Master 1 informatique – Université Claude Bernard Lyon 1 – vendredi 22 janvier 2021

M1IF 03 – Conception d'applications Web – Examen

Durée : 1 heure 30 – Documents autorisés (4 pages max) – Ordinateurs, calculatrices, tablettes, téléphones portables… interdits

Questions de cours (barème : 18 points)

Toutes les questions ont au moins une bonne réponse.
Il faut cocher toutes les bonnes propositions d'une question pour avoir un point.
Si vous cochez une mauvaise proposition, vous perdez des points.
Remplissez au stylo noir ou bleu la case de la ou des bonne(s) réponse(s) ; une croix ne suffit pas.
Ne barrez pas une mauvaise réponse, mettez du blanc.
Ne redessinez pas une case que vous avez effacée, laissez blanc.

1. Pourquoi, dans l'API serveur de l'application que vous avez réalisée en TP, la réponse à une requête de création d'une salle ne devait-elle pas renvoyer un code de réponse 200 OK ?
   a) Le déploiement en HTTPS interdit de renvoyer des codes de réponse « simples » comme 200
   b) La réponse doit s'accompagner d'un header incompatible avec ce code de réponse
   c) Le client en SPA n'était pas autorisé à créer une salle
   d) L'effet de bord provoqué par la requête est mieux décrit par un autre code de réponse
   e) JWT interdit ce code de réponse pour l'envoi d'une méthode avec effet de bord

2. Pour que l'application déployée sur votre VM soit accessible en HTTPS, vous avez dû :
   a) modifier la configuration du load balancing dans nginx
   b) utiliser le certificat racine du département informatique
   c) installer un certificat spécifique sur votre VM
   d) installer un certificat spécifique à votre VM sur votre client
   e) modifier la configuration du connector de Tomcat

3. Lors de l'initialisation d'une connexion HTTPS :
   a) le serveur envoie une clé asymétrique différente pour chaque client
   b) le client envoie une clé privée au serveur
   c) le serveur envoie une clé symétrique au client
   d) le serveur est à l'origine de la demande de passage de HTTP à HTTPS
   e) le serveur envoie une clé publique au client

4. Une application Web de type SPA :
   a) peut être générée dynamiquement à partir d'une API
   b) permet un faible couplage avec une API
   c) peut être déployée sur un serveur séparé de l'API qu'elle interroge
   d) peut être amenée à enfreindre la same-origin policy
   e) peut agréger des données de plusieurs serveurs

5. Une application Web de type SPA telle que celle vue en TP :
   a) utilise un système de routage
   b) utilise un système de templating
   c) utilise un système de mise en forme
   d) utilise un système de réécriture de requêtes
   e) est générée avec des JSP

6. Une servlet :
   a) doit posséder au moins une méthode de service
   b) doit posséder au plus une méthode de service
   c) implémente le pattern client-serveur
   d) peut accéder au contexte applicatif

e) permet de répondre à une requête en AJAX

7. En MVC, une JSP :
   a) permet de récupérer les données du modèle à l'aide de beans
   b) permet de récupérer les données du modèle à l'aide d'attributs de requête
   c) permet de récupérer les données du modèle à l'aide de paramètres de requête
   d) permet de récupérer les données du modèle à l'aide d'attributs de session
   e) permet de récupérer les données du modèle à l'aide de paramètres de session

8. Un descripteur de déploiement web.xml :
   a) contient la liste des servlets de l'application
   b) permet de spécifier l'ordre des filtres
   c) permet de passer des paramètres aux JSP
   d) contient les noms des servlets dans le contexte applicatif
   e) permet de spécifier l'URL de base du serveur

9. Le header HTTP Content-Type :
   a) est présent dans une réponse dont le code est 200 OK
   b) est présent dans une requête dont la méthode est PUT
   c) peut contenir plusieurs valeurs
   d) est toujours accompagné d'un paramètre charset
   e) est présent dans une requête dont la méthode est GET

