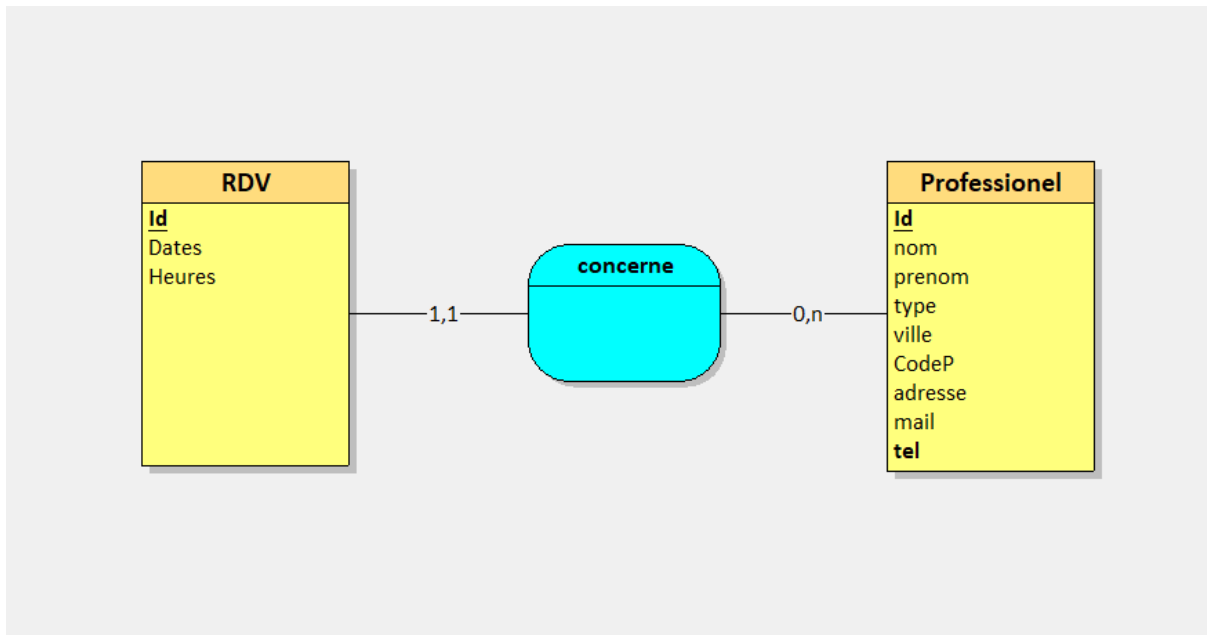


MCD :



MLD :

Professionnel(**id**, nom, prenom, types, ville, codeP, adresse, **tel**)

RDV(**id**, dates, heure, #idR)

Bienvenue

Prendre RDV

Voir Planning

Rechercher

Enregistrer un pro

Avec qui voulez prendre RDV ?

Dr

Heure :

Prendre RDV

Retour

Planning

Voir

Retour

Code Postal

Rechercher

Retour

Nom

Prenom

type

Adresse

Code P

Ville


N° de tel

Enregistrer

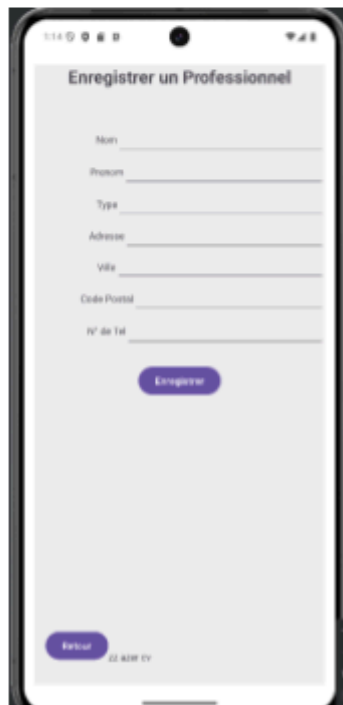
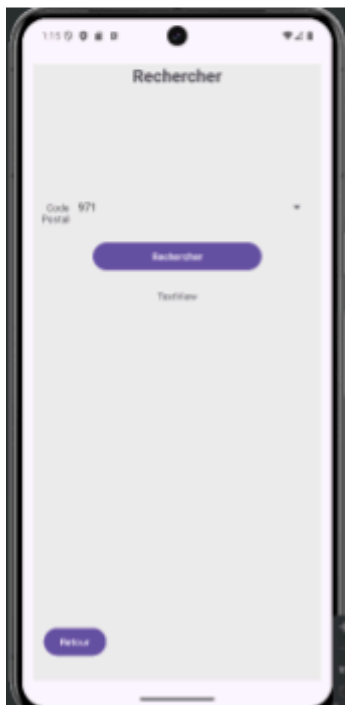
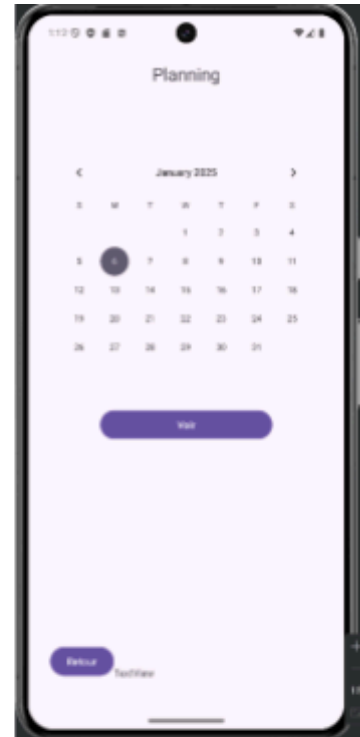
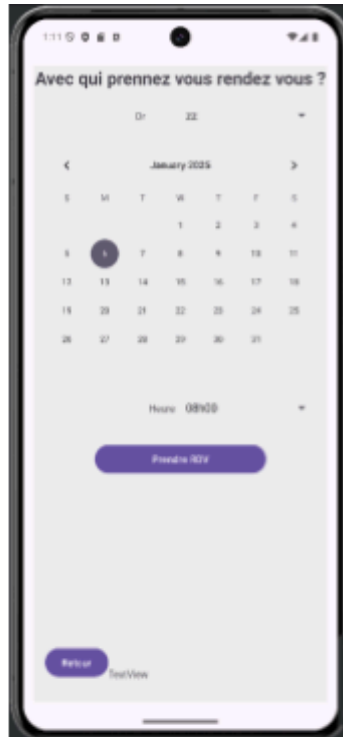
Retour

