Ce document regroupera les principaux programmes utilisés pour l'application médicale.

Medic'App

PPE 5

Fall Charles – Dubroca Bastien – Caillé Durand Dimitri

CONTENU

ÉLEMENTS DE PREPARATION DE L'APPLICATION

DIAGRAMME DE CAS D'UTILISATION

Le document ci-dessous est le diagramme des cas d'utilisations correspondant à notre application « Medic'App », il y présente les acteurs principaux de l'application ainsi que les attentes de chacun des acteurs.

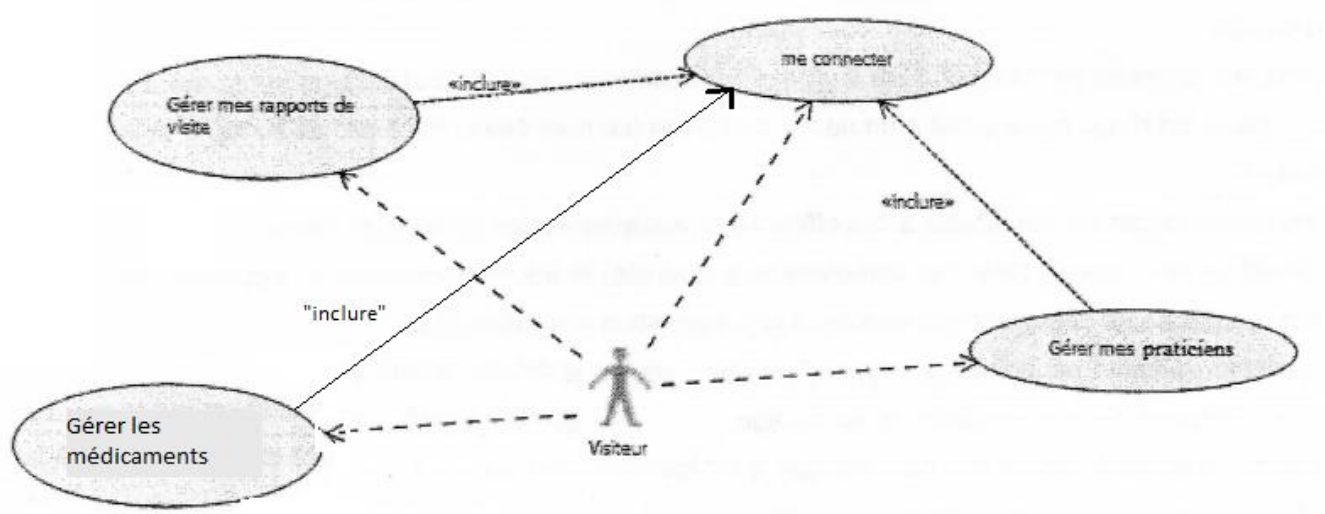
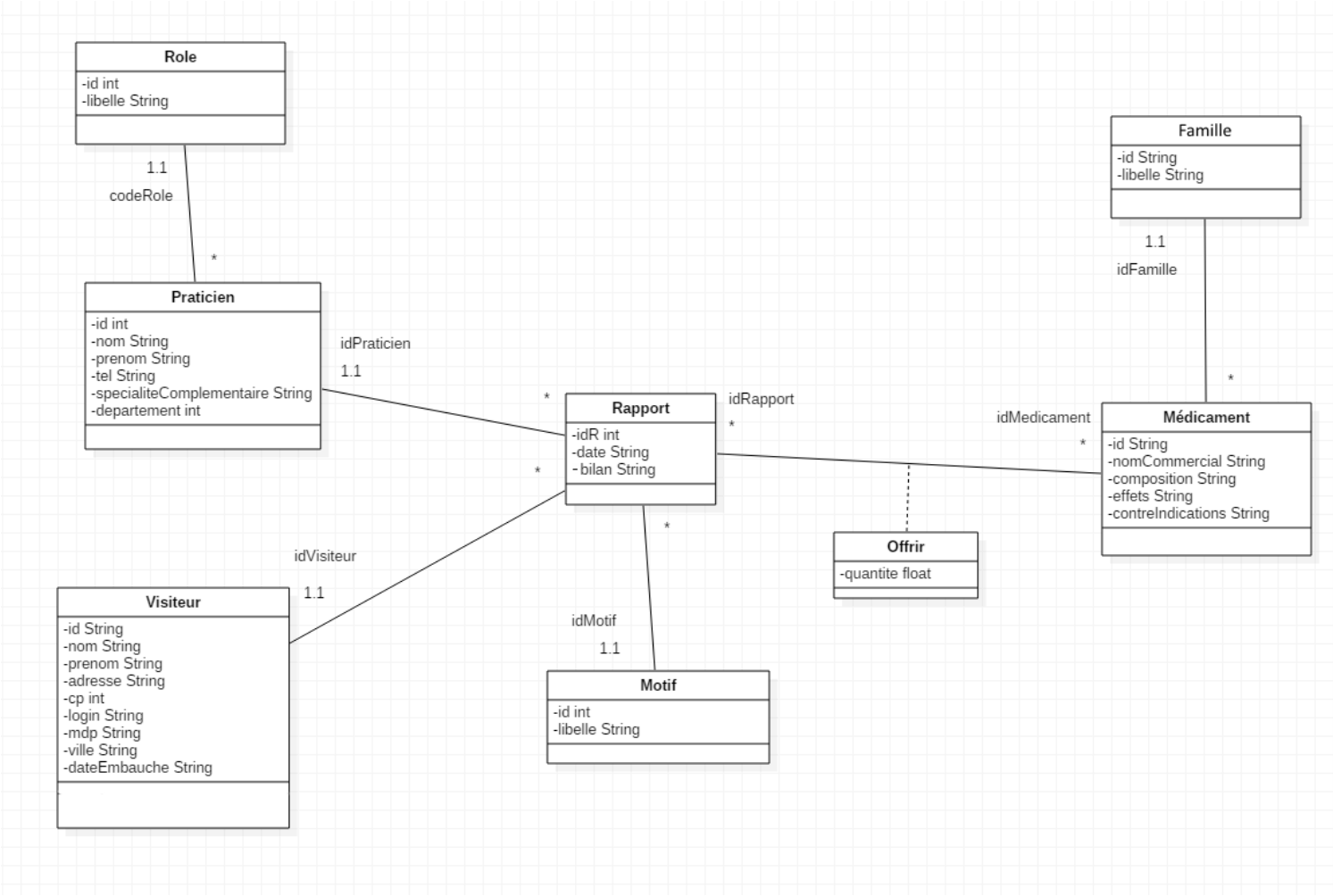


