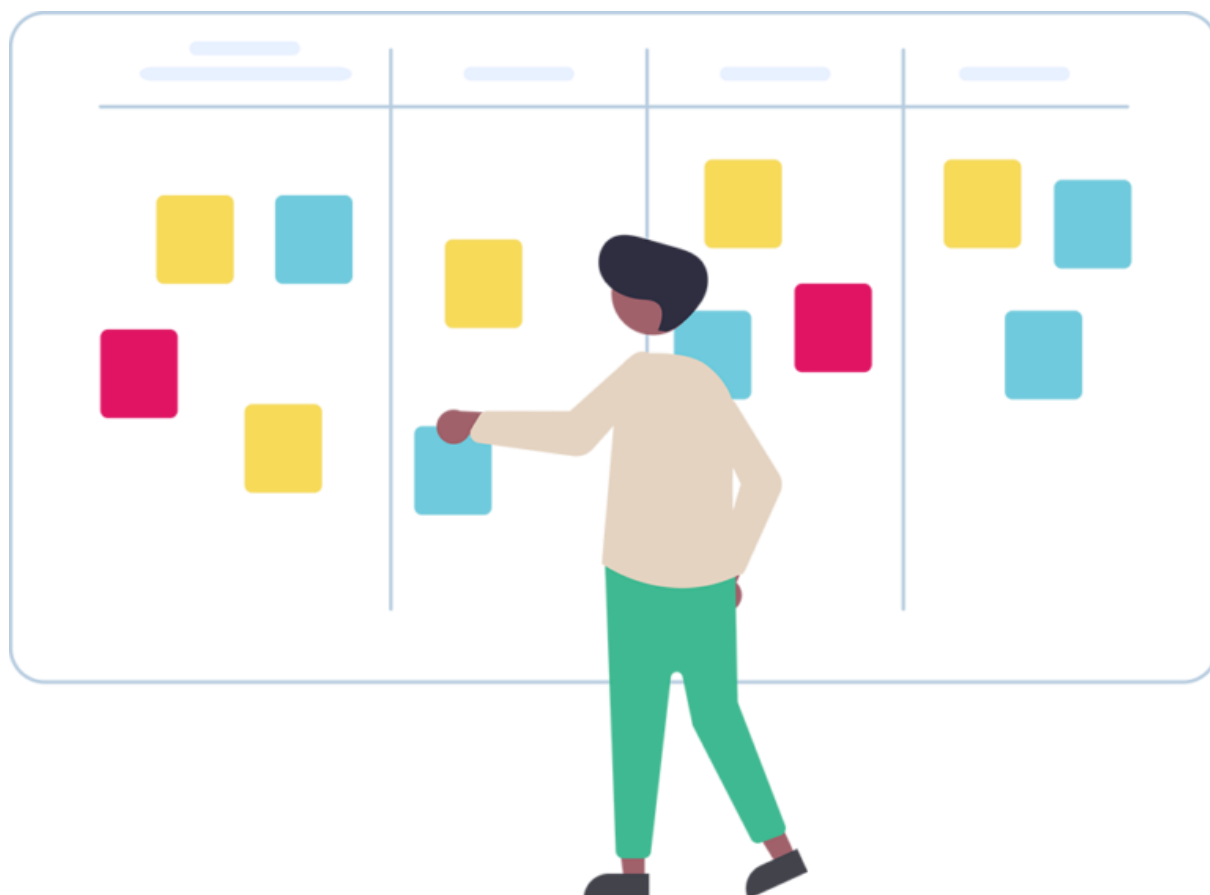


# Projet TPI – Gestionnaire de tâches

---



Tristan Gerber – MID4  
ETML N501  
Du 2 Mai 2022 au 1er Juin 2022  
Chef de projet : Roberto Ferrari  
Experts : Joseph Brandenburg, Serge Wenger

## Table des matières

<b>1</b>	<b>SPÉCIFICATIONS.....</b>	<b>3</b>
1.1	TITRE .....	3
1.2	DESCRIPTION .....	3
1.3	MATÉRIEL ET LOGICIELS À DISPOSITION .....	3
1.4	PRÉREQUIS.....	3
1.5	CAHIER DES CHARGES.....	3
1.5.1	Objectifs et portée du projet .....	3
1.5.2	Caractéristiques des utilisateurs et impacts .....	4
1.5.3	Contraintes.....	4
1.5.4	Si le temps le permet ... ..	4
1.6	MÉTHODES DE VALIDATION DES SOLUTIONS.....	4
1.7	LES POINTS SUIVANTS SERONT ÉVALUÉS .....	4
1.8	VALIDATION ET CONDITIONS DE RÉUSSITE .....	4
<b>2</b>	<b>ANALYSE.....</b>	<b>4</b>
2.1	OPPORTUNITÉS.....	4
2.2	MÉTHODOLOGIE DE PROJET .....	5
2.2.1	S'informer.....	5
2.2.2	Planifier .....	5
2.2.3	Décider.....	5
2.2.4	Réaliser.....	5
2.2.5	Contrôler.....	5
2.2.6	Évaluer .....	5
2.3	NORMES DE CODAGE .....	5
2.4	DÉPLOIEMENT DE L'APPLICATION .....	6
2.5	LANGAGES UTILISÉS .....	6
2.6	STOCKAGE DES DONNÉES.....	6
2.7	SAUVEGARDE DU PROJET .....	7
2.8	TESTS .....	7
<b>3</b>	<b>CONCEPTION .....</b>	<b>7</b>
3.1	STRUCTURE DES PAGES & NAVIGATION .....	8
3.2	CONCEPTION DES TESTS.....	12
3.3	PLANIFICATION DÉTAILLÉE .....	13
<b>4</b>	<b>RÉALISATION .....</b>	<b>14</b>
4.1	DOSSIER DE RÉALISATION .....	14
4.1.1	Programmes & versions utilisés.....	14
4.1.2	Base de données.....	15
4.1.3	MVVM & Code .....	15
4.2	COMPILATION .....	16
4.2.1	Réalisation des tests .....	16
<b>5</b>	<b>CONCLUSION.....</b>	<b>17</b>
5.1	BILAN DES FONCTIONNALITÉS DEMANDÉES .....	17
5.2	BILAN DE LA PLANIFICATION.....	17
5.3	BILAN PERSONNEL .....	17
<b>6</b>	<b>WEBOGRAPHIE .....</b>	<b>18</b>
<b>7</b>	<b>TABLE DES ILLUSTRATIONS.....</b>	<b>19</b>
<b>8</b>	<b>ANNEXES .....</b>	<b>20</b>
8.1	PLANIFICATION .....	20

# 1 SPÉCIFICATIONS

En dernière année, un projet qui sert d'examen final, un TPI (Travail Pratique Individuel) est réalisé par les élèves. Ce rapport concerne mon TPI dont le chef de projet est M. Ferrari. Cette partie du rapport correspond à la première étape de la méthode des six pas.

