# Rôles de chaque classe et de chaque méthode

## Package Contrôleurs

***Tous les contrôleurs ont la méthode public void actionPerformed(ActionEvent e). Cette méthode permet d’attribuer des actions à certains évènements.***

Controleur\_AjoutModif : Classe contrôleur de l'interface Ajout et modification d'éléments implémentant l'interface ActionListener

Controleur\_Ajout\_Listes : Classe contrôleur de l'interface Ajout Listes implémentant l'interface ActionListener. Cette classe dispose en plus des méthodes suivantes :

public DocumentListener effectuerRecherche() : Méthode qui se déclenche lorsque l’on saisit du texte dans le JTextFiled recherche. Cette méthode effectue une recherche dans le tableau pour trouver les objets.

public void actualiser() : Méthode qui actualise l'interface des listes de diffusion

public void trierTableau() : Méthode qui trie le tableau contenant les locataires avec différents critères ( Age / Date de naissance)

public ChangeListener changerNombre() : Méthode qui actualise le tableau lorsque l'on modifie le nombre

public PropertyChangeListener changerDate() : Méthode qui actualise le tableau lorsque l'on modifie la date

public TableModelListener cocherLesCases() : Méthode qui actualise la liste des id des cases cochés dans le modèle lorsque l'on clique sur une case

Controleur\_Association : Classe contrôleur association implémentant l'interface ActionListener. Cette classe dispose en plus de la méthode suivante :

public void remplirComboBoxRetirer() : Méthode qui vide et remplie les ComboBox pour mettre les logements associés aux Locataires

Controleur\_Connexion : Classe de l'interface de connexion implementant l'interface ActionListener. Cette classe dispose en plus des méthodes suivantes :

public KeyListener toucheEntree() : Méthode de la classe abstraite KeyListener

public void seConnecter() : Méthode qui vérifie la saisie des champs et change l'interface à l'utilisateur

Controleur\_Gestion : Classe de l'interface gestion des objets dérivant de KeyAdapter et implémentant ActionListener et MouseListener. Cette classe dispose en plus des méthodes suivantes :

public DocumentListener effectuerRecherche() : Méthode qui effectue une recherche dans le tableau pour trouver les objets contenant une similitude avec la chaine saisie dans le label de la vue du controleur

public void mousePressed(MouseEvent e) : Méthode de la classe abstraite MouseListener

Controleur\_Menu : Classe Controleur\_menu implementant l'interface ActionListener.

Controleur\_Statistique : Classe de l'objet Conroleur\_Statistique implementant l'interface ActionListener

Controleur\_dialog\_locataire : Classe de l'objet Controleur\_dialog\_locataire implementant ActionListener

## Package DAO

***Les classes suivantes permettent de se connecter à la base de données : ConnectionBDD, Connector, Connexion.***

***La classe abstraite générique DAO permet de déclarer des objets d’accès à une base de données. Cette classe est implémentée par Appartement\_DAO, Campagne\_DAO, ListeDeDiffusion\_DAO, Locataire\_DAO, Maison\_DAO, Utilisateurs\_DAO. Cette classe dispose des méthodes suivantes :***

public boolean create(T obj) : Méthode de création d'un enregistrement

public boolean delete(T obj) : Méthode de suppression d'un enregistrement

public boolean update(T obj) : Méthode de mise à jour d'un enregistrement

public T selectById(int id) : Méthode de recherche des informations par l'id

public int getLastInsertId(int id) : Méthode qui retourne l'id du dernier objet inséré

public List<T> getAll() : Méthode qui retourne tous les objets de la table sous forme de liste

protected MyDate getMyDate(Date d) : Méthode qui transforme une date dans une base de données en objet MyDate

protected MyTime getMyTime(Time t) : Méthode qui transforme une heure dans une base de données en objet MyTime

Appartement\_DAO : Classe permettant de se connecter à la base de données pour la table Appartement et d'effectuer diverses actions sur la table. Cette classe dispose en plus des méthodes suivantes :

public int getLogementEmpty() : Méthode qui récupère les logements vides dans la base de données

public int getLogementFull() : Méthode qui récupère les logements occupés de la base de données

public int getAllLogement() : Méthode qui récupère le nombre total de logements de la base de données

public boolean deleteAllLogementByIdLogementId(Appartement obj) : Méthode qui supprime les logements de la table habiter

