Liste de courses

L'exercice « liste de courses » est en deux parties.

Dans un premier temps je fournis le serveur et vous devez coder le client (HTML/CSS/JavaScript) permettant de maintenir une liste de courses hors connexion depuis plusieurs appareils.

Dans un second temps, vous pourrez faire la partie serveur répondant à votre client, mais aussi à ceux qui sont proposés pour vous assurer que votre serveur est conforme aux demandes.

En téléchargement sur DVO vous trouverez :

- ListeDeCourse-Sources.zip : contient la doc de l'API à mettre en place et les sources des programmes client et serveur en Delphi
- TODOList-VueJS.zip : le source d'une gestion de TODO liste utilisant VueJS pour son affichage. Vous pouvez vous en inspirer si vous voulez.
- ListeCoursesServeur-Win32.exe : un serveur local Windows 32 bits pour gérer la liste de courses sans connexion Internet.
- ListeCoursesServeur-Win64.exe : un serveur local Windows 64 bits pour gérer la liste de courses sans connexion Internet.
- ListeCourses-Win32.msix : un client de mise à jour de la liste de courses pour Windows en 32 bits.
- ListeCourses-Win64.msix : un client de mise à jour de la liste de courses pour Windows en 64 bits.
- ListeCourses-Mac.zip : un client de mise à jour de la liste de courses pour Mac (version pour processeur Intel mais fonctionnel sur M1 ou M2).

Les objectifs de ce TD sont :

- Vous faire manipuler JavaScript sur un vrai projet et éventuellement un framework d'affichage (VueJS, ReactJS, jQuery, ou celui que vous voulez, au choix).
- Vous faire manipuler des API récentes des navigateurs.
- Vous faire réfléchir sur une technique de synchronisation de données.
- Vous faire travailler sur un serveur d'API.

Serveurs de test

Pour créer votre client de gestion de liste de courses vous avez besoin d'un serveur.

Sous Windows vous pouvez utiliser celui qui est téléchargeable en 32 ou 64 bits.

Il suffit de lancer le programme et d'autoriser le firewall en « réseau privé » (jamais « réseau public ») si Windows vous le demande. Le serveur local propose son API à l'adresse http://localhost :8073

Si vous êtes sous Mac ou Linux je peux compiler un serveur pour votre version mais il est possible que ça ne passe pas pour différentes raisons. Il est préférable que vous travailliez avec la version disponible sur Internet.

Le serveur officiel pour cette liste de courses propose son API à l'adresse https://esilv.olfsoftware.fr/td5/

C'est quoi une liste de courses ?

La liste de courses contient des produits et des quantités pour chaque produit.

Vous devez pouvoir ajouter un nouveau produit à la liste et modifier les quantités de chaque.

Synchronisation de données

Il existe de multiples méthodes pour synchroniser des bases de données. La plus simple consiste à synchroniser l'historique des modifications et les appliquer sur chaque appareil. C'est ce que je vous propose ici.

Chaque ajout de produit et chaque modification de quantité doivent être stockés pour pouvoir être transmis au serveur de temps en temps ou sur demande.

Dans le cas de cet exercice je recommande un bouton de synchronisation, plus facile à mettre en place et à tester.

Dans des projets « grand public » il est préférable de masquer cette notion de synchronisation en automatisant l'envoi des informations en attente à partir d'une horloge ou lorsqu'on détecte une reconnexion à internet suite à une coupure. Certaines API JavaScript des navigateurs nous donnent l'information. C'est l'approche choisie par les éditeurs de suites bureautique en ligne (Google Docs, Microsoft Office 365 par exemple).

Lors de la synchronisation vous envoyez vos modifications et vous demandez également les mises à jour faites par d'autres applications depuis votre dernière synchronisation.

Selon le type de synchronisation et le type de base de données, cette étape doit être faite dans un sens particulier. Dans notre cas cela ne devrait rien changer.

La liste de courses complète et à jour n'est fournie qu'une seule fois : lors de la première connexion au serveur. Ensuite seules des mises à jour sont échangées.

Vous le verrez, ce n'est pas trop compliqué à mettre en œuvre.

