Guillaume MONTAVON
Benoît MEILHAC

Rapport Technique du projet

Gestionnaire de tâches pour Android

Google

Sommaire

1. Introduction

Dans le cadre de notre projet de Master 1 Informatique, nous avons choisis de développer une application Android qui permette à un utilisateur de gérer une liste de tâches sur son Smartphone. La particularité de cette application est le fait qu'un utilisateur puisse synchroniser ses tâches depuis et vers un serveur distant développer en PHP/MySQL. Le format utilisé pour transporter les données du mobile au serveur est JSON. La réalisation du projet se découpe donc en deux parties distinctes : le développement du serveur distant et le développement de l'application Android. L'application a été testée sous Android 2.2 et 2.3.

2. Outils utilisés

Les applications Android sont développées en Java, c'est pourquoi il faut au préalable installer le JRE ainsi que le JDK de Java avant de pouvoir développer.

Durant notre projet, nous avons utilisé principalement 3 outils de développement :

- Le **SDK Android**
- L'**IDE Eclipse** avec le plugin **ADT** (Android Development Tools)
- **WAMP** (Windows Apache MySQL PHP)

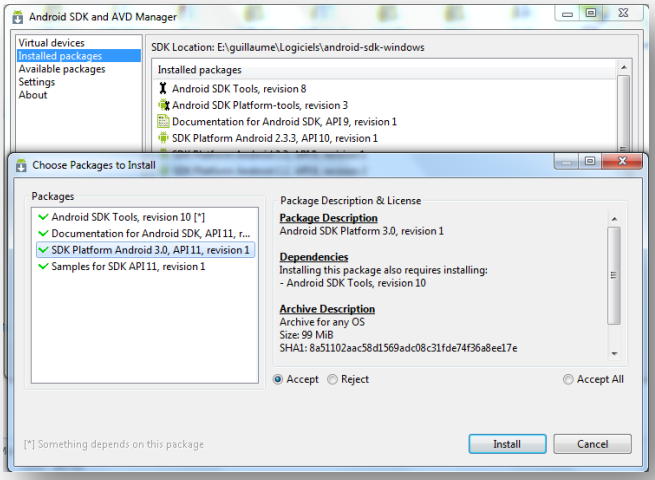
2.1. Android SDK

Afin de développer une application, il faut télécharger le kit de développement SDK Android à l'adresse suivante : http://code.google.com/android/download.html. Une fois téléchargé, il faut le décompresser dans un dossier et l'exécuter. Ce dernier contient plusieurs outils très utiles :

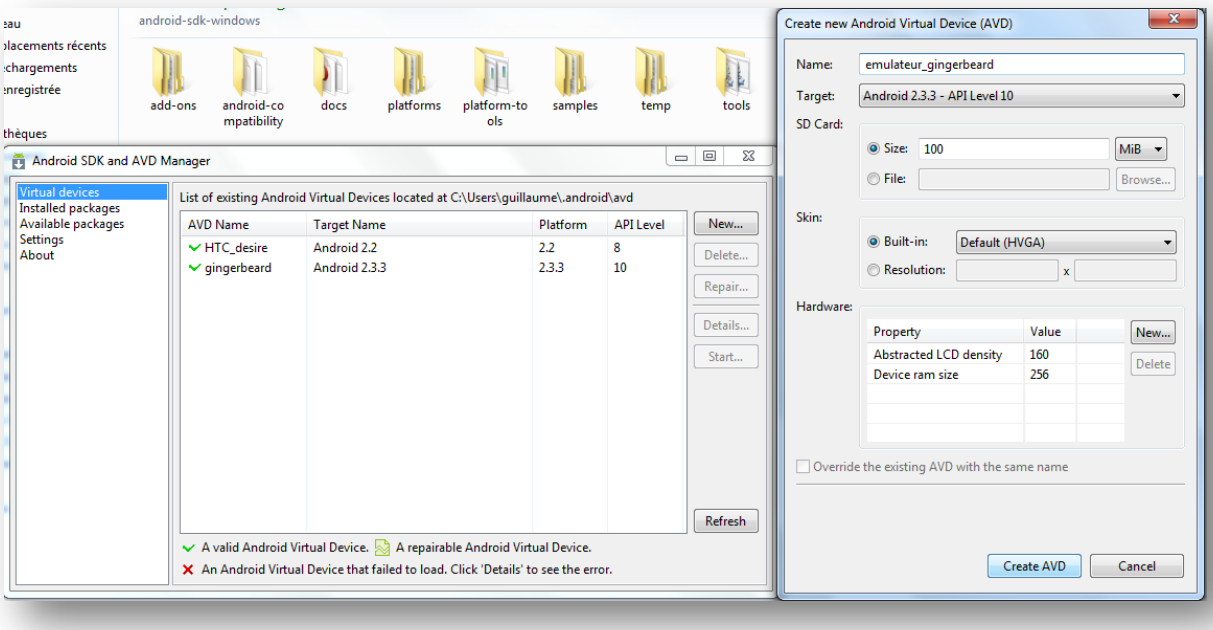
- L'API Android
- Des exemples de code, avec des exemples d'applications fonctionnels
- Une documentation
- Des pilotes d'un grand nombre de Smartphones qui permettent de tester ses applications directement sur son mobile en l'exécutant depuis Eclipse

