LOG4420 – Conception de sites web dynam. et transact.

Travail pratique 5

Chargé de laboratoire:

Antoine Béland

Automne 2017

Département de génie informatique et de génie logiciel

1 Objectifs

Le but de ce travail pratique est de vous familiariser avec le *framework* Angular et l'approche MVC (Modèle-vue-contrôleur) sur le client.

Plus particulièrement, vous aurez à concevoir une application web avec Angular 4 à partir du site web d'achats en ligne que vous avez réalisé lors des autres travaux pratiques. En ce sens, vous devrez migrer toute la logique du site web vers des composants et des services Angular, et vous aurez à communiquer avec les services web que vous avez mis en place lors du travail pratique 4 pour récupérer ou mettre à jour les données sur le serveur.

2 Introduction

Lors des autres travaux pratiques, vous aviez à réaliser un site web qui disposait de plusieurs routes pour accéder aux différentes pages. Ainsi, il était nécessaire d'effectuer des requêtes vers le serveur pour récupérer une page en particulier ainsi que pour charger les ressources contenues dans celle-ci. Tout cela demandait un certain temps de chargement et cela peut parfois affecter négativement l'expérience utilisateur.

Afin d'éviter les longs chargements entre les pages d'un site web, les applications web monopage (*single-page application*) sont devenues très populaires depuis quelques années. En effet, comme leurs noms l'indiquent, une seule page compose le site. En ce sens, il est uniquement nécessaire de récupérer les différentes ressources lors du chargement initial de la page. Par la suite, les mises à jour des vues et des données se font grâce à des requêtes AJAX, ce qui est transparent pour l'utilisateur. Lors de ce travail pratique, vous aurez justement à mettre en place une application web monopage à l'aide d'Angular.

3 Travail à réaliser

À partir du code que vous avez produit lors des travaux pratiques précédents, vous aurez à réaliser une application web pour la plateforme d'achats en ligne avec Angular 4. Puisqu'un gabarit assez complet vous est fourni et que la structure du code Angular est complètement différente, vous n'avez pas à réutiliser du code des travaux pratiques précédents. Le corrigé du travail pratique 4 demeure toutefois disponible sur Moodle pour vous aider au besoin.

Notez bien

Il est **fortement** recommandé de faire le tutoriel proposé sur le site web d'Angular avant de débuter le travail. En effet, dans le cas contraire, il peut être assez difficile de comprendre le *framework*. Dans ce tutoriel, comme dans ce travail pratique, vous aurez à écrire votre code côté client en TypeScript.

Avant de débuter, assurez-vous d'avoir récupéré l'archive associée à ce travail pratique sur Moodle. Cette archive contient le gabarit à utiliser pour organiser correctement votre code Angular.

3.1 Prise en charge du gabarit fourni

La première étape à réaliser pour ce travail pratique est de se familiariser avec le gabarit qui vous est fourni. En ce sens, cette section présente les éléments importants pour faciliter cette prise en charge.

3.1.1 Structure du gabarit

Le gabarit fourni comporte trois dossiers principaux : « `client` », « `server` » et « `tests` ». Le dossier « `client` » contient tout le code qui sera exécuté du côté client, le dossier « `server` » inclut l'ensemble du code nécessaire pour l'exécution du serveur et le dossier « `tests` » contient les tests d'acceptation à exécuter pour valider votre travail. Par souci de clarté, la structure du gabarit est illustrée à la figure 1.

Avant de débuter, vous devez mettre à jour le *connect string* (`mongodb://...`) qui se trouve dans le fichier « `server/lib/db.js` » afin que vous soyez en mesure de vous connecter à votre base de données sur mLab.

Les seuls fichiers que vous aurez à ajouter et à modifier se trouvent dans le dossier « `client` ». En effet, le code de l'application Angular se trouve dans le dossier « `client/src-/app` ». C'est dans ce répertoire que vous aurez à compléter l'application Angular qui vous est fournie.