Campagne\_DAO : Classe permettant de se connecter à la base de données pour la table campagne et d'effectuer diverses actions sur la table. Cette classe dispose en plus des méthodes suivantes :

public boolean updateEtatCampagne(Campagne obj) : Méthode qui modifie l'état d'une campagne après l'envoi d'un mail

public List<Campagne> getAllSurveillance() : Méthode qui récupère uniquement les campagnes en cours nécessitant l'envoi d'un mail

public ListeDeDiffusion getListeDeDiffusionByIdCampagne(int id) : Méthode qui récupère toutes les listes de diffusion pour n'en faire qu'une seule hors de la base de données par rapport à une campagne

public int getFinishCampagne() : Méthode qui récupère le nombre de campagnes finies

public int getNotBeginCampagne() : Méthode qui retourne le nombre de campagnes qui n'ont pas commencées

public int getNowCampagne() : Méthode qui récupère le nombre de campagnes en cours

public int getAllCampagne() : Méthode qui retourne le nombre total de campagnes

public List<Campagne> getIdCampagneByListeDeDiffusionBy(ArrayList<Integer> listeId) : Méthode qui récupère la liste des campagnes auxquelles une liste de diffusion est associée

ListeDeDiffusion\_DAO : Classe permettant de se connecter à la base de données pour la table Liste. Cette classe dispose en plus des méthodes suivantes :

public boolean createListeLocataires(int id, ArrayList<Personne> liste) : Méthode qui crée une liste de locataires dans la base de données

public boolean createListeUtilisateurs(int id, ArrayList<Personne> liste) : Méthode qui crée une liste d'utilisateurs dans la base de données

public boolean deleteListe(int id) : Méthode qui supprime les liens entre une liste et ses Utilisateurs / Locataires dans la base de données

public List<Personne> getAllLocataires(int id) : Méthode qui récupère tous les locataires de la base de données avec l'id d'une liste de diffusion

public List<Personne> getAllUtilisateurs(int id) : Méthode qui récupère tous les utilisateurs de la base de données avec l'id d'une liste de diffusion

public List<Integer> searchListLocataireByIdLocataire(int id) : Méthode qui récupère tous les id des listes de diffusion auxquelles l'id locataire est associé

Locataire\_DAO : Classe permettant de se connecter à la base de données pour la table Locataire et d'effectuer diverses actions sur la table. Cette classe dispose en plus des méthodes suivantes :

public List<Locataire> getRequete(String requete) : Méthode qui exécute une requete et retourne une liste de Locataires en fonction

public List<Batiment> getLocation(int id) : Méthode qui retourne la liste des batiments d'un locataire par son id

public Batiment selectBatimentById(int id) : Méthode qui récupère un batiment par rapport à son id

public int compterLocataire(List<Locataire> liste, int ageMin, int ageMax) : Méthode qui compte le nombre de locataires compris entre deux âges

public boolean deleteAllLogementByIdLocataire(Locataire obj) : Méthode qui supprime le lien entre les batiments d'un locataire

public boolean deleteAllListesByIdLocataire(Locataire obj) : Méthode qui supprime le locataire des listes de diffusion

Maison\_DAO : Classe permettant de se connecter à la base de données pour la table Logement pour les attributs n'ayant pas de numéro d'étage et d'effectuer diverses actions sur la table. Cette classe dispose en plus de la méthode suivante :

public boolean deleteAllLogementByIdLogementId(Maison obj) : Méthode qui supprime les maisons de la base de données

Utilisateurs\_DAO : Classe permettant de se connecter à la base de données pour la table Utilisateur et d'effectuer diverses actions sur la table. Cette classe dispose en plus des méthodes suivantes :

public Utilisateur selectByName(String login) : Méthode qui sélectionne un Utilisateur par son login dans la base de données

public List<Utilisateur> getAllUtilisateurs() : Méthode qui sélectionne tous les Utilisateurs de catégorie utilisateur 1 et 2 de la base de données

public List<Utilisateur> getAllUtilisateursEtGestionnaires() : Méthode qui sélectionne toutes les catégories d'Utilisateurs différentes de la catégorie administrateur de la base de données

public List<Utilisateur> getRequete(String requete) : Méthode qui exécute une requete et retourne une liste d'Utilisateurs en fonction

public boolean deleteAllListesByIdUtilisateur(Utilisateur obj) : Méthode qui supprime un utilisateur des listes de diffusion

***Les deux classes suivantes n’implémentent pas la classe abstraite DAO mais fonctionnent de la même manière.***

Habiter\_DAO : Classe permettant de se connecter à la base de données pour la table Habiter et d'effectuer diverses actions sur la table. Cette classe dispose des méthodes suivantes :

public boolean create(int idBatiment, int idLocataire) : Méthode qui insère le lien entre un batiment et un locataire

public boolean delete(int idBatiment, int idLocataire) : Méthode qui supprime le lien entre un batiment et un locataire

public List<Appartement> getAppartementByIdLocataire(int id) : Méthode qui récupère tous les appartements d'un locataire

public List<Maison> getMaisonByIdLocataire(int id) : Méthode qui récupère toutes les maisons d'un locataire

