

Documentation technique de l'application GSB Java/Swing



Sommaire

1.	AVANT-PROPOS	3
1.1	Présentation	3
1.2	Contexte	3
1.3	Fonctionnalités	3
2.	BASE DE DONNEES	3
2.1	Présentation	3
2.2	MCD	4
3.	APPLICATION GSB JAVA	4
3.1	Présentation	4
3.2	Diagramme de classe UML	5
3.3	Détail du contrôleur	6
3.4	Détail modèle, JDBC et classes métiers	6
3.5	Détail des vues	10
4.	TESTS FONCTIONNELS	12
4.1	Affichage des visiteurs et des compte-rendus	12

1. AVANT-PROPOS

1.1 Présentation

Ce document a pour objectif d'expliquer les grands axes de l'application GSB Java/Swing. Ainsi serons expliqués dans ce document : la structure de la base de données, la structure de l'application et son fonctionnement.

1.2 Contexte

Galaxy Swiss Gourdin est un laboratoire. Son application doit permettre de consulter les informations concernant les différents visiteurs, les compte-rendus rédigés par les visiteurs et les praticiens qualifiés « d'hésitants » après la visite des visiteurs. L'application sera codée avec le langage de programmation JAVA SWING et utilisera une base de données Mysql.

1.3 Fonctionnalités

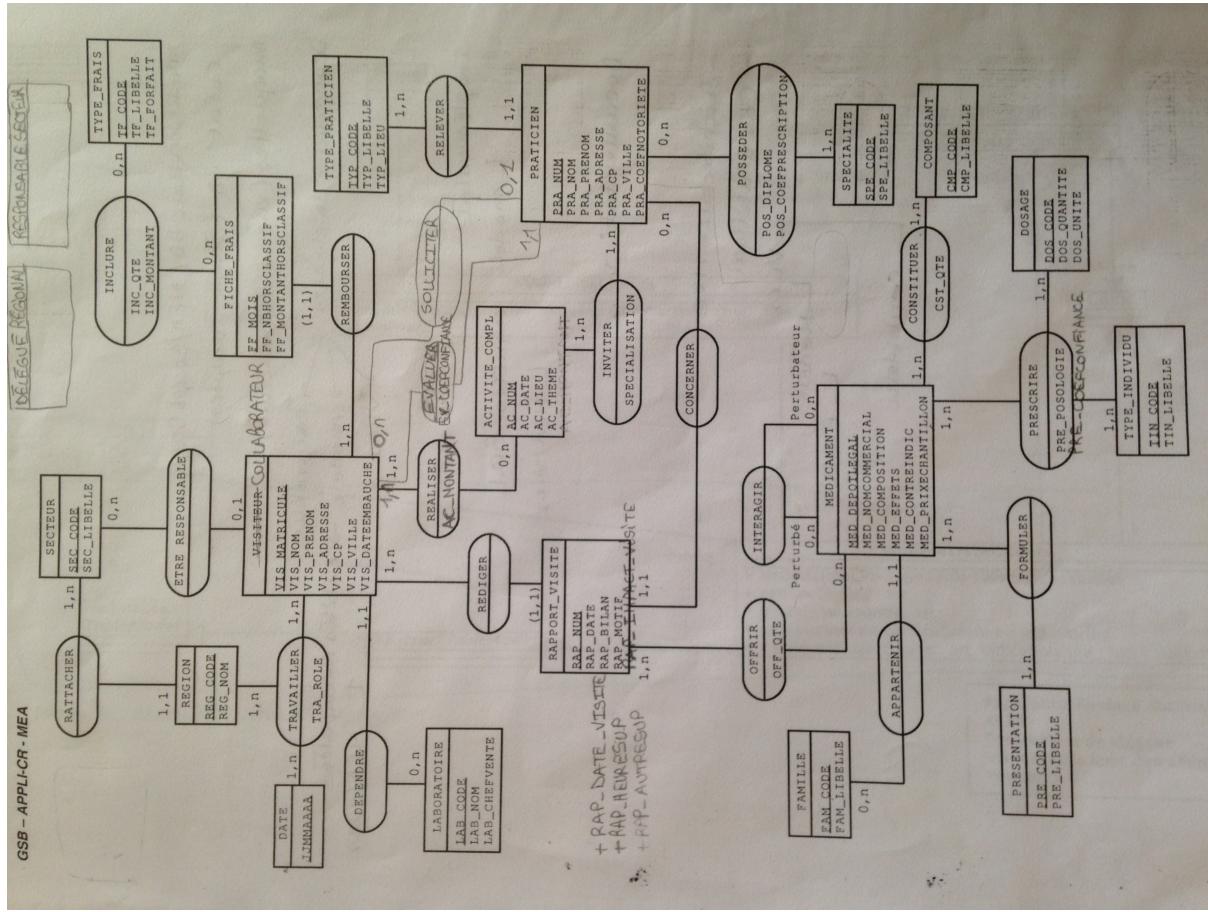
Cette documentation concerne les fonctionnalités d'affichage des visiteurs, des comptes-rendus et des praticiens hésitants.

