Cours php

Définition d'un system d'information : acteurs/rôles +taches/actions +données

Urbanisation : moyen de faire évoluer les choses ,

Les 5 éléments du système d'information :

Stratégique : représente les différents métiers de l'entreprise. Elle a un impact sur tt les autres visions

Métier : la seule qui peut influencer la stratégique .tt les opérations qui se réalise au sein de l'entreprise -> processus métiers

Processus : ensemble des taches et opérations qui résulte un livrable.

Fonctionnelle: des taches qui s'inscrire dans plusieurs dimensions

SOA :

ESB :

Asynchrone message

Micro services :MSA

CICS:

MVS

IMS:moteur transactionnel

//TDs:

Bootstrap pour le design

- Page web : HTML
- Pour accéder a une page web il faut utiliser des URL/URI , et ce qui est important c'est le R
- DNS: représentation facile des adresses ip:port
- C quoi un protocole : permet de définir une grammaire, un langage pour communiquer au serveur
- Possibilité d'avoir plusieurs serveurs sur une même machine mais pas avec le même port
- Requête : un client fait appel à un serveur
- Fire Wall: sécuriser l'environnement/SI
- Load balancer: installation de l'app sur plusieurs serveurs avec la même manière
- Le premier service qui va écouter la requête c'est le web serveur (apache,…)
- L'index.php : c l'apache qui passe la main au php à travers le php.exe
- Url rewriting :réécriture du format de l'url pour qu'il soit compréhensible par le appache (passer des paramètres)

Micro-service :

Image docker : Une image Docker est un fichier, composé de plusieurs couches, utilisé pour exécuter du code dans un conteneur Docker. Une image est essentiellement construite à partir des instructions pour une version complète et exécutable d'une application, qui repose sur le noyau du système d'exploitation hôte. Lorsque l'utilisateur Docker exécute une image, elle devient une ou plusieurs instances de ce conteneur.

NB : Pas Tt les couches techniques du Windows sont importées dans l'image docker

Docker : un conteneur de conteneurs

Kubernetes:

Cartographie applicative :

Namespace :

Helm chart:

web farm

++

1-Un system d'information est un ensemble de ressources et processus qui permettent de collecter ,stocker ,analyser et diffuser les données(l'information)

2-CI: Continuous integration consiste a faire des mises a jours dans l'application et succède les tests(TU,TI,TF)

CD: Continuous Déploiement consiste a faire du déploiement immédiat après les mises a jours apporte dans la phase CI

3- Le terme « urbanisation » est utilisé par analogie avec les travaux d'architecture et d'urbanisme dans une ville en comparant une entreprise avec une ville et ses différents quartiers, zones et blocs.

Une telle démarche commence par le recensement et la capitalisation de l'ensemble des informations sur le système d'information de l'entreprise (bases de données, applications, services, etc.), en relation avec leur fonction, afin de les rationnaliser et de permettre de valoriser le capital informationnel de l'entreprise.

4-SOA :service oriented Architectur :Consiste a concevoir son system d'information sur une architecture qui délivre des applications fournissant des services

5-Middleware: est un logiciel qui comble les écarts entre d'autres applications, outils et bases de données afin de fournir des services unifiés aux utilisateurs.

Servant de logiciel de transition qui connecte les systèmes d'exploitation et les protocoles de communication, le middleware fonctionne comme suit:

- Déguisez un réseau disjoint et distribué
- Créer une homogénéité à partir d'une collection hétérogène d'applications logicielles
- Fournir aux développeurs une interface uniforme pour prendre en charge le développement, la convivialité et l'interopérabilité des applications
- Offrir un ensemble de services à usage général qui permettent aux applications de fonctionner ensemble et empêchent les systèmes de dupliquer les efforts

6-L'architecture SI urbanise

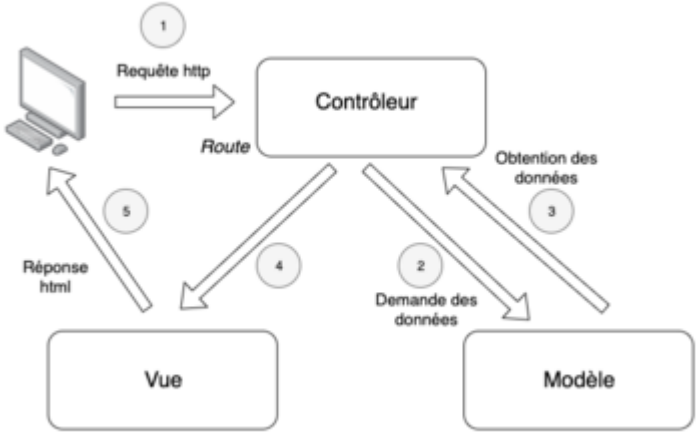
→ Métier/Stratégie : décision
→ Fonctionnelle
→ Applicative
→ Infrastructure Technique : matériel qui répond au besoin

Pour faciliter la planification des évolutions du Système d'Information, l'urbanisation s'appuie sur un plan d'urbanisme qui consiste à représenter le Système d'Information en s'appuyant sur une cartographie fonctionnelle et un découpage de plus en plus fin.

On distingue alors :

- La zone : elle représente un domaine fonctionnel
- Le quartier : il représente une opération à l'intérieur d'un métier
- L'îlot : il représente une application
- Le bloc fonctionnel : il représente la fonction.

