***Remise 2***

Équipe no 2

GAUDREAU Louis  
CHAAR Jacob  
BERNIER Alexandre

*Bases de données avancées*

*GLO 4035*

Travail présenté à

*Jean-Thomas Baillargeon*

Département d’informatique et de génie logiciel

Université Laval 2018

**La description du problème et des données**

Pour ce travail, nous avons choisi de développer le projet d’inventaire de savons. Notre mandat fut de créer une application ainsi qu’une base de donnée permettant de présenter l'état de l’inventaire d’une compagnie vendant des savons. Ce document permet de présenter des aspects techniques de ce travail.

Lorsque nous avons reçu les données, nous avons remarqué que l’on peut créer trois collection avec celles-ci. Nous avons donc distribué les données selon leurs collections respectives en les comparant grâce à des schémas de données. De plus, nous avons formater les données afin qu’elles respectent leurs types de donnée respective (ex : les totaux en float). Ces changements sont effectués dans le fichier utilities.py.

Nous avons ensuite créé une API permettant de retourner une interface html à l’utilisateur afin d’utiliser les différentes requêtes créées. Une section expliquant les différentes routes et ce qu’elles utilisent se retrouvent plus loin dans le rapport. Toutes les routes créées sont dans le fichier app.py.

**Pour faire fonctionner l’application**

docker pull grizzlys54/glo4035a18-projet2018-eq2

docker run --name testtp -p 80:80 grizzlys54/glo4035a18-projet2018-eq2

Note : Ce travail a été développé et testé pour une utilisation sur Google Chrome. Pour le bon fonctionnement de cette application, les « cookie » doivent être activé et aucun bloqueur de publicité ne doit fonctionner lors de l’utilisation de l’application.

**Collections**

* Purchases : contient les achats
* Densities : contient les densités des items
* Labors : contient les utilisations de matière première des items

**Requêtes**

Pour chaque aspect de ce travail, nous avons préféré être large avec le rendu des requêtes MongoDb afin d’exploiter la puissance du langage JavaScript et ainsi spécifier les requêtes avec celui-ci. Cela fait aussi en sorte que beaucoup de requêtes sont réutilisable. Toutes ces requêtes sont effectuées à travers les routes de l’API dans app.py.

Note : ces requêtes ont été faite en python. Elles sont traduites en mongodb et peut contenir des erreurs de traductions.

1. Purchases .find({})
2. Densities .find({})
3. Labors .find({})
4. Purchases .remove({})
5. Densities.remove ({})
6. Labors .remove ({})
7. Purchases .insert({"date": date d’achat, "item": item, "qte": quantité utilisée, “unit": unité utilisé, "total": total de l’acaht, "stotal": sous-total, "tax": taxe})
8. Densities.insert({"item": item, "ml": millilitre, "g": gramme})
9. Labors.insert({"date": date, "item": item, "qte": quantité, “unit": unité, “type": type de matière, "jobId": identifiant du «labor»})
10. Purchases.update({"\_id": ObjectId(id)} , {"date": date, "item": item, "qte": quantité utilisée, “unit": unité, "total": total, "stotal": sous-total, "tax": taxe})
11. Densities.update ({"\_id": ObjectId(id)},{"item": item, "ml": millilitrer, "g": gramme})
12. Labors.update ({"\_id": ObjectId(id)},{"date": date, "item": item, "qte": quantité utilisé, “unit": unite utilisé, “type": type de matière, "jobId": identifant du «labor»})
13. Purchases.find({"$and":[{"date": date voulue}, {"item" : {"$regex": “/type de matière voulue /”}}]})
14. Labors.find({"$and": [{"date": {"$lte":reqData["date"]}}, {"type" : "usage"}]})
15. Purchases.find({"date": {"$lte": reqData["date"]}})

**Index utilisés**

* Purchases :
  + Date : Les recherches de coûts totaux, de coût d’acquisition et de coût moyen se font par la date. Il est donc utile de mettre un index pour ce champ.
* Labors :
  + Date : Les recherches de coûts totaux, de coût d’acquisition et de coût moyen se font par la date. Il est donc utile de mettre un index pour ce champ.
* Density :
  + Item : nous l’avons mis unique afin qu’il n’y ait qu’une densité par item. Nous le vérifions aussi lors de l’ajout par javascript

**Sécurité**

Mot de passe administrateur pour la suppression de toutes les données : admin123

**API**

Toutes les routes créées sont dans le fichier app.py.

