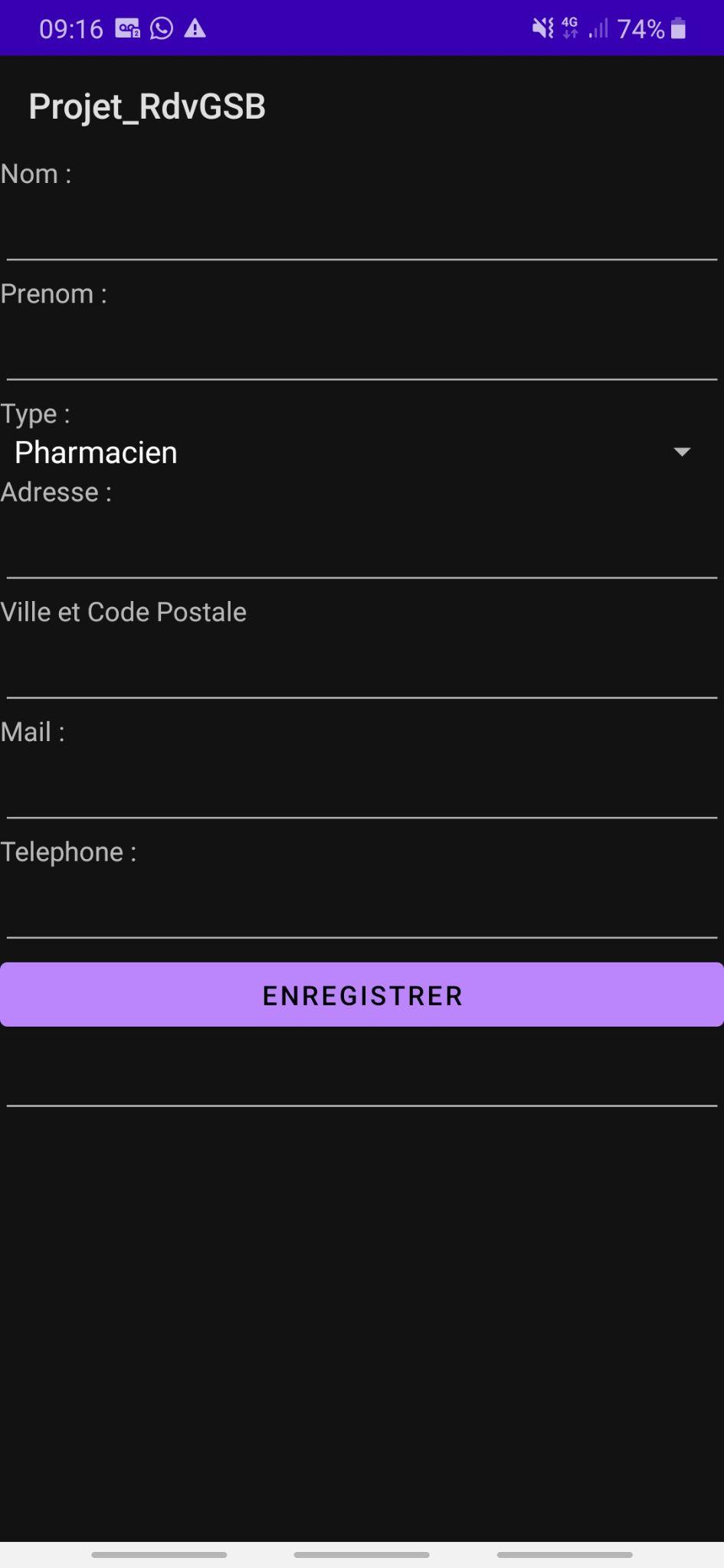
**Interface XML : Accueil** 

**Interface XML : Enregistrer un professionnel**



**Code XML**

**JAVA**

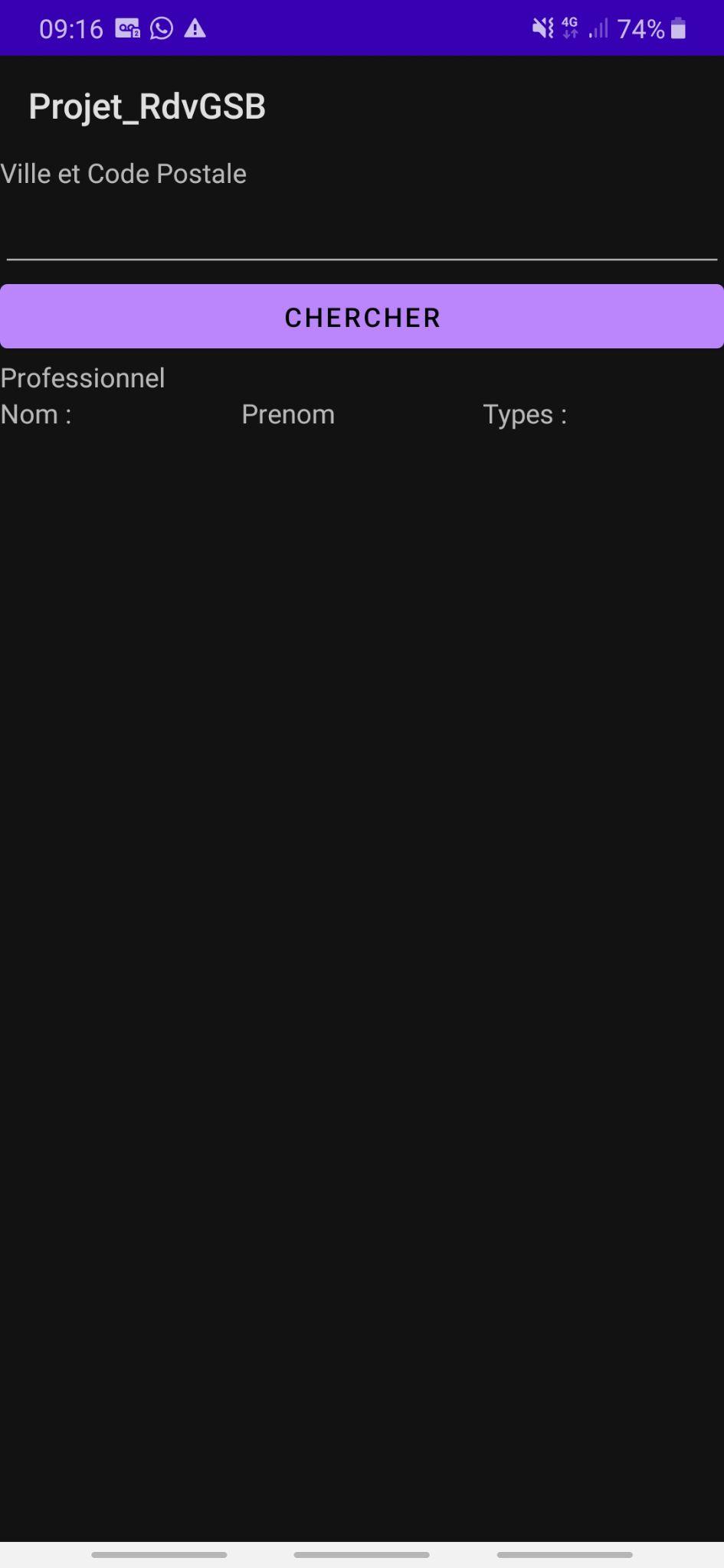
| package com.example.projet\_rdvgsb;  import androidx.appcompat.app.AppCompatActivity;  import android.database.Cursor;  import android.os.Bundle;  import android.view.View;  import android.widget.AdapterView;  import android.widget.ArrayAdapter;  import android.widget.EditText;  import android.widget.Spinner;  import android.widget.TextView;  import java.util.ArrayList;  import java.util.List;  public class EnregistrerPro extends AppCompatActivity  {  // Déclaration des variables membres  BD bd;  EditText nom;  EditText prenom;  EditText type;  EditText adresse;  EditText villeCodep;  EditText mail;  EditText tel;  Spinner typeSpinner;  int idSelected;  ArrayList<String> listeTypes;  @Override  protected void onCreate(Bundle savedInstanceState)  {  // Initialisation de l'activité  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_enregistrer\_pro);  // Initialisation de la base de données et des composants d'interface  bd = new BD(this);  nom = findViewById(R.id.EnregeditTextTextNom);  prenom = findViewById(R.id.EnregeditTextPrenom);  adresse = findViewById(R.id.EnregeditTextEnregType);  villeCodep = findViewById(R.id.EnregeditTextEnregVilleCodeP);  mail = findViewById(R.id.EnregeditTextMail);  tel = findViewById(R.id.EnregeditTextTel);  typeSpinner = findViewById(R.id.spinnerEnregPro);  // Mise à jour