<b>N °</b>	<b>Valeurs BD avant test</b>	<b>Action</b>	<b>Attendu</b>	<b>Valeurs BD après test</b>	<b>Résultat</b>
1	Aucune	Lancer l'application	Ouverture de la page d'accueil permettant la navigation vers les autres pages	Aucune	Ouverture de la page d'accueil
2	Aucune	Naviguer vers la page d'ajout d'un professionnel	Ouverture de la page d'ajout d'un professionnel	Aucune	Navigation réussie
3	Table : Professionnel	Saisie d'informations valides pour l'ajout d'un professionnel (nom, prénom, type, ville, code postal, adresse, téléphone)	Enregistrement des informations dans la table Professionnel	Table : Professionnel	Enregistrement dans la BD
4	Aucune	Naviguer vers la page de programmation des RDV	Ouverture de la page de programmation des RDV	Aucune	Navigation réussie
5	Table : RDV	Programmer un RDV en entrant une date, une heure et un professionnel valide	Enregistrement des informations du RDV dans la base de données	Table : RDV	Enregistrement dans la BD
6	Table : RDV	Entrer une date ou une heure invalide pour un RDV	Message d'erreur : "La date ou l'heure que vous avez saisie est invalide"	Table : RDV	Affichage d'une erreur
7	Aucune	Naviguer vers la page de consultation du planning	Ouverture de la page d'affichage du planning	Aucune	Navigation réussie

8	Table : RDV	Afficher le planning d'une journée	Liste des RDV prévus pour la journée avec leurs détails (date, heure, professionnel associé)	Aucune	Planning affiché
9	Aucune	Naviguer vers la page de recherche des professionnels	Ouverture de la page de recherche des professionnels	Aucune	Navigation réussie
10	Table : Professionnel	Rechercher des professionnels par ville ou code postal	Affichage de la liste des professionnels correspondant aux critères de recherche	Aucune	Liste affichée

 app-debug.apk

Screen émulateur:



Screens du code :

