| | |
|---|---|
| **Commencé le** | mardi 31 mai 2022, 20:33 |
| **État** | Terminé |
| **Terminé le** | mardi 31 mai 2022, 20:37 |
| **Temps mis** | 4 min 11 s |
| **Note** | **6,00** sur 20,00 (**30**%) |

Description

**LISEZ CES INSTRUCTIONS AVANT DE COMMENCER L'EXAMEN**

Votre examen sera évalué seulement après avoir cliqué sur le bouton "Tout envoyer et terminer" et avoir confirmé la soumission.

Voici les règles pour l'évaluation:

- L'examen possède **5** questions notées sur un total de 20 points.
- La note partielle associée à chaque question est marquée à gauche de la question.
- L'espace disponible n'est pas nécessairement représentatif de la taille de la réponse attendue : **ne vous sentez pas obligés de remplir tout l'espace donné**.

**Vous avez une seule tentative pour l'examen final! Ne soumettez pas votre tentative à moins d'être 100% sûr(e) d'avoir terminé l'examen !**

En cas de doute sur le sens d'une question, faites une supposition raisonnable, énoncez-la clairement dans votre réponse et poursuivez. Une "question" vide est disponible sur la première page d'examen si vous avez besoin de plus de place ou pour des questions spécifiques.

Les téléphones et les ordinateurs portables ne sont pas permis.

**Les notes de cours peuvent être consultées dans Moodle, dans la section Ressources. Vous avez aussi droit à de la documentation papier (manuscrite ou imprimée).**

Bon travail et bon été!

Question **1**

Non répondue

Non noté

Cette question est un espace dédié pour vos commentaires ou questions sur le contrôle pratique. Aucune réponse ne sera donnée pendant l'examen.

Priorisez de mettre vos commentaires et/ou hypothèses directement dans la question spécifique.

Question **2**

Terminé

Note de 2,75 sur 5,00

a) React utilise un DOM Virtuel pour détecter les changements à apporter au DOM du navigateur. Donnez un avantage et inconvénient potentiels à l'utilisation d'un DOM Virtuel au lieu de manipuler le DOM natif directement. (2 point)

b) Voici l'implémentation d'un Component qui gère un état quelconque sous la forme d'un tableau (state). Pourquoi sommes-nous obligés de faire appel à setState dans chaque fonction du Component pour modifier la variable *state* ? (1 point)

```
1  const App = () => {
2    const [state, setState] = useState([]);
3
4    function deleteElement(selectedElement) {
5      setState(state.filter(x=>x.id !== selectedElement.id))
6     }
7
8    function addElement(newElement) {
9      setState([...state].push(newElement))
10   }
11
12   function replaceElement(oldElement, newElement) {
13     setState(state.map(x=> x.id !== oldElement.id ? x : newElement))
14   }
15   return (
16     <>
17       <ChildComponent elementModifier = {setState} />
18       <ChildComponent elementModifier = {addElement} />
19     </>
20   );
21 };
```

a)
Avantage de l'utilisation d'un DOM Virtuel : Performances améliorées : En utilisant un DOM virtuel, React peut minimiser les manipulations directes du DOM natif, ce qui réduit le coût en termes de performances. Les modifications sont d'abord apportées au DOM virtuel, puis React détermine les changements réels nécessaires avant de les appliquer au DOM natif.

