

Sploks

Adam Gruber

Table des matières

1	Analyse préliminaire	3
1.1	Introduction	3
1.2	Objectifs.....	3
1.3	Planification initiale	4
2	Analyse / Conception.....	5
2.1	Concept	5
2.2	Stratégie de test.....	15
2.3	Risques techniques	Erreur ! Signet non défini.
2.4	Planification	Erreur ! Signet non défini.
2.5	Dossier de conception	Erreur ! Signet non défini.
3	Réalisation.....	Erreur ! Signet non défini.
3.1	Dossier de réalisation	Erreur ! Signet non défini.
3.2	Description des tests effectués	Erreur ! Signet non défini.
3.3	Erreurs restantes	Erreur ! Signet non défini.
3.4	Liste des documents fournis	Erreur ! Signet non défini.
4	Conclusions	Erreur ! Signet non défini.
5	Annexes.....	Erreur ! Signet non défini.
5.1	Résumé du rapport du TPI / version succincte de la documentation	Erreur ! Signet non défini.
5.2	Sources – Bibliographie.....	16
5.3	Journal de travail	Erreur ! Signet non défini.
5.4	Manuel d'Installation	Erreur ! Signet non défini.
5.5	Manuel d'Utilisation.....	Erreur ! Signet non défini.
5.6	Archives du projet.....	Erreur ! Signet non défini.

NOTE L'INTENTION DES UTILISATEURS DE CE CANEVAS:

Toutes les parties en italiques sont là pour aider à comprendre ce qu'il faut mettre dans cette partie du document. Elles n'ont donc aucune raison d'être dans le document final.

De plus, en fonction du type de projet, il est tout à fait possible que certains chapitres ou paragraphes n'aient aucun sens. Dans ce cas il est recommandé de les retirer du document pour éviter de l'alourdir inutilement.

1 Analyse préliminaire

1.1 Introduction

Sploks est une application desktop développée en Python avec une base MySQL.

Elle vise à potentiellement remplacer Coliks, une application MSAccess utilisée depuis 17 ans dans le magasin Sports-Time d'Echallens pour gérer la location de matériel de sports d'hiver.

Ce projet est réalisé dans le cadre d'un TPI d'information de 4^{ème} année.
J'ai choisi ce projet pour avoir quelque chose de concret sur mon CV plus tard.

J'ai déjà travaillé sur ce projet durant mon Pré-TPI ce qui implique le fait que la navigation entre les différentes interfaces graphiques a déjà été faite.

1.2 Objectifs

L'objectif de ce TPI est de réaliser uniquement la partie de gestion de stock de Sploks. La gestion des clients, des contrats de location et du personnel ne font pas partie de ce projet.

Gérer le stock de matériel :

- Pouvoir ajouter du matériel neuf au stock de location en introduisant un article à la fois.
- Pouvoir ajouter du matériel neuf au stock de location en introduisant plusieurs articles à partir d'un formulaire de saisie.
- Pouvoir retirer du matériel trop usagé (matériel retiré doit rester visible dans l'historique).
- Pouvoir mettre à jour l'état dans lequel le matériel se trouve, car le prix de location en dépend.
- Pouvoir montrer les revenus générés par chaque article pour pouvoir savoir lesquels sont rentables. Lors de la consultation des détails d'un article la liste des contrats de location dont il a fait l'objet est accessible et la somme encaissée à travers ceux-ci est affichée.
-

Gérer les prix de locations :

- Les durées de location doivent être prédéterminées.
- Doit permettre aux gérants de fixer les prix de location pour n'importe quel cas de figure, de la durée courte de matériel haut-de-gamme à la durée longue de matériel très usagé.

1.3 Planification initiale

Voici toutes les tâches qui m'ont été attribuées.

En rouges celles qui concernent le stock matériel.

En bleu celles qui concernent les prix de locations



Sprint 1 02.05.2022 -> 09.05.2022: Gérer le stock matériel (Lecture)

- Consulter le stock
- Filtrer le stock
- Consulter les détails d'un article

Sprint review 09.05.2022 à 08h05 jusqu'à 09h05.

Sprint 2 09.05.2022 -> 23.05.2022: Gérer le stock matériel (Ajout, Modification)

- Ajouter des articles
- Ajuster le nombre de pièces d'un article multiple
- Mettre à jour l'état du matériel

Sprint review 23.05.2022 à 08h05 jusqu'à 09h05.

