Institution Les Chartreux

Documentation

Application GSB

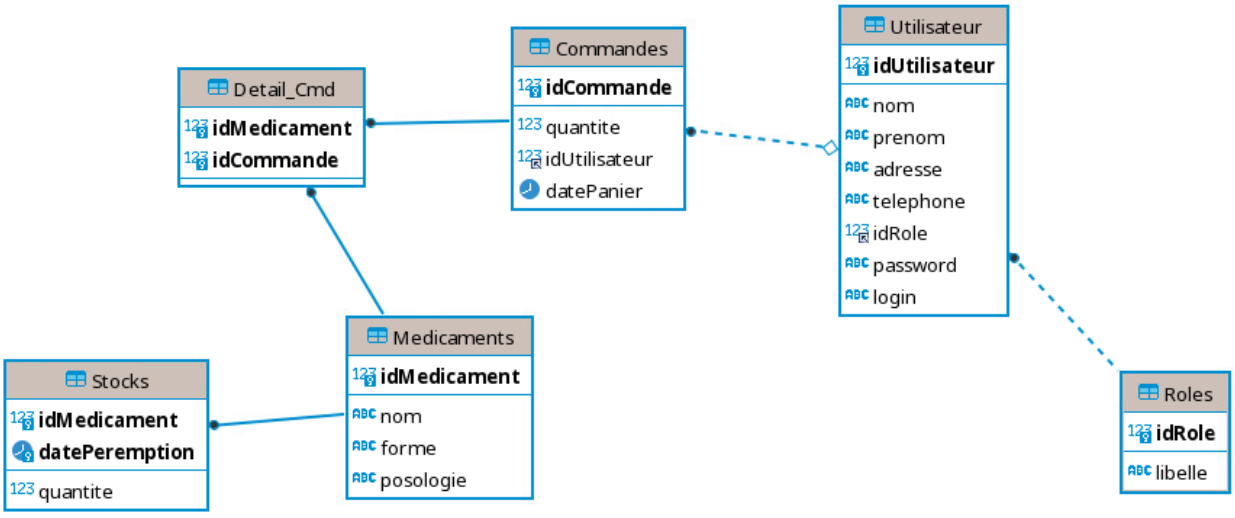
BOUQUET Nathan & BUYAT Dorian
09/06/2023

Le contexte :

Le laboratoire Galaxy Swiss Bourdin (GSB) est une entreprise pharmaceutique issue de la fusion entre Galaxy, un géant américain spécialisé dans les maladies virales, et Swiss Bourdin, un conglomérat européen travaillant sur des médicaments plus conventionnels. Suite à cette fusion, GSB cherche à optimiser son activité en réalisant des économies d'échelle dans la production et la distribution de médicaments. La France a été choisie comme témoin pour améliorer le suivi de l'activité de visite médicale. L'entreprise compte 480 visiteurs médicaux en France métropolitaine et 60 dans les départements et territoires d'outre-mer, répartis en 6 secteurs géographiques. Après deux années de réorganisations internes, GSB souhaite moderniser l'activité de visite médicale.

La base de donnée :

Avec un aperçu de ma base de données sous DBeaver :



Explication des liens de la base de donnée :

La création de la table "Detail_Cmd" dans notre projet a été réalisée dans un but éducatif pour approfondir notre compréhension des relations many-to-many et des concepts liés à la persistance des données bien que celle-ci puisse être évitée.

La table "Detail_Cmd" représente une relation de many-to-many (plusieurs-à-plusieurs) entre les entités "Medicaments" et "Commandes". Elle est utilisée pour stocker les détails spécifiques à chaque médicament dans une commande donnée. Chaque ligne de cette table représente une association entre un médicament et une commande.

La table "Stocks" permet de suivre et de gérer les niveaux de stock des différents médicaments. En enregistrant la quantité disponible et la date de péremption, elle fournit des informations essentielles pour la gestion des stocks, comme la possibilité de vérifier la disponibilité d'un médicament à une certaine date et de prévenir l'expiration des médicaments.

Le projet description :

Le projet "Client Loud" est une application de gestion de commandes de médicaments, développée par Dorian Buyat et Nathan Bouquet en utilisant le framework Hibernate. Cette application permet aux utilisateurs connectés de passer de voir les stocks de medicaments, aux secretaires de passer des commandes de médicaments, tandis que les administrateurs ont la possibilité d'ajouter, modifier ou supprimer des médicaments, ainsi que de gérer les stocks et avoir accès à l'historique des commandes.

