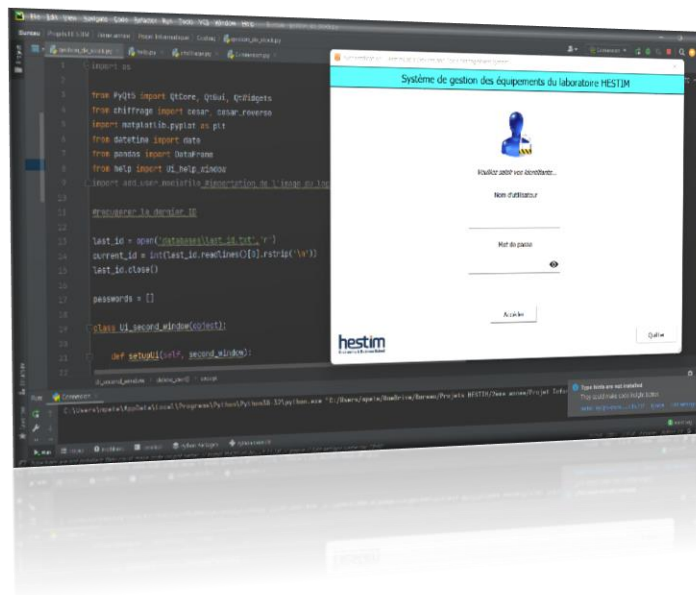


Rapport de projet informatique :
Réalisation d'un logiciel de gestion de stock



Proposé par :

M.SEKHARA Youssef

Réalisé par :

ATOEMNE NYANU louis

MPETEYE Jacintho

PALLI Sylvain

Table des matières

Introduction	3
1-Présentation du thème proposé	4
2-Présentation du programme	4
1.1-Organisation du programme	5
1.1.1-Arbre des onglets du logiciel.....	5
1.1.2-Fonctions	5
1.2-Explication du script.....	14
1.2.1-Architecture du script des fenêtres	14
1.2.2-Gestion des fichiers servant de base de données.....	15
1.2.3-Gestion de la notion d’ID unique	17
1.2.4-Implémentation des barres de recherche	17
1.2.5-Traçabilité à l’aide de l’historique.....	18
1.2.6- Auto-remplissage pour une modification simplifiée.....	18
1.2.7-Gestion du chiffage et du déchiffage des identifiants	18
1.2.8-Enjeu de la fonction update_table().....	20
1.3-Mode d’emploi du logiciel	21
1.3.1-Authentification	21
1.3.2-Onglet «Gestion des équipements »	21
1.3.3- Onglet « Gestion des utilisateurs ».....	22
1.3.4-Onglet « Historique »	23
3-Bilan du travail.....	23
Conclusion & Perspectives	25
Webographie (Ressources didactiques)	26
Table des illustrations	27

Introduction

En guise d'évaluation et afin de jauger nos compétences acquises tout au long de la matière <<*Algorithmique et programmation*>>, il nous a été proposé un projet : l'écriture d'un programme original de gestion d'un stock, en se fiant aux cahier des charges fourni à cet effet.

Gérer un stock n'a jamais été une tâche aisée dans la mesure où on ne dispose pas du bon outil numérique. En effet, plus les quantités s'incrémentent, plus il est difficile de garder une trace de tout. Et quand une entité de ce stock vient à être retirée, on ne se retrouve pas dans la mare des registres physiques contenant ces données, pour notifier cette modification sur l'ensemble du stock. Tel est donc l'intérêt de prendre en charge cette gestion à l'aide des nombreuses technologies du numérique dont nous nous sommes imprégnés lors de ce cours.

Dans ce document, nous présentons avec le plus de détails possibles, le programme doté d'une interface graphique que nous avons édité en réponse aux exigences du projet, que ce soit dans le fond (structure du code) ou dans la forme (interface graphique). Nous avons opté pour un thème original que nous avons concrétisé à l'aide de techniques simples bien que très efficaces.

1-Présentation du thème proposé

Le programme que nous allons présenter dans ce rapport est essentiellement écrit en langage python pour ainsi respecter le cahier des charges. Comme énoncé dans l'introduction, c'est un programme qui est sensé gérer un stock. Dans l'optique de se munir d'un cas concret, nous avons jugé bon de mettre ce calquer ce logiciel sur les besoins de notre école plus précisément, le laboratoire technique HESTIM. En effet, ce dernier se voit accueillir de nombreux équipements par lots dans les différents compartiments, au fur et à mesure qu'il se développe. La dynamique inverse se produit au même titre, car bon nombre de ces équipements sont utilisés ou écartés du stock d'équipement car défectueux ou pour tout autre raison. Ainsi lorsqu'un lot d'équipements est nouvellement acheté ou épuisé, il serait très commode d'avoir un logiciel à disposition afin de gérer toute cette dynamique des stocks d'équipement, ce qui va permettre d'avoir une vue et un total contrôle sur la désignation de l'équipement, la quantité, le prix unitaire, et l'appartenance aux différents départements présents dans le laboratoire. Bien évidemment, ce sera la fonctionnalité principale du programme mais pas la seule. Nous avons su implémenter dans le programme plusieurs fonctionnalités opportunes afin de rendre le travail de gestion simple, transparent et traçable.

2-Présentation du programme

En ce qui concerne l'interface graphique de ce programme, nous avons utilisé le logiciel *Qt designer*, dédié au module *PyQt5* de python en raison de sa flexibilité et sa simplicité en ce qui est de positionner, de dimensionner et de modifier les propriétés des widgets. L'interface du programme a été bâti à l'aide des types de widgets suivants: les *tabWidgets*, les *pushButton*, les *lineEdit*, les *label*, les *listWidgets*, les *tableWidgets*, les *comboBox*, les *spinBox*, les *doubleSpinBox*, et les (pour l'aide). Le logiciel est organisé autour de 3 principaux objectifs à savoir :

- La gestion des équipements (ID, nom, quantité, prix unitaire, département)
- La gestion des utilisateurs (nom d'utilisateur, mot de passe)
- La traçabilité à travers un historique (Date, action, utilisateur à l'origine)