1. Route : request Get \--url
   * ‘’adresse ip application’/
   * Paramètres : aucun
   * Requête utilisé : aucune
   * Retourne l’accueil de l’application.
2. Route : request GET \--url
   * ‘’adresse ip application’/transactions’
   * Paramètres : aucun
   * Requête utilisé : aucune
   * Retourne l’interface pour afficher les données de toutes les collections.
3. Route : request POST \--url
   * ‘’adresse ip application’/transactions’
   * Paramètres : {donnée csv}
   * Requêtes utilisé : 7,8,9
   * Vérifie les données et les insères dans la base de donnée selon les données établies. Retourne une erreur 400.
4. Route : request DELETE \--url
   * ‘’adresse ip application’’/transactions’
   * Paramètres : {mot de passe administrateur}
   * Requêtes utilisé : 4,5,6
   * Supprime l’entièreté de la base de données si le mot de passe est correct, sinon retourne une erreur 401.
5. Route : request Get \--url
   * ‘’adresse ip application’/transactions/purchases
   * Paramètres : aucun
   * Requête utilisé : 1
   * Retourne le contenu de la collection purchases.
6. Route : request Get \--url
   * ‘’adresse ip application’/transactions/densities
   * Paramètres : aucun
   * Requêtes utilisé : 2
   * Retourne le contenu de la collection densities.
7. Route : request Get \--url
   * ‘’adresse ip application’/transactions/labors
   * Paramètres : aucun
   * Requête utilisé : 3
   * Retourne le contenu de la collection labors.
8. Route : request Get \--url
   * ‘’adresse ip application’/transactions/add
   * Paramètres : aucun
   * Requête utilisé : aucune
   * Retourne l’interface pour ajouter un item dans une collection.
9. Route : request Post \--url
   * ‘’adresse ip application’/transactions/add/purchase
   * Paramètres: {"date": date, "item": item, "qte": quantité utilisé, “unit": unité, "total": total, "stotal": sous-total, "tax": taxe}
   * Requête utilisé : 7
   * Envoit un item dans la collection purchases. Retourne un Json de succès avec un status 200 ou 400.
10. Route : request Post \--url
    * ‘’adresse ip application’/transactions/add/density
    * Paramètres: {"item": item, "ml": millilitre utilisé, "g": gramme utilisé}
    * Requête utilisé : 8
    * Envoit un item dans la collection density. Retourne un Json de succès avec un status 200 ou 400.
11. Route : request Post \--url
    * ‘’adresse ip application’/transactions/add/labor
    * Paramètres: {"date": date, "item": item, "qte": quantitié utilisé, “unit": unité utilisé, “type": type de matière, "jobId": identifiant du «labor»}
    * Requête utilisé : 9
    * Envoit un item dans la collection labors. Retourne un Json de succès avec un status 200 ou 400.
12. Route : request Get \--url
    * ‘’adresse ip application’/transactions/modifyPurchase
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface de modification des achats.
13. Route : request Post \--url
    * ‘’adresse ip application’/transactions/modifyPurchase
    * Paramètres : {“id”: id, "date": date, "item": item, "qte": quantité utilisé, “unit": unité utilisé, "total": total de l’achat, "stotal": sous-total de l’achat, "tax": taxe}
    * Requête utilisé : 10
    * Modifie un item dans la collection purchases. Retourne un Json de succès avec un status 200 ou 400.
14. Route : request Get \--url
    * ‘’adresse ip application’/transactions/modifyDensity
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface de modification des densités.
15. Route : request Post \--url
    * ‘’adresse ip application’/transactions/modifyDensity
    * Paramètres: {“id”:id, "item": item, "ml": millilitre utilisé, "g": gramme utilisé}
    * Requête utilisé : 11
    * Modifie un item dans la collection densitiesDb. Retourne un Json de succès avec un status 200 ou 400.
16. Route : request Get \--url
    * ‘’adresse ip application’/transactions/modifyLabors
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface de modification des travaux effectué sur les savons.
17. Route : request Post \--url
    * ‘’adresse ip application’/transactions/modifyLabors
    * Paramètres: {“id”: id, "date": date, "item": item, "qte": quantité utilisé, “unit": unité utilisé, “type": type de matière, "jobId": identidant du «labor»
    * Requête utilisé : 12
    * Modifie un item dans la collection densities. Retourne un Json de succès avec un status 200 ou 400.
18. Route : request Get \--url
    * ‘’adresse ip application’/totalCost/
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface d’affichage pour le coût total à une date précise pour une catégorie de matériel.
19. Route : request Post \--url
    * ‘’adresse ip application’/totalCost/
    * Paramètres: {"date":date, "type": type de matière}
    * Requête utilisé : 13
    * Retourne le coût total à une date précise pour une catégorie de matériel.
20. Route : request Get \--url
    * ‘’adresse ip application’/averageCost/
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface d’affichage pour le coût moyen à une date précise pour une catégorie de matériel
21. Route : request Post \--url
    * ‘’adresse ip application’/averageCost/
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface d’affichage pour le coût moyen à une date précise pour une catégorie de matériel.
22. Route : request Get \--url
    * ‘’adresse ip application’/leftQuantity/
    * Paramètres : aucun
    * Requête utilisé : aucune
    * Retourne l’interface d’affichage pour les quantités restante d’un item à une date précise pour une catégorie de matériel.
23. Route : request Post \--url
    * ‘’adresse ip application’/leftQuantity /
    * Paramètres: {"date":date}
    * Requête utilisé : 14
    * Retourne les «labors» de type «usage» à la date demandé
24. Route : request Post \--url
    * ‘’adresse ip application’/leftQuantity /purchase
    * Paramètres: {"date":date}
    * Requête utilisé : 15
    * Retourne les achats de la date demandé.

