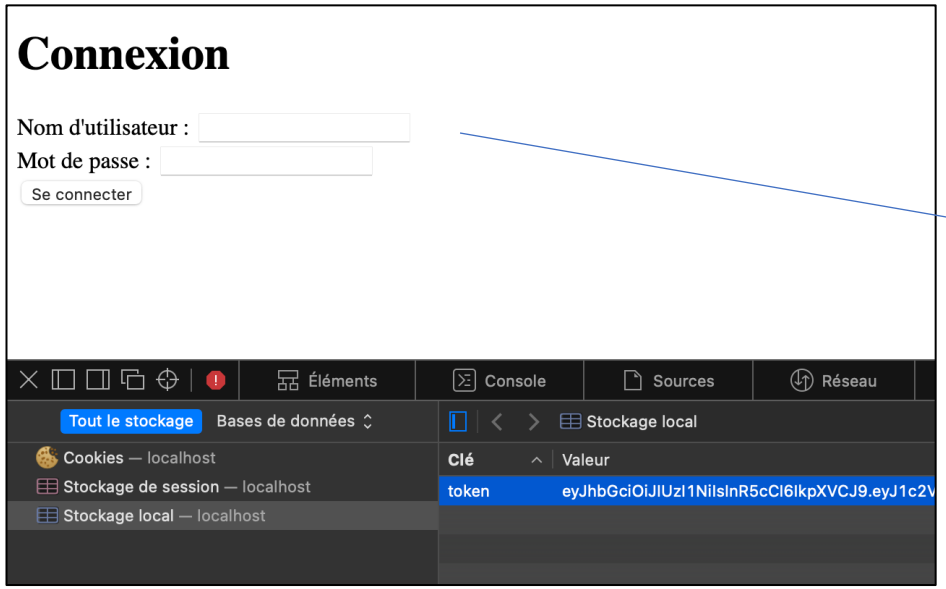
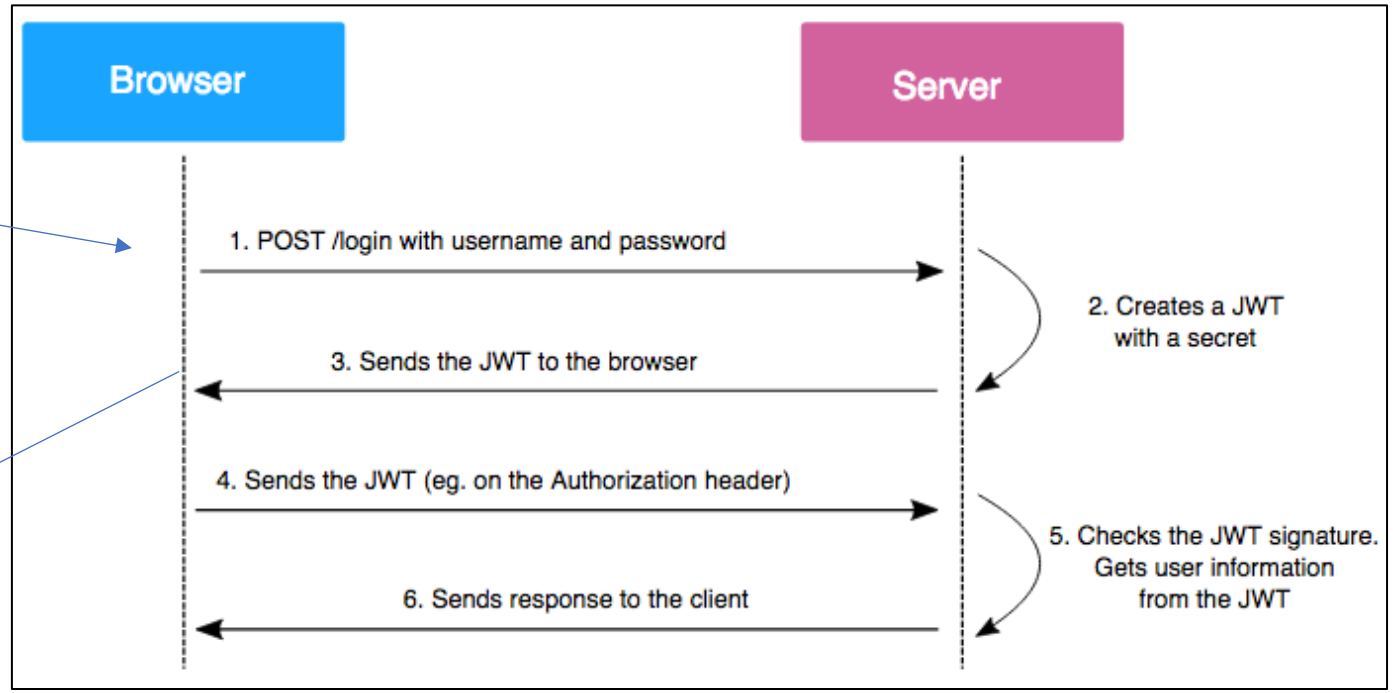
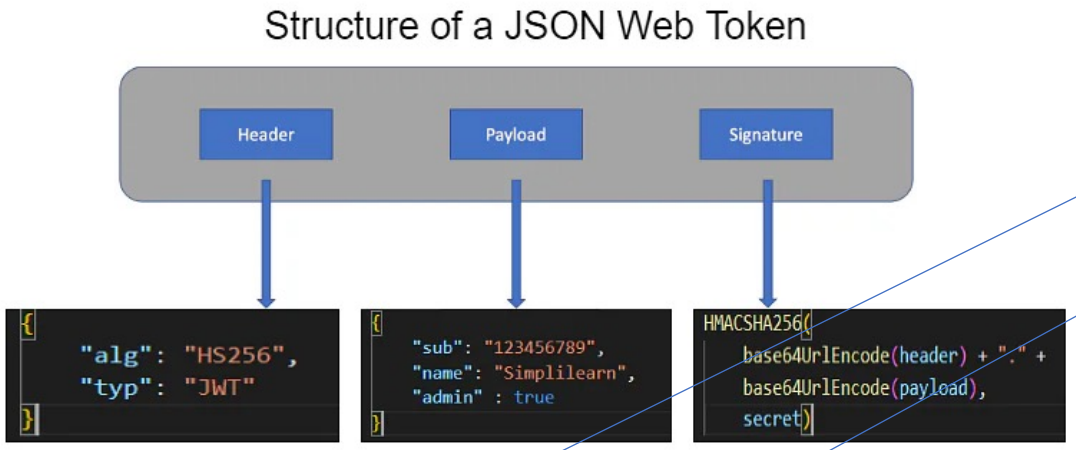
AUTHENTIFICATION PAR JETON JWT

