Location de matériel

Gestion de Stocks

01/02/2021

Paola Costa

Centre professionnel du Nord Vaudois

Table des matières

[1 Analyse préliminaire 3](#_Toc67907949)

[1.1 Introduction 3](#_Toc67907950)

[1.2 Objectifs 3](#_Toc67907951)

[1.3 Planification initiale 4](#_Toc67907952)

[2 Analyse / Conception 5](#_Toc67907953)

[2.1 Concept 5](#_Toc67907954)

[2.1.1 Fonctionnalités 5](#_Toc67907955)

[2.1.2 Modèles de données 5](#_Toc67907956)

[2.1.2.1 Modèle de données conceptuel 5](#_Toc67907957)

[2.1.2.2 Modèle de données logique 6](#_Toc67907958)

[2.2 Maquettes 7](#_Toc67907959)

[2.3 Use Cases & Scénarii 8](#_Toc67907960)

[2.3.1 Use Cases 8](#_Toc67907961)

[2.3.2 Scénarii 9](#_Toc67907962)

[2.3.2.1 Ajouter un objet 9](#_Toc67907963)

[2.3.2.2 Afficher les objets dont le délai est dépassé 9](#_Toc67907964)

[2.3.2.3 Générer un PDF 9](#_Toc67907965)

[2.3.2.4 Louer un objet 10](#_Toc67907966)

[2.3.2.5 Retourner un objet loué 11](#_Toc67907967)

[2.3.2.6 Rechercher par mot-clé 11](#_Toc67907968)

[2.4 Diagrammes 12](#_Toc67907969)

[2.4.1 Diagramme de classe 12](#_Toc67907970)

[2.4.2 Diagramme de flux 13](#_Toc67907971)

[2.5 Stratégie de test 14](#_Toc67907972)

[2.6 Risques techniques 14](#_Toc67907973)

[2.7 Planification définitive 15](#_Toc67907974)

[2.8 Infrastructure 16](#_Toc67907975)

[2.8.1 Matériel hardware et système d’exploitation 16](#_Toc67907976)

[2.8.2 Outils logiciels 16](#_Toc67907977)

[3 Réalisation 17](#_Toc67907978)

[3.1 Dossier de réalisation 17](#_Toc67907979)

[3.1.1 Répertoires et fichiers du projet 17](#_Toc67907980)

[3.1.1.1 Répartition physique des fichiers 17](#_Toc67907981)

[3.1.1.2 Fichiers et description 17](#_Toc67907982)

[3.1.2 Produit fini 18](#_Toc67907983)

[3.1.3 Programmation et scripts 18](#_Toc67907984)

[3.2 Description des tests effectués 19](#_Toc67907985)

[3.2.1 Tests unitaires 19](#_Toc67907986)

[3.2.2 Tests d’intégration 19](#_Toc67907987)

[3.2.3 Tests fonctionnels 19](#_Toc67907988)

[3.3 Problèmes rencontrés et résolution 20](#_Toc67907989)

[3.4 Erreurs restantes 21](#_Toc67907990)

[3.5 Liste des documents fournis 22](#_Toc67907991)

[4 Conclusions 23](#_Toc67907992)

[4.1 Atteinte des objectifs 23](#_Toc67907993)

[4.2 Maintien des délais 23](#_Toc67907994)

[4.3 Points positifs et négatifs 24](#_Toc67907995)

[4.4 Difficultés particulières 24](#_Toc67907996)

[4.5 Évolutions et améliorations 24](#_Toc67907997)

[5 Annexes 25](#_Toc67907998)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 25](#_Toc67907999)

[5.2 Sources – Bibliographie 26](#_Toc67908000)

[5.2.1 Pages internet consultées 26](#_Toc67908001)

[5.2.2 Personnes consultées 26](#_Toc67908002)

[5.3 Journal de travail 27](#_Toc67908003)

[5.4 Manuel d'Installation 31](#_Toc67908004)

[5.5 Manuel d'Utilisation 31](#_Toc67908005)

[5.6 Archives du projet 31](#_Toc67908006)

[5.7 Table des illustrations 31](#_Toc67908007)

# Analyse préliminaire

## Introduction

Avec les avancées actuelles de la technologie, il n’a jamais été aussi simple de gérer des stocks d’une entreprise. Cependant, tout le monde n’est pas encore équipé pareillement. Dans le but de préparer mon TPI, je vais donc réaliser une application de gestion de stocks pour mon pré-TPI. Celle-ci est créée dans le cadre d’une PME, spécialisée dans la location de matériel d’extérieur. Elle permettra à une personne, même novice en informatique, d’insérer ou sortir des pièces du stock (selon les arrivages ou les locations), de créer un PDF de l’inventaire d’un simple clic ainsi que de rechercher de plusieurs manières dans les données. L’intégralité des données propres à l’application seront stockées dans une base de données.

Ce projet servira tout d’abord à reprendre en main des technologies qui ont été abordées précédemment dans différents modules. Il s’agit ici du C# et du MySQL. Les mêmes technologies serviront à réaliser le TPI. Ensuite, le fait de compléter un projet individuel dans des conditions semblables à celles du TPI permet également de se projeter et de se rendre compte de comment le projet final (TPI) va se dérouler.

Enfin, aucun travail n’a été effectué eu préalable pour préparer le pré-TPI. L’intégralité du développement se fera lors du module et du temps mis à disposition.

## Objectifs

Afin de mener à bien ce projet, de nombreux objectifs sont à compléter. La validation de ceux-ci permettra de déterminer le degré de complétion du projet. L’élément principal de ce projet consiste à créer une application « clé en main ». Cela signifie que l’application sera fonctionnelle sans investissements ultérieurs. De plus, l’application doit être accessible à des novices en informatique. Un mode d’emploi sera donc mis à disposition, afin de faciliter l’utilisation.

De plus, l’application contiendra plusieurs fonctionnalités. Il s’agit de celles qui sont citées ci-dessous :

* L’application devra permettre d’ajouter des objets aux stocks.
* L’application devra permettre de louer et retourner une location.
* Il sera possible d’effectuer une recherche sur l’application.
* Les données de l’application seront stockées dans une base de données distante.
* L’application permettra de générer des rapports au format PDF, afin de réaliser des inventaires.
* L’application permettra d’effectuer un tri dans les données.
* L’application devra permettra d’afficher l’historique d’une pièce spécifique.

Enfin, afin de faciliter l’accessibilité à l’application, l’application sera accompagnée d’une procédure d’installation et de mise en service.

## Planification initiale

Comme le cadre de travail est clairement défini dès le départ, le projet se déroulera sur un mode cascade. La répartition des tâches est faite dès le départ et une comparaison des diagrammes de Gant sera réalisée à la fin du projet.