2. BASE DE DONNEES

2.1 Présentation

La base de données est déployée sous Mysql. Le script nous a été transmis au début du projet.

2.2 MCD



3. APPLICATION GSB JAVA

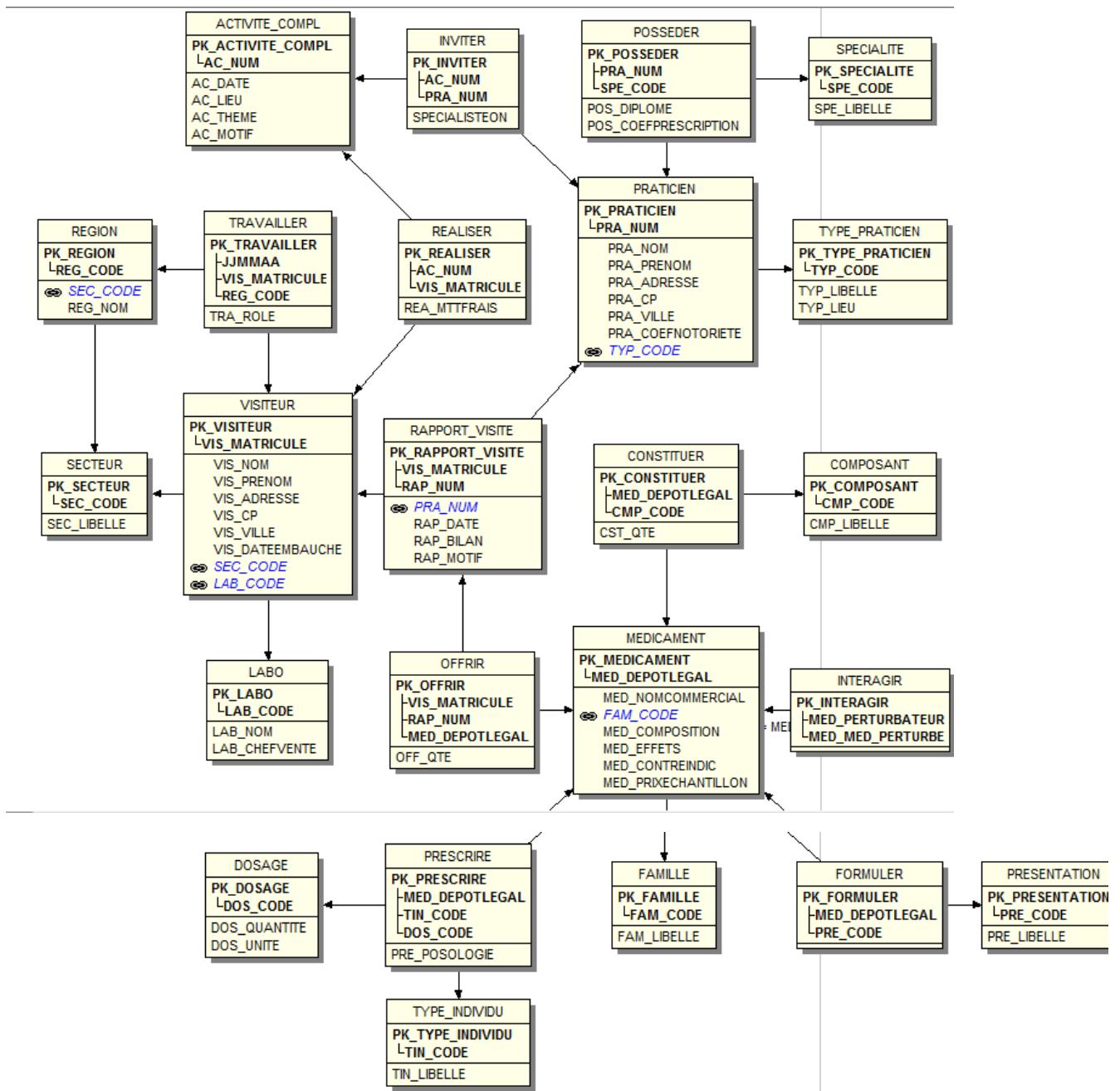
3.1 Présentation

Cette application a été réalisée avec le modèle MVC. C'est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