1.1-Organisation du programme

1.1.1-Arbre des onglets du logiciel

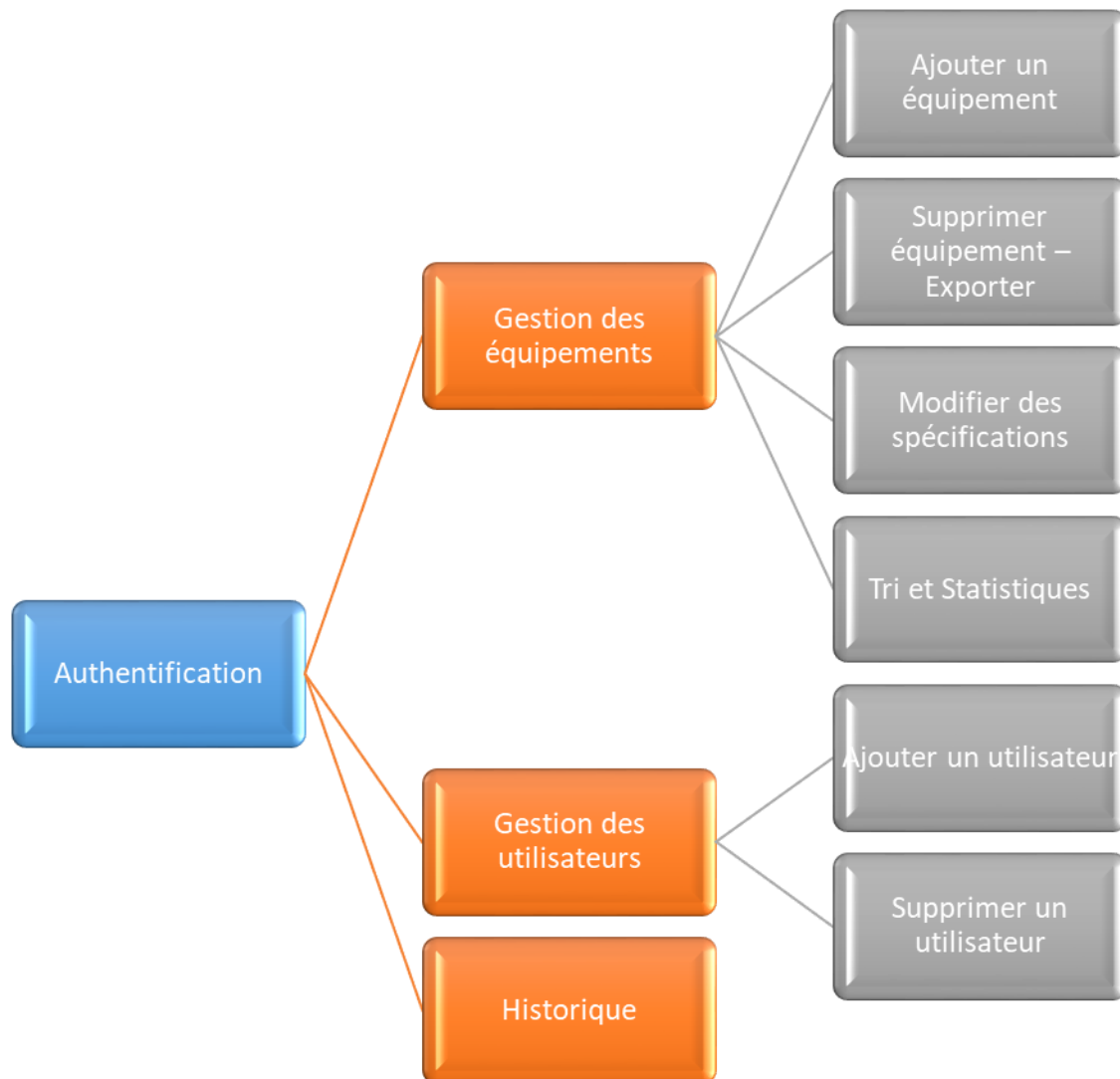


Figure 1 : Arbre montrant la hiérarchisation des onglets du logiciel

1.1.2-Fonctions

Fonctions de la fenêtre d'authentification

blink()

Cette fonction permet de faire clignoter le bouton « Accéder » de la fenêtre.

hide_show_password()

Fonction connectée à l'icône d'œil au bout du champ de saisie du mot de passe permettant de masquer ou d'afficher à volonté le mot de passe que l'utilisateur saisit.

check()

Il s'agit de notre police d'authentification. Par le biais de cette fonction, les identifiants saisis par l'utilisateur sont comparés à ceux contenus dans le fichier texte « identifiants.txt » faisant office de base de données. En cas de conformité le fenêtre principale est ouverte. Dans le cas contraire, un message de refus d'accès est affiché.



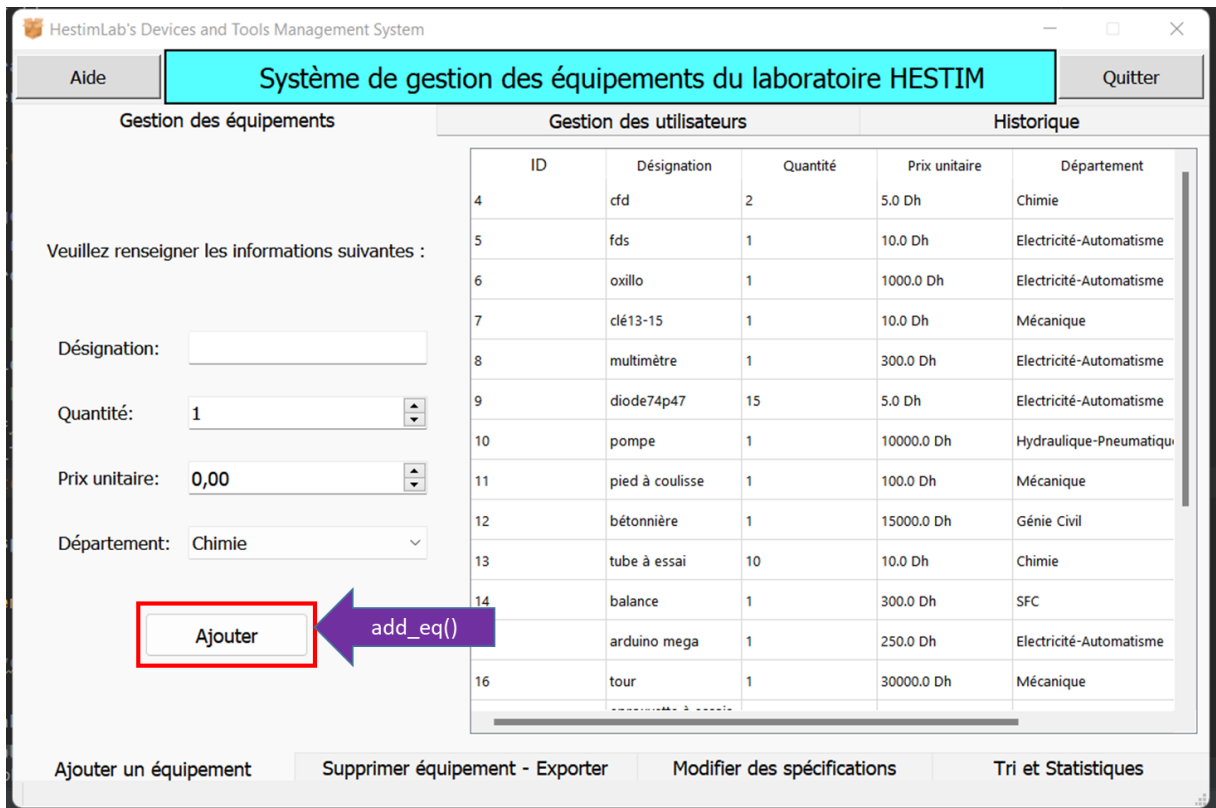
Figure 2 : Fonctions *blink*, *hide_show_password* et *check* de la fenêtre d'authentification

Fonctions de la fenêtre principale

Dans un premier temps, nous présenterons les fonctions qui sont directement connectées à des boutons ou autres éléments de l'interface avec une brève description de leurs rôles.


 **add_eq()**

Cette fonction est connectée au bouton « **Ajouter** » dans l'onglet «Ajouter un équipement » permettant d'ajouter un nouvel équipement par l'utilisateur dans les tables du logiciel, une fois que les informations nécessaires ont été saisies.



ID	Désignation	Quantité	Prix unitaire	Département
4	cfid	2	5.0 Dh	Chimie
5	fds	1	10.0 Dh	Electricité-Automatisme
6	oxillo	1	1000.0 Dh	Electricité-Automatisme
7	clé13-15	1	10.0 Dh	Mécanique
8	multimètre	1	300.0 Dh	Electricité-Automatisme
9	diode74p47	15	5.0 Dh	Electricité-Automatisme
10	pompe	1	10000.0 Dh	Hydraulique-Pneumatique
11	pied à coulisse	1	100.0 Dh	Mécanique
12	bétonnière	1	15000.0 Dh	Génie Civil
13	tube à essai	10	10.0 Dh	Chimie
14	balance	1	300.0 Dh	SFC
	arduino mega	1	250.0 Dh	Electricité-Automatisme
16	tour	1	30000.0 Dh	Mécanique

Figure 3 : Fonction add_eq() – onglet ajout d'équipement

 **search(search_entry_zone, table)**

C'est la fonction liée aux barres de recherche dans les onglet «Supprimer équipement – Exporter » et « Modifier des spécifications» qui opère une recherche dynamique à travers les désignations des équipements stockées dans la table. Elle prend en paramètre la table dans laquelle la recherche est effectuée et la zone de saisie (lineEdit) correspondante.

