TP 4
EFREI 3A WEB

Louis Cherel - 08/2020

Préambule

Merci de lire **toutes les consignes**, elles sont écrites pour être lues.

Vous avez 3 jours sans pénalité

Si vous avez cours lundi => jusqu'à mercredi 23h55
mardi => jusqu'à jeudi 23h55

Chaque jour de retard vous retire 1 point sur votre note de TP, avec une limite à 3 jours de retard, ce qui, passé ce délai, vous vaudra une note de 0 sur votre TP.

Seul un des TPs sera noté, mais ne pas rendre les autres vous vaudra quand même 0.

Les TP et le projet sont à réaliser en binôme ou trinômes, et doivent être enregistrés auprès de votre enseignant·e avant la fin de la première séance.
Sans votre enregistrement d'équipe, vous ne pourrez pas rendre votre travail.

Les ressources conseillées

Vos armes seront donc le site officiel de Vue.js, Mozilla Developer Network (MDN), Stack Overflow, et les recherches en anglais. **Évitez au maximum** les forums (pas les cours) des sites comme OpenClassrooms ou Comment Ça Marche, qui proposent des solutions souvent peu fiables. W3Schools peut être une bonne ressource, mais les bonnes pratiques et les standards ne sont pas leur fort. Leur préférer le MDN lorsque possible.

Quantité de blogs sont très bons, notamment alsacreations (en français) ou css-tricks (en anglais).

Installations préalables

Pour réaliser ce TP, vous aurez besoin d'installer sur votre ordinateur:

- Postman: Outil permettant de tester notre API sans passer par le navigateur
- Node.js: ce sera notre serveur Web, il devrait déjà être installé sur votre machine

Exercices

Exercice 1 (🕒 20 min pratique)

1) Téléchargez l'archive suivante qui contient le squelette du TP. Cette archive contient les éléments nécessaires au bon fonctionnement du reste du TP. Assurez-vous de bien avoir installé Node.js avant toute chose.

2) Extrayez l'archive dans un dossier quelconque de votre PC.

3) Ouvrez PowerShell (windows), ou un terminal sur linux/mac, naviguez jusqu'au nouveau dossier créé, et installez les dépendances du projet en tapant la commande suivante depuis le dossier créé: `npm install`

4) Si tout s'est bien passé, un nouveau dossier `node_modules` devrait être apparu dans le dossier du projet. Ce dossier contient tous les modules externes à notre propre code.

5) Pour lancer notre serveur, entrez la commande suivante: `npm start`

6) Ouvrez votre navigateur web et entrez l'URL suivante: http://localhost:3000

7) Si tout s'est bien passé, vous devriez constater que le site contient des articles avec des prix et des images

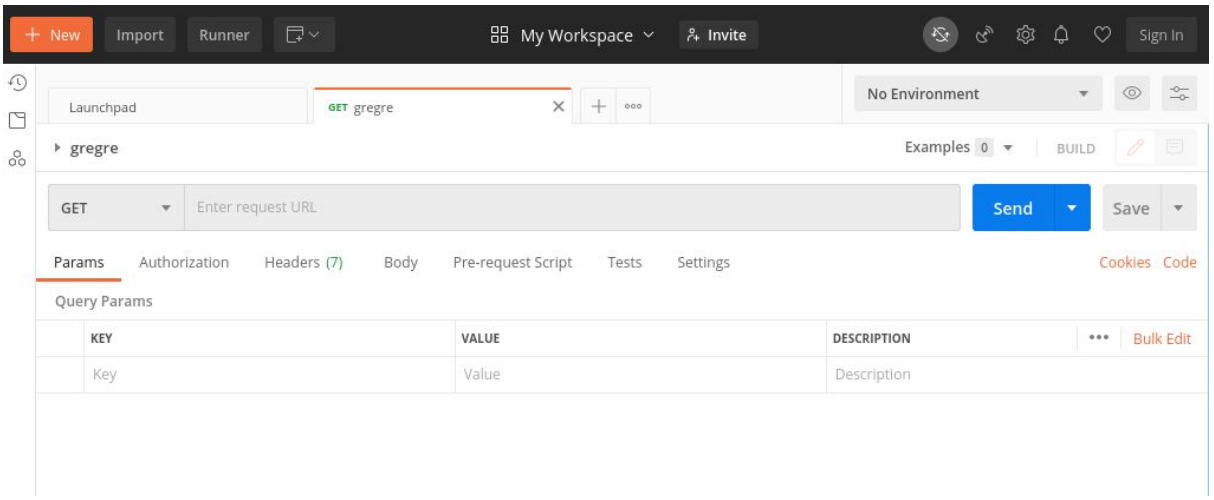
8) Lancez Postman

    a) Sur le premier écran, vous pouvez continuer sans vous inscrire avec un bouton en bas de l'interface

    b) Cliquez sur "New", puis "Request"

    c) Saisissez un nom quelconque dans "request name", puis cliquez sur "create collection", et enfin sur "save to 'your collection'"

d) Vous devriez arriver à un écran comme celui-ci:



9) Nous allons vérifier que l'API fonctionne bien

a) Entrez l'URL suivante: http://localhost:3000/api/articles

b) Cliquez sur send, et regardez le résultat. Cela devrait retourner la liste des articles du site

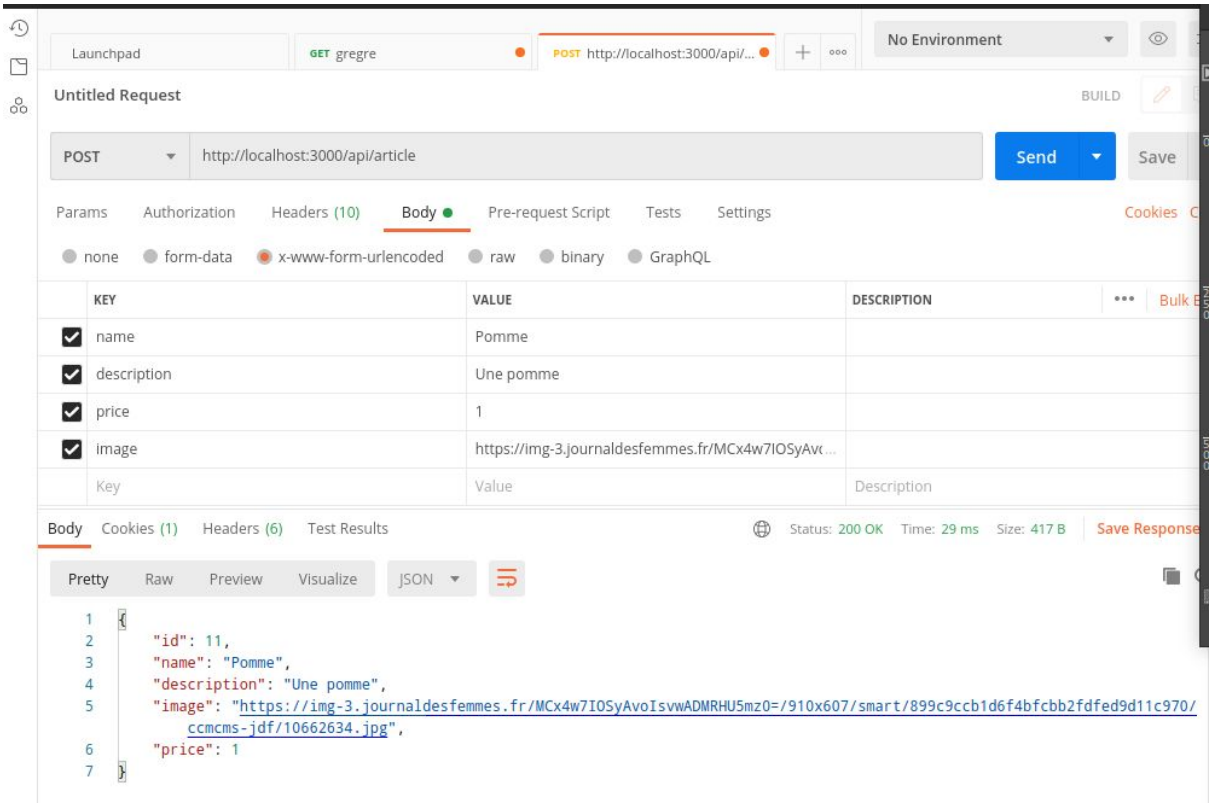
c) Ouvrez un nouvel onglet dans Postman avec le bouton "+" à droite de votre onglet actuel

d) Choisissez la méthode POST, l'url http://localhost:3000/api/article

e) Cliquez sur l'onglet "Body", choisissez le mode "x-www-form-urlencoded"

f) Ajoutez les champs suivants:
   i) name: Pomme
   ii) description: Une pomme
   iii) price: 1
   iv) image: https://img-3.journaldesfemmes.fr/MCx4w7IOSyAvoIsvwADMRHU5mz0=/910x607/smart/899c9ccb1d6f4bfcbb2fdfed9d11c970/ccmcms-jdf/10662634.jpg

g) Cliquez sur send, vous devriez arriver à ceci:



10) Vous savez maintenant vous servir de Postman, utilisez-le abondament dans les exercices suivants

11) Si l'une de ces étapes n'a pas fonctionné, demandez à votre responsable de TP.