Recevoir\_DAO : Classe permettant de se connecter à la base de données pour la table Recevoir et d'effectuer diverses actions sur la table. Cette classe dispose des méthodes suivantes :

public boolean create(int idCampagne, List<ListeDeDiffusion> obj) : Méthode qui crée un lien entre une campagne et une liste de diffusion dans la table recevoir

public boolean delete(int idCampagne) : Méthode qui supprime un lien entre une campagne et une liste de diffusion dans la table recevoir

public boolean deleteByListe(int idListe) : Méthode qui supprime un lien entre une campagne et une liste de diffusion dans la table recevoir

public boolean update(int idCampagne, List<ListeDeDiffusion> obj) : Méthode qui met à jour la table recevoir : elle supprime toutes les listes d'une campagne et les ajoute avec la liste à jour

public List<ListeDeDiffusion> getListes(int idCampagne) : Méthode qui recupére une liste de diffusion par un id de campagne

public ArrayList<String> getListeEmails(int idCampagne) : Méthode qui retourne la liste des emails de toutes les listes de diffusion d'une campagne

## Package Exceptions

***Ce package est constitué de cinq exceptions dérivées de la classe Exception. Ces 5 classes disposent de la méthode afficherErreur() qui affiche un message d’erreur(soit dans une popup, soit dans un JLabel).***

EmptyFieldException : Classe dérivée de la classe Exception, cette exception est levée si un champ n'est pas saisi

PasDeCaseCocheeException : Classe dérivée de la classe Exception, cette exception est levée si aucune case n'est cochée

PasDeLignesSelectionneesException : Classe dérivée de la classe Exception, cette exception est levée si aucune ligne n'est sélectionnée

UserNotFoundException : Classe dérivée de la classe Exception, cette exception est levée si l'utilisateur a saisi un login ou un mot de passe incorrect

ValeurIncorrecteException : Classe dérivée de la classe Exception, cette exception est levée si une valeur saisie est incorrecte

## Package Graphique

***Ce package est constitué de deux classes qui permettent simplement d’uniformiser certains éléments.***

Bouton : Classe dérivée de la classe JButton qui permet d'uniformiser tous les boutons

Panneau : Classe dérivée de la classe JPanel qui permet d'uniformiser tous les panneaux

## Package Modeles

***Ce package est constitué des modèles dans l’architecture MVC.***

Modele\_AjoutModif : Classe modèle de l'interface d'ajouts et de modifications. Cette classe dispose des méthodes suivantes :

public <O> void ajouter(O obj) : Méthode qui permet d'appeler la méthode create des objets DAO

public <O> void modifier(O obj) : Méthode qui permet d'appeler la méthode update des objets DAO

public ArrayList<ListeDeDiffusion> retournerListesDeDiffusion() : Méthode qui permet de récupérer les listes de diffusion pour les campagnes

Modele\_Ajout\_Listes : Classe modèle de l'interface d'ajouts et de modifications des listes de diffusion. Cette classe dispose des méthodes suivantes :

public void choisirModele() : Méthode qui définit le modèle en fonction du type de personne

public void modeleLocataires() : Méthode qui permet de générer un tableau des locataires

public void modeleUtilisateurs() : Méthode qui permet de générer un tableau des utilisateurs

public void cocher(int ligne) : Méthode qui permet de cocher une case si elle ne l'est pas et de décocher une case si elle est cochée

public void cocherTout() : Méthode qui permet de cocher toutes les cases

public void decocherTout() : Méthode qui permet de décocher toutes les cases

public void getTri(String categorie, String signe, String nombre) : Méthode qui génère une requête SQL permettant de filtrer avec des valeurs

public void trierPar(String categorie) : Méthode qui génère une requête SQL permettant de trier par catégorie

public void executerRequeteLocataire(String requete) : Méthode qui permet de récupérer des locataires triés grâce à une requête

public void executerRequeteUtilisateur(String requete) : Méthode qui permet de récupérer des utilisateurs triés grâce à une requête

public void ajouter(String nom) : Méthode qui permet de créer une liste de diffusion

public void modifier(ListeDeDiffusion listeDiffusion) : Méthode qui permet de modifier une liste de diffusion

private ArrayList<Personne> convertirEnListePersonnes() : Méthode qui permet de convertir la liste d'id en liste de personnes

