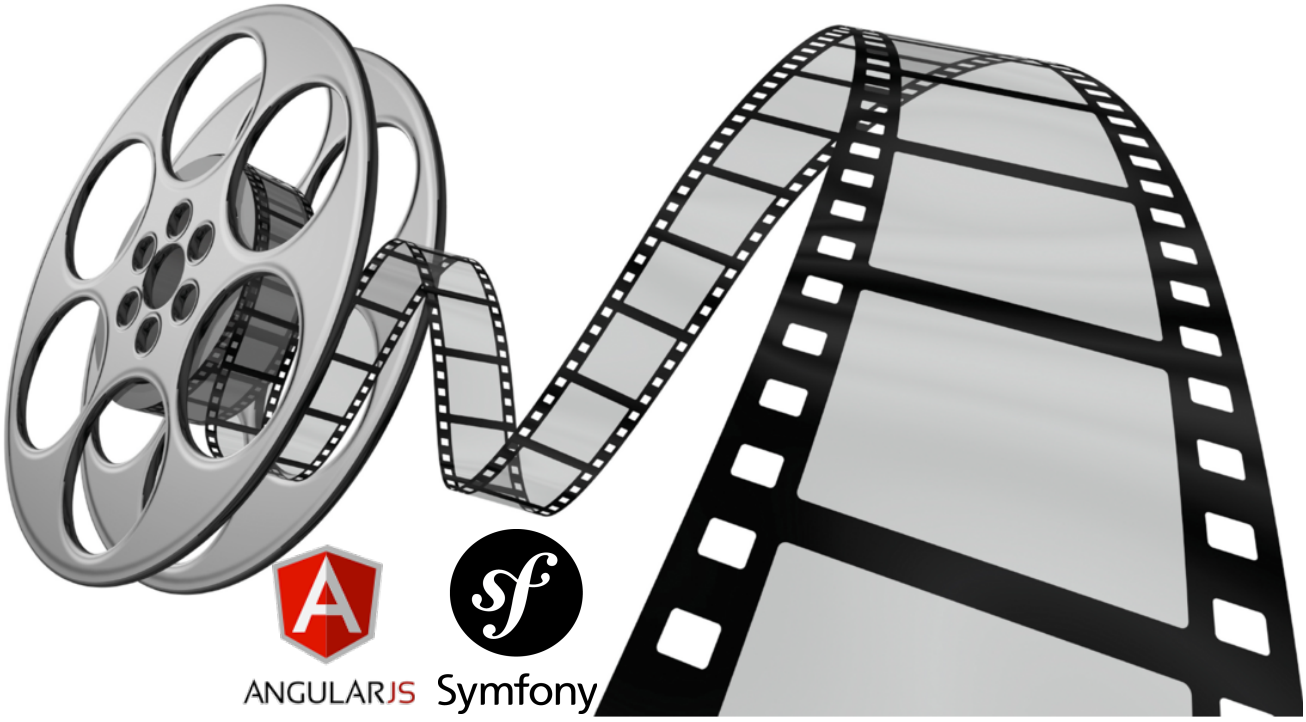
PROGRAMMATION D'APPLICATIONS WEB
CÉGEP DE SAINTE-FOY

Énoncé du Travail Pratique 3



Préparé par Guillaume Simard
http://atomrace.com

Résumé

Ce travail pratique consiste à réaliser des récits utilisateurs en mode « **fullstack** ». C'est à dire qu'il sera essentiel de développer le code client et le code serveur afin de livrer un récit utilisateur.

Le code serveur sera basé sur le cadriciel **Symfony 3**
et le code client sur **Angular JS 1.x**.

Ce travail doit être réalisé en équipe de 2 développeurs.

Les récits devront être accompagnés de tests automatisés.

1 Introduction au contexte du travail

L'application « Movies » du TP1 et du TP2 est un franc succès. Nous croyons qu'en ajoutant les fonctionnalités suivantes, nous pourrons augmenter le sentiment d'appartenance de nos utilisateurs et augmenter les conversions :

- Permettre à l'utilisateur de **téléverser une image** de profil.
- Mettre en place un système de **notification temps-réel des actions** des utilisateurs à l'aide de WebSocket.
- Permettre à l'utilisateur de **visualiser les dernières actions** sur le site (commentaire ajouté, nouveau favori, film regardé, utilisateur enregistré)
- Permettre à l'utilisateur **de devenir ami d'un utilisateur**.
- Permettre à un ami d'**afficher la liste des favoris de ses amis**.
- Permettre à l'utilisateur de **répondre au commentaire** d'un utilisateur.
- Permettre à l'utilisateur de **voter pour un commentaire**.
- Permettre à l'utilisateur de **filtrer les commentaires** et d'afficher ceux ayant le plus de votes.
- Permettre à l'utilisateur de **définir une cote (étoiles)** pour un film.
- Permettre à l'utilisateur de **modifier son profil**.
- Permettre à l'utilisateur d'**afficher les films les plus regardés**.
- Permettre à l'utilisateur d'**afficher les films les mieux cotés**.
- Permettre à l'utilisateur de visualiser tous les **commentaires récents**.