# Analyse / Conception

## Concept

### Fonctionnalités

En plus des différents éléments abordés dans le point [1.2 Objectifs](#_Objectifs), plusieurs points techniques spécifiques doivent être respectés. Ils sont explicités ci-dessous :

* Les données internes à l’application sont stockées dans une base de données.
* Au lancement de l’application, une connexion à une base de données distante est établie.
* Chaque pièce présente dans le stock est unique. Cela s’exprime grâce à son ID, qui est un code numérique unique.
* Il est possible d’ajouter une pièce dans le stock grâce à son ID. Celui-ci peut être déjà existant ou pas. S’il est nouveau, les informations propres à la pièce sont requises avant l’ajout.
* Il est possible de sortir une pièce du stock grâce à son ID. Il faut la durée de la location et les informations sur le loueur pour que cela s’effectue.
* Il est possible de générer deux types de rapports PDF différents, afin de réaliser un inventaire. Le premier regroupe le contenu actuellement en stock, le deuxième le matériel loué.
* Une fonction de recherche par mot-clé est disponible sur l’application.
* Une fonction de tri permet de voir les pièces dont la date de retour de location est dépassée.
* Il est possible de consulter l’historique de location d’une pièce.

### Modèles de données

#### Modèle de données conceptuel

Figure 1 : modèle conceptuel de données

#### Modèle de données logique

Figure 2 : modèle logique de données

Dans la table « locations », il y a trois clés étrangères :

* object\_id représente l’objet qui est loué
* locator\_id représente la personne qui loue l’objet
* employee\_id représente la personne de l’entreprise qui enregistre la location. Cette information est présente afin de pouvoir créer des logs plus complets.

La table « roles » a pour but de classifier les personnes en trois groupes : utilisateur, administrateur et client. Les clients n’ont pas accès à l’application. Les administrateurs gèrent les utilisateurs.

La fonctionnalité de gestion n’est pas encore mise en place, la présence de cette table a pour but de permettre une évolution future de l’application.

Dans la table « objects », la colonne adder\_id sert à identifier la personne qui a ajouté l’objet dans le stock. Encore une fois, il s’agit de pouvoir créer un log complet.

La raison pour laquelle adder\_id (table « objects ») et employee\_id (table « locations ») peut être nul est pour permettre la suppression d’un employé (rôle utilisateur) sans être impacté parce qu’il aurait procédé à une location non finie. Cela aura pour conséquence d’afficher « NULL » à l’emplacement de son nom si on récupère les logs.

## Maquettes

L’intégralité des maquettes se rapportant à l’application seront présentées ici. Celles-ci permettent de visualiser l’apparence du produit fini.

Cette maquette représente le visuel global de l’application, avec l’intégralité des fonctionnalités.



Figure 3 : maquette – vision globale de l'application

Les boutons à droite servent à effectuer les actions sur l’application. Celles-ci sont variables.

Certains des boutons permettent d’ouvrir des formulaires à remplir, afin de renseigner des détails supplémentaires.

Cette maquette montre l’apparence d’un formulaire. Il s’agit ici de celui d’ajout d’un objet au stock.



Figure 4 : maquette – formulaire d'ajout d'objet

Les textes en gris clair représentent des placeholders. Ils sont visibles tant que le curseur n’est pas dans le champ texte.

Tout le projet va fonctionner sur le principe de ces deux maquettes. Dès qu’un formulaire doit être utilisé, le visuel sera semblable à celui de la Figure 2.

Un pop-up de confirmation valide chaque clic sur un bouton, la sélection par défaut de la confirmation est « Valider ».

## Use Cases & Scénarii

### Use Cases

Comme montré par le schéma, l’intégralité des actions se font en tant qu’employé. Ce diagramme permet de montrer les cas d’utilisation de l’application.



Figure 5 : use cases

### Scénarii

#### Ajouter un objet

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé clique sur « ajouter un objet » |  |  |
| L’employé donne un nom générique à l’objet (clé à molette) | Le nom n’existe pas du tout | Un formulaire s’ouvre, il faut remplir les spécificités de l’objet |
|  | Le nom existe déjà (clé à molette 2) | Le système demande une confirmation qu’il s’agit du même objet |
| Validation du pop-up | il s’agit du même objet | L’objet est ajouté avec un nouveau numéro aux objets en stock |
|  | Il ne s’agit pas du même objet | Un formulaire s’ouvre, il faut remplir les spécificités de l’objet |
| L’employé entre les spécificités et valide et ajoute l’objet | Toutes les données sont entrées | L’objet est ajouté au stock, message comme quoi ça a bien fonctionné, retour à l’accueil |
|  | Des données sont manquantes, de type erroné | Un message d’erreur apparaît, retour à la page d’accueil de l’application |

#### Afficher les objets dont le délai est dépassé

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé clique sur « afficher les objets avec délai dépassé » |  | Seul les objets actuellement en location et dont la date de retour prévue est antérieure à la date actuelle sont affichés à l’écran |

#### Générer un PDF

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé clique sur « générer un PDF » |  | Un pop-up avec 3 options s’affiche : « annuler », « location », « en stock » |
| L’employé sélectionne « location » |  | Un PDF avec tous les objets loués est généré |
| L’employé sélectionne « en stock » |  | Un PDF avec tous les objets en stock est généré |
|  |  | Retour à la page d’accueil |
| L’employé sélectionne « annuler » |  | Retour à la page d’accueil |

#### Louer un objet

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé clique sur « louer un objet » |  |  |
| L’employé tape le nom ou l’id de l’objet | Le nom ou l’id existe dans la base de données et l’objet n’est pas loué | Un pop-up de confirmation apparaît, demandant si c’est le bon objet à louer |
|  | Le nom ou l’id sont incorrects  l’objet avec ce nom est déjà loué | Un message d’erreur apparaît, retour à la page d’accueil |
|  | Le nom ou l’id sont incorrects  l’objet avec cet ID est déjà loué |
| L’employé invalide la confirmation de l’objet sélectionné |  | Retour à la page d’accueil |
| L’employé valide la confirmation |  | Un formulaire de location s’ouvre, pour spécifier les données du locataire |
| L’employé remplit le champ n° de client et la date de retour de location. Le reste du formulaire est grisé | Le client existe | Un pop-up de confirmation apparaît |
|  | Le client n’existe pas | Retour à l’accueil |
| L’employé remplit tout le formulaire, sauf le champ n° de client | Un client identique existe déjà | Un pop-up pour savoir si le client voulu est celui affiché apparaît |
|  | Le client n’existe pas | Un pop-up de confirmation de la location apparaît |
| Le client est bien celui indiqué |  | Un pop-up de confirmation de la location apparaît |
| Le client n’est pas le même |  | Un message d’erreur apparaît, retour à l’accueil |
| Validation de la confirmation de location | Le client existait déjà | Message de réussite, retour à l’accueil |
|  | Le client n’existait pas | Message de réussite, message d’ajout du client à la base de données, retour à l’accueil |

