TD3 : Structure et authentification

1. Télécharger le squelette vu précédemment
2. Intégrer les éléments du TD2
3. Rajouter des champs mot de passe et id aux utilisateurs
4. Utiliser le module `bcrypt` pour hasher le mot de passe
5. Ajouter le middleware `express-session`
6. Créer une table `sessions` (userId, accessToken, createdAt, expiresAt)
7. Créer une resource REST `/sessions`
   - GET / => Affiche un formulaire user/pass
   - POST / => Génère un accessToken et l'enregistre dans la table `sessions`: en HTML on set un cookie `accessToken`, en JSON, on retourne simplement `{accessToken: XXXX}`
   - DELETE / => Supprime un accessToken
8. Créer un middleware qui gère l'authentification :
   - Si mode HTML, alors on vérifie le cookie AccessToken
   - Si mode JSON, alors on vérifie le header X-AccessToken
   - Si pas d'accessToken ou accessToken expiré, en JSON on retourne une erreur, en HTML on redirige vers la page d'authentification

```
// Pour générer un accessToken aléatoire
const hat = require('hat')
const accessToken = hat()
```

TD3 : Annexes

```
const bcrypt = require('bcrypt')

// Hasher un password
bcrypt.hash(lePasswordEnClair, 10).then((hash) => {
  // Insère dans ta db
})
```

```
// Comparer un password clair avec le hash en base
bcrypt.compare(lePasswordEnClair, leHashQuiVientDeLaBase).then((match) => {
  // match = true/false
})
```

npm install —save bcrypt

TD3 : Annexes

```
// app.js
const session = require('express-session')

// Middleware de sessions
app.set('trust proxy', 1)
app.use(session({
  secret: 'topkek',
  resave: false,
  saveUninitialized: true,
  cookie: {
    maxAge: 1000 * 60 * 60,
    httpOnly: true
  }
}))
```

```
// Enregistrer le token dans la session
req.session.accessToken = monToken

// Récupérer le token
console.log(req.session.accessToken)
```

npm install —save express-session