



MakeUrSport



Cahier de tests

Suivi d'activités sportives sur Android

Sommaire

Tests unitaires p5-7

Tests d'intégration p8-9

Tests de validation p10-11

Introduction

Ce document contient l'ensemble des tests du projet, en décrivant pour chaque test les résultats obtenus et la validité du test. Les tests unitaires permettent d'assurer l'avancement du projet en étant sûr que les méthodes sont bien codées. Les tests d'intégration testent la bonne intégration de chaque fonctionnalité à l'application. Enfin, les tests de validation évaluent le respect des besoins de notre client M. Broisin.

Tests unitaires

Les tests unitaires ont tous été effectués le 19 janvier 2013, en utilisant l'outil JUnit de l'environnement de développement Eclipse.

Classe Sportif

On instancie un nouveau sportif né en 1993, mesurant 183 cm et pesant 70kg.

Méthode `getAge()`

Scénario : On teste que l'âge de ce sportif est bien égal à 20.

Résultat : 20. **Test valide.** 

Classe Course

On instancie une nouvelle course en initialisant l'état en « course lancée » et le début de la course à l'heure actuelle.

Méthode `getDuree()`

Scénario 1 : On attend 3 secondes, puis on met en pause la course pendant 4 secondes. On la reprend pendant 5 secondes, puis on l'arrête. On vérifie alors que la durée de la course est bien de 8 secondes.

Résultat : 4. **Test non valide.** 

Anomalie : il semblerait que le temps de pause ait été retranché deux fois puisqu'on obtient 4 secondes de moins que prévu. On cherche alors si c'est effectivement le cas puis on relance le test.

Scénario 2 : Le temps de pause était effectivement retranché une fois de trop. On relance alors le même test avec les mêmes valeurs.

Résultat : 8. **Test valide.** 

Méthode `getVitesseMoyenne()`

Scénario : On attribue à la course une durée de 45 minutes et une distance de 8 km. On teste la vitesse moyenne de la course, qui doit être de 10,67 km/h.

Résultat : 10,67. **Test valide.** 

Méthode getCaloriesBrulees()

Scénario 1 : On appelle la méthode avec un MET de 7, un poids de 70 kg et une durée d'une heure. On vérifie que le résultat est de 514 calories.

Résultat : 513764. **Test non valide.** 

Anomalie : le résultat est environ 1000 fois plus grand que le résultat attendu, il doit donc y avoir une erreur dans les calculs où on prend en compte une mauvaise unité.

Scénario 2 : On effectue le même test après modification du code.

Résultat : 514. **Test valide.** 


Méthode addTempsPause()

Scénario : On attribue à la course un temps de pause de 30 secondes, puis on lui ajoute 90 secondes de pause. On vérifie ensuite que le temps de pause de la course est bien égal à 120 secondes.

Résultat : 120. **Test valide.** 

Méthode sauvegarderCourse()

Scénario : On instancie une nouvelle course sans la démarrer. Ses attributs sont donc initialisés à 0. On enregistre cette course avec la méthode, puis on vérifie que les attributs de la course sauvegardée sont bien égaux à 0 en la sélectionnant.

Résultat : Tous les attributs sont égaux à 0. **Test valide.** 

Classe GestionnaireCourse

On instancie un nouveau GestionnaireCourse pour tester les méthodes permettant le démarrage, la gestion des pauses et l'arrêt d'une course.


Méthode demarrerCourse()

Scénario : On appelle la méthode, puis on teste l'état de la course en vérifiant qu'il soit en CourseLancee. On stocke ses attributs, puis on attend 5 secondes. On compare alors les attributs de la course aux deux moments, qui doivent être différents.

Résultat : Etat CourseLancee et attributs différents. **Test valide.** 


Méthode mettreEnPause()

Scénario : On démarre une nouvelle course, puis on la met en pause. On vérifie alors que son état est en CourseEnPause. On stocke ses attributs, on attend 5 secondes, puis on vérifie qu'ils sont égaux aux deux moments.

Résultat : Etat CourseEnPause et attributs égaux. **Test valide.** 


Méthode reprendreCourse()

Scénario : Après avoir lancé une nouvelle course, on la met en pause, puis on appelle la méthode. On effectue alors exactement le même test que pour la méthode demarrerCourse().

Résultat : Etat CourseLancee et attributs différents. **Test valide.** 

Méthode stopperCourse()

Scénario : On arrête une course après l'avoir lancée. On teste ensuite que son état est bien en CourseArretee, puis on stocke ses attributs. 5 secondes plus tard, on vérifie que les attributs sont égaux aux deux moments.

Résultat : Etat CourseArretee et attributs égaux. **Test valide.** 

Classe GestionnaireHistorique

On instancie un nouveau GestionnaireHistorique ainsi qu'une Course dont on va se servir pour les tests.

Méthode selectionnerCourse()

Scénario : Après appel de cette méthode, on vérifie que l'identifiant de la course que l'on veut sélectionner est bien égal à celui de la course sélectionnée.

Résultat : Identifiants égaux. **Test valide.** 

Méthode supprimerCourse()

Scénario : On stocke l'identifiant de la course, puis on l'enregistre dans la base de données. On la supprime ensuite avec cette méthode, puis on exécute une requête SQLite de sélection pour l'id de la course. Cette requête ne doit retourner aucun résultat.


Résultat : null. **Test valide.** 

Tests d'intégration


Les tests d'intégration ont été effectués le dimanche 20 janvier avec pour smartphone de test un Google Galaxy Nexus, qui est un smartphone utilisé par les développeurs Android car il ne présente aucune surcouche d'Android.