## Classe BD

```
1 package com.example.projetandroid;
2
3 import android.content.ContentValues;
4 import android.content.Context;
5 import android.database.Cursor;
6 import android.database.sqlite.SQLiteDatabase;
7 import android.database.sqlite.SQLiteOpenHelper;
8
9 /**
10  * Classe BD pour gérer la base de données SQLite dans l'application.
11  * Elle permet la création, mise à jour et manipulation des données dans les tables "PRO" et "RDV".
12  */
13 public class BD extends SQLiteOpenHelper { 10 usages
14
15     // Nom de la base de données
16     public static final String bd = "projetAnd1.db"; 1 usage
17
18     // Table "PRO"
19     public static final String pro = "PRO";
20     public static final String idP = "ID"; no usages
21     public static final String nomP = "NOM"; 1 usage
22     public static final String prenomP = "PRENOM"; 1 usage
23     public static final String type = "TYPE"; 1 usage
24     public static final String adresse = "ADRESSE"; 1 usage
25     public static final String ville = "VILLE"; 1 usage
26     public static final String codeP = "CODEP"; 1 usage
27     public static final String tel = "TEL"; 1 usage
28
29     // Table "RDV"
30     public static final String rdv = "RDV"; 5 usages
31     public static final String idR = "ID"; no usages
32     public static final String dates = "DATES"; 1 usage
33     public static final String heure = "HEURE"; 1 usage
34     public static final String idPro = "IDPRO"; 1 usage
35
36     /**
37      * Constructeur de la classe BD.
38      *
39      * @param context Le contexte de l'application.
40      */
41     > public BD(Context context) { super(context, bd, factory: null, version: 1); }
44
```

```

44
45 /**
46  * Méthode appelée lors de la création initiale de la base de données.
47  * Elle crée les tables "PRO" et "RDV".
48  *
49  * @param db L'instance SQLiteDatabase utilisée pour exécuter les commandes SQL.
50  */
51 @Override
52 public void onCreate(SQLiteDatabase db) {
53     db.execSQL("CREATE table " + pro + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, NOM TEXT, PRENOM TEXT, TYPE TEXT, ADRESSE TEXT, VILLE TEXT, CODEP TEXT, TEL TEXT, " +
54         "IDPRO INTEGER, FOREIGN KEY(IDPRO) REFERENCES PRO(ID))");
55     db.execSQL("CREATE table " + rdv + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, DATES TEXT, HEURE TEXT, " +
56         "IDPRO INTEGER, FOREIGN KEY(IDPRO) REFERENCES PRO(ID))");
57 }
58
59 /**
60  * Méthode appelée lorsqu'une mise à jour de la base de données est nécessaire.
61  * Supprime les anciennes tables et crée de nouvelles versions.
62  *
63  * @param db L'instance SQLiteDatabase utilisée pour exécuter les commandes SQL.
64  * @param oldVersion L'ancienne version de la base de données.
65  * @param newVersion La nouvelle version de la base de données.
66  */
67 @Override no usages
68 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
69     db.execSQL("DROP TABLE IF EXISTS " + pro);
70     db.execSQL("DROP TABLE IF EXISTS " + rdv);
71     onCreate(db);
72 }
73
74 /**
75  * Supprime les tables "PRO" et "RDV".
76  */
77 public void supTable() { no usages
78     SQLiteDatabase db = this.getReadableDatabase();
79     db.execSQL("DROP TABLE IF EXISTS " + pro);
80     db.execSQL("DROP TABLE IF EXISTS " + rdv);
81 }
82
83 /**
84  * Recrée les tables "PRO" et "RDV".
85  */
86 public void creeTable() { no usages
87     SQLiteDatabase db = this.getReadableDatabase();
88     db.execSQL("CREATE table " + pro + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, NOM TEXT, PRENOM TEXT, TYPE TEXT, ADRESSE TEXT, VILLE TEXT, CODEP TEXT, TEL TEXT, " +
89         "IDPRO INTEGER, FOREIGN KEY(IDPRO) REFERENCES PRO(ID))");
90     db.execSQL("CREATE table " + rdv + "(ID INTEGER PRIMARY KEY AUTOINCREMENT, DATES TEXT, HEURE TEXT, " +
91         "IDPRO INTEGER, FOREIGN KEY(IDPRO) REFERENCES PRO(ID))");
92 }
93
94 /**
95  * Enregistre un professionnel dans la table "PRO".
96  *
97  * @param nom Nom du professionnel.
98  * @param prenom Prénom du professionnel.
99  * @param type Type du professionnel.
100  * @param adresse Adresse du professionnel.
101  * @param ville Ville du professionnel.
102  * @param codeP Code postal du professionnel.
103  * @param tel Téléphone du professionnel.
104  */
105 public void enrPro(String nom, String prenom, String type, String adresse, String ville, String codeP, String tel) { 1 usage
106     SQLiteDatabase db = this.getWritableDatabase();
107     ContentValues contentValues = new ContentValues();
108     contentValues.put(this.nomP, nom);
109     contentValues.put(this.prenomP, prenom);
110     contentValues.put(this.type, type);
111     contentValues.put(this.adresse, adresse);
112     contentValues.put(this.ville, ville);
113     contentValues.put(this.codeP, codeP);
114     contentValues.put(this.tel, tel);
115     db.insert(pro, nullColumnHack, null, contentValues);
116     db.close();
117 }
118
119 /**
120  * Crée un rendez-vous dans la table "RDV".
121  *
122  * @param dates Date du rendez-vous.