Sprint 3 23.05.2022 -> 31.05.2022: Gérer les prix de location (Lecture, Modification)

- Afficher les prix
- Travailler par type
- Modifier les prix de location d'un article
- Modifier les prix de location d'un type d'article

Sprint review 31.05.2022 à 13h30 jusqu'à 14h30.

2 Analyse / Conception

2.1 Concept

Les logiciels qui ont été utilisés :

Visual Studio Code (Version: 1.66.2 user setup) :

Environnement de travail utilisé pour programmer l'application Sploks.

Python (Version : 3.10.2) :

Langage de programmation utilisé pour le projet Sploks.

La librairie utilisé pour l'interface graphique est PyQt5.

La librairie utilisé pour faire la liaison avec la base de donnée est mysql connector

La librairie utilisé pour mettre à part les identifiants de la base de donnée.

Git Bash :

Permet d'effectuer le versioning du projet Sploks, et a aussi permis de mettre à jour le repository sur GitHub (<https://github.com/GruberAdam/TPI-Sploks>).

GitHub est une plateforme de versioning qui permet aussi d'assurer une certaine sécurité en cas de problème avec nos fichiers en local.

Et pour bien structurer mes fonctionnalités avec des branches, j'ai utilisé git-flow.

Suite Office :

Utilisé pour cette documentation, et pour le journal de bord.

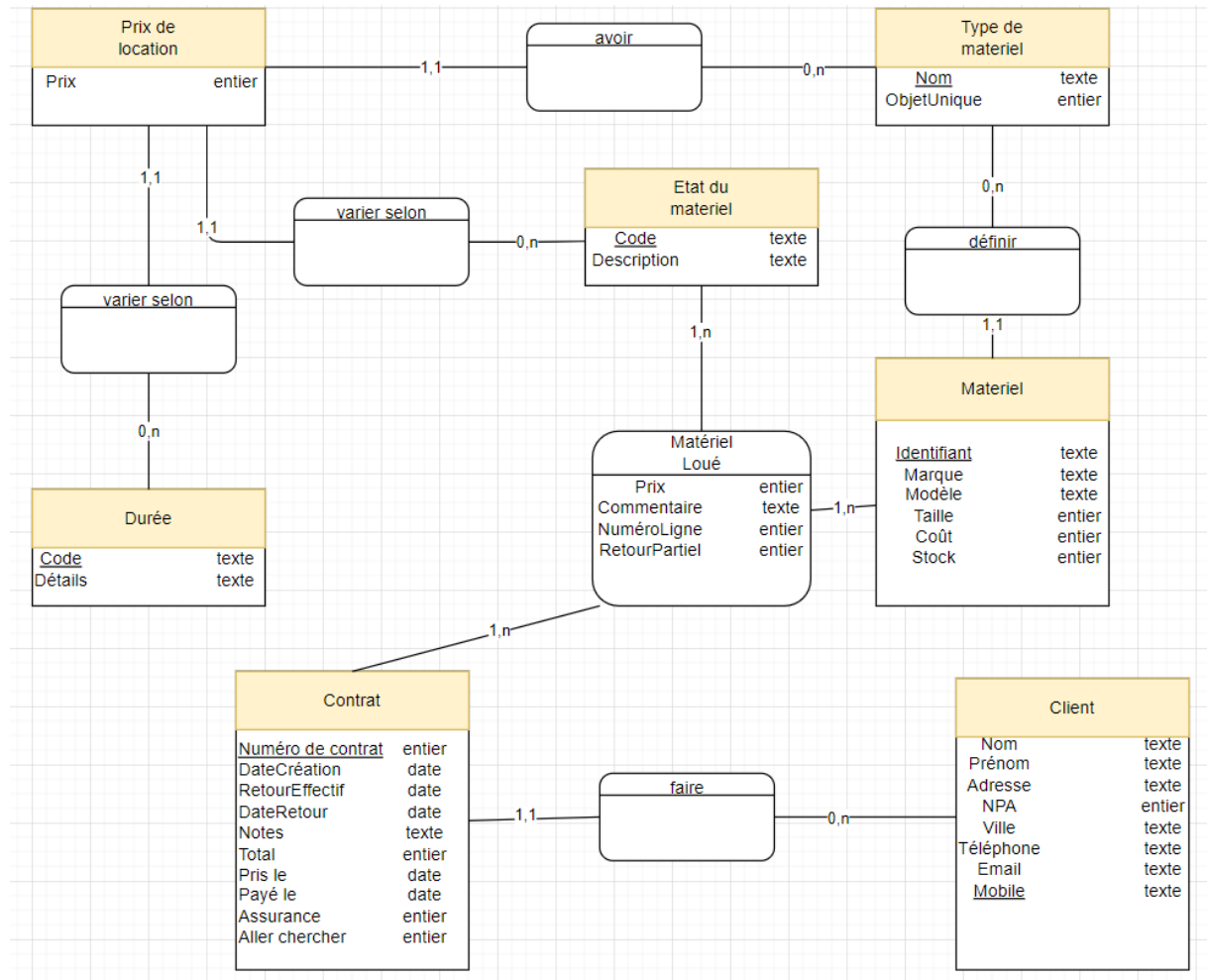
(https://docs.google.com/spreadsheets/d/1SHghMPKg4_q_OUBYOwOJVStz3ETPuOh1Vprxkh7PrmY/edit?usp=sharing)

