***Remise 1***

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 faire fonctionner l’application**

docker pull grizzlys54/glo4035a18-projet2018-eq2

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

**Collections**

* Transactions :
* Purchases :
* Densities :
* Labors :

**Requêtes**

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

* Purchases .find({})
* Densities .find({})
* Labors .find({})
* Purchases .remove({})
* Densities.remove ({})
* Labors .remove ({})
* Purchases .insert({"date": date, "item": item, "qte": qte, “unit": unit, "total": total, "stotal": stotal, "tax": tax})
* Densities.insert({"item": item, "ml": ml, "g": g})
* Labors.insert({"date": date, "item": item, "qte": qte, “unit": unit, “type": type, "jobId": jobId})
* Purchases.update({"\_id": ObjectId(id)},{"date": date, "item": item, "qte": qte, “unit": unit, "total": total, "stotal": stotal, "tax": tax})
* Densities.update ({"\_id": ObjectId(id)},{"item": item, "ml": ml, "g": g})
* Labors.update ({"\_id": ObjectId(id)},{"date": date, "item": item, "qte": qte, “unit": unit, “type": type, "jobId": jobId})

**Index utilisés**

Aucun index a été utilisé.

**Sécurité**

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

**API**

Route : request Get \--url

‘’adresse ip application’/

Paramètres : aucun

Retourne l’accueil de l’application.

Route : request GET \--url

‘’adresse ip application’/transactions’

Paramètres : aucun

Retourne l’interface pour afficher les données de toutes les collections.

Route : request POST \--url

‘’adresse ip application’/transactions’

Paramètres : {donnée csv}

Vérifie les données et les insères dans la base de donnée selon les données établies. Retourne une erreur 400.

Route : request DELETE \--url

‘’adresse ip application’’/transactions’

Paramètres : {mot de passe administrateur}

Supprime l’entièreté de la base de données si le mot de passe est correct, sinon retourne une erreur 401.

Route : request Get \--url

‘’adresse ip application’/transactions/purchases

Paramètres : aucun

Retourne le contenu de la collection purchases.

Route : request Get \--url

‘’adresse ip application’/transactions/densities

Paramètres : aucun

Retourne le contenu de la collection densities.

Route : request Get \--url

‘’adresse ip application’/transactions/labors

Paramètres : aucun

Retourne le contenu de la collection labors.

Route : request Get \--url

‘’adresse ip application’/transactions/add

Paramètres : aucun

Retourne l’interface pour ajouter un item dans une collection.

Route : request Post \--url

‘’adresse ip application’/transactions/add/purchase

Paramètres: {"date": date, "item": item, "qte": qte, “unit": unit, "total": total, "stotal": stotal, "tax": tax}

Envoit un item dans la collection purchases. Retourne un Json de succès avec un status 200 ou 400.

Route : request Post \--url

‘’adresse ip application’/transactions/add/density

Paramètres: {"item": item, "ml": ml, "g": g}

Envoit un item dans la collection density. Retourne un Json de succès avec un status 200 ou 400.

Route : request Post \--url

‘’adresse ip application’/transactions/add/labor

Paramètres: {"date": date, "item": item, "qte": qte, “unit": unit, “type": type, "jobId": jobId}

Envoit un item dans la collection labors. Retourne un Json de succès avec un status 200 ou 400.

Route : request Get \--url

‘’adresse ip application’/transactions/modifyPurchase

Paramètres : aucun

Retourne l’interface de modification des achats.

Route : request Post \--url

‘’adresse ip application’/transactions/modifyPurchase

Paramètres : {“id”: id, "date": date, "item": item, "qte": qte, “unit": unit, "total": total, "stotal": stotal, "tax": tax}

Modifie un item dans la collection purchases. Retourne un Json de succès avec un status 200 ou 400.

Route : request Get \--url

‘’adresse ip application’/transactions/modifyDensity

Paramètres : aucun

Retourne l’interface de modification des densités.

Route : request Post \--url

‘’adresse ip application’/transactions/modifyDensity

Paramètres: {“id”:id, "item": item, "ml": ml, "g": g}

Modifie un item dans la collection densitiesDb. Retourne un Json de succès avec un status 200 ou 400.

Route : request Get \--url

‘’adresse ip application’/transactions/modifyLabors

Paramètres : aucun

Retourne l’interface de modification des travaux effectué sur les savons.