Malheureusement cette technique ne s'applique pas à tout type de données comme par exemple lorsqu'on travaille sur des bases de données avec de nombreuses tables et des clés autoincrémentées puisque plusieurs utilisateurs peuvent créer des choses dans les mêmes tables hors connexion avec le serveur. Ca peut générer des anomalies lors de la synchronisation.

Validation de votre programme client

Si vous travaillez en local vous devrez pouvoir basculer en ligne pour valider votre application.

En fin de travail sur la première partie tout le monde doit pouvoir se synchroniser avec le même serveur et obtenir la même liste même si des mises à jour sont effectuées quelque part.

En fin de travail sur la seconde partie, tous les programmes clients doivent pouvoir se synchroniser avec les serveurs de chacun. Le changement de serveur de synchronisation réinitialisant la liste des données traitées.

Première partie : affichage et modification de la liste de courses depuis un navigateur web

Pour démarrer ce TD vous manipulerez uniquement des fichiers côté client. Considérez que vous êtes sur un site statique, sans serveur. Vous n'avez pas besoin de base de données ni de serveur web local.

Téléchargez les exemples et les ZIP dont vous avez besoin et configurez votre dossier de travail.

Choisissez un framework : ReactJS, VueJS, Bootstrap, TailwindCSS ou un autre si vous préférez. Vous pouvez aussi vous aider des exemples de « Test AJAX JavaScript » ou tout faire à la main.

Partez d'un exemple ou d'un modèle de page HTML5 vierge.

Mettez en place dans le code la partie HTML dont vous aurez besoin pour l'affichage et la mise à jour de la liste de courses (nom de produit et quantité).

Ajoutez ensuite la partie JavaScript vous permettant de passer de la liste à la saisie d'un nouveau produit ou la modification de la quantité d'un produit existant.

L'utilisateur doit pouvoir fermer le navigateur ou se déconnecter d'Internet puis retrouver ses données lorsqu'il revient sur votre page (en tant que fichier ou adresse web). Faites donc

en sorte que ces informations persistent entre deux lancements de votre navigateur et pensez à une option pour tout nettoyer sur demande.

Traitez enfin l'API permettant de récupérer la liste de courses de départ depuis un serveur et de lui transmettre les modifications.

Lorsque vous aurez réussi à synchroniser vos données avec le serveur, essayez avec plusieurs navigateurs ouverts pour vous assurer que vos modifications se synchronisent correctement. Vous pouvez aussi utiliser les programmes clients compilés fournis pour vos tests.

Seconde partie : développement d'un serveur d'API

Une fois que vous aurez fini de développer la partie cliente de votre application, lancez-vous dans le développement d'un serveur qui répondra à son API.

Dans ce cas vous devrez réutiliser les serveurs web locaux (XAMPP, WAMP, MAMP ou EasyPHP) afin de travailler en PHP. Vous pouvez travailler en natif ou passer par un framework si vous préférez (mais franchement, pour trois appels d'URL, ce n'est vraiment pas nécessaire).

Bien entendu l'API ne change pas. A vous de faire en sorte de pouvoir changer l'URL de votre gestion de liste de courses et pointer sur votre propre serveur ou sur celui de votre voisin.

Vous êtes libres d'utiliser ou pas une base de données sur le serveur. C'est recommandé car on est susceptible d'y accéder à plusieurs, mais une gestion sous forme de fichiers est aussi possible. A vous d'y réfléchir et faire vos choix.

Idéalement vous devez pouvoir mettre votre serveur en ligne sur l'hébergement que je vous ai fourni (ou un autre à vous) et partager l'URL de votre serveur pour qu'on le teste.

Corrections et solutions

La correction sera mise en ligne avant le prochain cours.

Elle contient :

- les codes sources d'un serveur d'API en PHP qui utilise une base de données MySQL pour stocker les informations (liste de courses, liste de clients et journal des modifications)
- les codes sources d'un client web utilisant VueJS et XMLHttpRequest pour Ajax

En plus du ZIP à télécharger sur DVO vous pourrez aussi accéder (sur demande) au dépôt GitHub privé qui vous permettra d'en suivre les évolutions.

Comme toujours si vous voulez que je regarde et commente ce que vous avez codé transmettez moi un ZIP de vos fichiers par email.