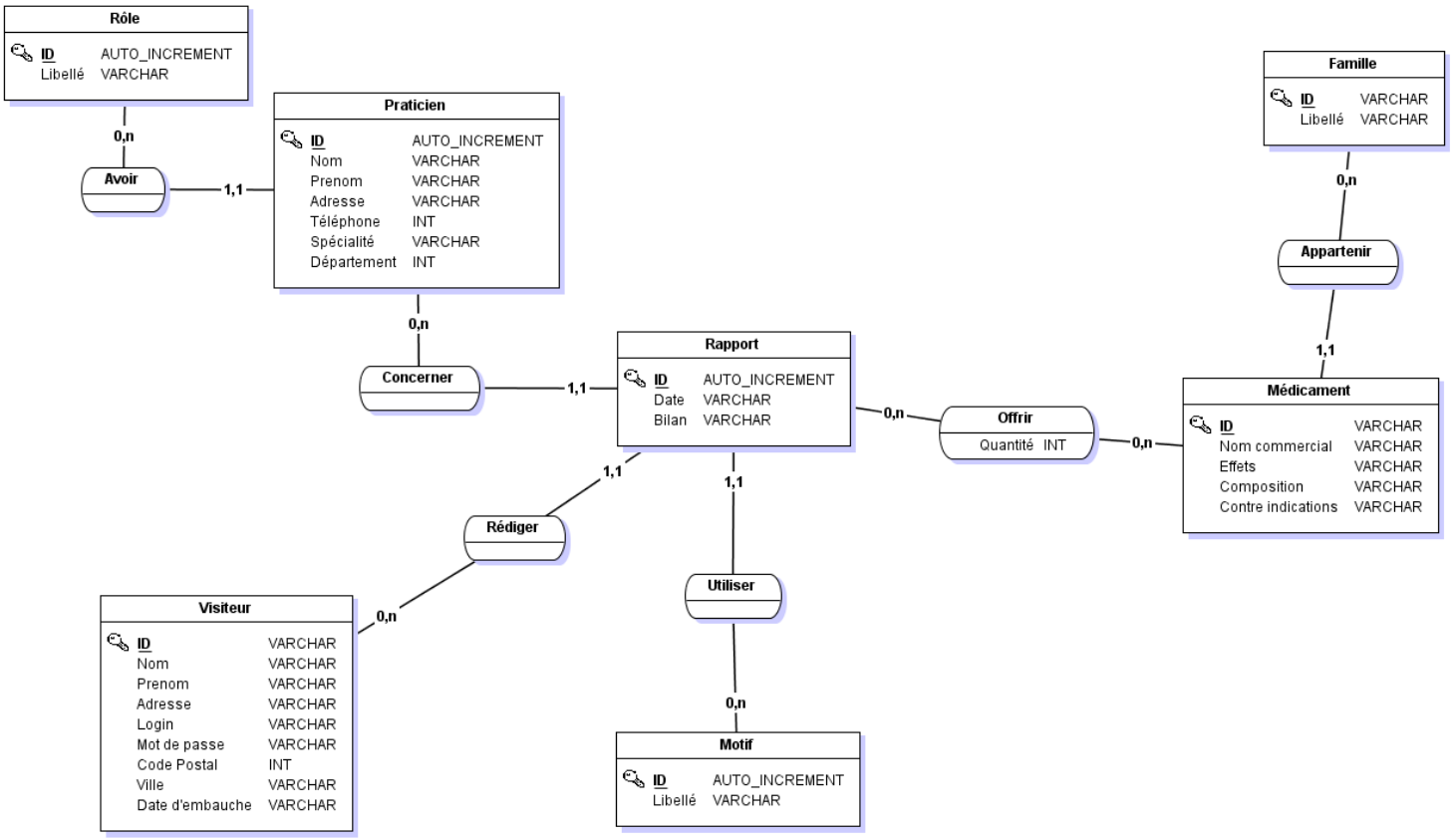
DIAGRAMME DE CLASSES METIERS

Le document ci-dessous est le diagramme de classes métiers correspondant à notre application « Medic'App », il y présente chacune des classes présentes dans le code de l'application.



