

## RAPPORT DE PROJET

### Projet : Gestionnaire simple de tâches

#### **Taskinator Android**



COUTABLE Guillaume, RULLIER Noémie  
10 avril 2013

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Les fonctionnalités</b>	<b>3</b>
2.1	Fonctionnalités principales . . . . .	3
2.2	Fonctionnalités secondaires . . . . .	3
<b>3</b>	<b>Scénarios</b>	<b>4</b>
3.1	Scénario 1 - Création/Utilisation/Suppression d'une liste . . . . .	4
3.2	Scénario 2 - Gestion des listes et sauvegarde . . . . .	4
<b>4</b>	<b>L'IHM</b>	<b>5</b>
4.1	L'activité principale . . . . .	5
4.1.1	Choix réalisés (Position des boutons, Gestures, ...) . . . . .	5
4.2	Affichage des listes et tâches . . . . .	7
<b>5</b>	<b>Le modèle</b>	<b>8</b>
<b>6</b>	<b>Sauvegarde des données</b>	<b>9</b>
<b>7</b>	<b>Expérimentation</b>	<b>10</b>
7.1	Suppression par Swipe . . . . .	10
7.2	Déplacement de listes/tâches par Drag and Drop . . . . .	10
<b>8</b>	<b>Limite de l'application</b>	<b>11</b>
<b>9</b>	<b>Conclusion générale</b>	<b>12</b>

# 1 Introduction

L'objectif de ce projet fut de développer un gestionnaire simple de tâches. Celui-ci devait permettre de créer des listes de tâches et de suivre l'avancement de celles-ci.

Afin de créer cette application que nous avons appelé *Taskinator*, nous avons établi plusieurs étapes dans l'avancement du projet. Ce rapport présentera ces étapes les unes après les autres.

## 2 Les fonctionnalités

La première étape fut d'analyser l'ensemble des fonctionnalités que notre application devait proposer.

### 2.1 Fonctionnalités principales

Voici dans un premier temps les fonctionnalités principales :

**Créer une liste :** cette fonctionnalité permet à l'utilisateur de créer une liste vide.

**Créer une tâche :** cette fonctionnalité permet à l'utilisateur de créer une tâche.

**Supprimer un élément :** cette fonctionnalité permet de supprimer une tâche ou une liste. Cette fonctionnalité est à manipuler avec précaution, en effet dans le cas d'une liste, la suppression de celle-ci implique aussi la suppression de toutes ses tâches.

### 2.2 Fonctionnalités secondaires

Voici les fonctionnalités secondaires :

**Monter / Descendre :** cette fonctionnalité permet de monter ou descendre une liste ou une tâche. Dans le cas d'une liste, toutes ces tâches sont aussi montées/descendues d'un rang. Dans le cas d'une tâche, on autorise un déplacement dans une autre liste. (Cette fonctionnalité n'a pas été implémentée)

### 3 Scénarios

Nous avons imaginé différents scénarios d'utilisation de notre application.

#### 3.1 Scénario 1 - Création/Utilisation/Suppression d'une liste

Ce premier scénario permet de créer une liste et d'y ajouter des tâches.

1. L'utilisateur crée une liste et lui donne un nom *Ski*.
2. Il crée ensuite une tâche dans cette liste et lui donne un nom *Bonnet*.
3. Il crée ensuite une autre tâche dans cette liste et lui donne un nom *Echarpe*.
4. L'utilisateur supprime la tâche *Bonnet*.
5. L'utilisateur supprime la liste *Ski*. Une popup apparaît pour l'avertir que cette suppression supprimera aussi toutes les tâches de la liste.

#### 3.2 Scénario 2 - Gestion des listes et sauvegarde

Ce scénario permet de créer des listes et de les modifier. Il permet aussi de constater que l'état de l'application est sauvegardé.