- Un émulateur qui permet de tester ses applications sans avoir de Smartphone

Une fois le SDK exécuté, il propose alors de télécharger et d'installer les API de Android (on peut choisir par exemple Android version 2.3.3 …), il suffit d'appuyer sur « accept » puis « install » (voir Image ci-dessous).



Une fois l'API installé, on peut créer un émulateur (dans l'onglet « Virtual devices »). On lui donne un nom puis la version d'Android utilisé (ici 2.3.3), la taille de la carte SD … On appuie ensuite sur « Create AVD » et l'émulateur est ainsi créé. Pour le lancer il suffit de sélectionner l'émulateur créé puis d'appuyer sur « start » (voir image ci-dessous).



2.2. Android dans Eclipse

Pour développer l'application, nous avons utilisé l'IDE Eclipse associé à son plugin **ADT** (Android Development Tools). Pour cela il faut télécharger Eclipse (http://www.eclipse.org/downloads/), le décompresser et l'exécuter. Ensuite il faut installer le plugin Eclipse : **ADT** (Android Development Tools), en allant dans le menu Help→install new software et en entrant https://dl-ssl.google.com/android/eclipse/ dans le champ d'ajout. Une fois le plugin installé, il suffit d'aller dans les paramètres d'Eclipse et de configurer le chemin où se trouve le SDK Android.

Pour créer un nouveau projet Android, il suffit de suivre les étapes suivantes. On entre la version minimum que l'on souhaite. On obtient alors l'arborescence de fichier ci-contre. Pour exécuter l'application il suffit de faire un clic droit sur le projet et de faire « Run as→Android Application ». Si un Smartphone est connecté à l'ordi, l'application s'exécutera sur celui-ci, sinon, elle s'exécutera sur l'émulateur.

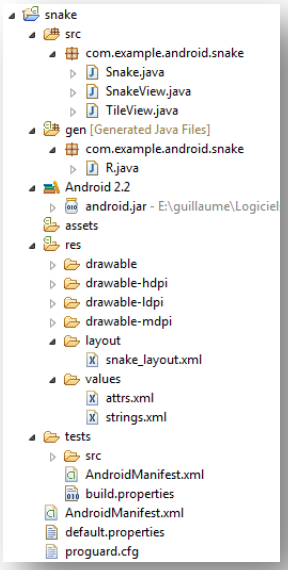


Image 1: Arborescence des fichiers dans une application

Une des fonctionnalités les plus importante dans le plugin Android est la perspective DDMS (Dalvik Debug Monitor Server), elle permet de débuguer facilement les applications, elle permet de :

- afficher les messages d'erreurs et de warning
- naviguer dans les fichier du Smartphone

- afficher les processus, les Thread, la pile du Smartphone
- simuler un appel ou un envoi de SMS
- modifier les coordonnées GPS du Smartphone