Inconvénient potentiel de l'utilisation d'un DOM Virtuel : Overhead supplémentaire : La gestion d'un DOM virtuel ajoute une couche supplémentaire de complexité et peut entraîner un léger surcoût en termes de mémoire et de temps de traitement. Cependant, dans la plupart des cas, les avantages en termes de performances l'emportent sur cet inconvénient mineur.

c) Basé sur le code de b), quel est le danger potentiel d'assigner *setState* à la propriété *elementModifier* du component ChildComponent à la ligne 17 ? Pourquoi est-ce que l'utilisation de *addElement* à la ligne 18 est-elle à préférer ? Assumez que le but de ChildComponent est de modifier l'état de l'application. (2 points)

a)Avantage: Le DOM virtuel est la représentation de l'état du DOM à un moment spécifique, Permet de mettre à jour une partie de la page (une composante et ses descendants) Sans recharger la page et sans manipuler directement le DOM

Inconvénient: Le DOM Virtuel n'est pas une copie 1:1 du DOM réel : on copie que les nœuds du DOM ○ Il y a quand même un coût supplémentaire en mémoire utilisée pour cette copie ○ En réalité, il y a toujours 2 copies : le DOM Virtuel en cours et celui de l'état précédant

b)Les classes ont un attribut state qui gère l'état et est modifiable seulement à travers la méthode setState qui remplace l'ancien état avec le nouveau.

b) Nous devons utiliser setState pour modifier la variable state dans chaque fonction du composant parce que React doit être informé des changements d'état pour déclencher le rendu du composant. L'utilisation de setState notifie React que l'état a été mis à jour, ce qui entraîne une nouvelle exécution du rendu avec les données d'état mises à jour.

c) Le danger potentiel d'assigner setState à la propriété elementModifier du composant ChildComponent à la ligne 17 est que cela expose directement la fonction setState à l'intérieur du composant enfant. Cela pourrait conduire à des modifications d'état non contrôlées depuis le composant enfant, contournant ainsi le modèle de gestion d'état de React.

Préférer l'utilisation de addElement à la ligne 18 est une meilleure approche, car cela limite l'accès direct à la fonction setState dans le composant enfant. En passant des fonctions spécifiques (comme addElement) plutôt que la fonction générique setState, vous limitez la portée des modifications d'état possibles depuis le composant enfant, rendant le code plus prévisible et plus facile à maintenir.

Commentaire :

a) En quoi est-ce que c'est un avantage par rapport à la manipulation directe ? -1

OK pour le désavantage de cout supplémentaire

b) Nous avons un component fonctionnel et non une classe ici. OK pour le fait que c'est remplaceable seulement à travers la méthode setState -0.25

c) -2

Question **3**

Terminé

Note de 2,00 sur 5,00

a) Une requête GET vers facebook.com retourne une réponse avec la page web et l'en-tête suivant :

**cache-control : private, no-cache, no-store, must-revalidate**

Qu'est-ce que les valeurs de cet en-têtes représentent au niveau de la gestion du document et la cache du client ?

Quelle est l'avantage d'utiliser cette approche pour un site comme Facebook ? (2 points)

b) Pourquoi est-ce que les requêtes HTTP faites par un navigateur, que ça soit avec XHR ou Fetch, sont-elle toujours asynchrones ? (2 points)

c) Vous envoyez une requête HTTP avec l'en-tête suivant :

**Accept: text/html, text/plain;q=0.4, */*; q=0.1**

Le serveur vous retourne un document XML. Est-ce que ceci est une réponse acceptable ou il y a eu une erreur ? (1 point)

a) L'en-tête cache-control spécifie comment les informations de la page doivent être stockées, mises en cache et récupérées. Dans le contexte de la requête que vous avez mentionnée pour facebook.com, les valeurs de cet en-tête ont les significations suivantes :

private: Indique que la réponse est spécifique à un utilisateur et ne doit pas être mise en cache de manière partagée.
no-cache: Indique que le navigateur ne doit pas utiliser une copie en cache de la page sans la valider avec le serveur, même si elle semble être valide.
no-store: Indique que le navigateur ne doit pas stocker de copie en cache de la page.
must-revalidate: Indique que même si une copie en cache existe, elle doit être validée avec le serveur avant d'être utilisée.
L'utilisation de ces directives dans l'en-tête cache-control permet à Facebook de garantir que les informations sensibles et spécifiques à un utilisateur ne sont pas stockées de manière persistante sur les machines clientes. Cela renforce la confidentialité et la sécurité des données utilisateur, car le contenu n'est pas stocké localement de manière permanente et nécessite une validation régulière avec le serveur.

b) Les requêtes HTTP faites par un navigateur avec XHR (XMLHttpRequest) ou Fetch sont généralement asynchrones par défaut pour améliorer la réactivité de l'interface utilisateur. L'asynchronisme signifie que le navigateur n'attend pas la fin de la requête pour continuer à exécuter d'autres tâches.