 **delete_eq()**

Connectée au bouton « **Supprimer** », cette fonction permet de supprimer un équipement de la table une fois que ce dernier y a été sélectionné.

export_table()

Il s'agit de la fonction d'export du contenu des tables d'équipements dans un fichier de tableur excel, soit encore un fichier « .xlsx » ou « .xls », qui sera enregistré dans le répertoire désigné par l'utilisateur.

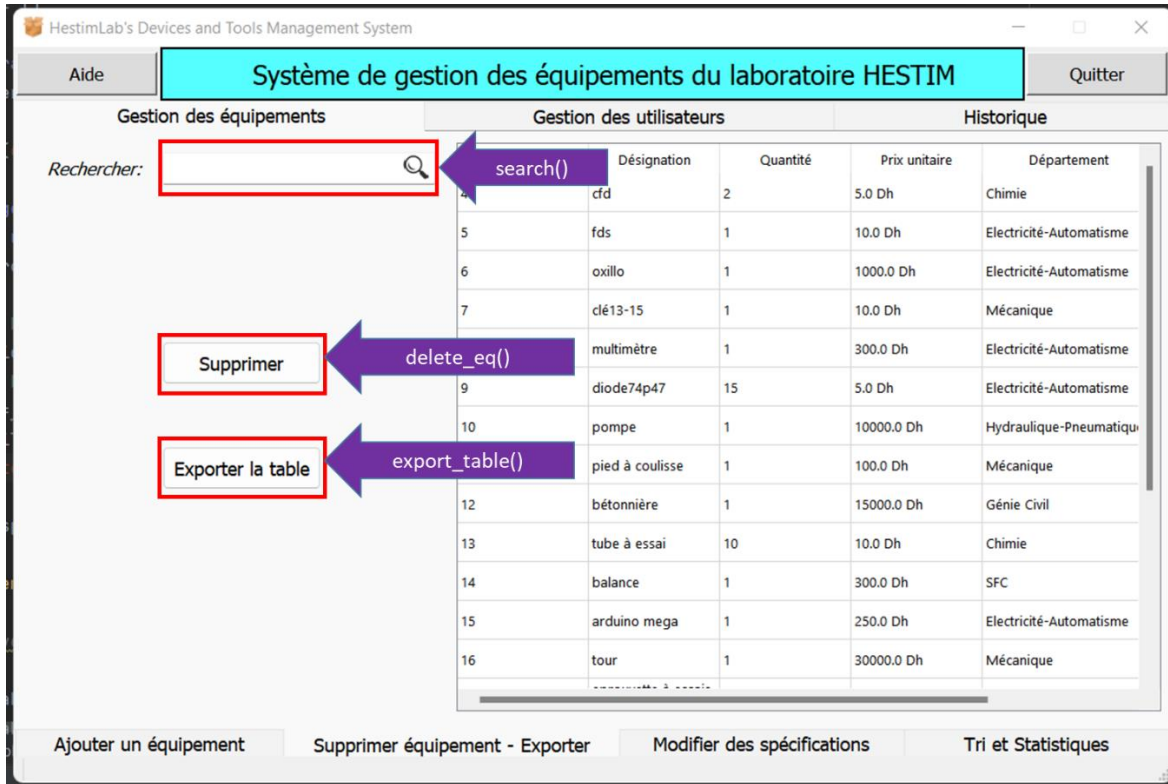


Figure 4 : Fonctions *search*, *delete_eq* et *export_table* de l'onglet de suppression et d'export

modify_eq()

Elle est reliée au bouton « **Modifier** » de l'onglet « Modifier des spécifications » et permet de changer certaines propriétés d'un équipement au besoin, à savoir son nom, sa quantité, son prix unitaire, son département d'appartenance.

fill_area_on_selection()

Fonction intervenant dès qu'un élément de la table est sélectionné et génère le remplissage automatique des champs à renseigner dans l'onglet « Modifier des spécifications ».

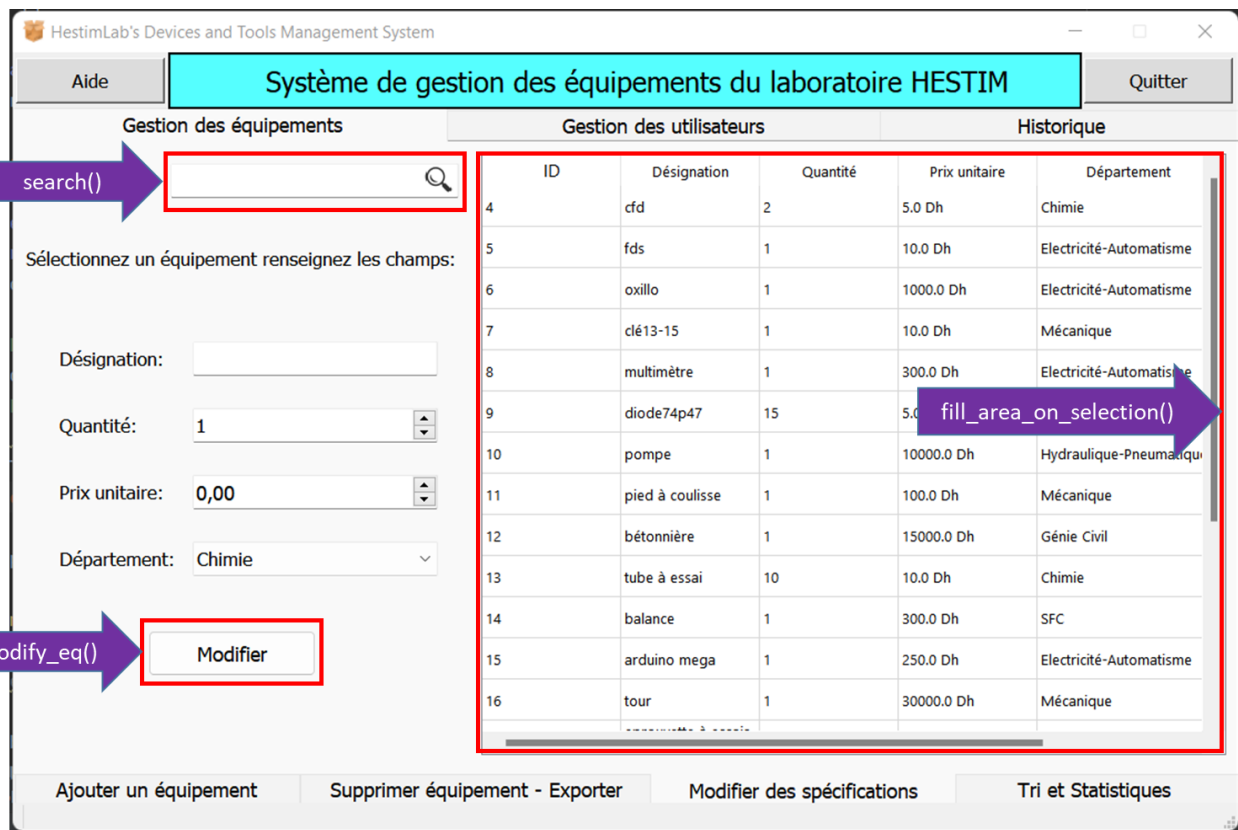


Figure 5 : Fonctions *search*, *modify_eq* et *fill_area_on_selection* dans l'onglet de modification des équipements

sort_eq()

C'est la fonction de tri des équipements selon le choix fait par l'utilisateur dans la liste déroulante. Les éléments triés s'affichent dans la liste de l'onglet «Tri et Statistiques ». Le tri peut se faire par ordre alphabétique ou par département.

draw_graph()

C'est la fonction connectée au bouton « **obtenir des stats** » de l'onglet «Tri et Statistiques » permettant d'obtenir des graphiques à partir du choix de l'utilisateur dans la liste déroulante du type de graphique à afficher. La bibliothèque *Matplotlib* est utilisée à cet effet.

