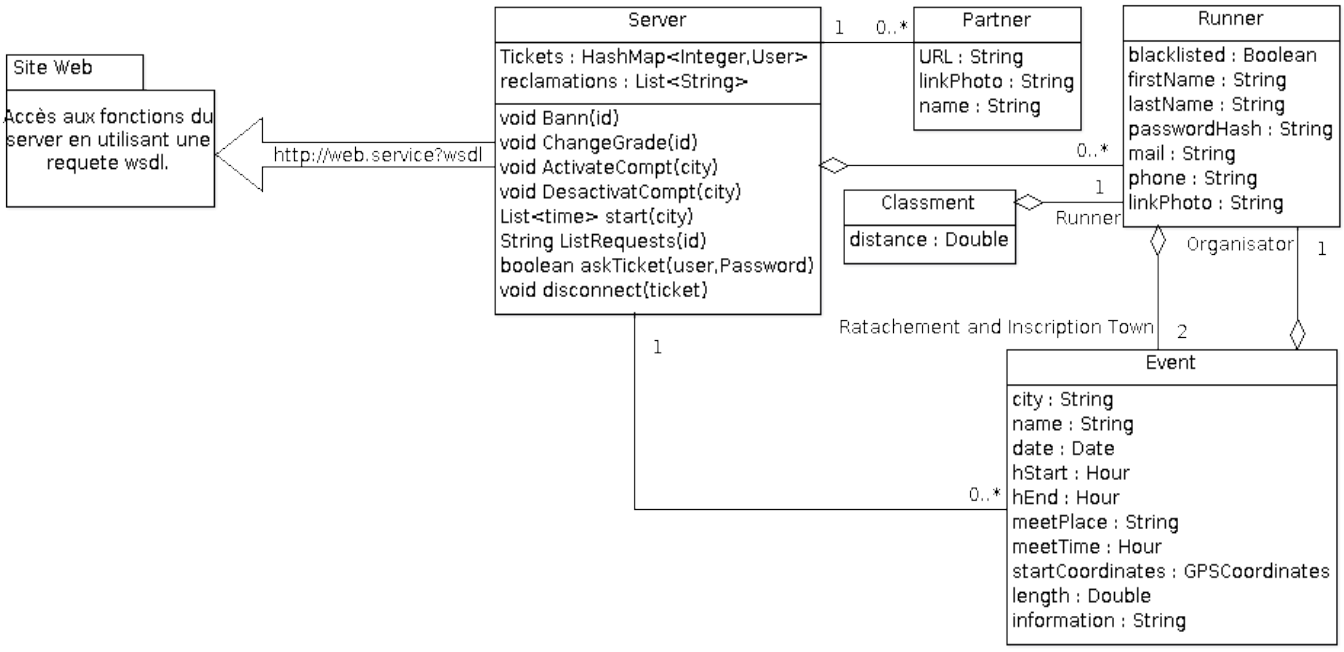
Architecture Client-Server

Le programme fonctionne via une page web effectuant des requêtes XML au web-service de l'application fournit par le server.



Ces requêtes s'effectuent sur la machine du client via le site web. Des fonctions JavaScript sont utilisées à cet effet. Ainsi c'est la machine du client qui émule ces fonctions et fait appel au server afin d'obtenir les informations et d'effectuer les actions désirées.

Les fonctions disponibles sont celles de l'objet Server et sont runnées sur le server lorsqu'une requête est effectuée et celles-ci renvoient les résultats au client.

Plan de tests

1 - Tests boite noire

Les tests boite noire s'effectuent sans connaissance obligatoires du fonctionnement interne de l'application et du serveur.

C'est l'application qui fait appel aux fonctions du web-service. Il suffit donc de vérifier que l'utilisation des différentes méthodes du serveur ne permettent pas de sortir du cadre normal d'utilisation.