des types au démarrage de l'activité  majTypes();  }  // Méthode appelée lorsqu'un bouton est cliqué pour enregistrer un professionnel  public void enregClic(View view)  {  // Appel de la méthode d'enregistrement dans la base de données  bd.enregPro(nom.getText().toString(), prenom.getText().toString(),  mail.getText().toString(), tel.getText().toString(), adresse.getText().toString(),  idSelected, villeCodep.getText().toString());  // Réinitialisation des champs de saisie  nom.setText("");  prenom.setText("");  adresse.setText("");  villeCodep.setText("");  mail.setText("");  tel.setText("");  }  // Méthode pour mettre à jour la liste des types  public void majTypes()  {  try{  // Création d'une liste de types  listeTypes = new ArrayList<>();  listeTypes.add("Pharmacien");  listeTypes.add("Médecin");  listeTypes.add("Podologue");  // Création d'un adaptateur pour le Spinner  ArrayAdapter<String> aaTypes = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, listeTypes);  aaTypes.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  // Liaison de l'adaptateur au Spinner  typeSpinner.setAdapter(aaTypes);  // Gestion de la sélection d'un élément dans le Spinner  typeSpinner.setOnItemClickListener(new AdapterView.OnItemClickListener() {  @Override  public void onItemClick(AdapterView<?> adapterView, View view, int i, long l)  {  // Mise à jour de l'identifiant sélectionné  idSelected = i;  }  });  }  catch(Exception e)  {  // Gestion des exceptions (non spécifiée dans le code)  }  }  } |
| --- |

