Compte rendu d'activité

Mission : Application Mobile

Tests unitaires

J'ai généré des classes de tests unitaires dans android studio :
FraisHFTest.java
FraisMoisTest.java
ProfilTEst.java

Documentation technique

J'ai généré une documentation technique javaDoc dans android studio :
http://namiktiab.com/GSB-AppliAndroid/javaDoc/index.html

Acces à la base de données

Fichier SQL pour générer la base MySQL remplie : https://github.com/grunam/GSB-AppliAndroid/blob/master/fichiers%20php/Suividevosfrais/dump_bdd_gsb_frais.sql

Tâche 1 : configuration

Configuration du projet dans L'IDE Android Studio avec le SDK de l'API 26 : Android 8,0 Oreo (avec un AVD type Nexus 65, 4,95, 1080x1920 xxhdpi).

Tâche 2 : Interdiction de saisie directe des quantités

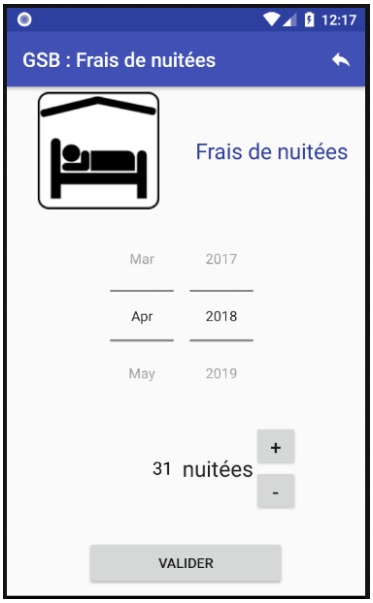
J'ai modifié le code existant pour interdire la saisie directe des KM dans l'activité correspondante : la quantité doit être obtenue uniquement en utilisant les touches + et -.

Dans le fichier GSB-AppliAndroid/app/src/main/java/fr/cned\emdsgil/suividevosfrais/activities/KmActivity.java ajout de la méthode interditSaisie() qui est appelée au démarrage de l'acivity dans la méthode onCreate(). Cette méthode désactive l'editText txtKm, elle fait perdre le focus à l'editText, elle affecte la couleur noire à l'editText, elle rend le focus invisible, elle désactive la capture d'évènements des touches de saisie de caractères sur l'editText, enfin cette méthode attribue une couleur de fond transparente à l'editText.

Tâche 3 : Enregistrement des autres catégories de frais forfaitisés

J'ai créé les autres activités de frais forfaitisés, sur le modèle de la saisie des frais km, en respectant la présentation des interfaces données ci-dessous. J'ai fait en sorte que les informations soient enregistrées (comme elles le sont déjà pour les km) ;

Sur le modèle des fichiers GSB-AppliAndroid/app/src/main/java/fr/cned\emdsgil/suividevosfrais/activities/KmActivity.java et GSB-AppliAndroid/app/src/main/res/layout/activity_km.xml déjà présents dans le projet, j'ai créé les fichiers activity_etape.xml, activity_repas.xml, activity_nuitee.xml, ainsi que les fichiers EtapeActivity.java, RepasActivity.java et NuiteeActivity.java. J'ai repris le code de KmActivity.java pour l'enregistrement des données via la sérialisation (notamment la méthode enregNewQte()) .