#### Retourner un objet loué

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé clique sur « retourner un objet » |  | Un formulaire s’affiche |
| L’employé entre le n° ou le nom de l’objet et valide | Le nom ou l’id est valide | La liste des éléments correspondants loués apparaît sur le formulaire |
|  | Le nom et/ou l’id sont invalides | Message d’erreur. Retour à la page d’accueil |
| L’employé sélectionne l’objet qu’il doit retourner et valide |  | Un pop-up s’affiche pour valider |
| L’employé valide |  | Message de succès et retour à la page d’accueil |
| L’employé annule. |  | Retour à la page d’accueil |

#### Rechercher par mot-clé

|  |  |  |
| --- | --- | --- |
| **Action** | **Condition** | **Réaction** |
| L’employé tape du texte dans le champ de recherche et appuie sur le bouton « Rechercher » | Le champ de texte est vide | Rien ne se passe |
|  | Le champ de texte contient du texte | Un tri s’effectue. Seules les données avec une correspondance partielle s’affichent |
|  | Il n’y a aucune correspondance entre les données et le texte de recherche | Aucune entrée n’est présente dans le tableau |
|  | Toutes les données correspondent à la recherche | Toutes les données s’affichent dans le tableau |
| L’employé tape un autre texte et appuie sur « Rechercher » |  | La recherche repart de zéro, la liste affichée à l’écran n’impacte pas le résultat |

## Diagrammes

### Diagramme de classe



Figure 6 : diagramme de classe

### Diagramme de flux

Figure 7 : diagramme de flux – ajouter un objet

Figure 8 : diagramme de flux – retourner un objet



## Stratégie de test

Afin de tester l’application, trois types de tests vont être effectués : les tests unitaires, les tests d’intégration et les tests fonctionnels. Ils sont réalisés de manière automatique grâce à des classes de tests spécifiques sur Visual Studio 2019. Les tests d’intégration s’effectuent entre différentes unités et valide le bon fonctionnement de ces unités entre elles. Les tests fonctionnels permettent de valider que l’application fonctionne comme elle doit fonctionner et fait ce qui est attendu.

Pour pouvoir détecter les problèmes au plus vite, les tests seront réalisés tout au long du développement. Chaque fonctionnalité subira tout d’abord un test unitaire. Lors de la liaison de cette fonctionnalité avec une autre, un test d’intégration sera fait. Les tests fonctionnels consisteront à vérifier que les fonctionnalités intégrées à l’application fonctionnent effectivement et retournent ce qu’elles doivent.

Les tests unitaires se doivent d’être exhaustifs, afin de détecter les soucis avant d’intégrer l’élément au projet. Autant que possible, les tests d’intégration et fonctionnels seront eux aussi exhaustifs.

Dans le cadre des tests fonctionnels, les données à utiliser pour réaliser les tests seront les données réelles, fournies avec le script de création de la base de données. Les tests unitaires et d’intégration se feront directement sur Visual Studio 2019, avec des classes de tests spécifiquement créées.

La majorité des tests étant automatisée, aucun testeur externe ne sera utilisé.

## Risques techniques

Malheureusement, le risque zéro n’existe pas. Voici donc les principaux risques liés à ce projet, ainsi que les solutions utilisées pour garantir la meilleure réalisation possible.

Tout d’abord, le risque de ne pas réussir à résoudre un souci d’implémentation est relativement élevé. Afin de garantir la détection la plus rapide et le temps de débogage le plus court possible, des tests unitaires seront mis en place. Cette mise en place est prioritaire, car elle permettra d’éviter une perte de temps conséquente par la suite.

Ensuite, la liaison de la base de données à l’application C# est un élément technique spécifique, qui n’a pas été abordé en cours. Pour éviter une perte de temps liée à l’apprentissage de l’implémentation de cet élément, l’aide de Mme F.Andolfatto sera demandée. Il s’agit d’une enseignante d’informatique au CPNV.

Enfin, la longueur du projet peut être un souci majeur. Afin d’éviter cela, la planification proposée respecte des délais assez serrés, laissant cependant un peu de marge en cas de besoin. Cela permet d’empiéter sur cette marge si une tâche en vient à s’éterniser.

## Planification définitive



Les variations par rapport à la planification initiale proviennent d’un manque de temps estimé pour la partie analyse. Cela a donc décalé l’intégralité des tâches de développement afin qu’elles se situent à la suite de l’analyse/conception.

## Infrastructure

### Matériel hardware et système d’exploitation

L’infrastructure hardware utilisée pour réaliser ce projet consiste uniquement en un ordinateur du CPNV, installé avec un système d’exploitation Windows10 x64.

### Outils logiciels

Développement en C# : Microsoft Visual Studio Enterprise 2019, Version 16.8.5

Paquet : MySql.Data version 8.0.23

Paquet : iTextSharp version 5.5.13.2(création de PDF)

Base de données : MySQLWorkbench, version 6.3.6 build 511 CE (64bits)

Historisation des données : L’intégralité du pré-TPI est disponible sut GitHub. Les données sont présentes à l’adresse suivante :

<https://github.com/AhVen98/gestionStocks_preTPI>

GitHub est la méthode de versioning utilisée dans ce projet. Il s’agit du moyen mis en place pour pouvoir récupérer les anciennes versions du projet en cas de soucis/besoins.

# Réalisation

## Dossier de réalisation

### Répertoires et fichiers du projet

#### Répartition physique des fichiers

Le projet C# consiste en une solution globale, « locationMateriel ». Celle-ci comporte 3 parties, appelés projets.

* La première, « locationMaterielLibr », contient l’intégralité des classes d’objets ainsi que les méthodes utilisées.
* La seconde, « locationMaterielView », regroupe l’intégralité des WindowsForms. C’est cette partie qui contient tout ce qui a trait au côté graphique du projet.
* La troisième et dernière partie est « locationMaterielTests ». Il s’agit du projet qui contient tous les tests unitaires effectués lors du projet, afin de s’assurer du bon fonctionnement des éléments individuellement.

#### Fichiers et description

Partie 1 : locationMaterielLibr

* Persons.cs : il s’agit du fichier où les attributs représentant une personne sont définis.
  + Clients.cs : contient les informations propres aux clients. Elle hérite de « Persons.cs »
  + Employees.cs : contient les informations contenant les employés. Il y a également toutes les méthodes permettant d’ajouter, louer et retourner des objets. Cette classe hérite également de « Persons.cs ».

Clients n’est actuellement pas utilisée, sa mise en place provient de l’analyse. La classe n’a pas été supprimée, car dans le cas où l’application serait poursuivie, sa présence permettra un système d’authentification et d’ajout de client.