Modele\_Association : Classe modèle de l'interface permettant d'associer un logement et un locataire. Cette classe dispose des méthodes suivantes :

public void insertHabiter(int idBatiment, Locataire loca) : Méthode qui permet de définir qu'un locataire a un logement

public void removeHabiter(int idLogement, Locataire loca) : Méthode qui permet de définir qu'un locataire n'habite plus dans un logement

Modele\_Connexion : Classe modèle de l'interface de connexion. Cette classe dispose de la méthode suivante :

public Utilisateur trouverUser(String login, String mdp, JLabel message) : Méthode qui permet de vérifier qu'un utilisateur a bien saisi des informations correctes

Modele\_Gestion : Classe modèle de l'interface de gestion des données. Cette classe dispose des méthodes suivantes :

public void initialiser() : Méthode qui permet de définir l'objet dao et de définir le tableau des données en fonction du type des données

public void modeleLocataires() : Méthode qui permet de définir le tableau des locataires

public void modeleUtilisateurs() : Méthode qui permet de définir le tableau des utilisateurs

public void modeleMaison() : Méthode qui permet de définir le tableau des maisons

public void modeleAppartement() : Méthode qui permet de définir le tableau des appartements

public void modeleCampagne() : Méthode qui permet de définir le tableau des campagnes

public void modeleListe() : Méthode qui permet de définir le tableau des listes de diffusion

public void insererViaCSV(List liste\_csv) : Méthode qui permet d'insérer des données via CSV

public void supprimer(int n) : Méthode qui permet de supprimer un élément

Modele\_dialog\_locataire : Classe du modèle de l'interface permettant de visualiser les campagnes dont un locataire fait partie. Cette classe dispose de la méthode suivante :

public ArrayList<Campagne> getListeCampagne(int idLocataire) : Méthode qui recherche les campagnes dont un locataire fait partie

## Package Objets\_Locatis

***Ce package est constitué des objets principaux.***

Appartement : Classe dérivée de Batiment décrivant un appartement.

Batiment : Classe abstraite décrivant les logements.

Campagne : Classe décrivant une campagne.

ListeDeDiffusion : Classe décrivant une liste de diffusion de personnes. Cette classe dispose des méthodes suivantes :

public String getTypeListe() : Méthode qui détermine si la liste est une liste de locataires ou une liste d'utilisateurs

public ArrayList<Integer> getListeId() : Méthode qui retourne une liste des id des personnes de la liste de diffusion

Locataire : Classe dérivée de Personne décrivant un locataire. Cette classe dispose de la méthode suivante :

public int getAge() : Méthode qui calcule l’âge du locataire à partir de sa date de naissance

Mailer : Classe qui permet d’envoyer un mail. Cette classe dispose de la méthode suivante :

public void sendEmail(String mailSender, String passwordSender, String object, String message, ArrayList<String> listeMailTo) : Méthode qui envoie un mail à une liste de personnes

Maison : Classe dérivée de Batiment décrivant un appartement.

MyDate : Classe décrivant une date. Cette classe dispose des méthodes suivantes :

public String getDateSQL() : Méthode qui permet d’avoir une chaine normalisé pour l'insertion en base de données

public String getDateEcrite() : Méthode qui retranscrit l'objet MyDate en date écrite

MyTime : Classe décrivant un temps. Cette classe dispose de la méthode suivante :

public String getDateSQL() : Méthode qui normalise un objet MyTime en string pour l'insérer en SQL

Personne : Interface personne. Une personne est un locataire ou un utilisateur.

PieChart : Classe dérivée de ApplicationFrame qui permet de faire des graphiques en camembert. Cette classe dispose des méthodes suivantes :

public PieDataset createDataset() : Méthode qui calcule le pourcentage selon le type de donnée de l'objet

public JFreeChart createChart( PieDataset dataset) : Méthode qui crée le graphique visuellement

public JPanel createDemoPanel() : Méthode qui lance la création du graphique

SurveillanceCampagne : Classe permettant de surveiller les campagnes en cours et d'envoyer les mails correspondants. Cette classe dispose des méthodes suivantes :

public void envoyer(Campagne cmp) : Méthode qui envoie les emails d'une campagne selon l'heure

public void run() : Méthode qui lance la surveillance des campagnes pour l'envoi d'emails

Utilisateur : Classe dérivée de Personne qui représente un Utilisateur.

## Package Popups

***Ce package est constitué de deux popups qui permettent d’afficher des messages.***

PopupInformation : Classe qui affiche une popup avec un message d'information.

PopupSuppression : Classe qui affiche une popup pour la suppression. Cette classe dispose de la méthode suivante :

public boolean getConfirmation() : Méthode qui retourne le choix de l'utilisateur.