Ce paradigme regroupe les fonctions nécessaires en trois catégories :

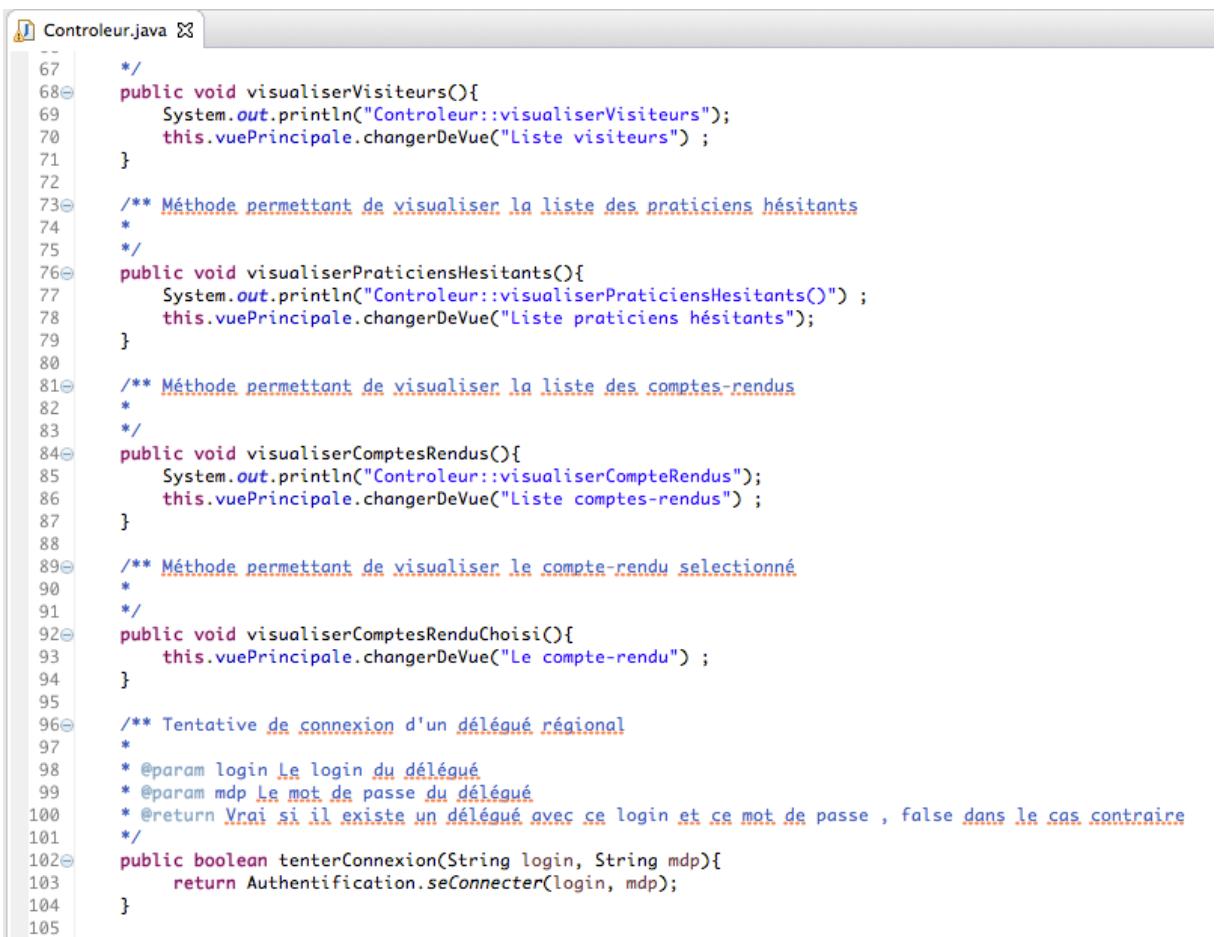
1. un modèle (les informations de la base de données),
 2. une vue (présentation, interface utilisateur)
 3. un contrôleur (logique de contrôle, gestion des événements, synchronisation)