## 1.1 Titre

Gestionnaire de tâches

## 1.2 Description

Ce projet consiste en la création d'une application de gestion des tâches.

## 1.3 Matériel et logiciels à disposition

**PC standard de l'ETML**

**Visual Studio 2019** – Utilisé pour coder – Préinstallé

**Windows 10** – Système d'exploitation de l'ordinateur

**Suite Office** – Utilisé pour réaliser la documentation – Préinstallé

**Pencil Project** – Utilisé pour réaliser les maquettes – Installé avec l'exécutable : K:\INF\Maitres-Eleves\Outils\\_APP-Portables\03-developpement\PencilProjectPortable\_3.0.3\_English.paf.exe

**Téléphone de modèle Google Pixel 2** – Utilisé pour héberger, lancer et tester l'application – Fourni par M. Lymberis.

**GitHub & Git Bash** – Utilisé dans le cadre de la sauvegarde et du versionning du projet – Installé depuis <https://git-scm.com/downloads>. Afin d'installer Git Bash, il suffit de télécharger le Setup ou la version Portable en appuyant sur « next » à toutes les étapes.

DBMain

## 1.4 Prérequis

Compétences en bases de données

→ Modules 104 et 153

Compétences en programmation

→ Modules 120, 226A et B, 326, 403, 404, 411, 335  
P\_APPRO (Xamarin)

Compétences en outils bureautiques

→ Modules 104 et 153

Compétences en gestion de projets

→ Modules 104 et 153

## 1.5 Cahier des charges

Le cahier des charges est disponible :

- Sous le fichier [DLS-Cdc-GestionStockEtml.docx](#)
- En annexe à ce rapport.

### 1.5.1 Objectifs et portée du projet

1. Catégorisation des tâches
2. Affichage des tâches par catégorie
3. Affichage des tâches par ordre de date d'échéance
4. Insertion, modification et suppression de tâches
5. Possibilité de marquer une tâche comme terminée, la rendant invisible
6. Infographie par date d'échéance

7. Création de « ToDoList » à partir des tâches

### 1.5.2 Caractéristiques des utilisateurs et impacts

Le public cible est constitué de personnes de toutes les tranches d'âge, sexe ou nationalité parlant français et souhaitant gérer leurs tâches quotidiennes et prendre en main leur temps. L'interface doit être simple d'utilisation, l'application utilisable en tout temps avec ou sans connexion à internet.

### 1.5.3 Contraintes

Temps pour le projet : 1 mois

Émulateur ou téléphone pour compiler

### 1.5.4 Si le temps le permet ...

## 1.6 Méthodes de validation des solutions

Tests manuels sur la sécurité de l'application et du bon fonctionnement des différentes fonctionnalités.

Si le temps le permet, Test de l'application sur différents appareils de différentes tailles.

L'application devra être testée sur iOS. Il donc faudra utiliser un émulateur iOS ou directement connecter un téléphone de ce même OS au PC. L'application sera également à tester sur des écrans de tailles différentes et des nouvelles versions d'Android et d'iOS.

## 1.7 Les points suivants seront évalués

- Le rapport
- Les planifications (initiale et détaillée)
- Le journal de travail
- Le code et les commentaires
- Les documentations de mise en œuvre et d'utilisation

## 1.8 Validation et conditions de réussite

- Compréhension du travail
- Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
- Etat de fonctionnement du produit livré
- Produit livré dans les temps

## 2 ANALYSE

Lors de l'analyse, on regarde le projet à faire sous tous les angles afin de se faire une bonne idée du travail à fournir et de la direction à prendre. Cette partie du rapport correspond à la deuxième et troisième étape de la méthode des six pas.

### 2.1 Opportunités

Ce projet a pour but de décider quant à ma qualification à recevoir mon CFC. Il servira de consolidation pour les bases en Xamarin déjà acquises précédemment durant le P\_APPRO et

viendra vérifier mes compétences en gestion de projet, en programmation, en bureautique et en gestion de projets.

## 2.2 Méthodologie de projet

Afin d'avoir un projet structuré, la méthode des 6 pas est utilisée lors de tout le projet. Ce choix est dû à sa bonne structuration. Celle-ci permet une meilleure planification à l'avance au détriment de la flexibilité lors de la création du projet. Étant donné que les objectifs de ce projet sont clairs, que le domaine est maîtrisé et que le cahier des charges ne changera pas, il est plus adapté de tout planifier à l'avance.

Elle consiste en six étapes :

### 2.2.1 S'informer

Des recherches sont réalisées autour du mandat afin d'analyser les besoins, les possibilités, les limitations et les mesures à prendre.

### 2.2.2 Planifier

Définir le temps mis à disposition, les priorités, les solutions et les moyens à disposition afin d'avoir des tâches claires et séparées durant l'entièreté du projet.

### 2.2.3 Décider

À l'aide des informations trouvées auparavant, décider comment et à l'aide de quel outil chaque tâche va être réalisée et les avantages et inconvénients de la solution par rapport aux autres.

### 2.2.4 Réaliser

Réaliser les différentes tâches, s'ajuster à la planification et la décision sur les tâches, constater ce qui peut être véritablement fait dans les conditions de travail actuelles.

### 2.2.5 Contrôler

Comprendre ce qui a pu mal se passer, ce qui s'est bien passé, le respect de la planification, les modifications à envisager, les conséquences des erreurs.

### 2.2.6 Évaluer

Mesurer l'état d'avancement des travaux, faire une rétrospective sur les tâches qui ont posé des problèmes, trouver des améliorations à faire sur l'approche de projet, tirer une leçon du déroulement du projet.