* Objects.cs : ce fichier contient la classe avec les objets. Un objet représente n’importe quel élément du stock. Il s’agit du point névralgique de l’application. Sans cette classe, aucune donnée ne s’affiche sur l’application. Les seules méthodes mises en place sont les constructeurs et les getters.
* ConnectionDB.cs : il s’agit du fichier contenant les méthodes génériques reliées à la base de données. Il permet d’éviter de copier-coller sans arrêt les mêmes lignes de code.
* RequestDB.cs : ce fichier contient toutes les méthodes liées aux requêtes à la base de données. Il s’agit d’un point central de l’application. Sans ce fichier, aucune information ne peut être lue, ajoutée ou modifiée dans la base de données.

Partie 2 : locationMaterielView

Chaque fichier cité ci-dessous contient en réalité deux fichiers. Le premier, « .Designer.cs », comprend la partie graphique du fichier. Le second, « .cs », comprend les méthodes liées à cette partie graphique.

* AddObjectForm : il s’agit du formulaire permettant d’ajouter un objet à la base de données
* ChoicePDFForm : ce formulaire permet de sélectionner le type de PDF à imprimer. Cette partie de l’application n’est pas encore fonctionnelle pour le moment, mais le formulaire est mis en place.
* locMateriel : il s’agit du visuel principal. C’est à partir de celui-ci que tous les autres formulaires sont accessibles par des boutons, placés sur l’application. Toutes les fonctionnalités demandées dans le cahier des charges sont placées sur le visuel, mais toutes ne sont pas fonctionnelles.
* RentObjectForm : il s’agit du formulaire permettant d’effectuer la location d’un objet.
* ReturnObjectForm : ce formulaire permet de rendre un objet déjà loué.

Partie 3 : locationMaterielTests

Cette partie est censée comporter l’intégralité des tests unitaires effectués dans ce projet. Au final, suite à une conception incompatible avec cette méthode, les tests unitaires n’ont pas été automatisés. Les fonctions ont cependant été testées lors des tests fonctionnels.

* ObjectsTests.cs : ce fichier contient les tests liés à la classe Objects.

### Produit fini

Le produit est actuellement en version 0.7. Il s’agit d’une version partiellement utilisable. Elle est simple d’utilisation, mais manque encore de certains éléments nécessaires. De même, certains des boutons affichés n’exécutent rien du tout.

Les logiciels utilisés ont les versions suivantes :

* Microsoft Visual Studio Enterprise 2019, Version 16.8.5
  1. MySql.Data version 8.0.23
  2. iTextSharp version 5.5.13.2(création de PDF)
* MySQLWorkbench, version 6.3.6 build 511 CE (64bits)

La version du système d’exploitation est du Windows10 x64. Étant donné le projet et les fichiers qui ont dû être créés, il est nécessaire d’être attentif au fait que l’application est fonctionnelle uniquement sur un système d’exploitation Windows.

### Programmation et scripts

**Interaction avec la base de données :**

La base de données s’intitule locationMateriel. Afin de générer la base de données, deux fichiers existent, « createDatabase.sql » et « insertData.sql ». Le 1er fichier permet de générer la structure de la base de données. Le 2ème contient les données afin de pouvoir tester la base.

Même si les infos sont peu nombreuses, elles sont suffisantes pour permettre de tester les différentes fonctionnalités.

Afin d’interagir avec la base de données, il est nécessaire d’installer le paquet MySql.Data sur Visual Studio. Cela permet d’obtenir les méthodes nécessaires à faire les requêtes.

Les deux fichiers concernant les requêtes à la base de données, dans le projet, sont les suivants :

* ConnectionDB.cs
* RequestDB.cs

Le premier (ConnectionDB) contient l’intégralité des méthodes nécessaires pour pouvoir effectuer l’entièreté des étapes d’une requête SQL. Le deuxième (RequestDB) contient les requêtes à proprement parler.

## Description des tests effectués

Les tests sont tous réalisés dans l’environnement sur lequel le développement a été réalisé. Les preuves sont apportées par capture d’écran, jointes au document ci-dessous.

Dans certains cas, les tests ont été mis en place et ont dû être supprimés par la suite, car leur réalisation n’était au final pas faisable. Cela est expliqué au point [3.2.1 Tests unitaires](#_Tests_unitaires).

### Tests unitaires

Les tests unitaires sont des tests effectués sur les éléments de manière séparée. Ils sont présents afin d’éviter les soucis lorsque tout est mis en commun par la suite.

* Tests de la classe « Objects »

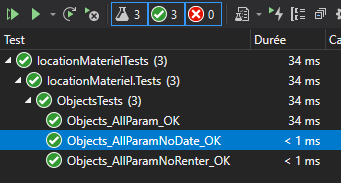


Figure 9 : validation des tests de la classe "Objects"

* Tests de la classe « Employees », « Clients », « Persons »

Les tests ne sont pas concluants et irréalisables de manière automatisée, car les getters (afin d’accéder aux données depuis l’extérieur de la classe) ne sont pas mis en place.

* Tests de la classe « RequestDB » et « ConnectionDB »

Il n’existe pas vraiment de manière de tester les deux classes, car il s’agit d’appels sur une base de données. SI des soucis avaient lieu dans ces méthodes, le programme ne s’exécuterait tout simplement pas.

Les différentes vues de l’application (fichiers WindowsForms) représentent une agglomération des tous les éléments individuels. Ils sont testés dans la partie [3.2.3 Tests fonctionnels](#_Tests_fonctionnels).

Pour les tests unitaires qui n’ont pas pu être réalisés, le bon fonctionnement des éléments sera prouvé lors des tests d’intégration et des tests fonctionnels. Il n’est en effet pas possible de faire fonctionner l’entièreté de l’application si une unité est défectueuse.

### Tests d’intégration

### Tests fonctionnels

Pour chaque partie testée de votre projet, il faut décrire :

* Les conditions exactes de chaque test
* Les preuves de test (papier ou fichier)
* Tests sans preuve : fournir au moins une description

## Problèmes rencontrés et résolution

[18.02.2021] Script d’insertion des données

Lors de la création du script d’insertion des données, deux soucis majeurs se sont posés. Le premier a été de trouver des données pertinentes. Le but est d’avoir des données suffisamment variées pour que toutes les possibilités soient clairement visibles, sans pour autant avoir 1'500 entrées dans la base de données.

Ce souci a été résolu à l’aide de Wikipédia, où je me suis inspirée pour nommer les catégories et les types.

[01.03.2021] Mise en place des classes pour les tests unitaires

Lors de la mise en place des classes des tests unitaires le lundi matin 1er mars 2021, un souci s’est présenté : les classes appelées depuis le reste du projet n’étaient pas reconnues par la classe de test.

