
# RAPPORT TECHNIQUE

IUT Lyon 1
l'excellence technologique

**Projet Tuteuré – G6S4 – 2016/2017**

Jackson DEROCK / Guillaume GEBAVI

Tutrice responsable – Mme CORDIER

# Sommaire

# 1. L'application GingerDo

## 1.1. Contexte

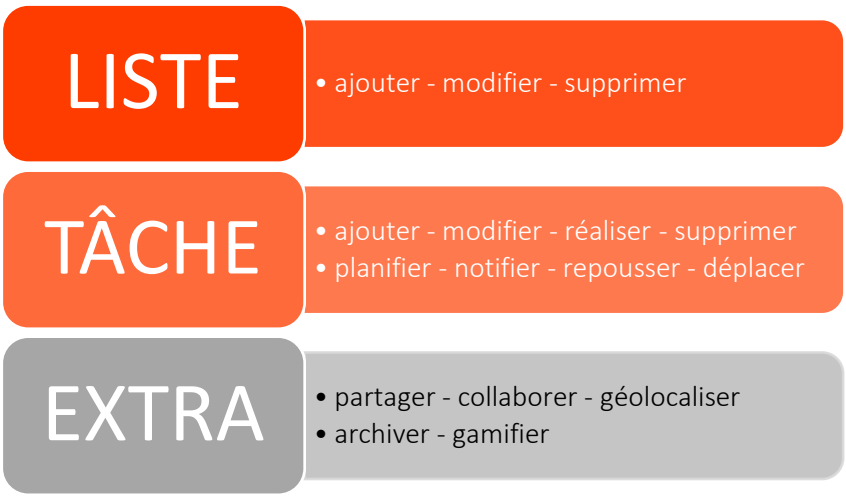
Vous êtes submergé par le travail ? Vous avez du mal à vous organiser ? Vous ne vous souvenez plus des besognes à réaliser pour la journée ? *GingerDo* est la solution à vos problèmes ! Cette application mobile Android est soucieuse de voir réaliser toutes vos tâches que vous avez prévu d'effectuer. Grâce à son interface intuitive et dynamique, rien de plus facile d'organiser ses activités et de les réaliser. Il vous suffit de créer vos propres listes et de répertorier toutes vos tâches dans chacune d'elles. Dites adieu aux longues listes griffonnées sur un vieux morceau de papier et détendez-vous en utilisant *GingerDo*.

## 1.2. La veille technologique des « ToDoList »

Avant de fixer nos objectifs, nous avons testé plusieurs applications existantes de ToDoList directement disponible depuis le *PlayStore* et l'*Apple Store* tels que *Todoist*, *Any.DO*, *Wunderlist*, *Remember The Milk*, *Notes*, *Clear* … Cette étude nous a permis de retenir les fonctionnalités essentielles d'une application ToDoList : l'ajout, la réalisation, la modification et la suppression de tâches ; tout en permettant d'éliminer celles qui nous semblaient superflues ; mais également des idées d'implémentation de fonctionnalités innovantes comme le balayage (*swipe*) pour effectuer une action sur une tâche.

## 1.3. Les objectifs de l'application

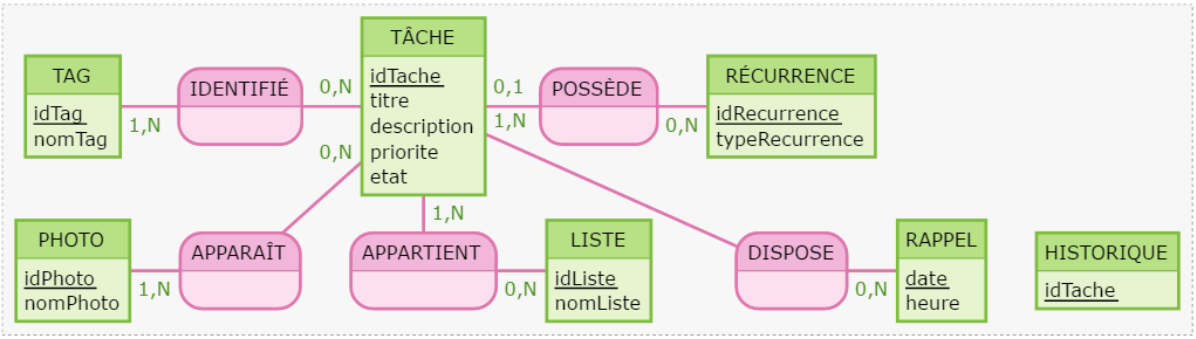
Voici donc les objectifs que nous nous sommes fixés au départ, classés en 3 parties : les fonctionnalités des parties « LISTE » et « TÂCHE » sont plus importantes que celle de la partie « EXTRA » :



# 2. Partie conception

## 2.1. Modélisation

Voici le schéma de la base de données relationnelle que nous avons établi avant de développer notre application. Cette partie fondamentale permet notamment d'assurer la persistance et l'intégrité des données du futur utilisateur de l'application.



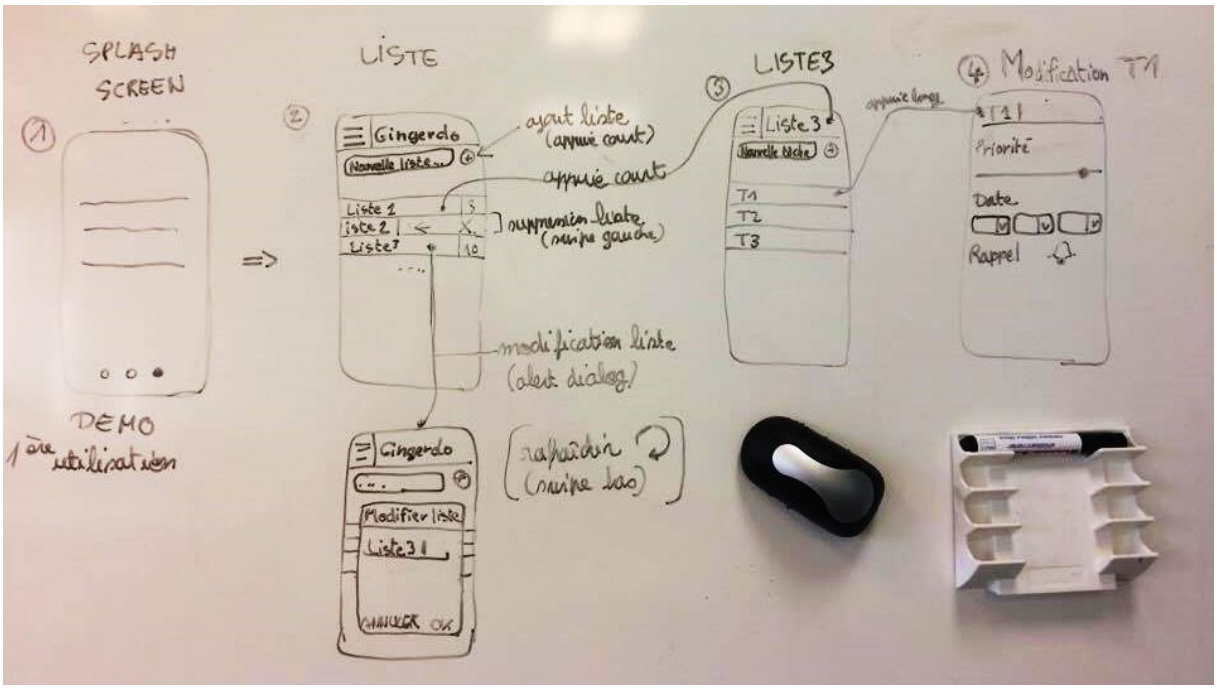

**TAG** ( idTag, nomTag )
**IDENTIFIÉ** ( *idTag*, *idTache* )
**TÂCHE** ( idTache, titre, description, priorite, etat, *idRecurrence* )
**RÉCURRENCE** ( idRecurrence, typeRecurrence )
**PHOTO** ( idPhoto, nomPhoto )
**APPARAÎT** ( *idPhoto*, *idTache* )
**APPARTIENT** ( *idListe*, *idTache* )
**LISTE** ( idListe, nomListe )
**DISPOSE** ( *date*, *idTache* )
**RAPPEL** ( date, heure )

