Oublie Pas

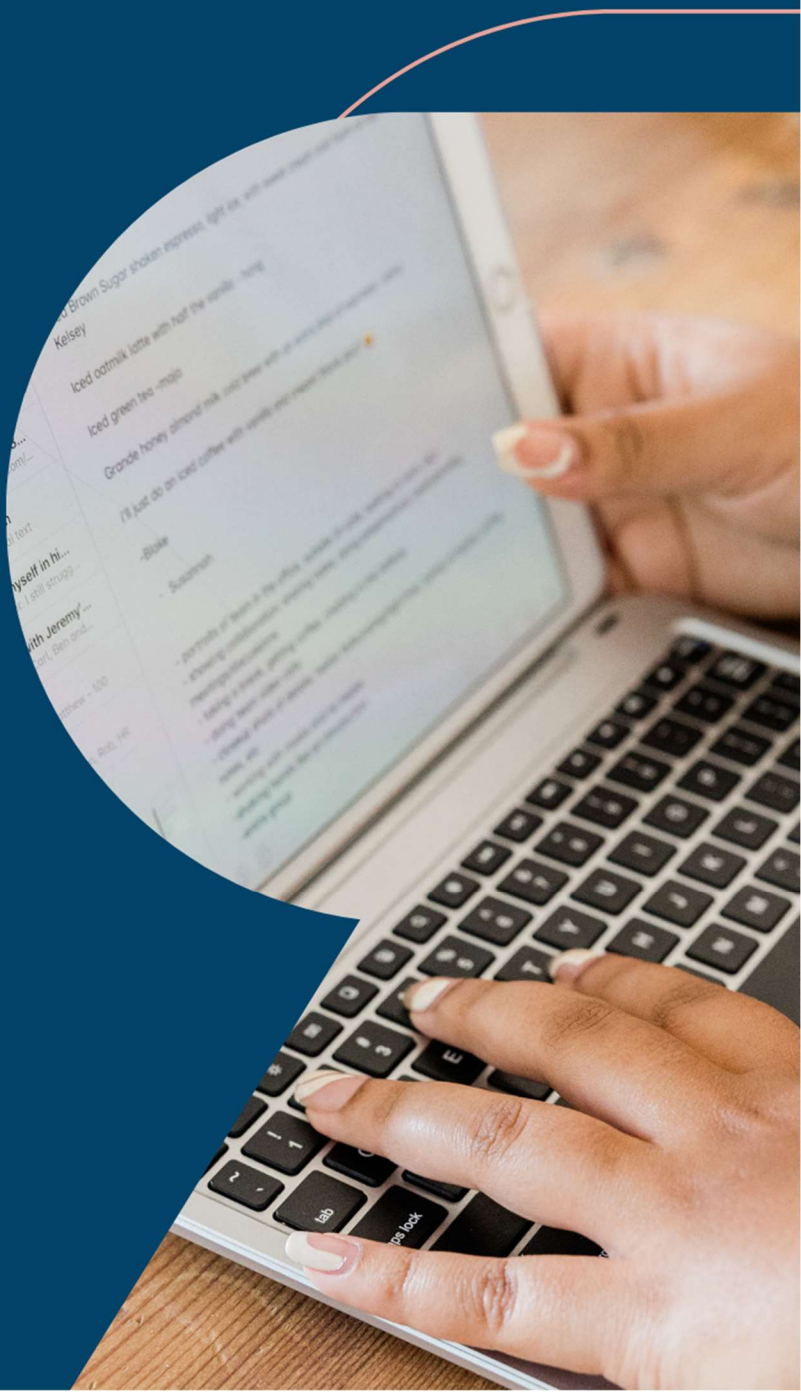
Dossier de projet

PAR
Valentin Wilhem

CHEF DE PROJET
Julien Savary

EXPERTS
Roberto Ferrari
Daniel Berney

MAI 2024



1 Table des matières

2 Analyse préliminaire

2.1 Introduction

"Oublie-pas" est une application mobile, réalisée dans le cadre de mon TPI, conçue pour aider les utilisateurs, en particulier ceux qui ont des difficultés d'organisation liées au TDAH, à définir et à suivre leurs objectifs quotidiens. L'application permet de créer des listes d'objectifs personnalisables pour la journée et de définir des rappels à des heures spécifiques pour encourager la régularité et la discipline dans l'accomplissement des tâches.

2.2 Objectifs

- L’application doit être codée en Kotlin.
- L’utilisateur devra pouvoir créer des objectifs et des rappels, composés d’un titre et d'une description.
- L’utilisateur doit pouvoir choisir s’il veut un rappel ou non.
- Les objectifs doivent être catégorisés.
- Les catégories doivent être personnalisables.
- Les objectifs auront différents statuts en fonction de si le rappel est dépassé ou non.
- Les objectifs doivent être affichés sous forme de liste avec uniquement le titre et la date.
- Les objectifs terminés doivent être séparés par un menu.
- Le tri des objectifs doit pouvoir se faire selon :
    - La date du rappel
    - La date de création
    - Leur statut
- L’utilisateur doit pouvoir afficher le détail d’un objectif en cliquant dessus.
- Toutes les informations d’un objectif doivent pouvoir être modifiées par l’utilisateur.
- L’utilisateur doit pouvoir supprimer un objectif.
- La couleur et la police des objectifs doivent être affichées différemment selon la date de rappel.
- Les objectifs doivent être sauvegardés dans une base de données.

2.3 Planification initiale

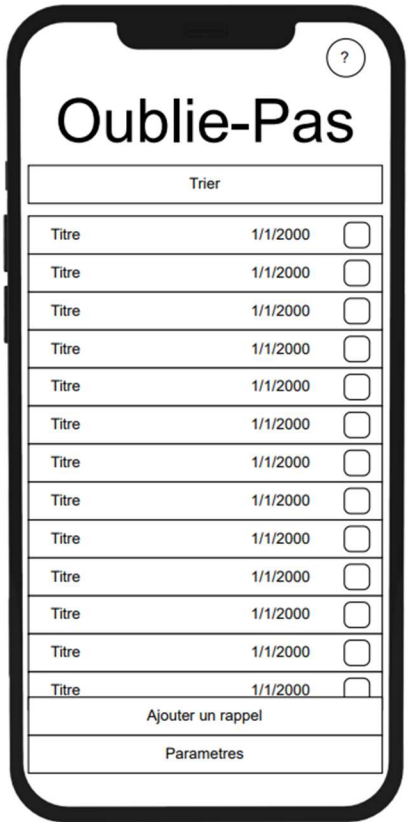
En raison de certains problèmes avec l'application Microsoft Project, le nombre d'heures total affiché pour le projet TPI et la documentation est de 92 heures.