1. L'utilisateur choisit de créer une liste et lui donne un nom *Course*.
2. Il décide ensuite de créer une nouvelle liste et lui donne un nom *Sac de voyage*.
3. Il décide ensuite d'inverser ces deux listes, pour cela il reste longtemps appuyé sur la liste *Sac de voyage* et la glisse un rang plus haut.
4. L'utilisateur va ensuite supprimer la liste *Course*.
5. L'utilisateur va ensuite ouvrir une autre application et revenir sur *Taskinator*. Il peut constater que l'application est ouverte avec l'état dans lequel on l'a quittée. C'est à dire que si une liste était ouverte ou fermée quand on a quitté l'application, celle-ci sera dans le même état quand on y reviendra.
6. L'utilisateur décide de créer une nouvelle tâche dans *Sac de voyage*. Pendant qu'il entre le nouveau nom de cette tâche, il va recevoir un appel. Lorsque l'appel est terminé, il sera automatiquement redirigé vers *Taskinator* exactement dans le même état qu'avant l'appel. S'il avait commencé à taper *Chauss*, il retrouvera le même début de mot.
7. L'utilisateur va recevoir un appel pendant qu'il est en train de rentrer un nom pour sa liste
8. Cette fois-ci, l'utilisateur quitte l'application. De même s'il l'a lancée à nouveau, celle-ci est ouverte avec le dernier état dans lequel on l'a quittée.

## 4 L'IHM

### 4.1 L'activité principale

L'application est tout simplement composée d'une zone réservée à l'affichage de la liste en cours de création et d'une barre d'action.

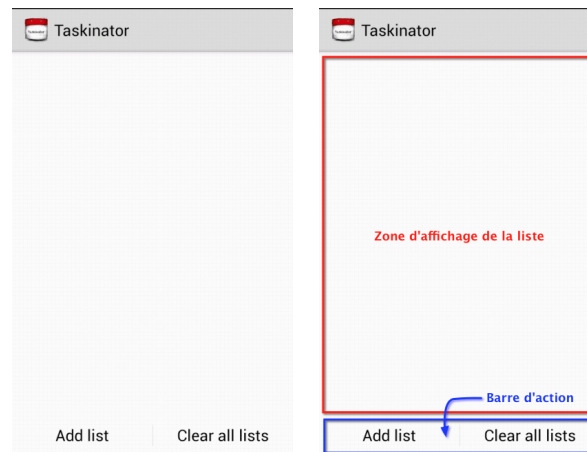


FIGURE 1 – L'application

#### 4.1.1 Choix réalisés (Position des boutons, Gestures, ...)

Dans cette section, nous expliquons les choix effectués et pourquoi nous avons choisi de mettre en place ces solutions :

- **Fonctionnalité de création de liste** : nous avons choisi de créer de nouvelles listes en cliquant sur le bouton "Add list" en bas de l'écran.

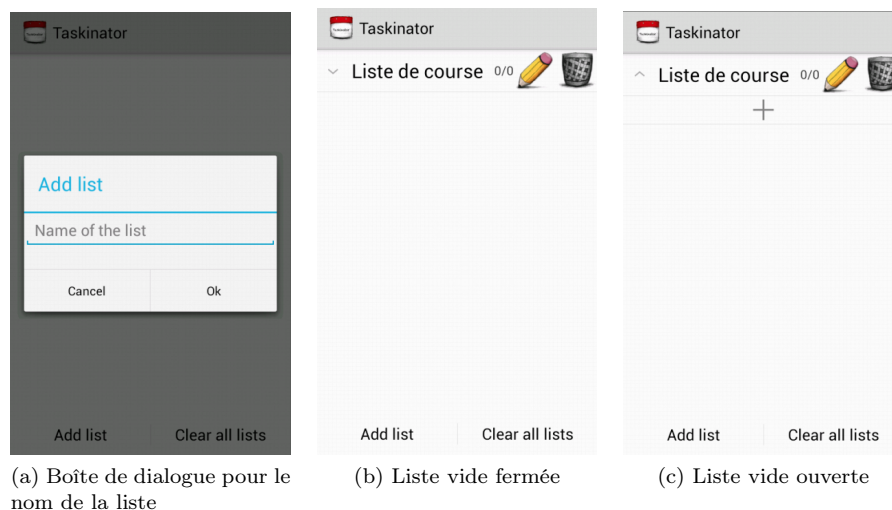
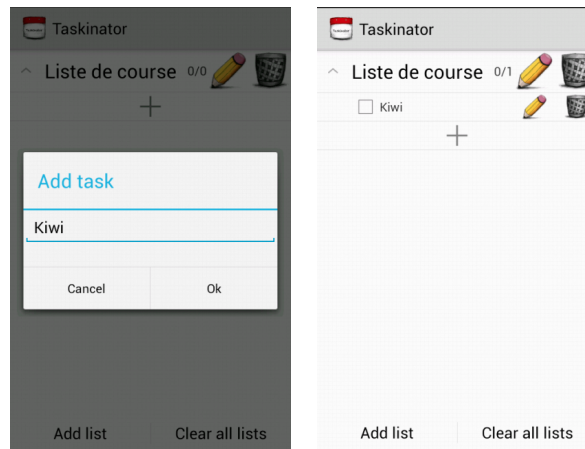