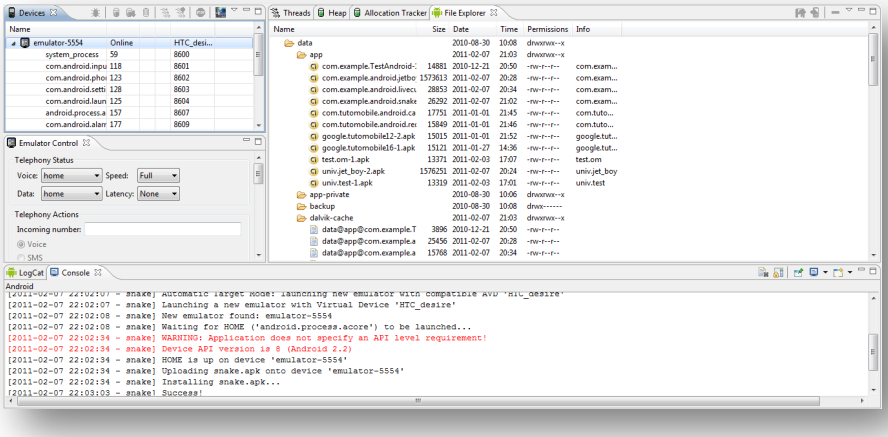
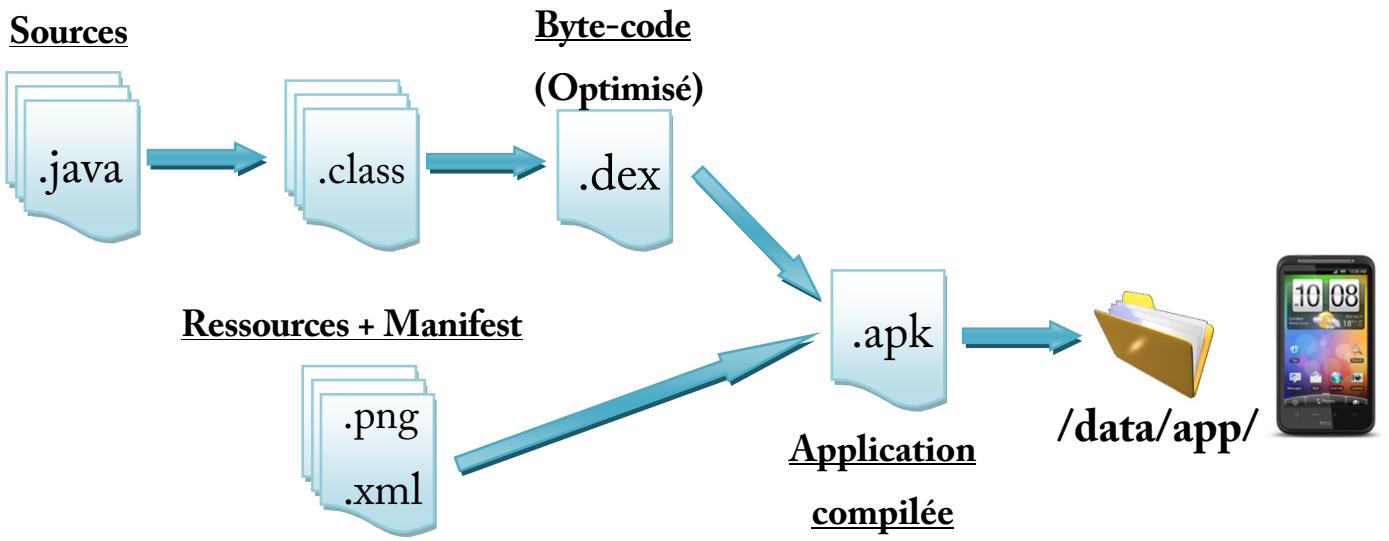


Image 2 : perspective DDMS dans Eclipse



2.3. WAMP

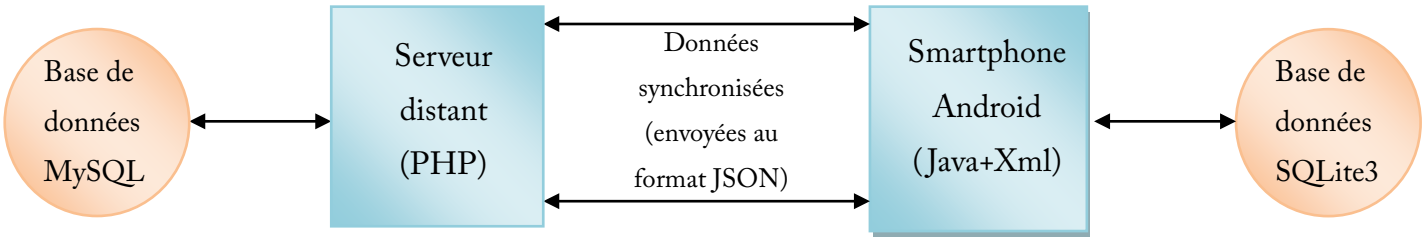
Afin de développer et de tester la partie serveur de notre projet, nous avons utilisé l'ensemble de logiciel WAMP. Il nous a permis de mettre en place un serveur PHP/MySQL en local sur notre ordinateur personnel, afin de tester la synchronisation entre le Smartphone (émulateur) et le serveur distant. Pour accéder au serveur local depuis l'émulateur, l'adresse IP à utiliser dans celui-ci est 10.0.2.2

et non 127.0.0.1 (localhost).