L’aide de l’enseignant J.Ithurbide a permis de résoudre le problème (la classe « Objects » n’était pas déclarée publique et le deuxième projet de la solution n’était pas référencé dans le projet de test) et a prévenu un souci qui aurait pu se poser au futur : l’organisation des fichiers.

L’organisation qu’il a proposée suit une architecture MVC, qui permet de séparer en 3 blocs distincts l’application.

[05.03.2021] Interaction entre plusieurs fichiers

Lors de l’implémentation des premières requêtes (fichier « RequestDB »), au moment de faire appel à des méthodes présentes dans un autre fichier, des erreurs de référence sont apparues.

L’aide de F.Andolfatto a permis de corriger le problème très rapidement : il s’agissait d’un oubli de ma part. Aucun objet n’était initialisé au début de la classe.

[05.03.2021] Faire appel à un autre projet dans une solution unique

La modification de la structure du projet en architecture MVC a vu un nouveau problème apparaître : la présence de types de projets différents. L’entièreté des données sont stockées dans un projet de type librairie, alors que les visuels et le contrôleur sont dans un projet de type WindowsForms. Hors, dans un projet de type « Bibliothèque de classe (.NET standard) », aucune référence ne s’affichait.

Avec F.Andolfatto, plusieurs tests successifs de création de projets ont permis de démontrer que le type de projet utilisé ne permettait pas le référencement. Afin de pouvoir avoir des références, il a été nécessaire de déplacer les fichiers dans un nouveau projet avec un type différent : « Bibliothèque de classe (.NET Framework) ». Ce type permet de référencer d’autres projets. Il est cependant incompatible avec des systèmes IOS ou Android.

[08.03.21] Débug des requêtes SQL

La création de nombreuses méthodes utilisées pour exécuter une seule requête augmente la probabilité d’avoir des erreurs/bugs lors de l’exécution du programme. Plusieurs de ces soucis se sont déclarés aujourd’hui. Entre eux, des messages de fonctionnement impossible, car un processus restait ouvert.

L’aide de X.Carrel a permis de résoudre l’entièreté de ces erreurs. La principale était une étape oubliée dans le fonctionnement d’une requête en C#. Il est nécessaire de supprimer le DataReader une fois qu’il a été exécuté et cette étape n’avait pas été faite.

[12.03.21] Absence

Un état grippal, avec une difficulté à respirer (nez et gorge pris) et un mal de crâne carabiné, ont fait qu’aucune avancée n’a été réalisée sur le projet ce vendredi. Il y a donc 2 périodes « à vide » cette semaine-là.

Aucune période de rattrapage n’est prévue, il y aura juste 90 minutes de moins sur le temps total, avec l’impact que cela peut avoir sur l’avancée du projet.

[26.03.21] Affichage de données en multiples colonnes

Un souci majeur, qui a posé problème tout le long du projet, a été l’affichage des données. Cela a posé comme problème majeur des retards accumulés tout le long du projet avec une application qui, ce vendredi 26 mars 2021, n’affichait encore rien du tout.

L’aide de F.Andolfatto a permis de trouver une solution. Il s’agissait de l’utilisation d’un mauvais outil d’affichage (ListView au lieu de DataGrid) et une complexité trop élevée au niveau des redirections dans les fichiers. La simplification qui a été apportée a permis de dépanner la majeure partie de l’application et d’avoir enfin un résultat qui affiche quelque chose de concret.

Il reste cependant beaucoup de travail et certaines données sont encore manquantes.

## Erreurs restantes

Arrivé à la fin du projet, de nombreuses erreurs sont encore présentes. Les voici présentées ci-dessous, avec une description associée, quelles conséquences en découlent et des moyens de le corriger.

Il est clair qu’il suffit de temps pour résoudre la majeure partie de ces erreurs, il n’est pas censé y avoir d’éléments qui pourrait rendre le projet infaisable.

Affichage des dates :

L’affichage des dates présente un souci. Peu importe si une date est donnée ou pas, on voit systématiquement « 01.01.0001 00:00:00 » affiché.

Cela n’a aucune conséquence réelle sur l’application, ce n’est que de l’affichage.

Afin de régler ce souci, il va falloir gérer les variables de type DateTime. Il s’agit probablement de l’élément qui a posé problème, lors des conversions.

Bouton de tri :

Le bouton permettant de trier par délai dépassé n’est pas fonctionnel à cause d’un manque de temps. Il n’a pas été possible d’arriver jusque-là dans l’application. Il est cependant possible de trier par la colonne « date de retour attendue », même si, avec le problème d’affichage de dates, ça n’a actuellement aucune utilité.

Par rapport aux fonctionnalités, cela n’a pas de conséquences autres qu’empêcher de trier les données pour voir uniquement les objets loués avec un délai dépassé.

Il n’y a rien à corriger à cet endroit, il faut juste implémenter le tout.

Bouton Rechercher :

Le champ texte et le bouton Rechercher à côté ne sont actuellement pas fonctionnels. Encore une fois, il s’agit d’un manque de temps pour la réalisation. Comme il ne s’agit pas d’une des fonctionnalités de base de l’application, nécessaire au bon fonctionnement, cela a été repoussé et n’a au final pas été fait.

La seule conséquence liée à ce souci est une absence de recherche par terme, cela n’impacte pas le reste de l’application.

L’implémentation du bouton est tout ce qui est nécessaire pour que cette partie soit fonctionnelle.

Accès à l’historique d’un objet :

Cette fonctionnalité, bien que centrale, n’a pas d’impact réel sur les autres. De plus, elle nécessite que l’intégralité des autres fonctionnalités soient en place pour pouvoir être développée. Vu le retard accumulé sur le reste du projet, il n’y a donc pas eu de temps pour la réaliser. Le bouton est présent, mais n’a aucun effet.

À part l’absence de la fonction d’historique de l’objet, il n’y a pas de conséquences négatives sur l’application.

Le développement du code derrière le bouton suffira à créer l’historique d’un objet.

Génération du fichier PDF :

La génération d’un PDF est un point technique très spécifique, qui nécessite, encore une fois, que le reste des fonctionnalités de base soit en place. Il n’a donc pas été possible de le développer, car le reste ne fonctionnait pas à temps.

L’absence de cet élément n’a aucun impact sur le reste de l’application.

Le formulaire pour le choix du type de PDF est prêt, ce qui manque est le code derrière ces boutons. Cela peut cependant être compliqué, il s’agit d’un paquet à part entière qui a été ajouté pour le permettre.

Affichage statique

Pour le moment, l’affichage sur la page principale est un affichage statique. Cela veut dire que, quand une modification est réalisée dans la base de données, cela ne modifie en rien ce qui s’affiche sur l’application.

D’un point de vue fonctionnalité, cela n’a aucun impact. Par contre, il n’est donc pas possible de se reposer sur ce qui est visible sur l’application pour savoir ce qui est réel comme information.