MySQL Workbench :

Utilisé pour pouvoir consulter la base données Sploks et exécuter des requêtes de test.

Qt Designer :

Utilisé pour pouvoir modifier / créer des interfaces graphiques qui seront directement compatibles avec PyQt5.

MCD

Toutes ces entités auront un impact dans mon application et donc doivent être comprises.

Voici certains points qui ne sont pas forcément évident :

Contrat :

- Retour Effectif :
- Aller chercher :

Matériel Loué :

- Numéro Ligne :
- Retour Partiel :

Sprint 1 :

Rappel du sprint 1

Le sprint 1 consiste à :

- Consulter le stock
- Filtrer le stock
- Consulter les détails d'un article

Fin du sprint 1 le 09.05.2022 à 08h05 (sprint review)

Les interfaces graphiques qui ont été utilisées, ont majoritairement déjà été faites au préalable. Cependant il est possible que quelques modifications ont été effectuées sur certaines d'entre elles.

Analyse

Je compte commencer ce sprint en finissant la consultation du stock parce que je l'avais déjà commencé durant mon pré-tpi et donc la tâche sera donc rapidement terminée.

Ensuite j'effectuerai la consultation des détails d'un article qui n'est rien d'autre que de l'affichage cherché dans la base de données par rapport à un seul article

Pour finir je finirai par le filtre parce que je pense que c'est la tâche la plus du sprint 1 à mon avis car je n'ai jamais fait de filtre.

Conception

Être apte à consulter le stock sera utile lorsque l'utilisateur voudra voir la liste complète de tous les articles.

De plus s'il veut chercher par exemple tous les skis qui sont de la marque Rossignol il lui suffira seulement, d'entrer « Rossignol » dans le filtre de la marque.

L'utilisateur souhaite maintenant avoir les revenus générés d'un article avec le numéro de série « 43212 », il suffit seulement d'effectuer un filtre avec ce numéro de série et ensuite d'ouvrir la vue détaillée de cet article.

