**Qu’est ce que des Services Web Full Rest? Quelles sont les contraintes imposées par ce type de service?**

Un service web full REST est un service respectant toutes les bonnes pratiques de l'architecture "Representational State Transfer". Cette architecture consiste a utiliser une URI pour chaque action à effectuer, à laquelle on associe a une requette HTTP.

Une architecture RESTful est :

-sans etat : le client doit donc envoyer pour chaque requete toutes les informations necessaires au traitement de cette derniere.

-orienté client-serveur : la separations des responsabilités entre le client et le serveur permet une meilleure portabilité sur différentes plateformes et une meilleure extensibilité.

-cacheable : certaines reponses au requetes peuvent etre misent en memoire afin de pouvoir y re-acceder rapidement et sans renvoyer de requette.

-en couche : le client ne peut pas determiner si il est conecté au serveur final ou a un serveur intermediaire

-interface uniforme :

-code a la demande : le serveur peut envoyer du cote executable au client.

**Qu’est ce qu’un gestionnaire de dépendance? Maven est-il le seul? Quel est l’avantage d’utiliser un gestionnaire de dépendances? Quelles sont les grandes étapes de fonctionnement de Maven?**

Un gestionnaire de dependance permet d'etablir un ensemble logiciel a partir de differents modules/sources. Il existe d'autre gestionnaires de dependance que Maven, par exemple : Gradle, Jam, Rake, Bower...

L'avantage de ce type de logiciel est qu'il permet de simplifier l'implementation des nombreux modules utiles a la conception d'une api web suivant une structure conventionné.

Un projet Maven comporte un fichier pom.xml qui permet de decrire et configurer le projet et ses dependances. Il procede ensuite comme suit:

-Maven compile les fichier sources du projet (dont le chemin est specifié dans le POM)

-il compile les sources tests

-il les executes sur l'application (issue de la compilation des fichier sources)

-il genere un fichier jar de l'application si les tests n'ont retourné aucune erreurs.

**Qu’est ce qu’une application classique Entreprise Java Application? Donner un exemple d’usage avec web service, JSP, JavaBean,EJB et JMS.**

Une application classique Entreprise Java Application est un framework permettant entre autre d'etablir une architecture MVC sur un serveur web pour creer une API.

**Qu’est ce que Spring? qu’apporte Spring boot vis-à-vis de Spring?**

Spring est un framework proposants differents modules permettant d'etablir une architecture web. Cependant son utilisation est complexe et l'elaboration d'un architecture est laborieuse.

Springboot est un micro framework permettant de faciliter l'elaboration d'un projet Spring.

**Qu’est ce que SpringBoot? Quels sont les points communs/différences entre JEE et Spring Boot?**

Springboot et JEE sont tous les deux des framework utilisant java bien que JEE soit en réalité plutot une specification des outils java. Tout deux permettent dont de creer du web service.

Springboot est plus simple d'utilisation, ce dernier proposant plus de modules prefaits ainsi que des annotations.

**Qu’est cequ’une annotation? Quels apports présentent les Annotations?**

Les annotations, précédés d'une @, sont des metadonnées permettant de preciser le type d'une classe ou d'une methode.

Elles permettent d'orienter le comportement de la classe, méthode etc sur laquelle porte l'annotation.

**Comment fait-on pour créer un Web Service Rest avec Spring Boot ?**

Pour créer un Web Service REST avec Spring Boot, il est d’abord nécessaire de créer une application Spring Boot, c’est-à-dire une classe, avec l’annotation @SpringBootApplication et une méthode main au sein de laquelle va être exécutée l'application-même. N'ayant pas besoin de serveur, l'application s'exécute de façon autonome et affiche un message d’erreur par défaut puisque nous n’avons donné pas fais correspondre l’URL à un script (l’application n’a pas de mapping particulier). D’où l’intérêt d’ajouter un contrôleur REST. Pour ce faire, il suffit de créer une classe avec l’annotation @RestController. Au sein de cette dernière, on déclare les méthodes correspondant aux requêtes effectuées. Il existe différentes annotations de mapping en fonction des requêtes souhaitées (GET, POST, PUT, DELETE).