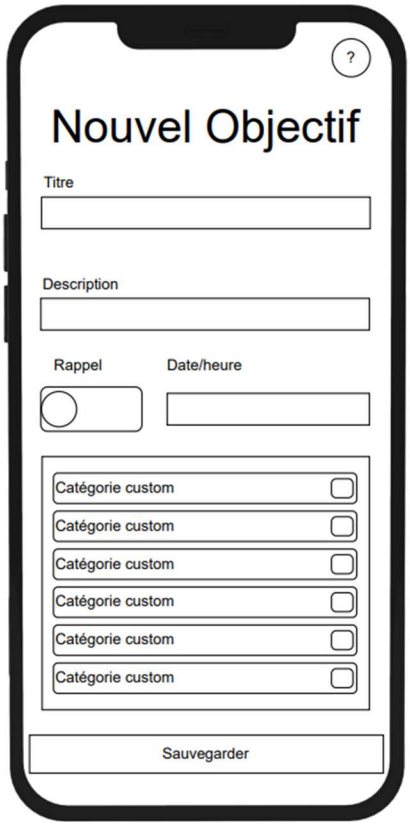
Voici les valeurs correctes :
Projet TPI : 90 heures
Documentation : 18 heures

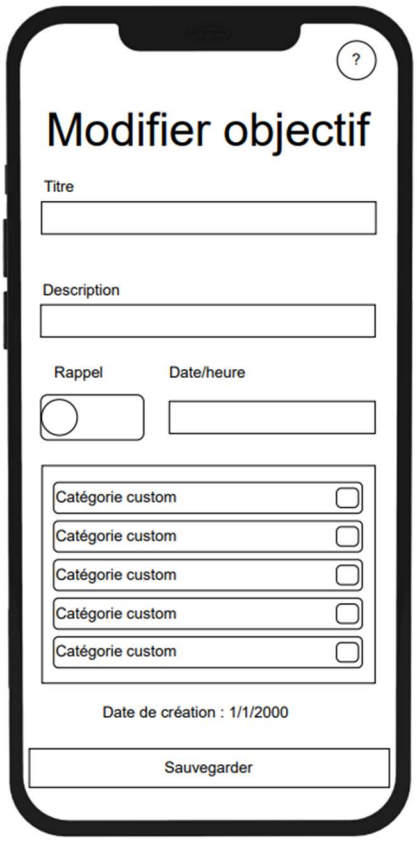
1		**Projet TPI**	**92 heures?**	**Mar 30.04.24**	**Jeu 30.05.24**
2		**Analyse**	**18 heures**	**Mar 30.04.24**	**Ven 03.05.24**
3		Introduction au projet	2 heures	Mar 30.04.24	Mar 30.04.24
4		Rédaction objectifs	1 heure	Mar 30.04.24	Mar 30.04.24
5		Diagramme de Gantt	4 heures	Mar 30.04.24	Mar 30.04.24
6		Analyse Concept	2 heures	Jeu 02.05.24	Jeu 02.05.24
7		Stratégie de test	2 heures	Jeu 02.05.24	Jeu 02.05.24
8		Risques Techniques	1 heure	Jeu 02.05.24	Jeu 02.05.24
9		Planification	1 heure	Jeu 02.05.24	Jeu 02.05.24
10		Dossier de conception	1 heure	Jeu 02.05.24	Jeu 02.05.24
11		**Implémentation**	**35 heures**	**Ven 03.05.24**	**Jeu 16.05.24**
12		**Création des layouts**	**4 heures**	Ven 03.05.24	Ven 03.05.24
16		Classe navigation	2 heures	Ven 03.05.24	Ven 03.05.24
17		**Android Manifest**	**1 heure**	Lun 06.05.24	Lun 06.05.24
18		ajout des scènes	0.5 heure	Lun 06.05.24	Lun 06.05.24
19		paramètre et permission de l'application	0.5 heure	Lun 06.05.24	Lun 06.05.24
20		**Classe RecycleurAdapter**	**2 heures**	**Lun 06.05.24**	**Lun 06.05.24**
21		Rendre les boutons cliquable	1 heure	Lun 06.05.24	Lun 06.05.24
22		Créer le layout du recycleur	1 heure	Lun 06.05.24	Lun 06.05.24
23		ajouter le recycleur dans la classe home	2 heures	Mar 07.05.24	Mar 07.05.24
24		Classe Alarm	4 heures	Mar 07.05.24	Mar 07.05.24
25		Classe AlarmReceiver	4 heures	Lun 13.05.24	Lun 13.05.24
26		Database Objectif	3 heures	Lun 13.05.24	Lun 13.05.24
27		**Classe DetailsObjectif**	**6 heures**	**Mar 14.05.24**	**Mar 14.05.24**
28		Implementation suppression d'objectif	3 heures	Mar 14.05.24	Mar 14.05.24
29		Ajout modification des valeurs par l'utilisat	3 heures	Mar 14.05.24	Mar 14.05.24
30		Implementation changement couleur de police	2 heures	Jeu 16.05.24	Jeu 16.05.24
31		Try des objectifs interactif	3 heures	Jeu 16.05.24	Jeu 16.05.24
32		**Tests**	**15 heures**	**Ven 17.05.24**	**Mar 21.05.24**
33		Test sur differents appareils	7.75 heures	**Ven 17.05.24**	**Ven 17.05.24**
34		Test selon les scénario de test préparé	4 heures	Ven 17.05.24	Mar 21.05.24
35		Correction de bug	1 heure	Jeu 16.05.24	Ven 17.05.24
36		**Documentation**	**92 heures?**	**Mar 30.04.24**	**Jeu 30.05.24**
37		Dossier de réalisation	5 heures		
38		Description des tests effectués	1 heure		
39		Résumé du rapport du TPI	2 heures		
40		Sources – Bibliographie	1 heure		
41		Journal de travail	4 heures		
42		Manuel d'Installation	0.5 heure		
43		Manuel d'Utilisation	0.5 heure		
44		Archives du projet	1 heure		
45		Conclusion	1 heure		
46		Correction orthographe	1 heure		
47		Mise en page	1 heure		

3 Analyse / Conception

3.1 Concept

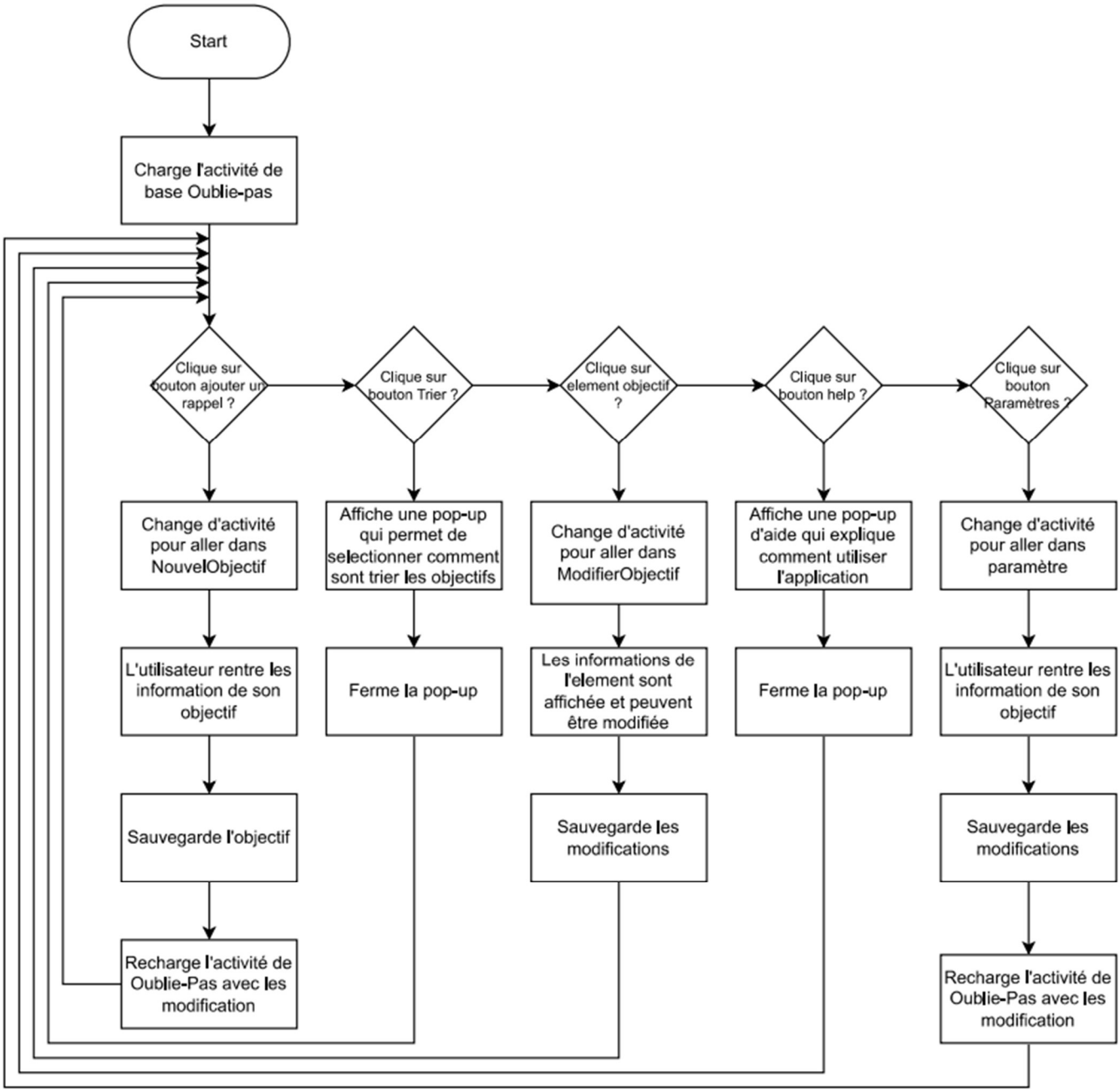






3.2 Diagrame de flux

Voici un diagramme qui montre dans les grandes lignes le déroulement d'une partie du point de vue d'un utilisateur.