7-Fonctionnement MVC



- **L'utilisateur envoie une requête http au serveur web**
- **La requête étant compose du Protocol //DNS :post ~~2~~=-departement&m=list**
- **Le serveur web interprète grâce au phpEngine la première page Index qui va rencontrer**
- **Dans cette page il y aura une sorte de routeur ou dispatcher qui va récupérer les paramètres de la requête pour identifier le contrôleur à appeler ainsi que la méthode associée (controllers/departementC.php)**

- **Le contrôleur alors va interroger le model à travers la méthode list() pour avoir la liste des département ,ensuite il va envoyer ces données à la vue(HTML) pour qu'elle soit affichée sur le navigateur du client**

SOA :

Interopérabilité des services - Les services doivent utiliser des normes qui permettent à divers abonnés d'utiliser le service.

Modularité: consiste à distinguer les fonctionnalités sous forme de bloc afin d'avoir des éléments plus indépendants possible tout en gardant un interconnexion transparente

La résilience: c'est la résistance au changement qui peuvent avoir lieu en cas par exemple de cession de l'entreprise a une autre qui a plus de charge en utilisant des services qui ne sont pas impactes par le volume

Réutilisation: Conçus comme des composants, les services peuvent être réutilisés plus efficacement, réduisant ainsi le temps de développement et les coûts associés.

Microservices:

un style de développement d'applications utilisant des services de petite envergure entièrement séparés et mutuellement exclusifs avec un contrôle décentralisé des données et des langues. Ce style de développement d'applications est principalement favorable à l'architecture d'applications distribuées.

Caractéristiques:

Parmi les caractéristiques d'un microservice c'est qu'il est indépendant dans la mesure ou il un faible couplage avec les autres composants comme ca il peut être évolutif et plus faciles à déployer et, comme ils sont autonomes, ils sont moins susceptibles de provoquer des défaillances du système en cas de problème. Aussi les MS sont orientes métier ;c'est à dire ils regroupent toutes les fonctionnalités lies a un métier :ex un microservice dédié a la gestion du panier dans un site e-Commerce, on peut également les dupliquer en fonction de la charge vu qu'ils peuvent être déployés dans des conteneurs(type Docker)qu'on peut ajouter ou supprimer.

La gestion de communication entre les MS se fait via un service REST, il y a donc peu de transformation

L'hétérogéniste des composants permet de bénéficier de la force liee au fait de tirer les profits de l'ensemble des technologies utilises dans chaque Ms distinct ,contrairement aux applications monolithes qui consiste a avoir un seul block de développement avec un seul langage pour faciliter après la maintenance du code ,mais le problème qui va se poser c'est gérer les fonctionnalités distinctes d'ou la force des MSs

Exemple microservices:

Par exemple, supposons que nous voulions créer une librairie en ligne qui permet aux utilisateurs enregistrés de soumettre des commandes, de vérifier et de mettre à jour l'inventaire et d'expédier des livres en utilisant la méthode la moins chère (si les livres sont disponibles). Cette application doit être accessible à l'aide d'une application mobile, d'une application monopage (SPA) côté client

(exécutée dans un navigateur) et directement via une interface REST, les interfaces mobile et SPA reposant sur l'API REST. En utilisant cette description, nous pouvons diviser notre système en trois services principaux, ainsi que leurs responsabilités respectives:

Service utilisateur: gère les utilisateurs enregistrés

Service d'inventaire: gère le stock de livres actuellement disponible

Service d'expédition: gère la recherche du service d'expédition le moins cher et l'expédition d'une commande

De même, nous pouvons classer trois interfaces principales dans notre système:

Application mobile: utilise l'interface REST via une passerelle / proxy API

SPA basé sur un navigateur: utilise l'interface REST via une passerelle / proxy API

Interface REST: une interface REST HTTP (Hypertext Markup Language) qui consomme et produit des ressources JSON (JavaScript Object Notation); pour notre implémentation de microservices, chaque service aura sa propre interface REST et une passerelle API sera utilisée pour composer ces API individuelles en une API REST apparemment singulière

Comparaison SOA vs MS:

SOA	**MS**
Suit une approche d'architecture «partager autant que possible»	Suit une approche d'architecture «partager aussi peu que possible»
L'importance est à la réutilisation des fonctionnalités de l'entreprise	L'importance est sur le concept de «contexte délimité»
Ils ont une gouvernance et des normes communes	Ils se concentrent sur les personnes, la collaboration et la liberté d'autres options
Utilise le bus de services d'entreprise (ESB) pour la communication	Système de messagerie simple
Maximise la réutilisabilité du service d'application	Se concentre sur le découplage
Un changement systématique nécessite de modifier le monolithe	Un changement systématique consiste à créer un nouveau service
DevOps / Continuous Delivery est en train de devenir populaire, mais pas encore grand public	Forte concentration sur DevOps /Continuous Delivery

Conclusion comparaison:

Au final, je dirai qu'il n'est pas si simple de dire quelle architecture est meilleure qu'une autre. Cela dépend principalement du but de l'application que’on crée. SOA est mieux adapté aux environnements d'applications commerciales grandes et complexes qui nécessitent une intégration avec de nombreuses applications hétérogènes; les petites applications ne sont pas adaptées à SOA, car elles n'ont pas besoin d'un composant middleware de messagerie. Les microservices, en revanche, conviennent mieux aux systèmes Web plus petits et bien partitionnés dans lesquels les microservices vous offrent un contrôle beaucoup plus important en tant que développeur. La conclusion est que, comme ils ont tous deux des caractéristiques d'architecture différentes, cela dépend principalement du but de l'application qu’on crée