L'avantage principal de l'asynchronisme est qu'il permet au navigateur de continuer à traiter d'autres opérations sans attendre la réponse du serveur. Si les requêtes étaient synchrones, le navigateur serait bloqué jusqu'à ce que la requête soit terminée, ce qui pourrait entraîner une expérience utilisateur lente et non réactive.

En utilisant des requêtes asynchrones, le navigateur peut déclencher plusieurs requêtes simultanément sans bloquer l'interface utilisateur, ce qui améliore l'efficacité et la réactivité globales de l'application web. Cela est particulièrement important dans les applications modernes qui effectuent de nombreuses opérations réseau, telles que le chargement de ressources, l'envoi de données au serveur, etc.

Commentaire :

a) Cet en-tête empeche la mise en cache de la réponse. L'avantage est que ceci force le client de toujours chercher la version la plus à jour du site web -2

b) OK

c) Il n'y a pas d'erreur, l'en-tête indique qu'il accepte n'importe quel document en dernier recours -1 Semaine 8, Page 44

Question **4**

Terminé

Note de 0,75 sur 4,00

a) L'application développée dans le cadre de votre TP5 implique l'utilisation d'au moins 3 serveurs différents. Quels sont ces serveurs et quel est le rôle de chacun ? (2 points)

b) Voici l'implémentation d'un gestionnaire de requête Express d'un service de connexion à son compte d'utilisateur. Assumez que la fonction *validate* est bien implémentée.
Quels sont les problèmes avec la gestion de cette requête et comment les corriger ? Justifiez votre réponse. (2 points)

```
1 app.get('/login/:username/:password', (req, res) => {
2     const user = req.params.username;
3     const password = req.params.password;
4     if (user && password) {
5         const isLoggedIn = validate(user, password);
6         res.send(isLoggedIn);
7     }
8     res.status(400).send(false);
9 });
```

a)lite-server , un serveur statique minimaliste qui permet d'avoir un déploiement local d'un serveur local qui fournit les fichiers sources.

b)On peut corriger les problèmes en ajoutant un objet (next) représentant le prochaine middleware à être exécuté dans la pile, on peut aussi définir un middleware qui prend comme premier argument un objet (err), qui sera défini si on a une situation d'erreur.

Commentaire :
a) On n'utilise pas lite-server pour le TP5, mais OK pour le serveur statique (0.75). Il manque les 2 autres serveurs : le serveur dynamique en NodeJS et le serveur de la base de données -1.25

b) 2 problèmes possibles :

- On envoie des informations sensibles dans un champ visible à tous (l'URI). Il faudrait utiliser une requête POST et non GET

- En cas de présence de user et password, le code va envoyer la réponse 2 fois (lignes 6 et 8), ce qui va causer une erreur -1

Question **5**

Terminé

Note de 0,00 sur 2,00

Votre base de données MongoDB contient des informations sur la population des différentes villes au Québec.

Chaque document possède un attribut "population" qui représente le nombre d'habitants de la ville.

Voici 2 manières d'obtenir toutes les villes dont la population est de 50 000 ou plus. Y a-t-il une manière qui est à privilégier ? Si oui, laquelle et pourquoi? Si non, pourquoi ? Justifiez votre réponse.

