Projet architecture orienté service - Tyfenn ELOY 2023

Ce projet est disponible sur github.

Avancement du projet

A ce jour seulement 6 test sur 48 ne passent pas car je n'ai pas entièrement couvert la dernière partie du projet.

Installation

Pour installer le projet il faut d'abord installer nodejs et npm. Ensuite il faut installer les dépendances du projet avec la commande suivante :

```
npm install
```

Lancement du projet

Pour lancer le projet il faut lancer les 3 micro services avec la commande suivante :

```
node wr_stats.js
node wr_search.js
node wr_ms.js
```

Note: Je recommande de lancer les micro services dans 3 terminaux différents pour une meilleur lisibilité et de préférence dans cet ordre.

Structutre du projet

Fichiers

Le projet est composé de 3 services :

- **wr_ms.js** : micro service de l'iteration 1 - 2
- **wr_stats_ms.js** : micro service de l'iteration 3 (statistiques)
- **wr_search_ms.js** : micro service de l'iteration 4 (recherche avec minisearch) (WIP)

Et des fichiers suivants :

- **restWr.js** est le fichier qui permet de faire le lien entre les micro services et le client (il fait ici office d'**API**). C'est ici que les **routes** sont définies.
- **wr.json** est le fichier qui contient les données de l'application. Ce fichier est généré par le micro service wr_ms.js dès qu'une opération **PUT** / **POST** / **DEL** est effectuée.

- **stats.json** est le fichier qui contient les statistiques. Ces statistiques sont générée et mise a jour par le micro service wr_ms.js dès qu'une opération **PUT** / **POST** / **DEL** est effectuée.
- **stat_del.json** est le fichier qui contient le nombre de wr supprimé par utilisateur. Ce fichier est généré par le micro service wr_ms.js dès qu'une opération **DEL** est effectuée.

L'ensemble du projet utilise seneca-web pour la communication entre les micro services et le client.

Routes

Les routes sont définies dans le fichier **restWr.js**. Elles sotel que :

Route	Paramètres	Fonction	Micro service	Role	Méthode
/api/wr/	-	create	wr_ms.js	wr	POST
/api/wr/	id (optionnel)	getById	wr_ms.js	wr	GET
/api/wr/	id	updateWr	wr_ms.js	wr	PUT
/api/wr/	id	deleteWr	wr_ms.js	wr	DEL
/api/wr/	-	deleteAllWr	wr_ms.js	wr	DEL
/api/wr/stats/	applicant (optionnel)	getWrStats	wr_stat_ms.js	stats	GET
/api/wr/search/	query	searchWr	wr_search_ms.js	search	GET

Micro services

Les microservices sont définis comme tel:

Micro service	Nom plugin	Port	Méthode	Fonction
wr_ms.js	wr	3000	POST / GET / PUT / DEL	Gestion des wr
wr_stat_ms.js	stats	4001	GET	Gestion des statistiques
wr_search_ms.js	wrSearch	4002	GET	Gestion de la recherche

Les messages

Les messages sont définis comme tel :

Service wr_ms.js:

Message	Role	Action	Paramètres	Description	Reponse
role:wr,cmd:create	wr	create	data	Créer un wr	L'element créer + succes
role:wr,cmd:getById	wr	getById	id?	Retourne la liste des wr, si id définie ne retourne que mes wr correspondant	La liste des + sucess

Message	Role	Action	Paramètres	Description	Reponse
role:wr,cmd:updateWr	wr	updateWr	id, data	Met à jour un wr	L'element mis à jour + succes
role:wr,cmd:deleteWr	wr	deleteWr	id	Supprime un wr	L'element supprimé + succes
role:wr,cmd:deleteAllWr	wr	deleteAllWr	-	Supprime tous les wr	Succes

Service wr_stat_ms.js:

Message	Role	Action	Paramètres	Description	Reponse
role:wr,cmd:getWrStats	stats	getWrStats	applicant?	Retourne les statistiques des wr, si applicant définie ne retourne que les statistiques de l'applicant spécifié	Les statistiques + succes

Service wr_search_ms.js:

Message	Role	Action	Paramètres	Description	Reponse
role:wr,cmd:searchWr	search	searchWr	query	Retourne les wr correspondant à la recherche	Les wr correspondant + succes

Difficultés rencontrées

Je me suis rapidement rendu compte qu'il vaudrais mieux mettre en place une architecture MVC mais je n'ai pas réussis à utiliser les modèles (les fichiers existent toujours dans ./models).

Je n'ai pas non plus réussis à faire communiquer les services wr_ms.js et wr_stat_ms.js par seneca-web.

C'est pour celà que j'ai fait la communication par fichier ce qui en soit me paraissait être la seule solution (*Note: Il doit rester des vestiges de mes essais dans mon repository git si vous voulez vérifier*).

Conclusion

Le projet à été compliqué à mettre en place à cause du peux de documentation sur seneca-web spécifiquement. A peut pret 70% des tests passent mais j'aurrais préférer avoir un résultat plus concluant sur la communication inter-service.