Interfaces

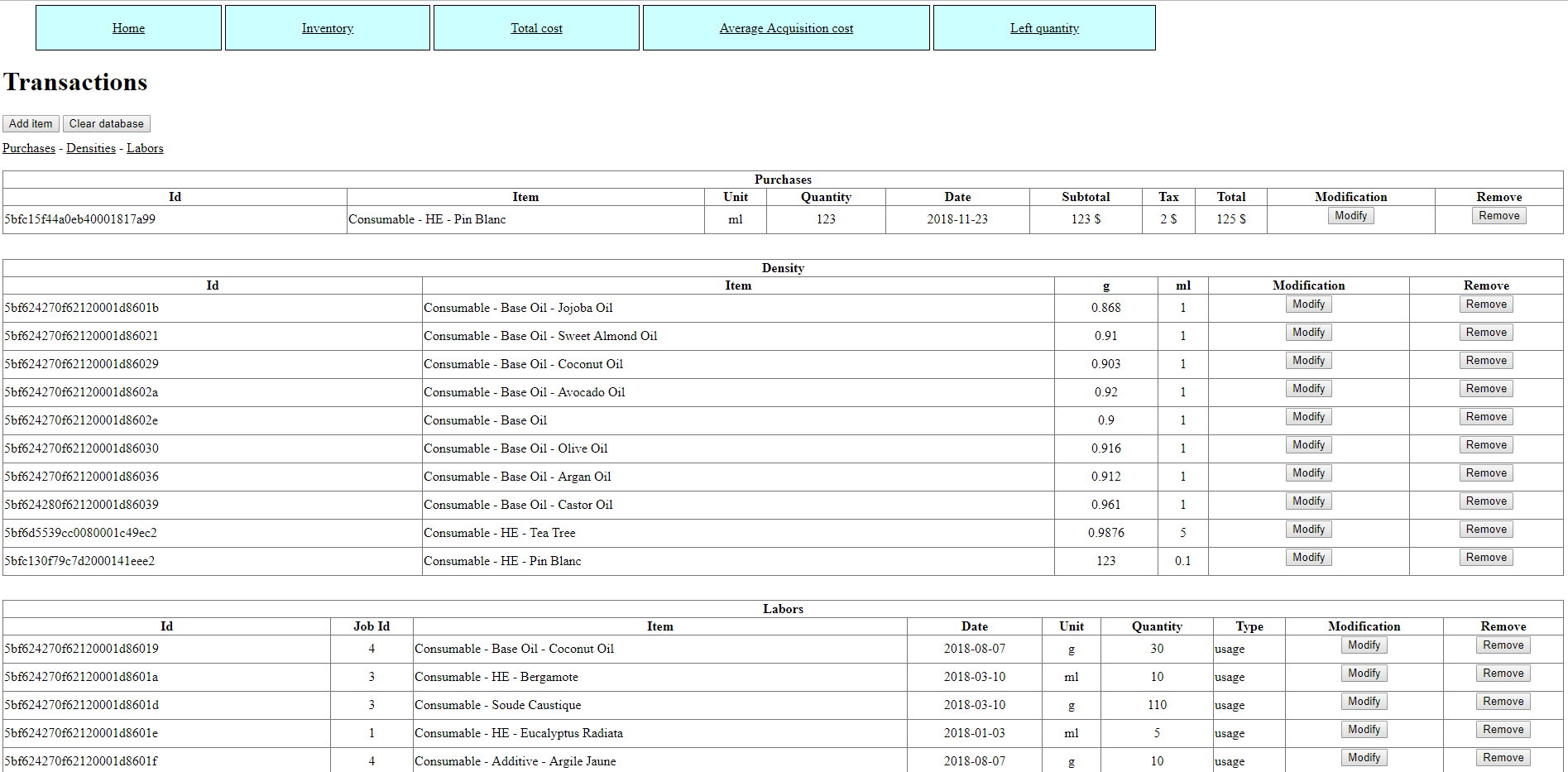
Chaque interface de l’application fut créée en HTML5 et JavaScript selon la norme ES6.