En ce qui concerne le dossier « `server` », vous n'avez pas à réaliser de changements dans ce répertoire. Effectivement, le code du serveur vous est fourni pour ce travail. Ainsi, celui-ci définit les mêmes API qui étaient demandées au TP4 à une différence près. Contrai-

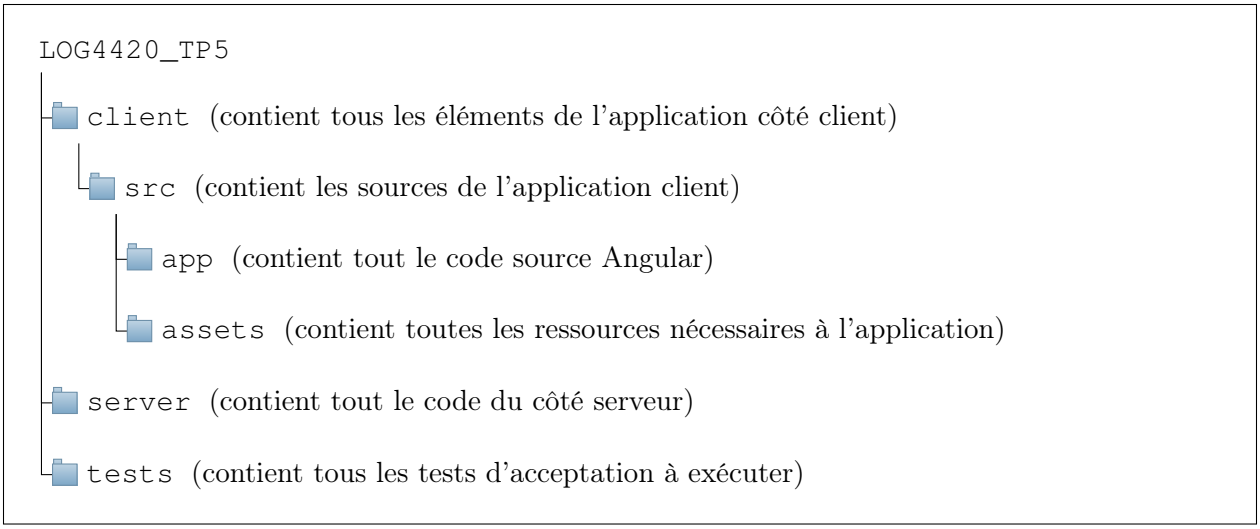

FIGURE 1 – Hiérarchie des dossiers du gabarit fourni

rement au quatrième travail pratique, le serveur prend en charge les requêtes multi origines (*cross-site*). En ce sens, le serveur pourra répondre à une requête provenant du domaine « `http://localhost:8000` » alors que le serveur fonctionne sur le domaine « `http://localhost:3000` ». Ce mécanisme est nécessaire pour ce travail pratique puisque le serveur fonctionnera sur un port différent de l'application client. Par défaut, HTTP n'autorise pas ce type de requêtes pour des raisons de sécurité. Pour en savoir plus sur le partage de ressources de différentes origines (*Cross-Origin Resources Sharing*), vous pouvez consulter ce lien.

3.1.2 Installation et exécution de l'application

Pour être en mesure d'exécuter l'application, il faut d'abord installer les dépendances nécessaires au fonctionnement du serveur et d'Angular en tapant la commande suivante dans le terminal à la racine du projet :

```
npm install
```

Une fois toutes les dépendances installées, il vous suffit de taper la commande suivante dans le terminal pour démarrer simultanément le serveur et l'application Angular :

```
npm start
```

Si tout se passe comme prévu, le serveur sera fonctionnel à l'adresse suivante : `http://localhost:3000`. L'application client, quant à elle, sera accessible à l'adresse suivante : `http://localhost:8000`.

3.2 Réalisation de l'application Angular

La deuxième étape de ce travail pratique est de compléter l'application Angular 4 qui vous est fournie. En effet, à partir de ce que vous avez appris dans le tutoriel se trouvant sur le site d'Angular, vous aurez à compléter les différents composants correspondants aux pages du site d'achats en ligne que vous aviez à réaliser tout au long de la session. Ces composants se trouvent dans le dossier « `client/src/app` ». Il est à noter que le projet est basé sur Angular CLI. Vous pouvez donc utiliser les commandes disponibles via le terminal pour créer de nouveaux éléments au besoin.