## 2.3 Normes de codage

Dans tout le programme en C#, les variables suivent le lowerCamelCase tandis que les constantes, les fonctions et les classes suivent le UpperCamelCase. Les commentaires en C# sont tous sous cette forme :

```
// Creates a row in the database with the borrow's informations
```

Chaque méthode et classe de l'application comporte un sommaire et chaque page de code comporte un entête sous cette forme :

```
/* Developper : Tristan Gerber
 * Place : ETML, N501
 * Project creation date : 02.05.2022
 * Last updated : 02.05.2022 */
```

La langue utilisée dans les commentaires est la même que celle du code, l'anglais. C'est une langue dans laquelle je suis à l'aise et d'après moi, il est logique d'écrire le nom des objets et des commentaires dans la même langue que s'écrit le code.

La langue utilisée pour la partie visible de l'application est évidemment le Français, étant donné que les utilisateurs de l'application parleront sans doute cette langue.

## 2.4 Déploiement de l'application

Le programme tournant sur un téléphone, le choix s'impose d'avoir :

- Un émulateur (Pour Android, iOS et Windows Phone)
- Un téléphone (Android)

M. Lymberis, s'occupant de la salle N501, me mettant à disposition un téléphone, je préfère le second choix. La création d'émulateurs aurait pris trop de temps sur le projet, l'émulateur iOS nécessite une installation spéciale (via iTunes) et l'exécution directe sur un téléphone évite tout problème lors du déploiement qui pourrait arriver si développé sur émulateur.

## 2.5 Langages utilisés

Le C# sera utilisé pour le back-end, soit toutes les fonctions en liaison avec la base de données, les modèles et le côté algorithmique et logistique du programme. Il est très important car il est également utilisé par XAML en tant que « code-behind file ».



Le XAML est utilisé en tant que front-end. Il comprend tout un système de styles permettant de modifier la partie graphique de l'application. Il est un « dialecte » de XML, mettant en valeur la liaison au C#, ou la séparation du code visible par l'utilisateur et le code lui-même.

Tous ces langages sont réunis par Xamarin, une plateforme de développement mobile développée par Microsoft.

Multiplateforme, elle supporte Android, iOS et Windows Phone.



## 2.6 Stockage des données

Après une analyse des différentes possibilités de stockage de données, quatre options s'imposent :

1. **Shared Preferences** : Système de stockage de données basé sur un système de Clé/Valeur. Pratique pour stocker des valeurs désorganisées, p.ex un email ou un nom d'utilisateur.
2. **SQLite** : Système de base de données pouvant organiser et stocker de grosses quantités de données structurées sur le téléphone de l'utilisateur. Il ne peut stocker des données que localement.
3. **Stockage interne / externe** : Stocke des fichiers dans la mémoire interne et externe (Carte SD)
4. **Base de donnée MySql** : Système de base de données distant, hébergé sur un serveur de la salle de classe. Utilisable uniquement avec une connexion à internet, lorsque la base de donnée tourne sur le serveur.

L'option retenue est SQLite, car elle semble être la plus adaptée au projet. Les Shared Preferences ne sont pas assez structurées pour l'ampleur du projet et la base de données MySql perd son sens car il faut être sur le même réseau que le serveur, qu'il peut y avoir de

problèmes du côté serveur et qu'il faut une connexion à internet. Or, l'application n'a pas besoin d'une connexion pour fonctionner. Quant au stockage interne / externe de fichiers, il manque de structure et c'est également le domaine dans lequel je possède le moins de compétences.

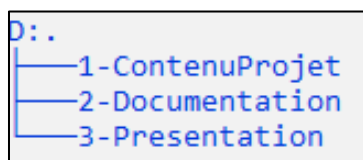
L'option retenue est SQLite car elle possède une bonne structure de base de données.

## 2.7 Sauvegarde du projet

Les fichiers du projet seront retrouvés :

- Sur Teams dans le dossier :
- Sur GitHub :
- Sur l'ordinateur « INF-N501-M804 » dans le dossier « D:\DATA\trigerber-TPI »

La structure des fichiers utilisée suit celle-ci depuis le dossier trigerber-TPI.



<b>1-ContenuProjet</b>	Contient tout le code, les ressources et l'application finie produits lors de mon projet.
<b>2-Documentation</b>	Contient la totalité de la documentation produite lors de mon projet.
<b>3-Presentation</b>	La présentation PowerPoint finale ainsi que les outils nécessaires à celle-ci.

## 2.8 Tests

Les tests font partie intégrante du projet et ont une grosse importance dans celui-ci.

Après analyse, deux choix principaux sont à ma disposition afin de réaliser les tests :

1. Tests Unitaires : Pratique pour les applications réalisant des calculs, possédant des algorithmes compliqués et beaucoup de processus tournant en arrière-plan.
2. Tests d'Interface Utilisateur (UI Tests) : Permet de tester une interface utilisateur sur différents appareils. Peut reconnaître les crashes, les inconsistances dans les transitions par exemple ainsi que des problèmes de layout sur différents appareils.
3. Tests manuels suivant des procédures prédéfinies.

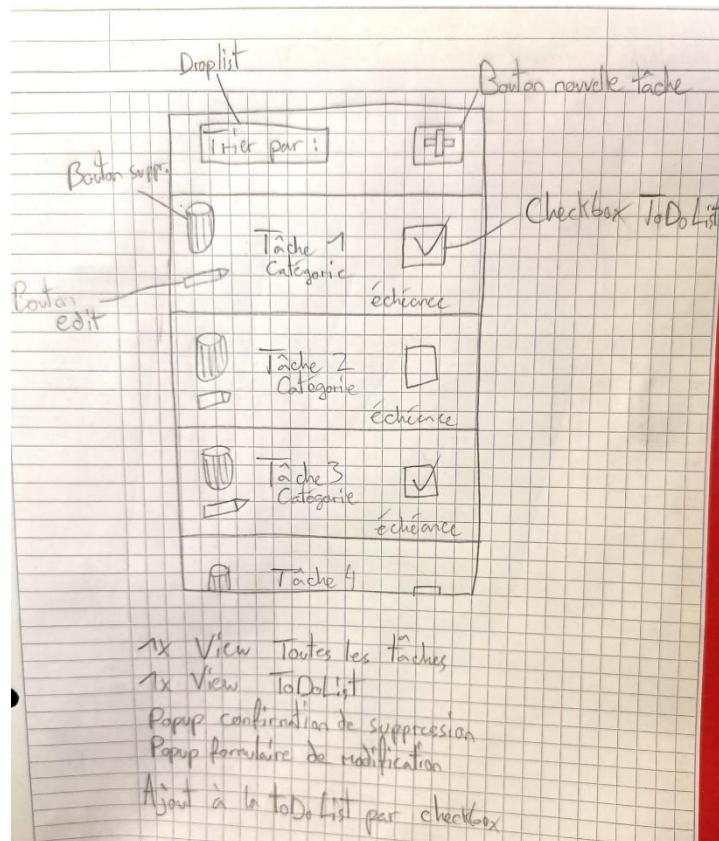
Les UI Tests étaient la méthode choisie par défaut, cependant, après discussion avec le chef de projet, il a été décidé de réaliser des tests manuels car l'application et le temps mis à disposition pour sa création étaient trop petits pour apprendre une méthode entière de test pouvant vite retarder le projet.