Bien entendu, le modèle a été dégradé progressivement par la suite et réadapté en fonction de l'avancée du projet. L'affectation d'une photo ou l'ajout d'un tag pour une tâche a été mis de côté par exemple.

## 2.2. Prototype et expérience utilisateur

Concevoir un prototype a également été nécessaire pour mieux déterminer les contours de l'application mais aussi pour réfléchir aux différentes interactions de l'utilisateur, et donc de proposer un produit ergonomique à la sortie. Pour réaliser nos maquettes, nous avons donc utilisé la plateforme JustInMind.

Nous avons également pensé l'application à la main en dessinant des croquis notamment vers les dernières semaines. Cette étape nous a permis de gagner un temps considérable pour préparer et faciliter la phase de développement : tout ce qui concerne l'agencement des composants graphiques (*views*), les évènements (*events*) qui y sont associés ou encore la création des gabarits (*layouts*).




Par la suite, nous avons pu réaliser une courte vidéo de présentation où nous mettons en avant ce qui a été créé à partir de ces maquettes. La vidéo est disponible en suivant ce lien : https://www.youtube.com/watch?v=YrlXQOLcGCE

# 3. Partie développement

## 3.1. Android Studio en quelques mots

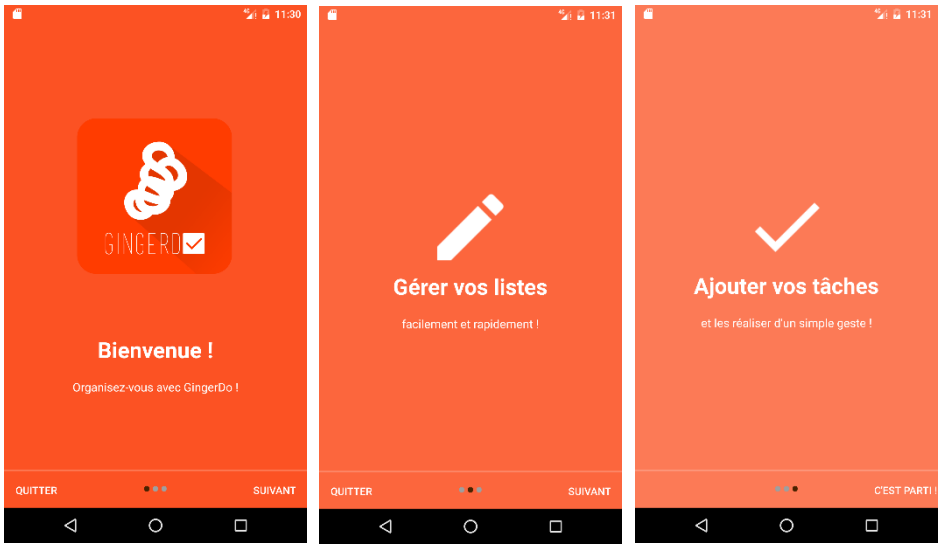
Disponible depuis 2013, Android Studio est un environnement de développement proposé par IntelliJ IDEA pour développer des applications Android. *GingerDo* a été développé en utilisant cet IDE.

## 3.2. Les différentes activités et fonctionnalités

Une activité est la composante principale d'une application Android. Elle représente l'implémentation et les interactions de nos interfaces. Ici nous avons créé 5 activités : les *splash screens*, les listes, les tâches d'une liste spécifique, la modification d'une tâche et la liste « Urgente ».

### 3.2.1. Les splash screens

Les *splash screens* apparaissent uniquement lors de la première utilisation de l'application. Ces derniers expliquent l'intérêt d'utiliser *GingerDo* en quelques étapes. L'utilisateur a également la possibilité de sauter cette partie en appuyant sur « QUITTER » pour arriver directement sur l'activité principale.



### 3.2.2. Les listes

Après les *splash screens* ou lors des prochains lancements de l'application, cette dernière aura toujours les listes comme activité d'accueil. Ici, l'utilisateur a la possibilité d'ajouter de nouvelles listes ou d'accéder à d'autres. On distingue 2 catégories de listes : la liste spéciale « Urgente » (*cf 3.2.5*) et celles de l'utilisateur « Mes listes ».

« Mes Listes » peuvent être modifiées et/ou supprimées par l'utilisateur. Les fonctionnalités implémentées ici sont :

★ L'ajout d'une nouvelle liste (1) – il suffit d'entrer le nom de la nouvelle liste au niveau de l'*EditText* puis d'appuyer sur le bouton « + ».

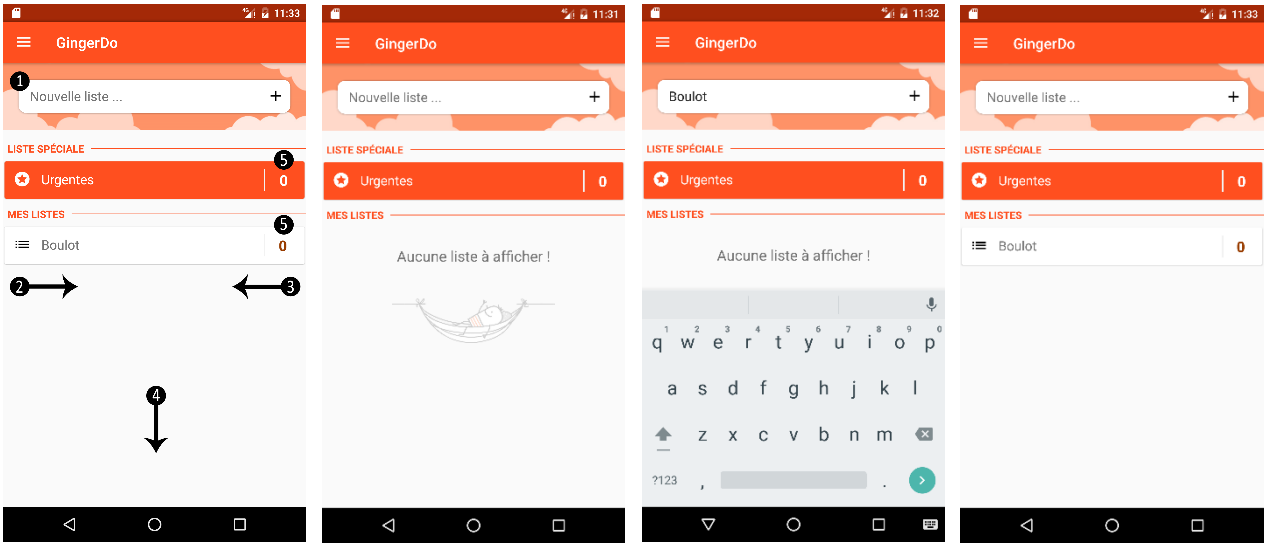
★ La modification d'une liste (2) – il suffit de *swiper* vers la droite au niveau de la liste : une boîte de dialogue (*AlertDialog*) apparaît alors pour renommer la liste en question.