10. Le mécanisme CGI :
    a) permet d'ajouter facilement des informations sur les pages Web
    b) permet d'écrire sur la sortie standard
    c) implémente le pattern MVC
    d) permet de déployer des SPA
    e) permet à une application d'accéder aux paramètres de la requête

11. Un JSON Web Token (JWT) :
    a) permet d'encrypter des ressources
    b) permet d'identifier un utilisateur
    c) définit des variables de session
    d) doit être transmis dans un header HTTP
    e) ne peut être vérifié que par le serveur qui l'a émis

12. Le Web :
    a) se développe de manière anarchique
    b) est contrôlé par le W3C
    c) s'appuie sur la notion d'hypermédia
    d) favorise la scalabilité des applications
    e) est un réseau qui normalise l'interopérabilité entre clients et serveurs

13. La notion d'affordance est utilisée en Web pour :
    a) harmoniser la sémantique des liens
    b) favoriser la testabilité du code
    c) améliorer l'utilisabilité de l'interface
    d) automatiser le déploiement des applications
    e) rendre les utilisateurs résistants aux changements

14. AJAX permet :
    a) d'améliorer la sécurité des applications Web
    b) d'accélérer le chargement des ressources

c) de réduire la bande passante nécessaire pour exécuter une application Web
d) déporter du métier côté client
e) implémenter correctement le pattern Observateur

15. Un moteur de templating :
    a) permet la génération de la vue
    b) est implémenté en programmation descriptive
    c) peut contenir des structures de contrôle
    d) peut générer du texte plein
    e) s'appuie sur le pattern Arbre de Substitution

16. L'Optimisation du CRP d'une SPA :
    a) se fait principalement côté serveur
    b) consiste à réduire le nombre d'octets critiques
    c) se mesure à l'aide du DOM
    d) se mesure à l'aide du CDN
    e) permet de réduire les coûts côté client

17. Dans cette UE, vous avez utilisé l'approche DevOps pour :
    a) automatiser le mécanisme de déploiement
    b) mettre en place l'infrastructure de déploiement
    c) améliorer la scalabilité verticale
    d) améliorer la scalabilité horizontale
    e) optimiser la performance de votre application

18. Dans votre TP, si vous aviez eu un framework côté serveur, vous n'auriez pas eu à implémenter :
    a) le pattern AJAX
    b) le pattern MVC
    c) le templating des ressources
    d) le pattern Inversion de Contrôle
    e) la gestion des utilisateurs

Exercice (barème : 4 points)

Dans l'application de gestion des présences dans les salles que vous avez réalisée en TP, vous devez maintenant ajouter une requête, réservée aux administrateurs, qui permettra d'obtenir la liste de toutes les salles présentement saturées.

Pour rappel, l'objet `Salle` comporte entre autres les attributs : `String nomSalle` et `boolean saturee`, ainsi que les getters et setters correspondants.

19. Écrivez la spécification de l'API correspondant au traitement de cette requête : (2 pts)
    - URL
    - Description des entrées attendues (paramètres, headers)
    - Description des réponses possibles (Codes de retour, texte explicatif, contenu, headers)

    base_url/salles/saturees, GET, pas de paramètre, headers : Authorization -> bearer token, Accept -> type MIME, codes de réponse : 200 -> liste (éventuellement vide) d'URLs de salles, 401 -> non identifié, 403 -> non admin.
    0,25 pour chaque élément correct.

20. Écrivez la méthode de service du contrôleur MVC push-based qui répond à cette requête. (2 pts)
    prototype de la méthode doGet, récupération de la liste des salles (dans le contexte, variable d'instance…), extraction des salles saturées, utilisation du résultat (attribut de requête, templating JSON…)
    0,5 pour chaque.

On ne s'occupe pas ici de la négociation de contenus, ni des filtres d'authentification et d'autorisation.