## 3 CONCEPTION

Dans ce document est montré le résultat de l'analyse réalisée précédemment, les maquettes du fonctionnement attendu de l'application, la structure des pages ainsi que différentes mesures prises dans la conception de l'application. Cette partie correspond à la deuxième et troisième étape de la méthode des six pas.

### 3.1 Maquettes & Structure de l'application

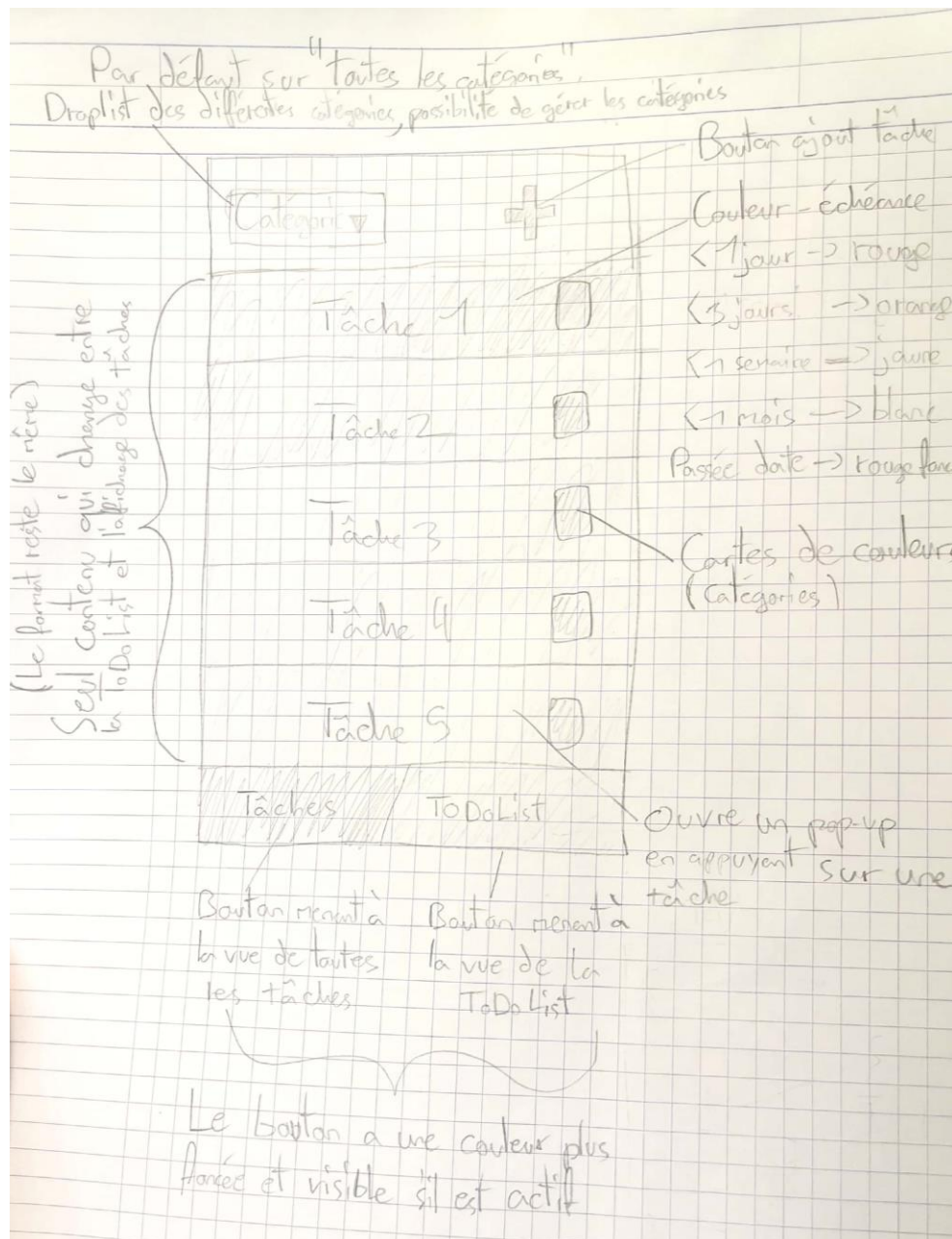
Au début du projet, une maquette « Tout en un » était le modèle de base sur lequel je partais. L'utilisateur pouvait accéder à la modification, suppression, ajout, tri et ajout à la ToDoList des tâches sur une seule même page.