Ensuite, après avoir pratiquement terminé notre développement et pour faciliter les tests sur un Smartphone réel, nous avons installé notre partie serveur distant sur un serveur gratuit qui se nomme http://projetandroid.hosting.olikeopen.com/gestionnaire_taches/. Ainsi, depuis n'importe quel endroit on peut utiliser l'application et la synchroniser sur ce serveur en utilisant le réseau mobile du Smartphone (3G, …).

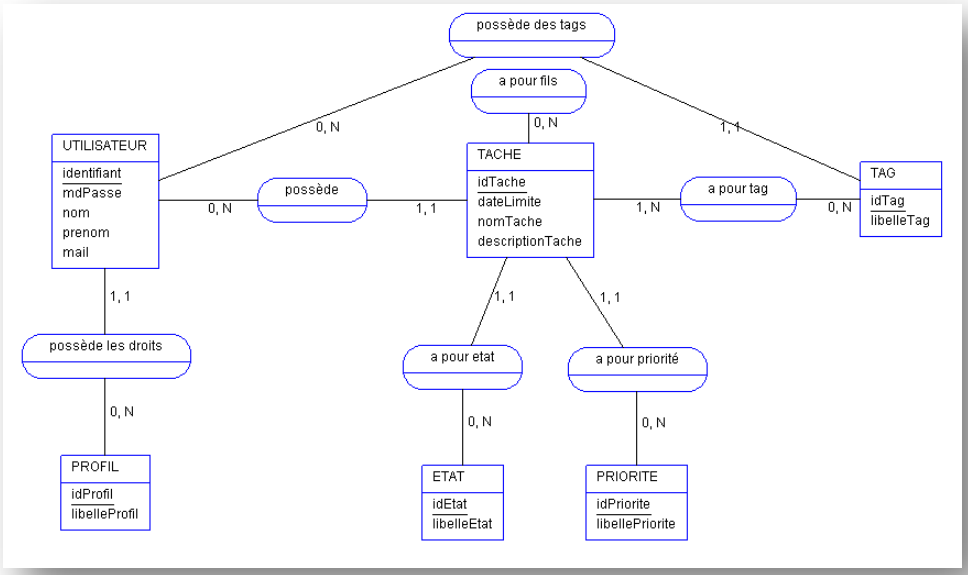
3. Description techniques

Notre application se décompose en plusieurs parties, comme le montre le schéma ci-dessous.



3.1. Serveur distant

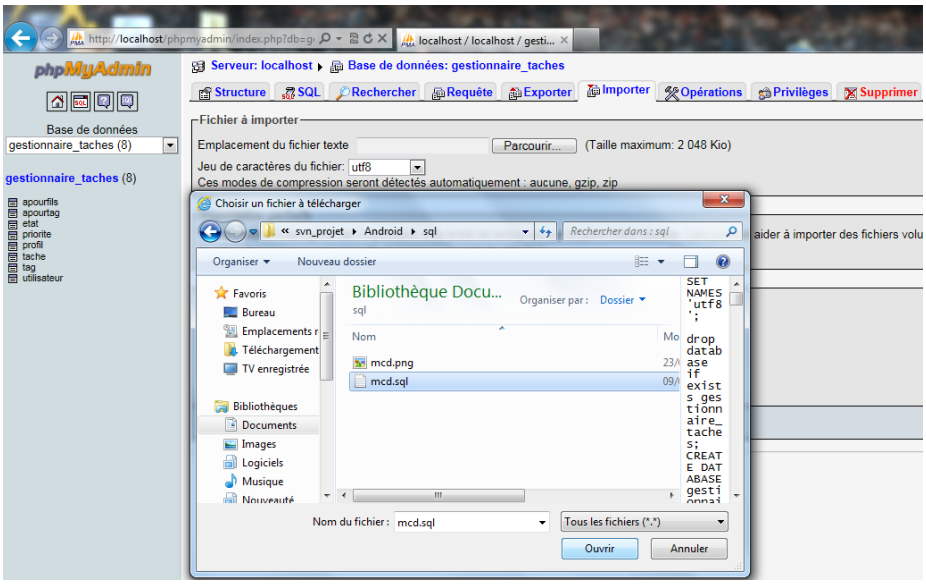
Afin de réaliser notre serveur distant qui permet de synchroniser les tâches, nous avons tout d'abord réalisé un MCD qui nous a permis par la suite de créer la base de données SQL. Certaines tables ont été créées mais ne sont pas utilisées par l'application (Etat, Priorité).