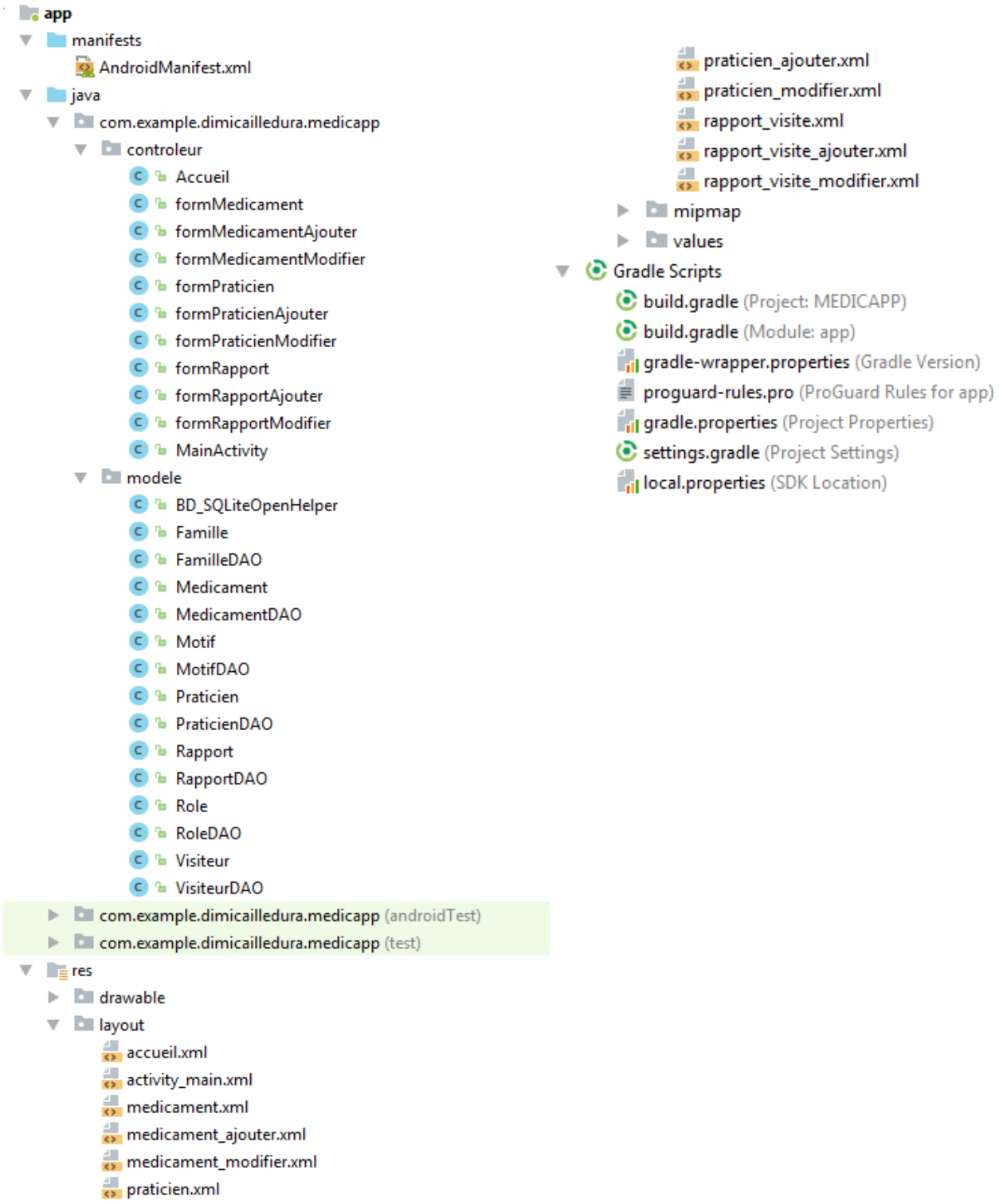
MODELE CONCEPTUEL DE DONNEES

Le document ci-dessous est le modèle conceptuel des données correspondant à notre application « Medic'App », il y présente la conception de la base de données.



ORGANISATION DU CODE

Le document ci-dessous est l'arborescence correspondant à notre application « Medic'App ».



LES CODES COMPOSANTS L'APPLICATION

L'ACTIVITE « MAINACTIVITY »

```
public class MainActivity extends AppCompatActivity {

    //Variables

    EditText loginText;
    EditText mdpText;
    VisiteurDAO unVisiteurDAO;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //accès à VisiteurDAO
        unVisiteurDAO = new VisiteurDAO( ct: this);

        //lien avec le fichier XML
        loginText = (EditText) findViewById(R.id.login);
        mdpText = (EditText) findViewById(R.id.mdp);
        Button BtnValider = (Button) findViewById(R.id.connexion);

        //gestion du bouton 'Valider'
        BtnValider.setOnClickListener((v) → {
                // TODO Auto-generated method stub
                Intent intent=null;

                //récupération des valeurs contenues dans les EditText
                String mdp = mdpText.getText().toString();
                String login = loginText.getText().toString();

                //test si les identifiants sont corrects
                Visiteur unVisiteur=unVisiteurDAO.getVisiteur(login,mdp);

                /*
                si identifiant faux : retour sur la même page et message d'erreur
                sinon passage à la page d'accueil en envoyant l'id de la personne connectée
                 */

                if(unVisiteur==null){
                    intent = new Intent(v.getContext(), MainActivity.class);
                    Toast.makeText(getApplicationContext(), text: "Nom de compte ou mot de passe incorrect", Toast.LENGTH_LONG).show();
                }else{
                    intent = new Intent(v.getContext(), Accueil.class);
                    intent.putExtra( name: "idVisiteurConnecte",unVisiteur.getId());
                }

                //lancement de la nouvelle activité
                startActivity(intent);
        });
    }
}
```

L'ACTIVITE « ACCUEIL »

```java
package com.example.dimicailledura.medicapp.controleur;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.example.dimicailledura.medicapp.R;

public class Accueil extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.accueil);

        //lien avec le fichier XML
        Button BtnPraticien = (Button) findViewById(R.id.praticien);
        Button BtnRapport_visite = (Button) findViewById(R.id.rapport_visite);
        Button BtnMedicament = (Button) findViewById(R.id.medicament);

        //gestion du bouton 'Praticien' et envoie vers la page portant ce nom
        BtnPraticien.setOnClickListener((v) -> {
                // TODO Auto-generated method stub
                Intent intent = new Intent(v.getContext(), formPraticien.class);
                startActivity(intent);
        });

        //gestion du bouton 'Rapport de visite' et envoie vers la page portant ce nom avec l'identifiant de la personne connectée
        BtnRapport_visite.setOnClickListener((v) -> {
                // TODO Auto-generated method stub
                Intent intent = new Intent(v.getContext(), formRapport.class);
                intent.putExtra( name: "idVisiteurConnecte",getIntent().getStringExtra( name: "idVisiteurConnecte"));
                startActivity(intent);
        });

        //gestion du bouton 'Médicament' et envoie vers la page portant ce nom
        BtnMedicament.setOnClickListener((v) -> {
                // TODO Auto-generated method stub
                Intent intent = new Intent(v.getContext(), formMedicament.class);
                startActivity(intent);
        });
    }
}
```

L'ACTIVITE « FORMMEDICAMENT »

```
public class formMedicament extends AppCompatActivity {
    //déclaration des variables
    private Spinner spinFamilles;
    private ArrayList<Medicament> listeMedicaments;
    private ArrayList<Famille> listeFamilles;
    private Spinner spinMedicaments;
    private MedicamentDAO medicamentAcces;
    private Button ajoutBouton;
    private Button modifierBouton;
    private Button supprimerBouton;
    private ArrayAdapter<String> spinMedicamentsAdapter;

    private TextView TVNomCommercial;
    private TextView TVEffets;
    private TextView TVContreIndications;
    private TextView TVComposition;
    private Medicament leMedicament;
    private int posFamille;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //ajout de la vue
        super.onCreate(savedInstanceState);
        setContentView(R.layout.medicament);

        //lien avec le fichier xml des TextView
        TVNomCommercial=(TextView)findViewById(R.id.nomcommercial);
        TVEffets=(TextView)findViewById(R.id.effets);
        TVContreIndications=(TextView)findViewById(R.id.contreindications);
        TVComposition=(TextView)findViewById(R.id.composition);

        //lien avec le fichier xml des boutons
        ajoutBouton = (Button) findViewById(R.id.ajoutermedic);
        modifierBouton = (Button) findViewById(R.id.modifiermedic);
        supprimerBouton=(Button) findViewById(R.id.supprimermedic);

        //déclaration des objets DAO nécessaires
        medicamentAcces = new MedicamentDAO( ct: this);
        FamilleDAO famillesAcces = new FamilleDAO( ct: this);

        //lien avec le fichier xml des Spinners
        spinFamilles = (Spinner) findViewById(R.id.spinFamille);
        spinMedicaments = (Spinner) findViewById(R.id.spinmedicament);
        spinMedicamentsAdapter = new ArrayAdapter<String>(this.getBaseContext(), android.R.layout.simple_spinner_item);

        //valorisation de la liste des familles grace à la fonction getFamilles() contenue dans FamilleDAO (liste d'objets famille)
        listeFamilles = famillesAcces.getFamilles();

        ArrayAdapter<String> spinFamillesAdapter = new ArrayAdapter<String>(this.getBaseContext(), android.R.layout.simple_spinner_item);
```

```
//clear pour éviter la redondance
spinFamillesAdapter.clear();

//valorisation de l'adapter avec les libelles de chaque famille
for (int i = 0; i < listeFamilles.size(); i++) {
    spinFamillesAdapter.add(listeFamilles.get(i).getLibelle());
}
spinFamilles.setAdapter(spinFamillesAdapter);

//listener sur le Spinner Famille
spinFamilles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        //changement de la liste de médicaments enfonction de la famille sélectionnée
        spinMedicamentsAdapter.clear();
        listeMedicaments = medicamentAcces.getMedicaments(listeFamilles.get(arg2).getLibelle());
        for (int i = 0; i < listeMedicaments.size(); i++) {
            spinMedicamentsAdapter.add(listeMedicaments.get(i).getNomCommercial());
        }
        spinMedicaments.setAdapter(spinMedicamentsAdapter);
        posFamille=arg2;
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});

//listener sur le Spinner Medicament
spinMedicaments.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        //changement des valeurs des textview en fonction du médicament sélectionnée
        TVNomCommercial.setText(listeMedicaments.get(arg2).getNomCommercial());
        TVEffets.setText(listeMedicaments.get(arg2).getEffets());
        TVContreIndications.setText(listeMedicaments.get(arg2).getContreIndications());
        TVComposition.setText(listeMedicaments.get(arg2).getComposition());
        leMedicament=listeMedicaments.get(arg2);


    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