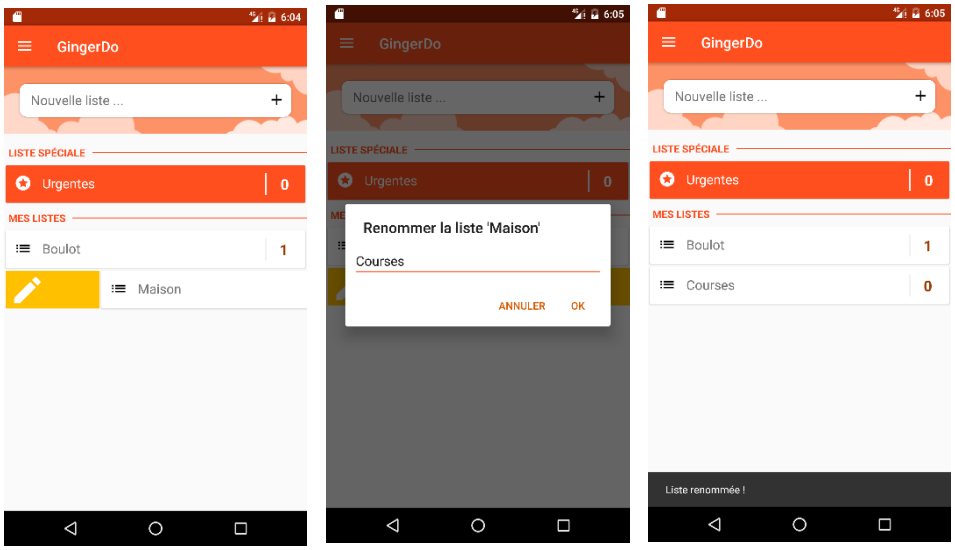
★ La suppression d'une liste (3) – il suffit d'effectuer un *swipe* vers la gauche au niveau de la liste : une boîte de confirmation de suppression apparaît alors avant d'effectuer définitivement l'action. En effet, la suppression d'une liste est irréversible, elle entraîne donc la suppression de toutes les tâches qu'elle contenait !

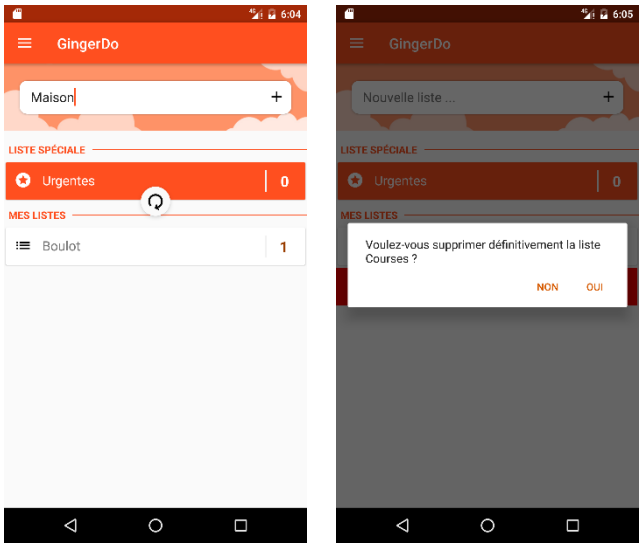
★ Le rafraîchissement des listes (4) – si le contenu est bloqué, il est fortement recommandé de rafraîchir la vue en tirant vers le bas.

★ Les compteurs (5) – situé à droite de chaque liste, ces indicateurs recensent le nombre de tâches non-réalisées.

Un tri par ordre alphabétique a été assigné pour faciliter la recherche des listes. Au niveau des saisies utilisateurs, des contrôles ont également été mis en place lorsqu'il s'agit de choisir un nom de liste, de le renommer ou s'il est vide. Par exemple, si jamais ce dernier est déjà existant en base, un message apparaît en bas de l'écran (*SnackBar*) pour l'avertir : la casse a donc son importance ici !


GingerDo
Nouvelle liste ...
LISTE SPÉCIALE
Urgentes
0
MES LISTES
Boulot
0
Aucune liste à afficher !
1
2
3
4
5


GingerDo
Nouvelle liste ...
LISTE SPÉCIALE
Urgentes
0
MES LISTES
Boulot
1
Maison
Renommer la liste 'Maison'
Courses
ANNULER
OK
Courses
0
Liste renommée !


GingerDo
Maison
Nouvelle liste ...
LISTE SPÉCIALE
Urgentes
0
MES LISTES
Boulot
1
Voulez-vous supprimer définitivement la liste Courses ?
NON
OUI

### 3.2.3. Les tâches d'une liste spécifique

Lorsque l'on appuie sur une liste en particulier, on est redirigé vers une nouvelle activité qui contient les tâches de la liste en question. Ici les fonctionnalités implémentées sont :

★ L'ajout d'une nouvelle tâche (1) – il suffit d'entrer le nom de la nouvelle tâche au niveau de l'*EditText* puis d'appuyer sur le bouton « + ». Ici il n'y a pas de restriction, plusieurs tâches peuvent présenter le même nom.

★ La modification d'une tâche (2) – il suffit de *swiper* vers la droite au niveau de la tâche : l'utilisateur est redirigé vers une nouvelle activité. (*cf. 3.2.4*)

★ La réalisation d'une tâche (3) – il suffit d'appuyer sur la *CheckBox* située à droite de la tâche pour changer son état a « réalisé » : la tâche apparaît barrée. En appuyant de nouveau, cette dernière retrouve son état d'origine c'est-à-dire « non-réalisé ».