## Package Vues

***Ce package est constitué des vues dans l’architecture MVC.***

Vue\_AjoutModif : Interface qui décrit les vues permettant d'ajouter ou de modifier un élément. Cette classe est implémentée par Vue\_AjoutModif\_Appartement, Vue\_AjoutModif\_Campagne, Vue\_AjoutModif\_Locataires, Vue\_AjoutModif\_Maison, Vue\_AjoutModif\_Utilisateurs. Cette interface dispose des méthodes suivantes :

public void afficherVue() : Méthode qui permet de rendre la vue visible

public void ajouterEcouteurBouton(String nomBouton, ActionListener listener) : Méthode qui permet d'ajouter des écouteurs à un bouton

public void verifierChamps() throws EmptyFieldException, PasDeLignesSelectionneesException,

ValeurIncorrecteException : Méthode qui permet de vérifier que tous les champs sont bien saisis et valides

public void reset() : Méthode qui permet de réinitialiser tous les champs

public void quitter() : Méthode qui permet de fermer la fenêtre

public Object getNouvelObjet() : Méthode qui retourne l'objet à ajouter

public Object getObjetModifie() : Méthode qui retourne l'objet à modifier

Vue\_AjoutModif\_Appartement : Classe implémentant l'interface Vue\_AjoutModif et qui décrit la vue permettant d'ajouter ou de modifier un appartement.

Vue\_AjoutModif\_Campagne : Classe implémentant l'interface Vue\_AjoutModif et qui décrit la vue permettant d'ajouter ou de modifier une campagne. Cette classe dispose en plus des méthodes suivantes :

public void remplirListe(ArrayList<ListeDeDiffusion> uneListe) : Méthode qui permet de remplir la JList avec les listes de diffusion

public void remplirComboBox(JComboBox comboBox, int min, int max) : Méthode qui permet de remplir une JComboBox avec des nombres entiers

public void selectionnerListes() : Méthode qui permet de sélectionner les listes de diffusion d'une campagne

Vue\_AjoutModif\_Locataires : Classe implémentant l'interface Vue\_AjoutModif et qui décrit la vue permettant d'ajouter ou de modifier un locataire.

Vue\_AjoutModif\_Maison : Classe implémentant l'interface Vue\_AjoutModif et qui décrit la vue permettant d'ajouter ou de modifier une maison.

Vue\_AjoutModif\_Utilisateurs : Classe implémentant l'interface Vue\_AjoutModif et qui décrit la vue permettant d'ajouter ou de modifier un utilisateur. Cette classe dispose en plus des méthodes suivantes :

public void afficherTableau() : Méthode qui permet d’afficher un tableau décrivant les catégories des catégories et leurs droits

public void ajouterCase(String s) : Méthode qui permt d'ajouter une case au tableau

Autres vues :

Vue\_Ajout\_Listes : Classe dérivée de JFrame qui décrit la vue permettant d'ajouter ou de modifier une liste de diffusion. Cette classe dispose des méthodes suivantes :

public void definirTableau(Object[][] donnees, String[] entetes) : Méthode qui permet de créer un nouveau tableau de personnes

public void changerTableau(Object[][] donnees, String[] entetes) : Méthode qui permet d'effacer le tableau et d'en ajouter un nouveau

public void afficherPanneauBoutonsRadios() : Méthode qui permet d'afficher ou d'enlever le panneau des boutons radios permettant de filtrer les données en fonction de certains paramètres

public void remplirComboBox : Méthode qui permet de remplir la JComboBox avec les différentes catégories en fonction du type de personne

Vue\_Association : Classe dérivée de JFrame qui décrit la vue permettant d'ajouter ou d'enlever une association entre un locataire et un logement.

Vue\_Connexion : Classe dérivée de JFrame qui décrit la vue permettant de se connecter.

Vue\_Gestion : Classe de l'objet Vue\_Gestion, l'objet fait l'affichage de la gestion des locataires, utilisateurs, maisons, appartements, campagnes et listes de diffusion. Cette classe dispose des méthodes suivantes :

public void definirTableau(String[][] donnees, String[] entetes) : Méthode qui permet de créer un nouveau tableau de données

public void changerTableau(String[][] donnees, String[] entetes) : Méthode qui permet d'effacer le tableau et d'en ajouter un nouveau

public void verifierSelection() throws PasDeLignesSelectionneesException : Méthode qui permet de vérifier que tous les champs sont bien saisis

Vue\_Menu : Classe dérivée de JFrame qui décrit la vue permettant d'afficher le menu principal. Cette classe dispose des méthodes suivantes :