3.3 Stratégie de test

La stratégie qui sera utilisée pour tester l'application.
L'application sera transmise au minimum à 5 personnes et demandé d'exécuter les actions ci-dessous puis me donner un retour

Ajouter un rappel	Va à l'écran Nouvel Objectif, l'utilisateur rentre les informations et met un rappel et le sauvegarde.	✓ ✗
Rappel sonne	Le rappel envoies une notification a l'heure demandée	✓ ✗
Cliquer sur le bouton	Passe la liste d'objectif sur un autre type de tri, date de rappel, date de création puis statut	✓ ✗
Cliquer un objectif	Passe sur Modifier objectif de l'objectif cliqué	✓ ✗
Modifier l'objectif	Modifie les informations et cliquer sur sauvegarder	✓ ✗
Supprimer un objectif	Cliquer sur un objectif, puis cliquer sur supprimer	✓ ✗
Cliquer sur bouton paramètre	Passe sur paramètre.	✓ ✗
Ajouter une catégorie	Rentre le nom de la catégorie que l'utilisateur veut créer, et clique sur le bouton ajouter la catégorie.	✓ ✗
Voir la nouvelle catégorie	La catégorie s'affiche dans la liste en dessous, s'affiche dans paramètre et dans modifier objectif	✓ ✗
Changer les paramètre	Clique sur les objectifs que l'utilisateur veut voir sur le menu principal puis cliquer sur sauvegarder	✓ ✗

3.4 *Risques techniques*

Durant la conception de la maquette du projet, j'ai été beaucoup plus ambitieux en ce qui concerne le design par rapport à mes projets précédents. J'ai cherché à créer une interface utilisateur attrayante et conviviale pour offrir une expérience agréable aux utilisateurs.

Une partie du développement de l'application me préoccupe également : l'ajout de catégories personnalisées. Cette fonctionnalité nécessite de repasser par toutes les étapes de la création de l'application, notamment la conception de trois mises en page (layouts), la création d'une base de données, ainsi que l'apprentissage de la programmation des boutons à cocher, une fonctionnalité avec laquelle je n'ai pas d'expérience préalable. L'intégration de toutes ces composantes représente un défi chronophage pour une fonctionnalité qui semble relativement simple.

3.5 Planification

J'ai créé un Projet GitHub afin de faire la planification de mon projet.
C'est pour moi la façon la plus pratique et efficiente étant donné que c'est également là où je stocke toute mes données.

Vous pouvez trouver la version PDF entière en annexe ou directement sur GitHub via ce lien : https://github.com/users/Bifmann/projects/2/views/1

3.6 Dossier de conception

3.6.1 Choix de l'IDE

L'application sera intégralement codée avec l'IDE Android Studio. C'est l'IDE officiel de Google pour le développement Android. Il prend en charge nativement Kotlin, ce qui me donne accès à de nombreuses fonctionnalités intéressantes, notamment pour le débogage.

Android Studio possède également un émulateur Android inclus, ce qui me permettra de tester "Oublie-pas" sur une multitude de systèmes d'exploitation et de formats d'écran.

3.6.2 Base de données

"Oublie-pas" utilisera non pas plusieurs fichiers JSON comme spécifié dans le cahier des charges pour stocker les données persistantes de l'application, mais Room, une bibliothèque de persistance de données pour Android qui agit comme une couche d'abstraction sur SQLite.

Cela permet une gestion plus robuste, type-safe et moins sujette aux erreurs des données grâce à la vérification au moment de la compilation des requêtes SQL et à l'intégration avec le code Kotlin.

Contrairement au stockage des données au format JSON, Room offre de meilleures performances pour les opérations de données complexes et garantit l'intégrité des données à travers des transactions et des requêtes optimisées. Cela est également beaucoup plus simple à mettre en place et sera bien plus stable et fiable.

Seront stocké comme données persistantes :

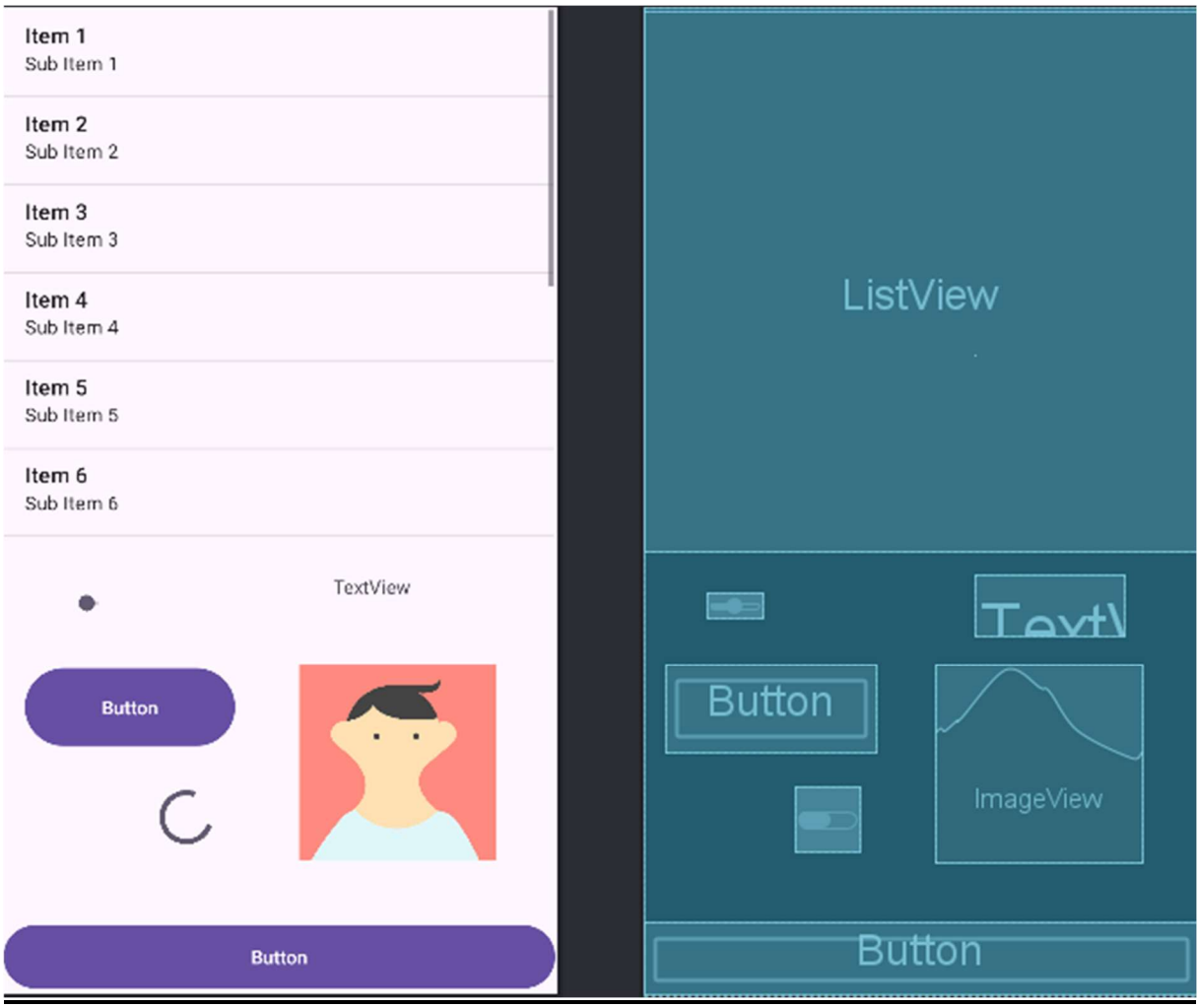
- Les objectifs
- Les paramètres de l'application

3.6.3 Maquette

Voir fichier annexe.