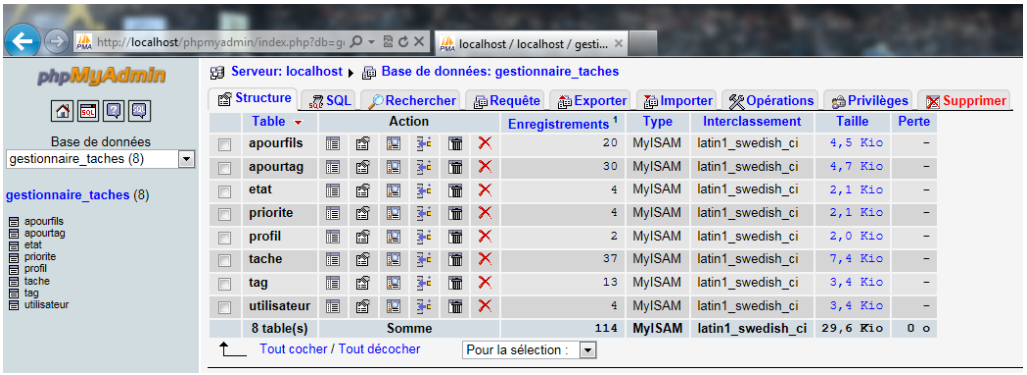


Chaque utilisateur possède une liste de tâches et de tags.

Le fichier SQL qui permet d'importer la base de données sur le serveur se nomme MCD.sql.

Pour l'importer dans la base de données, il suffit de se rendre dans phpMyAdmin, puis de créer une nouvelle base et ensuite d'importer le fichier SQL via l'onglet Importer de phpMyAdmin.





Les fichiers PHP du projet sont au nombre de 3 :

- **config.inc.php** : contient les paramètres de connexion à la base de données (mot de passe, identifiant, …)
- **fonctions.inc.php** : comporte un grand nombre de fonctions PHP qui permettent l'importation et l'exportation des données, la vérification de l'identité de l'utilisateur, la modification des tâches, ...
- **requeteAndroid.php** : page principale qui sera appelé par le Smartphone. Elle utilise les fonctions PHP du fichier fonctions.inc.php. Suivant l'objet de la requête, le Smartphone enverra une requête POST avec comme paramètre POST, l'identifiant et le mot de passe de l'utilisateur ainsi que l'objet de la requête dans l'url : ……requeteAndroid.php?objet=…..

Exemple d'utilisation pour une importation du serveur vers le mobile :

Le Smartphone va envoyer une requête POST à l'adresse :
http://projetandroid.hosting.olikeopen.com/gestionnaire_taches/requeteAndroid.php?objet=importer
avec comme paramètre de requête POST :

- identifiant=« identifiant de l'utilisateur »
- mdPasse=« mot de passe de l'utilisateur crypté en MD5 »

Exemple de tâches encodées au format JSON reçus par le Smartphone


```
{"tags": [
        {"idTag":1,"libelleTag":"Université"}, {"idTag":2,"libelleTag":"personnel"},
        {"idTag":3,"libelleTag":"professionnel"}, {"idTag":4,"libelleTag":"examen"}
```

```
        ],"nbTags":4,
"taches":[
        {"idTache":1,"nomTache":"test ahaaa","descriptionTache":"description test tache
        1","dateLimite":"","idEtat":2,"idPriorite":1,"apourtag":[1,2,3],"apourfils":[2,3]},
        {"idTache":2,"nomTache":"test 2","descriptionTache":"description test tache
        2","dateLimite":"","idEtat":1,"idPriorite":4,"apourtag":[1,3],"apourfils":[]},
        {"idTache":3,"nomTache":"ahaaa","descriptionTache":"description test tache
        3","dateLimite":"","idEtat":2,"idPriorite":4,"apourtag":[2],"apourfils":[]},
        {"idTache":4,"nomTache":"test tache 4","descriptionTache":"description test tache
        4","dateLimite":"","idEtat":3,"idPriorite":3,"apourtag":[1],"apourfils":[5]},
        {"idTache":5,"nomTache":"test tache 5","descriptionTache":"description test tache
        5","dateLimite":"","idEtat":4,"idPriorite":2,"apourtag":[3],"apourfils":[]},
],"nbTaches":8,
"apourtag":[
        {"idTache":1,"idTag":1}, {"idTache":1,"idTag":2}, {"idTache":1,"idTag":3},
        {"idTache":2,"idTag":1}, {"idTache":2,"idTag":3},
        {"idTache":3,"idTag":2}, {"idTache":4,"idTag":1}, {"idTache":5,"idTag":3}
        ],
"apourfils":[ {"idPere":1,"idFils":2}, {"idPere":1,"idFils":3}, {"idPere":4,"idFils":5} ]
}
```