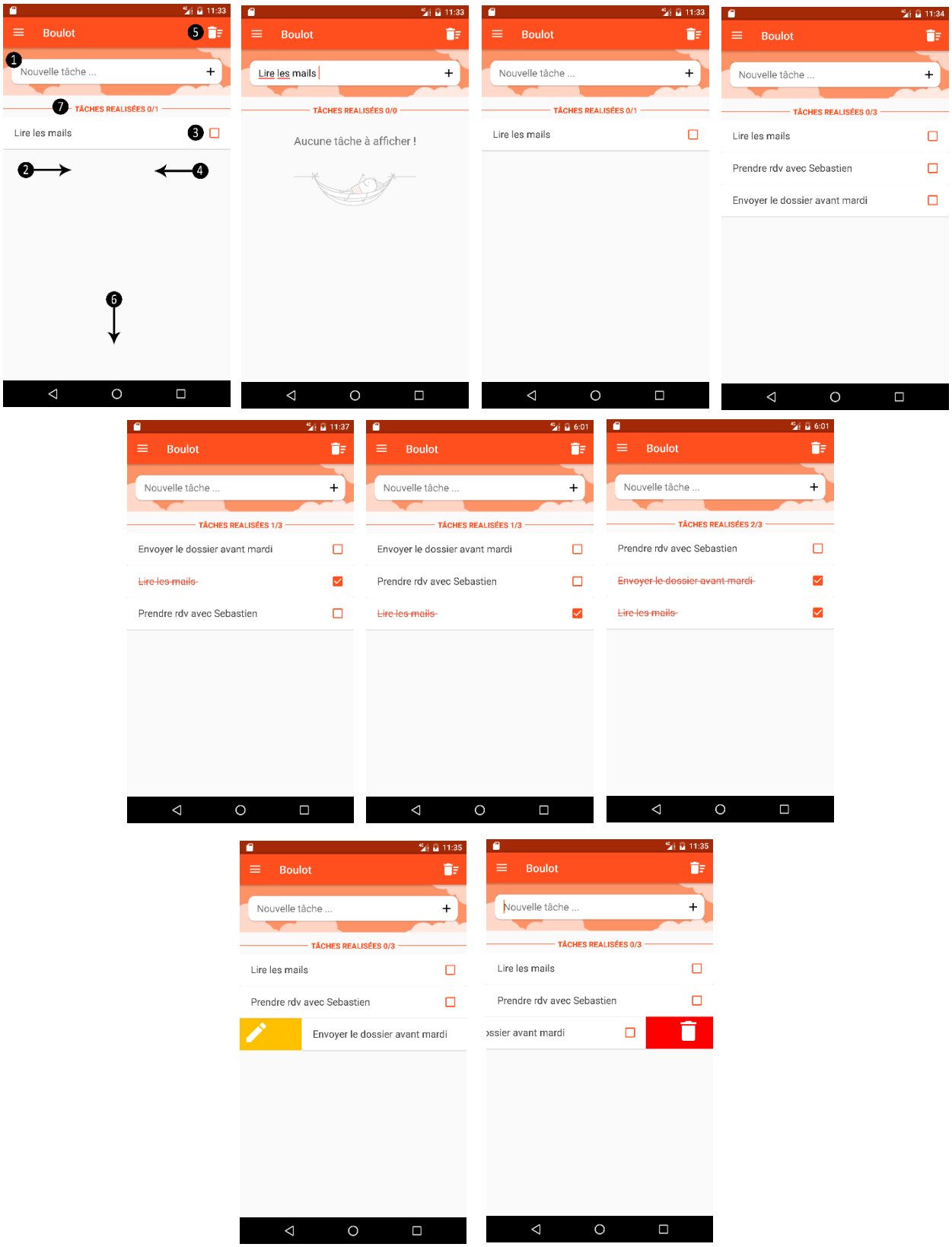
★ La suppression d'une tâche (4) – il suffit d'effectuer un *swipe* vers la gauche au niveau de la tâche.

★ La suppression de tâches réalisées (5) – il suffit d'appuyer sur le bouton « poubelle » en haut à droite au niveau de l'*ActionBar* pour les supprimer définitivement.

★ Le rafraîchissement de la liste spécifique (6) – si le contenu est bloqué, il est fortement recommandé de rafraîchir la vue en tirant vers le bas.

★ Le compteur (7) – situé en haut sous forme fractionnaire, il indique le nombre de tâches réalisées sur le nombre total de tâches que contient la liste en fonction des actions d'ajout, de réalisation et de suppression effectuées par l'utilisateur.

Au niveau de l'affichage d'une liste de tâches, un tri est effectué en fonction de l'état de la tâche, à savoir réalisé ou non ainsi qu'en fonction de sa priorité. Une tâche réalisée sera placée en bas de liste. A contrario, en tête de liste si elle n'a pas encore été réalisée. Les tâches sont aussi classées par ordre de priorité : les plus importantes étant situées en haut et inversement.


Boulot
Nouvelle tâche ...
TÂCHES REALISÉES 0/1
Lire les mails
Lire les mails
TÂCHES REALISÉES 0/0
Aucune tâche à afficher !
TÂCHES REALISÉES 0/1
Lire les mails
TÂCHES REALISÉES 0/3
Lire les mails
Prendre rdv avec Sebastien
Envoyer le dossier avant mardi
TÂCHES REALISÉES 1/3
Envoyer le dossier avant mardi
Lire les mails
Prendre rdv avec Sebastien
TÂCHES REALISÉES 1/3
Envoyer le dossier avant mardi
Prendre rdv avec Sebastien
Lire les mails
TÂCHES REALISÉES 2/3
Prendre rdv avec Sebastien
Envoyer le dossier avant mardi
Lire les mails
TÂCHES REALISÉES 0/3
Lire les mails
Prendre rdv avec Sebastien
Envoyer le dossier avant mardi
TÂCHES REALISÉES 0/3
Lire les mails
Prendre rdv avec Sebastien

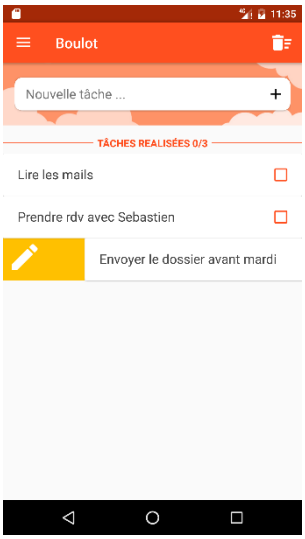
### 3.2.4. La modification d'une tâche

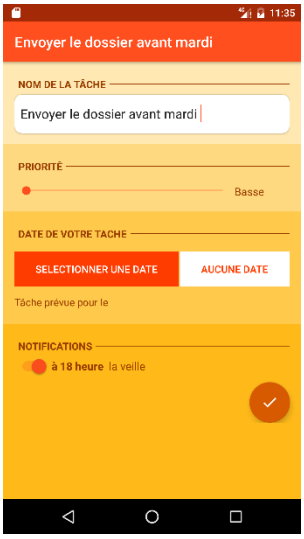
Au niveau de cette activité, l'utilisateur peut modifier le nom de sa tâche. Il peut aussi ajouter des options, lui définir une priorité en déplaçant le curseur (*seekBar*) : basse, normale ou haute. Attribuer une date en sélectionnant dans le calendrier (*datePicker*) ou encore activer la notification (*switch*).

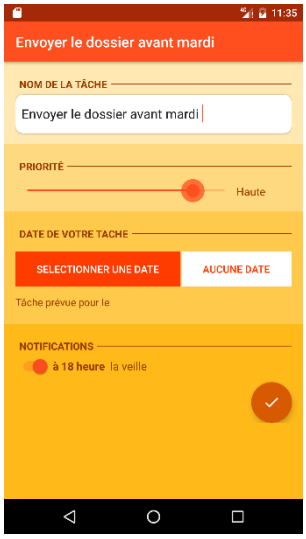
★ Définir une priorité influence la position de la tâche en question au niveau de la liste dont elle appartient.

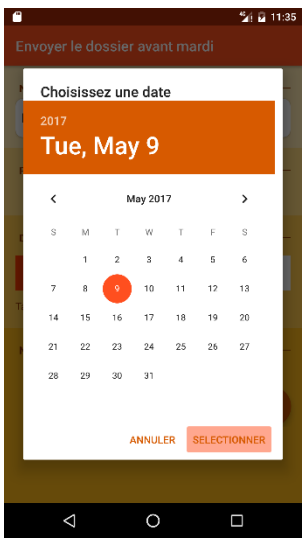
★ Définir une date permet de retrouver la tâche au niveau de la liste spéciale « Urgente » le jour J, en plus d'être répertoriée dans sa liste respective.

