Conception/Archi *API Web* / Client *JS*,
Génie Logiciel et Systèmes Informatique,
ISIMA 3ème année, 2017-2018

Conception et Architectures *API Web* / Client *JS*

Présentation des *TP* n° 1

Réalisation d'un projet : serveur *PHP* et *IHM JavaScript/HTML*

1 Principe Général du Projet

1.1 Notion de Parcours et Contraintes

Les travaux pratiques de cet enseignement visent à permettre, suivant un parcours choisi par les étudiants (généralement par binôme), de mettre en oeuvre la conception d'architectures clients/serveurs communiquant par requêtes asynchrones basées sur le protocole *HTTP* ou *HTTPS*. On pourra aussi mettre deux (ou plus) serveurs en réseaux (voir par exemple `http://www.php-fig.org/psr/psr-7/` pour la mise en oeuvre).

Les choix laissés aux étudiants portent sur le sujet du site, et donc sur le *métier*, mais aussi sur les technologies de mis en oeuvre. Des contraintes sont cependant imposées :

1. Indépendamment des choix faits pour les *TP*s, chaque étudiant doit avoir compris les principes d'architecture, relativement bas niveau et indépendants de tout *framework*, présentés dans le cours et sur les documents `http://www.malgouyres.org/programmation-php` et `http://www.malgouyres.org/programmation-javascript` (voir aussi `http://www.malgouyres.org/programmation-html-css`).
   En cas de doute, poser la question à l'Enseignant.

2. En particulier, le serveur doit implémenter la sécurité contre tous types d'injection *XSS* par des méthodes de filtrage systématiques ou des solutions robustes standardisées. Une politique raisonnable en matière de gestion des sessions et d'usurpation d'identité sera mis en oeuvre.
   En cas de doute, poser la question à l'Enseignant.

3. La conception doit comporter le lien entre un client *Web* où les modèles de données métier sont sous forme de structures de données *JavaScript*, et un serveur *HTTP* ou *HTTPS*, qui communiquent par requêtes asynchrones sur une *API Restful* implémentant les opérations *CRUD* sur les ressources métier, et où le serveur implémente la persistance via une base de données (généralement *SQL*).
   En cas de doute, poser la question à l'Enseignant.

4. Certaines contraintes supplémentaires sont imposées sur certains parcours pour limiter les disproportions dans la charge de travail.

1.2 Votre Serveur sur Machine Virtuelle (*VM*)

Vous disposerez, chacun, d'une machine virtuelle mise à disposition par l'*ISIMA* et accessible par son adresse *IP*. Vous pourrez ainsi configurer le serveur par *ssh* en utilisant vos identifiants habituels pour l'accès aux ressources informatiques de l'*UCA*.

Ce serveur sous *linux* n'est pas doté d'un serveur *X*, et l'administration d'*apache*, si vous en avez besoin, devra se faire en mode console avec *vi* ou *nano*. Le module *userdir* a été activé par défaut et votre *home* est monté, de sorte que vous pouvez commencer à mettre du code *PHP* dans votre répertoire `public_html`.

Pour qu'*apache*, certaines pemissions doivent être accordées sur votre home (`drwx--x--x`, soit `711`), sur votre `public_html` (`drwxr-xr-x`, soit `755`) et sur les sources *PHP* (`drw-r--r--`, soit `644`).

Lors de la manipulation des droits dans votre *home*, des vulnarabilités peuvent être créées. Vous pouvez limiter les risques en interdisant l'accès aux fichiers et sous-répertoires qui ne son pas directement concernés par l'accès au site *Web*. Faire `man chmod` et envisager `chmod` -R o=, ainsi que `chmod` -R g=.
En cas de doute, poser la question à l'Enseignant.

Le serveur *MySql* se gère via l'interface *Web* de *PhpMyAdmin* qui est acessible via l'*URI* *phpmyadmin* sur l'*URL* de votre *VM*. Vous pouvez créer, importer, exporter et configurer une base de données *MySql*. On s'identifiera en tant qu'utilisateur *MySql* `root` avec un mot de passe par défaut qui vous sera communiqué par l'enseignant.

Les identifiants de l'utilisateur *MySql* permettant au serveur *Web* étant inscrits en clair dans les sources (par exemple *PHP*) ouverts en lecture, vous devez prendre l'habitude de créer un utilisateur *MySql* distinct de l'utilisateur *MySql* `root` avec les droits nécessaires sur votre *BD*.
En cas de doute, poser la question à l'Enseignant.