**XML : Enregistrer un RDV** 

**CODE XML**

**JAVA**

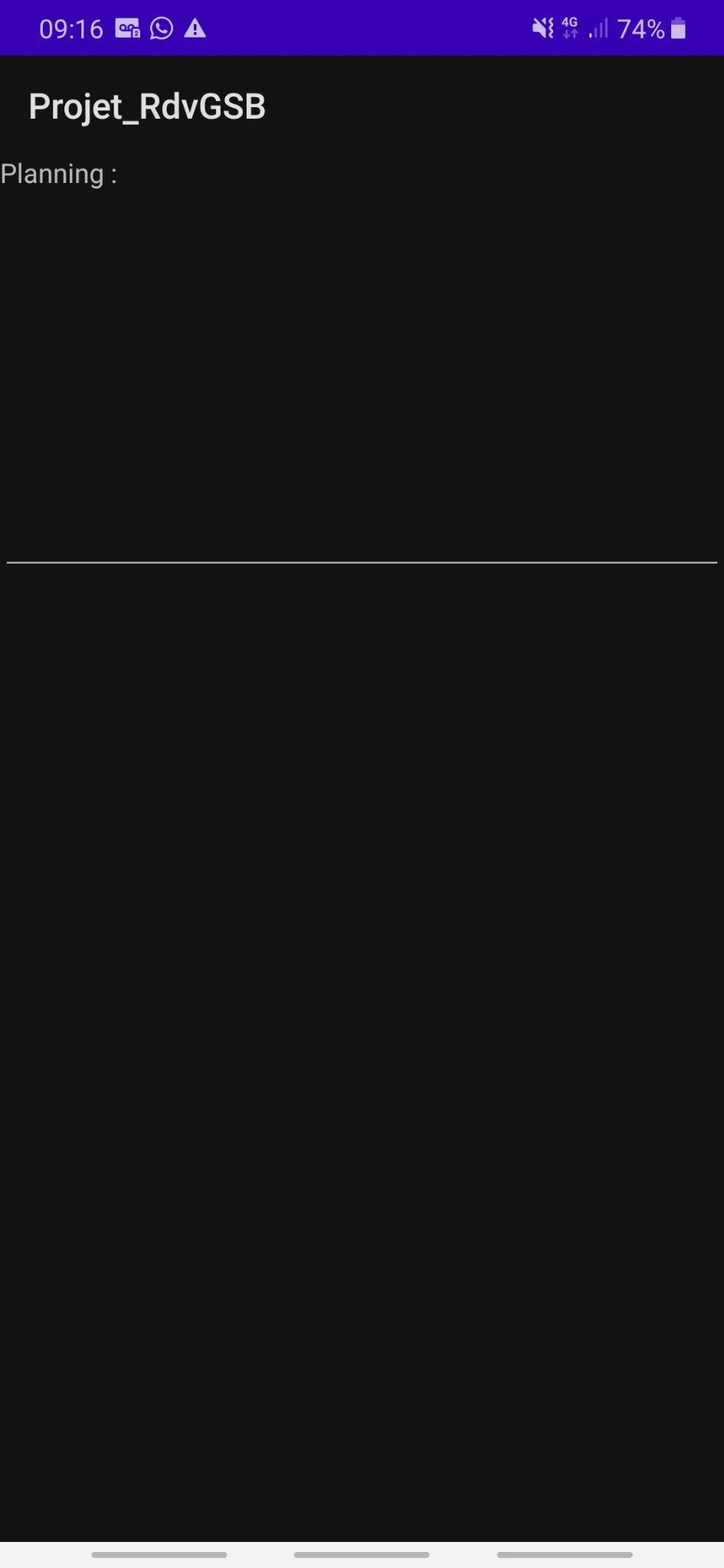
| package com.example.projet\_rdvgsb;  import androidx.annotation.NonNull;  import androidx.appcompat.app.AppCompatActivity;  import android.database.Cursor;  import android.os.Bundle;  import android.view.View;  import android.widget.ArrayAdapter;  import android.widget.CalendarView;  import android.widget.Spinner;  import android.widget.TextView;  import android.widget.Toast;  import java.util.ArrayList;  public class PrendreRDV extends AppCompatActivity  {  // Déclaration des variables membres  BD bd;  TextView heureDeb;  TextView heureFin;  CalendarView date;  Spinner nompreP;  TextView motif;  String curDate;  @Override  protected void onCreate(Bundle savedInstanceState)  {  // Initialisation de l'activité  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_prendre\_rdv);  // Initialisation de la base de données et des composants d'interface  bd = new BD(this);  heureDeb = findViewById(R.id.editTextHeureDeb);  heureFin = findViewById(R.id.editTextHeureFin);  date = findViewById(R.id.calendarViewDate);  nompreP = findViewById(R.id.spinnerChoisirType);  motif = findViewById(R.id.editTextMultiLineMotifRDV);  // Gestion de la sélection de la date dans le CalendarView  date.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {  @Override  public void onSelectedDayChange(CalendarView view, int annee, int mois, int jours)  {  curDate = String.valueOf(jours) + "/" + String.valueOf(mois) + "/" + String.valueOf(annee);  }  });  // Mise à jour des professionnels au démarrage de l'activité  majPro();  }  // Méthode appelée lorsqu'un bouton est cliqué pour enregistrer un RDV  public void clickEnregRDV(View