```
    //listener sur le bouton d'ajout
    ajoutBouton.setOnClickListener((v) → {
            //amène sur la page d'ajout lors du clic sur le bouton
            Intent intent = new Intent(v.getContext(), formMedicamentAjouter.class);
            startActivity(intent);
    });

    //listener sur le bouton de modification
    modifierBouton.setOnClickListener((v) → {
            //amène sur la page de modification lors du clic sur le bouton et envoie les différentes
            // valeurs pour permettre d'afficher les valeurs par défaut du médicament sélectionné sur la page de modification
            Intent intent = new Intent(v.getContext(), formMedicamentModifier.class);
            intent.putExtra( name: "id",leMedicament.getId());
            intent.putExtra( name: "posFamille",posFamille);
            intent.putExtra( name: "nomCommercial",leMedicament.getNomCommercial());
            intent.putExtra( name: "effets",leMedicament.getEffets());
            intent.putExtra( name: "contreIndications",leMedicament.getContreIndications());
            intent.putExtra( name: "composition",leMedicament.getComposition());
            startActivity(intent);
    });

    //listener sur le bouton de suppression
    supprimerBouton.setOnClickListener((v) → {
            //appel de la méthode deleteMedicament de la classe MedicamentDAO qui prend en paramètre un objet medicament
            medicamentAcces.deleteMedicament(leMedicament);
            Toast.makeText(getApplicationContext(),  text: "Le médicament a été supprimé.", Toast.LENGTH_LONG).show();
            //retourne sur la page d'accueil après la suppression
            Intent intent = new Intent(v.getContext(), Accueil.class);
            startActivity(intent);
    });

}
}
```

L'ACTIVITE « FORMMEDICAMENTAJOUTER »

```
public class formMedicamentAjouter extends AppCompatActivity {
    //déclaration des variables
    private Button ajoutMedicButton;
    private ArrayList<Famille> listeFamilles;
    private Spinner spinFamilles;
    private EditText ETnomCommercial;
    private EditText ETcomposition;
    private EditText ETeffets;
    private EditText ETcontreIndications;
    private EditText ETid;
    private Medicament leMedic;
    private String laFamille;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.medicament_ajouter);

        //lien avec le fichier XML
        ajoutMedicButton=(Button)findViewById(R.id.ajouterMedic);
        ETnomCommercial=(EditText)findViewById(R.id.nomcommercial);
        ETcomposition=(EditText)findViewById(R.id.composotion);
        ETeffets=(EditText)findViewById(R.id.effets);
        ETcontreIndications=(EditText)findViewById(R.id.contreindications);
        ETid=(EditText)findViewById(R.id.id);

        //valorisation de la listeFamilles grace à la méthode getFamilles de familleDAO
        FamilleDAO famillesAcces = new FamilleDAO( ct: this);
        final MedicamentDAO medicamentDAO=new MedicamentDAO( ct: this);
        listeFamilles = famillesAcces.getFamilles();

        //lien avec le fichier XML
        spinFamilles = (Spinner) findViewById(R.id.spinFamille);
        ArrayAdapter<String> spinFamillesAdapter = new ArrayAdapter<~>(this.getBaseContext(), android.R.layout.simple_spinner_item);
        spinFamillesAdapter.clear();

        //remplissage de l'adapter
        for (int i = 0; i < listeFamilles.size(); i++) {
            spinFamillesAdapter.add(listeFamilles.get(i).getLibelle());
        }

        //applique l'adapteur au spinner
        spinFamilles.setAdapter(spinFamillesAdapter);
```

```
    //listener sur le spinner famille
    spinFamilles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
            // TODO Auto-generated method stub
            //a chaque changement de famille, on récupère l'id de la famille selectionnée, qui sera nécessaire lors de l'insertion dans la base
            laFamille=listeFamilles.get(arg2).getId();
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });

    //listener sur le bouton d'ajout
    ajoutMedicButton.setOnClickListener((v) -> {
            // TODO Auto-generated method stub
            //création de l'objet médicament avec les infos sélectionnées
            leMedic=new Medicament(ETid.getText().toString(),ETnomCommercial.getText().toString(),ETcomposition.getText()
                    .toString(),ETeffets.getText().toString(),ETcontreIndications.getText().toString(),laFamille);
            //test pour vérifier que l'utilisateur a bien saisi un ID
            if (leMedic.getId().equals("")){
                Toast.makeText(getApplicationContext(), text: "Veuillez saisir un ID.", Toast.LENGTH_LONG).show();
            }else{
                //test pour vérifier que l'utilisateur a bien saisi un nom commercial
                if (leMedic.getNomCommercial().equals("")){
                    Toast.makeText(getApplicationContext(), text: "Veuillez saisir un nom commercial.", Toast.LENGTH_LONG).show();
                } else {
                    //test pour vérifier que l'ID saisi n'est pas déjà dans la base grave à la méthode testIdPresent de medicamentDAO
                    if(medicamentDAO.testIdPresent(leMedic)){
                        Toast.makeText(getApplicationContext(), text: "L'ID est déjà présent dans le base.", Toast.LENGTH_LONG).show();
                    }else{
                        //si les tests précédent sont validés, le medicament est ajouté dans la base avec laa methode addMedicament de MedicamentDAO
                        medicamentDAO.addMedicament(leMedic);
                        Toast.makeText(getApplicationContext(), text: "Le médicament a été ajouté.", Toast.LENGTH_LONG).show();
                        //retourne sur la page d'accueil après l'ajout du médicament
                        Intent intent = new Intent(v.getContext(), Accueil.class);
                        startActivity(intent);
                    }
                }
            }
    });
}
}
```

L'ACTIVITE « FORMMEDICAMENTMODIFIER »