Tests Boite Noire		
Nom	Objectif	Remarques
Test Performance	Vérifier si le serveur réagit correctement à un grand nombre de connexions en parallèle	Le but n'est pas d'atteindre les limites du programme, mais de tester une moyenne de connexions
Test Bannissement	Vérifier si un utilisateur bannit peut encore interagir avec le serveur.	
Test Grade	Vérifier si un utilisateur a accès à des fonctionnalités réservées à des grades supérieurs.	
Test Compteur	Vérifier si les compteurs s'arrêtent ou continuent de tourner selon si on les active ou non.	
Test déconnexion	Vérifier si un ticket est toujours utilisable après déconnexion ou reconnexion du même utilisateur.	Lorsqu'un utilisateur se reconnecte sans se déconnecter, son ancien ticket disparaît.
Test requête	Vérifier si les requêtes sont bien envoyées aux utilisateurs.	

2 - Tests boite blanche

Les tests boite blanche s'effectuent en interne du serveur. Il s'agit de vérifier si les préconditions, les postconditions et les invariants des différentes fonctions disponibles sont respectés.

Tests Boite Blanche		
Nom	Objectif	Remarques
Test Unicité	Vérifier si les valeurs des tickets sont uniques et si ils pointent chacun un utilisateur différent.	
Test Admin Désactivé	Vérifier s'il est impossible de désactiver l'admin.	Admin unique, et non supprimable.
Test Demande Ticket	Vérifier si un ticket demandé est différent de ceux présents et si ils ne réfèrent qu'à un seul user.	
Test déconnexion	Vérifier si un ticket est supprimé de la liste après déconnexion ou reconnexion d'un utilisateur.	Lorsqu'un utilisateur se reconnecte sans se déconnecter, son ancien ticket disparaît.

3 - Cahier de tests

3.1 - Pre-Requis et Environnement de tests

La plupart des tests s'effectuent directement sur le server. En effet, le site web se contente de faire appel aux fonctions via le web service.

Aussi faut-il avoir implémenter la base de données ainsi que le service de publication du web-service.

Les tests tests boite noir correspondant au côtés client peuvent s'effectuer directement sur navigateur web et les tests tests boite blanche correspondant au côté server s'effectuent directement dans l'environnement de développement Eclipse.

Identificateur	Description
<Navigator>	Navigateur web sur ordinateur ou mobile
<Eclipse>	IDE Eclipse

3.2 - Éléments à tester

La plupart des objets du server étant issus de la base de donnée, et le seul élément accessible étant l'objet Server, il suffira de vérifier que les limitations de la base de donnée soient respectées. Mais il faudra porter un attention tout particulière à l'objet Server.

Identificateur	Description
<Server>	Objet Server.
<Site>	Site Web

3.3 - Cas de test

Cas de test	Site-01		
Titre	Test Performance		
Objectif	Vérifier si le server peut supporter un grand nombre de connexions en parallèle.		
Spécification du test	Test Performance		
Élément à tester	File d'attente du web-service.		
ID	Entrées	Sorties	Ok ?
1	Le site web	Un compteur de réponses.	
Restauration			
Critères de succès/échec			Approbation
800 connexions simultanées sont simulées en un laps de temps raisonnable.			
Commentaires :	Ne possédant pas 800 machines, nous allons éteindre le server et laisser tourner le web-service en remplissant la file d'attente de 800 requêtes. Ensuite nous allumerons le service afin d'émuler les connexions simultanées.		

Cas de test	Site-02		
Titre	Test Bannissement		
Objectif	Vérifier si le bannissement empêche l'utilisateur d'utiliser le service.		
Spécification du test	Test Fonctionnel		
Élément à tester	N'importe-quelle requête du service.		
ID	Entrées	Sorties	Ok ?
1	Site admin	Un indicateur de bannissement	
2	Site client	Erreur	
Restauration	Modifier l'architecture du server.		
Critères de succès/échec			Approbation
Un client bannit n'a aucune fonctions proposées			
Si le client est bannit pendant sa connexion, les fonctions ne sont plus disponibles.			
Commentaires :	Nous vérifieront si lorsque l'administrateur place un coureur en bannis, est-ce que le coureur peut continuer d'utiliser les fonctions du server.		

