TP Web Services

Gestion d'une bibliothèque

Maintenant que vous êtes aguerri.e.s concernant les WebServices, vous allez créer un service dans la destination est la gestion d'une bibliothèque.

Fonctionnalités

La gestion de la bibliothèque est la suivante :

- Les notions manipulées sont :
  - Des éditeurs « Publisher »,
  - Des auteurs « Author »,
  - Et des livres « Book ».
- Un livre est :
  - Lié à un éditeur et 1 ou plusieurs auteurs,
  - Il est dans l'un des états suivants :
    - Disponible « AVAILABLE »,
    - Emprunté « BORROWED »
    - Ou indisponible « UNAVAILABLE » (pour restauration ou parce qu'on ne le retrouve plus)
- Nous avons différentes sortes d'utilisateurs « User », en voici les rôles « UserRole » :
  - CONSULT_ROLE : c'est le rôle le plus bas, il concerne les utilisateurs de la bibliothèque.
  - BORROW_ROLE : c'est le rôle d'un agent de bibliothèque, il peut :
    - Consulter les informations tout comme le rôle précédent,
    - Il peut créer des utilisateurs dont le seul rôle est le rôle précédent,
    - Il peut créer l'emprunt d'un livre et l'associer à un utilisateur quelconque.
  - CONTRIBUTOR_ROLE : c'est le rôle du bibliothécaire, il peut :
    - Faire ce que fait un agent,
    - Ajouter ou supprimer des livres, éditeurs et auteurs.
  - ADMINISTRATOR_ROLE :
    - On peut tout faire et notamment créer les agents et bibliothécaires,
    - Faire du ménage dans la base de donnée … etc.
- L'emprunt « Borrowing » est marqué par :
  - La liaison entre un utilisateur (par son identifiant) et un livre,
  - Une date d'emprunt,
  - Une durée d'emprunt en jours,
  - Une date de retour. Si on positionne cette date, cet emprunt disparaitra donc de la liste des emprunts en cours, c'est une manière de l'archiver.

Architecture

On scindera le développement en 2 ou 3 services :

- Un service d'accès à la base de donnée de gestion de bibliothèque, il permettra le lien avec la DB et une API permettant d'interroger la base et d'éviter qu'on puisse y accéder directement.
    - Je propose ici d'utiliser SpringBoot, Hibernate et H2 Database sur le port 8081
- Un service qui produira la gestion des utilisateurs (ajout, suppression, vérification du mot de passe).
    - Je propose ici d'utiliser Node.js + Express.js sur le port 8080 mais sur ce point vous pouvez explorer d'autres type de langages/frameworks.
- Vous pouvez ajouter un micro-service de redirection en fonction des requêtes ou le transformer en service complet qui fournit les pages Web et s'appuie sur les 2 autres services.

- Il serait donc judicieux de créer des écrans de gestion à destination des agents/bibliothécaires :

    - À l'aide du système de Template de l'un ou l'autre service (Thymeleaf pour Spring, Pug pour Node, …)
    - Sinon avec un framework d'application Web (React, Angular, View …). Ces technologies ne sont pas le sujet principal et il faudra que ça se lance bien. Je laisserais faire ça à ceux qui sont déjà familiers avec ces frameworks.
    - Ou simplement en HTML + CSS (Bootstrap) + javascript jQuery.

Travail à faire

Réaliser les différents services et les mécanismes de test de l'ensemble (Collections Postman qui seront fournies dans le rendu).

Cependant, on va prioriser le travail.

Dans un premier temps, on ne considérera pas la gestion des rôles. Si vous arrivez à faire tourner les deux services ensemble et à réaliser les opérations demandées sans tenir compte des rôles, ce sera déjà une bonne chose. Cette option est chronophage.

Mais avoir l'ajout et la recherche des auteurs gérée de bout en bout est le minimum, de là la gestion des éditeurs est facile. Vient ensuite la gestion des livres. Ensuite la gestion des utilisateurs (minimaliste) avec la gestion des emprunts de livre. Après il reste la gestion des rôles et la sécurité.