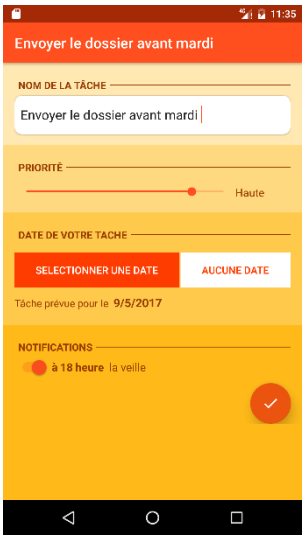
★ Définir un rappel permet de recevoir une notification à 18H la veille, à condition d'avoir choisi une date au préalable.

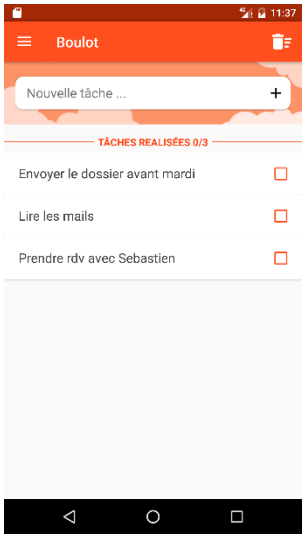


### 3.2.5. La liste « Urgente »

Cette liste ne peut être modifiée, ni supprimée. Elle sert à visualiser les tâches du jour. En revanche, la réalisation et la modification d'une tâche est possible au niveau de cette liste. Les tâches que l'utilisateur réalisera, modifiera ou supprimera, se répercuteront donc dans les listes respectives qu'il a créé initialement. Ici les fonctionnalités implémentées sont :

- ★ La modification d'une tâche (1) – il suffit de *swiper* vers la droite au niveau de la tâche : l'utilisateur est redirigé vers une nouvelle activité. (*cf. 3.2.4*)

- ★ La réalisation d'une tâche (2) – il suffit d'appuyer sur la *CheckBox* située à droite de la tâche pour changer son état a « réalisé » : la tâche apparaît barrée. En appuyant de nouveau, cette dernière retrouve son état d'origine c'est-à-dire « non-réalisé ».

- ★ La suppression d'une tâche (3) – il suffit d'effectuer un *swipe* vers la gauche au niveau de la tâche.

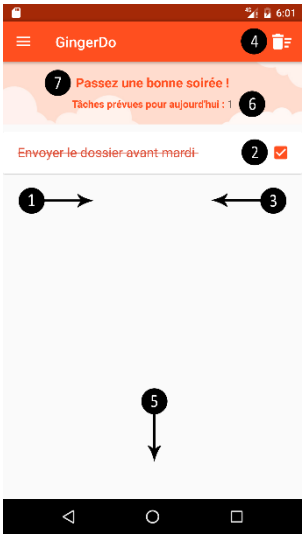
- ★ La suppression de tâches réalisées (4) – il suffit d'appuyer sur le bouton « poubelle » en haut à droite au niveau de l'*ActionBar* pour les supprimer définitivement.

- ★ Le rafraîchissement de la liste « Urgente » (5) – si le contenu est bloqué, il est fortement recommandé de rafraîchir la vue en tirant vers le bas.

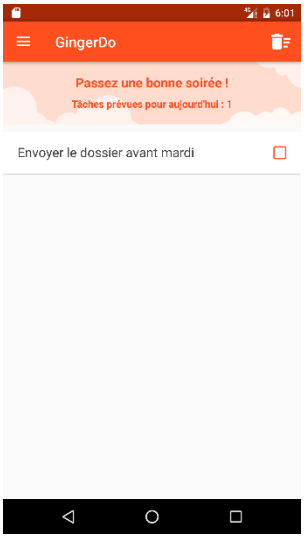
- ★ Le compteur (6) – Il indique le nombre de tâches total prévu pour aujourd'hui.

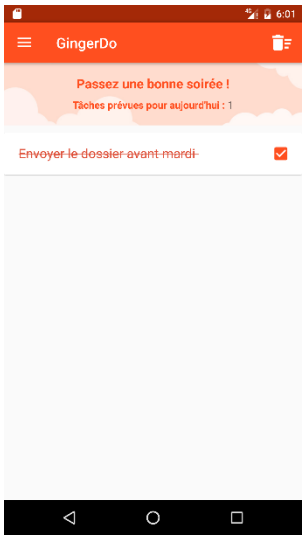
- ★ Le message (7) – Ce dernier s'adapte en fonction de l'heure : à partir de 18H, il affichera « Passez une bonne soirée », le reste du temps « Passez une bonne journée ».

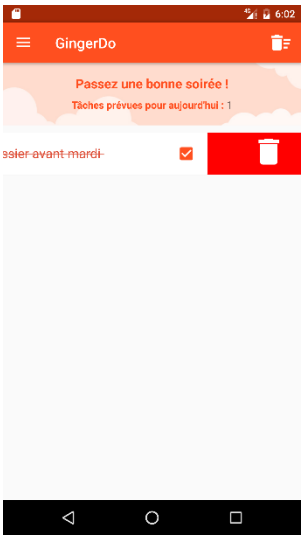
L'affichage de la liste est identique au tri effectué au niveau des tâches d'une liste spécifique. (*cf 3.2.3*)


GingerDo
Passez une bonne soirée !
Tâches prévues pour aujourd'hui : 1
Envoyer le dossier avant mardi
1
2
3
4
5
6
7


GingerDo
Passez une bonne soirée !
Tâches prévues pour aujourd'hui : 0
Aucune tâche à afficher !

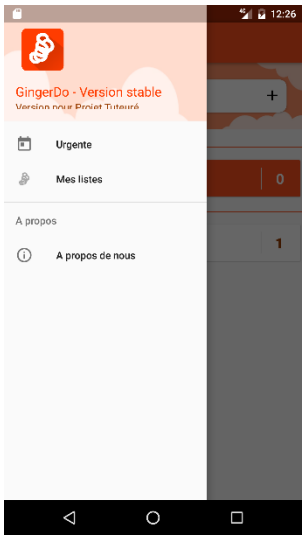

GingerDo
Passez une bonne soirée !
Tâches prévues pour aujourd'hui : 1
Envoyer le dossier avant mardi

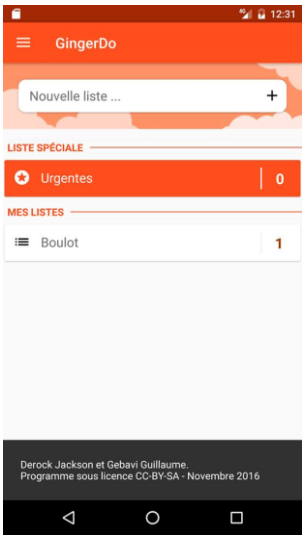

GingerDo
Passez une bonne soirée !
Tâches prévues pour aujourd'hui : 1
Envoyer le dossier avant mardi


GingerDo
Passez une bonne soirée !
Tâches prévues pour aujourd'hui : 1

### 3.3. Le menu pour la navigation

Pour faciliter la navigation, l'utilisateur peut accéder à tout moment au menu en appuyant sur l'icône en haut à gauche au niveau de l'*ActionBar*. Ce dernier propose un accès à la liste « Aujourd'hui » ainsi qu'à « Mes listes ». Lorsque l'on clique sur « A propos de nous », un message apparaît en bas d'écran (*SnackBar*) pour rappeler nos noms et la licence choisie.