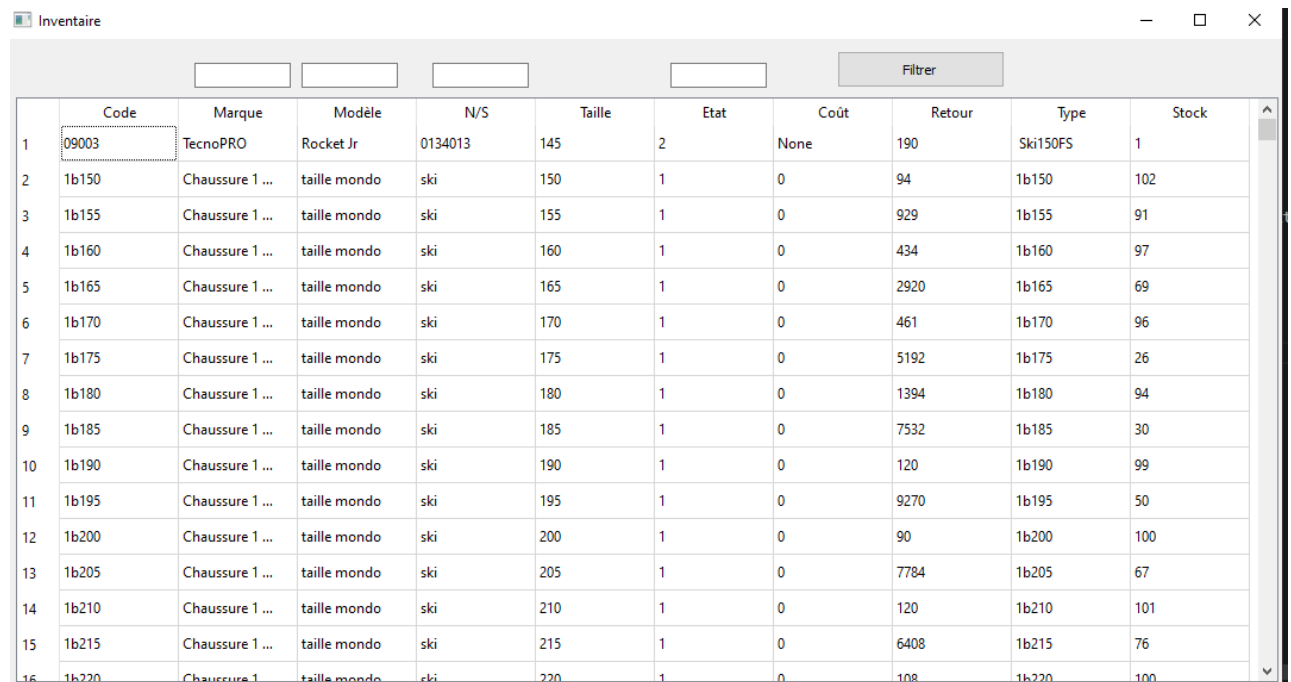
Réalisation du sprint 1

Consulter le stock

En ce qui concerne la consultation du stock, ou en reformulant : Chercher des données en lien avec le stock dans la base de donnée et les placer dans un tableau.

Ce n'était pas très dur étant donné que durant mon pré-tpi je devais effectuer la même manipulation sauf que c'était destiné aux clients.

Voilà à quoi ressemble le stock de tous les objets. Le tableau a déjà été fait, mais j'ai ajouté l'option qui va pouvoir filtrer.



	Code	Marque	Modèle	N/S	Taille	Etat	Coût	Retour	Type	Stock
1	09003	TecnoPRO	Rocket Jr	0134013	145	2	None	190	Ski150FS	1
2	1b150	Chaussure 1 ...	taille mondo	ski	150	1	0	94	1b150	102
3	1b155	Chaussure 1 ...	taille mondo	ski	155	1	0	929	1b155	91
4	1b160	Chaussure 1 ...	taille mondo	ski	160	1	0	434	1b160	97
5	1b165	Chaussure 1 ...	taille mondo	ski	165	1	0	2920	1b165	69
6	1b170	Chaussure 1 ...	taille mondo	ski	170	1	0	461	1b170	96
7	1b175	Chaussure 1 ...	taille mondo	ski	175	1	0	5192	1b175	26
8	1b180	Chaussure 1 ...	taille mondo	ski	180	1	0	1394	1b180	94
9	1b185	Chaussure 1 ...	taille mondo	ski	185	1	0	7532	1b185	30
10	1b190	Chaussure 1 ...	taille mondo	ski	190	1	0	120	1b190	99
11	1b195	Chaussure 1 ...	taille mondo	ski	195	1	0	9270	1b195	50
12	1b200	Chaussure 1 ...	taille mondo	ski	200	1	0	90	1b200	100
13	1b205	Chaussure 1 ...	taille mondo	ski	205	1	0	7784	1b205	67
14	1b210	Chaussure 1 ...	taille mondo	ski	210	1	0	120	1b210	101
15	1b215	Chaussure 1 ...	taille mondo	ski	215	1	0	6408	1b215	76
16	1b220	Chaussure 1 ...	taille mondo	ski	220	1	0	108	1b220	100

Les champs que vous apercevez au-dessus des colonnes « Marque », « Modèle », etc... Sont des champs textuels (**TextEdit**), qui vont donc pouvoir contenir ce que l'on veut filtrer.

Une fois les champs textuels remplis, il y a un bouton Filtrer sur la droite, qui permet d'exécuter le filtrage.

(Le champ « Etat », qui représente l'état des objets, a été gardé numériquement car sur Coliks il est aussi représenté numériquement)

Consulter les détails d'un article

Voilà à quoi ressemble le détail d'un article :



The screenshot shows a window titled 'Détail Article' with a standard macOS-style title bar (minimize, maximize, close buttons). The window contains two columns of text labels next to input fields. The left column includes 'Code d'article:', 'Numéro de série:', 'Prix d'achat:', and 'Revenues générés:'. The right column includes 'Marque:', 'Modèle:', 'Type:', 'Stock:', and 'Taille:'. Below these fields, there is a button labeled 'Liste des Contrats' and a text label 'Nb Contrats: 7'.