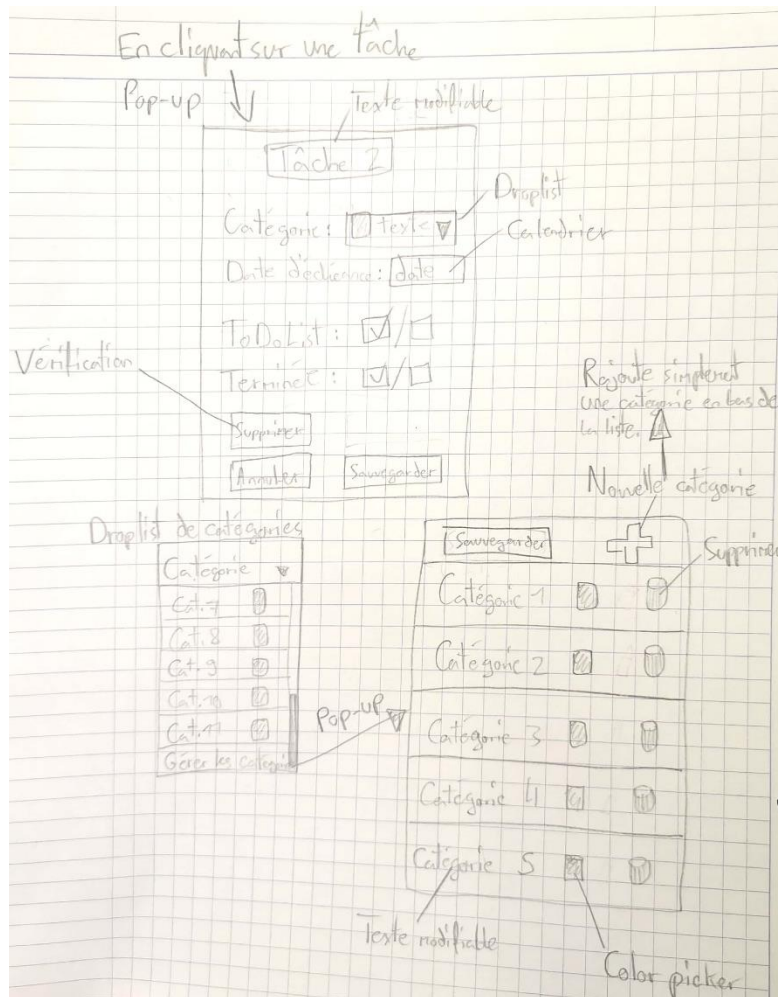


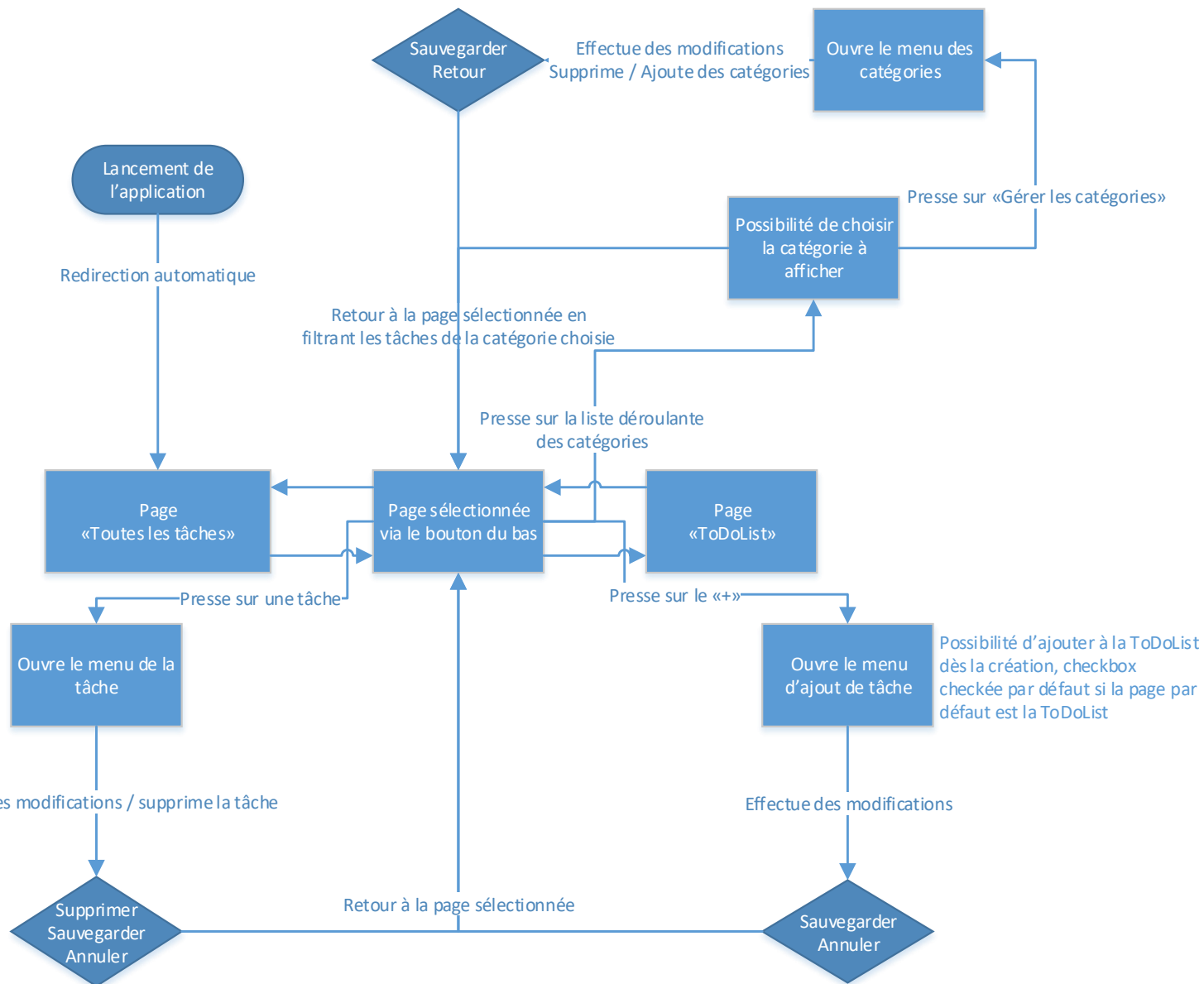
Cependant, cette solution me paraissait trop compacte et pourrait engendrer des problèmes dans le cas où l'on rajoute des informations ou des outils, qui n'auraient plus assez de place ou qui transformeraient un conteneur de tâche en menu de tâche, ajoutant trop de redondance dans le programme.



J'ai finalement opté pour une solution plus aérée dans laquelle on a juste le nom de la tâche et sa catégorie. En appuyant dessus, on peut accéder à un menu différent permettant d'en observer les détails et l'éditer. Depuis la page principale des tâches, on peut appuyer sur le « + » pour ajouter une tâche à la liste, choisir la catégorie à afficher en appuyant sur la droplist « Catégorie », cliquer sur une tâche pour accéder à son menu. La navigation entre la page des tâches et la page de ToDoList se fait par le biais d'un menu de navigation fixé au bas de l'application comprenant deux boutons liés à ces deux pages.







La navigation se fait via un menu statique tout au bas de l'application (voir figure x). Il n'y a que deux pages principales, toutes les autres étant sous forme de pop-ups. La navigation des pop-ups se fait comme s'ils étaient des pages, c'est-à-dire avec les flèches de navigation de base du téléphone.

### 3.2 Infographie de la date d'échéance

Les tâches affichées dans la liste des suivent un code de couleurs correspondant au temps restant avant la date d'échéance :

- > 1 mois → Blanc
- < 1 mois → Doré – Verger Clair Jaune (LightGoldenrodYellow)
- < 1 semaine → Doré - Verger (Goldenrod)
- < 3 jours → Orange
- < 1 jour → Orange-Rouge
- Passée date → Rouge

Les couleurs ne sont pas appliquées dans la ToDoList, étant donné que les tâches sont toutes à accomplir le jour même.