```
public class formMedicamentModifier extends AppCompatActivity {
    //déclaration des variables
    private Button modifMedicButton;
    private ArrayList<Famille> listeFamilles;
    private Spinner spinFamilles;
    private EditText ETnomCommercial;
    private EditText ETcomposition;
    private EditText ETeffets;
    private EditText ETcontreIndications;
    private Medicament leMedic;
    private String laFamille;
    private String lId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.medicament_modifier);

        //lien avec le fichier XML
        modifMedicButton=(Button)findViewById(R.id.modifierMedic);
        ETnomCommercial=(EditText)findViewById(R.id.nomcommercial);
        ETcomposition=(EditText)findViewById(R.id.composition);
        ETeffets=(EditText)findViewById(R.id.effets);
        ETcontreIndications=(EditText)findViewById(R.id.contreindications);

        //valeurs par défaut récupéré dans le intent
        ETnomCommercial.setText(getIntent().getStringExtra( name: "nomCommercial"));
        ETcomposition.setText(getIntent().getStringExtra( name: "composition"));
        ETeffets.setText(getIntent().getStringExtra( name: "effets"));
        ETcontreIndications.setText(getIntent().getStringExtra( name: "contreIndications"));
        lId=getIntent().getStringExtra( name: "id");

        //valorisation de la listeFamilles grace à la méthode getFamilles de familleDAO
        FamilleDAO famillesAcces = new FamilleDAO( ct: this);
        final MedicamentDAO medicamentDAO=new MedicamentDAO( ct: this);
        listeFamilles = famillesAcces.getFamilles();

        //lien avec le fichier XML
        spinFamilles = (Spinner) findViewById(R.id.spinFamille);
        ArrayAdapter<String> spinFamillesAdapter = new ArrayAdapter
                <String>(this.getBaseContext(), android.R.layout.simple_spinner_item);
        spinFamillesAdapter.clear();

        //remplissage de l'adapter
        for (int i = 0; i < listeFamilles.size(); i++) {
            spinFamillesAdapter.add(listeFamilles.get(i).getLibelle());
        }

        //applique l'adapteur au spinner
        spinFamilles.setAdapter(spinFamillesAdapter);
```

```
        //valeur par défaut spinner, la famille du médicament sera sélectionnée
        // par défaut grace à la valeur de position dan la liste récupérée dans le intent
        spinFamilles.setSelection(getIntent().getIntExtra( name: "posFamille", defaultValue: 0));

        //listener sur le spinner famille
        spinFamilles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                // TODO Auto-generated method stub
                //a chaque changement de famille, on récupère l'id de la famille selectionnée,
                // qui sera nécessaire lors de l'update dans la base
                laFamille=listeFamilles.get(arg2).getId();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        modifMedicButton.setOnClickListener((v) -> {
                // TODO Auto-generated method stub
                //création de l'objet médicament avec les infos sélectionnées
                leMedic=new Medicament(lId,ETnomCommercial.getText().toString(),ETcomposition.getText()
                        .toString(),ETeffets.getText().toString(),ETcontreIndications.getText().toString(),laFamille);
                //test pour vérifier que l'utilisateur a bien saisi un nom commercial
                if (leMedic.getNomCommercial().equals("")){
                    Toast.makeText(getApplicationContext(), text: "Veuillez saisir un nom commercial.", Toast.LENGTH_LONG).show();
                } else {
                    //modification du médicament sélectionné avec la methode updateMedicament de MedicamentDAO
                    medicamentDAO.updateMedicament(leMedic);
                    Toast.makeText(getApplicationContext(), text: "Le médicament a été modifié.", Toast.LENGTH_LONG).show();
                    //retourne sur la page d'accueil après l'ajout du médicament
                    Intent intent = new Intent(v.getContext(), Accueil.class);
                    startActivity(intent);
                }
        });
    }
}
```

L'ACTIVITE « FORMPRATICIEN »

```
public class formPraticien extends AppCompatActivity {
    /*
        Classe permettant d'afficher les informations d'un praticien en fonction de son role
     */

    // Déclaration des variables de classe
    private Spinner spinRoles;
    private Spinner spinPraticiens;
    private ArrayList<Role> listeRole;
    private ArrayList<Praticien> listePraticien;
    private TextView nom;
    private TextView prenom;
    private TextView adresse;
    private TextView telephone;
    private TextView specialiteComplementaire;
    private TextView departement;
    private Integer idRole;
    private Button btnModifierPraticien;
    private Button btnAjouterPraticien;
    private Button btnSupprimerPraticien;
    private Praticien unPraticien;
    private Integer positionRole;
    private Integer id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /*
            Procédure de classe qui initialise l'activité
        */
        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);

        // Méthode associant cette page au fichier xml praticien
        setContentView(R.layout.praticien);

        // Récupération des widgets de l'interface utilisateur(xml)
        spinRoles=(Spinner) findViewById(R.id.role);
        spinPraticiens=(Spinner) findViewById(R.id.praticiens);
        nom=(TextView) findViewById(R.id.nomP);
        prenom=(TextView) findViewById(R.id.prenomP);
        adresse=(TextView) findViewById(R.id.adrP);
        telephone=(TextView) findViewById(R.id.telP);
        specialiteComplementaire=(TextView) findViewById(R.id.speP);
        departement=(TextView) findViewById(R.id.depP);
        btnModifierPraticien = (Button) findViewById(R.id.modifierP);
        btnAjouterPraticien = (Button) findViewById(R.id.ajouterP);
        btnSupprimerPraticien = (Button) findViewById(R.id.supprimerP);
```

```
// Récupération des roles dans la base de donnée
RoleDAO roleAcces = new RoleDAO( ct: this);
listeRole = roleAcces.getRole();
ArrayAdapter<String> spinRoleAdapter = new ArrayAdapter<~>
        (this.getBaseContext(),android.R.layout.simple_spinner_item);

//Parcours des roles et ajout de ceux-ci dans l'adapteur
for(int i=0;i<listeRole.size();i++){
    spinRoleAdapter.add(listeRole.get(i).getLibelle());
}

//Ajout de l'adapteur dans le spinner(la liste déroulante)
spinRoles.setAdapter(spinRoleAdapter);

//Création d'un objet de la classe praticien DAO --> lien avec la BDD
final PraticienDAO praticienAcces = new PraticienDAO( ct: this);

//Adapter qui fournit une vue pour chaque objet de le collection inclus dans le spinner(liste déroulante)
final ArrayAdapter<String> spinPraticienAdapter = new ArrayAdapter
        <String>(this.getBaseContext(),android.R.layout.simple_spinner_item);