Exercice 2: Serveur (🕒 1h30 pratique)

Note: il vous est vivement recommandé de regarder le code déjà rédigé disponible dans client/components/Home.vue et dans server/routes/api.js, afin de vous en inspirer

Le dossier **client/** contient:

- index.html: contient le template principal de l'application Vue
- vue-application.js: contient la racine de l'application, à savoir l'instance de Vue et un composant
- **components/**
  - **Home.vue: la page d'accueil du site**
  - **Panier.vue: la page contenant le panier de l'utilisateur**

Le dossier **server/** contient:

- app.js: initialise et configure le framework express
- data/articles.js: contient la liste des articles affichés sur le site
- routes/
  - api.js: contient toutes les routes de l'API
    - les articles
    - le panier

Le site que l'on vous a donné permet de gérer des articles d'un site marchand. L'objectif de ce TP est de rajouter la fonctionnalité de panier au site. Ouvrez le fichier server/routes/api.js

Vous pouvez constater dans le fichier server/routes/api.js qu'il y a à la fin du fichier toutes les routes relatives à la gestion des articles du site, et au début du fichier toutes les routes que vous devez implémenter.

Sachez également que le middleware défini à la ligne 25 du fichier api.js permet de créer un nouveau panier lorsqu'un utilisateur se connecte pour la première fois au site. Le panier est stocké dans req.session.panier

Pensez à lire les commentaires dans le code, ils sont là pour vous aider.

1) Implémentez la route GET /api/panier qui retourne le panier de l'utilisateur en cours, et vérifiez qu'elle fonctionne grâce à Postman

   a) Rappelez-vous que req.session contient uniquement les données liées à la session en cours, par définition

   b) N'oubliez pas de redémarrer votre serveur lorsque vous modifiez vos routes

2) Implémentez la route POST /api/panier, qui ajoute un article au panier

    a) Le contenu du tableau req.session.panier.articles est à votre discrétion, le plus pertinent est d'y mettre des objets qui contiennent deux propriétés id et quantity. Pour rappel, la liste des articles du site est disponible dans server/data/articles.js

    b) Vérifiez qu'un article ayant l'id demandé existe bien grâce au tableau articles

    c) Vérifiez que la quantité envoyée est un nombre entier strictement positif

    d) Retournez une erreur 400 (bad request) si l'article avait déjà été ajouté au panier

    e) N'oubliez pas d'aller regarder comment POST /api/article est implémentée, cela pourra vous aiguiller

    f) Utilisez systématiquement Postman pour vérifier que votre API fonctionne correctement

3) Implémentez la route PUT /api/panier/:articleId, qui modifie une quantité d'un article dans le panier.

    a) La route doit refuser un article qui n'a pas déjà été ajouté au panier

    b) La route doit refuser les valeurs négatives ou nulles

    c) Utilisez systématiquement Postman pour vérifier que votre API fonctionne correctement

4) Implémentez la route DELETE /api/panier/:articleId, qui supprime un article dans le panier

    a) La route doit refuser un article qui n'a pas déjà été ajouté au panier

    b) Utilisez systématiquement Postman pour vérifier que votre API fonctionne correctement

5) Implémentez la route POST /api/panier/pay, qui doit recevoir un prénom et un nom, supprimer le panier grâce à req.session.destroy(), puis retourner un message "Merci {{prénom}} {{nom}} pour votre achat"

    a) Utilisez systématiquement Postman pour vérifier que votre API fonctionne correctement

Félicitations, vous avez codé votre première API CRUD ! (Create Read Update Delete)

Exercice 3: Client (🕒 1h30 min pratique)

Nous allons maintenant coder la partie client de notre application Web.