2 Développement *full-stack*

Puisque le temps de développement est limité, les récits utilisateurs seront définis et ensuite évalués en classe. Chaque équipe pourra ainsi sélectionner de 1 à 2 récits à réaliser. Les récits seront développés en mode « *full-stack* ». Il faudra donc programmer le côté client (Angular) ainsi que le côté serveur (Symfony).

Pour chaque récit utilisateur développé, on vous demande de fournir les tests unitaires et fonctionnels qui prouvent le bon fonctionnement.

3 Modification du code de l'API REST et fusion

Afin de réaliser ce travail, vous devez faire évolué le code de l'API REST utilisé dans le cadre du TP2. Il vous sera fourni au lancement du projet.

La première étape consiste à créer un « ***fork*** » du projet sur votre compte GitLab. Il vous sera ainsi possible de modifier le code et de le faire évoluer. Une fois la fonctionnalité testée, vous devrez procéder à un « ***merge request*** » afin que le responsable du projet puisse intégrer vos changements dans l'API REST.

La fusion du code pourrait être problématique si plusieurs équipes travaillent sur le même code. L'entité **User** devra être modifiée par plusieurs équipes. Il faudra donc une gestion serrée du code à ce niveau.

4 Conditions de réalisation du travail

Ce travail, d'une valeur de 15% de la note finale, s'effectue en équipe de deux.

Votre remise consiste à livrer un ou deux nouveaux récits avec le code client et serveur correspondant.

Côté client : Une directive Angular JS réutilisable avec tests unitaires.

Côté serveur : Un service REST (contrôleur, entité(s), *repository*, *fixtures* et tests).

5 Récits utilisateur du catalogue produit

Pour ce travail pratique, les récits utilisateur devront être rédigés par les membres de l'équipe. Chaque **récit devra comporter les 3 sections** (En tant que ... je veux ... afin de). De plus, les récits devront être accompagnés de **critères d'acceptation**. Une fois le travail réalisé, vous devrez **mettre à jour le catalogue produit** et le présenter au professeur.

6 Évaluation

Éléments	Pondération

• **Catalogue produit**
  ○ Qualité des récits utilisateur et des critères d'acceptation
  ○ Français soigné
  ○ Structure et critères complets
  ○ Estimation réaliste
	15%

• **Qualité du code client et serveur**
  ○ Utilisation des patrons de conception
  ○ Bien découpé
  ○ Bien indenté
  ○ Code clair et simple
  ○ Fonctionnel
Client
• Une directive est développée afin d'encapsuler la logique et la vue.
• Un service est utilisé afin d'appeler l'API REST.
• .constant() est utilisé pour définir les constantes (adresses API).
Serveur
• Les requêtes SQL sont encapsulées dans un *repository*.
• Les *fixtures* sont complètes.
• La documentation de l'API est mise à jour avec ApiDoc().
	40%

• **Qualité des tests**
  ○ Données de tests suffisantes (*fixtures*)
  ○ Tests suffisants
  ○ Teste la fiabilité et la robustesse
	30%

• **Déploiement du client sur Heroku (https)**
  ○ Adresse du déploiement fournie
  ○ Code CSS minimifié
  ○ Code JS minimifié
  ○ Nouvelle directive fonctionnelle
• ***Merge request* sur GitLab**
  ○ Le *merge request* a été réalisé et accepté
	15%

Pour chaque faute de français, 0,5% sera soustrait de la note finale (pour un maximum de 20%).

7 Résumé des étapes importantes du projet

Serveur - API REST avec Symfony

1. Fournir le nom d'utilisateur (courriel) de votre compte GitLab au chargé de projet.
2. Créer un *fork* du projet appXapi : https://gitlab.com/ifocus22/appxapi
3. Configurer votre environnement local :
    1. Installer les dépendances avec Composer.
    2. Configurer la base de données avec tables et données de tests.
    3. Exécuter les tests avec PHPUnit.
4. Créer une nouvelle branche pour votre récit utilisateur avec GIT.
5. Modéliser la route de votre service API REST (chemin et paramètres d'entrées et de sortie)
6. Écrire le code serveur PHP (entité, *repository*, *fixtures*, tests, contrôleur, routes)
    1. Si de nouveaux *bundles* Symfony doivent être installés, confirmer le tout avec le chargé de projet.
7. Assurez-vous d'avoir la dernière version du code source. (git pull)
8. Assurez-vous que tous les tests passent.
9. Créer un *merge request* sur GitLab avec le nom de l'équipe et une description du récit.
10. Votre code sera fusionné au projet principal par le chargé de projet.
11. Si tous les tests passent, le code sera fusionné et déployé sur Heroku par le chargé de projet.

Client - Angular JS

1. Écrire la nouvelle directive avec un nom pertinent et un préfixe unique.
2. Écrire le code (contrôleur, accès au service de l'API REST et vue).
3. Écrire les tests pour la directive.
4. Incorporer la nouvelle directive dans votre projet du TP2 et déployer sur Heroku.

8 Documents à réaliser et à remettre

Veuillez remettre sur « LÉA » une archive « zip » contenant les fichiers du projet.

- Fichier **Backlog_tp3.xls** complété et corrigé.
  - Lien vers le dépôt de code pour Symfony.
  - Lien vers l'application côté client déployée.
- Code client (sans les dépendances).
- Code serveur (sans les dépendances).

Bon succès!