view)  {  // Récupération de tous les rendez-vous  Cursor data = bd.getAllDataRDV();  ArrayList<String> listePlanning = new ArrayList<>();  boolean trouve = false;  // Parcours des rendez-vous  while (data.moveToNext())  {  // Vérification si l'heure et la date du RDV correspondent à un existant  if (heureDeb.getText().equals(data.getString(2)) && curDate.equals(data.getString(1)))  {  // Affichage d'un message d'erreur  Toast.makeText(getApplicationContext(), "Erreur horaire", Toast.LENGTH\_SHORT).show();  trouve = true;  }  // Si pas de conflit horaire, enregistrement du RDV  if (!trouve)  {  bd.enregRDV(curDate, Integer.parseInt(heureDeb.getText().toString()),  Integer.parseInt(heureFin.getText().toString()),  motif.getText().toString(),  Integer.parseInt(nompreP.getSelectedItem().toString().split(" ")[0]));  }  }  // Réinitialisation des zones de saisie  heureDeb.setText("");  heureFin.setText("");  motif.setText("");  }  // Méthode pour mettre à jour la liste des professionnels dans le Spinner  public void majPro()  {  try{  // Récupération de tous les professionnels  Cursor data = bd.getAllDataPro();  ArrayList<String> listePro = new ArrayList<>();  // Parcours des professionnels et ajout à la liste  while (data.moveToNext())  {  listePro.add(data.getInt(0) + " " + data.getString(1) + " " + data.getString(2));  }  // Création de l'adaptateur pour le Spinner  ArrayAdapter<String> aaPro = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, listePro);  aaPro.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  // Liaison de l'adaptateur au Spinner  nompreP.setAdapter(aaPro);  }  catch(Exception e)  {  // Gestion des exceptions (non spécifiée dans le code)  }  }  } |
| --- |

