Développement d'applications web

Projet

Vincent Simonet, 2013-2014

Introduction

Ce document définit le projet que vous devez réaliser dans le cadre du cours "Développement d'applications web (Architecture des Applications Réticulaires - AAR)". Les parties en noir spécifient des exigences que vous devez respecter. *Les parties en bleu italique sont des recommendations laissées à votre libre appréciation.*

Modalités pratiques

Groupes

Le projet doit être réalisé par groupe de **2 ou 3** étudiants. Lorsque votre groupe est constitué, merci de renseigner le tableau de suivi avec vos noms.

Originalité

L'ensemble du code du projet doit être écrit par les étudiants du groupe. L'usage de librairies publiques est bien sûr autorisé.

Temps de travail

Des périodes seront réservées lors des cours hebdomadaires pour la réalisation du projet (en particulier pour avancer en groupe et pour discuter de points techniques avec moi). Le reste de la réalisation du projet doit être effectué sur le temps de travail personnel.

Soumission

Les projets doivent être soumis au plus tard **la veille du dernier cours** (c'est-à-dire le dimanche 12 janvier). Les soumissions en retard seront pénalisées ou ignorées. Chaque soumission doit consister :

- D'une présentation d'au plus 6 diapositives (hors page de titre) présentant la réalisation (produit final et aspects techniques),
- D'un lien vers le code source du projet,
- D'un lien vers l'application en ligne réalisée.

Veillez à ce que ces éléments soient accessibles en lecture aux autres étudiants et à moi !
Lors du dernier cours, vous disposerez de 10 minutes pour présenter votre projet (5 minutes de présentation et 5 minutes de démo). Préparez votre présentation comme un *pitch* pour mettre en valeur votre travail et le produit réalisé. Évitez de vous apesantir sur les aspects communs à tous les projets (comme le fait que vous avez un serveur et un client...). Soyez originaux :)

Évaluation

Le projet sera évalué par une note sur 20, commune aux étudiants du groupe, et déterminée par l'enseignant. Le but du projet est de valider votre capacité à développer une application web de bout en bout, en tirant bénéfice des compétences acquises en cours. L'évaluation sera donc effectuée à la fois en boîte noire (on regarde le produit fini sans se préoccuper de la manière dont il a été réalisé), et en boîte blanche (on regarde à l'intérieur du produit pour regarder la qualité de la réalisation).
Le barême suivant sera suivi :

- Boîte noire :

    - La fonctionnalité du produit (le projet est un jeu complet qui permet de réaliser les fonctionnalités de base décrites dans la section "But du projet") : **5 points**.
    - La beauté et l'originalité du produit (les pages sont claires et agréables, les règles du jeu sont intéressantes et variées, etc.) : **2 points**.
    - Le bon fonctionnement de l'application (les performances sont raisonnables, il n'y a pas de bug visible de l'utilisateur, etc.) : **3 points**.
- Boîte blanche :
    - La satisfaction des exigences techniques (voir la section "Exigences techniques") : **4 points**,
    - La qualité et la documentation du code : **3 points**,
    - Le respect des règles de base du développement d'une application web (pas de faille de sécurité majeure, répartition des traitements entre le client et le serveur, bonne prise en compte des cas d'erreur, etc.) : **3 points**.

Moyens

Vous pouvez utiliser les moyens informatiques de l'Université ou vos moyens personnels pour la réalisation du projet. L'application finale doit être disponible en ligne.

Description du projet

But du projet

Le but du projet est de réaliser une application web permettant à des joueurs de réaliser des paris sportifs. Chaque joueur dispose d'un compte sur l'application, et d'un solde de points. Lorsque le joueur se connecte à l'application, une liste de rencontres sportives se déroulant dans les jours à venir lui est présentée. Le joueur peut choisir une ou plusieurs de ces rencontres, et faire des paris sur le résultat de ces rencontres. Une fois que la rencontre est terminée, l'application détermine si le pari du joueur est réussi, et attribue le cas échéant des points au joueur. L'application présente également au joueur la liste de ses paris réalisés ou en cours, et les gains ou pertes de points associés.

Vous êtes libres de choisir le ou les sports sur lesquels le joueur pourra effectuer des paris (e.g. Football, Tennis, Basketball, Formule 1, etc.), ainsi que la nature des paris que le joueur peut formuler (nom du vainqueur, score, temps, etc.). Vous êtes également libres de déterminer le système de points de votre jeu (e.g. le nombre de points gagnés pour un pari réussi, ou le nombre de points retirés pour un pari perdu).

Vous devez cependant faire des choix qui rendent le jeu intéressant (e.g. nombre suffisant de rencontres, système de points compréhensible, etc.)

Exigences techniques

Vous devez respecter les exigences techniques suivantes lors de la réalisation de votre projet :

- L'application doit être réalisée suivant une architecture client / serveur web.
- La partie serveur doit être réalisée en utilisant Java Servlet et éventuellement JSP. Elle doit être hébergée en ligne. *Vous pouvez par exemple utiliser Google App Engine ou Amazon Web Services qui proposent tous deux des services gratuits. Cet article donne une comparaison intéressante. Vous pouvez également configurer les logiciels sur la machine de votre choix (du moment qu'elle est accessible en ligne).*
- La partie client doit être réalisée en utilisant HTML, CSS et JavaScript. Elle doit fonctionner sur les versions récentes d'au moins deux navigateurs Web *(typiquement Firefox et Chrome)*, quel que soit le système d'exploitation. Le support des anciens navigateurs ou d'Internet Explorer n'est pas une exigence.