Label	Value
Code d'article:	1b150
Numéro de série:	ski
Prix d'achat:	0
Revenues générés:	80
Marque:	Chaussure 1 boude
Modèle:	taille mondo
Type:	1b150
Stock:	102
Taille:	150

Liste des Contrats Nb Contrats: 7

La seule modification qui a été faite sur cette interface graphique est au niveau des champs textuels.

L'ancien champ textuel s'appelait un « **Label** », le problème avec ce champ c'est qu'il n'est pas fait pour être modifié. C'est-à-dire que si on fait une modification sur un champ label, la variable qui contient ce champ ne sera pas mis à jour et gardera l'ancienne valeur.

C'est pourquoi j'ai mis à la place un champ textuel qui s'appelle « **TextEdit** » qui lui est mis à jour lors d'une modification.

Je l'ai fait en avance parce que je sais que dans le futur les utilisateurs devront être à même à modifier ces champs. Plutôt que de le faire plus tard, je l'ai fait tout de suite.

Pour le reste c'était comme **la consultation du stock**, je l'avais déjà fait auparavant donc ce n'étais pas très compliqué.

Il suffisait de mettre un « event Listener » qui permet d'écouter sur quel ligne l'utilisateur clique. Ensuite la fenêtre qui comporte les détail d'un article s'ouvre et affiche les données plus précise.

En ce qui concerne les revenus générés, il suffisait d'effectuer une addition dans la table qui contient tous les articles qui ont été loués.

Filtrer le stock

Pour filtrer le stock c'était une histoire différente, car je n'avais jamais effectué de filtre. J'ai donc décidé de garder les choses simples et de juste ajouter des

champs textuels modifiable au-dessus des colonnes qui doivent être filtrée, comme je vous l'ai montré sur l'image ou je consulte le stock.

Accompagné d'un bouton au côté droite qui permet d'exécuter le filtrage.

Pour finir

Je n'ai quand même pas décidé de mettre la story consulter les détails d'un article en review car il y avait un problème avec le nombre de contrats qui s'affiche.

J'ai n'ai pas eu le temps de régler ce problème, donc j'ai juste décidé de review les 2 première stories et finir celle l'a dans la suivante.

Sprint review

La sprint review c'est passée moins bien que prévu.

- J'ai oublié de mettre les 2 stories en « in-review » dans IceScrum.
- Les identifiants de la base de donnée ne devraient pas être présent sur GitHub.
- Je code sans comprendre ce que je fais (par exemple dans la consultation du stock un colonne « Retour » est présente, pourtant je ne savais pas quel était son utilité.)

La story « consulter le stock » a cependant été validée.

En ce qui concerne la story « Filtrer le stock », elle n'a pas été validée pour différentes raisons

- Lorsqu'on effectue un filtrage la touche entrer n'est pas fonctionnelle
- Lorsqu'on veut passer d'un champ texte à un autre, plutôt que d'appuyer avec notre souris sur le prochain champ texte, la touche tab devrait être fonctionnel.
- Il manque une redirection en haut du tableau lorsqu'on effectue un filtrage.

D'autres points qui ont été ajouté :

- Mettre la colonne « Etat » textuellement et non numériquement.
- Mettre au tâches effectué des tags (Mais le journal de travail était bien exécuté en général)
- Faire tester son application avant de la présenter

En somme, ce sprint n'était pas forcément positif, mais je compte m'améliorer pour le suivant.

Sprint 2

Rappel du sprint 2 selon la planification initiale :

- Ajouter des articles
 - Ajuster le nombre de pièces d'un article multiple
 - Mettre à jour l'état du matériel
- Fin du sprint 2 le 23.05.2022 à 08h05 (sprint review)

Contenu du sprint 2 :

- Filtrer le stock (fonctionnalités à ajouter)
 - Consulter les détails d'un article (à finir)
 - Ajouter des articles
 - Ajuster le nombre de pièces d'un article multiple
 - Mettre à jour l'état du matériel
- Fin du sprint 2 le 23.05.2022 à 08h05 (sprint review)

Analyse

Je compte commencer ce sprint en finissant ce que je devais faire durant le sprint 1. Donc en résumant, finir la consultation des détail d'un article. Ainsi que le Filtrage du stock ou plus précisément la fonctionnalité d'appuyer sur « **entrer** » pour valider le filtrage et la fonctionnalité de pouvoir faire « **tab** » pour passer au prochain champ textuel.