Enfin, lors de la mise en oeuvre des la programmation du client *JavaScript*, ne pouvant évidement pas obfusquer votre code en cours de développement dans des *TP*s, celui-ci pourrait être copié. Il faudra dinc envisager une protection (par exemple authentification *HTTP* par `.htpasswd` dans la configuration d'*apache*. (voir `http://www.malgouyres.org/administration-reseau` pour des éléments de sécurité et configuration d'un serveur *apache*).

1.3 Démarche et Gestion de Projet

Le projet se réalise sur l'ensemble de l'enseignement qui compte 36 heures de temps présentiel.

1. La première semaine sera consacrée au choix du parcours, du sujet du site et à la réalisation d'un diagramme de *GANTT* prévisionnel.

2. La deuxième sera concentrée sur la modélisation des classes métier sur la base de *storyboards*, et à la réalisation d'un diagramme de classes métiers.

3. L'ordre de développement et le diagramme de *GANTT* dépendront fortement du parcours choisi, en particulier de la question de savoir si, par exemple, un *framework* ou un *ORM* sera utilisé côté serveur. L'ordre logique est dans tous les cas celui du développement par couches, dans l'ordre des chapitres de `http://www.malgouyres.org/programmation-php`, puis l'ordre des chapitre de `http://www.malgouyres.org/programmation-ja`

le code *PHP* et *JavaScript* ne devant jamais être mélangé dans un fichier source.
En cas de doute, poser la question à l'Enseignant.

4. La priorité sera donc d'avoir un module métier (fonctionnel et validé par tests unitaires), puis une couche d'accès aux données (fonctionnelle et validé par tests unitaires), encore avant d'avoir un ensemble de contrôleurs permettant les opérations *CRUD* (fonctionnels et validé par tests unitaires), et un ensemble de modèles à mettre en relation avec les vues (fonctionnels et validé par tests unitaires). Seulement alors, le développement côté client pourra commencer, suivant le même type de démarche par couche.
En cas de doute, poser la question à l'Enseignant.

2 Description des Parcours

2.1 Parcours *à la mano* du *SQL* à l'*IHM*

Dans ce parcours, l'ensemble de l'architecture est codé de manière bas-niveau, par exemple suivant les *patterns* suggérés dans `http://www.malgouyres.org/programmation-php` et de `http://www.malgouyres.org/programmation-javascript`.

Il sera dans ce cas compréhensible que les classes métier soient relativement simples, car ce mode de développement est plus long. On admettra aussi que la communication entre *API* côté serveur ne soit pas envisagée.

Notons que les *Patterns JavaScript* décrits dans les documents cités comportent une certaine généricité qui permet d'utiliser, en gros, le même code indépendamment du métier. Cependant, il faut rappeler que le but de ces documents est d'expliquer des *Patterns* de conception et la construction d'une architecture, et non pas uniquement la technique du copier-coller, dont on suppose qu'elle a été acquise en 1ère année.

Ce parcours est le plus adapté pour les étudiants qui veulent exploiter les *TP*s pour assimiler les notions d'architecture du cours, qui seront évaluées lors de l'examen écrit.
En cas de doute, poser la question à l'Enseignant.

2.2 Parcours *à la mano* côté serveur et *AngularJS* côté client

Il s'agit d'un parcours mixte, qui permet de mettre en oeuvre en détail les éléments d'architecture et de sécurité côté serveur, et d'aborder un *framework* côté client qui a le vent en poupe dans les entreprises.

Des variantes consistent à utiliser uniquement un *ORM* pour la persistance côté serveur, libérant du temps pour faire un accès à une *API* tierce, par exemple qui envoie des publicités à insérer dans les vues.

2.3 Parcours *FosRestBundle* côté serveur et *AngularJS*

Parcours au plus près de ce qui se pratique en entreprise, ce parcours demandera des prestations plus élevées (complexité du métier et/ou réseau d'*API* côté serveur) pour être équilibré en terme de charge de travail.

Ce parcours peut être un handicap pour l'examen écrit qui demande un compréhension intime des *Patterns* d'architecture.

2.4 Parcours *IHM* Côté client et Rendu *WebGL*

Dans ce parcours, le serveur *Resful* est minimaliste et on met le paquet sur une belle *IHM* côté client, sur la maîtrise de *JavaScript*, et sur les possibilités graphiques.

Ce parcours est exigeant sur le plan technique et n'est recommandé que pour les étudiants plutôt confiants, avec un goût pour l'aventure dans des territoires sauvages et difficiles d'accès.

2.5 Parcours Sécurité

Vous mettrez en oeuvre la sécurisation du site par *SSL* avec des certificats auto-signés (`http://www.malgouyres.org/administration-reseau`) et envisagerez des attaques de type *Cross Site Injections*...

2.6 Parcours libre

Vous concevrez vous-même votre parcours en piochant ce qui vous intéresse dans les parcours précédents...