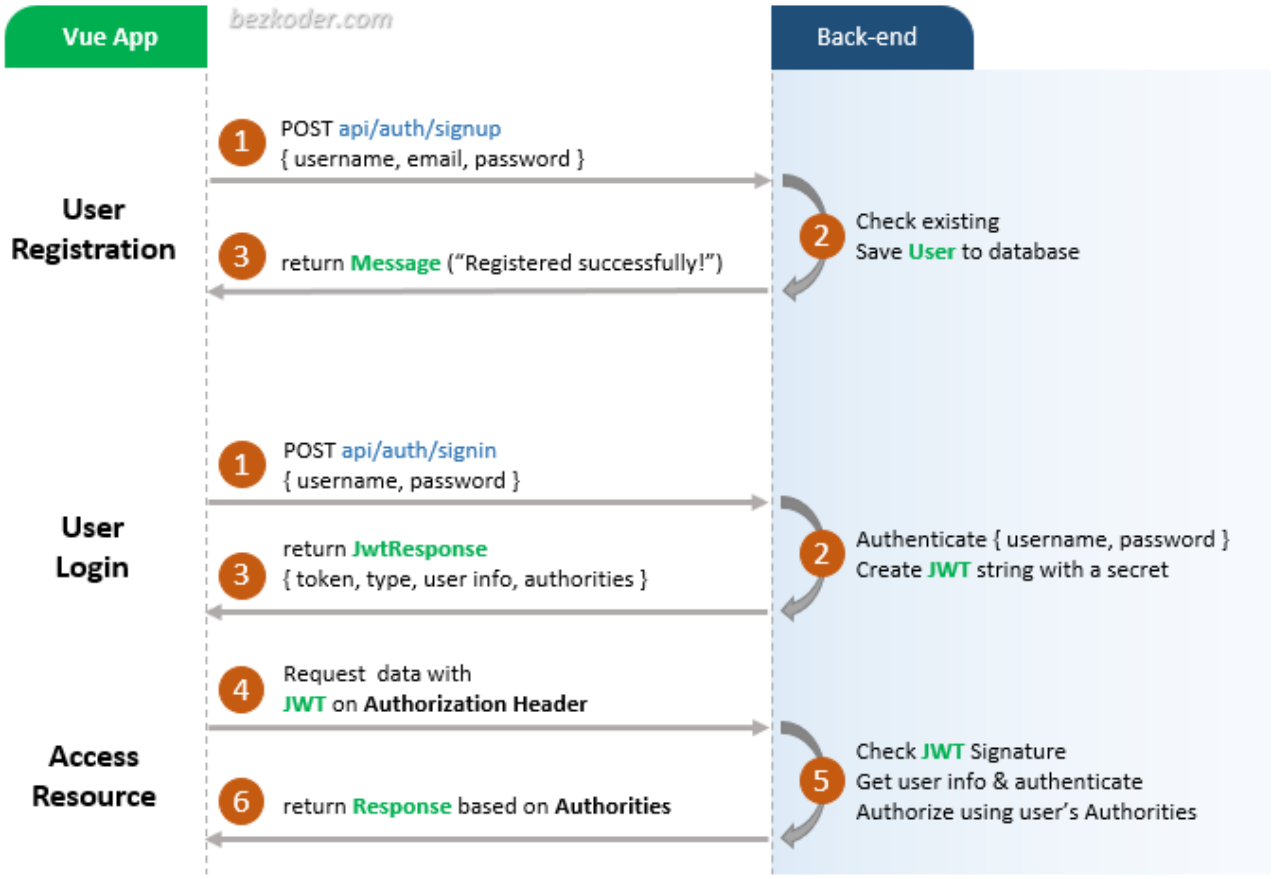
JSONWEBTOKEN

Lors d'une connection authentifiée , un jeton (token) est crée par le serveur et injecté dans l'entête HTTP de la réponse à une requête. Ce jeton est accompagné d"un "payload" qui contient des données clés/valeurs renseignées côté serveur (exemple nom / prenom / role …etc …)

Lorsque q'un autre requête est effectuée , le client doit renvoyer le token dans l'entête de la requête afin d'offrir la possibilité au serveur d'authentifier que la requête effectuée a bien l'autorisation.