**Qu’est-ce qu’un container de Servlet ? Comment fonctionne un container de Servlet ?**

Un conteneur de Servlet qui est spécifique à Java, est un composant du serveur Web qui interagit avec les Servlets en gérant leur cycle de vie, mais également le mapping des URL vers le Servlet correspondant. Il permet de générer dynamiquement les pages Web du côté serveur. Une requête HTTP dynamique est reçue par le serveur, puis transmise au container. Le Servlet correspondant est récupéré et chargé dans le container qui va ensuite initialiser le Servlet. Puis, la requête HTTP est traitée, son contenu est lu et une réponse est formulée. Enfin, le serveur Web renvoie la réponse HTTP correspondant au résultat dynamiquement généré.

**Expliquer la philosophie « Convention over Configuration » de Spring boot ?**

La philosophie « Convention over Configuration » signifie que Spring Boot privilégie un faible nombre de décisions à prendre par le développeur. Ainsi, Spring Boot définit des configurations par défauts qui sont conventionnels, qui ne sont modifiées uniquement dans les cas non conventionnels. Cela permet de se consacrer davantage sur le résultat plutôt que sur la technique.

**Expliquer ce qu’il se passe lors de l’exécution «SpringApplication .run(App.class,args) »**

Ce code permet d’exécuter une application Spring. Ainsi, lors de l’exécution du code, l’application se lance, le conteneur de servlet s’initialise sur un port (ex : Tomcat sur le port 8080) et enfin on peut voir le résultat sur un navigateur à l’adresse [http://localhost/{num\_port}/URI](http://localhost/%7Bnum_port%7D/URI).

**Qu’est-ce qu’un DAO ? En quoi est-ce intéressant d’utiliser ce pattern ? Qu’est-ce qu’un Singleton ? Que permet de réaliser les Entity dans Spring boot ? Est-ce spécifique à SpringBoot ?**

Un DAO *(data access object)* est un modèle qui regroupe les accès aux données persistantes dans des classes à part, plutôt que de les disperser. Il est intéressant de l’utiliser puisqu’il permet une indépendance avec le moyen de stockage de données. Un singleton est un patron de conception qui restreint l’instanciation d’une classe à un seul objet. Les entity, quant à elles, sont des classes qui permettent de représenter des données qui peuvent être conservées dans la base de données. On peut trouver des Entity dans la JPA

**Combien d’instances avez-vous crées lors de l’usage de «Service » en Spring boot? Pourquoi ?**

Nous avons créé 2 instances de services : un pour UserService et l’autre pour CardService. Ces instances sont créées au lancement de l’application et sont injectés dans les différents RestControleurs qui utilisent ces services.

**Que fournit le CRUD Repository de Spring boot ? Que sont les CRUD ?**

Les CRUD permettent de manipuler un objet afin de lire ou modifier une donnée en base. L’acronyme CRUD pour create, read, update, delete. Cela fait référence aux requêtes possible sur la base de données. CrudRepository est une interface qui permet d’effectuer différentes actions sur les entity.

**Qui réalise l’implémentation de la méthode findByProperty lors de la création d’un repository en Spring Boot ?**

Spring implémente automatiquement la fonction findByProperty (entre autre) à partir du gestionnaire de base de données CRUD.

**Comment gère -t-on les relations One To One, One to Many et Many to Many avec JPA ?**

Les relations One to one, one to many et many to many sont gérées par l’ajout de tables à la base de données qui enregistre les liaisons entre nos différents objets (par exemple, le set de cartes d’un utilisateur).

**Qu’est-ce qu’une Architecture SOA ? Qu’est-ce qu’une architecture Micro Service ? Il y a-t-il d’autres architectures existantes ? Quels sont leurs avantages/inconvénients ?**

Une architecture SOA consiste à encapsuler un service qui renvoie uniquement une réponse au client. Les micro-services quant à eux, sont des services interconnectés via des interfaces (par exemple REST) ce qui permet une grande adaptabilité du service en cas de modification. La différence majeure est que l’on passe d’un système basé sur la relation client-serveur à une relation à plusieurs entités différentes.