Ensuite, la tâche **ajouter** des articles est un peu à part des 2 autres tâches qui consistent à **modifier** soit, l'état ou le nombre de pièces d'un article multiple.

Je vais commencer par l'**ajout** des articles puis poursuivre avec l'**ajustement** du nombre de pièces d'un article multiple pour finir avec la **mise à jour** de l'état du matériel.

Conception

L'ajout des articles sera utile lorsqu'un manager veut pouvoir ensuite les mettre en location.

Ajuster le nombre de pièces d'un article multiple sera utile lorsqu'un manager veut que le nombre dans l'application corresponde à la réalité du stock.

Mettre à jour l'état du matériel sera utile lorsqu'un manager veut être prêt à démarrer la saison de location

Réalisation du sprint 2

Consultation des détails d'un article :

J'ai fait une requête qui compte le nombre de fois que « item id » se trouve dans la table « rented items », pour afficher le nombre de contrats présents.

Il fallait aussi renommer les fenêtres détaillées par rapport au « code article » que l'article comprenais.

Grace à la visite de Mr. Lymberis, j'ai remarqué que l'état n'était pas visible dans la vue détaillée. Donc j'ai rajouté une « comboBox » qui n'est pas modifiable (juste visuelle). Voilà à quoi ça ressemble :

Article 1b150

Code d'article:	1b150	Marque:	Chaussure 1 boude
Numéro de série:	ski	Modèle:	taille mondo
Etat	Terminé	Type:	1b150
Prix d'achat:	0	Stock:	100
Revenues générés:	80	Taille:	150

Liste des Contrats **Nb Contrats: 7** Editer Valider

Filtrer le stock :

Pour gérer la touche tabulateur, il suffisait seulement de modifier les propriétés des champs textuels dans QtDesigner, ensuite on peut choisir l'ordre dans lequel le tabulateur va être exécuté.

Pour gérer la touche entrer c'était un peu plus dur. Je m'explique, comme je vous l'avais dit j'utilisais comme champ textuel des **TextEdits**. Etant donné que les TextEdits peuvent contenir des retours à la ligne, à chaque fois qu'on appuie sur « entrer » dans un TextEdit, on effectue un retour à la ligne.

J'ai essayé de supprimer le texte lorsque l'utilisateur appuie sur entrer, sauf que le retour à la ligne ne voulait pas être supprimé.

Après avoir effectué plusieurs recherches j'ai trouvé un champ textuel qui s'appelle « **LineEdit** ». Il comporte les mêmes fonctionnalités que le « TextEdit »

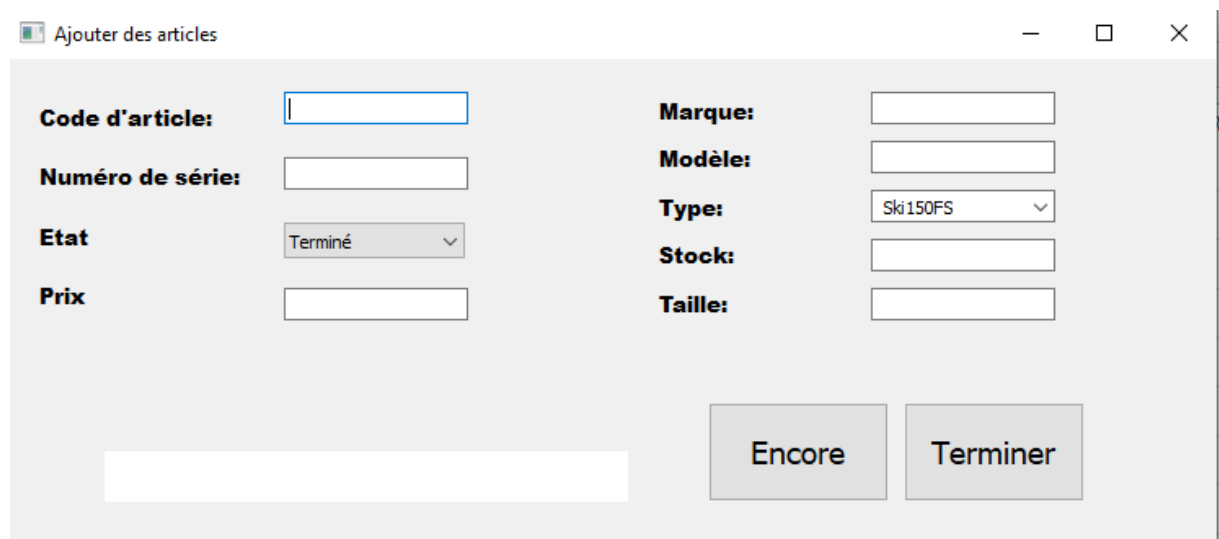
sauf qu'il reste toujours sur une ligne. J'ai donc modifié tous mes filtres avec champ « TextEdit » en « LineEdit » et tout fonctionne.