Afin de corriger ce souci, il faut régler le souci de recharger les données à chaque fois qu’une action est réalisée. Cela n’est pas très compliqué en théorie, mais peut vite prendre du temps.

## Liste des documents fournis

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

* Le rapport de projet
* Le manuel d'Installation (en annexe)
* Le manuel d'Utilisation avec des exemples graphiques (en annexe)
* Autres…

# Conclusions

## Atteinte des objectifs

Comme spécifié au début du projet, dans [1.2 Objectifs](#_Objectifs) et [2.1.1 Fonctionnalités](#_Fonctionnalités), de nombreux objectifs ont été émis. Tous n’ont pas pu être rempli, voici un récapitulatif rapide de l’atteinte de ces objectifs :

Le stockage des données a bien lieu dans une base de données.

La connexion à la base de données s’effectue bien au lancement de l’application. Il ne s’agit cependant pas d’une base de données distantes pour le moment, mais d’une base de données locales.

Toutes les pièces (objets) du stock sont uniques et définis comme tel par un ID.

L’ajout d’une pièce au stock peut être fait. Une pièce ajoutée par un ID déjà existant est un retour de location. Un ajout d’une nouvelle pièce demande de préciser tous les paramètres.

La location d’une pièce se fait par l’ID de l’objet et le numéro de client.

La génération du rapport en PDF n’est pas mise en place pour le moment, mais est prévue dans l’application.

La recherche par mot-clé est présente, mais n’a pas été développée actuellement.

La fonction de tri par date de retour dépassée est prévue (bouton déjà en place), mais n’a pas été implémentée actuellement.

L’historique d’une pièce n’est pas disponible, mais le bouton est prêt pour pouvoir le faire.

L’application est simple d’utilisation et convient tout à fait à des personnes débutantes dans le domaine de l’informatique.

L’application n’est pas « clé en main ». Il reste du travail de développement à réaliser pour pouvoir considérer l’application comme terminé.

D’un point de vue global, même si l’entièreté des objectifs n’ont pas été réalisés, les points les plus importants selon moi sont complétés et l’avancée du projet est significative malgré tout. Il reste cependant de la place pour de l’amélioration.

## Maintien des délais

Malgré la fin du délai imparti, il reste encore de nombreuses erreurs et de nombreux soucis présents dans l’application.

La raison principale de ce manquement a été une erreur d’analyse de ma part : lors de l’analyse, il m’était complètement sorti de tête que j’étais absente trois jours la semaine du 15 mars 2021. La résultante en a été presque 8h de moins à accorder au projet.

Une deuxième raison, qui a d’ailleurs été énoncé au tout début dans les risques techniques, a été la mise en place des requêtes liées à la base de données. De nombreuses erreurs et faux-pas à plusieurs reprises ont représenté une perte de temps conséquente.

Ensuite, ce projet s’est révélé être beaucoup plus complexe que prévu, intégrant des notions qui n’avaient pas été abordées en cours. Bien que cela ait été un plaisir de les découvrir, ainsi que de se rendre compte de la réalité du monde du travail, où l’on ne connaît pas forcément tout de la technologie que l’on doit utiliser, cela a rendu la réalisation de ce projet très compliquée et l’a énormément ralentie.

La présence de nombreuses erreurs tout au long du projet a également causé une baisse de motivation en fin de semaine 4, ce qui fait que du temps a été perdu à ce moment-là, car l’efficacité de travail n’était plus la même.

## Points positifs et négatifs

Ce projet m’a permis de découvrir beaucoup de choses. D’un certain côté, j’ai appris énormément sur l’importance d’une analyse et conception complète et détaillée. J’ai également découvert l’importance de connaître la technologie utilisée autant que possible. Ainsi, malgré une connaissance importante des WindowsForms et des projets de type librairie, je n’étais pas du tout préparée à devoir intégrer les deux éléments ensemble et cela m’a causé de nombreux soucis.

Ce projet a donc un gros point positif : pour mon TPI, qui sera le prochain projet, je suis bien plus informée sur les risques techniques, le temps nécessaire à la réalisation des différentes tâches, ainsi qu’à la manière de débugger de manière optimale un code. Un élément supplémentaire très positif a été que je me suis rendue compte de mes capacités. Je ne suis clairement pas la plus douée en développement, mais je suis suffisamment compétente pour faire une application qui tient la route, bien documentée avec tous les éléments nécessaires pour que le projet soit repris par quelqu’un d’autre si besoin est.

Malgré cela, il y a eu un côté un peu plus noir à ce projet.

Un des points négatifs du projet a été son avancée. Malgré tous les efforts fournis, le projet n’est pas complété et c’est un élément qui m’embête particulièrement. Un autre point négatif a été mon incapacité à régler certains problèmes seuls et d’avoir dû demander de l’aide pour le faire. Les résoudre seule aurait pu, potentiellement, se faire, mais cela aurait demandé un temps impressionnant et aurait donc diminué l’avancée du développement de manière significative.

## Difficultés particulières

## Évolutions et améliorations

# Annexes

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

## Sources – Bibliographie

### Pages internet consultées

Page des TPI vaudois : <http://www.tpivd.ch/> [12.02.21]

Recherche d’informations sur le contenu nécessaire dans le rapport de projet

Documentation Microsoft, pour la création de tests unitaires [01.03.2021]

<https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019>

Recherche de la manière correcte de mettre en place un projet de tests dans une solution

Forum commentcamarche.net, pour la syntaxe d’appel de formulaire [04.03.2021]

<https://codes-sources.commentcamarche.net/forum/affich-150911-appel-d-un-form-a-partir-d-un-autre-en-c>

Recherche de la syntaxe pour faire un appel sur un autre formulaire, afin d’avoir un code le plus propre possible

Article sur c-sharpcorner.com, pour la mise en place du PDF [15.03.21]

<https://www.c-sharpcorner.com/UploadFile/f4f047/generating-pdf-file-using-C-Sharp/>

Recherche d’un paquet nugget pour pouvoir créer des fichiers PDF à partir d’une application C#

Documentation Microsoft, sur le fonctionnement des listViews [26.03.21]

<https://docs.microsoft.com/en-us/dotnet/desktop/winforms/controls/how-to-add-and-remove-items-with-the-windows-forms-listview-control?view=netframeworkdesktop-4.8>

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.listviewitem?view=net-5.0>

Recherche sur la manière de faire fonctionner les listViews. Cet élément a finalement été abandonné pour le DataGrid

### Personnes consultées

Mme F.Andolfatto

* [05.02.21] Précisions sur la liaison des bases de données à un projet en C#
* [12.02.21] Retour sur la pertinence et l’exactitude d’un diagramme de classe
* [05.03.21] Interactions entre différents fichiers et projets
* [26.03.21] Résolution du problème d’affichage des données, à l’aide de l’élément « DataGrid »