Les couleurs choisies et le format sera celui d'Android par défaut. En plus de me faire gagner du temps au développement, ce sont exactement les couleurs et simplicité que je recherche. Est de plus automatiquement adapté au dark mode.

Voici un Screenshot de plusieurs éléments d'Android studio pour référence.

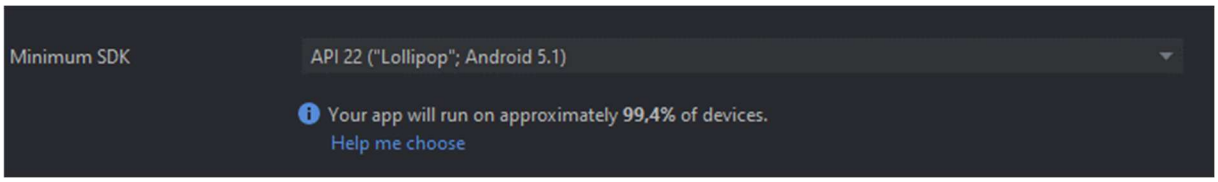


4 Réalisation

4.1 Dossier de réalisation

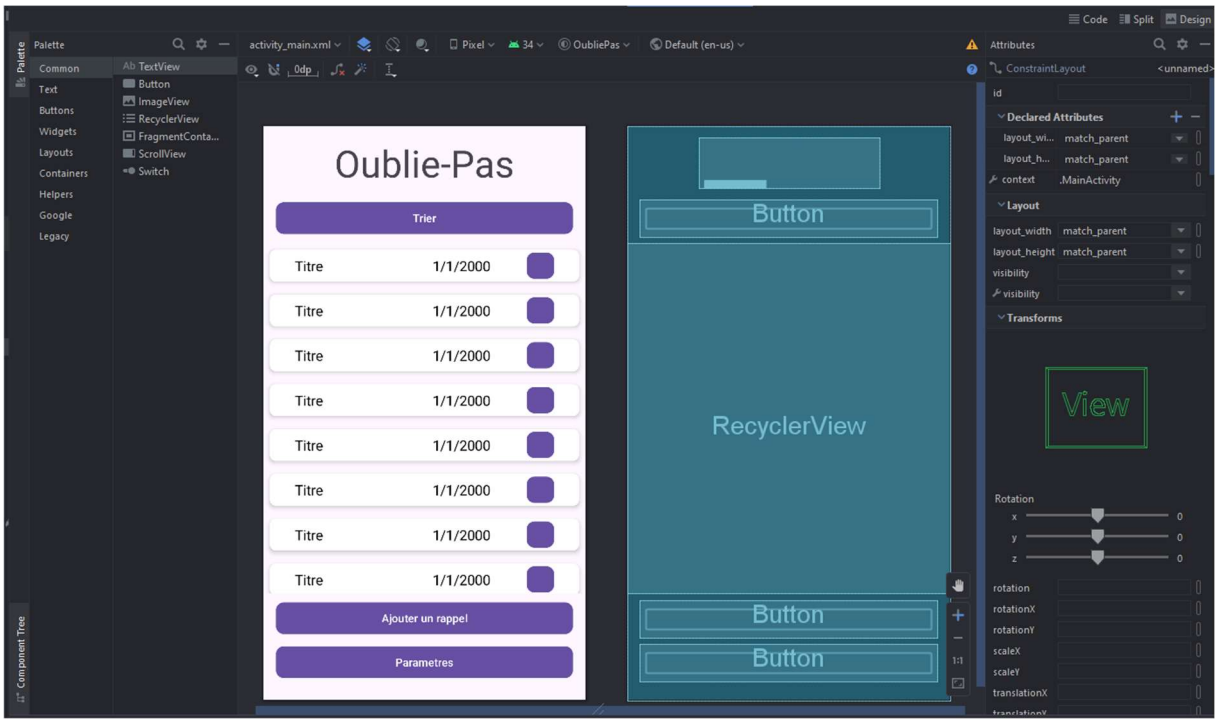
4.1.1 Version d'Android

Pour créer mon projet, j'ai choisi comme SDK minimum l'API 21, qui correspond à Android 5.0 (version Lollipop). Cela permet à l'application de fonctionner sur un maximum de téléphones (99,4%).



4.1.2 Création des layouts

J'ai décidé de commencer par m'occuper de toute la partie graphique de l'application en premier afin de ne plus avoir à y toucher à l'avenir et de fonctionner avec les mêmes bases partout dans mon projet. J'ai donc reproduit toutes mes maquettes dans Android Studio, séparées en quatre fichiers. Ces "Activités" sont codées en XML grâce au mode "Design" d'Android Studio et stockées dans le dossier "layout".



4.1.3 Préparations des classes principales

Afin de pouvoir travailler fluidement par la suite du projet, j'ai préparé dès le départ les classes de chaque activité, ainsi que la plupart des événements de clic.

Premièrement, les quatre classes "d'activité" où se trouve la logique propre à ce qui se passe dans chacune des activités. Par exemple, NouvelObjectif appellera les fonctions de la base de données et y insérera les données des nouveaux objectifs :

- MainActivity.kt
- ModifierObjectif.kt
- NouvelObjectif.kt
- Parametres.kt

Ensuite, j'ai créé les deux adaptateurs de mes recycleurs. Les adaptateurs servent à lier les données aux vues, en transformant des éléments de données en éléments d'interface utilisateur. Les recycleurs optimisent l'affichage des listes en recyclant les vues, ce qui permet de gérer efficacement de grandes collections de données. Les classes d'adaptateurs modifient les recycleurs par défaut d'Android pour afficher des listes de CardView :

- RecycleurAdapter.kt
- RecycleurAdapterCategorie.kt

Et finalement, pour que l'utilisateur puisse naviguer entre les différentes scènes, j'ai adapté et implémenté ma classe NavigationHandler, provenant d'un de mes projets précédents.

4.1.4 Ajout d'un système de notification.

Dans la réalisation du projet, un système d'alarme et de notification est nécessaire. Étant encore débutant dans ces fonctions particulières, j'ai créé un projet de test à part afin de suivre un tutoriel et de créer mes classes, réutilisables dans de multiples projets.