### 3.4. La base de données avec SQLite

La mise en place de la base de données occupe une place importante pour assurer la persistance et les contraintes d'intégrité des données de l'utilisateur. D'autres choix s'offraient à nous, par exemple la possibilité d'utiliser les préférences partagées ou la sérialisation. La base de données est plus flexible et modulable : SQLite étant une base de données relationnelle, il est plus facile de lier et d'organiser nos données. Par exemple lorsqu'il s'agit d'ajouter une nouvelle table en milieu de projet.

# 4. Gestion du projet

## 4.1. Méthode agile

Nous avons opté pour la méthode Scrum plutôt qu'une méthode traditionnelle en V. L'application de cette méthodologie de travail nous a permis de nous focaliser sur une ou plusieurs fonctionnalités en particulier et de nous assurer qu'elle(s) fonctionne(nt), avant de passer à d'autres. Ainsi nous pouvons déceler rapidement les dysfonctionnements.

## 4.2. Découpage du projet en activités

Nous avons découpé notre projet en phase comme ceci :

- La phase de **conception** avec une attention particulière à l'aspect et l'ergonomie de l'application en tenant compte de l'expérience utilisateur. Le modèle a été établit au tout début mais nous avons tout de même ajouté des correctifs au fur et à mesure de l'avancement du projet.

- La phase de **développement** reste la plus importante et la plus longue. Nous avons séparé les tâches : une personne s'est occupée en majeure partie de l'aspect visuel en agençant les différents composants graphiques et en créant les différents *layouts* ; tandis qu'une autre a implémenté les évènements de chacun de ces composants sans oublier de mettre en place la base de données sous SQLite pour assurer la persistance des données.

- La phase **finale**, étape non négligeable avec les derniers tests et correctifs avant le déploiement de l'application même si des tests ont été effectués en amont à chaque itération.

## 4.3. Gestion du temps

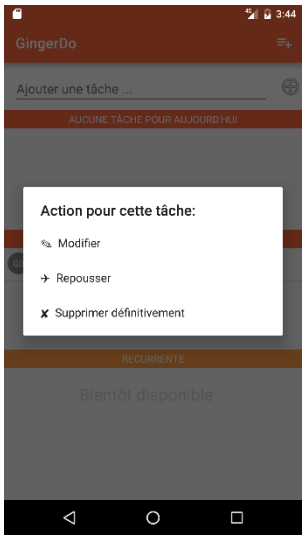
Nous n'avons pas établi de planning strict à l'avance pour nous laisser une certaine liberté et flexibilité de travail. Nous nous sommes donc réunis pendant notre temps libre et pendant les vacances pour pouvoir avancer le projet. Le déroulement d'une réunion se base toujours sur la méthode Scrum : avant de commencer, on fait un petit bilan pour situer l'état d'avancement du projet pour ainsi définir les objectifs à atteindre à la fin de la réunion.

## 4.4. Refonte majeure

Suite à la première soutenance au semestre 3, nous avons pris la décision de revoir l'application dans son ensemble avec l'utilisation d'un *RecyclerView* à la place d'une *ListView* par exemple pour la gestion des listes. Le module d'application mobile Android dispensé au semestre 4, nous a également permis de mettre en place quelques bonnes pratiques en matière d'organisation du code. Cette refonte n'est pas sans conséquence, nous avons donc revu nos priorités et objectifs en termes de fonctionnalités en fonction du temps qu'il nous restait.

Voici quelques captures de notre ancienne version *GingerDo* présentée au semestre 3 :






## 4.5. Gestion de version

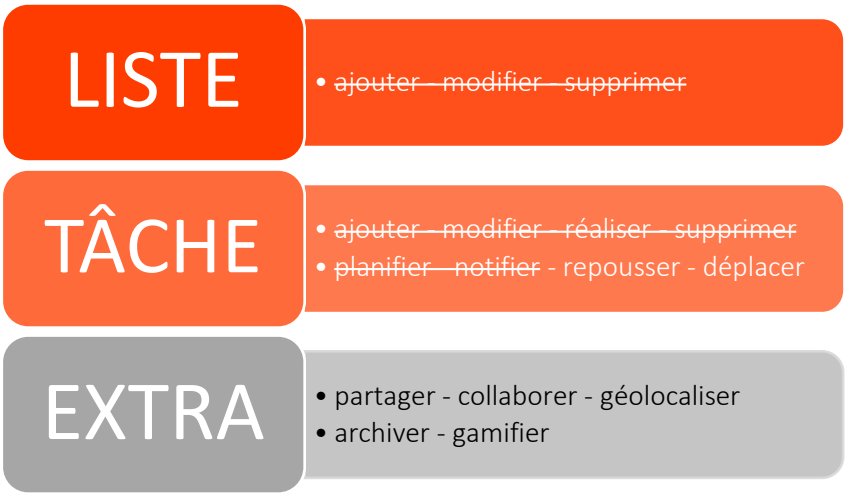
Pour garder un historique et visualiser les changements de notre projet, nous avons aussi mis en place un dépôt sur GITHub. Ce dernier est disponible en suivant ce lien : https://github.com/neexs/GingerDo-S4

# 5. Bilan du projet

## 5.1. Objectifs atteints ?

Finalement, pour reprendre nos objectifs de départ, nous avons implémenté l'ajout, la modification et la suppression de listes ainsi que de tâches avec la fonctionnalité de réalisation en plus pour cette dernière. Sans oublier, les options permettant la gestion des dates et des notifications. Les autres fonctionnalités non-implémentées sont celles qui n'ont pas été barrées.

La partie « EXTRA » comme son nom l'indique propose des fonctionnalités en plus. Nous ne nous sommes donc pas attardés sur cette partie et avons préféré implémenter au maximum les fonctions essentielles des parties « LISTE » et « TÂCHE » pour avoir un produit fonctionnel à la sortie.



## 5.2. Conclusion

En définitive, ce projet tuteuré de fin d'étude étalé sur 2 semestres a été un véritable travail de longue haleine. Ce dernier nous a notamment permis de mettre en pratique les notions acquises en gestion de projet et les méthodes de travail enseignées en Cycle de Vie Des Applications : à savoir la démarche d'un projet dans son intégralité en partant de la phase de conception jusqu'à son aboutissement en passant par la phase de développement.

Cela a également été un véritable défi pour nous deux, car il nous a fallu apprendre de zéro le développement mobile sous Android au semestre 3. La gestion du temps est également un facteur à prendre en compte.

En ce qui concerne *GingerDo*, nous sommes assez fiers de notre application même si toutes les fonctionnalités n'ont pas été implémentées. En effet, nous avons consacré énormément de temps à réfléchir à l'ergonomie de l'application pour qu'elle reste simple, fonctionnelle et intuitive pour l'utilisateur. Ce souci du détail nous a orienté vers une refonte majeure entre le semestre 3 et le semestre 4.

Enfin avoir la liberté de choisir un sujet et de le réaliser en binôme a également été un parti pris que nous avons particulièrement apprécié. Ce choix de groupe a été rendu possible grâce à notre tutrice Mme Amélie CORDIER que nous tenons d'ailleurs à remercier sincèrement.