M J.Ithurbide

* [01.03.21] Résolution du problème de mise en place du projet de test et explication sur l’architecture MVC dans un projet incluant WindowsForms et librairie
* [08.03.21] Conversion des données d’un type à un autre (ici, string à int) en C#

M X.Carrel

* [08.03.21] débug de requêtes SQL en C#

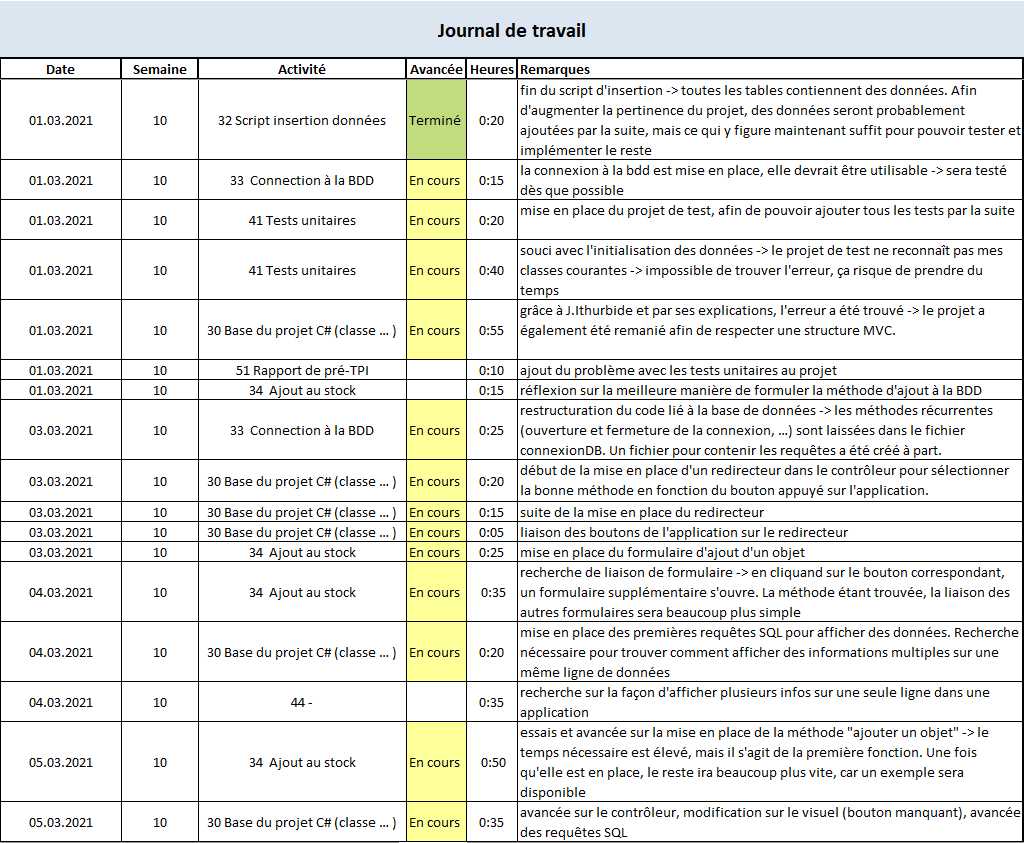
## Journal de travail

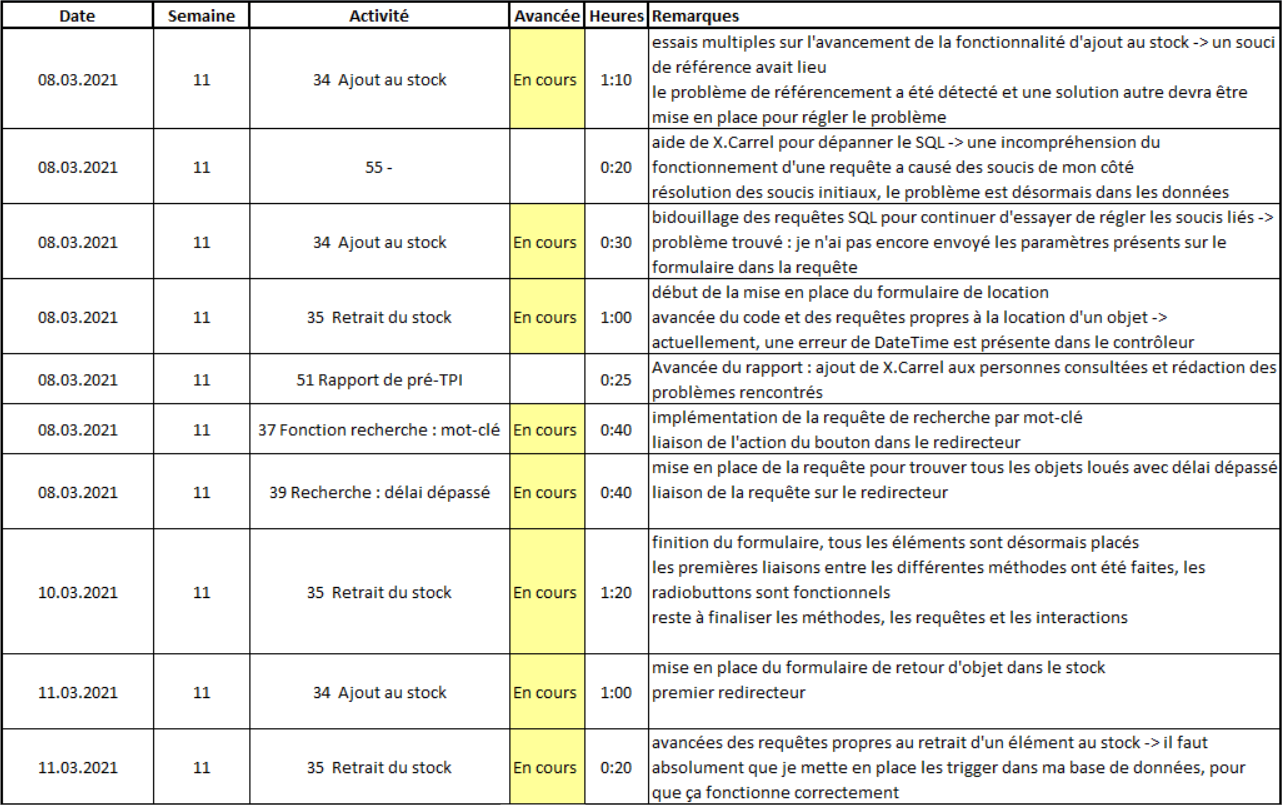
Ceci est le journal de travail de la semaine 6 de 2021. Il s’agit de la semaine n°1 du projet, du 1er février 2021 au 5 février 2021.