Le framework Hibernate facilite l'interaction avec les bases de données en simplifiant la manipulation des données. Le projet utilise la base de données mariaDB pour le stockage et la gestion des informations relatives aux médicaments, aux commandes et aux utilisateurs.

Ainsi, ce Client Loud offre une solution efficace et conviviale pour la gestion des commandes de médicaments, en mettant l'accent sur la facilité d'utilisation et la sécurisation des données.

Les technologies utilisées :

Le projet que j'ai réalisé utilise un certain nombre de technologies pour offrir une expérience utilisateur agréable et des fonctionnalités. Voici les principales technologies que j'ai utilisées :

- hibernate : pour avoir différentes fonctionnalités telles que les contrôleurs de route, les migrations de base de données et les vues.
- java : pour avoir un système d'authentification moderne.
- mariadb : pour stocker les données de votre application, y compris les informations sur les utilisateurs, les medicament,...

Fonctionnalités :

Descriptif des fonctionnalités utilisateur et administrateur du projet Hibernate

Fonctionnalités pour les utilisateurs :

Configuration : La configuration d'Hibernate à été assez compliqué. Il faut bien configurer le fichier hibernate.cfg.xml ou persistence.xml avec les informations de connexion à la base de données, les paramètres de transaction et les paramètres de cache de deuxième niveau.

Code Technique :

Mapping des objets et des tables : Nous avons eu une difficulté concernant la création des fichiers de mapping pour décrire la relation entre les objets Java et les tables de base de données. Il faut définir correctement les annotations ou les fichiers XML de mapping pour éviter des erreurs de correspondance.

```
32
33      @Id
34      @GeneratedValue(strategy=GenerationType.AUTO)
35      private int idCommande;
36
37      @Temporal(TemporalType.DATE)
38      private Date datePanier;
39
40      @Column(name="idUtilisateur")
41      private int idUtilisateur;
42
43      //bi-directional many-to-many association to Medicament
44      @ManyToMany (cascade = {CascadeType.ALL})
45      @JoinTable( name = "Detail_Cmd",
46                  joinColumns = @JoinColumn(name="idCommande"),
47                  inverseJoinColumns = @JoinColumn( name = "idMedicament"))
48      private List<Medicament> medicaments = new ArrayList<>();
49
50      //bi-directional many-to-one association to Utilisateur
51      @ManyToOne
52      @JoinColumn(name="idUtilisateur", insertable = false, updatable = false)
53      private Utilisateur utilisateur;
54
55      public Commande() {
```

Dans cette partie, on établit une relation many-to-many entre les entités Commande et Medicament.

L'annotation @ManyToMany indique qu'il s'agit d'une relation many-to-many, et l'attribut cascade spécifie que toutes les opérations (persist, merge, remove, etc.) sur une commande doivent être propagées aux médicaments associés.

L'annotation @JoinTable est utilisée pour définir la table intermédiaire, appelée "Detail_Cmd", qui gère cette relation many-to-many. Les attributs joinColumns et inverseJoinColumns sont utilisés pour indiquer les clés étrangères dans la table intermédiaire.

La déclaration private List<Medicament> medicaments = new ArrayList<>(); crée une liste de médicaments associée à une commande.

```
 9  *
10  */
11 @Entity
12 @Table(name="Stocks")
13 @NamedQuery(name="Stock.findAll", query="SELECT s FROM Stock s")
14 public class Stock implements Serializable {
15     private static final long serialVersionUID = 1L;
16
17     @EmbeddedId
18     private StockPK id = new StockPK();
19
20     @ManyToOne
21     @JoinColumn(name="idMedicament", insertable = false, updatable = false)
22     private Medicament medicament;
23
24     private int quantite;
25
26     public Stock() {
27     }
28
29     public StockPK getId() {
30         return this.id;
31     }
32
```

@EmbeddedId est utilisé pour spécifier qu'un attribut de l'entité fait partie de la clé primaire composite de la table. Dans ce cas, l'attribut id de type StockPK est utilisé comme clé primaire composite. La classe StockPK doit contenir les attributs qui composent la clé primaire.

L'annotation @ManyToOne indique qu'il existe une relation many-to-one entre les entités Stock et Medicament. Cela signifie qu'un enregistrement de la table "Stocks" est associé à un seul enregistrement de la table "Medicament".