Public void menuUtilisateurs1() : Méthode qui ajoute les boutons en fonction des droits des utilisateurs dont la catégorie est "Utilisateur 1"

Public void menuUtilisateurs2() : Méthode qui ajoute les boutons en fonction des droits des utilisateurs dont la catégorie est "Utilisateur 2"

Public void menuGestionnaires() : Méthode qui ajoute les boutons en fonction des droits des utilisateurs dont la catégorie est "Gestionnaire" (1,2 et 3)

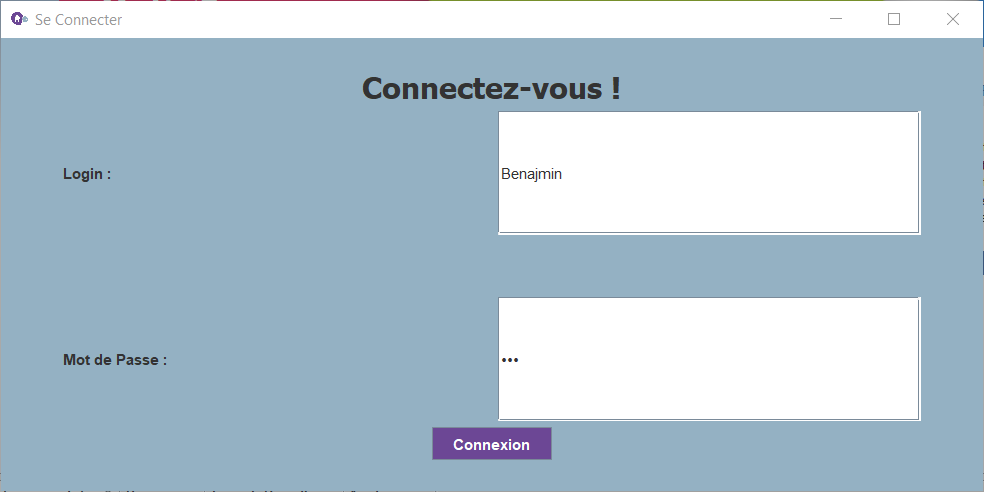
Public void menuAdministrateurs() : Méthode qui ajoute les boutons en fonction des droits des utilisateurs dont la catégorie est "Administrateur"

Vue\_Statistique : Classe dérivée de JFrame qui décrit la vue permettant d'afficher les statistiques

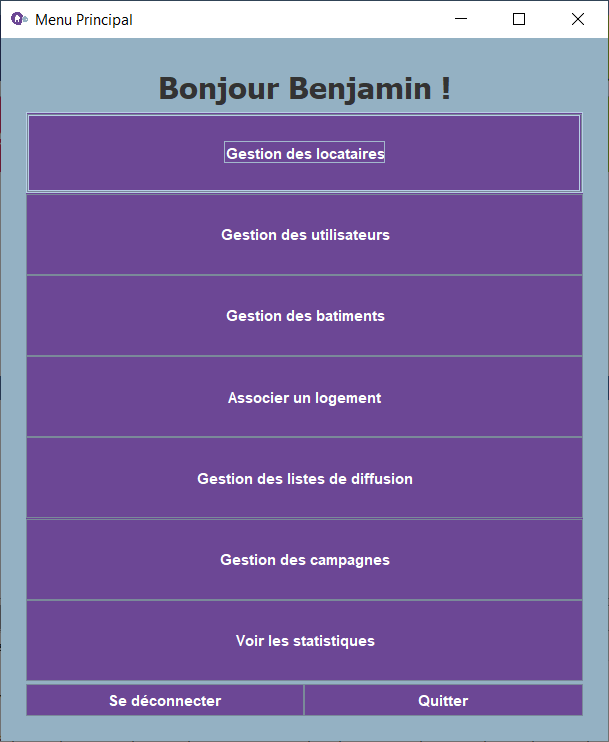
Vue\_dialog\_locataire : Classe dérivée de JDialog qui décrit la vue permettant visualiser les campagnes dont un locataire fait partie

# Les principales fonctionnalités applicatives et leur mode d’emploi

Pour accéder à l’application, il faut tout d’abord se connecter en saisissant son login et son mot de passe. Si l’un des champs n’est pas saisi, une popup d’erreur s’affiche et si l’un des deux champs n’est pas correcte un message d’erreur s’affiche.