Il faut noter que si la tâche est cochée comme terminée, sa date de péremption est égale à `DateTime.MaxValue`, c'est-à-dire l'année 2100. Cela la propulse tout en bas de la liste et sa couleur devient Vert-Jaune.

### 3.3 Navigation

La navigation principale se fait avec les deux boutons du menu. L'utilisateur peut ensuite naviguer à travers l'application en appuyant sur une tâche dans la liste de tâche pour accéder à son menu, en appuyant sur une catégorie dans le menu de catégories pour y accéder, en appuyant sur le « + » pour accéder au menu d'ajout de tâches, etc...

Le retour en arrière se fait avec les flèches de navigation de base du téléphone.

Toutes les pages sauf les deux principales – La page des tâches et la ToDoList – sont sous forme de pop-ups.

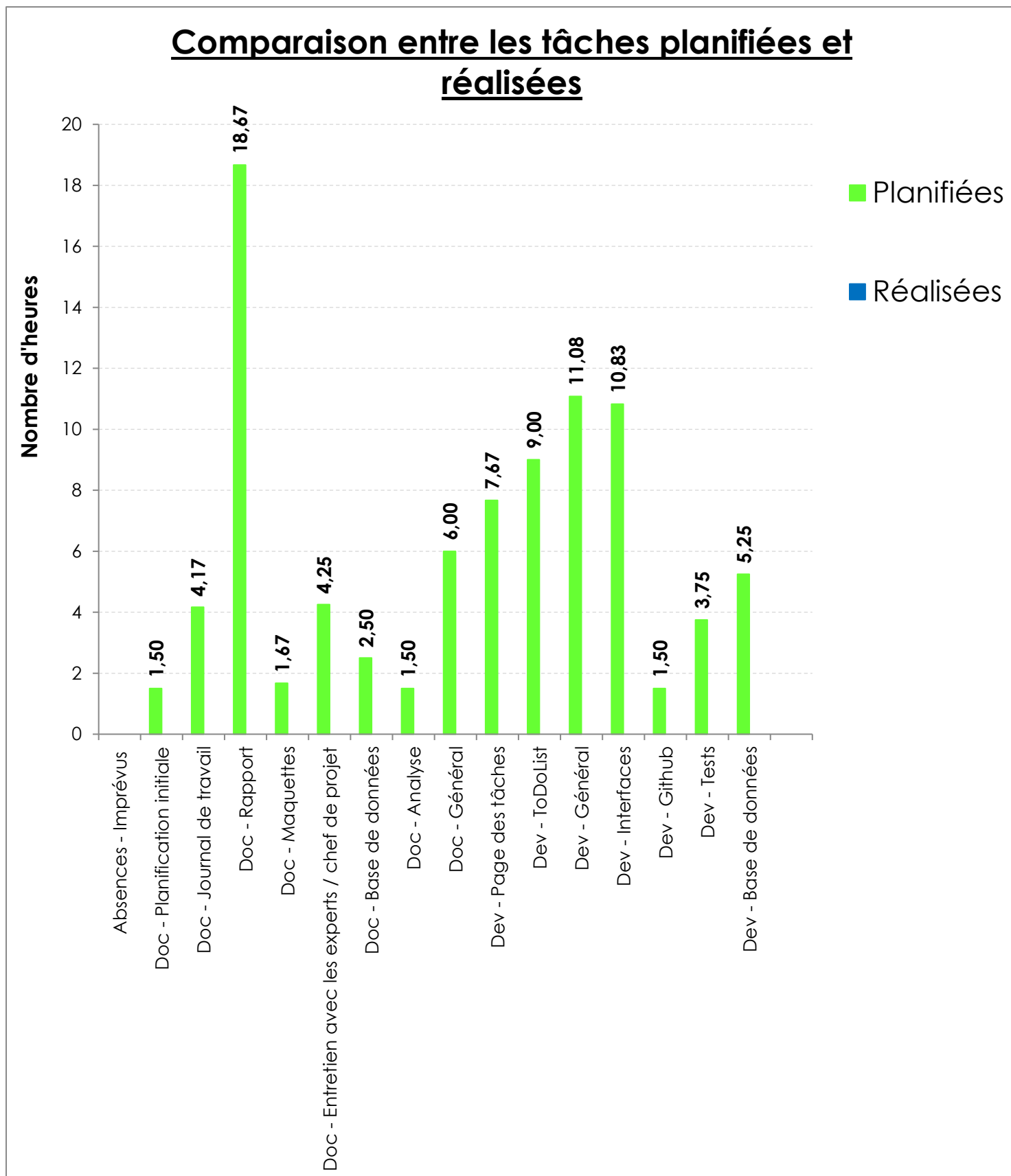
### 3.4 Conception des tests

Les tests manuels consistent en cette liste :

1. La navigation entre l'onglet des tâches et la ToDoList est fonctionnelle
2. Les tâches sont triées par ordre croissant du temps restant avant leur date d'échéance
3. Les tâches suivent le code de couleur définit plus tôt correspondant au temps restant avant leur date d'échéance.
4. L'ajout de tâche fonctionne et la tâche est directement triée avec les autres par date d'échéance
5. La suppression de tâche fonctionne, la tâche disparaît de la liste
6. Lorsqu'une tâche est cochée comme terminée, elle n'apparaît plus dans la liste
7. Le menu de tâche s'ouvre comme pop-up en appuyant sur une tâche
8. Toutes les modifications sauvegardées dans le menu de tâche sont entrées dans la base de données
9. Après avoir ajouté un objet à la ToDoList, il est visible dans celle-ci
10. Après avoir supprimé un objet de la ToDoList, il n'est plus visible dans celle-ci
11. Le menu des catégories s'ouvre comme pop-up en cliquant sur « Gérer les catégories » dans la droplist de catégorie.
12. La suppression, modification et ajout de catégories sont enregistrés dans la base de données et affichées dans la droplist de catégorie lors d'un appui sur le bouton « sauvegarder »

### 3.5 Planification détaillée

La planification initiale a été réalisée le Mercredi 4 Mai, soit après environ 1 journée de travail