**Interface XML : Rechercher un professionnel par code postale ou/et ville**

**CODE XML**

**JAVA**

| package com.example.projet\_rdvgsb;  import androidx.appcompat.app.AppCompatActivity;  import android.database.Cursor;  import android.os.Bundle;  import android.view.View;  import android.widget.ArrayAdapter;  import android.widget.TextView;  import java.util.ArrayList;  public class RecherchePro extends AppCompatActivity  {  // Déclaration des variables membres  TextView codevilleP;  BD bd;  TextView afficherP;  TextView Nom;  TextView Prenom;  TextView villecodeP;  @Override  protected void onCreate(Bundle savedInstanceState)  {  // Initialisation de l'activité  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main\_activity2\_recherche\_pro);  // Initialisation de la base de données  bd = new BD(this);  // Liaison des variables avec les composants d'interface  codevilleP = findViewById(R.id.editTextRecherProCodeP);  Nom = findViewById(R.id.textView21AffiNom);  Prenom = findViewById(R.id.textView22AffiPren);  villecodeP = findViewById(R.id.textView23AAffiType);  // Mise à jour des professionnels au démarrage de l'activité  majPro();  }  // Méthode pour mettre à jour l'affichage des professionnels  public void majPro()  {  try  {  // Récupération de toutes les données des professionnels  Cursor donnee = bd.getAllDataPro();  String prenom = "";  String nom = "";  String villecode = "";  // Parcours du résultat de la requête  while (donnee.moveToNext())  {  nom += donnee.getString(1) + "\n";  prenom += donnee.getString(2) + "\n";  villecode += donnee.getString(6) + "\n";  }  // Affichage des données dans les TextView correspondants  Nom.setText(nom);  Prenom.setText(prenom);  villecodeP.setText(villecode);  }  catch (Exception e)  {  // Gestion des exceptions (non spécifiée dans le code)  }  }  // Méthode pour afficher les professionnels en fonction du code ville  public void Affic(String codeville)  {  try  {  // Réinitialisation des TextView  Nom.setText("");  Prenom.setText("");  villecodeP.setText("");  // Récupération des données des professionnels en fonction du code ville  Cursor donnee = bd.selectionnerPro(codeville);  String prenom = "";  String nom = "";  String villecode = "";  // Parcours du résultat de la requête  while (donnee.moveToNext())  {  nom += donnee.getString(1) + "\n";  prenom += donnee.getString(2) + "\n";  villecode += donnee.getString(6) + "\n";  }  // Affichage des données dans les TextView correspondants  Nom.setText(nom);  Prenom.setText(prenom);  villecodeP.setText(villecode);  }  catch (Exception e)  {  // Gestion des exceptions (non spécifiée dans le code)  }  }  // Méthode appelée lorsqu'un bouton est cliqué  public void clickBtn(View view)  {  // Appel de la méthode Affic avec le code ville entré par l'utilisateur  Affic(codevilleP.getText().toString());  }  } |
| --- |

**Interface XML : Planning**

**CODE XML**

**JAVA**

| **package com.example.projet\_rdvgsb;**  **import androidx.appcompat.app.AppCompatActivity;**  **import android.database.Cursor;**  **import android.os.Bundle;**  **import android.widget.ArrayAdapter;**  **import android.widget.CalendarView;**  **import android.widget.EditText;**  **import java.util.ArrayList;**  **import java.util.Date;**  **import java.util.HashMap;**  **import java.util.Locale;**  **import java.util.Map;**  **public class Planning extends AppCompatActivity**  **{**  **// Déclaration des variables membres**  **EditText Planing;**  **BD bd;**  **String curDate;**  **@Override**  **protected void onCreate(Bundle savedInstanceState)**  **{**  **// Initialisation de l'activité**  **bd = new BD(this);**  **super.onCreate(savedInstanceState);**  **setContentView(R.layout.activity\_main\_planning);**  **// Liaison de la variable Planing avec le composant d'interface editTextTextMultiLinePlanning**  **Planing = findViewById(R.id.editTextTextMultiLinePlanning);**  **try{**  **// Récupération de tous les rendez-vous depuis la base de données**  **Cursor data = bd.getAllDataRDV();**  **ArrayList<String> listePlanning = new ArrayList<>();**  **String planning = "";**  **// Parcours du résultat de la requête**  **while (data.moveToNext())**  **{**  **// Construction d'une chaîne de caractères représentant le rendez-vous**  **planning += data.getString(1) + " " + data.getString(2) + "h -" + data.getString(3) + "h avec : " + bd.recherchProfromId(data.getInt(5)) + "\n";**  **}**  **// Affichage du planning dans le composant d'interface**  **Planing.setText(planning);**  **}**  **catch(Exception e)**  **{**  **// En cas d'erreur, affichage du message d'erreur dans le composant d'interface**  **Planing.setText(e.getMessage());**  **}**  **}**  **}** |
| --- |