3.2 Diagramme de classe UML



3.3 Détail du contrôleur

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle, et ce dernier informe la vue que les données ont changé pour qu'elle se mette à jour.



```
Controleur.java
-- 67  /*
68  *   public void visualiserVisiteurs(){
69  *       System.out.println("Controleur::visualiserVisiteurs");
70  *       this.vuePrincipale.changerDeVue("Liste visiteurs") ;
71  *   }
72
73  /** Méthode permettant de visualiser la liste des praticiens hésitants
74  *
75  */
76  public void visualiserPraticiensHesitants(){
77  *   System.out.println("Controleur::visualiserPraticiensHesitants()");
78  *   this.vuePrincipale.changerDeVue("Liste praticiens hésitants");
79  }
80
81  /** Méthode permettant de visualiser la liste des comptes-rendus
82  *
83  */
84  public void visualiserComptesRendus(){
85  *   System.out.println("Controleur::visualiserCompteRendus");
86  *   this.vuePrincipale.changerDeVue("Liste comptes-rendus");
87  }
88
89  /** Méthode permettant de visualiser le compte-rendu sélectionné
90  *
91  */
92  public void visualiserComptesRenduChoisi(){
93  *   this.vuePrincipale.changerDeVue("Le compte-rendu");
94  }
95
96  /** Tentative de connexion d'un délégué régional
97  *
98  * @param login Le login du délégué
99  * @param mdp Le mot de passe du délégué
100 * @return Vrai si il existe un délégué avec ce login et ce mot de passe , false dans le cas contraire
101 */
102 public boolean tenterConnexion(String login, String mdp){
103  *   return Authentification.seConnecter(login, mdp);
104 }
105 }
```

3.4 Détail modèle, JDBC et classes métiers

Le *modèle* représente le cœur (algorithmique) de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application. Il regroupe la gestion de ces données et est responsable de leur intégrité. La base de données sera l'un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, changement de

valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle ne s'occupent pas de la présentation.

