8

Sommaire

I – Contexte..................................................................................................................................….........2 Contexte général :........….................................................……………….……….................................2

Contexte technique :..................................………………...…..........................................................2

II - Fonctionnement de l’application (IHM).......................................................................… ....................2 Rôle Restaurateur.........................………………..............................................................................3

Rôle Infirmier.................................…………………………..................................................................4

III – MCD et BDD.....................................................................................................................................7

IV - Diagramme de classe.......................................................................................................................7

V - Classe Passerelle...............................................................................................................................8

VI – GIT / Redmine.................................................................................................................................8

# I – Contexte

## Contexte général :

L’hôpital 2 a eu le besoin d’avoir une application desktop utilisée par la restauration et par les services de l’hôpital. Le service de restauration propose différents types de repas chaque jour (végétarien, sans sel, sans lactose, …). Cette application permet la transmission automatique des nombres des repas de différents types.

## Contexte technique :

Cette application est codée sous NetBeans 12.0 sous Maven en Java (Java Swing pour les IHM).

Maven est un outil de construction de projets qui facilite et automatise certaines tâches de gestion. Ici il va permettre l’affichage du .pdf des demandes particulières pour les restaurateurs. Grâce à Maven, on pourra mettre à jour plus facilement les bibliothèques (dependency) nécessaires au projet.

Côté Serveur : Apache

Côté Gestion Base de Données : PgAdmin 4 (PostGreSQL)

Système d’exploitation : Windows 10

# II - Fonctionnement de l’application (IHM)



Lors du lancement de l’application, l’utilisateur devra s’identifier afin d’avoir accès aux données jugés sensibles de l’hôpital.

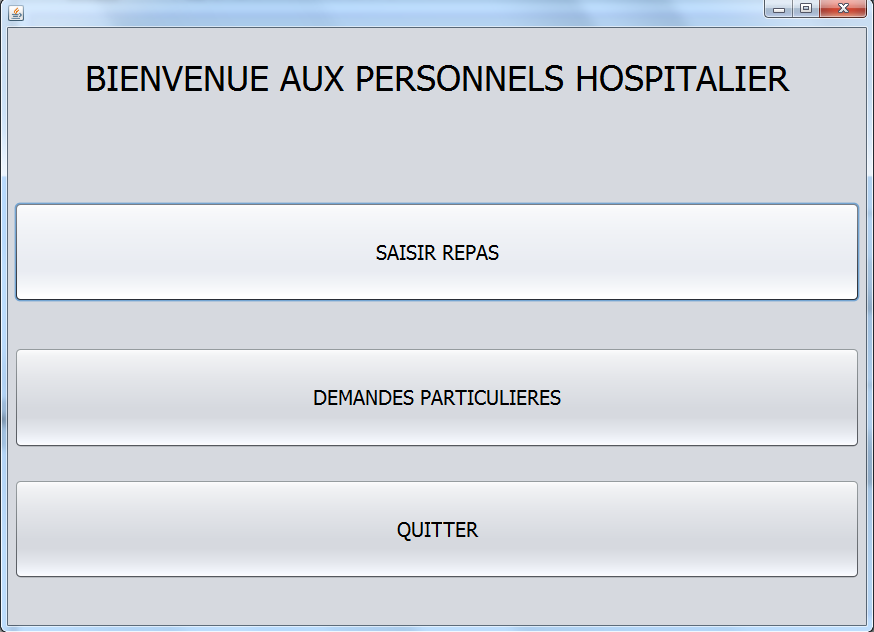
Une fois l’authentification réussie, nous avons 2 chemins différents pour l’IHM : Le Rôle Restaurateur et le Rôle Infirmier.