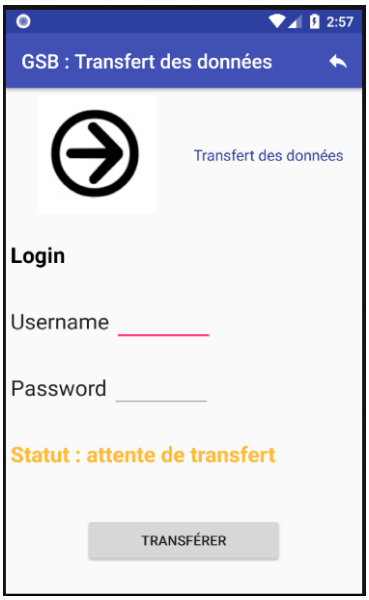


Tâche 4 : Suppression de frais hors forfait

Codage dans l'adapter de la liste des frais hors forfait GSB-AppliAndroid/app/src/main/java/fr/cned/emdsgil/suividevosfrais/activities/FraisHfAdapter.java des instructions nécessaires à la supprésion d'un élément de la liste dans la méthode getView. La classe gère l'affichage et les évènements liés à chaque élément d'une liste. Dans la méthode getView j'ai ajouté au viewHolder le bouton de suppression d'un élément de liste. J'ai récupéré l'année et le mois du DatePicker et j'ai déduit la clé nécessaire pour récupérer les frais hors forfait des données désérialisées. J'ai ajouté une méthode qui est lancé au clic sur le bouton de suppression d'un élément de liste. Lors de l'appel de cette méthode ; il est retiré, grâce à l'index de l'élément cliqué, les frais hors forfait de la liste des frais hors forfait nécessaires à la construction de la liste dans la méthode getView() de l'adapter. Il est également supprimé les frais hors forfait des données désérialisées des frais enregistrés. Ces frais hors forfait correspondant à la clé déduite plus haut et à l'index de l'élément supprimé. Cette liste des frais désérialisée est ensuite de nouveau sérialisée. l'adapter est ensuite mise à jour par l'appel à notifyDataSetChanged() : la méthode getView() est rappelée ce qui a pour effet de mettre à jour l'affichage de la liste .

Tâche 5 : Synchronisation avec la base de données distante

J'ai écrit le code derrière le bouton de synnchronisation du menu principal, qui me permet d'insérer dans la base distante tout ce qui a été enregistré en local. Le bouton de synchronistaion, codé dans le fichier MainActivity.java, donne accès à une nouvelle activity transfert que j'ai créé avec les fichiers activity_transfert.xml et TransfertActivity.java. Cette activity est une page d'authentification du visiteur médical avec l'username et le password. Cette page est également (si l'authentification réussie) une page de transfert des frais enregistrés dans l'application Android vers une base distante MySQL.



Pour cette tâche j'ai créé les fichiers activity_transfert.xml, TransfertActivity.java. J'ai utilisé et modifié les classes vue lors de mon TP Android de deuxième année AccesDistant.java, ControleAcces.java et Profil.java. J'ai récupéré et utilisé des classes techniques AccesHTTP .java et AsyncResponse.java vues en TP.

Dans la classe TransfertActivity à la création avec la méthode onCreate, la méthode statique getInstance de la classe ControleAcces est appelée avec en argument l'instance de la classe TransfertActivity (mot clé « this »). La méthode statique ControleAcces.getInstance instancie la classe ControleAcces.
Dans le fichier TransfertActivity.java lors du clic sur le bouton de transfert « transférer » de l'activity « Transfert de données » try sur la récupération du champ de saisie de l'username et du champ de saisie du password. En cas d'échec du try, rien ne se fait. Test pour savoir si le champ d'username ou de password est vide. S'il est vide affichage d'une fenêtre toast avec le texte : « Veuillez saisir tous les champs ». Sinon appel à la méthode TransfertActivity.afficheResult avec des arguments de type string : success affecté à '0', status affecté à '0', username avec la valeur du champ de saisie username, mdp avec avec la valeur du champ de saisie mdp. La méthode TransfertActivity.afficheResult prend également en argument la variable data de type ArrayList avec un ArrayList vide (pour le moment). Sortie du test. La méthode TransfertActivity.afficheResult appelle la méthode creerProfil de la classe ControleAcces qui reçoit en argument tous les paramètres de la méthode afficheResult (la méthode ControleAcces.creerProfil a les mêmes paramètres que la méthode TransfertActivity.afficheResult). Dans la méthode controleAcces.creerProfil la classe profil est instanciée avec en argument tous les paramètres de la méthode controleAcces.creerProfil. La classe AccesDistant est instanciée, la méthode envoi de la classe accesDistant est appelée avec en argument operation de type string, instancié à « connexion ». La méthode AccesDistant.envoi reçoit également en argument la méthode convertToJSONArray() de la classe Profil. La méthode Profil.convertToJSONArray() convertit en un tableau JSON les propriétés de la classe Profil : success, username, mdp et data. La classe AccesDistant implémente l'interface AsyncResponse chargée de redéfinir la méthode AccesDistant.processFinish pour qu'elle soit appelée au retour du serveur. Dans la méthode accesDistant.envoi la classe AccesHTTP est instanciée. La classe technique AccesHTTP hérite de AsyncTask qui est une classe de type Thread. Cela signifie qu'elle s'exécute dans un processus isolé, sans bloquer le reste de l'application. Dans la méthode AccesDistant.envoi, on attribut ensuite à la propriété delegate de l'instance de la classe AccesHTTP le mot clé « this » soit l'instance de la classe AccesDistant. Cette propriété delegate est de type AsyncResponse. Toujours dans la méthode AccesDistant.envoi la méthode addParam de l'instance de la classe AccesHTTP est appelé deux fois. Une fois pour ajouter le paramètre « operation » de valeur du paramètre operation d'AccesDistant.envoi. AccesHTTP.addParam est applelé une autre fois pour ajouter le paramètre « les données » de valeur du paramètre correspondant au tableau JSON des propriétés de la classe Profil : success, username, mdp et data transformée en chaîne. Cette méthode AccesHTTP.addParam permet d'insérer des paramètres dans la collection de paramètres qui seront transférés. La méthode execute de l'instance de la classe AccesHTTP est appelé avec en argument l'adresse du script PHP côté serveur : "http://192.168.1.40/Suividevosfrais/serveurfrais.php".
La méthode AccesHTTP.execute provoque l'appel de la méthode doInBackground de la classe AccesHTTP. Elle a en paramètre l'adresse du script PHP côté serveur. C'est la méthode qui va gérer en tâche de fond (donc dans un processus indépendant) la connexion au serveur distant et donc par exemple à une page PHP qui va interroger une base de données et retourner un résultat. L'essentiel du code est dans un try/catch car la connexion peut éventuellement échouer (serveur non disponible...) et cela ne doit pas pour autant poser de problème à l'application. Dans la méthode AccesHTTP. DoInBackground, remarquez la ligne de code suivante :
HttpResponse reponse = cnxHttp.execute(paramCnx)
Cette ligne attend une réponse du serveur et la récupère dans la variable reponse.