## 5.3. Devenir de l'application

Le projet *GingerDo* ne s'arrête pas ici ! L'idée est de poursuivre le développement de l'application dans le but de l'enrichir et d'atteindre tous les objectifs. La finalité reste de publier *GingerDo* sur le *PlayStore* afin de la rendre disponible gratuitement au public.

Concernant la licence de distribution de notre projet, nous avons choisi de mettre le code en libre accès sur la plateforme GitHub (*cf 4.5*) grâce à la licence BY-NC-SA qui précise que : *« cette licence permet aux autres de remixer, arranger, et adapter votre œuvre à des fins non commerciales tant qu'on vous crédite en citant votre nom et que les nouvelles œuvres sont diffusées selon les mêmes conditions. »*

## 6. Annexes

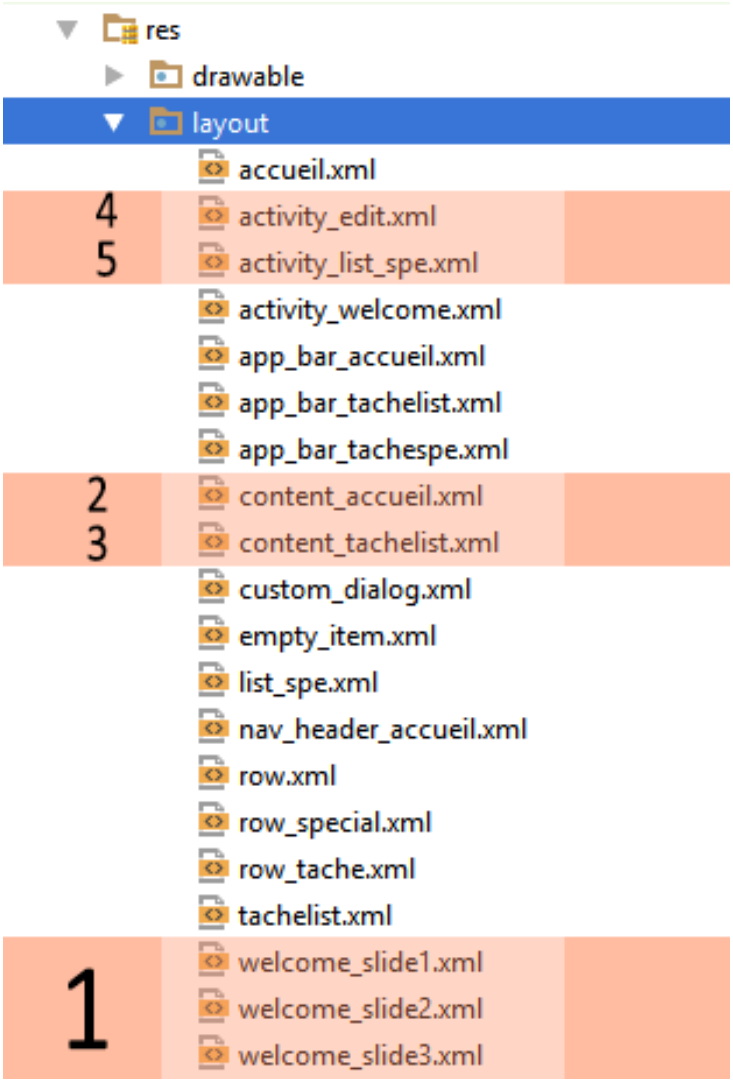
Synthèse de la structure de notre projet par ressources :

| Type de fichier | Utilisation | Nombre de fichiers |
|---|---|---|
| XML | Layouts | 20 |
| Java | Classes et activités | 21 |
| Images | Icones, fond | 39 |

La taille de notre projet (fichiers sources) : 110 MB

Taille du fichier compilé (apk) : 3,9 Mb

Version compatible : Android 6.0 (Marshmallow)



## 6.1. Les fichiers XML

Notre projet garde les fichiers XML dans un dossier générique nommé « *Layout* » ou bien calque en français. Ces fichiers sont l'essence visuelle de toute application Android. Ils permettent de structurer les composants graphiques qui font d'une application ce qu'elle est.

Sur l'image ci-dessus :

★ Les fichiers XML notés **1** correspondent aux *splash screens*. Nous en avons 3 consécutifs (*cf 3.2.1*). Ce sont ces 3 fichiers XML qui contiennent nos textes et nos logos.

★ Les fichiers notés **2, 3, 4 et 5** correspondent aux activités accueil avec la liste spéciale « Urgente » et les listes utilisateurs « Mes listes », les tâches d'une liste spécifique, les tâches de la liste « Urgente », ainsi que l'activité modification d'une tâche.

★ Les autres *layouts* correspondent au menu de navigation de gauche (*Navigation Drawer*), à la forme d'un item d'une liste spécifique et d'une tâche spécifique.

## 6.2. Les classes Java

| Classe | Détails sur la classe |
|---|---|
| WelcomeActivity | Activité |
| AccueilActivity | Activité |
| TacheListe | Activité |
| TacheListeSpecial | Activité |
| EditActivity | Activité |
| Tache | Classe unique |
| AdapterList | Adaptateur |
| AdapterListSpecial | Adaptateur |
| AdapterTache | Adaptateur |
| AdapterTacheSpecial | Adaptateur |
| ListTouchHelper | TouchHelper |
| ListSpecialTouchHelper | TouchHelper |
| TacheTouchHelper | TouchHelper |
| TacheSpeTouchHelper | TouchHelper |
| TacheManager | Base de données |
| ListManager | Base de données |
| MySQLite | Base de données |
| ClickListener | Interface |
| AlarmReceiver | Ecouteur d'alarme |
| PrefManager | Préférences |

★ Activité

On a 5 activités qui correspondent aux 5 différentes interfaces de notre application. Ces classes implémentent toutes une méthode « *OnCreate* » qui sera automatiquement appelée par l'application : c'est le point d'entrée de chaque activité. C'est ici que se trouve la partie essentielle du code.

★ Classe unique

Notre classe « Tache » va nous permettre de créer des objets « tâches » qui contiendra toutes les informations d'une tâche (id, titre, date etc…). Ces informations seront manipulées par les activités et la base de données. Par exemple, lors d'un clic sur une tâche, une activité va sélectionner notre objet tâche et l'envoyer à une autre activité pour la modifier. Sans cette classe, l'envoi devient plus complexe.

★ Adaptateur

Ces classes sont importantes car elles vont faire le lien entre les items d'une liste : on parle ici de la liste en tant que composant visuel (*RecyclerView*) et les évènements correspondants. Ces dernières vont également jouer le rôle d'adaptateur visuel, c'est-à-dire définir l'apparence d'un item d'une liste (ajouts de texte, *CheckBox* etc…). Les adaptateurs vont donc permettre d'écouter les évènements qui correspondent à un item et d'y effectuer des actions particulières.

★ Touch Helper

Ce sont des classes qui vont définir les évènements à effectuer lorsque l'on effectue un mouvement particulier sur un item d'une tâche. C'est une classe complémentaire non nécessaire mais qui nous a permis d'améliorer très fortement l'ergonomie de notre application. C'est elle qui va notamment implémenter le *swipe*.