Le système de notification est composé de deux classes :

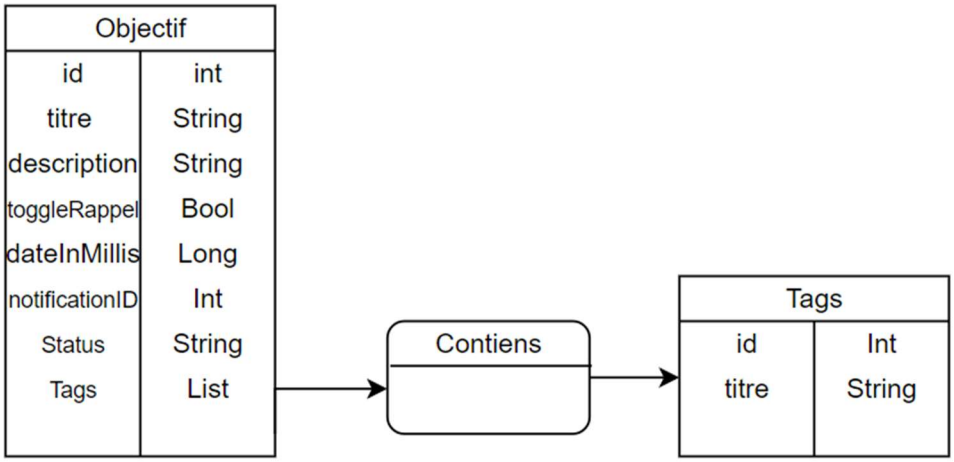
- Notification.kt
- NotificationHelper.kt

La classe Notification sert de cadre et va définir quel type de notification est créé ainsi que servir de BroadcastReceiver. La classe NotificationHelper est là pour créer un canal de notification dans l'application et planifier, déplanifier les notifications à une heure donnée.

Avec ces deux classes, utilisables dans n'importe quel projet, je peux définir des notifications avec un titre, une description et une heure souhaitée. Le design est également modulable au besoin en modifiant la classe Notification.

4.1.5 Création d'une base de données.

N'ayant pas créé de MCD lors de l'analyse de mon projet, j'ai commencé par en créer un rapidement à la demande de Monsieur Ferrari.



Je l'ai ensuite modifié pour qu'il corresponde mieux à la version qui a finalement été utilisée, plus simple.

Objectif	
id	int
titre	String
description	String
toggleRappel	Bool
dateInMillis	Long
notificationID	Int
Status	String
Tags	List of String
CreationDate	Long

Je n'ai pas lié ma table Tags à ma table Objective pour des raisons de simplicité. Elle était initialement prévue pour que tout soit bien synchronisé entre les deux. Mais je me suis rendu compte en codant ma classe que c'était bien trop complexe pour l'envergure du projet.

J'ai donc stocké ma liste de Tags globale dans une classe appelée PreferencesManager qui va utiliser des SharedPreferences pour stocker les données.

Voici donc les classes utilisées finalement ainsi qu'un court résumé de leurs utilités:

RoomData.kt : Cette classe définit l'entité de la base de données avec les champs spécifiés.

RoomDataDao.kt : Cette interface définit les méthodes d'accès aux données.

AppDatabase.kt : Cette classe crée et gère l'instance de la base de données.

PreferencesManager.kt : gère la sauvegarde et la récupération de ma liste de Tags. Elle permet d'ajouter et de supprimer des éléments de cette liste de manière persistante.

4.1.6 Ajout d’un objectif dans la base de données

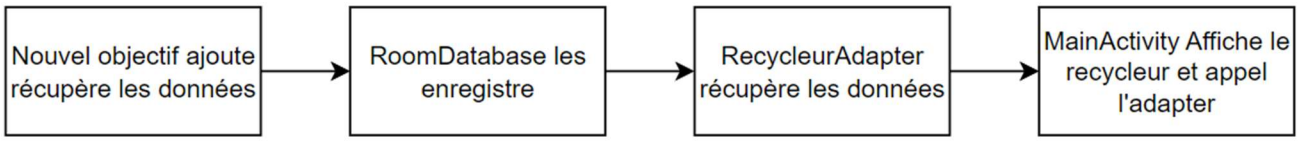
Afin d’insérer les données dans ma base de données j’ai édité ma classe NouvelObjectif qui gère la logique de l'activité.

Elle initialise les vues, configure les écouteurs d'événements et interagit avec la base de données pour ajouter de nouveaux objectifs.

Je récupère les données de l’utilisateur via les éléments EditText, puis les transmets à la base de données à l’aide de la classe RoomDao lorsque l’utilisateur appuie sur le bouton "Sauvegarder", grâce à la fonction « addObjectif ».

4.1.7 Affichage des objectifs

Pour afficher les données de mon application, j’ai édité mes RecycleurAdapter pour qu’ils récupèrent les données dans ma base de données. Ils vont ensuite les lister par éléments et leur attribuer un ID afin qu’ils puissent être cliqués.



4.1.8 Ajout de statut

Jusqu’à présent, la valeur "Statut" dans ma base de données a toujours été vide.

J’ai donc mis en place quatre statuts possibles selon mon cahier des charges, indiqués sur le menu principal par la couleur de la police.

actif – vert : l’objectif est dans le futur

urgent – Rouge : l’objectif est dans moins d’un jour

done – bleu : l’objectif est dans le passé

no notif – gris : la notification est désactivée pour l’objectif

J’ai décidé de stocker la logique dans mon Recycleur, qui est appelé à chaque changement de la base de données ainsi qu’au lancement de l’application, restant toujours à jour.

Le statut peut être modifié dans l’activité ModifierObjectif et choisi dans NouvelObjectif via un bouton. La valeur actuelle choisie est indiquée par la couleur du bouton.

4.1.9 Modification d'un objectif

Le layout étant extrêmement similaire entre les deux classes j'ai pu gagner beaucoup de temps en reprenant les fonctions de classe NouvelObjectif. Je n'ai eu qu'à implémenter updateRoom que j'avais déjà préparée lors de la création de la base de données.

4.1.10 Tri de l'affichage des objectifs

Pour le trier mes objectifs dans mon activité principale, j'ai dû implémenter plusieurs fonctions :

- showSortDialog.kt
- sortByReminderDate.kt
- sortByCreationDate.kt
- sortByStatus.kt
- getStatusPriority.kt

Ces options sont choisies par l'utilisateur via le bouton trier sur le menu principal qui vas faire apparaitre un menu gérer par la fonction showSortDialog.

Cette fonction est intéressante car elle utilise L'AlertDialog Builder d'Android. Cela me permet d'afficher des petits menu et texte complètement customisable.
On retrouve l'utilisation de cette fonctionnalité dans la Classe NouvelObjectif pour confirmer à l'utilisateur la création d'un objectif.

4.1.11 Annulation d'une notification

Pour annuler une notification j'ai dû créer un évènement pour le bouton de mon recycleur.
Un problème que j'ai rencontré est qu'il fallait que j'exclue l'évènement du bouton de celui du click de l'élément entier. Android ne fessant pas la différence par défaut.

J'appelle donc la fonction unscheduleNotification de la classe Notification helper si le bouton pressé. Les notifications utilisent le même ID entre la notification et la base de données, les retrouver n'est pas un problème.

4.2 Description des tests effectués

Afin de tester mon projet dans les conditions les plus réels possibles, j'ai transmis mon application à 5 volontaires. Ces volontaires ont ensuite rempli la grille prévue lors de mon analyse, modifiée avec certains ajouts afin de mieux refléter le projet final et quelques précisions. Voici les résultats reçus.

L'application a pu fonctionner chez tous les participants malgré certains crash découvert et documentés dans les erreurs restantes.