Le serveur distant accepte 3 modes de synchronisation :

- Ecraser le serveur avec les données du mobile
- Ecraser le mobile avec les données du serveur
- Combiner les données du serveur avec celles du Smartphone :
    - Le mobile envoie toutes ses tâches au serveur (en format JSON)
    - Le serveur vérifie la version de chaque tâche du Smartphone, si la version d'une tâche reçu est supérieur à celle dans le serveur on la modifie, sinon on ne fait rien. Si une tâche n'existe pas sur le serveur, on l'ajoute.
    - Le serveur envoie au mobile (en format JSON) les tâches qu'ils ne possèdent pas et celle qui ont une version supérieur
    - Le mobile reçoit les tâches modifiées ou ajoutées et les met à jour dans sa base de données.

3.2. Application Android

3.2.1. Arborescence des fichiers

L'application est composé d'un grand nombre de fichier Java et XML, voici la présentation des principaux fichiers et packages contenu dans le projet :

- **Dossier src** : contient les sources du projet (fichiers Java) :
  - **Package univ_fcomte.bdd** : gère la base de données SQLite de l'application (voir paragraphe 3.2.7).
  - **Package univ_fcomte.gtasks** : gère l'interface graphique de l'application ainsi que les événements (contient toutes les activités du projet, dont la principale : GestionnaireTaches)
  - **Package univ_fcomte.synchronisation** : gère la synchronisation entre le serveur et le Smartphone.
  - **Package univ_fcomte.tasks** : modèle de l'application (Tâches, Tags, paramètre courants, …)
- **Dossier res** : ressources de l'application (fichiers XML, images, …) :
  - **Anim** : animations qui sont utilisées pour la transition entre plusieurs activités
  - **Drawable-….** : icones utilisés par l'application
  - **Layout :** contient l'interface de l'application, chaque fichier xml correspondant à l'interface graphique d'une activité de l'application. Exemple : details_taches.xml correspond au formulaire qui est affiché lorsqu'on modifie ou ajoute une tâche.
  - **Menu** : gère le menu de la page principale de l'application (menu.xml), et le menu de la page de modification de tâche (menu_details.xml)
  - **Values** : Contient tous les messages affichés à l'utilisateur (strings.xml). Permet l'internalisation, si le Smartphone est configuré en français, il va afficher les messages du dossier values-fr sinon, il affichera les messages de values.
  - **Xml** : contient le formulaire qui est utilisé pour la gestion des préférences de l'application.
- **AndroidManifest**.xml : permet de modifier toutes les propriétés du projet : nomde l'application, version de l'application, activité principale, permissions, …

3.2.2. Activité principale

Lorsque l'utilisateur lance l'application, il arrive sur la page ci-dessous

Il obtient la liste des tâches de son application s'il en a déjà créé. La barre du dessus affiche le nombre de tâches et le nom de l'application. En dessous, se trouve 3 boutons : retour aux tâches racines, retour à la tâche mère et ajout d'une tâche. L'utilisateur peut appuyer sur le bouton « menu » pour afficher les options de l'application : synchronisation, trie, réglages, ajout d'utilisateur …

Pour modifier une tâche on peut tout simplement appuyer sur la tâche et une nouvelle fenêtre s'ouvrira. Afin de voir les tâches filles d'une tâche, on peut appuyer sur le bouton bleu tout à droite de cette dernière.

Lorsque l'utilisateur change l'orientation de son Smartphone l'application redimensionne automatiquement cette fenêtre avec la liste des tâches (voir image ci-dessous).

Cet écran est géré par l'activité principale du projet : GestionnaireTaches et son interface graphique et géré par les fichiers « main.xml » et « affichageitem.xml ». Le premier affiche les boutons et la liste, le second représente une seule tâche. Lorsque l'application se lance elle fait appelle à la méthode *onCreate()* de la classe GestionnaireTaches. Cette dernière est donc assez longue car elle contient le code principal de l'application.

3.2.3. Préférences

La gestion des préférences est gérer automatiquement par Android. Pour y accéder, l'utilisateur peut appuyer sur le bouton menu, puis choisir réglage. Elle permet à l'utilisateur de choisir si il souhaite utiliser un compte utilisateur (mot de passe et l'identifiant), utiliser un proxy (adresse et port) ou encore s'il souhaite afficher les taches annulées.