**Accueil**



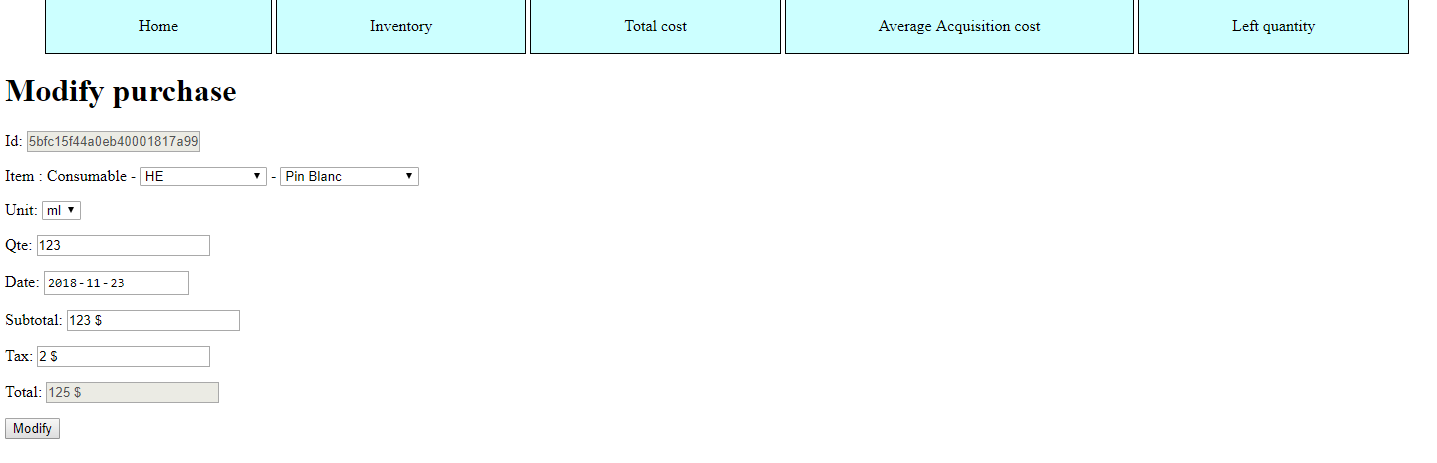
L’accueil de l’application. Permet de naviguer vers les autres pages.

