**Projet JEE – AppShop**

**Introduction :**

Ce projet a été développé par deux étudiants de Licence Pro Développement d’application Internet/Intranet : SOUQUIERE Alexis & FOUGEROUSE Audrey

**Architecture :**

* La couche de persistance des données du projet utilise JPA.
* La couche logique métier du projet utilise les EJB3.
* La couche de présentation utilise à la fois les Servlet/JSP et JSF.

Enfin, le projet est composé de deux Web Service :

* Un web service Soap de validation de carte bancaire.
* Un web service Restful de gestion de commentaire (ajout, suppression, modification, consultation -> CRUD).

**Fonctionnalités du projet :**

* **Accès aux données**

Pour l’accès aux données, nous avons développé une couche applicative d’accès aux données basée sur JPA. Nous avons-nous même conçu et administré cette base de données en nous basant sur celle présenté lors des TP.

De plus, l’accès aux données se fait par l’utilisation d’une DataSource à l’aide d’un pool de connexion JDBC précédemment configuré.

* **Logique Applicative**

La couche applicative contient la logique métier de l’application est réalisée à l’aide d’EJB.

Dans cette couche nous avons ajouté une contrainte de sécurité, pour cela, nous avons créé un domaine de sécurité dans GlassFish.

Ceci permet de gérer, et limiter les accès à certaines parties de notre application. En effet, certaines méthodes sont autorisé que pour une certaine catégorie d’utilisateur (ex : administrateur). Par exemple, la méthode de suppression d’application est autorisée que pour un administrateur.

De plus, notre application permet à un utilisateur de se logguer. Ainsi en fonction de ses droits d’accès, un utilisateur aura accès à toutes ou partie des pages de notre application. Ainsi, un administrateur peut accéder au « Backend » de l’application, alors qu’un simple utilisateur ne possédant pas de droit particulier aura seulement accès au « Frotend » de notre application.

Par ailleurs, notre couche applicative comporte des méthodes plus ou moins complexe, en effet nous pouvons par exemple trouver une méthode qui nous permet de calculer tous les lundis, le nombre de téléchargement d’une application durant la semaine écoulée.

Ou bien encore, une méthode qui permet d’alerter un utilisateur lorsqu’une de ses applications propose une mise à jour. Pour satisfaire ce besoin, nous avons décidé de créer un système de message. Lorsqu’une application est créée, si celle-ci existée déjà avant (avec le même nom et même plaforme) alors c’est que c’est une nouvelle version. Dans ce cas on créer un nouveau message, que l’on fait afficher sur la page d’information du membre (qui a télécharger une ancienne version) afin qu’il soit au courant de la sortie d’une nouvelle version.

Enfin, notre couche applicative, étant le cœur de l’application et possédant un certain nombre de méthodes complexe, nous avons choisi de réaliser des tests avec « hsqldb » afin de valider l’intégrité de nos méthodes.

* **Présentation Web**

Note pour ce loguer :

Administrateur : LOGIN 🡪 admin MDPS 🡪 admin

Membre : LOGIN 🡪 member MDPS 🡪 member

Notre application Web, ce décompose en deux parties : Frotend et Backend.

Elle propose à l’utilisateur, 2 langues d’affichage : Français / Anglais.

La couche présentation de notre application est principalement basée sur JSF. Cependant, l’application possède aussi deux uses cases réalisés en utilisant la technologie Servlet+JSP. Nous avons choisi les uses cases:

* Un admin ajoute une application dans le catalogue
* Un membre du site télécharge une application

Ces use cases contiennent des actions JSP, de la JSTL de l’expression language (EL) et au un Tag custom (fragment JSP). Ce Tag, permet l’affichage de statistique de téléchargement d’une application. En effet, il permet d’afficher le nombre de téléchargement total qu’il y a eu pour l’application. Le nombre d’utilisateur qui l’on télécharger (un utilisateur pouvant télécharger plusieurs fois l’application), ainsi que la liste des utilisateurs qui ont téléchargé l’application).

* + **Fronted**

Cette partie de l’application est accessible à n’importe qu’elle utilisateur, il n’y a pas besoin de droit d’accès particulier, elle est accessible sans être logué.

Dans le fronted, l’utilisateur à accès à :

* L’index, la page d’accueil ou l’utilisateur peut consulter les 5 dernières applications ajoutées ainsi que les 5 derniers commentaires ajoutés.