```

```

123      * @param heure Heure du rendez-vous.
124      * @param idP ID du professionnel lié au rendez-vous.
125      */
126      public void CreerRdv(String dates, String heure, int idP) { 1 usage
127          SQLiteDatabase db = this.getWritableDatabase();
128          ContentValues contentValues = new ContentValues();
129          contentValues.put(this.dates, dates);
130          contentValues.put(this.heure, heure);
131          contentValues.put(this.idPro, idP);
132          db.insert(rdv, nullColumnHack: null, contentValues);
133          db.close();
134      }
135
136      /**
137       * Récupère les noms et prénoms de tous les professionnels.
138       *
139       * @return Un curseur contenant les résultats.
140       */
141      public Cursor getDoc() { 1 usage
142          SQLiteDatabase db = this.getReadableDatabase();
143          return db.rawQuery(sql: "SELECT NOM, PRENOM FROM " + pro, selectionArgs: null);
144      }
145
146      /**
147       * Récupère toutes les données de la table "PRO".
148       *
149       * @return Un curseur contenant les résultats.
150       */
151      public Cursor getAllData() { 1 usage
152          SQLiteDatabase db = this.getReadableDatabase();
153          return db.rawQuery(sql: "SELECT * FROM " + pro, selectionArgs: null);
154      }
155
156      /**
157       * Récupère les dates et heures des rendez-vous.
158       *
159       * @return Un curseur contenant les résultats.
160       */
161      public Cursor getRDV1() { 1 usage

```



```

162     SQLiteDatabase db = this.getReadableDatabase();
163     return db.rawQuery(sql: "SELECT DATES, HEURE FROM RDV", selectionArgs: null);
164 }
165
166 /**
167  * Récupère les rendez-vous pour une date donnée, avec le nom et prénom du professionnel associé.
168  *
169  * @param date La date recherchée.
170  * @return Un curseur contenant les résultats.
171  */
172 public Cursor getRDV(String date) { 2 usages
173     SQLiteDatabase db = this.getReadableDatabase();
174     return db.rawQuery(sql: "SELECT DATES, HEURE, NOM, PRENOM FROM RDV JOIN PRO ON RDV.IDPRO=PRO.ID WHERE DATES LIKE " + date + "%", selectionArgs: null);
175 }
176
177 /**
178  * Récupère tous les codes postaux des professionnels.
179  *
180  * @return Un curseur contenant les résultats.
181  */
182 public Cursor getCode() { 1 usage
183     SQLiteDatabase db = this.getReadableDatabase();
184     return db.rawQuery(sql: "SELECT CODEP FROM PRO", selectionArgs: null);
185 }
186
187 /**
188  * Récupère les noms des professionnels pour un code postal donné.
189  *
190  * @param code Le code postal recherché.
191  * @return Un curseur contenant les résultats.
192  */
193 public Cursor getNomDoc(String code) { 1 usage
194     SQLiteDatabase db = this.getReadableDatabase();
195     return db.rawQuery(sql: "SELECT NOM FROM PRO WHERE CODEP LIKE " + code + "%", selectionArgs: null);
196 }
197
198 /**
199  * Récupère l'ID d'un professionnel à partir de son nom.
200  *

```

```

201  * @param nom Le nom du professionnel.
202  * @return L'ID du professionnel ou une chaîne vide si non trouvé.
203  */
204 public String getIdDoc(String nom) { 1 usage
205     SQLiteDatabase db = this.getReadableDatabase();
206     Cursor result = db.rawQuery(sql: "SELECT ID FROM " + pro + " WHERE NOM LIKE " + nom + "%", selectionArgs: null);
207     if (result != null && result.moveToFirst()) {
208         return result.getString(0);
209     }
210     return "";
211 }
212 }
213

```

## Mainactivity.java

```

1 package com.example.projetandroid;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import androidx.activity.EdgeToEdge;
6 import androidx.appcompat.app.AppCompatActivity;
7 import androidx.core.graphics.Insets;
8 import androidx.core.view.ViewCompat;
9 import androidx.core.view.WindowInsetsCompat;
10 import android.view.View;
11
12 /**
13  * Activité principale de l'application.
14  * Gère les interactions avec la page d'accueil et la navigation vers d'autres activités.
15  */
16 <> public class MainActivity extends AppCompatActivity {
17
18     /**
19      * Instance de la classe BD pour gérer la base de données.
20      */
21     BD bd; 1 usage
22
23     /**
24      * Méthode appelée lors de la création de l'activité.
25      * Configure l'interface utilisateur et initialise la base de données.
26      *
27      * @param savedInstanceState État sauvegardé de l'activité (si disponible).
28      */
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         EdgeToEdge.enable(this); // Active le mode Edge-to-Edge pour une meilleure intégration visuelle.
33         setContentView(R.layout.activity_main);
34
35         // Configure les marges pour prendre en compte les barres système (status bar et navigation bar).
36         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
37             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
38             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
39             return insets;
40         });
41