Ajouter un rappel	Va à l'écran Nouvel Objectif, l'utilisateur rentre les informations, met un rappel avec une alarme mise a quelque minute dans le futur et le sauvegarde.	✓
Affichage sur le menu principal	L'objectif s'affiche sur le menu principal, la couleur de police de la date est rouge.	✓
Rappel sonne	Le rappel envoie une notification à l'heure demandée	✓
Cliquer sur le bouton trier	Passe la liste d'objectif sur un autre type de tri, date de rappel, date de création puis statut	✓
Cliquer un objectif	Passe sur Modifier objectif de l'objectif cliqué.	✓
Modifier l'objectif	Modifier les informations et cliquez sur sauvegarder, l'objectif est maintenant modifié	✓
Supprimer un objectif	Cliquer sur un objectif créer, puis cliquer sur supprimer, l'objectif est supprimé	✓
Cliquer sur bouton paramètre	Passe sur paramètre.	✓
Ajouter une catégorie	Rentre le nom de la catégorie que l'utilisateur veut créer, et clique sur le bouton ajouter la catégorie.	✗
Voir la nouvelle catégorie	La catégorie s'affiche dans la liste en dessous, s'affiche dans paramètre et dans modifier objectif	✗
Changer les paramètre	Clique sur les objectifs que l'utilisateur veut voir sur le menu principal puis cliquer sur sauvegarder	✗

4.3 Erreurs restantes

4.3.1 Notifications non recréées après modification

Si la notification d'un objectif est appliquée et qu'il est modifié pour repasser dans le futur, une seconde notification n'est pas créée.
C'est un patch relativement facile à appliquer, il suffit d'empêcher la modification d'un objectif s'il est dans le passé.

4.3.2 Permissions manquantes

Si les permissions ne sont pas activées, l'application plante au moment de la création d'un objectif.
Il faut simplement empêcher l'utilisateur de créer une notification s'il n'a pas donné les permissions à l'application.

4.3.3 Objectifs vides

L'application plante si l'on tente de créer ou de modifier un objectif vide.
Je peux corriger cela en ajoutant une vérification que tous les champs de données soient remplis.

4.3.4 Syntaxe de la date incorrecte

L'application plante si un objectif est créé ou modifié avec une mauvaise syntaxe pour la date.
Une solution possible est de créer un petit onglet qui s'ouvre avec une horloge interactive afin que la donnée reçue soit toujours du même type.

4.3.5 Entrées vides en cas de crash

Si l'application plante pour une raison ou une autre au moment de la création d'un objectif, cela crée une entrée vide dans la base de données, ce qui provoque un décalage dans le menu principal ainsi qu'une instabilité dans le code, non prévue pour lister des données vides.
Il faut simplement empêcher l'utilisateur de faire planter l'application en la rendant assez sûre à utiliser. Le terme technique pour cela est "robustesse" ou "résilience".

4.3.6 Problèmes de couleur des boutons de toggle

Les couleurs du bouton de toggle alarme dans les activités ModifierObjectif et NouvelObjectif posent quelques soucis : une fois cliqué, le bouton ne reprend jamais sa couleur d'origine.
Il faut que je rende toutes les couleurs de l'application uniformes en les notant dans le fichier res/values/colors.xml.

4.3.7 Adaptation au mode paysage

L'application n'a pas été adaptée au mode paysage du téléphone.
Cela a pour conséquence que si l'utilisateur tourne son téléphone et qu'il a activé le mode paysage, l'application ne pourra pas s'afficher correctement.
J'envisage de créer une version de mes activités spécialement pour le mode paysage. Il est également possible de simplement désactiver cette option afin d'éviter toute confusion chez l'utilisateur.

4.3.8 Notifications non désactivées correctement

L'application ne désactive pas toujours les notifications correctement.
Je pense que j'ai fait une erreur dans ma fonction unscheduleNotification. C'est probablement la mauvaise ID qui est annulée. Android ne fournissant aucun log spécifique sur les notifications créées ou supprimées, je n'ai pas moyen de savoir quelle notification est supprimée. Il faudrait que je crée une base de données supplémentaire pour garder le fil de tout cela.

4.3.9 Problème de cache

L'application ne vide pas le cache et à chaque changement dans la navigation, un nouvel onglet est créé.
Il faut que je mette en place un système pour supprimer les onglets au bout du troisième retour afin de toujours pouvoir utiliser le bouton retour par défaut d'Android.

4.3.10 Problème système de tri

Si les objectifs sont triés, les ID ne sont plus au bon endroit, ce qui peut entraîner des clics sur les mauvais éléments. Une solution consiste à garder une trace des déplacements des boutons dans la base de données pour s'assurer que chaque élément reste associé à son ID correct. Cela permet de maintenir la cohérence entre l'affichage et les données stockées.

4.4 Liste des documents fournis

Tous les documents et code de mon projet sont présents dans mon Github :
https://github.com/Bifmann/Oublie-pas

L'exécutable se trouve dans le dossier « executable », a la racine du projet.

Les documents de documentation ont tous été tenu a jour en PDF, en voici la liste :

- Diagramme de flux
- Diagramme_de_Gantt_planification_initiale
- Dossier de projet
- Journal de travail
- Maquette
- MCD
- Planification

5 Conclusions

5.1.1 Objectifs atteints / non-atteints

Je vais commencer par dresser la liste de tous les objectifs atteints ou non.

Atteint :

- L'application doit être codée en Kotlin.
- L'utilisateur devra pouvoir créer objectif et rappel, composé d'un titre et une description.
- L'utilisateur doit pouvoir choisir s'il veut un rappel ou non.
- Les objectifs auront différents statuts en fonction de si le rappel est dépassé ou non.
- Les objectifs doivent être affichés sous forme de liste avec uniquement le titre et la date.
- Les objectifs terminés doivent être séparés par un menu.
- Le tri des objectifs doit pouvoir se faire selon :
    - La date du rappel
    - La date de création
    - Leur statut
- L'utilisateur doit pouvoir afficher le détail d'un objectif en cliquant dessus.
- Toutes les informations d'un objectif doivent pouvoir être modifiable par l'utilisateur
- L'utilisateur doit pouvoir supprimer un objectif
- La couleur et la police des objectifs doivent être affichées différemment selon la date de rappel.
- Les objectifs doivent être sauvegardés dans une base de données.

Non-atteint

- Les objectifs doivent être catégorisés.
- Les catégories doivent être personnalisables.

5.1.2 Points positifs / négatifs

Points positifs :

- J’ai pris personnellement énormément de plaisir à faire ce projet c’est vraiment quelque chose que j’ai apprécié faire, ce qui est très important pour moi.
- J’ai pu réaliser la majeure partie des objectifs que je m'étais fixé
- L’application est fonctionnelle si on sait l’utiliser.
- J’ai appris énormément de choses sur la gestion de mon temps et la programmation en Kotlin en réalisant ce projet.

Points négatifs

- En progressant dans mon projet je me suis rendu compte petit à petit que mon analyse ne couvrait pas bien l'intégralité du projet.
- J’ai pris trop de temps sur certains fonctionnalité basiques, en particulier les notifications
- J’ai eu une implémentation beaucoup trop longue par rapport à ce que j’avais prévu. Me laissant peu de temps pour peaufiner ma documentation et code.

5.1.3 Difficultés particulières

Comme mentionné dans les points négatifs j’ai eu divers soucis durant l’implémentation. J’ai pensé lors de l’analyse que l’ajout des fonctionnalités de bases tel que créer une notification serait simple et le plus serait d’assembler toutes ces fonctions et classes créée ensemble

5.1.4 Suites possibles pour le projet (évolutions et améliorations)

- La correction des erreurs connue
- Ajout de toutes les fonctionnalités de catégorisation, il suffit que je termine l’ajout de nouveau tags dans l’application et que j’effectue un tri sur le menu principal en fonctions des tags sélectionné
- Amélioration de l’interface graphique, peu intuitive et ne donnant pas le maximum d’information possible à l'utilisateur.
- Ajout de récompense lors de réussite d’objectif pour la journée, afin d’améliorer le principe d’aide au personne atteinte de trouble de l’attention un système point gagner et peut-être de vignette gagnée améliorerait grandement cet aspect, comme le prouve certaines études.

6 Annexes

6.1 Résumé du rapport du TPI / version succincte de la documentation

"Oublie-pas" est une application mobile réalisée dans le cadre de mon TPI, conçue pour aider les utilisateurs, notamment ceux atteints de TDAH, à organiser et suivre leurs objectifs quotidiens. L'application permet de créer des listes d'objectifs personnalisables avec des rappels à des heures spécifiques, favorisant ainsi la régularité et la discipline dans l'accomplissement des tâches. Les objectifs peuvent être catégorisés, triés par date de rappel, date de création ou statut, et affichés dans une liste où leur état est visuellement représenté.