Route : request Post \--url

‘’adresse ip application’/transactions/modifyLabors

Paramètres: {“id”: id, "date": date, "item": item, "qte": qte, “unit": unit, “type": type, "jobId": jobId}

Modifie un item dans la collection densities. Retourne un Json de succès avec un status 200 ou 400.

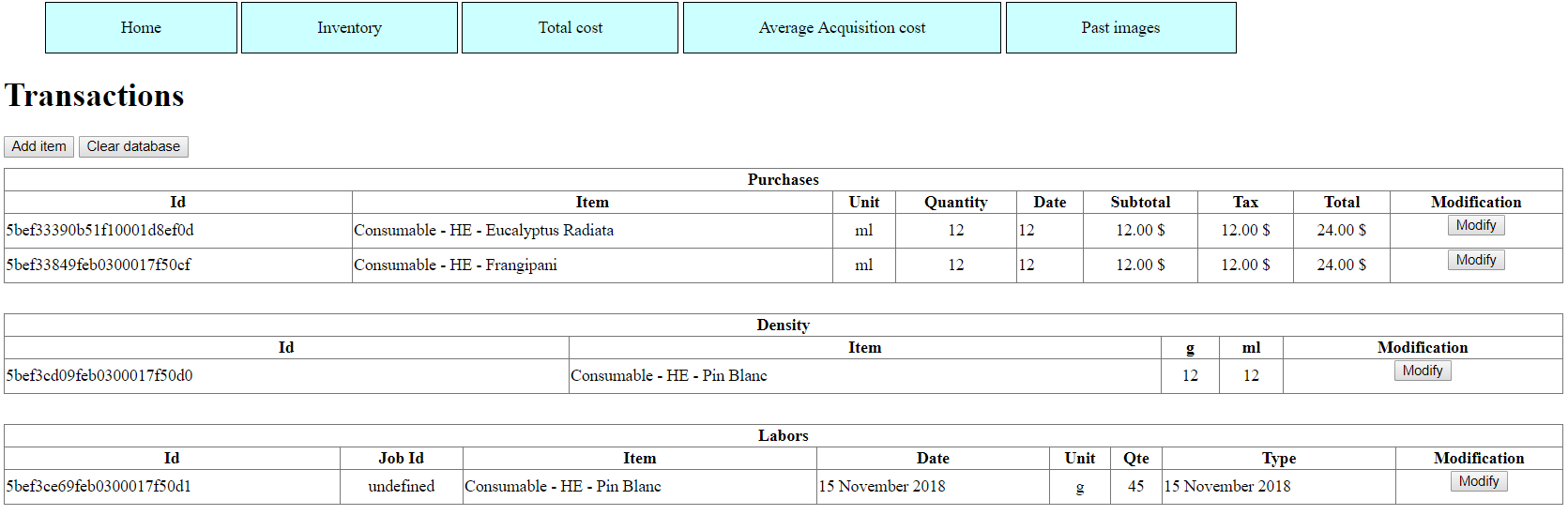
Interfaces

**Accueil**



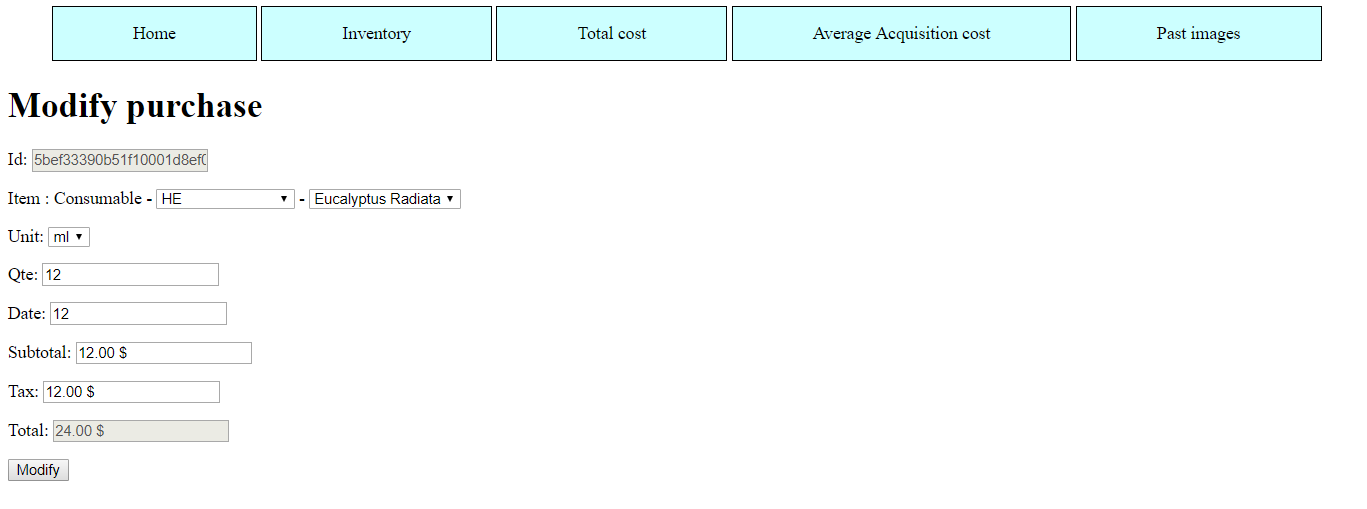
L’accueil de l’application. Permet de naviguer vers les autres pages.