Ajouter des articles :

Tout d'abord il fallait créer un formulaire d'entrée pour créer des articles. La seule contrainte qui était donnée sur ce formulaire, c'est qu'il y aura 2 radiobox. Une radiobox aura écrit à côté « multiple » et l'autre aura écrit à côté « unique », si l'utilisateur clique sur multiple, on lui demandera combien le nombre d'article qu'il souhaite ajouter, inversement s'il clique sur unique, on lui demandera un numéro de série pour son article.

Durant cette tâche Mr. Carrel a vu comment je programmais et il a trouvé que ma manière de développer n'était pas la plus optimale. Il s'avère que je faisais de la logique métier dans mon contrôleur (Je travaille en MVC). Il m'a donc proposé de refaire tout ce qui se passe au niveau du model, ce que j'ai accepté. C'est pourquoi après cette tâche je vais devoir retravailler mon code et donc devoir consacrer du temps là-dedans.

En continuant sur la tâche d'ajout d'article, voilà à quoi ressemblais mon formulaire : (Je n'avais pas mis les 2 radios box parce que je n'avais pas compris leurs utilité)

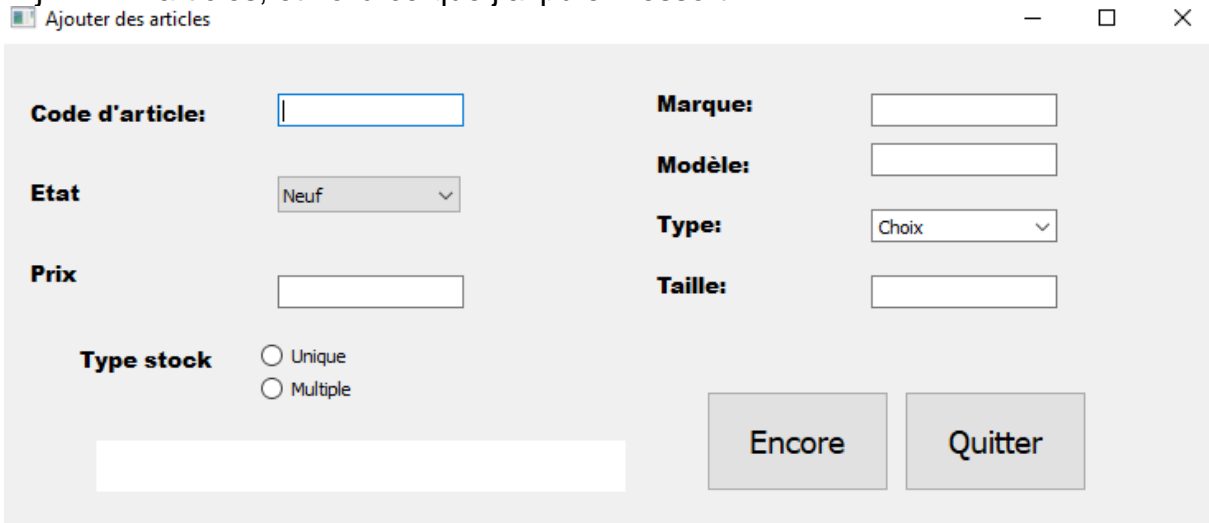


On peut apercevoir plusieurs champs qu'il faut remplir. Le champ type est une liste de choix chargé depuis la base de donnée. Donc tous les types possibles sont affichés.

Les seules contraintes par rapport à la base de donnée sont les champs « Prix », « Taille » et « stock » qui doivent impérativement être des nombres. Si ce n'est pas le cas, un message d'erreur qui indique l'erreur apparaît en bas à gauche du formulaire.

De plus si les valeurs de ces nombres sont trop grandes, la base de donnée n'arrive pas à les ajouter. C'est pourquoi j'ai mis une limite de 7 caractères sur ces 3 champs.