L'application a été développée en Kotlin et utilise Room pour la gestion de la persistance des données, garantissant ainsi une manipulation sécurisée et efficace des informations. Un système de notifications a été intégré, composé des classes Notification et NotificationHelper, pour créer, planifier et annuler des rappels. Ces fonctionnalités permettent à l'utilisateur de recevoir des notifications à des moments précis pour ne pas oublier leurs objectifs. Une attention particulière a été portée à l'interface utilisateur, avec des layouts créés en XML et une gestion fluide de la navigation entre les différentes activités grâce à la classe NavigationHandler.

Des tests approfondis ont été réalisés pour assurer le bon fonctionnement de l'application. Cependant, plusieurs problèmes ont été identifiés, comme les notifications non désactivées correctement, les permissions manquantes, et les crashes dus à des entrées invalides ou vides. Ces problèmes nécessitent des améliorations futures pour garantir une expérience utilisateur optimale. En conclusion, bien que le projet ait atteint la plupart de ses objectifs, certaines fonctionnalités restent à peaufiner pour offrir une application robuste et pleinement fonctionnelle.

6.2 Sources – Bibliographie

J'ai repris des éléments de mon projet de pre-tpi, JoQuiz

Kotlin Course - Tutorial for Beginners
De : freeCodeCamp.org
https://youtu.be/F9UC9DY-vIU?si=9Ud6biwNBrvoh6N4

Schedule Local Notifications Android Studio Kotlin Tutorial
De : Code With Cal
https://youtu.be/_Z2S63O-1HE?si=APXQGXCUdfTx7Hug

https://www.practicalcoding.net/blog/categories/kotlin

https://chat.openai.com/

Les 5 personnes qui m'ont aidé à tester mon application, certain ont choisi de garder l'anonymat :

- Joshua Surico
- Edward Stewart
- Levy Stoller
- Marc Arthur (nom d'emprunt)
- George Duvoisin (nom d'emprunt)

6.3 Journal de travail

Voir annexe « Journal de travail »

6.4 Manuel d'Installation

6.4.1 Prérequis

Un smartphone ou une tablette Android.

L'application "Oublie-pas" au format APK. (Livré en annexe du projet)

6.4.2 Étapes d'installation

Étape 1 : Autoriser l'installation d'applications de sources inconnues
Avant d'installer une application qui n'est pas téléchargée depuis le Google Play Store, vous devez autoriser l'installation d'applications de sources inconnues.

Ouvrez les paramètres de votre appareil.
Naviguez jusqu'à "Sécurité" ou "Confidentialité".
Recherchez l'option "Sources inconnues" et activez-la.
Confirmez en cliquant sur "OK" ou "Autoriser" lorsque vous voyez un avertissement concernant les risques potentiels.

Étape 2 : Télécharger l'APK
Téléchargez le fichier APK de l'application "Oublie-pas" sur votre appareil Android. Vous pouvez le faire directement depuis votre appareil en cliquant sur un lien de téléchargement, ou vous pouvez transférer le fichier depuis votre ordinateur via un câble USB.

Étape 3 : Installer l'APK
Ouvrez le gestionnaire de fichiers sur votre appareil Android.
Naviguez jusqu'à l'emplacement où le fichier APK a été téléchargé ou transféré.
Cliquez sur le fichier APK "Oublie-pas.apk" pour lancer l'installation.
Suivez les instructions à l'écran et cliquez sur "Installer".

Étape 4 : Autoriser les permissions de notifications
Après l'installation, ouvrez l'application "Oublie-pas".

Lors du premier lancement, l'application vous demandera des permissions.

Accordez les permissions nécessaires, notamment l'accès aux notifications, pour que l'application puisse envoyer des rappels.

Une boîte de dialogue apparaîtra pour demander l'autorisation d'envoyer des notifications. Cliquez sur "Autoriser".

Étape 5 : Vérification de l'installation
Assurez-vous que l'application "Oublie-pas" est bien installée en la recherchant dans la liste de vos applications.
Ouvrez l'application et configurez vos premiers objectifs et rappels pour vérifier que tout fonctionne correctement.

Étape 6 : Désactiver les sources inconnues (optionnel)
Pour des raisons de sécurité, il est recommandé de désactiver l'option d'installation à partir de sources inconnues après avoir installé l'application.

Retournez dans les paramètres de votre appareil.
Naviguez jusqu'à "Sécurité" ou "Confidentialité".
Désactivez l'option "Sources inconnues".

Vous avez maintenant installé avec succès l'application "Oublie-pas" sur votre appareil Android.

6.5 Glossaire

API (Application Programming Interface) Interface de programmation qui permet à différentes applications de communiquer entre elles. Dans ce projet, l'API 21 d'Android est utilisée comme version minimale du SDK.

Base de données Système permettant de stocker, organiser et gérer des informations de manière structurée. Dans ce projet, la base de données est gérée par la bibliothèque Room.

BroadcastReceiver Composant Android qui permet de recevoir et de répondre à des messages diffusés par d'autres applications ou par le système Android lui-même.

CardView Composant d'interface utilisateur Android qui permet d'afficher des informations dans une carte avec des ombres et des coins arrondis, souvent utilisé dans des listes.

EditText Widget d'interface utilisateur Android qui permet à l'utilisateur de saisir et de modifier du texte.

IDE (Integrated Development Environment) Environnement de développement intégré. Dans ce projet, Android Studio est l'IDE utilisé pour développer l'application.

Kotlin Langage de programmation utilisé pour développer l'application "Oublie-pas". Kotlin est officiellement supporté par Google pour le développement Android.

Layout Disposition des éléments d'interface utilisateur dans une activité Android. Dans ce projet, les layouts sont définis en XML.

LifecycleScope Composant de gestion du cycle de vie dans les applications Android qui permet de lancer des coroutines liées au cycle de vie d'une activité ou d'un fragment.

Recycleur (RecyclerView) Composant d'interface utilisateur Android qui permet d'afficher de grandes quantités de données de manière efficace en recyclant les vues.

Room Bibliothèque de persistance de données pour Android qui simplifie l'utilisation de SQLite en fournissant une couche d'abstraction et de vérification des requêtes SQL au moment de la compilation.

SharedPreferences Mécanisme de stockage persistant clé-valeur pour les données primitives dans les applications Android. Utilisé pour stocker des paramètres simples.

XML (Extensible Markup Language) Langage de balisage utilisé pour définir les layouts et autres configurations dans les applications Android.

6.6 Manuel d'Utilisation

6.6.1 Important

Si l'application plante veuillez la réinitialisé ou la désinstallé puis réinstallé afin d'éviter des instabilités dans fonctionnement.

6.6.2 Menu principal

Depuis là vous pouvez voir tous les objectifs que vous avez créé.

Voici toutes les actions possibles sur cette page :

Appuyez sur le bouton trier pour changer l'ordre d'affichage des objectifs.

Cliquez sur un objectif pour le modifier.

