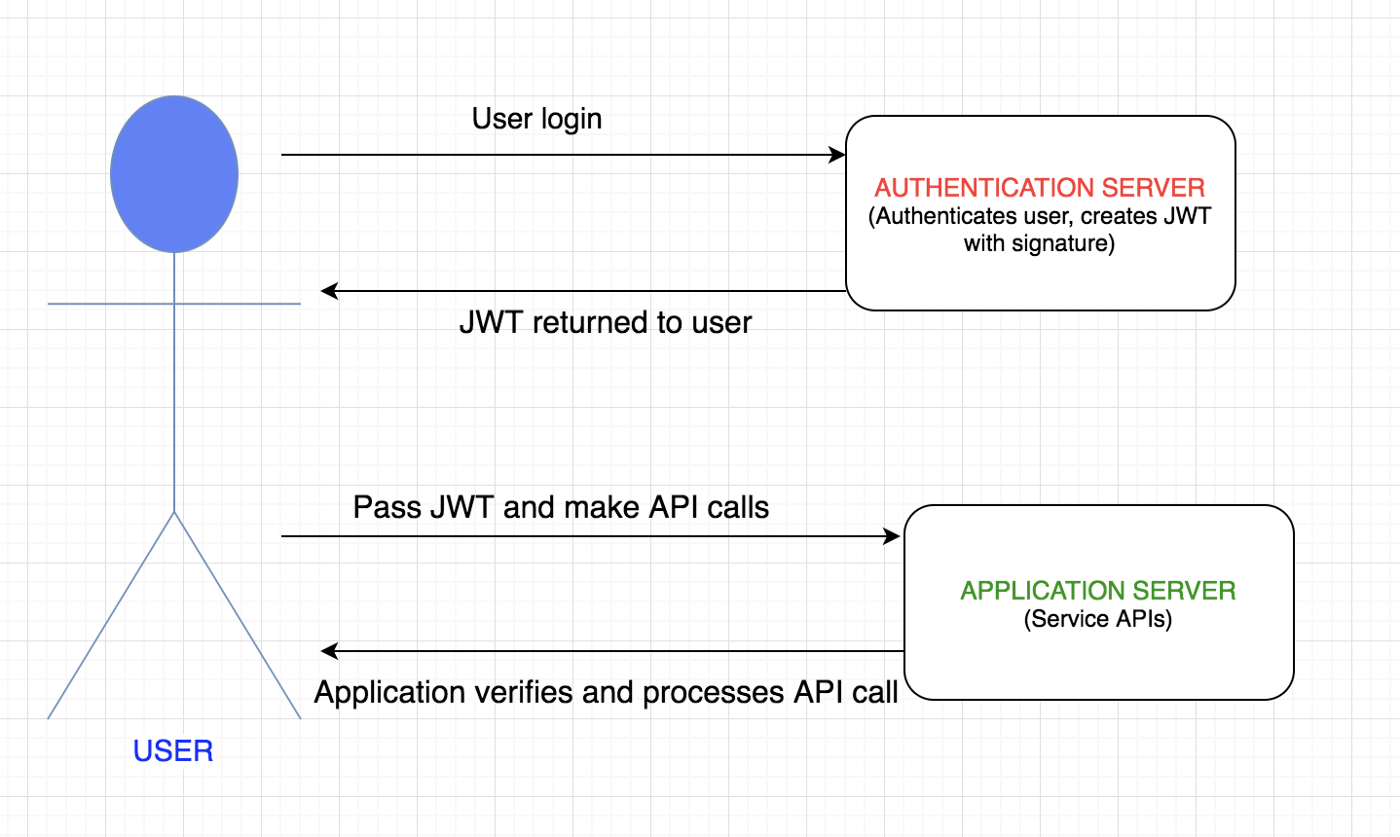
**payload** - in payload where we place the info (for ex userID) -encodeded with base64url

**signature** -used to verify that the sender of the JWT is who it says it is and to ensure that the message wasn't changed along the way

\* The tokens are digitally signed by the server’s private key. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair (with RSA or ECDSA). Do note that with signed tokens.

\* A user logs in by placing a request to the authentication server with their credentials from the client side. The server authenticates their request and also creates a JWT token that is signed and contains all necessary information for the user to access resources. With JWT, the user can then safely communicate with the application.

\* Now, when the user makes API calls to the application, the user passes the JWT along with the API call. In this setup, the application server is configured to verify that the incoming JWT is created by the authentication server. Thus, trust is established between the client and the server.



EXAMPLE:-

const login = async function (req, res) {

const emailId = req.body.emailId;

const password = req.body.password;

const userData = await userModules.findOne({

emailId: emailId,

password: password,

});

if(!userData){

res.send({ status: false, message:"invalied emailId or password"});

}

const token = await jwt.sign(

{ token: userData.\_id.toString() },

"jsonwebtoken"

);

req.headers.token = token;

// console.log({ token: token });

res.send({ status: true, msg:token});

};

const userId=async function(req, res){

const token= req.headers["x-auth-token"];

if(!token){ res.send({status:false,msg:"token must be persent"})};

const decodedToken= await jwt.verify(token,'jsonwebtoken');

if(!decodedToken){ res.send({status:false,msg:"token is not valid"})};

const userid=req.params.userId;

const userData=await userModules.findById(userid);

if(!userData){ res.send({status:false,msg:"no such user exists"})};

return res.send({status:true,user:userData})

}

Requirement Need:

1. To reduce repeated database queries because JWTs are self-contained, that is, they have all the necessary information to allow an authenticated user to access all server resources and routes.
2. To pass data in HTML and HTTP environments because of its small size. JSON is less verbose than XML so its encoded size is also smaller. It can be sent through a URL, POST parameter, or inside an HTTP header.
3. To authenticate a user via single sign-on. This is the most common use case of JWT and can be easily used among systems of different domains.
4. To exchange information as JWT is a good way to securely transmit information between parties because it can be signed by the sender. It also allows checking if the content has been tampered with.
5. Additionally, JSON parsers are common in most programming languages, because they map directly to objects as compared to XML that doesn’t have a natural document-to-object mapping.