Rappelez-vous que vous pouvez vous inspirer fortement du fichier client/components/Home.vue ainsi que de clients/vue-application.js qui contient des exemples d'appels à l'API /api/article (GET, POST, PUT, DELETE)

La syntaxe @click est un raccourci pour v-on:click, et :articles un raccourci pour v-bind:articles

*Pour rappel, utilisez sur VSCode le plugin **Vetur** afin d'activer la coloration syntaxique et d'autres fonctionnalités des fichiers .vue*

Explication globale

Voici ce que vous pouvez constater sur le code existant:

- Le code est réparti en trois parties:
    - **vue-application.js**: l'instance de vue s'occupe d'effectuer les appels à l'API, et de modifier le tableau this.articles
    - **Home.vue**: composant enfant de vue-application.js, contient les boutons et les champs de texte, et envoie des événements pour informer l'instance de vue des changements à opérer
    - **index.html**: contient <router-view>, qui correspond soit à Home.vue soit à Panier.vue en fonction du contexte
        - le **tableau articles** et le panier est envoyé comme **prop** à ces deux composants
        - les événements transmis par Home.vue sont interceptés grâce aux directives @add-article="addArticle", ce qui fait que lorsque Home.vue déclenche this.$emit('add-article', article), la fonction addArticle de l'instance de vue est appelée, avec article en paramètre
- La raison pour laquelle on découpe le code de cette façon est de permettre que le tableau *articles* soit **synchronisé entre toutes les parties de l'application**, et pas seulement dans Home.vue

1) Fonctionalité "Ajouter au panier"

a) Dans le composant Home.vue, ajoutez un bouton HTML "ajouter au panier", à droite de chaque article, à côté des boutons Supprimer et Modifier

b) Toujours dans Home.vue, Écrivez une méthode addToPanier(articleId) qui déclenche l'événement add-to-panier en transmettant l'id de l'article (reposez-vous sur la syntaxe de la méthode add-article)

c) Dans vue-application.js, sur le modèle de addArticle, concevez une méthode addToPanier(articleId) qui effectue la requête à l'API POST /api/panier afin d'ajouter l'article au panier

   i) Indiquez une quantité de 1

d) Afin de lier addToPanier de vue-application.js et addToPanier de Home.vue, il nous reste à rajouter dans index.html @add-to-panier="addToPanier" comme attribut de <router-view>

e) Maintenant que tout est branché, ouvrez l'onglet Réseau de la console de debug du navigateur, cliquez sur le bouton "ajouter au panier", et vérifiez que la requête est correctement effectuée

f) Avec l'extension navigateur Vue DevTools, vérifiez que panier.articles de <Root> contient bien votre nouvel article

2) Fonctionalité "Retirer du panier"

a) Dans vue-application.vue, décommentez les deux lignes à la fin de la fonction mounted(), pour récupérer l'état actuel du panier côté serveur au démarrage de l'application

b) Revenons dans Home.vue. Pour rappel, le panier est donné comme prop à Home.vue. Faites en sorte d'afficher un bouton "Retirer du panier" lorsqu'un article est déjà dans le panier, puis de cacher le bouton "Ajouter au panier" le cas échéant

c) En se basant sur ce que vous aviez réalisé pour la fonction addToPanier, effectuez un comportement similaire pour removeFromPanier(articleId)

   i) Cela inclut la méthode dans Home.vue, la méthode dans vue-application.js et le lien entre les deux dans index.html

3) Lister les éléments du panier

a) Le composant Panier.vue, qui s'affique lorsque l'url du site est http://localhost:3000/#/panier, est configuré pour recevoir le tableau article et le panier en props

b) Dans ce composant Panier.vue, affichez la liste des articles qui se trouvent dans le panier de l'utilisateur ainsi que la quantité voulue

   i) utilisez v-for

   ii) Comme le panier ne contient que les ids et les quantités, vous aurez à faire l'association entre les ids et les éléments dans le tableau articles

c) Sans recharger la page, allez sur la page d'accueil, ajoutez des nouveaux articles au panier, et vérifiez qu'ils se mettent correctement à jour dans votre page /paniers

d) Faites la même chose pour vérifier que la suppression se synchronise bien

4) Modifier la quantité

   a) Dans Panier.vue, permettez à l'utilisateur de modifier la quantité d'un article dans son panier

      i) Vous aurez à suivre la même chaîne d'opérations que pour l'ajout et la suppression, mais pour l'API PUT /panier/:articleId cette fois-ci

Exercice 4: CSS (🕒 30 min pratique)

Si vous avez eu le temps de réaliser tous les exercices précédents, rendez le site agréable à consulter avec du CSS, bootstrap, etc. Vous êtes libres du résultat