//Ecoute sur le spinner role
spinRoles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    //Ecoute sur l'item du spinner role
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {
        idRole = listeRole.get(arg2).getId();
        positionRole=arg2;
        //Liste des praticiens en fonction du role
        listePraticien = praticienAcces.getPraticiensParRole(idRole);
        //On efface les anciennes données inutiles
        spinPraticienAdapter.clear();
        for(int i=0;i<listePraticien.size();i++){
            spinPraticienAdapter.add(listePraticien.get(i).getNom()+ " " +listePraticien.get(i).getPrenom());
        }
        //Ajout des praticiens dans le spinner praticien
        spinPraticiens.setAdapter(spinPraticienAdapter);
```

```
    //Ecoute sur le spinner praticien
    spinPraticiens.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
        @Override
        //Ecoute sur l'item du spinner praticien
        public void onItemSelected(AdapterView<?> arg0, View arg1,
                                   int arg2, long arg3) {
                unPraticien = listePraticien.get(arg2);
                //Affichage des informations du praticien selectionné
               id=unPraticien.getId();
               nom.setText(unPraticien.getNom());
               prenom.setText(unPraticien.getPrenom());
               adresse.setText(unPraticien.getAdresse());
               telephone.setText("0"+unPraticien.getTel().toString());
                if (unPraticien.getSpecialiteComplementaire()!=null){
                    specialiteComplementaire.setText(unPraticien.getSpecialiteComplementaire());
                }
                else{
                    System.out.println("Le champ spécialité complémentaire est à null  ");
                    specialiteComplementaire.setText("Aucune");
                   }
               departement.setText(unPraticien.getDepartement().toString());

        }

        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
            // TODO Auto-generated method stub

        }
    });
}

@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub

}
});
```

```
    //Ecoute du boutton modifier
    btnModifierPraticien.setOnClickListener((v) → {
            // TODO Auto-generated method stub
            //Envoie dans la page formPraticienModifier des informations du praticien selectionné
            Intent intent = new Intent(v.getContext(), formPraticienModifier.class);
            intent.putExtra( name: "id", id);
            intent.putExtra( name: "nom", nom.getText().toString());
            intent.putExtra( name: "prenom", prenom.getText().toString());
            intent.putExtra( name: "adresse", adresse.getText().toString());
            intent.putExtra( name: "tel", telephone.getText().toString());
            intent.putExtra( name: "spe", specialiteComplementaire.getText().toString());
            intent.putExtra( name: "dep", departement.getText().toString());
            intent.putExtra( name: "posRole", positionRole);
            System.out.println("positionRole = "+positionRole);
            startActivity(intent);
    });

    //Ecoute du boutton ajouter
    btnAjouterPraticien.setOnClickListener((v) → {
            // TODO Auto-generated method stub
            //Envoie dans la page formPraticienAjouter de l'id du praticien selectionné
            Intent intent = new Intent(v.getContext(), formPraticienAjouter.class);
            intent.putExtra( name: "id", id);
            startActivity(intent);
    });

    //Ecoute du boutton supprimer
    btnSupprimerPraticien.setOnClickListener((v) → {
            // TODO Auto-generated method stub
            //Supression du praticien selectionné et retour à la page d'accueil
            Toast.makeText(getApplicationContext(), text: "Le praticien "
                    + unPraticien.getPrenom()+" a été supprimé.", Toast.LENGTH_LONG).show();
            praticienAcces.deletePraticien(unPraticien);
            Intent intent = new Intent(v.getContext(), Accueil.class);
            startActivity(intent);
    });


}
}
```

L'ACTIVITE « FORMPRATICIENAJOUTER »

```
public class formPraticienAjouter extends AppCompatActivity {
    /*
       Classe permettant d'ajouter un praticien
    */

    // Déclaration des variables de classe
    private Role selectedRole;
    private Spinner spinRoles;
    private ArrayList<Role> listeRole;
    private EditText nom;
    private EditText prenom;
    private EditText adresse;
    private EditText telephone;
    private EditText specialite;
    private EditText departement;
    private Button btnAjouterPraticien;
    private Integer idRole;
    private Integer  telephoneInt;
    private Integer  departementInt;
    private PraticienDAO  praticienAcces;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /*
            Procédure de classe qui initialise l'activité
        */
        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);

        // Méthode associant cette page au fichier xml praticien_ajouter
        setContentView(R.layout.praticien_ajouter);

        // Récupération des widgets de l'interface utilisateur(xml)
        nom=(EditText) findViewById(R.id.nomPra);
        prenom=(EditText) findViewById(R.id.prenomPra);
        adresse=(EditText) findViewById(R.id.adrPra);
        telephone=(EditText) findViewById(R.id.telPra);
        specialite=(EditText) findViewById(R.id.spePra);
        departement=(EditText) findViewById(R.id.depPra);
        btnAjouterPraticien = (Button) findViewById(R.id.ajouterP);
        spinRoles=(Spinner) findViewById(R.id.rolePra);

        // Récupération des roles dans la base de donnée
        RoleDAO roleAcces = new RoleDAO( ct: this);
        listeRole = roleAcces.getRole();
        ArrayAdapter<String> spinRoleAdapter = new ArrayAdapter<~>
                (this.getBaseContext(),android.R.layout.simple_spinner_item);

        //Création d'un objet de la classe praticien DAO --> lien avec la BDD
        final PraticienDAO praticienAcces = new PraticienDAO( ct: this);
```

```
//Parcours des roles et ajout de ceux-ci dans l'adapteur
for(int i=0;i<listeRole.size();i++){
    spinRoleAdapter.add(listeRole.get(i).getLibelle());
}
spinRoles.setAdapter(spinRoleAdapter);

//Ecoute sur le spinner role
spinRoles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    //Ecoute sur l'item du spinner role
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {
        idRole = listeRole.get(arg2).getId();

    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }

});
```

```
    //Ecoute du boutton ajouter
    btnAjouterPraticien.setOnClickListener((v) → {
            // TODO Auto-generated method stub

            try {
                telephoneInt = Integer.parseInt(telephone.getText().toString());
                departementInt = Integer.parseInt(departement.getText().toString());

                Praticien unPraticien = new Praticien(praticienAcces.dernierId(), nom.getText()
                        .toString(), prenom.getText().toString(), adresse.getText().toString(),
                        telephoneInt, specialite.getText().toString(), departementInt, idRole);