1) lancer mongodb
2) dans un repertoire
3) npm i dotenv path jsonwebtoken express mongoose bcrypjs cors
4) lancer node en mode interactif (taper node + <enter>) puis

```
require('crypto').randomBytes(64).toString('hex')
```

génère un chiffre secret qui sera utilisé par jsonwebtoken de node pour créer le token (jeton).

5) copier ce chiffre dans un fichier .env ainsi:

TOKEN_SECRET=<le chiffre généré>

6) écrire le server node avec ces routes

a) route inscription

b) route login (creation du token)

c) route(s) authentifiée(s) pour accéder à des informations.

```
//------
// npm i dotenv express mongoose cors jsonwebtoken bcryptjs path --save
//------
const dotenv = require('dotenv');
const path = require('path');
const jwt = require('jsonwebtoken');
const mongoose = require('mongoose');
const bcrypt = require('bcryptjs'); //
const cors = require('cors'); //



const express = require('express');
// recuperation du contenu de .env (ici TOKEN_SECRET )
dotenv.config();

const app = express();

require('./models/user.js');

var User = mongoose.model('User');

app.use(cors());
```

```
// pour l'echange de données en json

app.use(express.json());

// lancer mongodb si ce n'est pas fait
var db = mongoose.connect('mongodb://localhost/afpauser');
//----------------------------- Generation du TOKEN ------
// en lisant la clé privé sous .env
// la clé est créée par =>require('crypto').randomBytes(64).toString('hex')
// sous le shell node.
// créer un fihier .env et ajouter TOKEN_SECRET = le Token
//-----------------------------------------------------
function generateAccessToken(user) {
  // expire apres 50 minutes
  return jwt.sign(user, process.env.TOKEN_SECRET, { expiresIn: '50m' });
}
/*
exiresIn options
'2 days'  // 172800000
'1d'      // 86400000
'10h'     // 36000000
'2.5 hrs' // 9000000
'2h'      // 7200000
'1m'      // 60000
'5s'      // 5000
'1y'      // 31557600000
'100'     // 100
*/

//-------------------middleware de verification ------
function authenticateToken(req, res, next) {
  // const token = req.headers['x-access-token']
  console.log(req.headers);
  const token = req.headers.authorization.split(" ")[1]
  //const token = req.headers.authorization
  console.log("TOKEN",token);
  if (token == null) { return res.sendStatus(401) } // if there isn't any token
  jwt.verify(token, process.env.TOKEN_SECRET, (err, user) => {
    if (err) return res.sendStatus(403)
    req.user = user
```

```
    next()
  })
}

app.post ('/user/signup', (req,res) =>
{
console.log(req.body);
 User.findOne({ email: req.body.email })
  .exec((err, user) => {
      if (err) {
        res.status(500).send({ message: err });
      }
      if (user) {
        // si le user existe
        res.status(400).send({ message: "Email existe deja!" });
      } else {
        var salt = bcrypt.genSaltSync(10);
        var hash = bcrypt.hashSync(req.body.password, salt);

        // creation de l'objet user à partir de la modelisation User mongoose
        var user = new User({
          name: req.body.name,
          password: hash,
          email: req.body.email
        });

    user.save( function(err) {
      if (err) return handleError(err);
      res.status(200).send(user);
    })
    }
  }) // end exec
}); // end post


app.post ('/user/login', (req,res) => {
console.log(req.body);
 User.findOne({ email: req.body.email })
  .exec((err, user) => {
      if (err) { res.status(500).send({ message: err }); }
```

```
      if (!user) {
        console.log("no user")
        return res.status(404).send({ message: "User Not found." });
      }

      if (user) {
          const isValidPass =
bcrypt.compareSync(req.body.password,user.password)
          if (isValidPass) {
            const token = generateAccessToken({user:user})
            res.status(200).send(token);
          }
        }

   })
})

// test du token
app.get('/api/orders', authenticateToken , function(req, res) {
  console.log("OK TU PASSES!!");
    res.send('Pass OK !!!');
})

app.listen(8092);
console.log('8092');
```

7) Tester les routes de ce serveur avec postman