Ceci est le journal de travail de la semaine 7 de 2021. Il s’agit de la semaine n°2 du projet, du 8 février 2021 au 12 février 2021.

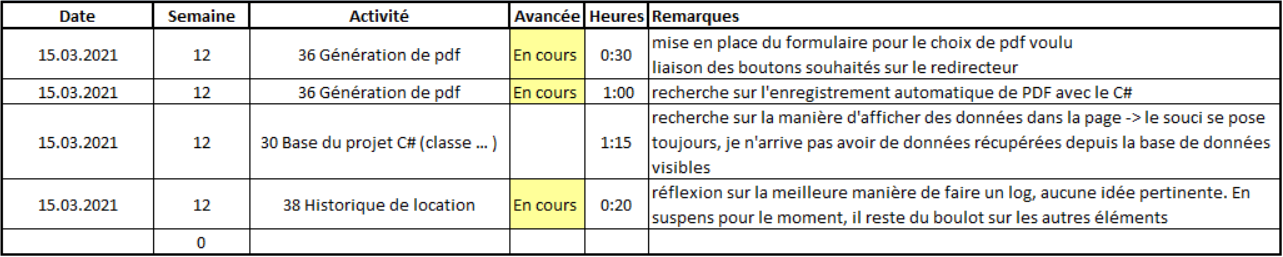
Ceci est le journal de travail de la semaine 8 de 2021. Il s’agit de la semaine n°3 du projet, du 15 février 2021 au 19 février 2021.

Ceci est le journal de travail de la semaine 10 de 2021. Il s’agit de la semaine n°4 du projet, du 1er mars 2021 au 5 mars 2021. La semaine non-présente (semaine 9) découle des vacances de carnaval.

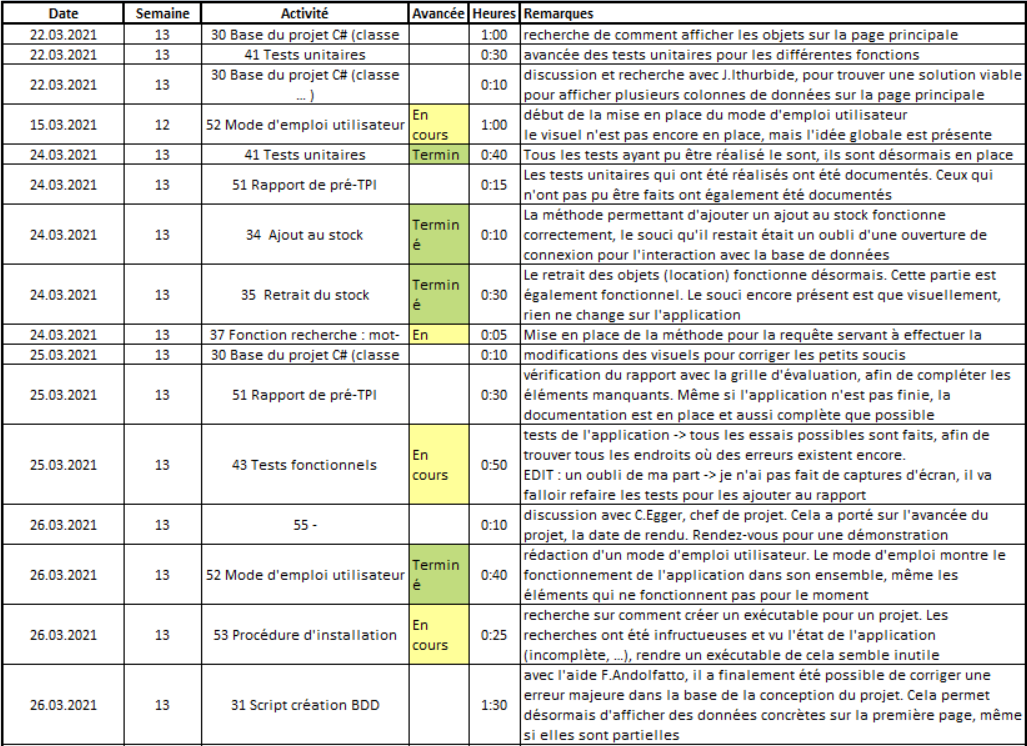
Ceci est le journal de travail de la semaine 11 de 2021. Il s’agit de la semaine n°5 du projet, du 8 mars 2021 au 12 mars 2021.

Ceci est le journal de travail de la semaine 12 de 2021. Il s’agit de la semaine n°6 du projet, du 15 mars 2021 au 19 mars 2021. La raison pour laquelle le journal est aussi court vient du fait qu’une formation à l’extérieur a été suivie du mercredi au samedi de cette semaine. Ainsi, seul le lundi contenait des heures mises à disposition pour avancer sur le TPI.

L’analyse présente de ce fait une erreur au tout début, comme l’intégralité des heures de cette semaine ont été prises en compte pour effectuer la répartition des tâches.



Ceci est le journal de travail de la semaine 13 de 2021. Il s’agit de la semaine n°7 du projet, du 22 mars 2021 au 26 mars 2021. Il s’agit de la dernière semaine « pleine » du projet, le rendu étant prévu le lundi 29 mars 2021.



## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

L’archive du projet se situe dans un dossier archivé .zip. Il contient :

* Le rapport de projet en format .PDF
* Le journal de travail en format .PDF
* L’application en elle-même, qui doit être lancée avec Visual Studio (vu l’avancée du développement, cela ne fait pas de sens de passer à une version exécutable)
* Les deux scripts permettant de créer la base de données
* Le mode d’emploi utilisateur en format .PDF
* Le manuel d’installation en format .PDF

## Table des illustrations

[Figure 1 : modèle conceptuel de données 5](#_Toc67908145)

[Figure 2 : modèle logique de données 6](#_Toc67908146)

[Figure 3 : maquette – vision globale de l'application 7](file:///C:\Users\paola.costa\Documents\GitHub\gestionStocks_preTPI\preTPI_COSTA.docx#_Toc67908147)

[Figure 4 : maquette – formulaire d'ajout d'objet 7](file:///C:\Users\paola.costa\Documents\GitHub\gestionStocks_preTPI\preTPI_COSTA.docx#_Toc67908148)

[Figure 5 : use cases 8](#_Toc67908149)

[Figure 6 : diagramme de classe 12](#_Toc67908150)

[Figure 7 : diagramme de flux – ajouter un objet 13](#_Toc67908151)

[Figure 8 : diagramme de flux – retourner un objet 13](file:///C:\Users\paola.costa\Documents\GitHub\gestionStocks_preTPI\preTPI_COSTA.docx#_Toc67908152)

[Figure 9 : validation des tests de la classe "Objects" 19](#_Toc67908153)