## 4 RÉALISATION

### 4.1 Dossier de Réalisation






#### 4.1.1 Programmes & versions utilisés

**Code** : Visual Studio 2019

**Planification** : Excel 2016

**Schémas** : Visio 2016

**NuGet** :

	<b>NETStandard.Library</b> par Microsoft	2.0.3
	A set of standard .NET APIs that are prescribed to be used and supported together. 18a36291e48808fa7ef2d00a764ceb1ec95645a5	
	<b>Rg.Plugins.Popup</b> par Kirill Lyubimov	2.1.0
	Plugin for Xamarin forms. Allows you to open any page as a popup.	
	<b>sqlite-net-pcl</b> par SQLite-net	1.8.116
	SQLite-net is an open source and light weight library providing easy SQLite database storage for .NET, Mono, and Xamarin applications. This version uses SQLitePCLRaw to provide platform independent versions of SQLite.	
	<b>Xamarin.Essentials</b> par Microsoft	1.6.1
	Xamarin.Essentials: a kit of essential API's for your apps	
	<b>Xamarin.Forms</b> par Microsoft	5.0.0.2012
	Build native UIs for iOS, Android, UWP, macOS, Tizen and many more from a single, shared C# codebase	
		5.0.0.2401

**Rg.Plugins.Popup** permet de d'afficher des Pop-Ups et d'assurer leur bon fonctionnement.

**Sqlite-net-pcl** est nécessaire sur Xamarin pour utiliser SQLite.

**Téléphone & Version d'Android :**

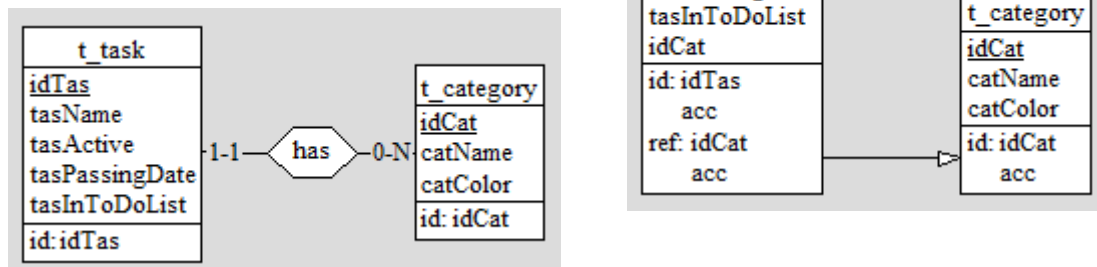


**Spécifications de l'ordinateur :**

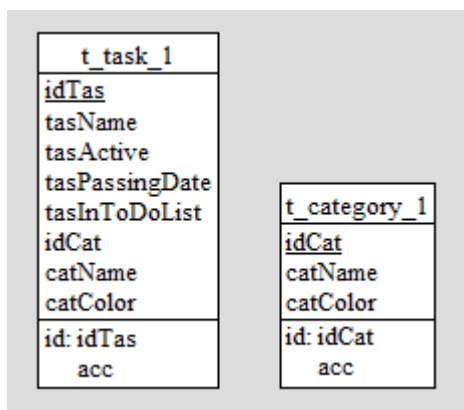
- Nom : INF-N501-M804
- Processeur : Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.41 GHz
- Mémoire RAM installée : 32,0 Go
- Type de système : Système d'exploitation 64 bits, processeur x64
- OS : Windows 10 Éducation, Version 20H2

### 4.1.2 Base de données

Les modèles de base de données sont très simples. Ils consistent en deux tables, la table des tâches et la table des catégories.



À cause d'un problème dans le code pour la connexion entre les deux tables et du manque de temps pour le régler, les deux tables ont été séparées, les attributs de la table des catégories étant chargés automatiquement dans celle des tâches au lancement et à chaque modification de la table des catégories.



Cette modification ne change rien au fonctionnement du programme, si ce n'est le rendre un peu plus lent.

### 4.1.3 MVVM & Code

La structure du code est en MVVM (Model / View / ViewModel), ce qui signifie que ce que l'utilisateur voit (View) est séparé mais connecté à un ViewModel. Le Model est un modèle de données (expliqué plus bas).

La View, vue en français est tout ce que l'utilisateur peut voir. Elle est codée en XAML (C# en arrière), et est liée au ViewModel avec plusieurs paramètres, notamment le terme « Binding » qui lie une valeur affichée avec une valeur du ViewModel ou une commande qui exécute du code.

Les ViewModels contiennent des méthodes qui traitent tout ce qui est dans le programme. Ils sont en lien avec le Model, la View et les Services.

Les Services sont en dehors de MVVM, ils peuvent faire office de classe statique, de variables globales. Dans mon programme, ils assurent la connexion avec la base de donnée

Les Modèles représentent la base de données. Pour la base de données contenant `t_task` et `t_category`, il y a un modèle `TaskModel` et un modèle `CategoryModel`.

## 4.2 Compilation

Afin de compiler l'application sur le téléphone, il faut avant tout mettre le téléphone en mode développeur. Cela se fait en suivant ces étapes :

1. Ouvrir les paramètres du téléphone
2. Aller dans « À propos du téléphone »
3. Trouver le numéro de build
4. Appuyer sur le numéro de build jusqu'à ce qu'une notification avertisse que le mode développeur est activé.

Il faut ensuite le connecter à l'ordinateur contenant le code et il se connectera automatiquement à Visual Studio si l'environnement de code Xamarin est ouvert.

Une fois cela fait, l'application peut être lancée depuis l'ordinateur et se lancera sur le téléphone.

Afin de débbugger l'application, tout ce qu'il y a à faire est de mettre un break point et de lancer le débbugger. L'application sur téléphone se figera et l'on pourra débbugger le code.

### 4.2.1 Réalisation des tests

Les tests n'ont pas été réalisés d'une traite. Ils ont été réalisés au cours du projet, car chaque test prédéfini fait d'une manière ou d'une autre partie des objectifs du projet. Tous les tests sont fonctionnels, sauf le 11<sup>ème</sup> : « *Le menu des catégories s'ouvre comme pop-up en cliquant sur « Gérer les catégories» dans la droplist de catégorie* ». Le moyen d'ouvrir le menu des catégories a légèrement changé ; il s'agit d'un bouton, au lieu d'un choix dans une droplist comme prévu. Cela n'a aucun impact sur le bon fonctionnement du programme.



## 5 CONCLUSION

Cette rubrique contient les différentes conclusions des différents thèmes de ce projet et ce que l'on peut en tirer.