Après s’être connecté, la fenêtre du menu principal s’affiche. Ce menu sera différent selon la catégorie de l’utilisateur :

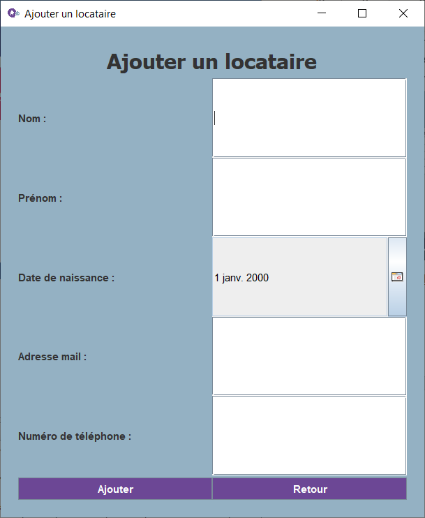


## Gestion des locataires

Pour visualiser les locataires, il faut cliquer sur « Gestion des locataires ». La fenêtre avec la liste des locataires et leurs informations s’affiche. Il sera possible de rechercher des locataires, en ajouter, en modifier, en supprimer et même en insérer via un fichier CSV.

Une image contenant table

Description générée automatiquement

Ajout et modification d’un locataire :

Pour ajouter ou modifier un locataire, il faut cliquer sur le bouton correspondant et remplir tous les champs dans la nouvelle fenêtre qui s’ouvre.

Suppression d’un locataire :

Pour supprimer un locataire, il suffit de sélectionner un locataire et de cliquer sur « Supprimer ». Une popup apparaît pour confirmer son choix.

Insertion d’un locataire :

Pour insérer des locataires, il faut cliquer sur le bouton « Insérer » et choisir le fichier CSV.

Visualisation des campagnes d’un locataire :

On peut visualiser les campagnes d’un locataire en double-cliquant sur celui-ci.

## Gestion des utilisateurs

Le principe de gestion des utilisateurs est le même que celui des locataires. Les champs à remplir lors de la création ou la modification d’un locataire sont différents : il faut saisir un login, un mot de passe, sélectionner une catégorie (la description des catégories est fournie sur le côté), saisir l’adresse email et le password d’une boîte mail pour permettre d’envoyer des mails.

Une image contenant table

Description générée automatiquement

## Gestion des logements

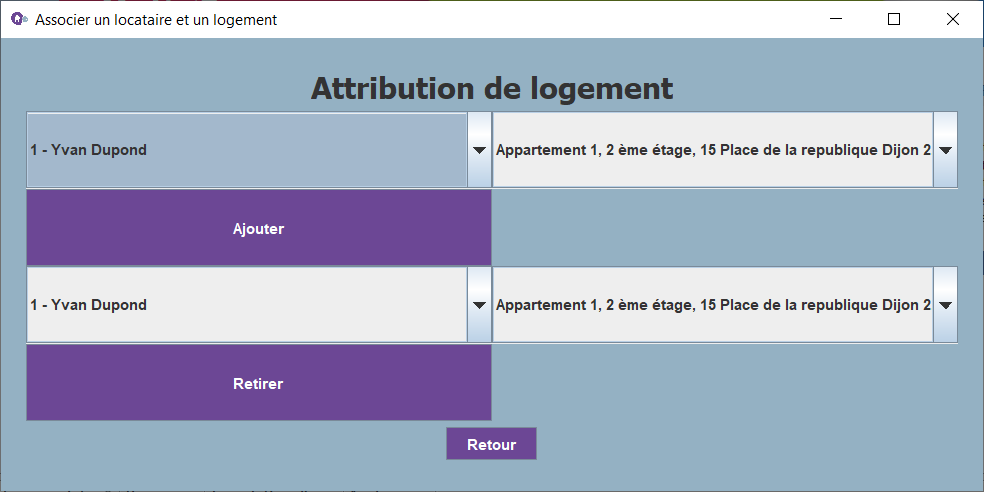
Le principe de gestion des logements est le même que celui des locataires. Cependant, les logements peuvent être des appartements ou bien des maisons. Pour cela, il y a les boutons radios qui permettent de choisir si on veut s’occuper des appartements ou bien des maisons.

Une image contenant texte, table

Description générée automatiquement

## Association des locataires et des logements

La première liste déroulante concerne les locataires et la seconde les logements. La partie du haut sert à ajouter une association et celle du bas à en retirer.