```

```

42      // Initialisation de la base de données.
43      bd = new BD(context: this);
44
45
46      // Supprime les tables existantes. bd.supTable();
47      // Recrée les tables. bd.creeTable();
48
49  }
50
51  /**
52   * Ouvre la page pour prendre un rendez-vous.
53   *
54   * @param view La vue associée à l'action (bouton ou autre).
55   */
56  public void pageRDV(View view) { 1 usage
57      Intent intent = new Intent(packageContext: this, PrendreRdv.class);
58      startActivity(intent);
59      finish(); // Termine l'activité actuelle.
60  }
61
62  /**
63   * Ouvre la page pour consulter le planning.
64   *
65   * @param view La vue associée à l'action (bouton ou autre).
66   */
67  public void pagePlanning(View view) { 1 usage
68      Intent intent = new Intent(packageContext: this, Planning.class);
69      startActivity(intent);
70      finish(); // Termine l'activité actuelle.
71  }
72
73  /**
74   * Ouvre la page pour effectuer une recherche.
75   *
76   * @param view La vue associée à l'action (bouton ou autre).
77   */
78  public void pageRecherche(View view) { 1 usage
79      Intent intent = new Intent(packageContext: this, Recherche.class);
80      startActivity(intent);
81      finish(); // Termine l'activité actuelle.


```

```

81      finish(); // Termine l'activité actuelle.
82  }
83
84  /**
85   * Ouvre la page pour enregistrer un professionnel.
86   *
87   * @param view La vue associée à l'action (bouton ou autre).
88   */
89  public void pageEnregistrer(View view) { 1 usage
90      Intent intent = new Intent(packageContext: this, Enregistre.class);
91      startActivity(intent);
92      finish(); // Termine l'activité actuelle.
93  }
94  }
95

```

## Recherche.java

```
1 package com.example.projetandroid;
2
3 > import ...
4
19
20 /**
21  * Activité Recherche : permet à l'utilisateur de rechercher des informations
22  * basées sur un code postal sélectionné dans un spinner.
23  */
24 <> public class Recherche extends AppCompatActivity {
25     /**
26      * Spinner pour afficher les codes postaux.
27      */
28     Spinner spinCode; 4 usages
29
30     /**
31      * Instance de la base de données.
32      */
33     BD bd; 3 usages
34
35     /**
36      * Liste contenant les codes postaux récupérés depuis la base de données.
37      */
38     ArrayList<String> listeCode; 5 usages
39
40     /**
41     *  Code postal sélectionné par l'utilisateur.
42     */
43     String code1; 1 usage
44
45     /**
46      * TextView pour afficher la liste des documents correspondants au code postal sélectionné.
47      */
48     TextView liste; 4 usages
49
50     /**
51      * Méthode appelée à la création de l'activité.
52      * Initialise les composants graphiques, récupère les données de la base,
53      * et configure le spinner.
54      *
55      * @param savedInstanceState Instance sauvegardée de l'état de l'activité.
56      */
57     @Override
```

```

58  protected void onCreate(Bundle savedInstanceState) {
59      super.onCreate(savedInstanceState);
60      EdgeToEdge.enable(this);
61      setContentView(R.layout.activity_recherche);
62
63      ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
64          Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
65          v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
66          return insets;
67      });
68
69      bd = new BD(context);
70      liste = (TextView) findViewById(R.id.liste);
71
72      listeCode = new ArrayList<>();
73
74      // Récupération des codes postaux depuis la base de données.
75      Cursor cursor = bd.getCode();
76      if (cursor != null && cursor.moveToFirst()) {
77          do {
78              int indexCP = cursor.getColumnIndex("CODEP");
79
80              if (indexCP >= 0) {
81                  String nom = cursor.getString(indexCP);
82                  listeCode.add(nom);
83              }
84          } while (cursor.moveToNext());
85          cursor.close();
86      } else {
87          listeCode.add("Aucun code ");
88      }
89
90      spinCode = findViewById(R.id.spinnerCodeP);
91
92      // Configuration de l'adaptateur pour le spinner.
93      ArrayAdapter<String> code = new ArrayAdapter<>(context, android.R.layout.simple_spinner_item, listeCode);
94      code.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
95      spinCode.setAdapter(code);
96

```

```

97 // Gestion de la sélection dans le spinner.
98 spinCode.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
99     @Override no usages
100     public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
101         code1 = listeCode.get(i);
102     }
103
104     @Override no usages
105     public void onNothingSelected(AdapterView<?> adapterView) {
106     }
107 });
108 }
109
110 /**
111  * Méthode appelée lors du clic sur le bouton "Voir Doc".
112  * Met à jour la liste des documents en fonction du code postal sélectionné.
113  *
114  * @param view Vue du bouton "Voir Doc".
115  */
116 public void VoirDoc(View view) { 1 usage
117     maj();
118 }
119
120 /**
121  * Met à jour le TextView avec les documents associés au code postal sélectionné.
122  * Les données sont récupérées depuis la base de données.
123  */
124 public void maj() { 1 usage
125     liste.setText("");
126     try {
127         Cursor data = bd.getNomDoc(spinCode.getSelectedItem().toString());
128         String texte = "";
129         while (data.moveToNext()) {
130             texte = texte + String.valueOf(data.getString(1));
131         }
132         liste.setText(texte);