### 5.1 Bilan des fonctionnalités demandées

Il doit être possible de catégoriser les tâches	OK
Il doit être possible d'afficher les tâches par catégorie.	OK
Il doit être possible d'afficher les tâches par ordre de date d'échéance.	OK
Il doit être possible d'insérer, de modifier ou de supprimer une tâche.	OK
Il doit être possible de marquer une tâche comme terminée. A ce moment elle ne doit plus être visible.	OK
Une infographie par date d'échéance. Par exemple différentes couleurs par proximité de l'échéance.	OK
Il doit être possible de créer des « ToDo list » à partir des tâches. Par exemple si l'utilisateur veut choisir les tâches qu'il veut réaliser dans la journée.	OK

### 5.2 Points techniques évalués spécifiques au projet

L'ergonomie de l'application (user friendly et simple), la facilité d'utilisation.	OK
La modélisation de la base de données (MCD, MLD et MPD).	OK
La pertinence de l'infographie proposée.	OK
Le bon fonctionnement des créations et modification des tâches.	OK
Le bon fonctionnement des affichages des tâches.	OK
Le bon fonctionnement de création d'une « ToDo list ».	OK
La qualité et lisibilité du code source (respect des normes ETML).	OK

### 5.3 Bilan de la planification

### 5.4 Bilan personnel

## 6 WEBOGRAPHIE

La webographie contient tous les liens des sites qui m'ont apporté des réponses à mes questions ou qui m'ont aidé. Les liens consultés durant une recherche qui n'ont pas été utiles sont laissés de côté, faute de pertinence.

[Comprenez les différents moyens de stocker des données sur Android - Gérez vos données localement pour avoir une application 100 % hors-ligne - OpenClassrooms](#)

[Testing Xamarin Applications with Visual Studio App Center - App Center Blog \(microsoft.com\)](#)

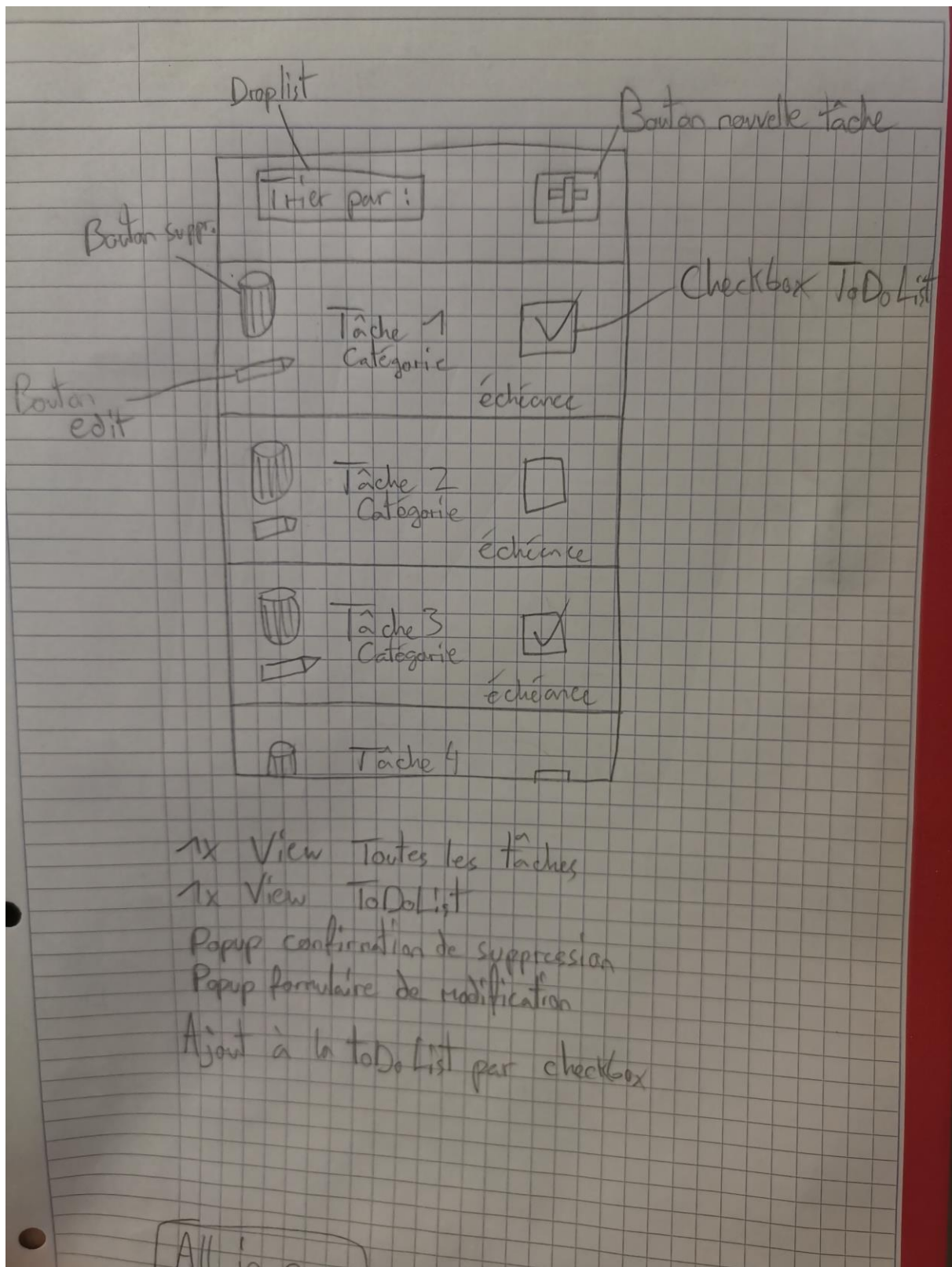
[Getting started · rotorgames/Rg.Plugins.Popup Wiki \(github.com\)](#)

[PopupPage · rotorgames/Rg.Plugins.Popup Wiki \(github.com\)](#)

[Using SQLite.NET with Android - Xamarin | Microsoft Docs](#)

[c# - Adding a button to the title bar Xamarin Forms - Stack Overflow](#)

[Rg Popup In Xamarin.Forms Using Fresh MVVM \(c-sharpcorner.com\)](#)



## 7 TABLE DES ILLUSTRATIONS

Aucune entrée de table d'illustration n'a été trouvée.

## **8 ANNEXES**

### **8.1 Planification**

## 8.2 Journal de travail