Ces réglages sont stockés automatiquement par Android dans un fichier sur le Smartphone, ainsi lorsque l'utilisateur quitte l'application et la lance à nouveau, il garde ses réglages. La gestion de ces préférences se trouve dans la classe Preferences (package univ_fcomte.gtasks) et est très simple. Le formulaire avec les champs identifiant …, les cases à cocher se trouve dans le fichier prefs.xml (dossier res/xml). Il est alors très facile de rajouter des champs dans le formulaire de préférence en les ajoutant dans ce fichier XML.

Pour obtenir la valeur d'un champ ou d'une case des préférences, il suffit dans n'importe quel Activity du projet de récupérer la valeur. Par exemple pour savoir si l'utilisateur souhaite utiliser le proxy : *PreferenceManager.getDefaultSharedPreferences(context).getBoolean("utilise_proxy", false).*

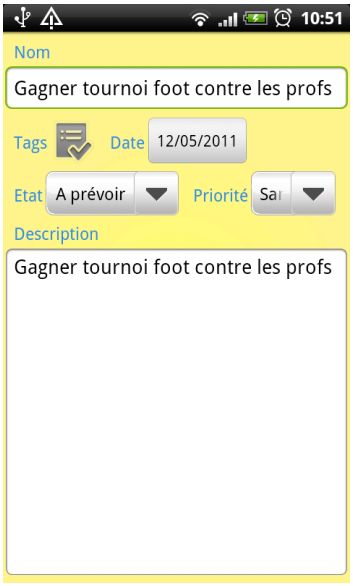
3.2.4. Ajout utilisateur

La gestion de la synchronisation est réalisé grâce, notamment aux comptes utilisateurs. En effet, le serveur permet d'enregistrer une liste de tâches et tags associées à un utilisateur. C'est pourquoi, si l'utilisateur de l'application souhaite synchroniser ses tâches sur le serveur, il doit auparavant créer un compte utilisateur via le menu « Ajout d'un utilisateur ». Pour cela, il entre les informations demandées (nom, mot de passe, mail, …), un message lui précisera si le compte a bien été créé et s'il souhaite se connecter automatiquement. Sinon, il pourra toujours se connecter plus tard dans les préférences de l'application.

L'ajout d'un utilisateur est réalisé par la classe « AjoutUtilisateur » (package univ_fcomte.gtasks) et son interface graphique par le fichier xml « ajout_utilisateur.xml » (res/layout/).

3.2.5. Description d'une tâche

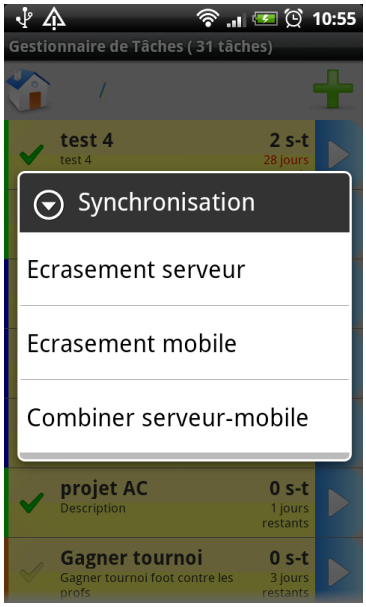
Lorsque l'on clique sur une tâche pour la modifier ou lorsque l'on créer une tâche, un formulaire s'affiche à l'écran avec les différentes informations de la tâche (nom, description, etat, priorite, date, tags). Cette page permet de modifier (ou d'ajouter) les informations d'une tâche, lorsqu'on a terminé, on appuie sur le bouton « back », la tâche est alors enregistré et on arrive ensuite sur la page principale avec la liste des tâches.



La gestion de cette écran se situe dans la classe DetailsTaches et son interface graphique dans le fichier « details_taches.xml » (res/layout/).