★ Base de données

C'est grâce à ces classes que nous arrivons à sauvegarder les données, même après l'arrêt du téléphone. C'est aussi grâce à elles que nous parvenons à récupérer des informations, ajouter des listes et des tâches, compter le nombre de tâches réalisées... La classe MySQLite va donc créer la base de données et les tables correspondantes tandis que les classes TacheManager et ListManager vont implémenter les méthodes de manipulations.

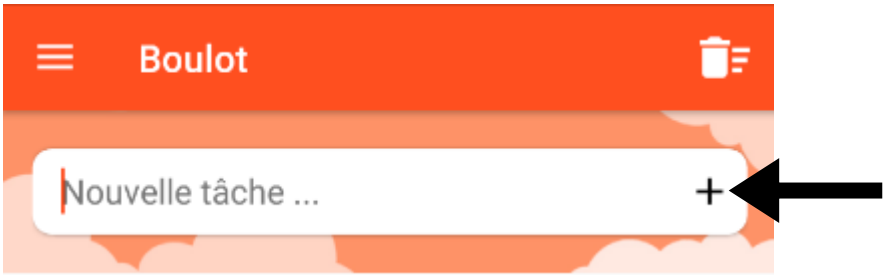
★ Autres classes

Elles sont toutes aussi importantes. ClickListener est une *interface java* au sens technique du terme qui va nous permettre de définir les méthodes qui géreront les évènements liés au clic (court, long sur un élément etc…). AlarmReceiver va écouter les évènements liés aux notifications, même lorsque l'application est éteinte. Enfin, PrefManager va forcer les *splash screens* à s'ouvrir uniquement lors de la première prise en main de l'application.

## 6.3. Explication du code

```
ImageButton addTache1 = (ImageButton) findViewById(R.id.createTache);
addTache1.setOnClickListener((v) → {

        TacheManager tm = new TacheManager(Tachelist.this);

        switch(v.getId()) {

            case R.id.createTache:

                if(et.getText().toString().isEmpty()){

                    Snackbar snackbar = Snackbar.make(mRecyclerView, "Vous devez entrer un nom de tâche !", Snackbar.LENGTH_LONG);
                    snackbar.show();

                    return;
                }

                int max = tm.getMaxTache() + 1;
                title = bundle.getString("title");

                Tache t = new Tache(max,et.getText().toString(),0,null,0,title,0);

                tm.open();
                tm.addTache(t);

                listeTache.add(t);

                mRecyclerView.setAdapter(mAdapterTache);
                mAdapterTache.notifyDataSetChanged();

                InputMethodManager im = (InputMethodManager) Tachelist.mContext().getSystemService(Context.INPUT_METHOD_SERVICE);
                im.hideSoftInputFromWindow(et.getWindowToken(), InputMethodManager.HIDE_NOT_ALWAYS);

                getCounter(title);

                et.setText("");

                break;

        }
});
```

La partie du code ci-dessus montre l'étape de création d'une tâche. Nous nous situons donc dans l'activité « TacheListe » et nous ajoutons un écouteur d'évènement, qui va écouter dans notre cas, un clic sur le bouton « + » :



Ce que nous voulons, c'est que lorsque l'utilisateur appuie sur ce bouton, le contenu écrit dans le champ prévu devienne le titre de la tâche et s'ajoute dans notre vue en bas. Nous voulons aussi ajouter cette même tâche dans notre base de données pour ne pas la perdre.

Après la saisie du titre de la tâche, on vérifie que le champ n'est pas vide. Si ce dernier l'est, un message d'erreur apparaît puis on sort de la méthode. Sinon on poursuit en définissant les champs de notre tâche :

| | |
|---|---|
| **ID** | On récupère l'ID de la dernière tâche en base et on incrémente de 1 |
| **Titre** | On récupère le titre de la tâche qui a été saisi par l'utilisateur |
| **Priorité** | Initialisé à 0 par défaut, modifiable en swipant à droite |
| **Date** | Initialisé à « null » par défaut, modifiable en swipant à droite |
| **Rappel** | Initialisé à « null » par défaut, modifiable en swipant à droite |
| **Catégorie** | Correspond à la liste dans laquelle la tâche a été écrite. Dans notre exemple « Boulot » |
| **Réalisé** | Initialisé à 0 par défaut ; état initial d'une tâche non-réalisée ; 1 si réalisée |

Notre tâche a été créée ! Nous devons maintenant l'ajouter dans la base de données.

On ouvre notre base en écriture grâce à « *TacheManager.open()* » et on l'ajoute grâce à « *TacheManager.add()* ».

Il faut maintenant l'ajouter dans la liste correspondante à l'activité, c'est-à-dire « *listeTache* ».

« *mRecyclerView* » étant le composant qui va contenir la liste, on va lui assigner notre adaptateur « *mAdapterTache* » et actualiser la vue pour que notre tâche s'affiche.

Afin d'avoir un ajout propre, le reste du code permet de cacher le clavier Android après l'ajout, de mettre à jour les compteurs sur la page et de vider le champ pour permettre la saisie d'une nouvelle tâche.

On présente ici 2 méthodes de la classe TacheManager. L'intérêt ici est de voir comment s'effectue une suppression et une récupération de tâche dans une base de données.

```
public static int supTacheFromList(String nomListe){
    String where = COLUMN_LISTETACHE + " = ?";
    String[] whereArgs = {nomListe+""};

    return db.delete(TABLE_TACHE, where, whereArgs);
}
```

Cette méthode va supprimer toutes les tâches qui se trouvent dans la liste « nomListe ». On aurait bien pu écrire une requête linéaire grâce à la méthode rawQuery, nous perdons cependant tout l'intérêt de cette méthode.

Il faut donc lire : « DELETE FROM TABLE_TACHE WHERE LISTETACHE=nomListe ». Cette méthode nous renvoie 0 si aucun élément correspondant à nomListe n'a été trouvé ou 1 à contrario.

```
public Tache getTache(int id){
    Tache t = new Tache();

    Cursor c = db.rawQuery("SELECT * FROM " + TABLE_TACHE + " WHERE " + COLUMN_IDTACHE + " = " + id , null);
    if(c.moveToFirst()){
        t.setIdTache(c.getInt(c.getColumnIndex(COLUMN_IDTACHE)));
        t.setTitreTache(c.getString(c.getColumnIndex(COLUMN_TITRETACHE)));
        t.setPrioriteTache(c.getInt(c.getColumnIndex(COLUMN_PRIORITETACHE)));
        t.setDateTache(c.getString(c.getColumnIndex(COLUMN_DATETACHE)));
        t.setRappelTache(c.getInt(c.getColumnIndex(COLUMN_RAPPELTACHE)));
        t.setCategorieTache(c.getString(c.getColumnIndex(COLUMN_LISTETACHE)));
        t.setRealise(c.getInt(c.getColumnIndex(COLUMN_DONE)));
        c.close();
    }
    return t;
}
```

Cette méthode-ci va récupérer une tâche en fonction de son identifiant. Grâce à un curseur, nous allfons donc parcourir une seule fois le résultat retourné par notre requête rawQuery et assigner à notre tâche t les valeurs que retourne cette requête.

# 7. Références


LIVRE

★ Android 5 - Les fondamentaux du développement d'application Java par Nazim BENBOURAHLA

SITES WEB

★ Androidhive
★ Developer Android
★ OpenClassroom
★ StackOverflow
★ Vogella