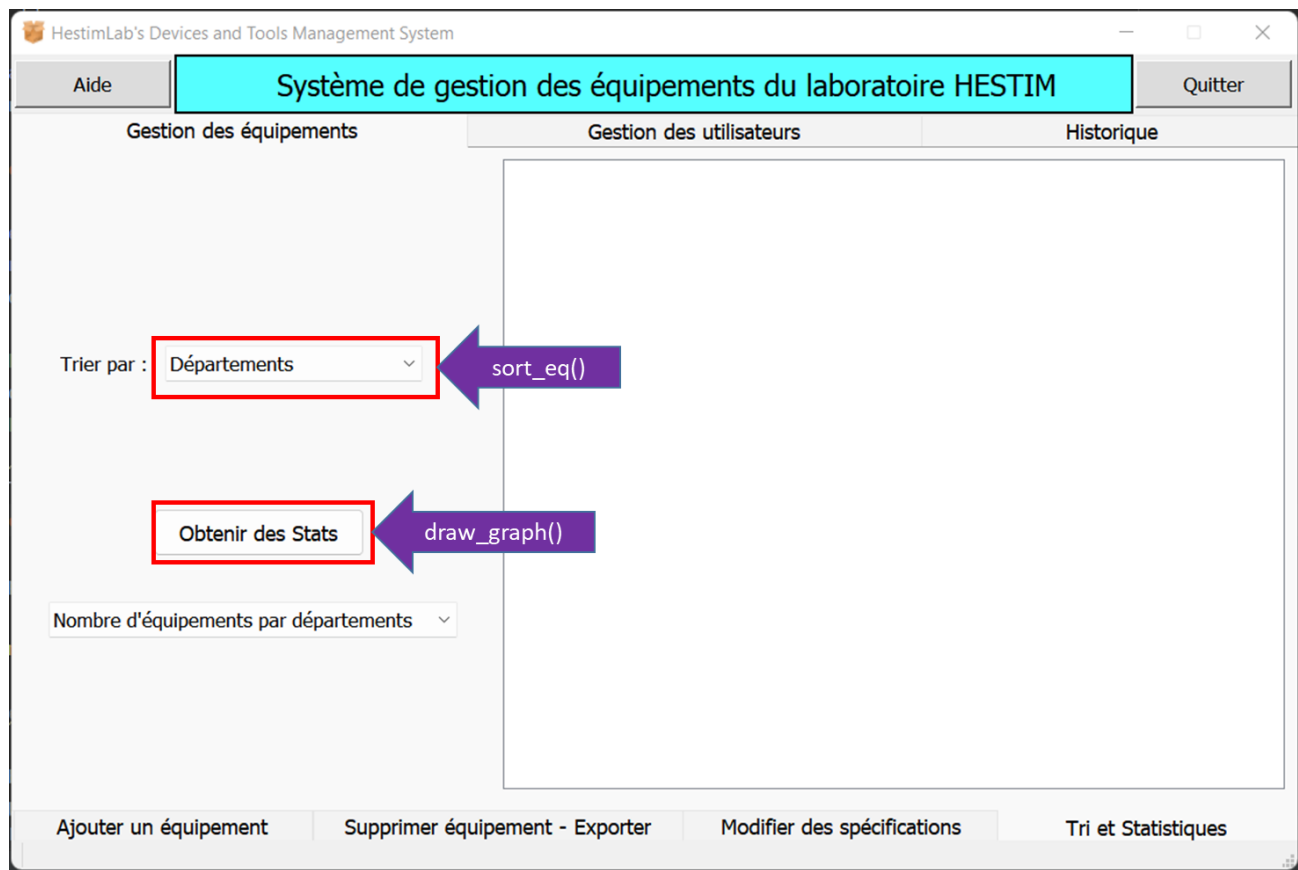


Figure 6 : Fonctions `sort_eq` et `draw_graph` de l'onglet de tri et de statistiques

`add_user()`

Cette fonction est directement connectée au bouton « **Ajouter** » sur l'interface et permet d'ajouter un nouvel utilisateur dans le système du logiciel. Ce dernier pourra dès la prochaine ouverture de session se servir de ses identifiants pour accéder à l'application.

`delete_user()`

Elle est connectée au bouton « **supprimer** » de l'onglet «supprimer un utilisateur » et nous permet de retirer l'accès au logiciel d'un utilisateur et effaçant ses identifiants du système.



Figure 7 : Fonction `add_user` de l'onglet d'ajout d'utilisateur

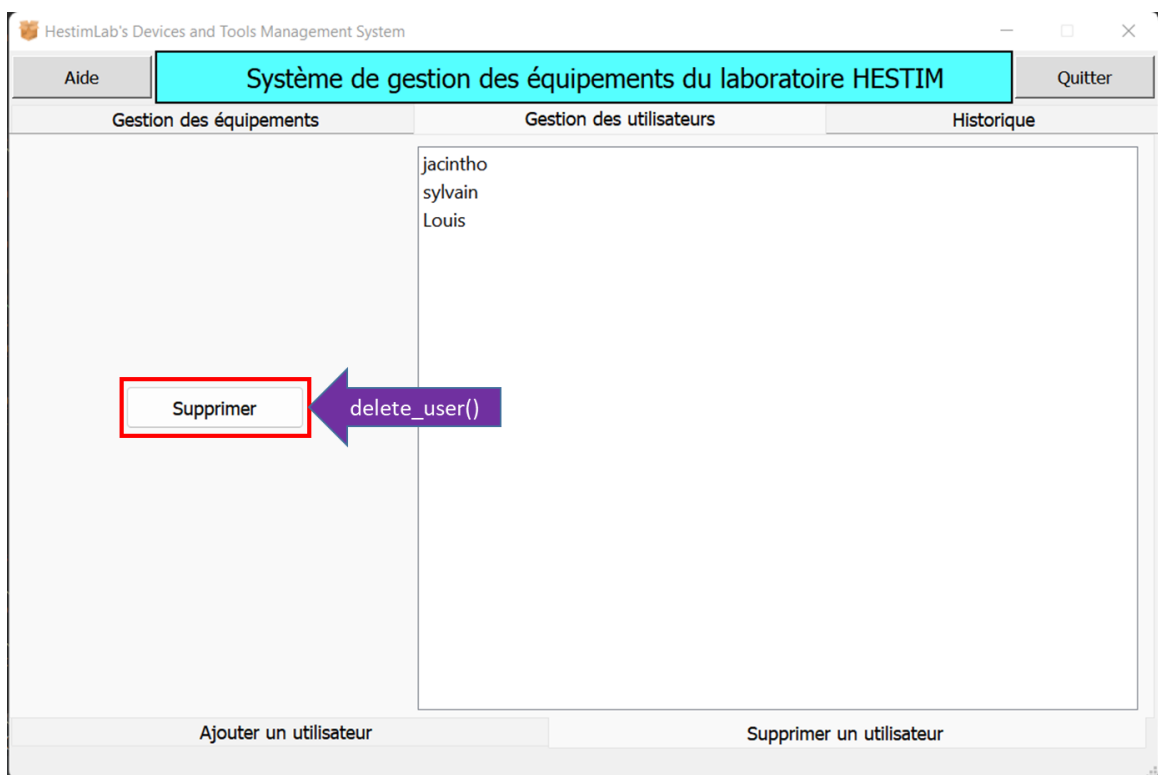


Figure 8 : Fonction `delete_user` de l'onglet de suppression d'utilisateur

delete_history_element()

Elle sous-tend la suppression d'un élément dans l'historique et est connectée au bouton « **effacer un élément** ». Encore une fois, un simple clic sur l'élément suffit.

delete_whole_history()

Connectée au bouton, « **effacer l'historique** » cette fonction permet de supprimer tous les éléments de l'historique.

open_help()

Fonction directement connectée au bouton « **Aide** » dans le coin supérieur gauche de l'interface ouvre une fenêtre renfermant des instructions et des directives concernant le mode d'emploi du produit.