## Gestion des listes de diffusion

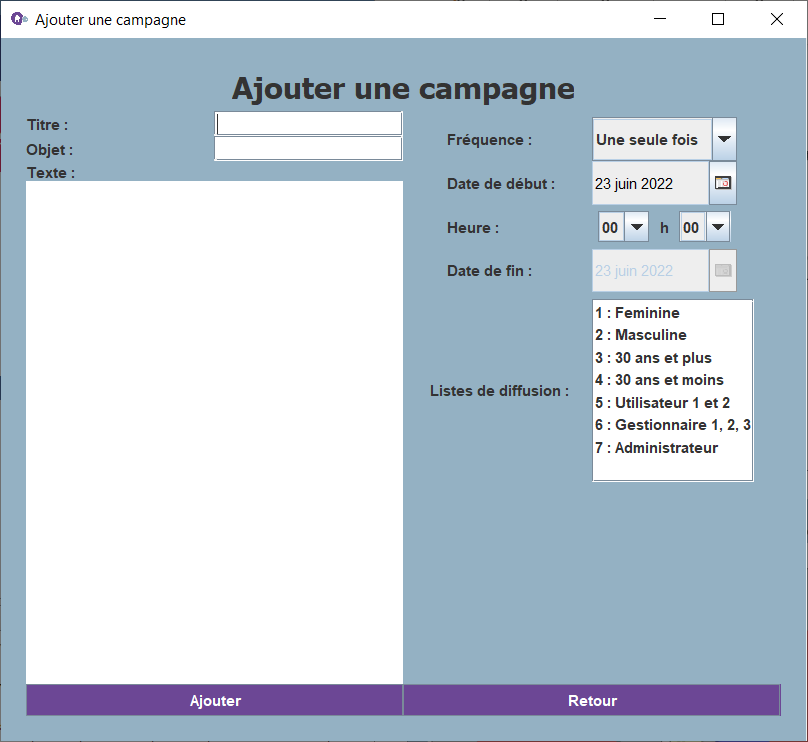
Le principe de gestion des listes de diffusion est le même que celui des locataires. La création d’une liste est différente : on peut choisir de faire une liste de locataires ou une liste d’utilisateurs mais pas les deux. La liste des locataires / utilisateurs s’affiche. A partir de cette liste on peut effectuer une recherche, on peut trier les personnes par catégories et même les trier de manière plus poussée avec l’âge et la date de naissance. On peut aussi sélectionner toutes les personnes affichées ou les désélectionner.

Une image contenant table

Description générée automatiquement

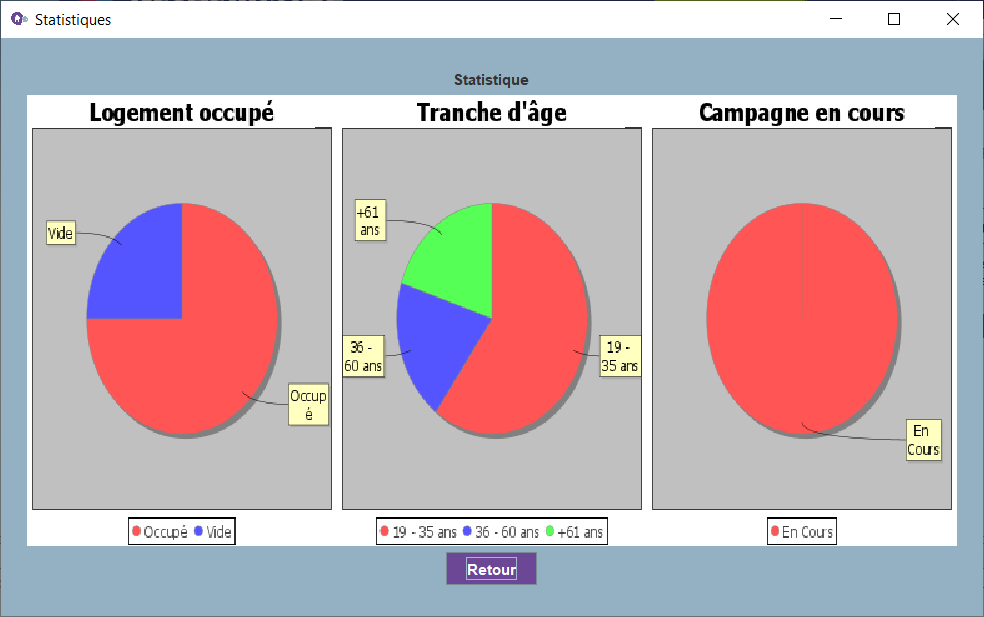
## Gestion des campagnes

La gestion des campagnes est également similaire. Pour créer une campagne, il faudra saisir un titre, l’objet des mails, le contenu des mails, la fréquence de l’envoi des mails, une date de début, une date de fin, une heure et une liste de listes de diffusion.



## Voir les statistiques

Il y a trois types de statistiques : le pourcentage de logements occupés, les tranches d’âge des locataires et l’état actuel des campagnes.



# Schéma de la base de données