```

```

133     } catch (Exception e) {
134         liste.setText(e.getMessage());
135     }
136 }
137
138 /**
139  * Méthode appelée lors du clic sur le bouton "Retour".
140  * Retourne à l'activité principale.
141  *
142  * @param view Vue du bouton "Retour".
143  */
144 public void retour(View view) { 4 usages
145     Intent intent = new Intent(packageContext: this, MainActivity.class);
146     startActivity(intent);
147 }
148 }
149

```

## Prendrerdv.java

```
1 package com.example.projetandroid;
2
3 > import ...
4
5 /**
6  * Activité PrendreRdv : permet à l'utilisateur de prendre un rendez-vous
7  * avec un docteur en choisissant une date, une heure et un docteur disponible.
8  */
9
10 public class PrendreRdv extends AppCompatActivity {
11     /**
12      * Instance de la base de données.
13      */
14     BD bd; 5 usages
15
16     /**
17      * TextView pour afficher la liste des rendez-vous enregistrés.
18      */
19     TextView liste; 4 usages
20
21     /**
22      * Spinner pour sélectionner un docteur.
23      */
24     Spinner spinDoc; 4 usages
25
26     /**
27      * Spinner pour sélectionner une heure de rendez-vous.
28      */
29     Spinner spinH; 3 usages
30
31     /**
32      * Liste des docteurs disponibles.
33      */
34     List<String> listeDoc; 5 usages
35
36     /**
37      * Tableau des heures disponibles pour un rendez-vous.
38      */
39     String[] heureDeRdv = {"08h00", "08h30", "09h00", "09h30", "10h00", "10h30", 2 usages
40         "11h00", "11h30", "12h00", "12h30", "13h00", "13h30",
41         "14h00", "14h30", "15h00", "15h30", "16h00", "16h30",
42         "17h00", "17h30", "18h00"};
```

```

59
60 /**
61  * Composant calendrier pour sélectionner une date.
62  */
63 CalendarView cal; 2 usages
64
65 /**
66  * Date sélectionnée dans le calendrier.
67  */
68 String Date; 4 usages
69
70 /**
71  * Heure sélectionnée dans le spinner.
72  */
73 String heure; 4 usages
74
75 /**
76  * Docteur sélectionné dans le spinner.
77  */
78 String doc; 1 usage
79
80 /**
81  * Méthode appelée à la création de l'activité.
82  * Initialise les composants graphiques, récupère les données nécessaires
83  * et configure les spinners et le calendrier.
84  *
85  * @param savedInstanceState Instance sauvegardée de l'état de l'activité.
86  */
87 @Override
88 protected void onCreate(Bundle savedInstanceState) {
89     super.onCreate(savedInstanceState);
90     EdgeToEdge.enable( $this$enableEdgeToEdge: this);
91     setContentView(R.layout.activity_prendre_rdv);
92
93     ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
94         Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
95         v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
96         return insets;
97     });
98

```



```

98         bd = new BD( context, this);
99
100
101         // Initialisation du calendrier pour sélectionner une date.
102         cal = (CalendarView) findViewById(R.id.calendarViewRDV);
103         cal.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
104             @Override no usages
105             public void onSelectedDayChange(@NonNull CalendarView calendarView, int year, int month, int day) {
106                 // Ajustement : ajouter 1 au mois (les mois dans CalendarView commencent à 0).
107                 Date = day + "/" + (month + 1) + "/" + year;
108             }
109         });
110
111         // Initialisation du spinner pour les heures de rendez-vous.
112         spinH = (Spinner) findViewById(R.id.spinnerHeure);
113         ArrayAdapter<String> heures = new ArrayAdapter<>( context, this, android.R.layout.simple_spinner_item, heureDeRdv);
114         heures.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
115         spinH.setAdapter(heures);
116         spinH.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
117             @Override no usages
118             public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
119                 heure = heureDeRdv[i];
120             }
121
122             @Override no usages
123             public void onNothingSelected(AdapterView<?> adapterView) {
124             }
125         });
126
127         // Récupération des docteurs disponibles depuis la base de données.
128         listeDoc = new ArrayList<>();
129         Cursor cursor = bd.getDoc();
130         if (cursor != null && cursor.moveToFirst()) {
131             do {
132                 int indexNom = cursor.getColumnIndex( s: "NOM");
133
134                 if (indexNom >= 0) {
135                     String nom = cursor.getString(indexNom);
136                     listeDoc.add(nom);
137                 }
138             } while (cursor.moveToNext());
139         }
140     }
141 }