**Listes des transactions**

****

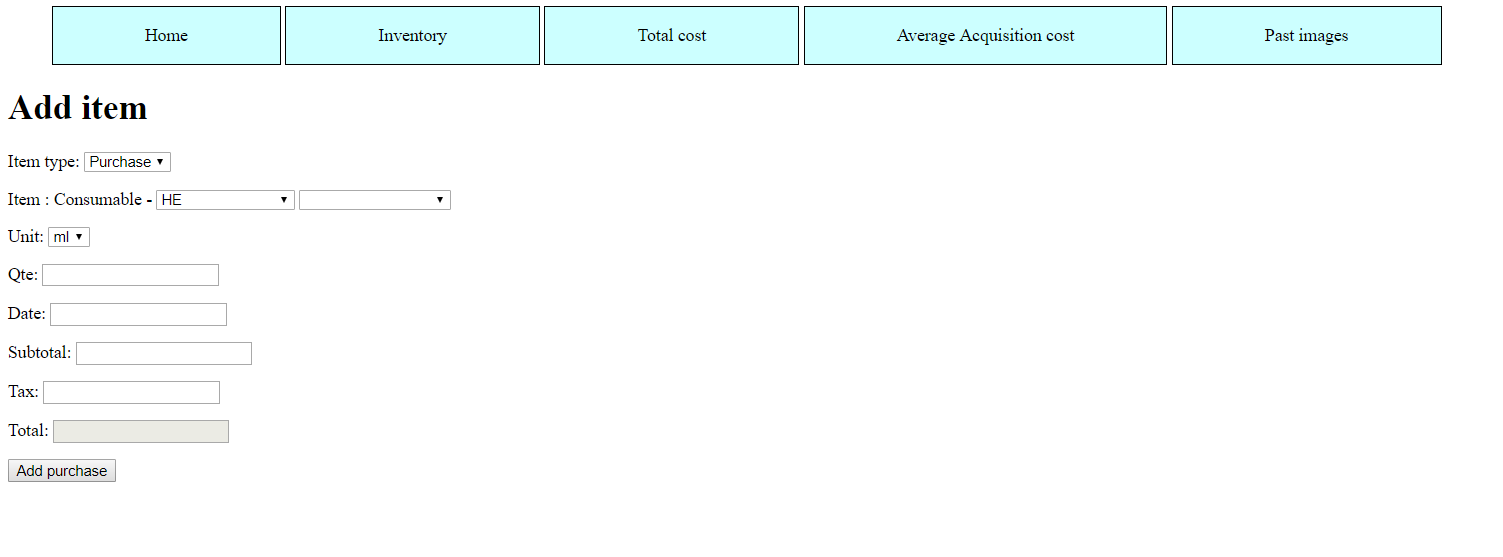
Permet de lister les données de toutes les collections. En cliquant sur «Add item», l’utilisateur sera transféré vers la page d’ajout d’un item. En cliquant sur «Clear database», l’utilisateur sera invité à entrer un mot de passe permettant de supprimer les données de toutes les collections. En cliquant sur «Modify», l’utilisateur sera transféré vers la page de modification de l’item associé au bouton.

**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. 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.