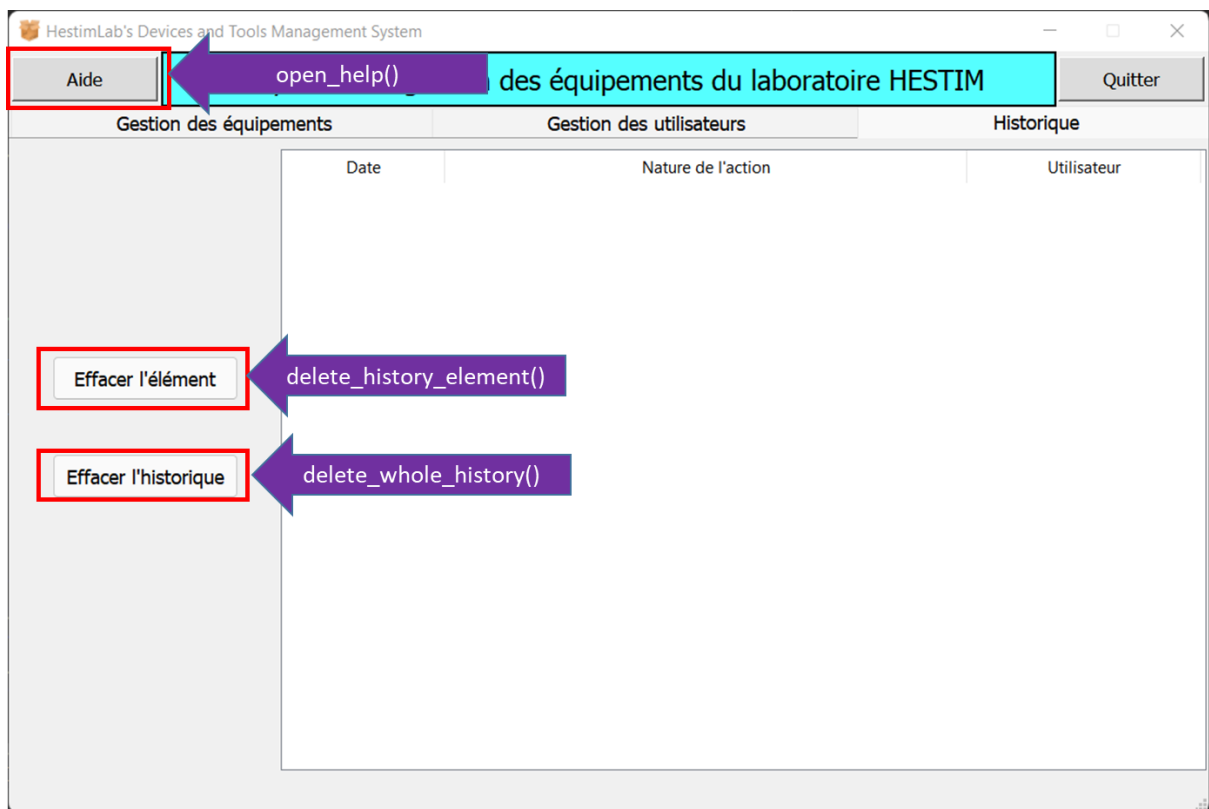


Figure 9 : Fonctions *delete_history_element*, *delete_whole_history*, et *open_help*

Par-delà, d'autres fonctions, qui ne sont pas des fonctions liées à des éléments d'interface, ont été éditées dans le programme. Ces dernières peuvent être appelées ou non à l'intérieur de la définition des fonctions précitées. Ces fonctions sont :

save_data()

Elle permet de stocker la table des équipements (ID, nom, quantité, prix unitaire et le département) dans le fichier texte « equipments.txt » faisant office de base de données.

save_user_data()

C'est la fonction d'enregistrement des données des utilisateurs dans le fichier texte « identifiants.txt » faisant office de base de données.

load_data()

Elle charge dans les tables (tableWidget) les données des équipements (ID, nom, quantité, prix unitaire et le département) stockées le fichier « equipments.txt » faisant office de base de données.

load_user_data()

Elle charge dans la liste des utilisateurs (listWidget) et dans la liste(variable au sein du code) « passwords » les données des utilisateurs préalablement stockées le fichier « identifiants.txt » faisant office de base de données.

save_history_data()

Cette fonction permet d'enregistrer les éléments de l'historique dans le fichier texte « historique.txt » faisant office de base de données.

load_history_data()

Elle charge dans les tables (tableWidget) les éléments de l'historique stockés le fichier « historique.txt » faisant office de base de données

set_history_element (action)

Cette fonction prend en argument une chaîne de caractères qui décrit l'action réalisée par l'utilisateur et ajoute à la table d'historique un élément d'historique comprenant la date, l'action réalisée et le nom de l'utilisateur en cause. Les actions pouvant être répertoriées sont les suivantes :

- ✓ Nouvel équipement <désignation> ajouté
- ✓ Equipement <désignation> supprimé
- ✓ Spécification de l'équipement <désignation> modifié
- ✓ Nouvel utilisateur <nom> ajouté
- ✓ Utilisateur <nom> supprimé

update_table()

C'est une fonction auxiliaire de mise à jour de ligne des tables, utilisée pour réduire le nombre de lignes de codes et éviter la redondance dans l'édition des tables.

cesar(chaîne, clé)

Fonction de chiffage importée depuis le script «chiffage.py », et utilisée pour crypter les données écrites dans le fichier « identifiant.py ». Elle prend en paramètre la chaîne à crypter et la clé de chiffage et renvoie la chaîne cryptée.

cesar_reverse(chaîne, clé)

Fonction de déchiffage, également importée depuis le script «chiffage.py », utilisée pour décrypter les données les du fichier « identifiants.py ». Elle prend en paramètre la chaîne à décrypter et la clé de chiffage et renvoie la chaîne décryptée.