- L'application doit faire appel à au moins une API Web Service externe *(e.g. ESPN, voir ci-dessous)*, et respecter ses règles d'utilisation.
- Le client de votre application doit faire un appel AJAX ou JSONP au serveur pour au moins une fonctionnalité.
- L'application doit permettre à plusieurs joueurs de jouer en parallèle.

Les points suivants sont volontairement laissés à votre libre appréciation:

- *L'utilisation (ou non) d'un framework pour le serveur (type Google App Engine, Spring ou Struts),*
- *L'utilisation (ou non) d'un framework JavaScript (type Dojo, jQuery ou Google Closure),*
- *Le choix du protocole de communication entre les parties client et serveur de votre application,*
- *Le choix de l'API WebService fournissant les plannings de rencontres et les résultats,*
- *Le choix de la technologie de stockage sur le serveur (base de données).*

Exigences techniques optionnelles

Selon du temps dont vous disposez, et de votre rapidité d'avancement, vous pouvez améliorer votre application en ajoutant des fonctionnalités techniques, comme par exemple :

- Utilisation du protocole HTTPS,
- Envoi de mails aux joueurs pour les informer des résultats de leurs paris,
- Bon fonctionnement de plusieurs sessions en parallèle pour un même utilisateur,
- Mise en oeuvre de techniques de base pour assurer l'efficacité de l'application (minification du JavaScript et/ou du CSS, sprites, etc.),
- Une version mobile (HTML ou native).

Vous pouvez également enrichir votre application d'un point de vue fonctionnel (plus de sports, paris plus élaborés, classement des joueurs, etc.)

Recommendations

Organisation du projet

Avant de commencer à coder

Avant de commencer à coder, je vous recommande de :

- *Choisir l'API WebService que vous allez utiliser, de vous familiariser avec son protocole et de bien comprendre les données qu'elle fournit.*
- *Choisir et configurer le système d'hébergement que vous allez utiliser.*
- *De définir la règle de base de votre jeu (quels sports / quelles compétitions, quels types de paris, quel système de point).*
- *De définir l'architecture d'ensemble de votre projet (quels sont les principaux composants les leurs fonctionnalités, quelles sont les techniques utilisées, etc.).*

Écrivez un petit document partagé entre vous fixant ces différents choix. Vous pourrez me demander mon avis sur ce document lors des séances de travaux dirigés.

*Dans tous les cas, je vous conseille vivement de **commencer par une idée simple**, en particulier au niveau de la règle du jeu (un type de compétitions qui a suffisamment de rencontres pour être intéressant, un type de pari simple, etc.). Une fois que vous aurez une première version opérationnelle, il sera plus simple d'ajouter des choses plus compliquées. Il faut mieux réaliser un jeu simple qui soit pleinement opérationnel, qu'un jeu compliqué à moitié fini.*

Implémentation

- *Répartissez vous les rôles dans la réalisation du projet, et définissez clairement les interfaces entre vous.*

- *Créez un repository partagé pour vous permettre d'échanger facilement le code (type CVS, SVN ou GIT),*
- *Visez d'avoir rapidement une première application qui tourne rapidement, de manière à avoir une base commune de travail, et itérez pour ajouter des fonctionnalités.*

Choix de l'API WebService externe

Comme indiqué ci-dessus, vous êtes libre de choisir l'API WebService vous fournissant les plannings de rencontres sportives et les résultats. Je vous recommande cependant l'utilisation de l'API ESPN qui donne accès de manière gratuite à un large choix de sports et de compétitions.

La documentation de cette API est disponible ici. Notez que pour utiliser l'API, vous devez créer un compte sur le site et obtenir une clef d'identification pour votre application. La version gratuite vous donne droit à 3 appels par seconde et 7500 appels par jour, ce qui doit être amplement suffisant pour votre projet. Veillez cependant à respecter ces limites, en particulier si vous faites des appels en série au niveau du serveur.

L'API ESPN est purement RESTful (requêtes GET avec les arguments dans l'URL), et les réponses peuvent être retournées en JSON (par défaut), JSONP (ajouter `&_accept=application/json &callback=mycallback` à l'URL) ou XML (ajouter `&_accept=text/xml` à l'URL).

Pour vous familiariser avec l'API, vous pouvez directement faire des requêtes en saisissant l'URL (avec votre clef) dans votre navigateur. Pour lire facilement le contenu des réponses JSON, je vous conseille d'installer un plugin dans votre navigateur (par exemple pour Firefox ou Chrome).

Les principales requêtes qui devraient vous intéresser sont :

- *`/v1/sports`, pour connaître la liste des sports supportés par l'API,*
- *`/v1/sports/<sport>`, pour obtenir la liste des compétitions d'un sport donné,*
- *`/v1/sports/<sport>/<competition>/events`, pour obtenir la liste des recontres passées ou à venir pour une compétition.*

Notez que les dates et heures sont données au format ISO 8601, en temps universel (UTC).

La clef d'identification doit être incluse dans toutes les requêtes. A priori vous devez garder cette clef sur le serveur, car si les utilisateurs en ont connaissance, ils pourraient faire des requêtes utilisant votre quota à votre insu. Cela signifie que les appels à l'API doivent être réalisés au niveau du serveur (il y a plusieurs autres bonnes raisons pour cela d'ailleurs !).

Google App Engine

Pour vous familiariser avec Google App Engine :

- *Lire la page "What is Google App Engine?",*
- *Lire la série de pages "Getting Started" pour Java,*
- *Vous pouvez également faire le tutoriel complet (Codelab).*

Amazon Web Services

Pour vous familiariser avec Amazon Web Services :

- *Lire la page "Niveau d'utilisation gratuit d'AWS",*
- *Lire la page "Qu'est-ce que le niveau d'utilisation gratuite d'AWS" et le reste du tutoriel.*