Vous pouvez fournir un ensemble de jeu de test via Postman, ça permet d'éviter une interface compliquée. Bien sûr, ce sera plus simple avec des pages Web même graphiquement sommaires mais capable de réaliser précisément ce que l'on demande.

L'un de vos objectifs prioritaires est de vous assurer que l'ensemble tournera bien au moment de la correction → donc cela doit tourner sur le PC de l'enseignant où être hébergé quelque part et prêt à être testé.

Au moment de la correction, je regarderais donc votre code et je « tenterais » de lancer ce que vous avez fait. Mon expérience m'a montré que l'exécution de ce type de projet de Travaux Pratiques n'est

vraiment pas acquise. Donc des captures d'écrans fonctionnelles seront les bienvenues en compte-rendu.

Les axes d'évolution

La première évolution serait de passer en HTTPS, surtout côté Node.js, cela demande peut d'effort.

La première évolution de ce TP concerne une véritable gestion des rôles. Côté SpringBoot, pour effectuer une action, on a juste besoin d'un jeton définissant le rôle. Pour l'emprunt, on a juste besoin d'un identifiant utilisateur. Mais cet utilisateur n'a pas besoin d'être connu.

Côté Node, on a besoin d'un jeton utilisateur permettant de vérifier que l'utilisateur est connecté.

Et bien sûr, des clients Web plus poussés.

Bibliographie / Webographie

Spring

De nombreux tutos à cet endroit : https://www.baeldung.com/

MongoDB

Documentation : https://docs.mongodb.com/manual/reference/method/
https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/#run-mongodb-from-cmd

Node

https://www.codementor.io/olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd

MongoClient

Nodejs MongoDB : comment utiliser MongoDB sur une app NodeJS ? (practicalprogramming.fr)
mongodb - npm (npmjs.com)

Mongoose

Documentation : https://mongoosejs.com/
Tutorial for Express.js : https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose
Multiple insertion : https://stackoverflow.com/questions/10266512/how-can-i-save-multiple-documents-concurrently-in-mongoose-node-js
Validation : https://vegibit.com/mongoose-validation-examples/
Validation for Embedded Objects: http://jasonjl.me/blog/2014/10/23/adding-validation-for-embedded-objects-in-mongoose/
Unique validator : https://www.npmjs.com/package/mongoose-unique-validator
http://www.codingpedia.org/ama/cleaner-code-in-nodejs-with-async-await-mongoose-calls-example

Javascript & Lodash

https://lodash.com/

Javascript : map, reduce, filter
https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d
Substring
https://stackoverflow.com/questions/1789945/how-to-check-whether-a-string-contains-a-substring-in-javascript
Error handling
https://medium.com/@iaincollins/error-handling-in-javascript-a6172ccdf9af

PUG / JADE

https://pugjs.org/language/code.html
PUG tutorial : Server-Side rendering
https://gist.github.com/joepie91/c0069ab0e0da40cc7b54b8c2203befe1 exemple avec l'affichage des objets locaux

CSS

https://medium.freecodecamp.org/how-to-make-your-html-responsive-by-adding-a-single-line-of-css-2a62de81e431

HTTPS

https://medium.com/@tbusser/creating-a-browser-trusted-self-signed-ssl-certificate-2709ce43fd15
https://certsimple.com/blog/localhost-ssl-fix
https://stackoverflow.com/questions/5998694/how-to-create-an-https-server-in-node-js

Generate self signed certificate

```
openssl genrsa -out client-key.pem 2048
openssl req -new -key client-key.pem -out client.csr
openssl x509 -req -in client.csr -signkey client-key.pem -out client-cert.pem
```

À partir de l'adresse <https://stackoverflow.com/questions/34835859/node-js-https-example-error-unknown-ssl-protocol-error-in-connection-to-localh/35053638#35053638>

Convert PEM to CRT :

`openssl x509 -outform der -in your-cert.pem -out your-cert.crt` (dans mon cas : `-in client-cert.pem -out client-cert.crt`)