1.2-Explication du script

1.2.1-Architecture du script des fenêtres

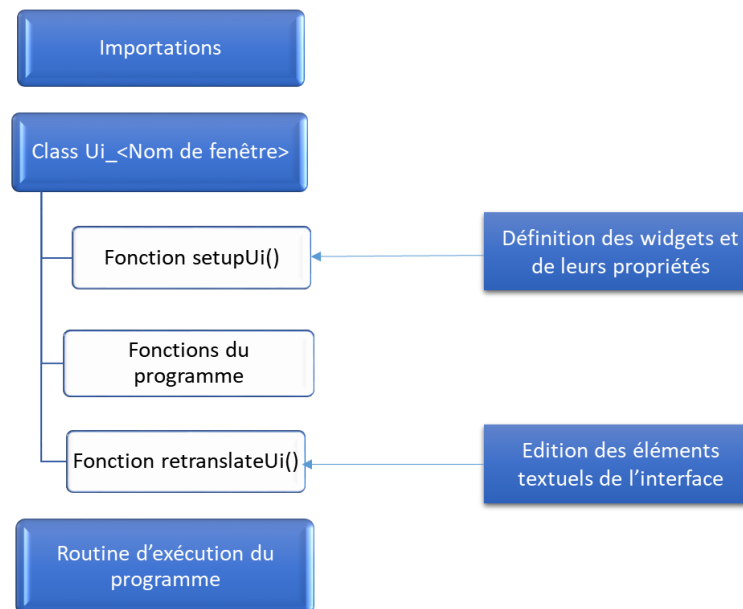
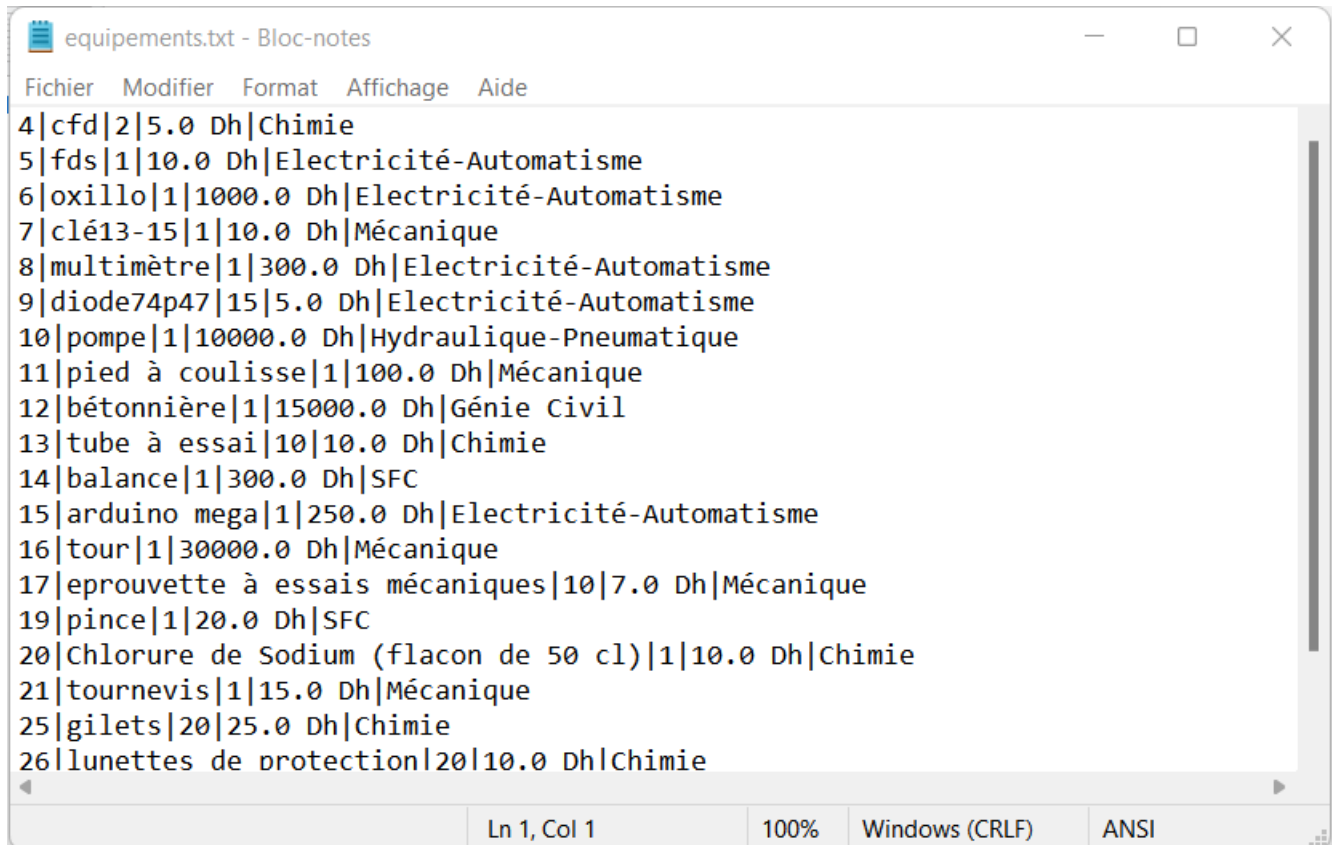


Figure 10 : Architecture du code des fenêtres en PyQt5

1.2.2-Gestion des fichiers servant de base de données

Comme constaté plus haut, parmi les fonctions qui interagissent avec nos pseudo-bases de données on peut distinguer des « **load** » et des « **save** ». Pour se faire une idée de comment les données sont chargées depuis les fichiers dans les tables du programme, intéressons-nous à un exemple :



```
4|cfd|2|5.0 Dh|Chimie
5|fds|1|10.0 Dh|Electricité-Automatisme
6|oxillo|1|1000.0 Dh|Electricité-Automatisme
7|clé13-15|1|10.0 Dh|Mécanique
8|multimètre|1|300.0 Dh|Electricité-Automatisme
9|diode74p47|15|5.0 Dh|Electricité-Automatisme
10|pompe|1|10000.0 Dh|Hydraulique-Pneumatique
11|pied à coulisse|1|100.0 Dh|Mécanique
12|bétonnière|1|15000.0 Dh|Génie Civil
13|tube à essai|10|10.0 Dh|Chimie
14|balance|1|300.0 Dh|SFC
15|arduino mega|1|250.0 Dh|Electricité-Automatisme
16|tour|1|30000.0 Dh|Mécanique
17|eprouvette à essais mécaniques|10|7.0 Dh|Mécanique
19|pince|1|20.0 Dh|SFC
20|Chlorure de Sodium (flacon de 50 cl)|1|10.0 Dh|Chimie
21|tournevis|1|15.0 Dh|Mécanique
25|gilets|20|25.0 Dh|Chimie
26|lunettes de protection|20|10.0 Dh|Chimie
```

Figure 11 : Fichier texte « equipments.txt » ouvert

Comme on peut le voir, chaque ligne du fichier est une ligne de la table des équipements et les éléments de cette ligne sont séparés par le caractère « | ». En ouvrant ce fichier à l'aide de la fonction « open » en mode lecture 'r', il suffit d'un traitement sur chacune des lignes du document, qui ne sont autres que des chaînes de caractères pour extraire les données dont nous avons besoin. Pour plus de compréhension, voici la partie d'extraction éditée dans la fonction **load_data()** :

```

582     def load_data(self):
583
584         #ouverture du fichier texte de stockage
585
586         equipement_saved = open('databases\equipements.txt','r')
587         saved_data = [] #variable de recuperation des données
588
589         #recuperation des données
590         for line in equipement_saved.readlines():
591             line = line.rstrip('\n')
592             saved_data.append(line.split("|"))
593
594         equipement_saved.close()

```

Figure 12 : Fonction load_data partie extraction des données

Par la suite, à l'aide de la méthode *setItem()* des *QTables* de *QtWidgets*, on remplit chaque cellule des tables avec les données extraites.

Pour enregistrer les données, il suffit de faire l'opération inverse en ouvrant cette fois-ci le fichier texte concerné en mode écriture 'w' et d'y ajouter ligne par ligne la concaténation des données converties en chaînes de caractère, le séparateur '|' et le retour à la ligne '\n'.

Les fonctions de sauvegarde sont systématiquement appelées à la fin de la définition de toutes les fonctions précitées susceptibles d'en altérer le contenu. Pour mieux comprendre, ci-après l'exemple de la fonction *delet_eq()* :

```

804     def delete_eq(self):
805
806         try:
807             #Notification dans l'historique
808
809             self.set_history_element("Equipement "+self.table_add_eq.item(self.table_suppress_eq.currentRow(),1).text()+" supprimé")
810
811             #suppression dans la table de l'onglet ajout d'équipements
812             self.table_add_eq.removeRow(self.table_suppress_eq.currentRow())
813             #suppression dans la table de l'onglet modification d'équipements
814             self.table_modify_eq.removeRow(self.table_suppress_eq.currentRow())
815             #suppression dans la table de l'onglet suppression d'équipements
816             self.table_suppress_eq.removeRow(self.table_suppress_eq.currentRow())
817
818             #Retirer l'équipement du fichier texte
819             self.save_data()
820             self.save_history_data()
821

```

Figure 13 : Fonction delete_eq(), lignes de sauvegardes visibles en fin de définition

Par contre les fonctions de chargement des données ne sont appelées qu'une fois en règle générale dans la fonction *setupUi()* du début.

1.2.3-Gestion de la notion d'ID unique

Afin de nous assurer que tous les ID attribués à chaque stock d'équipement seront uniques, nous avons opté pour un système d'identification **auto incrément**. Ces ID ne sont pas réutilisables non plus. Un fichier texte « *last_id.txt* » conserve le dernier ID attribué, ce qui permet même après le redémarrage de la session de reprendre l'incrément des ID comme il se doit.

```
10      #recuperer le dernier ID
11
12      last_id = open('databases\last_id.txt', 'r')
13      current_id = int(last_id.readlines()[0].rstrip('\n'))
14      last_id.close()
15
```

Figure 14 : récupération du dernier ID

1.2.4-Implémentation des barres de recherche

Comme la suppression et la modification de stock d'équipement sont basés sur le clic de l'utilisateur, il est primordiale que ce dernier ait la possibilité de rechercher un équipement par sa désignation. Une barre de recherche est d'autant plus nécessaire car au fil de l'utilisation du logiciel, le nombre de stocks d'équipements ajoutés deviendra considérable, et donc pénible à parcourir par le scroll.