**CODE XML : Class BD**

**JAVA**

| **package com.example.projet\_rdvgsb;**  **import android.content.ContentValues;**  **import android.content.Context;**  **import android.database.Cursor;**  **import android.database.sqlite.SQLiteDatabase;**  **import android.database.sqlite.SQLiteOpenHelper;**  **import java.sql.Date;**  **import java.sql.Time;**  **/\*\***  **\* Classe qui étend SQLiteOpenHelper pour gérer la base de données.**  **\*/**  **public class BD extends SQLiteOpenHelper {**  **// Définition des noms de la base de données et des tables**  **public static final String DATABASE\_NAME = "RDVVendeur.db";**  **public static final String TABLE\_NAME = "professionnels";**  **public static final String idP = "id\_professionnel";**  **public static final String nomP = "nom";**  **public static final String prenomP = "prenom";**  **public static final String emailP = "email";**  **public static final String telP = "tel";**  **public static final String adreseP = "adresse";**  **public static final String villecodeP = "ville\_codep";**  **public static final String IDType = "id\_type";**  **public static final String TABLE\_NAME2 = "types";**  **public static final String idT = "id\_type";**  **public static final String nomT = "nom";**  **public static final String TABLE\_NAME3 = "RDV";**  **public static final String idR = "id\_rdv";**  **public static final String dateRDV = "date\_RDV";**  **public static final String heuredeDeb = "heure\_Deb";**  **public static final String heuredeFin = "heure\_Fin";**  **public static final String motifR = "motif";**  **public static final String IDPro = "id\_professionnel";**  **/\*\***  **\* Constructeur de la classe.**  **\***  **\* @param context Le contexte de l'application.**  **\*/**  **public BD(Context context) {**  **super(context, DATABASE\_NAME, null, 1);**  **}**  **/\*\***  **\* Méthode appelée lors de la création de la base de données.**  **\***  **\* @param db La base de données SQLite.**  **\*/**  **@Override**  **public void onCreate(SQLiteDatabase db) {**  **// Création des tables avec les champs correspondants**  **db.execSQL("CREATE table " + TABLE\_NAME + "(id\_professionnel INTEGER PRIMARY KEY AUTOINCREMENT, nom TEXT,prenom TEXT,email TEXT,tel TEXT,adresse TEXT,ville\_codep TEXT,id\_type INTEGER NOT NULL,FOREIGN KEY(id\_type) REFERENCES types(id\_type))");**  **db.execSQL("CREATE table " + TABLE\_NAME2 + "(id\_type INTEGER PRIMARY KEY AUTOINCREMENT,nom TEXT)");**  **db.execSQL("CREATE table " + TABLE\_NAME3 + "(id\_rdv INTEGER PRIMARY KEY AUTOINCREMENT, date\_RDV TEXT, heure\_Deb INT, heure\_Fin INT,motif TEXT, id\_professionnel INTEGER NOT NULL, FOREIGN KEY(id\_professionnel) REFERENCES professionnels(id\_professionnel))");**  **}**  **/\*\***  **\* Méthode appelée lors de la mise à niveau de la base de données.**  **\***  **\* @param db La base de données SQLite.**  **\* @param oldVersion Ancienne version de la base de données.**  **\* @param newVersion Nouvelle version de la base de données.**  **\*/**  **@Override**  **public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {**  **// Suppression des tables existantes et recréation**  **db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME);**  **db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME2);**  **db.execSQL("DROP TABLE IF EXISTS " + TABLE\_NAME3);**  **onCreate(db);**  **}**  **/\*\***  **\* Méthode pour obtenir toutes les données des professionnels.**  **\***  **\* @return Un objet Cursor contenant les données des professionnels.**  **\*/**  **public Cursor getAllDataPro() {**  **SQLiteDatabase bd = this.getReadableDatabase();**  **Cursor result = bd.rawQuery("SELECT \* FROM " + TABLE\_NAME, null);**  **return result;**  **}**  **/\*\***  **\* Méthode pour enregistrer un professionnel.**  **\***  **\* @param nom Le nom du professionnel.**  **\* @param prenom Le prénom du professionnel.**  **\* @param email L'adresse e-mail du professionnel.**  **\* @param tel Le numéro de téléphone du professionnel.**  **\* @param adresse L'adresse du professionnel.**  **\* @param type L'identifiant du type du professionnel.**  **\* @param villecode Le code postal de la ville du professionnel.