FIGURE 2 – Ajout d'une liste

Une boîte de dialogue apparaît et permet de rentrer le nom de cette nouvelle liste. L'utilisateur doit ensuite valider en cliquant sur "OK", s'il appuie sur "Cancel", la nouvelle liste ne sera pas créée. Si l'utilisateur déroule sa liste, un nouvel item apparaît sur la page muni d'un bouton "+". Celui-ci permettra de créer de nouvelles tâches pour la liste.

- **Fonctionnalité de création de tâche** : nous avons choisi de créer de nouvelles tâches en cliquant sur le bouton "+" présent sous chaque liste pour ajouter une tâche à la liste

correspondante. Lorsque l'utilisateur clique sur ce bouton, une nouvelle boîte de dialogue apparaît et permet de rentrer le nouveau nom de la tâche.





(a) Boîte de dialogue pour le nom de la tâche

(b) Tâche

FIGURE 3 – Ajout d'une tâche

L'utilisateur doit valider l'ajout en cliquant sur "OK". Un nouvel item de tâche apparaît dans la liste (au-dessus du bouton "+"). Nous avons choisi de procéder ainsi plutôt que de mettre un bouton de création dans l'entête de la liste, pour éviter à l'utilisateur de remonter en haut de la liste à chaque fois qu'il souhaite ajouter une tâche (cela peut vite devenir contraignant).

- **Fonctionnalité Monter et Descendre** : nous avons décidé d'accéder à cette fonctionnalité seulement grâce aux gestes. En effet, il suffit à l'utilisateur de rester longtemps appuyer sur un bouton d'une liste ou tâche et de la glisser à l'endroit voulu. (Fonctionnalité non implémentée mais expérimentée voir 7).
- **Fonctionnalité Modifier** : nous avons choisi d'ajouter cette possibilité de modification du nom sur chaque élément par la présence du bouton .
- **Fonctionnalité Supprimer** : nous avons choisi d'ajouter cette possibilité de suppression sur chaque élément (liste ou tâche) pour que l'utilisateur puisse supprimer plus rapidement. De plus nous avons décidé de faire apparaître une fenêtre d'avertissement car la suppression d'une liste entraîne la suppression de toutes ses tâches. Le bouton représentant cette fonctionnalité est le suivant .
- **Fonctionnalité Supprimer tout** : nous avons choisi d'ajouter cette possibilité de supprimer l'ensemble des listes créées par l'utilisateur afin que celui-ci n'ait pas à supprimer toutes les tâches une par une s'il souhaite vraiment tout supprimer. De plus nous avons décidé de faire apparaître une fenêtre d'avertissement car la suppression de l'ensemble des listes est une opération importante.
- **Indicateur** : nous avons choisi d'ajouter sur la liste un indicateur permettant de connaître le nombre de tâches validées sur le nombre de tâche de la liste, afin que l'utilisateur puisse voir l'état de sa liste plus rapidement.

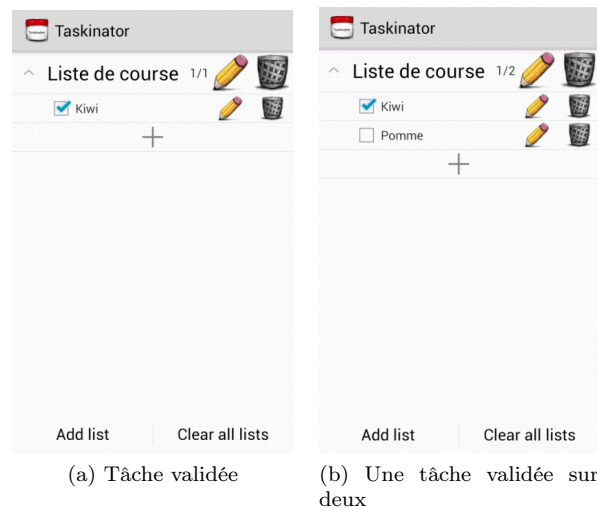


FIGURE 4 – Validation de tâche et indicateur

## 4.2 Affichage des listes et tâches

Afin d'afficher l'ensemble des listes créées et de leur(s) tâche(s), on a décidé d'utiliser `ExpandableListView`, permettant d'afficher l'ensemble des listes et pour chacune d'entre elle sa ou ses tâche(s).

