VI-Youcef BOUNEKTA

spécifications fonctionnelles :

***use case* :** Réaliser une opération de maintenance

Description du travail :

les acteurs concernés par cette partie sont les personnent capable de fair une opération de réparation que ce soit un manutentionnaire ou un réparateur .

tout d'abord le réparateur se connecte , après avoir réussi l'authentification ce dernier doit avoir la possibilité de réaliser plusieur tache et évidemment plusieurs changement au niveaux de l'interface graphique ainsi qu' au niveau de la base de données .

une foi le réparateur connecté il peut dans un premier temps consulter la liste des opération existante en en récupérant toute les lignes de la table " operation_sort " .cette opération est effectué après le clique sur un bouton précis .

le réparateur doit pouvoir sélectionner une opération pour s'en occuper cependant il n'a pa la possibilité de choisir quelle opération ,vue que les opération sont priorisées il existe la plus urgent , donc de la sélection le réparateur choisi automatiquement l'opération la plus urgente ceci doit se fair par un clique sur un bouton "prendre une operation ", suite a ce clique le réparateur doit pouvoir consulter toutes les informations concernant ce véhicule afin d'avoir une idé sur le type de panne le placement au dépôt ainsi que la durée estimé pour sa réparation qui été défini lors de son diagnostic (entré au dépôt).

on préfère voir sur la vue principale de l'application les information suivante : voiture et le type de panne (motif) .
par contre pour le reste des information (marque , type , date d'entré au dépot ...etc)doivent être accessibles que par un clique sur un bouton nommé (info vehicule),
a ce moment le réparateur a bien choisi le véhicule qui correspond à l'opération la plus urgente , il doit procéder donc à sa réparation pour cela il en a besoin des pièces , donc l'application doit lui permettre notamment de commander des pièces ainsi que de vérifier la disponibilité des piece .

un bouton "ajouter pièce" doit être présent sur la vue , en cliquant sur ce dernier une vue de selection de piece apparait , au niveau de cette dernière le réparateur a la possibilité de sélectionner le nom de la pièce souhaité ainsi que la quantité par contre , si le nom de la pièce est faux ou un des champs de saisie (piece , quantité) est vide l'application doit indiquer par un message d'erreur , si la quantité saisie par le réparateur est supérieur a celle de la pièce choisie en stock le système doit indiquer par un message d'erreur que la quantité demandé et supérieure à celle de la pièce en question en stock .

si la pièce existe en stock et la quantité est bien inférieure à celle de la pièce en stock le système doit non seulement réservé cette pièce , et informé le réparateur .
le système réserve la pièce en décrémentant le nombre associé a sa sa disponibilité en stock suivant le calcul suivant :

quantité stock := quantité stock - quantité saisie par le réparateur

le système affichera pour le réparateur un message de réussite de cette tâche , et la nom de la pièce ainsi que sa quantité (qte choisie par le réparateur) doivent être visible au niveau de la grande vue .
au cours de l'opération de réparation le réparateur peut constater de nouvelles pannes non détectées lors du diagnostic de ce véhicule et, pour cela l'application doit permettre l'ajout d'une opération (une opération et lié à une panne précise , bijection entre pannes et opérations) , donc un bouton "ajouter panne" doit être présent , en effet une vue doit s'afficher cette dernière elle permets la saisie du nom de la panne en cliquant sur le bouton ok de cette dernière vue une ligne s'ajoutera au niveau de la base de données dans la table opération avec les attribue suivant :

→ date début et date fin : nulle .
→ heur début heur fin : nulle .
→ id panne : c'est l'id lié au nom de la panne saisie par le réparateur .
→ login réparateur : c'est le login du réparateur connecté .
→ etat (donne) : en attente de réparation .

bien évidemment si le réparateur saisi un nom de panne qui n'existe pas dans le référentiel des pannes , un message d'erreur lui sera affiché , ainsi ,si le nom est bien valide un message lui sera affiché indiquant la réussite de l'ajout de panne à ce véhicule .

quand le réparateur termine son opération il clique sur le bouton fin d'opération , un autre bouton s'affichera "fin" , en cliquant sur ce dernié le système affiche au réparateur le nombre d'opération restantes concernant ce véhicule(recherché dans la table opération par numéro d'immatriculation) et effectue des modification au niveau de trois tables dans la base de données : "véhicule" et "opération" et "operation sort"

→ **au niveau de la table "vehicle"** : changement de statut de véhicule et le passage de "en attente de réparation" à "réparé " si et seulement si le nombre d'opération non encore effectuées (done= en attente) est égale à zéro , dans le cas contraire l'état du véhicule reste toujours "en attente de réparation" .