Classe	Description
ModeleAppliCR	Modèle de l'application
Visiteur	Entité Visiteur
CompteRendu	Entité CompteRendu
Praticien	Entité Praticien
ConnexionBD	Classe JDBC

```
57     super();
58 }
59
60
61 /**
62 * Récupération de la liste des visiteurs de la région du délégué régional qui s'en occupe
63 *
64 * @return la liste des visiteurs
65 */
66 public List<Visiteur> getVisiteurs(){
67     String requete = "SELECT VISITEUR.VIS_MATRICULE, VISITEUR.VIS_NOM, VISITEUR.VIS_PRENOM, VISITEUR.VIS_ADRESSE, VISITEUR.VI
68
69     try{
70         connexion = ConnexionBD.getConnexion();
71         String visMat = Authentification.getMatric();
72         pstmt = (PreparedStatement) connexion.prepareStatement(requete);
73         pstmt.setString(1,visMat);
74         resultat = pstmt.executeQuery();
75         System.out.println("ModeleAppliCR::getVisiteurs()");
76
77         while(resultat.next()){
78
79             String matricule = resultat.getString("VIS_MATRICULE");
80             String nom = resultat.getString("VIS_NOM") ;
81             String prenom = resultat.getString("VIS_PRENOM") ;
82             String adresse = resultat.getString("VIS_ADRESSE");
83             String cp = resultat.getString("VIS_CP");
84             String ville = resultat.getString("VIS_VILLE");
85             Date dateEmbauche = resultat.getDate("VIS_DATEEMBAUCHE");
86             this.visiteurs.add(new Visiteur(matricule,nom,prenom,adresse,cp,ville,dateEmbauche));
87         }
88     }
89     catch(Exception e){
90         System.out.println("Erreur requête getVisiteurs()");
91         e.printStackTrace();
92     }
93
94     return visiteurs;
95
96 }
```

```

1 CompteRendu.java ☐
11 /**
12  * 
13 
14  private int numero;
15  private Date dateRedac;
16  private Date dateVisite;
17  private String bilan;
18  private String motif;
19  private String nomPra;
20  private String villePra;
21  private String nomVis;
22  private String read ;
23  public static final String NOT_READ = "Non" ;
24  public static final String READ = "Oui" ;
25 
26 
27 /**
28  * @param numero Le numéro
29  * @param dateRedac La date de rédaction
30  * @param dateVisite La date de visite
31  * @param bilan Le bilan
32  * @param motif Le motif
33  * @param nomPra Le nom du praticien
34  * @param villePra La ville du praticien
35  * @param nomVis Le nom du visiteur
36  * @param read L'état du compte-rendu
37  */
38 public CompteRendu(int numero, Date dateRedac, Date dateVisite, String bilan, String read) {
39     this.numero = numero;
40     this.dateRedac = dateRedac;
41     this.dateVisite = dateVisite;
42     this.bilan = bilan;
43     this.motif = motif;
44     this.nomPra = nomPra ;
45     this.villePra = villePra;
46     this.nomVis = nomVis;
47     this.read = read ;
48 }
49

5 Visiteur.java ☐
11 /**
12  * 
13 
14  private String matricule ;
15  private String nom ;
16  private String prenom ;
17  private String adresse ;
18  private String cp ;
19  private String ville ;
20  private Date dateEmbauche ;

21 
22 
23 /**
24  * @param matricule Le matricule
25  * @param nom Le nom
26  * @param prenom Le prénom
27  * @param adresse L'adresse
28  * @param cp Le code postal
29  * @param ville La ville
30  * @param dateEmbauche La date d'embauche
31  */
32 public Visiteur(String matricule, String nom, String prenom, String adresse, String cp, String ville, Date dateEmbauche) {
33     super();
34     this.matricule = matricule ;
35     this.nom = nom ;
36     this.prenom = prenom ;
37     this.adresse = adresse ;
38     this.cp = cp ;
39     this.ville = ville ;
40     this.dateEmbauche = dateEmbauche ;
41 }
42 

43 
44 
45 /**
46  * @return Le matricule du visiteur
47  */
48 public String getMatricule() {
49     return matricule;
50 }
51

```

```

Praticien.java ✘

17 public class Praticien {
18     private int numero ;
19     private String nom ;
20     private String prenom ;
21     private String adresse ;
22     private String cp ;
23     private String ville ;
24     private float coefnotoriete ;
25     private float coefconfiance ;
26     private Date dateVisite ;
27     private long tempsEcoule ;
28     private long dateAujourd'hui = new GregorianCalendar().getTimeInMillis();
29     private String nbJours = "jours";
30
31     /**
32      * @param numero Le numéro
33      * @param nom Le nom
34      * @param prenom Le prénom
35      * @param adresse L'adresse
36      * @param cp Le code postal
37      * @param ville La ville
38      * @param coefnotoriete Le coefficient de notoriété
39      * @param coefconfiance Le coefficient de confiance
40     */
41     public Praticien(int numero, String nom, String prenom, String adresse, String
42                     super();
43     this.numero = numero ;
44     this.nom = nom;
45     this.prenom = prenom ;
46     this.adresse = adresse ;
47     this.cp = cp ;
48     this.ville = ville ;
49     this.coefnotoriete = coefnotoriete ;
50     this.coefconfiance = coefconfiance ;
51     this.dateVisite = dateVisite;
52 }
53
54     /**
55      *
56      */
57
ConnexionBD.java ✘

6 /**
7  * Connexion à la base de données
8  *
9  */
10 public class ConnexionBD {
11
12     private static final String dbURL = "jdbc:mysql://localhost/GsbCRSlam?useUnicode=yes&characterEncoding=UTF-8" ;
13     private static final String user = "root" ;
14     private static final String password = "ndiaye" ;
15     private static Connection connexion = null ;
16
17     /**
18      * @throws ClassNotFoundException
19      * @throws SQLException
20      */
21     public ConnexionBD() throws ClassNotFoundException, SQLException {
22         Class.forName("com.mysql.jdbc.Driver") ;
23         connexion = DriverManager.getConnection(dbURL,user,password) ;
24     }
25
26
27     /**
28      * @return connexion
29      * @throws ClassNotFoundException
30      * @throws SQLException
31      */
32     public static Connection getConnexion() throws ClassNotFoundException, SQLException{
33         if(connexion == null){
34             new ConnexionBD() ;
35         }
36         return connexion ;
37     }
38
39
40
41