3.2.6. Synchronisation

La synchronisation est effectué par les classes qui se trouve dans le package « univ_fcomte.synchronisation ». Ce dernier possède 4 classes :



- La classe Synchronisation qui possède une méthode getHTML(…) qui permet d'envoyer une requête vers un site web et de recevoir la réponse dans un format String.
- La classe EnvoyerJson qui permet de générer une chaîne de caractères en format JSON de la liste de tâches de l'application.
- La classe JsonParser qui permet de convertir du JSON et de l'importer dans la base de données du Smartphone.
- La classe ThreadSynchronisation qui permet de lancer un nouveau Thread pour que l'application ne soit pas indisponible lorsque l'on synchronise. Cette classe va s'occuper d'envoyer les requêtes au serveur distant et de recevoir la réponse. Elle possède une variable modeSynchronisation qui peut prendre comme valeur :
    - ECRASEMENT_SERVEUR
    - ECRASEMENT_MOBILE
    - COMBINER_SERVEUR_MOBILE
    - SUPPRESSION_TACHES
    - SUPPRESSION_TAGS
    - AJOUT_UTILISATEUR

L'utilisateur peut synchroniser ses tâches à tout moment en appuyant sur « menu », puis en sélectionnant « Synchronisation ». Il suffit qu'il est entré son identifiant et mot de passe dans les réglages. Il a alors le choix entre 3 modes de synchronisations :

- Ecraser le serveur avec les données du mobile
- Ecraser le mobile avec les données du serveur
- Combiner les données du serveur avec celles du Smartphone

Il attend ensuite quelques secondes afin que les données soit synchronisées.

Si l'on souhaite modifier l'adresse du serveur distant il suffit d'aller dans la classe « Modele » et entrer l'url du serveur distant dans la variable serveur (dans le constructeur).

3.2.7. Modèle

Le modèle de l'application se trouve dans le package « univ_fcomte.tasks ». Les tâches et tags de l'application sont stockés dans la base de données SQLite. Ces tâches sont chargées au démarrage de l'application dans des classes (Tache, Tag) qui permettent par la suite de les réutiliser plus facilement (affichage, modification, trie, …). Ainsi l'application possède un objet Modele, qui lui-même est composé d'une liste d'objets Tâches et une liste d'objets Tags, ainsi que les différents paramètres actuelles (dossier racine, tâche courante, adresse du serveur distant, …). Cette classe Modele permet dans chaque activité de l'application d'accéder aux données et donc de les afficher. Par exemple chaque activité peut accéder au modèle et donc à la liste des tâches en faisant :

(MonApplication)getApplication().getModele()……

3.2.8. SQLite

Notre application utilise la base de données présente par défaut dans tous les Smartphone Android : SQLite3. Cette base de données nous permet de stocker les tâches de l'application de manière permanente. Ainsi, à chaque fois que l'utilisateur démarre l'application, celle-ci va charger les tâches de la BdD dans l'application et les afficher. Lorsqu'on modifie, ajoute, supprime des tâches, ou que l'on synchronise avec le serveur, l'application va mettre automatiquement la base de données à jour. Cette base de données est la même que celle utilisé sur le serveur distant mais avec l'utilisateur en moins (même MCD avec en moins : tables utilisateur et profil ainsi que les champs « identifiant » dans les tables tag et tache).

Cette base de données se trouve sur le Smartphone dans le dossier data/data/univ_fcomte.gtasks/databases/ et se nomme « gestionnaire_taches.db ». Il est possible, pour le débogage de l'application, d'afficher les données contenu dans cette base de données. Pour cela, il faut lancer l'émulateur, puis de lancer un invité de commande, ensuite on se rend dans le dossier du SDK Android puis dans le dossier « tools ». Il faut ensuite exécuter « adb shell », se rendre dans le dossier contenant la base de données (« cd data/data/univ_fcomte.gtasks/databases/ »), puis exécuter « sqlite3 gestionnaire_taches.db ». On peut maintenant utiliser les commandes habituelles de SQL pour afficher, modifier les données (exemple : «select * from tag; »). L'image ci-dessous présente cette procédure.

Au niveau du Smartphone la gestion de la base de données se situe dans le fichier MaBaseSQLite.java (package univ_fcomte.bdd). Cette classe permet de modifier, ajouter et supprimer des données sur la BdD SQLite.

Exemple : la méthode **public int modifTache(Tache tache)** permet de modifier une tâche dans la base de données, elle prend en paramètre un objet Tâche qui en fonction de son identifiant va permettre de modifier dans la base de données cette dernière.

3.3. Mise à jour de l'application

La mise à jour d'une application Android est assez simple, il suffit de …

Différentes étapes :

- Modifier le code source de l'application.
- Modifier le numéro de version (Version Code) dans AndroidManifest.xml
- Ajouter dans strings.xml (dans values et values-fr) un item dans la string-array « version_news » qui contiendra les modifications effectuées par cette mise à jour. Le contenu de l'item sera affiché au premier démarrage lorsqu'un utilisateur aura fait la mise à jour de l'application.

4. Conclusion

Table des illustrations