```

```

137         }
138     } while (cursor.moveToNext());
139     cursor.close();
140 } else {
141     listeDoc.add("Aucun docteur disponible");
142 }
143
144 // Initialisation du spinner pour les docteurs.
145 spinDoc = findViewById(R.id.spinnerDr);
146 ArrayAdapter<String> docteurs = new ArrayAdapter<>(context, this, android.R.layout.simple_spinner_item, listeDoc);
147 docteurs.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
148 spinDoc.setAdapter(docteurs);
149 spinDoc.setOnItemClickListener(new AdapterView.OnItemClickListener() {
150     @Override no usages
151     public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
152         doc = listeDoc.get(i);
153     }
154
155     @Override no usages
156     public void onNothingSelected(AdapterView<?> adapterView) {
157     }
158 });
159
160 liste = (TextView) findViewById(R.id.liste);
161 }
162
163 /**
164  * Méthode appelée lors du clic sur le bouton "Enregistrer RDV".
165  * Valide les champs sélectionnés, crée le rendez-vous et affiche un message.
166  *
167  * @param view Vue du bouton "Enregistrer RDV".
168  */
169 public void enregistrerRDV(View view) { 1 usage
170     if (Date == null || Date.isEmpty()) {
171         Toast.makeText(context, this, text: "Veuillez sélectionner une date.", Toast.LENGTH_LONG).show();
172         return;
173     }
174
175     if (heure == null || heure.isEmpty()) {
176         Toast.makeText(context, this, text: "Veuillez sélectionner une heure.", Toast.LENGTH_LONG).show();

```

```

176 Toast.makeText( context: this, text: "Veuillez sélectionner une heure.", Toast.LENGTH_LONG).show();
177 return;
178 }
179
180 String docteurSelectionne = spinDoc.getSelectedItem().toString();
181 String idDocteur = bd.getIdDoc(docteurSelectionne);
182 if (idDocteur != null && !idDocteur.isEmpty()) {
183     try {
184         bd.CreerRdv(Date, heure, Integer.parseInt(idDocteur));
185         maj();
186         Toast.makeText( context: this, text: "Rendez-vous enregistré avec succès.", Toast.LENGTH_LONG).show();
187     } catch (Exception e) {
188         Toast.makeText( context: this, text: "Erreur lors de la création du rendez-vous.", Toast.LENGTH_LONG).show();
189         e.printStackTrace();
190     }
191 } else {
192     Toast.makeText( context: this, text: "Veuillez sélectionner un docteur valide.", Toast.LENGTH_LONG).show();
193 }
194 }
195
196 /**
197  * Met à jour le TextView pour afficher la liste des rendez-vous enregistrés.
198  */
199 public void maj() { 1 usage
200     liste.setText("");
201     try {
202         Cursor data = bd.getRDV1();
203         String texte = "";
204         while (data.moveToNext()) {
205             texte = texte + String.valueOf( obj: data.getString( 0) + " " + data.getString( 1) + " ");
206         }
207         liste.setText(texte);
208     } catch (Exception e) {
209         liste.setText(e.getMessage());
210     }
211 }
212
213 /**
214  * Méthode appelée lors du clic sur le bouton "Retour".
215  * Retourne à l'activité principale et termine l'activité en cours.
216  *
217  * @param view Vue du bouton "Retour".
218  */
219 public void retour(View view) { 4 usages
220     Intent intent = new Intent( packageContext: this, MainActivity.class);
221     startActivity(intent);
222     finish();
223 }
224 }
225

```

## Enregistre.java

```

1 package com.example.projetandroid;
2
3 > import ...
4
19
20 /**
21  * Activité Enregistre : permet d'enregistrer des professionnels dans la base de données.
22  */
23 ▶ </> public class Enregistre extends AppCompatActivity {
24     ≡
25     /**
26      * Instance de la base de données.
27      */
28     BD bd; 3 usages
29
30     /**
31      * Champs d'entrée pour les informations du professionnel.
32      */
33     EditText nom, prenom, type, adresse, ville, codeP, num; 2 usages
34
35     /**
36      * TextView pour afficher la liste des professionnels enregistrés.
37      */
38     TextView liste; 4 usages
39
40     /**
41      * Méthode appelée à la création de l'activité.
42      * Initialise les composants graphiques et met à jour la liste des professionnels.
43      *
44      * @param savedInstanceState Instance sauvegardée de l'état de l'activité.
45      */
46     Ⓢ @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
49         EdgeToEdge.enable( $this$enableEdgeToEdge: this);
50         setContentView(R.layout.activity_enregistre);
51
52         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
53             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
54             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
55             return insets;
56         });
57
58         // Initialisation des champs d'entrée.