## Rôle Restaurateur

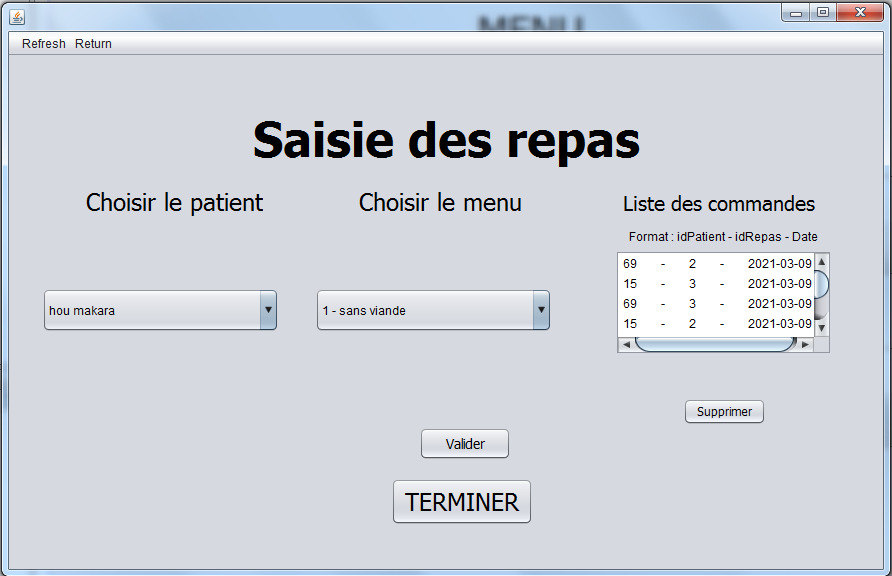
Page d’accueil :

Le restaurateur n’aura le droit que d’imprimer les demandes particulières en format .pdf afin de pouvoir réaliser les différents plats pour les patients.

## Rôle Infirmier

Page d’accueil :

L’infirmier n’aura le droit que d’enregistrer les repas pour les patients afin de les communiquer au service restauration.



Lorsque l’on clique sur « Saisir Repas », on accède à la fenêtre ci-dessus qui nous propose de choisir plusieurs paramètres afin qu’une saisie des repas soit enregistrée.

* Le bouton Valider (jButton1MouseClicked) récupère les valeurs choisis dans la liste des patients (jComboBox1) et la liste des menus disponible (jComboBox2) pour les insérer dans la liste des commandes en effectuant la fonction **commander(Patient, Repas)** dans **Passerelle.**
* Le bouton Supprimer utilise la fonction supprimerCommande(uneCommande) pour supprimer une commande.
* Dans cette fenêtre, on affiche les données grâce aux fonctions de la Class

Passerelle ( **lesCommandes() ; tousLesPatients() ; tousLesRepas()** ).

* Le bouton Refresh, importante dans cette fenêtre, va venir actualiser les Repas avec la fonction dispose() de Java.

La page demande particulière est un moyen de faire inscrire un nouveau patient et de lui ajouter un repas le plus rapidement possible.

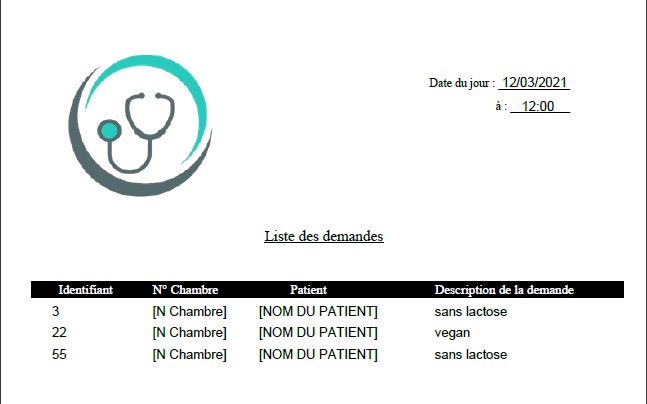
Comme la précédente fenêtre, la page demande particulière va insérer une commande grâce au bouton ENTRER (jButton2ActionPerformed) cependant en plus de rajouter une commande, il va créer en même temps un nouveau patient avec les données passées en paramètre.

