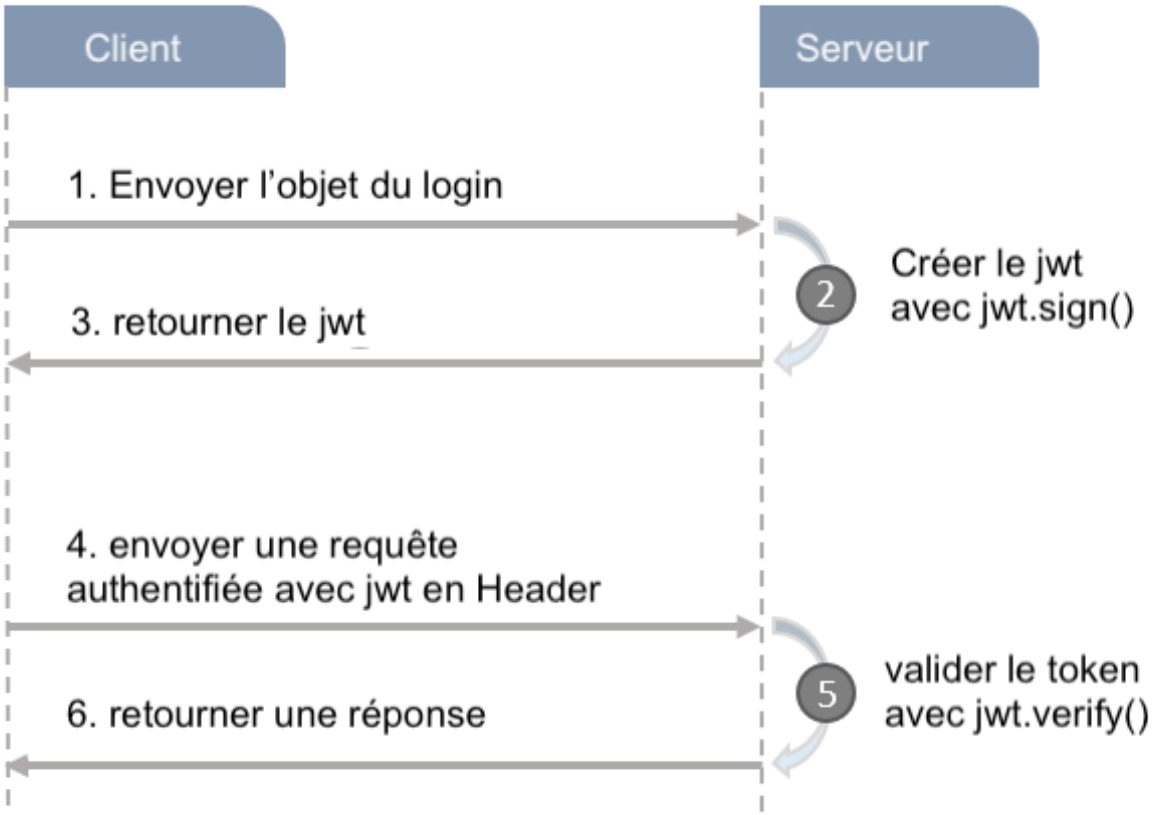
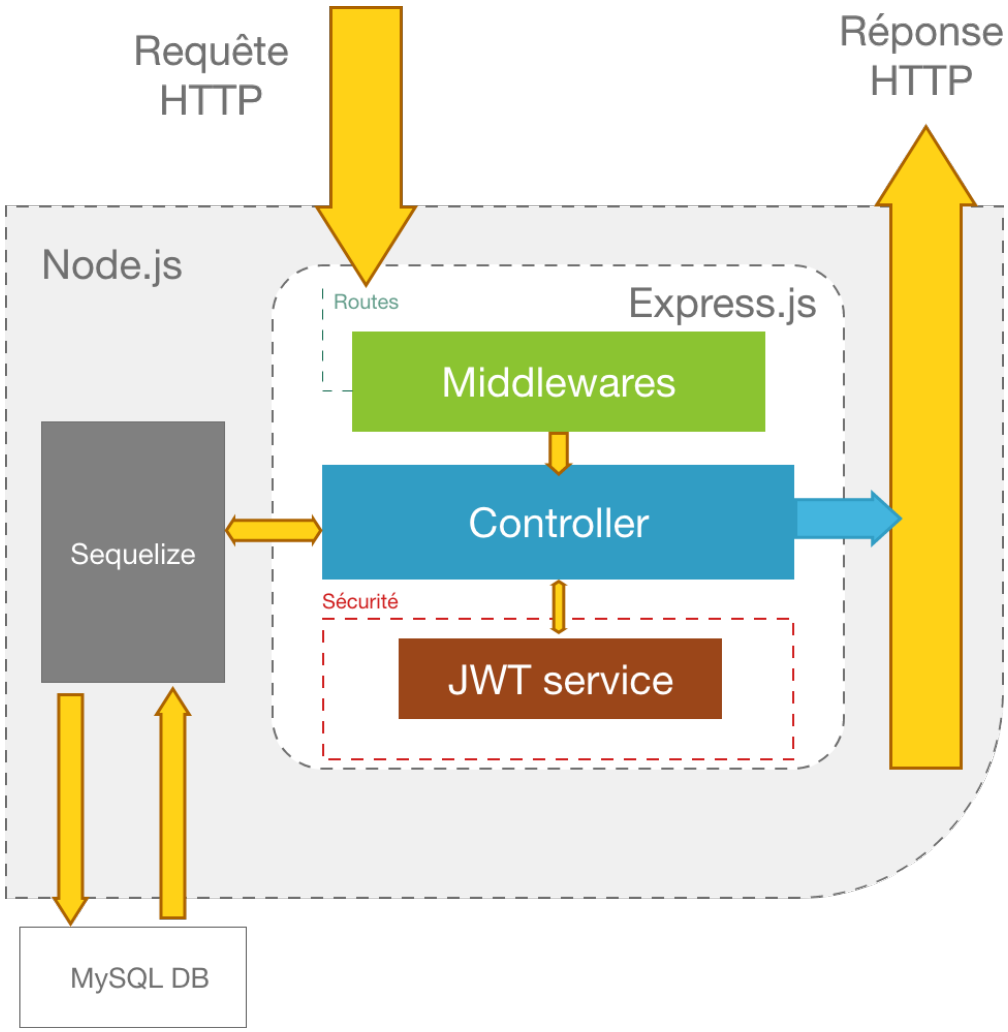
Comment ajouter JWT à mon projet Node.js ?