                //Vérification que le praticien existe déja
                boolean exister = praticienAcces.ExistePraticien(unPraticien);
                System.out.println(" le praticien est : " + unPraticien.getNom() + " " + unPraticien.getPrenom());
                if (exister) {
                    Toast.makeText(getApplicationContext(), text: "Ce praticien existe déja. "
                            , Toast.LENGTH_LONG).show();
                    nom.setText(" ");
                    prenom.setText(" ");
                    adresse.setText(" ");
                    telephone.setText(" ");
                    specialite.setText(" ");
                    departement.setText(" ");
                } else {
                    //Ajout du praticien dans la BDD
                    long ajoute = praticienAcces.addPraticien(unPraticien);
                    Toast.makeText(getApplicationContext(), text: "Le praticien " + unPraticien.getNom() + " "
                            + unPraticien.getPrenom() + " a été ajouté. ", Toast.LENGTH_LONG).show();
                    Intent intent = new Intent(v.getContext(), Accueil.class);
                    startActivity(intent);
                }
            }catch(NumberFormatException error) {
                System.out.println("Impossible de parser  " + error);
                Toast.makeText(getApplicationContext(), text: "Veuillez saisir des informations : Le numéro de " +
                        "téléphone et le département sont des nombres entiers", Toast.LENGTH_LONG).show();
            }


    });

}
}
```

L'ACTIVITE « FORMPRATICIENMODIFIER »

```
public class formPraticienModifier extends AppCompatActivity {
    /*
        Classe permettant de mofidier les informations d'un praticien
     */

    // Déclaration des variables de classe

    private Spinner spinRoles;
    private ArrayList<Role> listeRole;
    private EditText nom;
    private EditText prenom;
    private EditText adresse;
    private EditText telephone;
    private EditText specialite;
    private EditText departement;

    private String nomP;
    private String prenomP;
    private String adresseP;
    private String telephoneP;
    private String specialiteP;
    private String departementP;
    private Integer positionRole;
    private Integer idP;
    private Integer idRole;

    private Button btnModifierPraticien;

    private PraticienDAO praticienAcces;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /*
            Procédure de classe qui initialise l'activité
        */

        // Méthode permettant de récupérer l'activité en cas de crash de l'application
        super.onCreate(savedInstanceState);

        // Méthode associant cette page au fichier xml praticien_modifier
        setContentView(R.layout.praticien_modifier);



        //Récupération des informations du praticien selectionné de la page formPraticien
        idP=getIntent().getIntExtra( name: "id", defaultValue: 0);
        nomP=getIntent().getStringExtra( name: "nom");
        prenomP=getIntent().getStringExtra( name: "prenom");
        adresseP=getIntent().getStringExtra( name: "adresse");
```

```
adresseP=getIntent().getStringExtra( name: "adresse");
telephoneP=getIntent().getStringExtra( name: "tel");
specialiteP=getIntent().getStringExtra( name: "spe");
departementP=getIntent().getStringExtra( name: "dep");
positionRole=getIntent().getIntExtra( name: "posRole", defaultValue: 0);

// Récupération des widgets de l'interface utilisateur(xml)
btnModifierPraticien = (Button) findViewById(R.id.modifierP);
spinRoles=(Spinner) findViewById(R.id.rolePra);
nom=(EditText) findViewById(R.id.nomPra);
prenom=(EditText) findViewById(R.id.prenomPra);
adresse=(EditText) findViewById(R.id.adrPra);
telephone=(EditText) findViewById(R.id.telPra);
specialite=(EditText) findViewById(R.id.spePra);
departement=(EditText) findViewById(R.id.depPra);
btnModifierPraticien = (Button) findViewById(R.id.modifierP);
spinRoles=(Spinner) findViewById(R.id.rolePra);

//Affichage des informations du praticien sélectionné précédemment
nom.setText(nomP);
prenom.setText(prenomP);
adresse.setText(adresseP);
telephone.setText(telephoneP);
specialite.setText(specialiteP);
departement.setText(departementP);

// Récupération des roles dans la base de donnée
RoleDAO roleAcces = new RoleDAO( ct: this);
listeRole = roleAcces.getRole();
final ArrayAdapter<String> spinRoleAdapter = new ArrayAdapter<~>(this.getBaseContext(),android.R.layout.simple_spinner_item);

//Création d'un objet de la classe praticien DAO --> lien avec la BDD
final PraticienDAO praticienAcces = new PraticienDAO( ct: this);

//Parcours des roles et ajout de ceux-ci dans l'adapteur
for(int i=0;i<listeRole.size();i++){
    spinRoleAdapter.add(listeRole.get(i).getLibelle());
}
spinRoles.setAdapter(spinRoleAdapter);

//Positionnement du role du praticien sélectionné dans le spinner rôle
spinRoles.setSelection(positionRole);
```

```
        //Ecoute sur le spinner role
        spinRoles.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
            @Override
            //Ecoute sur l'item du spinner role
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                                       int arg2, long arg3) {
                idRole = listeRole.get(arg2).getId();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }

        });

        //Ecoute du boutton modifier
        btnModifierPraticien.setOnClickListener((v) -> {
                // TODO Auto-generated method stub
                //On essaye de créer un praticien avec les informations entrée
                try {
                    int telephoneInt = Integer.parseInt(telephone.getText().toString());
                    int departementInt = Integer.parseInt(departement.getText().toString());

                    Praticien unPraticien = new Praticien(idP, nom.getText().toString(), prenom.getText()
                            .toString(), adresse.getText().toString(), telephoneInt, specialite.getText()
                            .toString(), departementInt, idRole);
                    //Modification du praticien sélectionné et retour à la page d'accueil
                    int modifier = praticienAcces.updatePraticien(unPraticien);
                    if (modifier == -1) {
                        Toast.makeText(getApplicationContext(), text: "Ce praticien n'as pas pu être modifié. "
                                , Toast.LENGTH_LONG).show();

                    } else {
                        Toast.makeText(getApplicationContext(), text: "Le praticien " + nomP + " " + prenomP +
                                " a été modifé. ", Toast.LENGTH_LONG).show();
                        Intent intent = new Intent(v.getContext(), Accueil.class);
                        startActivity(intent);
                    }
                }catch(NumberFormatException error) {
                    //On lève l'erreur dans le cas ou le téléphone et le département ne sont pas des entiers
                    System.out.println("Impossible de parser  " + error);
                    Toast.makeText(getApplicationContext(), text: "Veuillez saisir des informations cohérentes:" +
                            " Le numéro de téléphone et le département sont des nombres entiers", Toast.LENGTH_LONG).show();
                }


        });

    }
}
```

L'ACTIVITE « FORMRAPPORT »