* Le processus est le suivant :

1. On insère d’abord le Patient dans la BDD via la fonction passerelle.creerPatient(int pId, int pNumChambre, String pNom,String pPrenom, int pIdService)
2. On insère ensuite la commande dans la BDD via la fonction passerelle.commander(unPatient, unRepas)

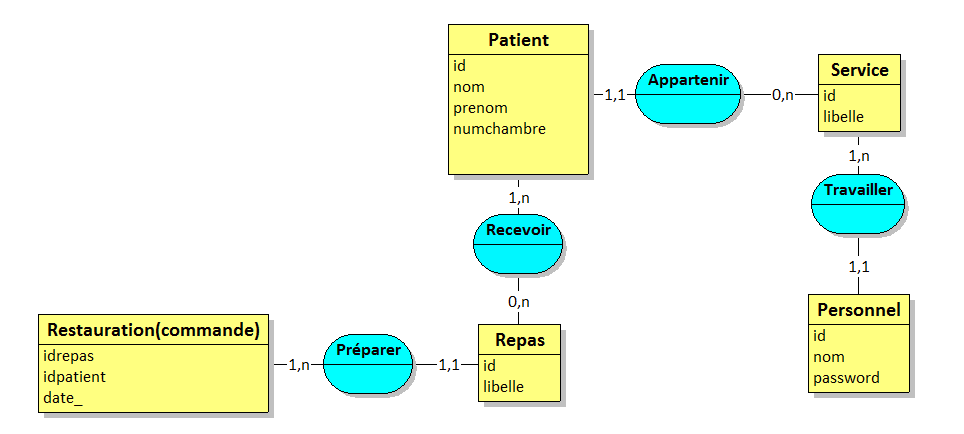
Comme cité précédemment, le bouton Refresh est utilisé pour rafraichir la page et le bouton Return permet de retourner à la page précédente (le plus souvent vers la page d’accueil).

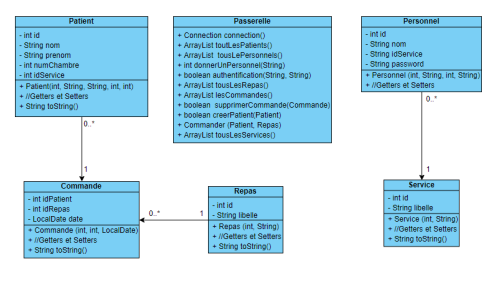
Exemple de demande particulière pour le personnel de restauration :



# III – MCD et BDD

# IV - Diagramme de classe





# V - Classe Passerelle

Cette classe Passerelle va nous permettre de regrouper toutes les fonctions possibles permettant le bon fonctionnement de l’application Java. Elle est répartie en X partie :

**CONNEXION :**

→ connection() : Permet la connexion à la base de données de l’hôpital

→ authentificationPersonnel(String id, String mdp) : Permet la connexion à l’application

**PATIENT :**

→ tousLesPatients() : Permet d’afficher tous les patients de l’hôpital

→ unPatient(int pId) : Permet d’afficher un seul patient en fonction de l’id

→ creerPatient (Patient unPatient) : Permet la création d’un nouveau patient au sein de l’hôpital

**PERSONNEL :**

→ tousLePersonnels() : Permet d’avoir la liste de tous le personnels hospitalier

→ donnerUnPersonnel(String id) : Permet d’avoir un membre du personnel en fonction de son id

**REPAS :**

→ tousLesRepas() : Permet d’afficher tous les repas dans la restauration de l’hôpital

→ unRepas(int pId) : Permet d’afficher un seul repas en fonction de l’id

**SERVICE :**

→ tousLesServices() : Permet d’avoir tous les services de l’hôpital

**COMMANDE :**

→ commander(Patient unPatient, Repas unRepas) : Cette fonction permet au personnel hospitalier de créer une commande repas pour un patient et un repas selectionné

→ lesCommandes() : Permet d’afficher toutes les commandes enregistrées

→ supprimerCommande(Commande uneCommande) : En cas d’erreur cette fonction permet de supprimer une commande

# VI – GIT / Redmine

Git : git clone https://github.com/YassineGuendouzi/PPE\_java.git

Redmine : 192.168.1.44