Qu'est ce que l'authentication basée sur le token ?

L'avantage du json web token (jwt) est que le token (le code d'accès à notre backend) sera enregistré dans l'application client / front : Cache, LocalStorage, Keychain, Sharedpreferences etc …
Dans ce cas le backend pourra être servi en cross plateformes, il sera compatible pour tout types de plateformes et on n'aura pas besoin d'écrire un code spécifique d'authentication pour chacune.



II. Comment l'authentication basée sur jwt sera gérée par le serveur Node.js?



Pour utiliser jwt, il faut installer le module jsonwebtoken : npm install jswonwebtoken
let jwt = require("jsonwebtoken");

L'authentication avec JWT se fait en deux étapes :

- *1. Lors du login.*
- *2. Lors d'un accès à une route sécurisée.*

1. Lors du Login

Lorsqu'un utilisateur se connecte à l'application via /auth/ login , on fait appel au service JWT pour appeler la méthode jwt.sign()

Syntaxe *:*

let token = jwt.sign({ id: id }, config.secret, { expiresIn: 3600 });

Le paramètre expiresIn spécifie la durée de vie du token en secondes. Au delà de cette durée, le token n'est plus valide.

Ce token va être envoyé en réponse à la requête du login avec l'objet User. Et il sera par la suite géré par l'application front pour être sauvegardé et utilisé dans chaque requête pour les routes sécurisées.

```
Response = {
user : user
token : token }
```

2. *Lors d'un accès à une route sécurisée*

Lorsqu'un utilisateur accède à une route sécurisée (uniquement accessible si l'utilisateur est authentifié), on fait appel au service JWT pour utiliser la méthode jwt.verify()