**  **\*/**  **public void enregPro(String nom, String prenom, String email, String tel, String adresse, int type, String villecode) {**  **SQLiteDatabase bd = this.getWritableDatabase();**  **ContentValues contentValues = new ContentValues();**  **contentValues.put(nomP, nom);**  **contentValues.put(prenomP, prenom);**  **contentValues.put(emailP, email);**  **contentValues.put(telP, tel);**  **contentValues.put(adreseP, adresse);**  **contentValues.put(IDType, type);**  **contentValues.put(villecodeP, villecode);**  **bd.insert(TABLE\_NAME, null, contentValues);**  **bd.close();**  **}**  **/\*\***  **\* Méthode pour enregistrer un rendez-vous.**  **\***  **\* @param date\_RDV La date du rendez-vous.**  **\* @param heureDeb L'heure de début du rendez-vous.**  **\* @param heureFin L'heure de fin du rendez-vous.**  **\* @param motifRDV Le motif du rendez-vous.**  **\* @param IDProfe L'identifiant du professionnel associé au rendez-vous.**  **\*/**  **public void enregRDV(String date\_RDV, int heureDeb, int heureFin, String motifRDV, int IDProfe) {**  **SQLiteDatabase bd = this.getWritableDatabase();**  **ContentValues contentValues = new ContentValues();**  **contentValues.put(dateRDV, date\_RDV);**  **contentValues.put(heuredeDeb, heureDeb);**  **contentValues.put(heuredeFin, heureFin);**  **contentValues.put(motifR, motifRDV);**  **contentValues.put(IDPro, IDProfe);**  **bd.insert(TABLE\_NAME3, null, contentValues);**  **bd.close();**  **}**  **/\*\***  **\* Méthode pour obtenir toutes les données des rendez-vous.**  **\***  **\* @return Un objet Cursor contenant les données des rendez-vous.**  **\*/**  **public Cursor getAllDataRDV() {**  **SQLiteDatabase bd = this.getReadableDatabase();**  **Cursor Plan = bd.rawQuery("SELECT \* FROM RDV GROUP BY date\_RDV ORDER BY date\_RDV", null);**  **return Plan;**  **}**  **/\*\***  **\* Méthode pour obtenir toutes les données des types.**  **\***  **\* @return Un objet Cursor contenant les noms des types.**  **\*/**  **public Cursor getAllTypes() {**  **SQLiteDatabase bd = this.getReadableDatabase();**  **Cursor result = bd.rawQuery("SELECT nom FROM types", null);**  **bd.close();**  **return result;**  **}**  **/\*\***  **\* Méthode pour enregistrer des types dans la table 'types'.**  **\*/**  **public void enregType() {**  **SQLiteDatabase bd = this.getWritableDatabase();**  **ContentValues contentValues = new ContentValues();**  **contentValues.put(nomT, "Pharmacien");**  **bd.insert(TABLE\_NAME2, null, contentValues);**  **contentValues.put(nomT, "Medecien");**  **bd.insert(TABLE\_NAME2, null, contentValues);**  **contentValues.put(nomT, "Chirugien");**  **bd.insert(TABLE\_NAME2, null, contentValues);**  **bd.close();**  **}**  **/\*\***  **\* Méthode pour sélectionner des professionnels par code postal.**  **\***  **\* @param codeVilleP Le code postal de la ville.**  **\* @return Un objet Cursor contenant les données des professionnels sélectionnés.**  **\*/**  **public Cursor selectionnerPro(String codeVilleP) {**  **SQLiteDatabase bd = this.getReadableDatabase();**  **Cursor result = bd.rawQuery("SELECT \* FROM " + TABLE\_NAME + " WHERE ville\_codep LIKE '%" + codeVilleP + "%'", null);**  **return result;**  **}**  **/\*\***  **\* Méthode pour obtenir des rendez-vous par date.**  **\***  **\* @param date La date des rendez-vous.**  **\* @return Un objet Cursor contenant les données des rendez-vous pour la date spécifiée.**  **\*/**  **public Cursor getRDVByDate(String date) {**  **SQLiteDatabase db = this.getReadableDatabase();**  **String query = "SELECT \* FROM " + TABLE\_NAME3 + " WHERE " + dateRDV + " = ?";**  **return db.rawQuery(query, new String[]{date});**  **}**  **/\*\***  **\* Méthode pour rechercher un professionnel à partir de son ID.**  **\***  **\* @param id L'identifiant du professionnel.**  **\* @return Le nom et le prénom du professionnel s'il est trouvé, sinon null.**  **\*/**  **public String recherchProfromId(int id) {**  **SQLiteDatabase bd = this.getReadableDatabase();**  **Cursor result = bd.rawQuery("SELECT nom, prenom FROM " + TABLE\_NAME + " WHERE id\_professionnel = + '" + id + "'", null);**  **if (result.moveToFirst()) {**  **return result.getString(0) + " " + result.getString(1);**  **}**  **return null;**  **}**  **/\*\***  **\* Méthode pour supprimer toutes les données de la table 'professionnels'.**  **\*/**  **public void supprTable() {**  **SQLiteDatabase db = this.getWritableDatabase();**  **db.execSQL("DELETE FROM " + TABLE\_NAME, null);**  **}**  **/\*\***  **\* Méthode pour vérifier une heure dans la table 'RDV'.**  **\*/**  **public void verifHeure() {**  **SQLiteDatabase db = this.getWritableDatabase();**  **db.execSQL("SELECT heure\_Deb FROM " + TABLE\_NAME3, null);**  **}**  **}** |
| --- |