Lorsque l'utilisateur saisie une chaîne de caractère dans la barre de recherche, un parcours à travers la deuxième colonne de la table permet de vérifier si cette chaîne ne se retrouve pas dans l'une des désignations présentes. Si elle ne s'y retrouve pas, le ligne est masquée à l'aide de la méthode *setRowHidden()* des *QTables* de *QtWidgets*. Cette recherche est dynamique car liée directement au signal *textChanged* des *tableWidgets*, en d'autres termes il suffit à l'utilisateur de saisir la chaîne et n'a pas besoin de lancer la recherche à l'aide d'un bouton.

```

925 def search(self, search_entry_zone, table):
926
927     designation_to_search = search_entry_zone.text().lower()
928
929     for row in range(table.rowCount()):
930
931         item = table.item(row, 1)
932
933         table.setRowHidden(row, designation_to_search not in item.text().lower()) #masquer la ligne qui ne comporte pas l'élément recherché
934

```

Figure 15 : Fonction search()

1.2.5-Tracabilité à l'aide de l'historique

A l'aide de la fonction *set_history_element(action)* , une notification est ajoutée à la table d'historique à chaque fois qu'une action est menée par un utilisateur.

Ceci a pour sous-bassement que cette dernière est appelée à l'intérieur de la définition de chacune des fonctions liées aux différentes actions que l'utilisateur pourrait mener. La méthode *today()* de *datetime.date* permet d'obtenir la date du jour à cet effet.

Dans la barre de statue, un petit label permet de voir l'utilisateur connecté à la session. C'est son nom qui apparaîtra pour toutes les actions réalisées lors de cette session.

1.2.6- Auto-remplissage pour une modification simplifiée

Dans l'onglet de modifications des stocks d'équipements, afin de s'assurer que seules les informations modifiées seront altérées dans les tables, nous avons trouvé astucieux de pré-remplir les champs de saisie de ces dernières de sorte à pouvoir récupérer l'intégralité du contenu des champs pour la mise à jour des tables. Pour ce faire la fonction *fill_area_on_selection()*, est connectée à la table de l'onglet de modification. Cette fonction se présente comme suit :

```

812 def fill_area_on_selection(self): #remplir les champs pour la modification
813
814     self.designation_entry_modify.setText(self.table_add_eq.item(self.table_modify_eq.currentRow(),1).text())
815     self.quantite_spinBox_modify.setValue(int(self.table_add_eq.item(self.table_modify_eq.currentRow(),2).text()))
816     self.prix_dSpinBox_modify.setValue(float(self.table_add_eq.item(self.table_modify_eq.currentRow(),3).text().rstrip(' Dh')))
817     self.dep_comboBox_modify.setCurrentIndex(self.dep_comboBox_modify.findText(self.table_add_eq.item(self.table_modify_eq.currentRow(),4).text()))
818

```

Figure 16 : Fonction fill_area_on_selection

1.2.7-Gestion du chiffage et du déchiffage des identifiants

Les identifiants sont aussi des données chargées et sauvegardées tout au long du programme. Cependant une dynamique de chiffrement et de déchiffrement est implémentée dans le code. C'est cet aspect que nous allons toucher dans cette partie.

Le fichier `chiffrement.py` contient la fonction de chiffrement `cesar(chaine, clé)` et la fonction de déchiffrement `cesar_reverse(chaine, clé)`.

```
3  #Chiffrement
4
5  def césar(chaine,k):
6
7      chaine_crypt = ''
8      for car in chaine:
9          if car == ' ':
10             chaine_crypt += ' '
11          else:
12             chaine_crypt += chr((ord(car)+k)%126+32*((ord(car)+k)//126))
13      return chaine_crypt
14
```

Figure 17 : Fonction `cesar(chaine, k)`

```
17 def césar_reverse(chaine,clé):
18
19     chaine_decrypt = ''
20
21     for element in chaine:
22         if element == ' ':
23             chaine_decrypt += ' '
24         else:
25             chaine_decrypt += chr(ord(element)-clé+94*((126-ord(element)+clé)//94))
26
27     return chaine_decrypt
```

Figure 18 : Fonction `cesar_reverse(chaine,clé)`

C'est deux fonctions sont importées du fichier `chiffrement.py`. Le chiffrement est systématiquement fait pour toute informations à écrire dans le fichier « `identifiants.txt` » et le déchiffrement est appelée systématiquement pour toutes les données lu dans le fichier « `identifiants.txt` ». En d'autres termes tout se passe dans le programme comme-ci le chiffrement n'existait pas .

1.2.8-Enjeu de la fonction `update_table()`

Au fil de l'écriture du programme, les tables des onglets d'ajout de suppression et de modification sont actualisées dans plusieurs fonctions. La mise à jour des tables se fait en écrivant le contenu des cellules à l'aide de la méthode `setItem()` des *Qtables* de *QtWidgets*. Dans l'optique d'épargner au code une certaine redondance et afin de réduire les lignes de codes superflues, la fonction `update_table()` a été éditée dans le programme. Ceci a permis de gagner 4 lignes de codes à chaque appellation de cette fonction. Elle prend en paramètre :

_Table : (objet : *QtWidgets.Qtable*) La table dans laquelle la mise à jour va être faite

_Designation : (Type *str*) élément à ajouter dans la colonne des désignations

_Quantite : (Type *str*) élément à ajouter dans la colonne des quantités

_Prix : (Type *str*) élément à ajouter dans la colonne des prix

_Département : (Type *str*) élément à ajouter dans la colonne des départements

_id: (Type *str*) élément à ajouter dans la colonne des id au besoin (déterminé par le paramètre `need_id`)

_need_id : (Type *bool*) variable booléenne permettant de savoir si l'id doit être mis à jour ou pas. Il a la valeur *False* uniquement pour la table de modification.

_row : (Type *int*) index de la ligne à laquelle l'élément (item) doit être mis.

```
742 def update_table(self, table, designation, quantite, prix, departement, id, need_id, row):
743
744     if need_id :
745         table.setItem(row, 0, QtWidgets.QTableWidgetItem(id))
746         table.setItem(row, 1, QtWidgets.QTableWidgetItem(designation))
747         table.setItem(row, 2, QtWidgets.QTableWidgetItem(quantite))
748         table.setItem(row, 3, QtWidgets.QTableWidgetItem(prix))
749         table.setItem(row, 4, QtWidgets.QTableWidgetItem(departement))
750
```

Figure 19 : Fonction `update_table()`

1.3-Mode d'emploi du logiciel

1.3.1-Authentification

Au lancement du logiciel, vous verrez apparaître une fenêtre d'authentification.

Renseignez les champs à l'aide de vos identifiants si vous êtes un utilisateur déjà enregistré dans le système et cliquez sur le bouton « **accéder** ».

Si vous n'êtes pas un utilisateur enregistré dans le système l'accès vous sera refusé. Vous pouvez toutefois, demander à un des utilisateurs enregistrés de se connecter et de vous ajouter comme utilisateur.

Le mot de passe est masqué par défaut. Pour le rendre visible, cliquez sur l'icône d'œil au bout du champ de saisie, et encore une fois pour le masquer.

1.3.2-Onglet «Gestion des équipements »

- **Ajouter un équipement**

Pour ajouter un nouveau stock d'équipements, renseignez les informations demandées sur la fenêtre. Si vous ne connaissez pas le prix unitaire des éléments du stock, pas de panique, vous pouvez laisser la valeur à 0 et venir la modifier une fois que vous aurez connaissance du prix unitaire.

Une fois cela fait, cliquez sur le bouton « **Ajouter** » pour ajouter le nouveau stock d'équipements.

Pour vérifier que votre stock a bien été ajouté, vérifiez l'élément à l'ID le plus grand dans la table de l'onglet, ou alors vous pouvez effectuer une recherche de sa désignation dans les onglets : « Supprimer équipement – Exporter » et « Modifier des spécifications ».