Afin de vous simplifier la tâche, un bon nombre d'éléments ont été réalisés pour vous. En ce sens, les routes vers les différents pages ont déjà été configurées à l'aide du module de routage d'Angular et le service permettant de communiquer avec l'API des produits vous est fourni. De plus, la déclaration des composants nécessaires pour le site web d'achats en ligne ainsi que la définition du code HTML des vues utilisées par les composants ont été faites pour vous. Le tableau 1 décrit d'ailleurs les différents composants qui vous sont fournis.

TABLEAU 1 – Description des composants fournis

Nom	Description
`AppComponent`	Composant principal de l'application.
`ConfirmationComponent`	Composant responsable d'afficher la page de confirmation.
`ContactComponent`	Composant responsable d'afficher la page de contact.
`HomeComponent`	Composant responsable d'afficher la page d'accueil.
`OrderComponent`	Composant responsable d'afficher la page de commande.
`PageNotFoundComponent`	Composant responsable d'afficher un message d'erreur si la page est non trouvée.
`ProductComponent`	Composant responsable d'afficher la page d'un produit.
`ProductsComponent`	Composant responsable d'afficher la page des produits.
`ShoppingCartComponent`	Composant responsable d'afficher la page du panier d'achats.

Pour ce travail, vous aurez à compléter les composants suivants : **`AppComponent`**, **`ProductsComponent`**, **`ProductComponent`**, **`ShoppingCartComponent`**, **`OrderComponent`** et **`ConfirmationComponent`**. Pour ce faire, vous devez vous assurer de respecter les spécifications des différentes pages du site web qui ont été décrites dans l'énoncé du travail pratique 3. De plus, vous devez

communiquer avec les API des produits, du panier d'achats et des commandes pour mettre à jour les informations sur le client. Afin de vous assurer que vous respectez toutes les exigences demandées pour chacune des pages du site web, des tests d'acceptation automatisés vous ont été fournis afin que vous puissiez valider votre application (voir §Exécution des tests d'acceptation automatisés).

Vous aurez également à créer un *pipe* pour formater correctement les prix des différents produits. À titre de rappel, les prix doivent utiliser la virgule (« , ») comme séparateur décimal tout en ayant deux nombres décimaux en tout temps.

Assurez-vous que les communications entre votre application et vos services web sont encapsulées dans des services Angular. Également, veuillez utiliser le service Http pour réaliser des requêtes AJAX vers vos API. Comme cela a été mentionné plus tôt, le service responsable de communiquer avec l'API des produits a déjà été défini pour vous (voir `ProductsService`). Vous avez donc uniquement à définir deux nouveaux services : un service pour la gestion du panier d'achats et un autre pour la gestion des commandes. Assurez-vous de consulter l'annexe A du présent document pour comprendre comment manipuler correctement un témoin de session (*cookie*) provenant d'une autre origine.

Notez bien

Vous ne devez pas utiliser jQuery ou manipuler le DOM directement pour mettre à jour les différentes vues. En effet, vous devez utiliser la syntaxe Angular pour créer et manipuler vos vues (liaison unidirectionnelle et bidirectionnelle, boucles, conditions, etc.).

3.3 Exécution des tests d'acceptation automatisés

Tout comme pour les deux derniers travaux pratiques, des tests d'acceptation automatisés vous ont été fournis afin que vous puissiez vérifier votre application. En effet, ces tests permettront de valider que votre application web monopage respecte toutes les exigences fonctionnelles demandées. De plus, ces tests serviront à l'évaluation de votre travail. En ce sens, il est important que vous vous assuriez que tous les tests fonctionnent **avant** de remettre votre travail pratique. Pour exécuter les tests d'acceptation, vous devez entrer la commande suivante dans un terminal :

```
npm run e2e
```

Il est à noter que l'environnement de tests d'acceptation automatisés est actuellement configuré pour fonctionner dans le laboratoire de Polytechnique (système d'exploitation Linux). Si vous souhaitez exécuter les tests sur un autre environnement, vous pouvez jeter un coup d'œil à ce lien (voir fichier « `nightwatch.json` » dans le dossier « `tests` »).

Avertissement

Il est important de n'apporter **aucune** modification aux fichiers dans le dossier `tests` (sauf « `nightwatch.json` »). En effet, ces fichiers contiennent tous les tests automatisés qui sont exécutés sur le site web et se doivent de ne pas être modifiés.