La liste est décrite à l'aide du xml `list.xml` et est représentée par un `TextView` permettant d'afficher le nom de celle-ci, d'un autre `TextView` permettant d'afficher le nombre de tâches validées sur le nombre de tâche de la liste, d'un `ImageButton` permettant d'ajouter la fonctionnalité de modification du nom de la liste et d'un `ImageButton` permettant d'ajouter la fonctionnalité de suppression de liste.

La tâche est décrite par le fichier `task.xml` et est représentée par une checkbox, un `TextView` permettant de taper d'afficher le nom de la tâche, d'un `ImageButton` permettant d'ajouter la fonctionnalité de modification du nom sur la tâche et d'un `ImageButton` permettant d'ajouter la fonctionnalité de suppression sur la tâche.



## 5 Le modèle

Afin de réaliser cette application nous nous sommes basés sur le modèle suivant dont voici le diagramme de classes :

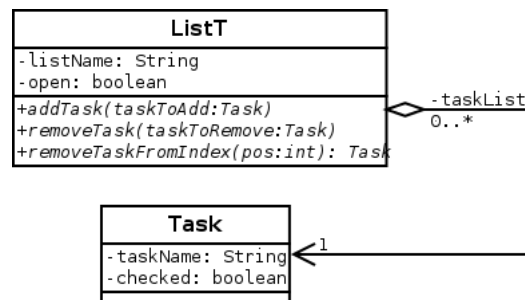


FIGURE 5 – Diagramme de classe du modèle

Le modèle de cette application est très simple. Il est composé de deux classes, *ListT* et *Task*. Nous avons donc ici une *ListT* qui va pouvoir contenir un ensemble de tâches et d'autres propriétés comme le nom et si elle est ouverte ou non. Nous avons de plus une *Task* qui va elle représenter une tâche et contenir certaines informations comme son nom et si elle est cochée ou non. Nous pouvons donc gérer toutes ces listes au sein de ce modèle et ainsi, ajouter une tâche dans celle-ci. On peut pour les listes comme les tâches les supprimer, les déplacer d'un rang (dans les deux sens).

## 6 Sauvegarde des données

Afin que l'utilisateur retrouve l'ensemble de ses données quand il quitte l'application, ou qu'il tourne son téléphone et que l'orientation change, ou qu'il reçoive un appel pendant qu'il utilise l'application ; il a été nécessaire de sauvegarder ses données. Pour cela, nous avons décidé d'utiliser l'Internal Storage. Celui-ci permet de stocker en interne des fichiers sur l'appareil et que ces fichiers soient privés pour notre application. Ces fichiers seront supprimés lors de la désinstallation de l'application par l'utilisateur.

Nous avons donc décidé de sauvegarder notre liste dans un fichier *xml* et de la restaurer via ce fichier. Cette sauvegarde s'effectue dans la fonction *onPause()*. Comme on peut le voir sur le schéma ci-dessous, dès que l'activité n'est plus au premier plan, elle passe par la fonction *onPause()*. On est donc sûr d'avoir la dernière version de la liste enregistrée. Nous devons donc maintenant restaurer cette liste à partir du fichier (que nous avons décidé d'appeler *save.xml*). Nous avons décidé d'effectuer cette restauration dans la fonction *onCreate()*. En effet à chaque nouveau lancement de l'application, on recharge la liste à partir du fichier *save.xml*.