Cliquez sur le petit bouton violet de d'un objectif pour désactiver sa notification

Cliquer sur le bouton Ajouter un rappel vous mène sur le menu pour créer un objectif

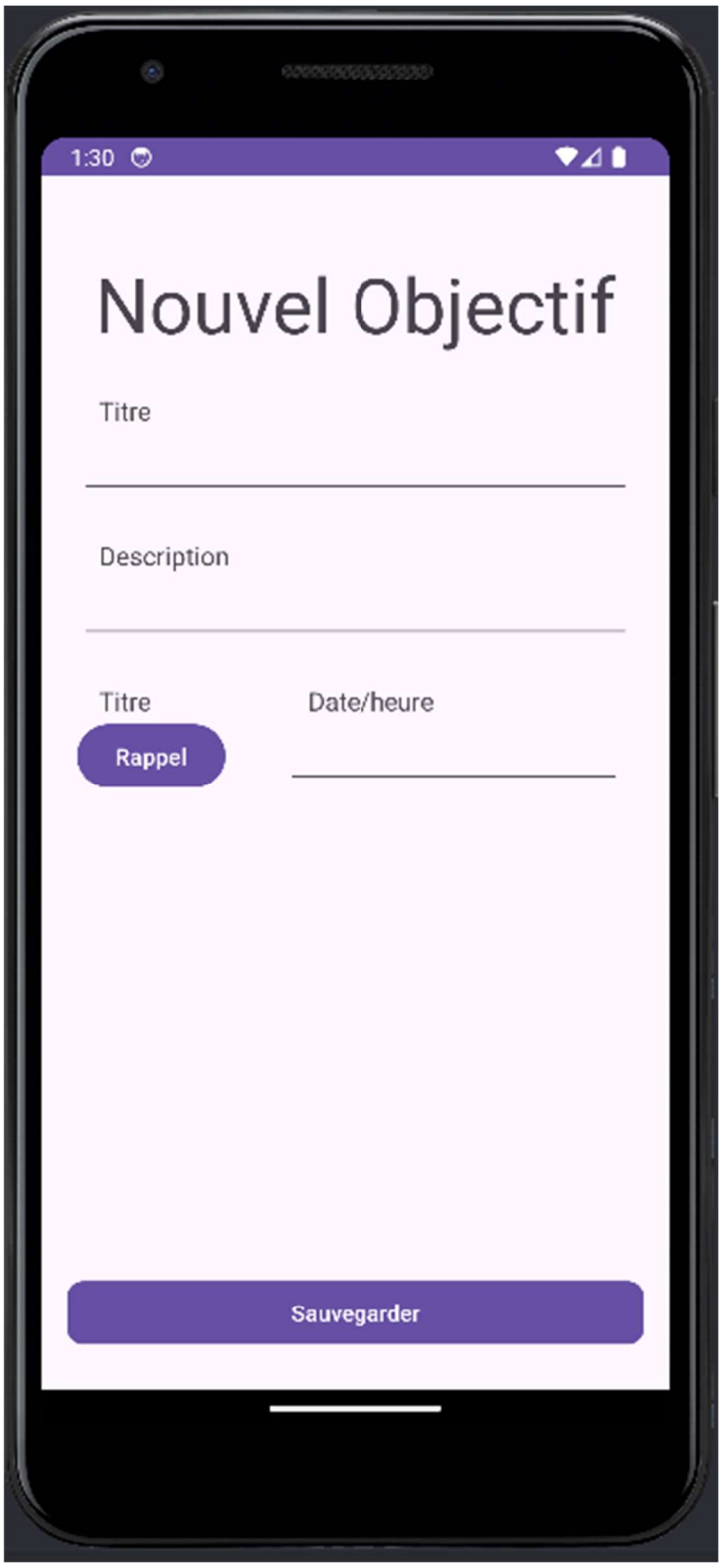
Il y a 4 codes couleurs afin de repérer rapidement quand sont dut les objectifs :

actif – vert : l'objectif est dans le futur

urgent – Rouge : l'objectif est dans moins d'un jour

done – bleu : l'objectif est dans le passé

no notif – gris : la notification est désactivée pour l'objectif



6.6.3 Nouvel Objectif.

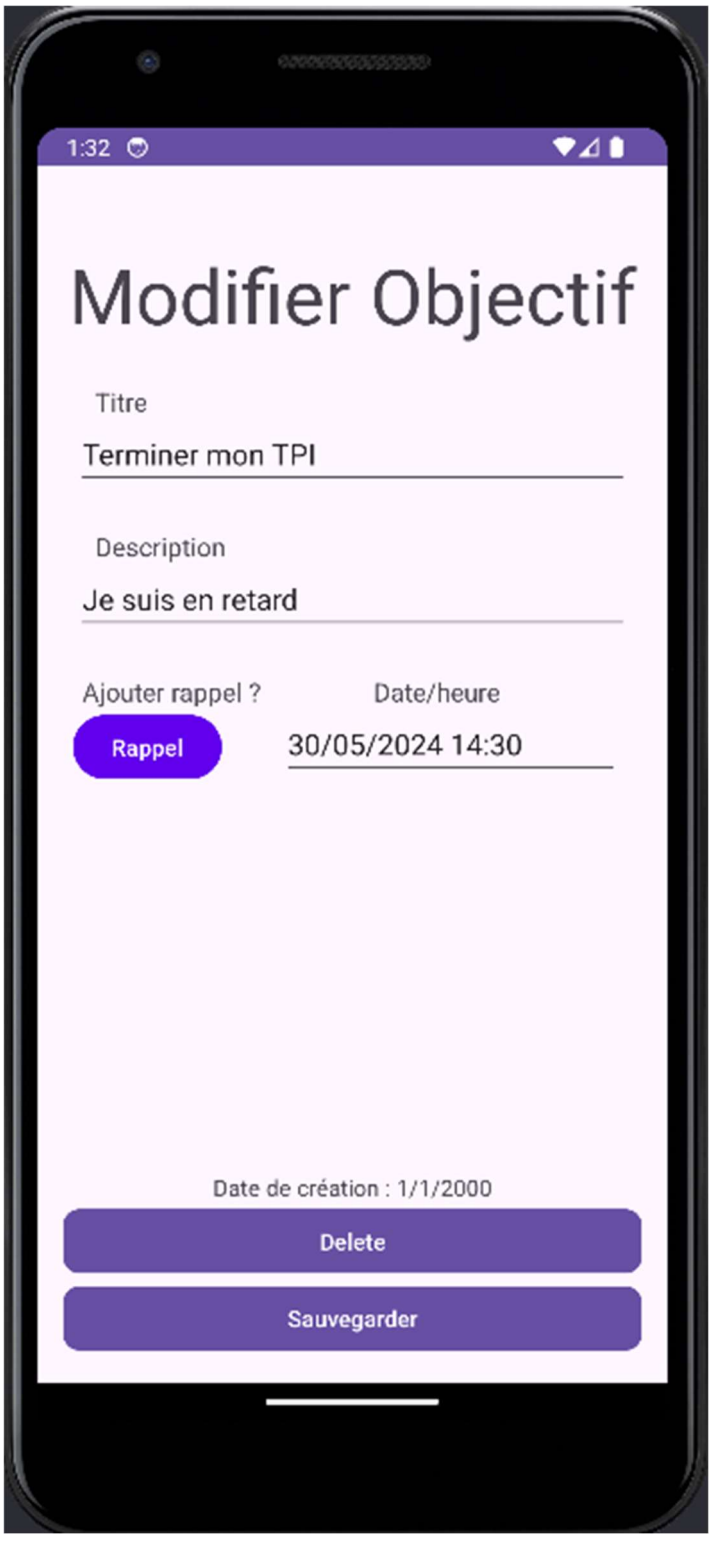
Il faut que tous les champs soient remplis pour créer une notification.

Le bouton Rappel change de couleur si vous lui cliquez dessus, bleu = une notification, rouge = pas de notification.

Le champs date/heure doit être remplis avec exactement le format « dd/MM/yyyy HH:mm » Exemple :
30/05/2024 14:30

Cliquez sur sauvegarder pour ajouter l'objectif

Appuyez sur la touche retour de votre téléphone pour annuler l'ajout d'un objectif



6.6.4 Modifier Objectif

Similaire au menu nouvel Objectif à la différence que les données sont préremplie.

Vous pouvez supprimer l'objectif en appuyant sur Delete ou sauvegarder les modifications en pressant Sauvegarder.

Les mêmes règles que nouvel Objectif s'applique à propos des champs et du formats de date/heure.