**Listes des transactions (Inventory)**

****

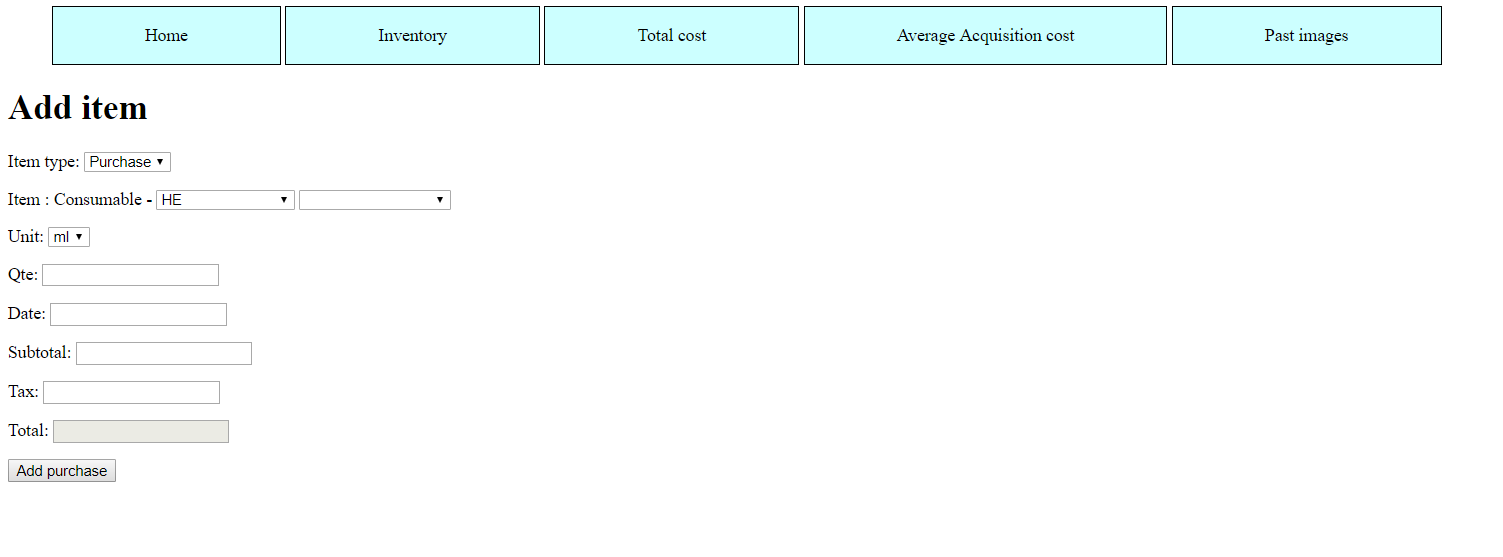
Permet de lister les données de toutes les collections. En cliquant sur «Add item», l’utilisateur sera transféré vers la page d’ajout d’un item. En cliquant sur «Clear database», l’utilisateur sera invité à entrer un mot de passe permettant de supprimer les données de toutes les collections. En cliquant sur «Modify», l’utilisateur sera transféré vers la page de modification de l’item associé au bouton. En cliquant sur «Remove», l’utilisateur peut supprimer l’item voulu après confirmation. Les «purchases» et les «labors» sont classés par date d’ajout et les «densities» par nom d’items.

**Modification d’un item**



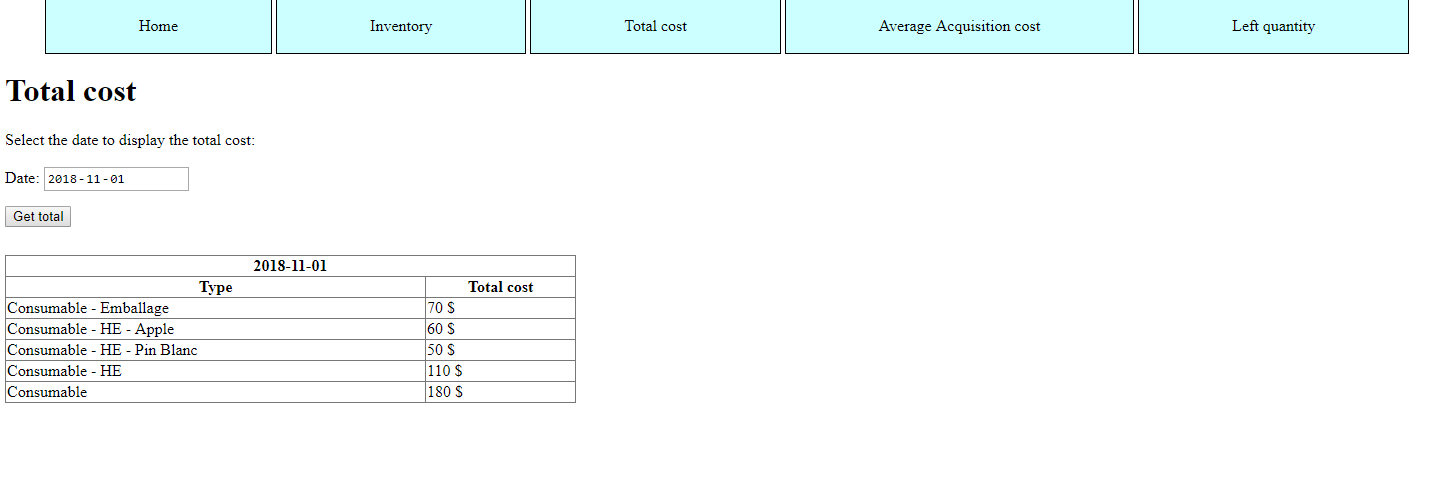
Cette interface permet de modifier un item dans la collection choisi dans l’interface de la liste des transactions. Les données de l’item seront automatiquement affichées. Après avoir modifié les données, l’utilisateur clique alors sur le bouton «Modify» et les données s’envoit au serveur. Une alerte informera ensuite à l’utilisateur si la modification a bien été fait.