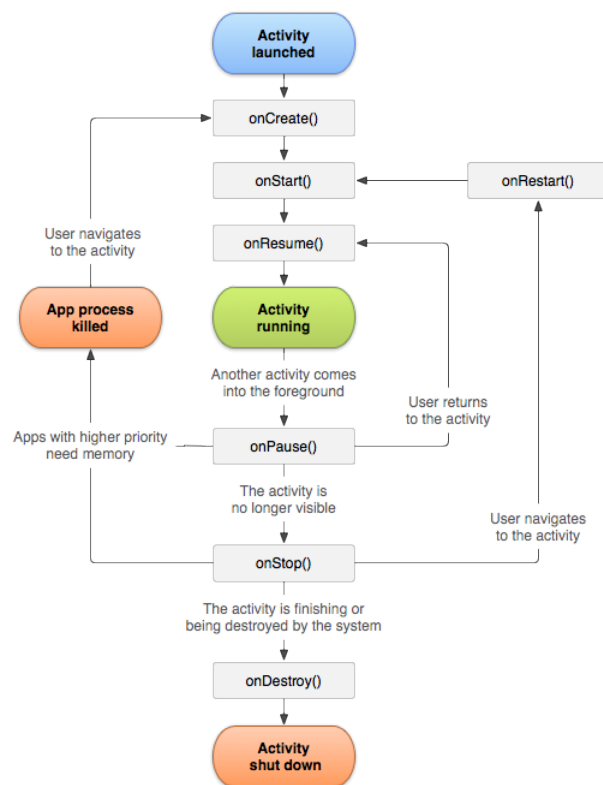


FIGURE 6 – Le cycle de vie d'une activité

Afin d'enregistrer cette liste dans le fichier *save.xml*, nous avons créé notre propre parseur. Ce parseur *xml* a été développé à partir de la librairie **JDOM** et permet de charger et sauvegarder des fichiers xml de la forme :

```

<backup>
  <list name="listName" open="true|false">
    </task name="taskName" checked="true|false"/>
  </list>
</backup>
  
```

Le parseur va parcourir le fichier et lorsqu'il rencontre une balise *list*, il crée un objet de type *ListT* et lorsqu'il rencontre une balise *task*, il crée un objet de type *Task* qu'il ajoute à la dernière liste créée.

## 7 Expérimentation

Pour rendre notre application nous avons essayé de mettre en place un système de *Swipe* pour la suppression des listes et des tâches, et un système de *Drag and Drop* pour les déplacer. Pour cela nous avons essayé d'utiliser un [Framework](#) qui permet d'appliquer des effets graphiques sur des *ListView*. Seulement notre application est basée sur des *ExpandableListView*. Ce qui ne permet son utilisation sans réécrire une bonne partie du Framework.

### 7.1 Suppression par Swipe

Le but de cette gesture aurait été de pouvoir supprimer une tâche ou une liste de tâches sans avoir de boîte de dialogue pour confirmer la suppression, ce qui ajouterait du dynamisme à notre application

### 7.2 Déplacement de listes/tâches par Drag and Drop

Dans la configuration actuel de notre application, il n'est pas possible de déplacer ni les listes, ni les tâches. Cette gesture aurait permis de remédier à cela. Le *Drag and Drop* aurait été implémenté de la manière suivante :

- Sur la liste et la tâche, aurait été affiché un bouton qui lorsque l'on appui dessus active le *Drag and Drop* et permet le déplacement de la liste ou de la tâche. L'utilisation du bouton est importante puisqu'elle empêche à l'application de confondre le *Drag and Drop* et le *Swipe*.
- Les tâches pourraient ainsi être déplacées soit à l'intérieur de la liste, soit déplacées d'une liste à un autre.

## 8 Limite de l'application

Notre application a ses propres limites. On pourrait imaginer d'ajouter certaines fonctionnalités comme par exemple :

- Recherche ( 🔍 ) : qui permettrait à l'utilisateur d'afficher seulement les listes qui ont un nom ou qui ont des tâches dont le nom contient la chaîne du filtre.
- Partage ( 📎 ) : qui permettrait à l'utilisateur de partager sa ou ses listes sur des réseaux sociaux ou par mail.
- Favoris ( ❤️ ) : qui permettrait à l'utilisateur de mettre certaines listes ou tâches en favoris et d'y avoir accès via un onglet favoris. Cela permettrait d'avoir un accès plus rapide.
- Rappel ( 📅 ) : qui permettrait à l'utilisateur de définir un rappel sur une tâche afin que celle-ci soit exécutée dans les temps.

## 9 Conclusion générale

Ce projet nous a permis de réfléchir à la structure de l'IHM afin que celle-ci soit la plus intuitive possible pour l'utilisateur. Réfléchir à la position des boutons, le nom ou l'icône les représentant, comment l'ajout des composants se fera au sein de la liste, la gesture à utiliser, l'utilisation d'un menu ... Et donc, d'orienter notre conception sur l'IHM, plus que sur le modèle, et ainsi expérimenter une autre approche de conception.