Grace à la barre de menu accessible via n’importe quelle page, l’utilisateur peut :

* Accéder au catalogue des applications
  + En cliquant sur le nom d’une application, une page avec la différente version existante s’affiche, ainsi en cliquant sur une version, l’utilisateur peut consulter les détails de l’application, la noter ajouter un commentaire ou la télécharger.
  + Si l’application est gratuite alors l’utilisateur peut la télécharger et ainsi accéder aux informations de la page de téléchargement avec les statistiques (tag+ servlet)
  + En revanche si l’application est payante, l’utilisateur ne pourra pas la télécharger et verra afficher l’erreur « Le serveur a envoyé le code de statut HTTP 401 : Unauthorized »
* Accéder à la liste des platformes
  + En cliquant sur le nom d’une platforme, il peut voir la liste des applications disponible sur la platforme.
* Accéder à la liste des Editeurs
  + En cliquant sur le nom d’un éditeur, il peut voir la liste des applications créé par cet éditeur.
* Accéder à son compte.
  + Si l’utilisateur n’est pas loguer, il peut créer un compte ou alors ce loguer.
* Rechercher une application avec différent critère de filtrage.

**En plus des fonctionnalités précédentes, un utilisateur loguer (Membre) peut :**

* Consulter ses messages (ex : Alertes en cas de nouvelles versions applications)
* Gérer son compte (modifier infos + mots de passe + supprimer)
* Gérer ses collections : ajouter/ supprimer/ éditer (modifier nom de la collection) + ajouter/supprimer des applications dans la collection.
* Voir ses téléchargements
* Se déconnecter
  + **BackEnd**

Cette partie de l’application, concerne tout ce qui est accessible pour un utilisateur possédant les droits d’administration.

L’administrateur peut :

* Consulter la liste des applications, les éditer (infos), les supprimer ou bien ajouter une nouvelle application.
* Consulter la liste des plateformes, les éditer (infos), les supprimer ou bien ajouter une nouvelle plateforme.
* Consulter la liste des éditeurs, les éditer (infos), les supprimer ou bien ajouter une nouvel éditeur.
* Consulter les achats effectués.
* Gérer les utilisateurs (éditer les informations, ajouter, supprimer un utilisateur)
* Voir les statistiques
* **Interopérabilité**
  + **Web Service SOAP**

Ce service SOAP, permet de simuler l’accès à une banque afin de valider la saisi d’une carte bancaire. L’algorithme de validation de la carte est très simple, une carte est validée si elle comporte 16 caractères.

Ce service web est appelé lorsque l’utilisateur clique sur le bouton de téléchargement de l’application qu’il consulte, ainsi, si le prix de cette application est supérieur à 0, nous affichons une Template qui permet à l’utilisateur de saisir les numéros de sa carte bancaire.

Ainsi, lorsqu’il clique sur le bouton d’achat, notre application fait appel au service SOAP afin de valider ou non la carte bancaire.

* + **Web Service Restful**

Ce Service REST, permet d’exposer le système de commentaire de l’application en respectant l’architecture RESTful.

Avec ce service REST, on peut alors :

Afficher la liste des commentaires + filtre par id de commentaire, par id d’utilisateur ou bien par id d’application.

De plus, on peut aussi poster, modifier et supprimer un commentaire.

Pour ce service, le format d’exposition/consommation des ressources est JSON.

(Voir « Test WSRESTCommentaire.doc » pour tester le service REST dans notre application).

* **Divers**
* L’affichage des messages d’alerte de mise à jour de nouvelle version d’application ce fait qu’une fois le navigateur fermer puis ré ouvert.

Vu que le UserManagedBean à un scope session cela fait qu'il persiste tant que l'utilisateur ne ferme pas son navigateur. Donc l'EJB persiste aussi du début jusqu'à la fin et celui si n'est pas rafraîchi.

Soit pour voir les messages, l’utilisateur doit avoir fermé et ré ouvert son navigateur afin de rafraichir l’EJB après que l’admin est créer une application.

* Problème sur le changement de langue. Celui-ci ne se fait pas de page en page, lorsque l’utilisateur change de langue (ex anglais), et qu’il change de page, la langue par défaut français et afficher. Si l’utilisateur veut afficher en anglais il doit repasser la page en anglais.