Côté serveur j'ai créé les fichiers :

GSB-AppliAndroid/fichiers php/Suividevosfrais/class.pdogsb.inc.php
GSB-AppliAndroid/fichiers php/Suividevosfrais/class.utils.inc.php
GSB-AppliAndroid/fichiers php/Suividevosfrais/serveurfrais.php

Côté serveur dans le script PHP serveurfrais.php l'appel à la méthode PdoGsb::getPdoGsb(); récupère l'instance $pdo de la classe PdoGsb. Deux Test, un pour savoir pour savoir si le paramètre operation envoyé par l'instance de la classe AccesHTTP existe, et un autre pour savoir si ce paramètre est égal à « connexion ».
Si oui pour le dernier test ouverture d'un try. Récupération du login et du password à partir du paramètre « lesdonnees ». Appel à la méthode $pdo->getInfosVisiteur avec en arguments le login et le mot de passe. Test pour savoir si la méthode $pdo->getInfosVisiteur ne renvoie pas un tableau ou si le tableau renvoyé a le paramètre comptable à 1 (soit le login et mot de passe correspond à un comptable). Si oui initialisation des paramètres d'un array response. Le paramètre success de cet array response est affecté avec la valeur '0'. Le paramètre status de cet array response est affecté avec la valeur 'username ou password incorrect(s)!'. Les paramètres username et mdp de cet array response sont affectés avec les valeurs chaine vide ''. Si non Le paramètre success de cet array response est affecté avec la valeur '1'. Le paramètre status de cet array response est affecté avec la valeur 'authentification réussie !'. Les paramètres username et mdp de cet array response sont affectés avec les valeurs des paramètres login et mdp du tableau renvoyé par la méthode $pdo→getInfosVisiteur. Ouverture du catch, le paramètre success de l'array response est affecté avec la valeur '0'. Le paramètre status de cet array response est affecté avec la valeur 'Erreur !' associé à l'erreur renvoyée. Les paramètres username et mdp de cet array response sont affectés avec les valeurs chaine vide ''. print sur l'encodage JSON de cet array response.

Le tableau est récupéré par la méthode AccesHTTP.doInBackground dans la variable ret. La fonction AccesHTTP.onPostExecute envoie à la méthode processFinish de l'instance delegate de la classe AccesDistant qui implémente l'interface AsyncResponse le tableau de données transformé en chaîne.