Exercices – Création d'une application Cloud native

INTRODUCTION AU CLOUD NATIVE

1. Quels sont les avantages du cloud?
2. Quelles sont les 4 caractéristiques du cloud computing?
3. Quels sont les fournisseurs le plus célèbres du cloud computing?
4. Citer les trois types du cloud computing? la différence?
5. Définir les termes suivants: On-site, Iass, Paas, Saas
6. Définir la virtualization.
7. Quelle est la différence entre application monolithique et application cloud native?
8. L'approche développement cloud native a vu le jour suite à l'apparition du cloud computing, expliquer comment.
9. Pourquoi peut-on utiliser plusieurs technologies et systèmes hétérogènes au sein d'une même application cloud native?
10. Quels sont les 4 piliers de l'approche cloud native?
11. Quels sont les avantages de l'approche cloud native?
12. Comment l'approche DevOps permet aux développeurs de converger vers l'approche Cloud Native?
13. Définir le terme microservice. Détailler.
14. Définir le terme Conteneur dans le contexte d'une application Cloud Native
15. Citer les mots constituant le jargon du développement cloud native.

API REST

1. Quelle est la différence entre le méthodes GET et POST du protocole Http?
2. Quelle est la différence entre une API et une API REST ?
3. Comment créer un projet node ?
4. Quel est le rôle du fichier package.json?
5. Que fait la commande suivante? npm init
6. Que fait la commande nom start ?
7. Donner la commande pour installer Express dans le répertoire du projet
8. Donner la commande pour lancer le serveur Express
9. Comment créer un module en node js? Donner un exemple

10. Quelle est l'utilité de Postman? Détailler
11. On veut créer une API REST qui permettra de faire le crud sur une collection de modules (id, libellé, masse horaire) logée dans le fichier modules.json.
    a. Donner un exemple du fichier modules.json
    b. Donner le code du fichier index.js permettant de lancer le serveur web écoutant sur le port 6789
        i. Ajouter le code permettant de renvoyer la liste des modules si le client saisit l'url localhost:6789/modules avec la méthode cet
        ii. Modifier le code de la question précédente pour fixer le statut de réponse à 200.
        iii. Donner le code pour ajouter un module
        iv. Donner le code pour mettre à jour le libellé et la masse horaire d'un module via son id
        v. Donner le code pour supprimer un module étant donné son id
12. Définir le produit MongoDb. En quoi est-il différent des SGBDR ?
13. Quel est l'équivalent des tables en MongoDb ?
14. Que fait le code suivant ? commenter ligne par ligne

```
const MongoClient = require('mongodb').MongoClient;
const url = 'mongodb://localhost:27017';
const dbName = 'bdmonapi';
let db
MongoClient.connect(url, function(err, client) {
 console.log("Connexion réussi avec Mongo");
 db = client.db(dbName);
});
```

15. Idem pour ce code

```
app.get('/equipes', (req,res) => {
  db.collection('equipe').find({}).toArray(function(
err, docs) {
      if (err) {
          console.log(err)
          throw err
      }
      res.status(200).json(docs)
    })
})
```

16. Idem pour ce code :

```
app.get('/equipes/:id', async (req,res) => {
  const id = parseInt(req.params.id)
  try {
      const docs = await db.collection('equipe').find({id})
.toArray()
      res.status(200).json(docs)
  } catch (err) {
      console.log(err)
      throw err
  }
})
```

JWT

1. Qu'est-ce que le JWT ?
2. Quelle est son utilité ?
3. Décrire le principe de fonctionnement du JWT.
4. Que fait le code suivant ? expliquer ligne par ligne

```
L1      exports.login = (req, res, next) => {
L2        User.findOne({ email: req.body.email })
L3            .then(user => {
L4                if (!user) {
L5                    return res.status(401).json({ error: 'Utilisateur non trouvé !' });
L6                }
L7                bcrypt.compare(req.body.password, user.password)
L8                    .then(valid => {
L9                        if (!valid) {
L10                           return res.status(401).json({ error: 'Mot de passe incorrect !' });
L11                       }
L12                       res.status(200).json({
L13                           userId: user._id,
L14                           token: jwt.sign(
L15                               { userId: user._id },
L16                               'RANDOM_TOKEN_SECRET',
L17                               { expiresIn: '24h' }
```

```
L18                     )
L19                 });
L20             })
L21             .catch(error => res.status(500).json({ error }));
L22     })
L23     .catch(error => res.status(500).json({ error }));
L24 };
```

5. De même pour ce code :

```
L1  const userModel = require('../models/userModel');
L2  const jwt = require('jsonwebtoken');
L3  const isAuthenticated = async (req,res,next)=>{
L4      try { const {token} = req.cookies;
L5          if(!token){
L6              return next('Please login to access the data');
L7          }
L8          const verify = await jwt.verify(token,process.env.SECRET_KEY);
L9          req.user = await userModel.findById(verify.id);
L10         next();
L11     } catch (error) {
L12         return next(error);
L13     }
L14 }
L15 module.exports = isAuthenticated;
```

APPLICATION MICROSERVICE

1. Définir une application micro-services
2. Une application micro-service peut être développée en utilisant une seule technologie ?
3. On veut développer la gestion de la formation à l'OFPPT en utilisant l'approche cloud native. Proposez un découpage de l'application en micro-services.
4. Comment communiquent les micro-services entre eux ? et pourquoi ?

5. Comment une application cloud native peut réduire le time to market ?
6. Citer quelques inconvénients de l'architecture microservices.

CONTENEURS

1. Quelle est la différence entre machine virtuelle et conteneur ?
2. Quel est l'intérêt d'utiliser un conteneur ?
3. Quels sont les avantages de la conteneurisation ?
4. Expliquez la notion de mise à échelle.