- **Supprimer équipement – Exporter**

Pour supprimer un stock d'équipements, cliquez sur la ligne contenant les informations relatives au stock et cliquez sur le bouton « **Supprimer** ».

N'hésitez pas à saisir la désignation de votre stock dans la barre de recherche pour la faire remonter en haut de la table.

Vérifiez ensuite dans la table si l'élément a bien été supprimé

En cliquant sur le bouton « **Exporter la table** », un répertoire s'ouvrira. Vous devrez alors choisir le dossier dans lequel vous souhaitez enregistrer la feuille de calcul Excel qui sera créée. Cela peut être pratique si vous voulez imprimer le contenu des tables pour un bilan ou quelque tâche que ce soit.

- **Modifier des spécifications**

Pour modifier les informations relatives à un stock donné, cliquez dans la table sur la ligne correspondant à votre stock.

Ces informations rempliront automatiquement les champs à gauche de la table. Ensuite vous n'avez plus qu'à changer celles des informations que vous désirez modifier.

Une fois cela fait, cliquez sur le bouton « **modifier** »

Vérifiez ensuite si les informations ont bien été mises à jour dans la table

- **Tri et statistiques**

La première liste déroulante vous propose les options de tri existantes. Il vous suffit d'en sélectionner une pour voir le résultat apparaître dans le champ à droite.

Pour obtenir des graphiques construits à partir des données des tables, sélectionnez un type de graphique dans la dernière liste déroulante, puis cliquez sur le bouton « **Obtenir des stats** ». Vous verrez apparaître une fenêtre avec un diagramme à bandes. N'hésitez pas à personnaliser le graphique à votre guise à l'aide des boutons de la barre d'outils.

1.3.3- Onglet « Gestion des utilisateurs »

- **Ajouter un utilisateur**

Pour ajouter un nouvel utilisateur, renseigner le nom d'utilisateur ainsi que son mot de passe. Ensuite cliquez sur le bouton « **Ajouter** ». Vous verrez s'ouvrir une notification de confirmation.

N.B : Le nom d'utilisateur n'a que 15 caractères admissibles et le mot de passe 10 caractères admissibles.

- **Supprimer un utilisateur**

Pour supprimer un utilisateur sélectionnez le nom de ce dernier dans la liste des utilisateurs à droite, puis cliquez sur le bouton « **Supprimer** ».

1.3.4-Onglet « Historique »

Pour supprimer un élément de l'historique, cliquez sur la ligne correspondant à ce dernier dans la table puis cliquez sur le bouton « **Effacer l'élément** ».

En cliquant sur le bouton « **Effacer l'historique** », vous viderez tout le registre.

Il est recommandé d'effacer tout l'historique dès que cela est possible, afin d'économiser les ressources du logiciel.

3-Bilan du travail

Dans l'ensemble la conception du logiciel a été un succès, car les objectifs que nous nous sommes fixés au début ont été atteints voire dépassés. La chose n'a toutefois pas été simple à tous les niveaux. Il y a eu plusieurs difficultés rencontrées en route, des problèmes que nous avons su résoudre au fur et à mesure de l'édition du programme.

Le premier de ces problèmes était **l'abondance des lignes de codes dans le fichier d'extension .py obtenu une fois le fichier .ui converti**. Le code obtenu était long et compact. Il va sans dire que son édition était au début assez ardue. Cependant, les noms des widgets ayant été bien définies au départ, il suffisait de réorganiser le code proprement pour en faciliter l'édition. L'IDE *PyCharm* a également été très utile dans la mesure où il rend l'édition et l'organisation du code plus limpide.

Le second problème était **la connexion des fonctions aux signaux des widgets**. Nous avons pu le surmonter en effectuant bon nombre de recherches non seulement sur les forums mais aussi à travers la documentation du module *PyQt5*.

Un autre des défis que nous avons dû relever est le **passage de données à travers les fenêtres**. Notamment l'enregistrement de l'utilisateur de la session dans la fenêtre principale en en prenant la valeur à partir de la fenêtre d'authentification. Une série de tutoriel *YouTube*, que nous citerons en référence, a été très instructive à cet effet et nous a permis de régler le problème.

Un problème majeur et pas des moindres a été **l'ensemble des conflits de classe parent-enfant qui pouvaient survenir**. Bien que la programmation orientée objet soit très simple avec le langage python, certaines de ses subtilités si elles ne sont pas considérées peuvent renvoyer une erreur.

Pour finir, il fallait aussi parcourir le logiciel dans tous ses scénarios possibles, détecter les éventuels bugs et les corriger sans dégrader la structure du code.

Conclusion & Perspectives

Somme toute, travailler sur ce projet a été une expérience très enrichissante pour chacun d'entre nous. Il nous a notamment permis de nous orienter vers de nouvelles technologies, d'acquérir de nouvelles compétences tout en renforçant les précédentes. Ce travail a su mobiliser en nous une grande capacité de réflexion et de résolution dynamique des problèmes de programmation. L'esprit de recherche était également au rendez-vous car comme nous l'avons appris, en ce qui est de la programmation nous ne sommes limités que par notre imagination mais pour le reste il suffit d'être un bon chercheur.

Ceci étant dit les perspectives d'amélioration de ce programme sont nombreuses. Il pourrait déjà s'agir d'utiliser la bibliothèque Sqlite pour gérer les bases de données et rendre le programme encore plus professionnel. Il serait également bien d'éditer un panneau de contrôle de divers paramètres relatifs au logiciel, ou encore d'animer les widgets et les onglets. D'énormes possibilités s'offrent à nous, du moment que nous sommes mues par la curiosité et la passion pour l'édition de programme utiles à la société.

Webographie (Ressources didactiques)

- [1] J. Elder, «PyQt5 GUI Thursdays,» [En ligne].
- [2] J. Jenn, «Add, Copy, Delete selected row on a Table Widget (QTableWidget) | PyQt5 Tutorial,» [En ligne].
- [3] J. Jenn, «How To Launch Matplotlib Chart With A Button In PyQt5,» [En ligne].
- [4] A. Academy, «Bar Graph | Bar Chart | Matplotlib | Python Tutorials,» [En ligne].
- [5] «Tutorials Point,» [En ligne]. Available:
https://www.tutorialspoint.com/pyqt/pyqt_qlineedit_widget.htm.
- [6] «Qt Documentation,» [En ligne]. Available: <https://doc.qt.io/qt-5/>.
- [7] [En ligne]. Available: <https://stackoverflow.com/>.

Table des illustrations

Figure 1 : Arbre montrant la hiérarchisation des onglets du logiciel	5
Figure 2 : Fonctions blink, hide_show_password et check de la fenêtre d'authentification	6
Figure 3 : Fonction add_eq() – onglet ajout d'équipement	7
Figure 4 : Fonctions search, delete_eq et export_table de l'onglet de suppression et d'export	8
Figure 5 : Fonctions search, modify_eq et fill_area_on_selection dans l'onglet de modification des équipements.....	9
Figure 6 : Fonctions sort_eq et draw_graph de l'onglet de tri et de statistiques	10
Figure 7 : Fonction add_user de l'onglet d'ajout d'utilisateur	11
Figure 8 : Fonction delete_user de l'onglet de suppression d'utilisateur	11
Figure 9 : Fonctions delete_history_element, delete_whole_history, et open_help.....	12
Figure 10 : Architecture du code des fenêtres en PyQt5	14
Figure 11 : Fichier texte « equipments.txt » ouvert	15
Figure 12 : Fonction load_data partie extraction des données.....	16
Figure 13 : Fonction delete_eq(), lignes de sauvegardes visibles en fin de définition.....	16
Figure 14 : récupération du dernier ID.....	17
Figure 15 : Fonction search()	18
Figure 16 : Fonction fill_area_on_selection	18
Figure 17 : Fonction cesar(chaine, k).....	19
Figure 18 : Fonction cesar_reverse(chaine, clé)	19
Figure 19 : Fonction update_table().....	20