Manière #1 : const data = db.myCollection.find( { population: { $gte: 500000 } } ).toArray();

Manière #2 : const data = db.myCollection.find( { } ).toArray().filter( ( x )=> x.population >= 500000 );

Pour moi aucune des 2 manière est a privilégier puisqu' elles permettent de faire la même chose. Certain peuvent privilégier la manière 2 puisque find( { } ) permet la récupération de tous les documents ensuite toArray() Transformer le tout en tableau JS et puis filtrer la population dans le tableau. Tandis que d autres préfèrent de filtrer la population avant de tous transformer en tableau JS.

Commentaire :
La manière #1 est à préférer puisqu'elle fait le tri directement sur la BD. #2 force le transfert de toutes les données au client qui a fait la requête et qui doit faire le tri localement. Ceci est problématique si on a beaucoup de données sur la BD, mais très peu qui correspondent au critère de filtrage. -2

Question **6**

Terminé

Note de 0,50 sur 4,00

a) Votre collègue de TP voudrait transformer le serveur fait dans le cadre de votre TP5 en un système avec une architecture en micro-services.
Selon lui, il serait suffisant de séparer et déployer les 2 services (RecipesService et ContactsService) sur 2 serveurs HTTP différents.
Est-ce que ce changement est suffisant ou il faudrait faire autre chose pour avoir une architecture en micro-services ? (2 points)

b) Votre compagnie commence de travailler sur un jeu vidéo multijoueur accessible seulement à travers un site web. Vous êtes à l'étape de conception de l'architecture de votre système global.
Voici vos contraintes :
- Votre jeu doit supporter plusieurs joueurs en même temps sur des machines différentes.
- Afin d'éviter la triche, la logique du jeu doit être exécutée sur des machines autres que ceux des joueurs.
- Votre jeu est un jeu multijoueur seulement : vous n'avez pas à gérer un mode solo.

Vous avez le choix entre une architecture Client-Serveur et une architecture Pair-à-Pair. Laquelle serait plus adaptée pour ce projet ? Justifiez votre réponse. (2 points)

a)Les microservices priorisent une approche par produit plutôt que par projet, ce changement n est pas suffisant sachant que les microservices sont donc adaptés pour des petits projets ou des très grands projets qui peuvent être divisés en plusieurs fonctionnalités distinctes et aussi la communication entre les services peut rapidement devenir très compliquée.

b) Pour un jeu vidéo multijoueur accessible uniquement via un site web, une architecture Client-Serveur serait plus adaptée que l'architecture Pair-à-Pair, en particulier compte tenu des contraintes spécifiées.

Justification :

Logique du Jeu Centralisée : La nécessité d'éviter la triche en exécutant la logique du jeu sur des serveurs distants est mieux prise en charge par une architecture client-serveur. Dans un modèle client-serveur, la logique du jeu est centralisée sur le serveur, ce qui rend plus difficile pour les joueurs de manipuler le jeu et tricher.

Contrôle Centralisé : Pour gérer plusieurs joueurs en même temps sur des machines différentes, une architecture client-serveur offre un contrôle centralisé. Le serveur peut gérer l'état global du jeu, la coordination des actions des joueurs, la gestion des connexions, etc.

Gestion des Ressources : Avec un jeu multijoueur en ligne, la gestion des ressources (comme la synchronisation des données du jeu entre les joueurs) est plus facile à gérer de manière centralisée sur le serveur.

Évolutivité : Une architecture client-serveur permet une évolutivité plus aisée en ajoutant des ressources au serveur pour prendre en charge un nombre croissant de joueurs.

Commentaire :
a) Les services sont déjà assez spécifiques et il suffit de les découpler sur 2 serveurs différents. La 2e partie de la réponse n'est pas en lien avec la question -2

b) On ne fait pas de partages de fichiers et on veut explicitement éviter de faire certaines manipulations sur les machines des clients (joueurs). Ici, il faut utiliser une architecture Client-Serveur. -1.5

Points partiels pour avoir mentionné l'enjeu du grand nombre de clients en même temps.

Aller à…

Introduction ►