```

```

57 // Initialisation des champs d'entrée.
58 bd = new BD(context: this);
59 nom = (EditText) findViewById(R.id.editTextNom);
60 prenom = (EditText) findViewById(R.id.editTextPrenom);
61 type = (EditText) findViewById(R.id.editTextType);
62 adresse = (EditText) findViewById(R.id.editTextAdresse);
63 ville = (EditText) findViewById(R.id.editTextVille);
64 codeP = (EditText) findViewById(R.id.editTextCodeP);
65 num = (EditText) findViewById(R.id.editTextN_tel);
66
67 // Initialisation du TextView pour la liste des professionnels.
68 liste = (TextView) findViewById(R.id.liste);
69
70 // Mise à jour de la liste des professionnels.
71 maj();
72 }
73
74 /**
75  * Méthode appelée lors du clic sur le bouton "Enregistrer".
76  * Enregistre les données du professionnel dans la base de données
77  * et met à jour la liste affichée.
78  *
79  * @param view Vue du bouton "Enregistrer".
80  */
81 public void enregistrerPro(View view) { 1 usage
82     bd.enrPro(
83         nom.getText().toString(),
84         prenom.getText().toString(),
85         type.getText().toString(),
86         adresse.getText().toString(),
87         ville.getText().toString(),
88         codeP.getText().toString(),
89         num.getText().toString()
90     );
91     maj();
92 }

```

```

94      /**
95       * Met à jour le TextView pour afficher la liste des professionnels enregistrés.
96       */
97      public void maj() { 2 usages
98          liste.setText("");
99          try {
100              Cursor data = bd.getAllData();
101              String texte = "";
102              while (data.moveToNext()) {
103                  texte += data.getString(1) + " " + data.getString(2) + "\n";
104              }
105              liste.setText(texte);
106          } catch (Exception e) {
107              liste.setText(e.getMessage());
108          }
109      }
110
111      /**
112       * Méthode appelée lors du clic sur le bouton "Retour".
113       * Retourne à l'activité principale et termine l'activité en cours.
114       *
115       * @param view Vue du bouton "Retour".
116       */
117      public void retour(View view) { 4 usages
118          Intent intent = new Intent( packageContext: this, MainActivity.class);
119          startActivity(intent);
120          finish();
121      }
122  }
123

```

planning.java

```

1 package com.example.projetandroid;
2
3 > import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17 /**
18  * Activité Planning : permet de visualiser les rendez-vous planifiés pour une date donnée.
19  */
20 <> public class Planning extends AppCompatActivity {
21
22     /**
23      * Instance de la base de données.
24      */
25     BD bd; 3 usages
26
27     /**
28      * Widget pour la sélection d'une date.
29      */
30     CalendarView cal; 2 usages
31
32     /**
33      * Date sélectionnée dans le calendrier.
34      */
35     String date; 3 usages
36
37     /**
38      * TextView pour afficher les rendez-vous planifiés.
39      */
40     TextView liste; 4 usages
41
42     /**
43      * Méthode appelée lors de la création de l'activité.
44      * Initialise les composants graphiques et configure le calendrier.
45      *
46      * @param savedInstanceState Instance sauvegardée de l'état de l'activité.
47      */
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         EdgeToEdge.enable($this$enableEdgeToEdge: this);
52         setContentView(R.layout.activity_planning);
53
54         // Gestion des marges pour les barres système.

```

```

61 // Initialisation des champs.
62 bd = new BD(context: this);
63 liste = (TextView) findViewById(R.id.textViewPlanning);
64 cal = (CalendarView) findViewById(R.id.calendarViewPlanning);
65
66 // Listener pour détecter la sélection d'une date.
67 cal.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
68     @Override no usages
69     public void onSelectedDayChange(@NonNull CalendarView calendarView, int year, int month, int day) {
70         // Les mois commencent à 0 dans CalendarView.
71         date = day + "/" + (month + 1) + "/" + year;
72     }
73 });
74 }
75
76 /**
77  * Méthode appelée lors du clic sur le bouton "Voir Planning".
78  * Récupère les rendez-vous pour la date sélectionnée et met à jour l'affichage.
79  *
80  * @param view Vue du bouton "Voir Planning".
81  */
82 public void voirPlanning(View view) { 1 usage
83     bd.getRDV(date);
84     maj();
85 }
86
87 /**
88  * Met à jour le TextView pour afficher les rendez-vous de la date sélectionnée.
89  */
90 public void maj() { 1 usage
91     liste.setText("");
92     try {
93         Cursor data = bd.getRDV(date);
94         String texte = "";
95         while (data.moveToNext()) {
96             String dateRdv = data.getString(0);
97             String heureRdv = data.getString(1);
98             String nomDoc = data.getString(2);
99             String prenomDoc = data.getString(3);
100             String nomDoc = data.getString(2);
101             String prenomDoc = data.getString(3);
102
103             // Ajout des détails du rendez-vous au texte.
104             texte += "Date: " + dateRdv + " - Heure: " + heureRdv + " - Docteur: " + nomDoc + " " + prenomDoc + "\n";
105         }
106         liste.setText(texte);
107     } catch (Exception e) {
108         liste.setText(e.getMessage());
109     }
110 }
111
112 /**
113  * Méthode appelée lors du clic sur le bouton "Retour".
114  * Retourne à l'activité principale et termine l'activité actuelle.
115  *
116  * @param view Vue du bouton "Retour".
117  */
118 public void retour(View view) { 4 usages
119     Intent intent = new Intent(context: this, MainActivity.class);
120     startActivity(intent);
121     finish();
122 }

```