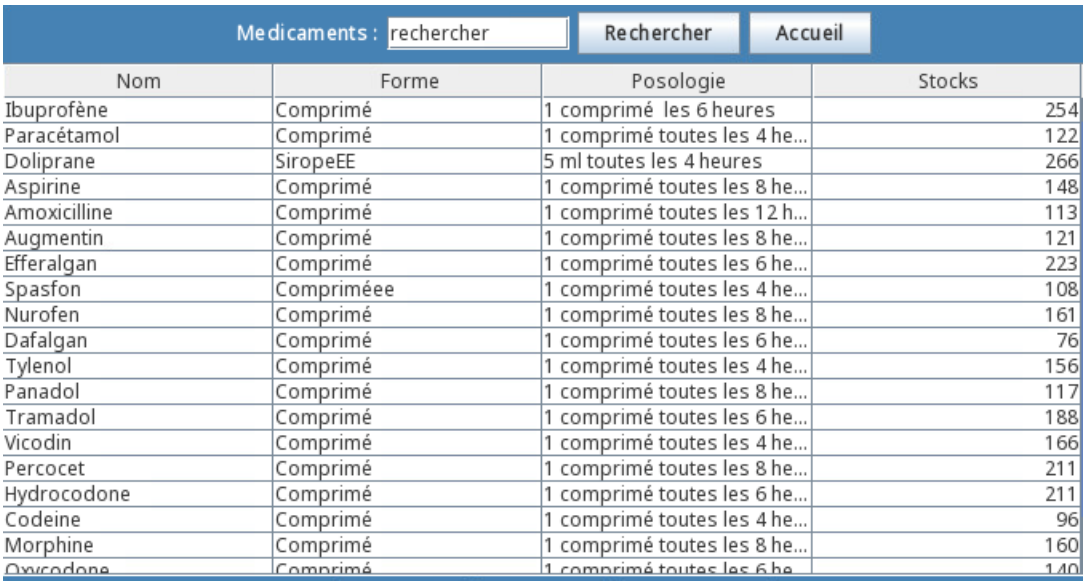
L'annotation @JoinColumn est utilisée pour définir la clé étrangère (nommée "idMedicament") dans la table "Stocks" qui fait référence à l'entité Medicament. Les attributs insertable et updatable sont définis à false pour indiquer que la colonne "idMedicament" ne doit pas être mise à jour ou insérée directement dans la table "Stocks". La mise à jour ou l'insertion de cette colonne doit être réalisée en manipulant l'objet Medicament associé.

Le site web propose les fonctionnalités suivantes :

- Connexion : les utilisateurs peuvent se connecter à leur compte pour accéder aux fonctionnalités réservées aux utilisateurs connectés en fonction de leur rôle.

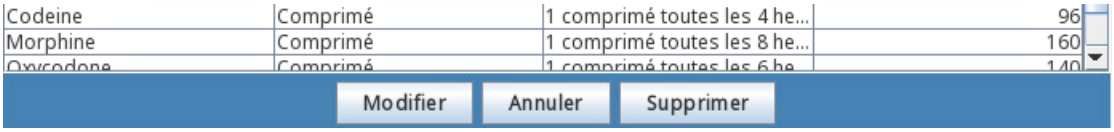
- Ils peuvent accéder à leur profil.

- Affichage des médicaments : les utilisateurs peuvent consulter la liste des médicaments disponibles sur l'application.