```

3.5 Détail des vues

Ce avec quoi l'utilisateur interagit se nomme précisément la *vue*. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toute action de l'utilisateur. Ces différents événements sont envoyés au contrôleur. La vue n'effectue pas de traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

The screenshot shows two views of a web application interface. The top view displays a logo for 'gsb' in a blue oval on a grey background. The bottom view shows a table titled 'Visiteurs :'. The table has columns for Matricule, Nom, Prénom, Adresse, Code postal, Ville, and Date d'embauche. Each row contains a 'Choisir' button in the last column. The data in the table is as follows:

Matricule	Nom	Prénom	Adresse	Code postal	Ville	Date d'embauche	Sélectionner
b19	Bunisset	Francis	10 r Nicolas Chorier	85000	LA ROCHE SUR YON	2013-09-23	Choisir
b25	Bunisset	Denise	1 r Lionne	49100	ANGERS	2007-03-04	Choisir
d13	Debelle	Jeanne	134 r Stalingrad	44000	NANTES	2012-05-25	Choisir
l14	Le	Jean	39 r Raspail	53000	LAVAL	2000-05-01	Choisir
l46	Lecornu	Jean-Bernard	4 bd Mar Foch	72000	LA FERTE BERNARD	1998-11-23	Choisir

○ ○ ○ GSB / Appli-CR

Fichier Comptes-Rendus Aide

Rapports de visite

Numéro	Date de rédaction	Date de visite	Nom du praticien	Ville du praticien	Sélectionner	Consulté
5	2015-01-01	2015-01-01	Goussard	BLOIS	<input type="button" value="Choisir"/>	Non
8	2015-01-01	2015-01-01	Goussard	BLOIS	<input type="button" value="Choisir"/>	Non
23	2015-01-01	2015-01-01	Goussard	BLOIS	<input type="button" value="Choisir"/>	Non

○ ○ ○ GSB / Appli-CR

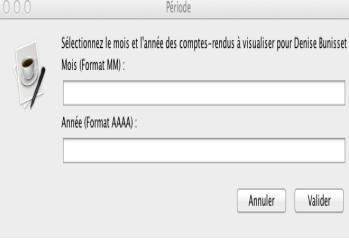
Fichier Comptes-Rendus Aide

Praticiens :

Numero	Nom	Prénom	Adresse	Cp	Ville	Coefficient de no...	Coefficient de co...	Date de visite	Temps éoulé
41	Ain	Jean-Pierre	4 rÈsid Olympia	2000	LAON	5.59	3.0	2014-02-12	438 jours
24	Goussard	Emmanuel	9 r Demolombe	41000	BLOIS	40.72	4.0	2015-01-01	115 jours
24	Goussard	Emmanuel	9 r Demolombe	41000	BLOIS	40.72	4.0	2015-01-01	115 jours
24	Goussard	Emmanuel	9 r Demolombe	41000	BLOIS	40.72	4.0	2015-01-01	115 jours
4	Leroux	AndrÈ	47 av Robert Sc...	60000	BEAUVAIIS	172.4	2.0	2014-03-01	421 jours
23	Flament	Elisabeth	11 r Pasteur	35000	RENNES	315.6	3.0	2013-04-20	736 jours

4. TESTS FONCTIONNELS

4.1 Affichage des visiteurs et des compte-rendus

Jeu d'essai	Résultat attendu	Résultat obtenu	Correction	Résultat après correction
Afficher les visiteurs en cliquant sur l'item « Liste des visiteurs »	Affichage de la liste des visiteurs			
Modifier les informations contenues dans les champs du tableau	Aucun moyen de cliquer	Possibilité de changer le texte	Rendre les champs non-éditables grâce à une méthode <code>isCellEditable</code> qu'on retourne à false dans l'abstract table model	Les champs sont non-éditables
Choisir un compte-rendu du visiteur en cliquant sur le bouton « Choisir »	Affichage d'une fenêtre où l'on doit saisir un mois et une année			
Lire le compte-rendu du visiteur	Affichage des informations du compte-rendu sous forme de JTextArea	