Un token JWT (JSON Web Token) est un jeton d'authentification auto-suffisant qui est utilisé pour sécuriser les communications entre deux parties, telles qu'un serveur et un client. Il est souvent utilisé pour implémenter une authentification basée sur les tokens, où le serveur émet un jeton JWT à un client après que celui-ci s'est authentifié avec succès.








```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J1c2VybmFtZSI6InVzZXIiLCJpYXQiOjE2Nzc2O
DY5ODEsImV4cCI6MTY3NzY5MDU4MX0.4sFo0cae
HKM2wtdIQsrt9qQY027051CtG1YewPzfN-I
```

Un token JWT est constitué de trois parties, chacune encodée en base64 et séparée par un point. Les trois parties sont les suivantes :

- Header : Le header contient les informations sur le type de jeton et l'algorithme de signature utilisé pour signer le jeton. Il s'agit généralement d'un objet JSON qui contient deux clés : "typ" pour le type de jeton et "alg" pour l'algorithme de signature.
- Payload : Le payload contient les informations liées à l'utilisateur ou à la session. Ces informations sont encodées en base64 et décodées côté serveur pour vérifier l'authenticité du jeton et accorder l'accès à des ressources protégées. Le payload peut contenir des informations comme l'identifiant de l'utilisateur, la date d'expiration du jeton, etc
- Signature : La signature est générée à partir de la combinaison du header et du payload, et est utilisée pour vérifier l'authenticité du jeton. La signature est calculée en utilisant une clé secrète connue seulement du serveur, et l'algorithme de signature spécifié dans le header. La signature est encodée en base64 et ajoutée au jeton.

JETONS JWT AVEC NODEJS

Les tokens JWT sont un moyen courant d'authentifier les utilisateurs dans les applications Web modernes. Ils sont souvent utilisés pour la création de sessions d'utilisateur, l'accès à des API sécurisées et d'autres scénarios où une identification est nécessaire. Les tokens JWT sont généralement créés par le serveur lorsqu'un utilisateur s'authentifie et envoyés au client qui les stocke localement pour les utiliser dans des demandes futures.

```
npm install jsonwebtoken
```

```
const jwt = require('jsonwebtoken');
const secret = 'votre_secret';

const token = jwt.sign({ email }, secret, { expiresIn: '1h' });
```

Nous utilisons la méthode jwt.sign() pour créer un token JWT. Nous passons un objet qui contient l'identifiant de l'utilisateur, une clé secrète pour signer le token et une option expiresIn qui définit la durée de validité du token.

Ensuite, vous devez vérifier si un token JWT est valide lorsque vous recevez une demande de l'utilisateur. Voici un exemple de code pour vérifier un token JWT :

```
app.post('/verify', (req, res) => {
  const token = req.headers.authorization.split(' ')[1];

  // Vérifier la validité du token
  try {
    jwt.verify(token, secretKey);
    res.sendStatus(200);
  } catch (err) {
    res.sendStatus(401);
  }
});
```

Le token transmis par le client :

```
Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InVzZXIiLCJpYXQiOj
E2Nzc2Nzk1ODAsImV4cCI6MTY3NzY3OTc2MH0.UACc36m9GzP2KmFwcs78xFiqFaH45DR3rOh2eMWFrj0
```

- Tout d'abord, cette méthode POST est associée à l'URL /verify. Lorsqu'un client envoie une demande POST à cette URL, cette méthode est appelée.

- La méthode utilise la propriété headers de l'objet req pour récupérer la valeur de l'en-tête Authorization. Cette propriété est un objet qui contient les en-têtes de la demande HTTP. L'en-tête Authorization contient le token JWT, qui est nécessaire pour la vérification de l'authentification.

- La valeur de l'en-tête Authorization est une chaîne de caractères qui commence par le mot "Bearer", suivi d'un espace, puis du token JWT. La méthode split() est utilisée pour diviser cette chaîne de caractères en deux parties en utilisant l'espace comme séparateur. La méthode renvoie un tableau contenant les deux parties. La deuxième partie, qui contient le token JWT, est stockée dans la variable token.

- La méthode verify() de la bibliothèque jsonwebtoken est utilisée pour vérifier la validité du token JWT. Si le token est valide, cette méthode ne lève aucune exception et la méthode renvoie une réponse HTTP avec le code de statut 200 OK. Sinon, la méthode verify() lève une exception, qui est capturée par la clause catch de l'instruction try-catch. Dans ce cas, la méthode renvoie une réponse HTTP avec le code de statut 401 Unauthorized.