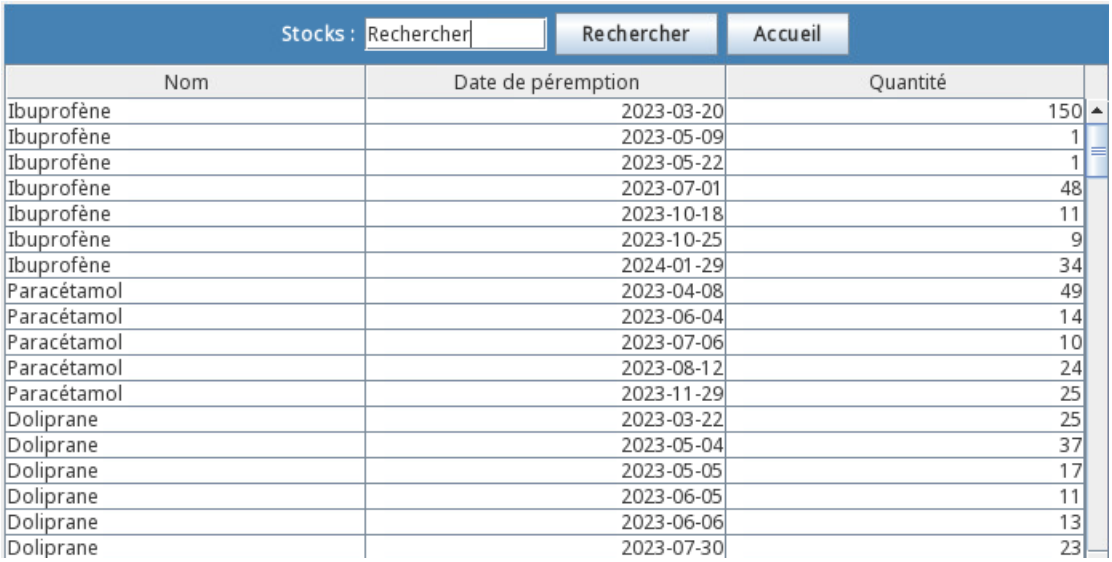


Nom	Forme	Posologie	Stocks
Ibuprofène	Comprimé	1 comprimé les 6 heures	254
Paracétamol	Comprimé	1 comprimé toutes les 4 he...	122
Doliprane	SiropeEE	5 ml toutes les 4 heures	266
Aspirine	Comprimé	1 comprimé toutes les 8 he...	148
Amoxicilline	Comprimé	1 comprimé toutes les 12 h...	113
Augmentin	Comprimé	1 comprimé toutes les 8 he...	121
Efferalgan	Comprimé	1 comprimé toutes les 6 he...	223
Spasfon	Compriméee	1 comprimé toutes les 4 he...	108
Nurofen	Comprimé	1 comprimé toutes les 8 he...	161
Dafalgan	Comprimé	1 comprimé toutes les 6 he...	76
Tylenol	Comprimé	1 comprimé toutes les 4 he...	156
Panadol	Comprimé	1 comprimé toutes les 8 he...	117
Tramadol	Comprimé	1 comprimé toutes les 6 he...	188
Vicodin	Comprimé	1 comprimé toutes les 4 he...	166
Percocet	Comprimé	1 comprimé toutes les 8 he...	211
Hydrocodone	Comprimé	1 comprimé toutes les 6 he...	211
Codeine	Comprimé	1 comprimé toutes les 4 he...	96
Morphine	Comprimé	1 comprimé toutes les 8 he...	160
Oxycodone	Comprimé	1 comprimé toutes les 6 he...	140

- Gestion des médicaments : les administrateurs peuvent ajouter, modifier ou supprimer des médicaments sur l'application.



- Affichage des médicaments : les utilisateurs peuvent consulter la liste de stock sur l'application.



Nom	Date de péremption	Quantité
Ibuprofène	2023-03-20	150
Ibuprofène	2023-05-09	1
Ibuprofène	2023-05-22	1
Ibuprofène	2023-07-01	48
Ibuprofène	2023-10-18	11
Ibuprofène	2023-10-25	9
Ibuprofène	2024-01-29	34
Paracétamol	2023-04-08	49
Paracétamol	2023-06-04	14
Paracétamol	2023-07-06	10
Paracétamol	2023-08-12	24
Paracétamol	2023-11-29	25
Doliprane	2023-03-22	25
Doliprane	2023-05-04	37
Doliprane	2023-05-05	17
Doliprane	2023-06-05	11
Doliprane	2023-06-06	13
Doliprane	2023-07-30	23