**Ajout d’un item**



Cette interface permet d’ajouter un item dans n’importe lequel des collections (purchase, labor ou density). Simplement choisir le type d’item voulue et les informations vont s’ajuster selon le type. Lorsque l’utilisateur clique sur le bouton «Add purchase», les données s’envoit au serveur. Une alerte informera ensuite à l’utilisateur si l’ajout a bien été fait.

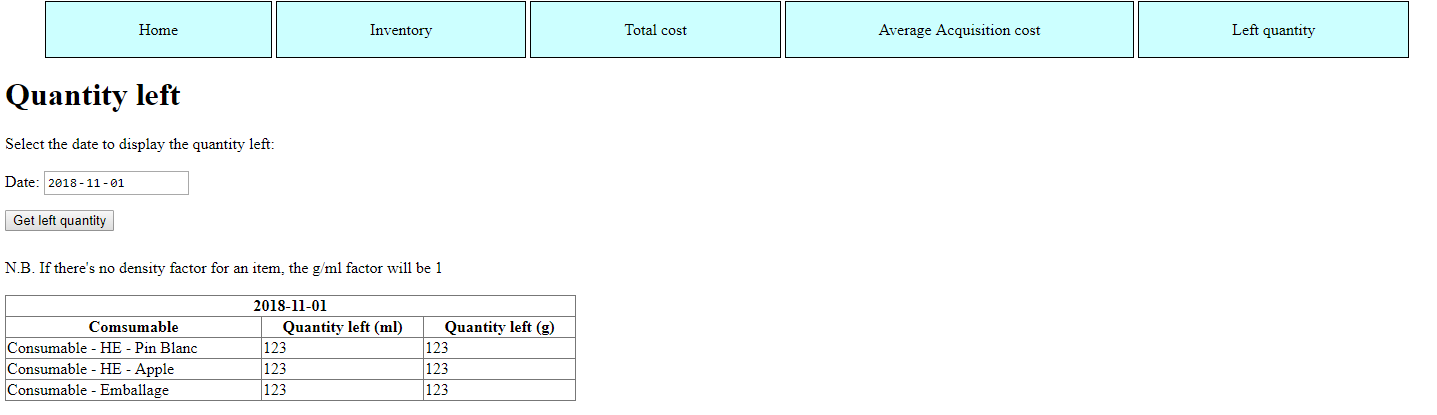
**Coût total**



Cette interface permet d’afficher le coût total selon une date précise pour une catégorie de matériel. L’utilisateur doit entrer les informations voulue et cliquer sur le bouton «Get total». Un tableau s’affichera avec les coûts totaux. S’il n’y a pas de totaux pour cette date ou catégorie, une mention l’affichera.

**Coût moyen d’acquisition**

**Quantité restante**

****

Cette interface permet d’afficher la quantité restante d’item selon une date précise pour une catégorie de matériel. L’utilisateur doit entrer les informations voulue et cliquer sur le bouton «Get letf quantity». Un tableau s’affichera avec les quantités restantes en millilitres et en gramme selon le facteur g/ml trouvé dans la base de donnée. S’il n’y a pas de densité pour une catégorie trouvée, le facteur sera alors de 1 g/ml. S’il n’y a pas de quantité restante lors de cette date, le tableau sera vide.