À partir de l'adresse <https://stackoverflow.com/questions/13732826/convert-pem-to-crt-and-key>

Redirect HTTP to HTTPS or return indications :

https://stackoverflow.com/questions/7450940/automatic-https-connection-redirect-with-node-js-express
https://www.npmjs.com/package/express-http-to-https

Authentication

First simple approach with passport : https://medium.freecodecamp.org/learn-how-to-handle-authentication-with-node-using-passport-js-4a56ed18e81e
Interesting article and comparison : https://hackernoon.com/your-node-js-authentication-tutorial-is-wrong-f1a3bf831a46
Which leads to : https://github.com/P-H-C/phc-winner-argon2 but it is in C … this one https://github.com/emilbayes/secure-password seems a port for Node.
OAuth2 server : https://oauth2-server.readthedocs.io/en/latest/
Authentication libraries : https://blog.bitsrc.io/6-javascript-user-authentication-libraries-for-2019-6c7c45fbe458

Other tutorials
https://scotch.io/tutorials/authenticate-a-node-js-api-with-json-web-tokens
https://medium.com/dev-bits/a-guide-for-adding-jwt-token-based-authentication-to-your-single-page-nodejs-applications-c403f7cf04f4

The token problematics

https://stackoverflow.com/questions/44133536/is-it-safe-to-store-a-jwt-in-localstorage-with-reactjs
https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage

https://stackoverflow.com/questions/5207160/what-is-a-csrf-token-what-is-its-importance-and-how-does-it-work
https://stackoverflow.com/questions/39163413/node-js-passport-jwt-how-to-send-token-in-a-cookie
https://stackoverflow.com/questions/31919067/how-can-i-revoke-a-jwt-token

https://www.npmjs.com/package/passport-cookie utilisation de CookieStrategy à la place de LocalStrategy ?

Lancement Powershell & Linux

https://stackoverflow.com/questions/25112510/how-to-set-environment-variables-from-within-package-json-node-js
https://www.npmjs.com/package/better-npm-run
Le 'cross-env' ne fonctionne pas bien dans package.json, on peut procéder directement par la commange.

Pour **Windows Powershell** :
`$env:NODE_ENV = "production"`
`node bin/www` ou `npm run start` (voir la commande dans package.json)

Et pour visualiser les variables : `Get-ChildItem Env:` ou `Get-ChildItem Env:NODE_ENV`

Pour **Linux** :
export NODE_ENV=production
et lancement similaire à Windows

HTTPS

https://medium.com/@tbusser/creating-a-browser-trusted-self-signed-ssl-certificate-2709ce43fd15
https://certsimple.com/blog/localhost-ssl-fix
https://stackoverflow.com/questions/5998694/how-to-create-an-https-server-in-node-js

Generate self signed certificate

```
openssl genrsa -out client-key.pem 2048
openssl req -new -key client-key.pem -out client.csr
openssl x509 -req -in client.csr -signkey client-key.pem -out client-cert.pem
```

À partir de l'adresse <https://stackoverflow.com/questions/34835859/node-js-https-example-error-unknown-ssl-protocol-error-in-connection-to-localh/35053638#35053638>

Convert PEM to CRT :

`openssl x509 -outform der -in your-cert.pem -out your-cert.crt` (dans mon cas : `-in client-cert.pem -out client-cert.crt`)

À partir de l'adresse <https://stackoverflow.com/questions/13732826/convert-pem-to-crt-and-key>

Lancement Powershell & Linux

https://stackoverflow.com/questions/25112510/how-to-set-environment-variables-from-within-package-json-node-js
https://www.npmjs.com/package/better-npm-run
Le 'cross-env' ne fonctionne pas bien dans package.json, on peut procéder directement par la commande.

Pour **Windows Powershell** :

```
$env:NODE_ENV = "production"
node bin/www
```

ou `npm run start` (voir la commande dans package.json)

Et pour visualiser les variables : `Get-ChildItem Env:` ou `Get-ChildItem Env:NODE_ENV`

Pour **Linux** :
export NODE_ENV=production
et lancement similaire à Windows