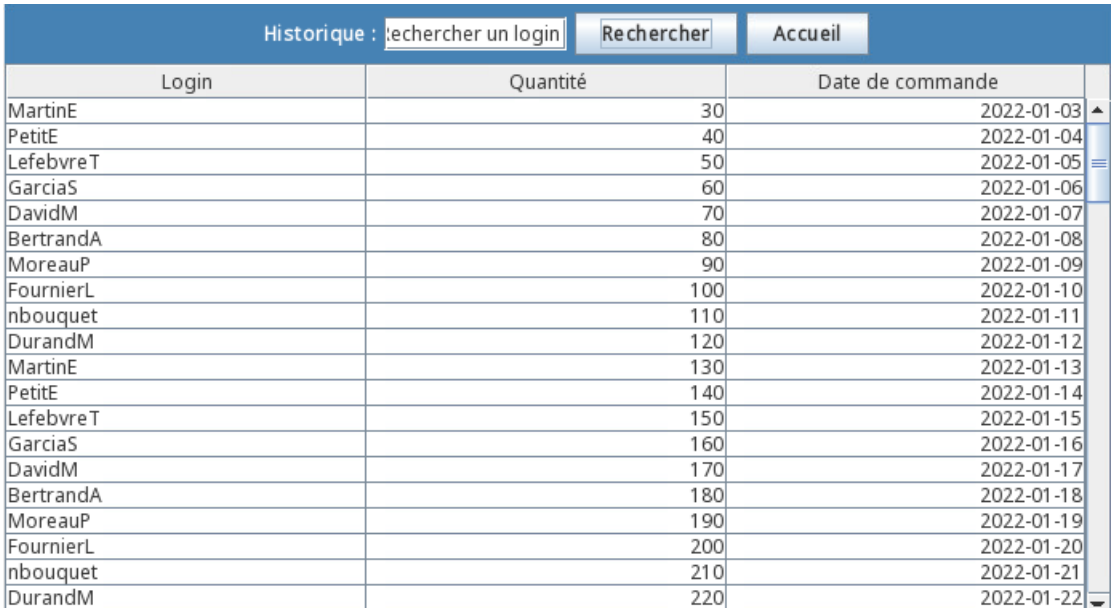
- Gestion des futurs médicaments expirés : affichage des futurs médicaments périmés avec la possibilité de les ajouter dans le panier des prochaines commandes puis de voir celui-ci.

Nom	Date de péremption	Quantité expiré
Tylenol	2023-06-17	36
Vicodin	2023-06-26	13
Percocet	2023-06-11	35
Percocet	2023-06-17	29
Morphine	2023-06-17	46
Demerol	2023-06-11	29
Demerol	2023-06-18	44

- Commande de médicaments : les secrétaires peuvent commander des médicaments en indiquant la quantité souhaitée ou un prix souhaité.

Le prix varie en fonction de la quantié souhaité. Cela fonctionne comme un pannier marchand que l'on pouurait retrouver sur un site de e-commerce.

- Voir l'historique des commande : L'administrateur peut voir les commandes effectué par ses secrétaires ou lui même.



Login	Quantité	Date de commande
MartinE	30	2022-01-03
PetitE	40	2022-01-04
LefebvreT	50	2022-01-05
GarciaS	60	2022-01-06
DavidM	70	2022-01-07
BertrandA	80	2022-01-08
MoreauP	90	2022-01-09
FournierL	100	2022-01-10
nbouquet	110	2022-01-11
DurandM	120	2022-01-12
MartinE	130	2022-01-13
PetitE	140	2022-01-14
LefebvreT	150	2022-01-15
GarciaS	160	2022-01-16
DavidM	170	2022-01-17
BertrandA	180	2022-01-18
MoreauP	190	2022-01-19
FournierL	200	2022-01-20
nbouquet	210	2022-01-21
DurandM	220	2022-01-22

Conclusion :

Au terme de ce projet, l'application "Client Lourd" développée par Dorian Buyat et Nathan Bouquet permet d'adresser les besoins du laboratoire Galaxy Swiss Bourdin en termes de gestion des commandes de médicaments, de suivi des stocks et de communication entre les visiteurs médicaux et le siège parisien. En s'appuyant sur des technologies éprouvées telles qu'Hibernate, Java et MariaDB, cette solution contribue à l'optimisation des processus internes et à l'amélioration de l'efficacité de la force commerciale de l'entreprise.

Les fonctionnalités offertes par l'application répondent aux attentes des utilisateurs et des administrateurs en proposant une interface conviviale et sécurisée. La gestion des médicaments, des stocks et des commandes est facilitée, permettant ainsi un meilleur suivi de l'activité et une optimisation des frais engagés par les visiteurs.

Le succès de ce projet repose sur la maîtrise des différentes technologies utilisées et la compréhension des enjeux métier du laboratoire GSB. Le déploiement de l'application "Client Lourd" constitue une étape importante dans la modernisation et l'optimisation des activités de visite médicale de GSB, avec pour objectif ultime d'améliorer la satisfaction des professionnels de santé et d'accroître la performance commerciale de l'entreprise.

Dans le futur, des améliorations pourraient être apportées à l'application, notamment en intégrant de nouvelles fonctionnalités liées à l'analyse des données et à l'intelligence d'affaires pour aider GSB à prendre des décisions éclairées et à identifier de nouvelles opportunités de croissance.

Lien GitLab : https://gitlab.com/sco-chartreux/slam-22-23/t5/ppelourd