Conseils pour la réalisation du travail pratique

1. Consultez l'aide-mémoire d'Angular pour vous aider à identifier rapidement la commande dont vous avez besoin.
2. Utilisez un environnement de développement intégré (IDE) pour faciliter le développement de votre application Angular (voir WebStorm ou Visual Studio Code).
3. N'attendez pas à la dernière minute pour commencer le laboratoire ! L'apprentissage et la prise en charge d'Angular prennent un certain temps.

4 Remise

Voici les consignes à suivre pour la remise de ce travail pratique :

1. Vous devez placer le code de votre projet dans un dossier compressé au format ZIP nommé « `TP5_matricule1_matricule2.zip` ». Assurez-vous d'exécuter la commande « `npm run clean` » dans un terminal à la racine du projet et d'exclure le dossier « `tests` » avant de remettre votre travail.
2. Vous devrez également créer un fichier nommé « `temps.txt` » à l'intérieur du dossier de votre projet. Vous indiquerez le temps passé au total pour ce travail.

3. Le travail pratique doit être remis avant **23h55**, le **6 décembre 2017** sur Moodle.

Aucun retard ne sera accepté pour la remise de ce travail. En cas de retard, le travail se verra attribuer la note de zéro. Également, si les consignes 1 et 2 concernant la remise ne sont pas respectées, une pénalité de -5% est applicable.

Le navigateur web **Google Chrome** sera utilisé pour tester votre site web.

5 Évaluation

Globalement, vous serez évalué sur votre structure de code Angular ainsi que sur le respect des fonctionnalités du site web. Plus précisément, le barème de correction est le suivant :

Exigences	Points
Respect des exigences du site web	
Résultats des tests d'acceptation automatisés	12
Application Angular	
Utilisation adéquate des différents éléments Angular	6
Qualité et clarté du code TypeScript	2
Total	20

L'évaluation se fera en grande partie grâce aux tests d'acceptation automatisés qui vous sont fournis.

Ce travail pratique a une pondération de **9%** sur la note du cours.

6 Questions

Si vous avez des interrogations concernant ce travail pratique, vous pouvez poser vos questions sur le canal `#tp5` sur Slack. N'hésitez pas à poser vos questions sur ce canal afin qu'elles puissent également profiter aux autres étudiants. De plus, vous pouvez rejoindre le chargé de laboratoire sur Slack (`@antoinebeland`) si vous souhaitez lui poser une question en privé.

Annexes

A Utilisation d'un témoin de connexion (*cookie*) provenant d'une autre origine

Dans ce travail pratique, il sera nécessaire d'utiliser un témoin de connexion provenant d'une autre origine (*cross-domain cookie*) pour conserver l'identifiant de la session qui sera générée par le serveur. Puisque le serveur fonctionne sur un port différent (port 3000) que l'application client (port 8000), il est nécessaire de spécifier un argument supplémentaire dans les requêtes AJAX qui seront réalisées vers l'API du panier d'achats pour que l'application client puisse conserver le témoin de session. Par défaut, cela n'est pas autorisé par HTTP pour des raisons de sécurité. Ainsi, il est obligatoire de spécifier le paramètre « `withCredentials` » à « `true` » pour que ce mécanisme fonctionne.

Par souci de clarté, un exemple de code effectuant une requête de type POST vers l'API du panier d'achats en utilisant le service Http d'Angular est illustré à la figure 2. Il est à noter que les variables « `productId` » et « `quantity` » ne sont pas définies pour cet exemple. De plus, l'exemple suppose que la variable « `http` », qui correspond au service Http d'Angular, est une propriété de la classe du service utilisant ce code.

```
const headers = new Headers({ 'Content-Type': 'application/json' });

// ***** Il est nécessaire de mettre la propriété "withCredientials" à TRUE. *****
const options = new RequestOptions({ headers: headers, withCredentials: true });

this.http.post('http://locahost:3000/api/shopping-cart', JSON.stringify({
  productId: productId,
  quantity: quantity
  }), options);
```

FIGURE 2 – Exemple de code effectuant une requête utilisant une session provenant d'une autre origine