J'ai eu une conversation avec Mr. Carrel (Image en fin de document) concernant l'ajout des articles, et voilà ce que j'ai pu en ressortir :



The screenshot shows a web form titled "Ajouter des articles" with a close button (X) in the top right corner. The form contains the following fields and controls:

- Code d'article:** A text input field.
- Marque:** A text input field.
- Etat:** A dropdown menu with "Neuf" selected.
- Modèle:** A text input field.
- Prix:** A text input field.
- Type:** A dropdown menu with "Choix" selected.
- Taille:** A text input field.
- Type stock:** Two radio buttons labeled "Unique" and "Multiple".
- At the bottom right, there are two buttons: "Encore" and "Quitter".
- At the bottom left, there is a large empty text area.

Donc il m'a bien parlé des champs qui étaient obligatoires (avec une astérisque). J'ai donc pu comprendre l'histoire avec les 2 radio box (comme sur le schéma) contient le nom « Unique » et « Multiple ».

Et donc si l'utilisateur clique sur unique, il doit entrer (s'il y en a un) le numéro de série de l'objet. En échange s'il s'agit de plusieurs articles qu'il ajoute à la fois, il clique sur Multiple et doit remplir le nombre d'article qu'il souhaite ajouter.

Factorisation du code

J'ai perdu certes du temps sur la factorisation du code mais c'est vraiment important de bien savoir séparer son contrôleur et son model.

Avant j'avais juste des fonctions dans mon model qui communiquaient avec la base de donnée, mais Mr Carrel m'a recommandé de faire des classes. C'est pourquoi j'ai refait toute la structure du model avec des classes.

Dans le stock par exemple, j'ai la classe parente « Stock » qui contient différentes méthodes comme « Obtenir tout le stock » et en classe enfant j'ai la classe « Item » qui elle va permettre d'obtenir des informations sur un seul article.

Ajuster le nombre de pièces d'un article multiple

Lors de l'ajustement de l'article je me suis demandé si un article n'a pas le droit d'être ajouté à cause d'un champ n'autorise pas les doublons. J'ai donc pensé que c'est le code article qui serait l'identifiant naturel, étant donné que le numéro de série n'est pas obligatoire dans le formulaire d'ajout d'article.


J'ai quand même décidé d'avoir une conversation avec Mr. Carrel et il m'a confirmé mon choix (et a même dit que le numéro de série sera supprimé.), suite à ce questionnement j'ai fait un MCD, qui peut être trouvé plus haut.

00000

2.2 Stratégie de test

Je fais tester mon application une fois par jour par mon collègue Kaarththigan. Grâce à lui je peux voir des choses qui ne sont pas forcément évidentes pour l'utilisateur.

Par exemple il m'a dit que les filtres n'étaient pas forcément explicites donc j'ai rajouté du texte pour bien me faire comprendre, voilà ci-dessous les précisions ajoutées :

 Inventaire

Filtrer par : <input type="text" value="Marque"/> <input type="text" value="Modèle"/> <input type="text" value="N/S"/> <input type="text" value="Etat"/> <input type="button" value="Filtrer"/>											
ID	Code	Marque	Modèle	N/S	Taille	Etat	Coût	Retour	Type	Stock	
1	09003	TecnoPROx	Rocket Jr	0134013	145	2	0	195	Ski150FS	1	
2	1b150	Chaussure 1 ...	taille mondo	ski	150	1	0	94	1b150	102	
3	1b155	Chaussure 1 ...	taille mondo	ski	155	1	0	929	1b155	91	
4	1b160	Chaussure 1 ...	taille mondo	ski	160	1	0	434	1b160	97	
5	1b165	Chaussure 1 ...	taille mondo	ski	165	1	0	2920	1b165	69	
6	1b170	Chaussure 1 ...	taille mondo	ski	170	1	0	461	1b170	96	

C'est seulement du texte ainsi que des placeholders ajoutés, mais moi aussi je trouve que c'est beaucoup plus simple à comprendre.

Dans mon cahier des charges, il est demandé de tester la robustesse de mon programme, notamment face aux accès concurrents d'au moins quatre postes clients.

Ce test sera effectué la dernière semaine de TPI parce que c'est important de faire ce test lorsque l'application est presque finie et avec la version la plus complète.

2.3 **Sources – Bibliographie**

Les sources utilisées pour les questions concernant python :

<https://stackoverflow.com/>

<https://www.w3schools.com/>

Les sources utilisés pour les questions concernant PyQt5

<https://doc.qt.io/qt-5/>

<https://stackoverflow.com/>