L'enregistrement d'une course dans l'historique se fait correctement

Scénario 1 : On effectue une nouvelle course, puis on la met en pause. On note ensuite les informations affichées avant de l'arrêter. Enfin on va consulter l'historique et vérifier que les informations de la course enregistrée correspondent à celles notées.

Résultat : Informations identiques. **Test valide.** 

Scénario 2 : On réitère le même scénario mais après avoir générer le parcours.

Résultat : Informations identiques. **Test valide.** 

La suppression d'une course dans l'historique fonctionne

Scénario : On lance une nouvelle course, puis on l'arrête pour l'enregistrer. On consulte alors cette course dans l'historique pour la supprimer. On vérifie ensuite que la course a bien été supprimée de l'historique.

Résultat : Course supprimée. **Test valide.** 

Le choix du sport est bien pris en compte

Scénario : On choisit comme sport le cyclisme, on lance une nouvelle course et on l'arrête pour l'enregistrer. On vérifie alors que le sport de la course en registrée est bien le cyclisme.

Résultat : Cyclisme. **Test valide.** 


Le tracé d'un parcours généré s'affiche sur la carte

Scénario : On génère un parcours de 10 km et on vérifie qu'un parcours s'affiche bien sur la carte.


Résultat : Parcours affiché. **Test valide.** 

Le text-to-speech (guidage vocal) fonctionne correctement lors d'une course

Scénario 1 : On active le guide audio dans les paramètres puis on lance une nouvelle course. On met en pause au bout de 3 minutes pendant 2 minutes, puis on reprend. A 5 minutes, le text-to-speech doit se lancer.

Résultat : Le text-to-speech se lance à une durée de 5 minutes. **Test valide.** 

Scénario 2 : Cette fois on désactive le guide audio dans les paramètres. On lance une nouvelle course, et au bout de 5 minutes, aucun son ne doit sortir du téléphone.

Résultat : Le text-to-speech ne se lance pas. **Test valide.** 

Tests de validation

Les tests de validation ont été effectués le dimanche 20 janvier 2013 avec deux smartphones : un Google Galaxy Nexus, et un Google Nexus One à la version Android 2.2. Ces deux smartphones sont utilisés par les développeurs Android. L'ensemble des tests de validation seront refaits le lundi 21 janvier 2013 avec notre client M. Broisin, afin de valider avec lui le produit qu'il a demandé.

Cohérence des informations affichées sur l'écran pendant une activité sportive

Scénario : On démarre une nouvelle course, et on vérifie la cohérence des informations affichées à l'aide d'un chronomètre et d'un autre smartphone utilisant Google Maps, et en notant les informations affichées avant d'arrêter la course. Les informations notées doivent être vérifiées par le calcul.

Résultat : Informations cohérentes. **Test valide.**



La distance d'un parcours généré doit être égale à la distance demandée à 3% près

Scénario : On génère un parcours aller-retour de 10 km. Le parcours affiché doit donc être de 5 km à 3% près, soit entre 4,85 et 5,15. On vérifie la distance en utilisant Google Maps.

Résultat : 5,12 km. **Test valide.**



Possibilité de consulter l'ensemble des activités sportives effectuées

Scénario : On consulte l'historique, où un aperçu de chaque course effectuée doit être affiché. En sélectionnant une course, on doit obtenir le détail des performances de cette course.

Résultat : Aperçu présent. Affichage des performances en sélectionnant. **Test valide.**



Le partage de statistiques doit fonctionner

Scénario : On partage une course de 10 km en 1 heure avec une vitesse de 10 km/h et 685 calories brûlées. Pour cela on choisit d'envoyer un e-mail. La phrase prédéfinie doit être « J'ai parcouru 10 km en 1h avec une vitesse de 10 km/h, et j'ai brûlé 685 calories ! ».

Résultat : Phrase correcte dans l'e-mail reçu. **Test valide.**



L'utilisateur ne doit pas se rendre compte de la perte du signal GPS pendant sa course

Scénario : On démarre une nouvelle course, et on traverse le bâtiment de l'IUT. Le smartphone ne détectera aucun signal GPS, mais le parcours devra tout de même être tracé une fois sorti du bâtiment.

Résultat : Parcours tracé. **Test valide.** 

Une course doit pouvoir fonctionner hors connexion de données cellulaires

Scénario : On démarre une nouvelle course, puis on désactive le réseau de données cellulaires. On vérifie alors la cohérence des informations affichées, de la même manière que le test concernant la cohérence des informations.

Résultat : Informations cohérentes. **Test valide.** 

Dans le cas où la batterie se vide complètement, la course doit tout de même être enregistrée

Scénario : On démarre une nouvelle course, puis on éteint le smartphone pour simuler une décharge complète de la batterie. On le rallume, puis on vérifie que la course a été enregistrée dans l'historique.

Résultat : Course enregistrée. **Test valide.** 


Compatibilité avec les smartphones Android à la version 2.2 ou ultérieure

Scénario : On lance l'application sur l'autre smartphone de test, le Google Nexus One qui est tourné sous Android à la version 2.2. L'application doit fonctionner avec une interface graphique adaptée.

Résultat : Application fonctionne. Interface graphique adaptée. **Test valide.** 

Choix de 3 sports différents pour une activité

Scénario : On choisit le sport à effectuer, et on doit avoir le choix entre trois sports différents : la course à pied, le cyclisme, et le roller.

Résultat : Choix entre course à pied, cyclisme, et roller. **Test valide.** 

Conclusion

Tous les tests ont été validés, nous pouvons donc affirmer que le projet est terminé et qu'il est en accord avec les besoins de notre client. Nous pouvons maintenant lui remettre l'application, tout en effectuant une nouvelle fois les tests de validation avec lui pour lui prouver que l'application a été développée en respectant sa demande.