Cas de test	Site-03		
Titre	Test Grade		
Objectif	Vérifier si les fonctions disponibles certains types d'utilisateurs sont utilisables par les autres		
Spécification du test	Test Fonctionnel		
Élément à tester	N'importe-quelle requête du service.		
ID	Entrées	Sorties	Ok ?
1	Le site web	Page d'erreur	
Restauration	Modifier l'architecture du server.		
Critères de succès/échec			Approbation
Les utilisateurs ne peuvent utiliser des fonctions qui ne sont pas de leur grade.			
Commentaires :	Similaire au test Site-02.		

Cas de test	Site-04		
Titre	Test Compteur		
Objectif	Vérifier le comptage des compteurs		
Spécification du test	Test Fonctionnel		
Élément à tester	Compteurs		
ID	Entrées	Sorties	Ok ?
1		Valeur compteur	
Restauration	Modifier le système de comptage.		
Critères de succès/échec			Approbation
Un compteur éteint ne tourne pas et inversement.			
Commentaires :			

Cas de test	Site-05		
Titre	Test Déconnexion		
Objectif	Vérifier si un utilisateur a toujours accès aux fonctions après déconnexion.		
Spécification du test	Test Fonctionnel		
Élément à tester	N'importe-quelle requête du server.		
ID	Entrées	Sorties	Ok ?
1	Site web	Erreur	
Restauration	Modifier le système de gestion des connexions.		
Critères de succès/échec			Approbation
Les fonctions sont indisponibles.			
Commentaires :			

Cas de test	Site-06		
Titre	Test Requête		
Objectif	Vérifier si les requête sont bien renvoyées au bons utilisateurs.		
Spécification du test	Test Sécurité		
Élément à tester	File d'attente du web-service.		
ID	Entrées	Sorties	Ok ?
1	Requête client	Réponse server	
Restauration	Modifier le système de gestion des requêtes.		
Critères de succès/échec			Approbation
Un client reçoit bien les requêtes qui lui sont destinées.			
Commentaires :	Similaire au test Site-01		

Cas de test	Serv-01		
Titre	Test Unicité		
Objectif	Vérifier si chaque ticket est unique et si un utilisateur n'est pointé que par un seul ticket.		
Spécification du test	Test Sécurité		
Élément à tester	Server.tickets		
ID	Entrées	Sorties	Ok ?
1	HashMap<Integer,User>		
Restauration	Modifier le système de gestion des tickets.		
Critères de succès/échec			Approbation
Le ticket est unique.			
Un client n'a pas plusieurs tickets.			
Commentaires :			

Cas de test	Serv-02		
Titre	Test Admin Désactivé		
Objectif	Verifier que l'admin est unique et n'est pas supprimable.		
Spécification du test	Test Sécurité		
Élément à tester	Server.User		
ID	Entrées	Sorties	Ok ?
1			
Restauration	Modifier le système de gestion des utilisateurs.		
Critères de succès/échec			Approbation
Admin non supprimable.			
Admin unique.			
Commentaires :			

Cas de test	Serv-03		
Titre	Test Déconnexion		
Objectif	Vérifier la suppression des tickets des utilisateurs déconnectés et reconnectés.		
Spécification du test	Test Sécurité		
Élément à tester	Server.User		
ID	Entrées	Sorties	Ok ?
1	Déconnexion	HashMap<Integer,User>	
2	Reconnexion	HashMap<Integer,User>	
Restauration	Modifier le système de gestion des tickets.		
Critères de succès/échec			Approbation
Ticket supprimé en cas déconnexion.			
Ticket supprimé en cas reconnexion.			
Commentaires :			