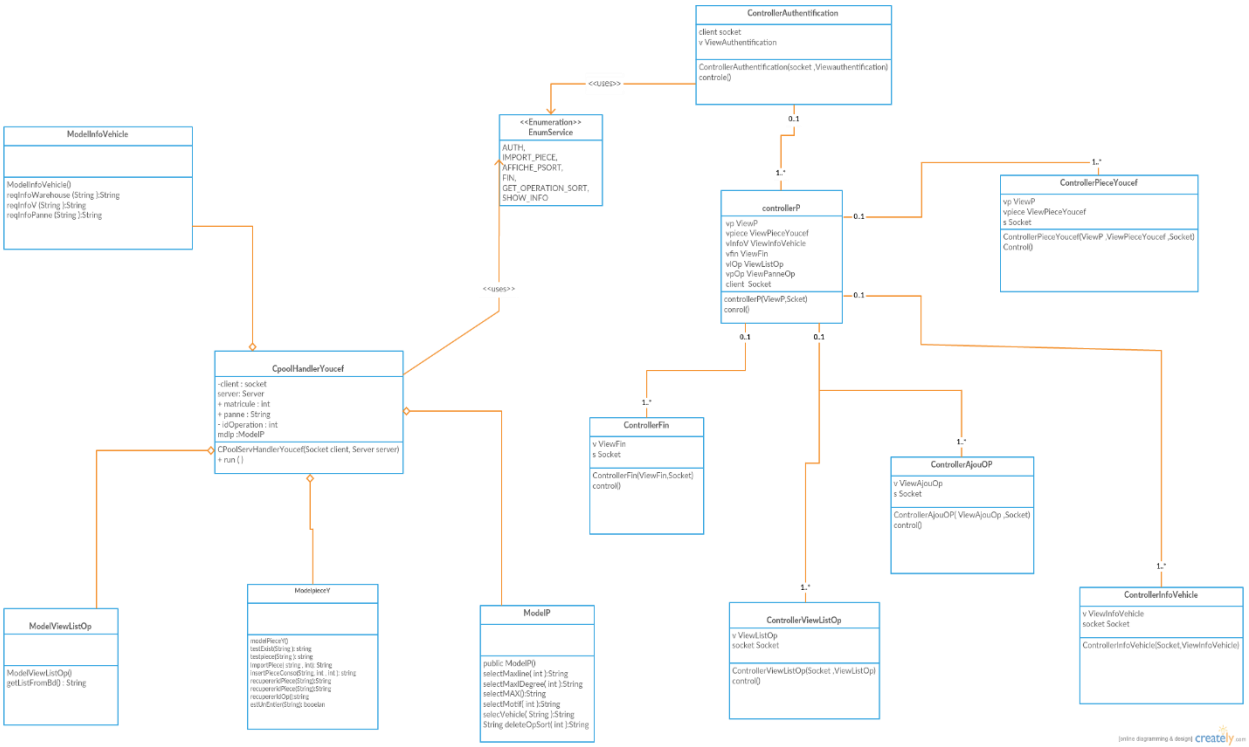
→ **au niveau de la table "operation sort"** :suppression de la ligne qui contient l'opération achevé .

→ **au niveau de la table "opération" :**le statut de l'opération passe de "non effectuée" à "effectuée " ,la date de début , date de fin , heur du début ainsi que l'heur de fin d'opération seront renseigné automatiquement par le système .

à ce moment les différents colonnes de la table "opération" sont renseignés et le véhicule réparé : soit sort de la file d'attente si le nombre d'opération restante concernant celui ci est nul soit il sera placé dans un autre emplacement dans la file d'attente selon l'algorithme de priorisation .

Diagramme de classe (BOUNEKTA YOUCEF) :

(le diagramme est aussi disponible sur le dossier LivrableR3 sur gitHub)
Le lien google pour le diagramme de classe :
https://drive.google.com/file/d/0B87rqGV0cuSbTzVicVUzU3pxVVE/view


ControllerAuthentification
client socket
v ViewAuthentification
ControllerAuthentification(socket ,Viewauthentification)
controle()
<<uses>>
<<Enumeration>>
EnumService
AUTH,
IMPORT_PIECE,
AFFICHE_PSORT,
FIN,
GET_OPERATION_SORT,
SHOW_INFO
0..1
1..*
ModelInfoVehicle
ModelInfoVehicle()
reqInfoWarehouse (String):String
reqInfoV (String):String
reqInfoPanne (String):String
controllerP
vp ViewP
vpiece ViewPieceYoucef
vInfoV ViewInfoVehicle
vFin ViewFin
vOp ViewListOp
vpOp ViewPanneOp
client Socket
controllerP(ViewP,Scket)
conrol()
0..1
ControllerPieceYoucef
vp ViewP
vpiece ViewPieceYoucef
s Socket
ControllerPieceYoucef(ViewP ,ViewPieceYoucef ,Socket)
Control()
<<uses>>
CpoolHandlerYoucef
<client : socket
server: Server
+ matricule : int
+ panne : String
- oiOperation : int
mdlp :ModelP
CPoolServHandlerYoucef(Socket client, Server server)
+ run ( )
ControllerFin
v ViewFin
s Socket
ControllerFin(ViewFin,Socket)
conrol()
ControllerAjouOP
v ViewAjouOp
s Socket
ControllerAjouOP(ViewAjouOp ,Socket)
control()
ControllerInfoVehicle
v ViewInfoVehicle
socket Socket
ControllerInfoVehicle(Socket,ViewInfoVehicle)
ModelViewListOp
ModelViewListOp()
getListFromBd() : String
ModelpieceY
modelPieceYO
testExistString): string
testpieceString): string
importPiece(string , int): String
insertPieceConnot(String, Int , Int): string
recupererlePieceString():String
recupererlePieceString():String
recupererlelOp():string
estUnExistant(String): boolean
ModelP
public ModelP()
selectMaxline(int):String
selectMaxIDegree(int):String
selectMAX():String
selectMotif(int):String
selecVehicle(String):String
String deleteOpSort(int):String
ControllerViewListOp
v ViewListOp
socket Socket
ControllerViewListOp(Socket ,ViewListOp)
control()
1..*
0..1
[online diagramming & design] creately.com