**CODE XML : Test Unitaire**

**JAVA**

| import android.content.Context;  import android.database.Cursor;  import androidx.test.core.app.ApplicationProvider;  import androidx.test.ext.junit.runners.AndroidJUnit4;  import org.junit.After;  import org.junit.Before;  import org.junit.Test;  import org.junit.runner.RunWith;  import static org.junit.Assert.assertEquals;  import static org.junit.Assert.assertFalse;  import static org.junit.Assert.assertNotNull;  import static org.junit.Assert.assertTrue;  @RunWith(AndroidJUnit4.class)  public class BDTest {  private BD dbHelper;  @Before  public void setUp() {  Context context = ApplicationProvider.getApplicationContext();  dbHelper = new BD(context);  }  @After  public void tearDown() {  dbHelper.close();  }  @Test  public void testEnregPro() {  dbHelper.enregPro("John", "Doe", "john.doe@example.com", "123456789", "123 Main St", 1, "12345");  Cursor cursor = dbHelper.getAllDataPro();  assertNotNull(cursor);  assertTrue(cursor.moveToFirst());  assertEquals("John", cursor.getString(cursor.getColumnIndex(BD.nomP)));    }  @Test  public void testEnregRDV() {  dbHelper.enregRDV("2023-01-01", 10, 12, "Checkup", 1);  Cursor cursor = dbHelper.getAllDataRDV();  assertNotNull(cursor);  assertTrue(cursor.moveToFirst());  assertEquals("2023-01-01", cursor.getString(cursor.getColumnIndex(BD.dateRDV)));    }  @Test  public void testGetRDVByDate() {  dbHelper.enregRDV("2023-01-01", 10, 12, "Checkup", 1);  Cursor cursor = dbHelper.getRDVByDate("2023-01-01");  assertNotNull(cursor);  assertTrue(cursor.moveToFirst());  assertEquals("Checkup", cursor.getString(cursor.getColumnIndex(BD.motifR)));    }  @Test  public void testRecherchProfromId() {  dbHelper.enregPro("John", "Doe", "john.doe@example.com", "123456789", "123 Main St", 1, "12345");  String result = dbHelper.recherchProfromId(1);  assertNotNull(result);  assertEquals("John Doe", result);  }    } |
| --- |