Syntaxe *:*

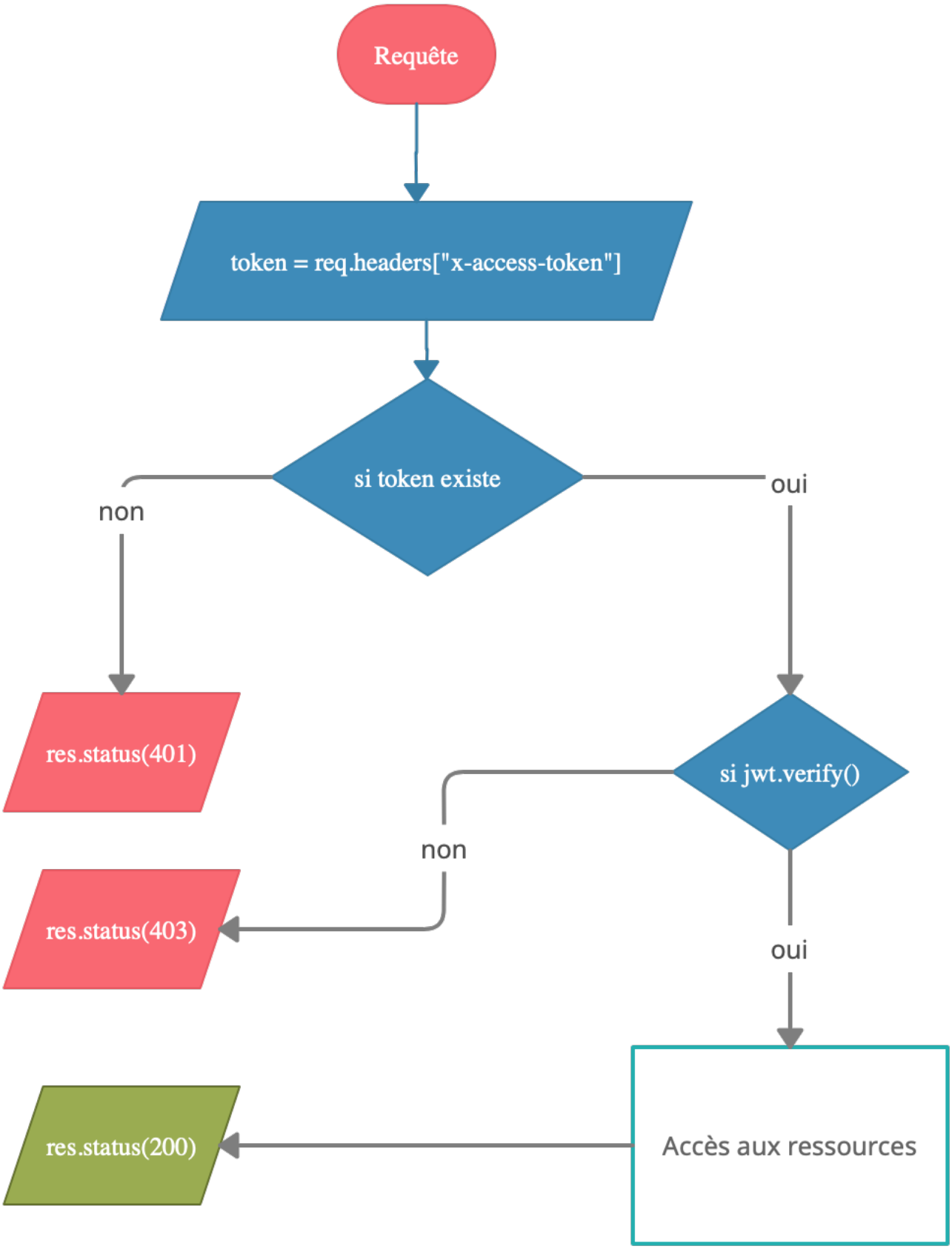
let response = await jwt.verify(token, config.secret);

Le token est récupéré du Header de la requête pour le paramètre "x-access-token".
On accède au token avec :

```
token =req.headers["x-access-token"]
```

Ce Token doit être donc préalablement passé au header dans la requête pour pouvoir être récupéré.

Si le token n'est pas nul et que la fonction jwt.verify() ne produit pas d'erreur , l'utilisateur est considéré comme authentifié et peut donc accéder aux ressources sécurisées


Requête
token = req.headers["x-access-token"]
si token existe
non
oui
res.status(401)
si jwt.verify()
non
res.status(403)
oui
Accès aux ressources
res.status(200)

Comment Gérer les mots de passe dans Node.js ?

Pour des raisons évidentes, les mots de passe ne sont jamais enregistrés en clair dans la base de données. Un hachage est donc requis pour garantir une autre couche de sécurité.

Dans Node.js on peut facilement hasher les mot de passe en utilisant un module spécialement développé pour cette opération : ***bcrypt***

```
npm install bcrypt
let bcrypt = require("bcrypt");
```

Deux méthodes seront utilisées pour **hasher** puis **verifier** le mot de passe.

La premiere méthode **bcrypt.hashSync()** est appelée à la création du compte /auth/register.

```
let hashedPassword =
bcrypt.hashSync(req.body.password, 10)
```

Cette méthode prend le mot de passe et un autre nombre en paramètre.

Ce nombre représente le “salt round” : c’est le nombre d’opérations nécessaires pour hasher le mot de passe 10 salt round = 2^10 opération = 1024 opérations. Plus le salt round est grand plus le mot de passe sera difficile à décrypter et plus de ressources seront demandées pour le crypter.

Notre résultat hadhedPassword contient le mot de passe hashé et sera sauvegardé dans la base de donnée comme étant le mot de passe de l’utilisateur

Exemple de résultat :
“MyPassword “=>
“$2y$10$DQq1jL1djU4f.tdMwVQOi.zmtQdSIKfzhNBac3Xjtcjhc5pWditBu"

La deuxième méthode **bcrypt.compareSync()** est appelée lors du login /auth/login. Elle sert à verifier que le mot de passe envoyé en clair dans la requête correspond bien au mot de passe hashé.

```
var passwordIsValid =
bcrypt.compareSync(
req.body.password,user.password
);
```

req.body.password = “MyPassword”
user.password =
“$2y$10$DQq1jL1djU4f.tdMwVQOi.zmtQdSIKfzhNBac3Xjtcjhc5pWditBu”

Cette méthode retourne un booléen : true si le mot de passe est valide.
On peut par la suite fournir le token jwt à l’utilisateur