```
public class formRapport extends AppCompatActivity{

    //Variables

    private Spinner spinPraticien;
    private ArrayList<Praticien> listePraticien;
    private PraticienDAO praticienAcces = new PraticienDAO( ct: this);
    private Spinner spinRapport;
    private ArrayList<Rapport> listeRapport;
    private RapportDAO RapportAcces = new RapportDAO( ct: this);
    private VisiteurDAO visiteurAcces = new VisiteurDAO( ct: this);
    private MotifDAO motifAcces = new MotifDAO( ct: this);
    private ArrayAdapter<String> spinRapportsAdapter;

    private TextView date_visite_valeur;
    private TextView motif_valeur;
    private TextView bilan_valeur;
    private TextView nom_prenom_visiteur_valeur;
    private TextView nom_prenom_medecin_visite_valeur;
    private Button boutonAjouter;
    private Button boutonModifier;
    private Button boutonSupprimer;

    private Integer idRapport;
    private Integer idMotif;
    private String messageVisiteur;
    private String messagePraticien;
    private Integer idPraticien;
    private String idVisiteur;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rapport_visite);

        //lien avec le fichier XML

        date_visite_valeur=(TextView) findViewById(R.id.date_visite_valeur);
        motif_valeur=(TextView) findViewById(R.id.motif_valeur);
        bilan_valeur=(TextView) findViewById(R.id.bilan_valeur);
        nom_prenom_visiteur_valeur=(TextView) findViewById(R.id.nom_prenom_visiteur_valeur);
        nom_prenom_medecin_visite_valeur=(TextView) findViewById(R.id.nom_prenom_medecin_visite_valeur);
        boutonSupprimer=(Button) findViewById(R.id.supprimer);
        boutonAjouter=(Button) findViewById(R.id.ajouter);
        boutonModifier=(Button) findViewById(R.id.modifier);
        spinPraticien = (Spinner) findViewById(R.id.spinPraticien);
        spinRapport = (Spinner) findViewById(R.id.spinCode_rapport);

        //création de l'adapteur utilisé dans l'écoute du premier spinner
        spinRapportsAdapter = new ArrayAdapter<String>(this.getBaseContext(),android.R.layout.simple_spinner_item);
```

```
//récupère tous les praticiens de la BDD
listePraticien = praticienAcces.getPraticiens();

//création de l'adapteur
ArrayAdapter<String> spinPraticienAdapter = new ArrayAdapter<~>(this.getBaseContext(),android.R.layout.simple_spinner_item);

//remplit l'adapteur
for(int i=0;i<listePraticien.size();i++){
    spinPraticienAdapter.add(listePraticien.get(i).getNom());
}

//applique l'adapteur au spinner
spinPraticien.setAdapter(spinPraticienAdapter);

//gestion de la valeur du spinner
spinPraticien.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {

        //récupère tous les rapports concernant un praticien
        listeRapport = RapportAcces.getRapport(listePraticien.get(arg2).getId());

        //on vide l'adapteur et on le rerempli avec tous les rapports préparés
        spinRapportsAdapter.clear();
        for(int i=0;i<listeRapport.size();i++){
            spinRapportsAdapter.add(listeRapport.get(i).getIdR().toString());
        }
        //applique l'adapteur au spinner
        spinRapport.setAdapter(spinRapportsAdapter);

        //remet tous les TextView à vide
        date_visite_valeur.setText("");
        motif_valeur.setText("");
        bilan_valeur.setText("");
        nom_prenom_visiteur_valeur.setText("");
        nom_prenom_medecin_visite_valeur.setText("");

        //gestion de la valeur du spinner
        spinRapport.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1,
                                       int arg2, long arg3) {

                //récupère informations utiles pour 'Modifier'
                idRapport=listeRapport.get(arg2).getIdR();
                idMotif=listeRapport.get(arg2).getIdMotif();
                idVisiteur=listeRapport.get(arg2).getIdVisiteur();
                idPraticien=listeRapport.get(arg2).getIdPraticien();
```

```
                //rempli les TextView avec les informations du rapport sélectionné
                date_visite_valeur.setText(listeRapport.get(arg2).getDate());
                String messageMotif=motifAcces.getMotif(idMotif).getLibelle();
                motif_valeur.setText(messageMotif);
                bilan_valeur.setText(listeRapport.get(arg2).getBilan());
                messageVisiteur=visiteurAcces.getVisiteur(idVisiteur).getNom()+" "+visiteurAcces
                        .getVisiteur(idVisiteur).getPrenom();
                nom_prenom_visiteur_valeur.setText(messageVisiteur);
                messagePraticien=praticienAcces.getPraticien(idPraticien).getNom()+" "+
                        praticienAcces.getPraticien(idPraticien).getPrenom();
                nom_prenom_medecin_visite_valeur.setText(messagePraticien);
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {

            }
        });
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {

    }
});

//gestion du bouton 'Ajouter' et envoie vers la page portant ce nom
boutonAjouter.setOnClickListener((v) -> {
        Intent intent = new Intent(v.getContext(), formRapportAjouter.class);
        startActivity(intent);
});

//gestion du bouton 'Modifier'
boutonModifier.setOnClickListener((v) -> {
        Intent intent=null;
        //si il cherche à modifier un de ses rapports, sinon reste sur la page et message d'erreur
        if(getIntent().getStringExtra( name: "idVisiteurConnecte").equals(idVisiteur.toString())){
            //remplissage du intent avec toutes les informations sur le rapport
            intent = new Intent(v.getContext(), formRapportModifier.class);
            intent.putExtra( name: "date",date_visite_valeur.getText().toString());
            intent.putExtra( name: "motif",idMotif);
            intent.putExtra( name: "bilan",bilan_valeur.getText().toString());
            intent.putExtra( name: "visiteur",messageVisiteur);
            intent.putExtra( name: "praticien",messagePraticien);
            intent.putExtra( name: "idRapport",idRapport);
            intent.putExtra( name: "idVisiteur",idVisiteur);
            intent.putExtra( name: "idPraticien",idPraticien);
            startActivity(intent);
        }else{
            Toast.makeText(getApplicationContext(),  text: "Vous ne pouvez modifier que vos rapports", Toast.LENGTH_LONG).show();
        }
```

```
        //gestion du bouton 'Supprimer'
        boutonSupprimer.setOnClickListener((v) → {

                //suppression du rapport sélectionné
                long ok = RapportAcces.deleteRapport(idRapport);

                //si suppression échoué, reste sur la page et message d'errreur sinon
                // retour à l'accueil et message de confirmation
                if(ok==-1){
                    Toast.makeText(getApplicationContext(), text: "Une erreur est survenue, " +
                            "merci de contacter le service applicatif.", Toast.LENGTH_LONG).show();
                }else{
                    Intent intent = new Intent(v.getContext(), Accueil.class);
                    Toast.makeText(getApplicationContext(), text: "La rapport a bien été " +
                            "supprimé.", Toast.LENGTH_LONG).show();
                    startActivity(intent);
                }
        });
    }
}
```

L'ACTIVITE « FORMRAPPORTAJOUTER »

```
public class formRapportAjouter extends AppCompatActivity{

    //Variables

    private Spinner spinMotif;
    private Spinner spinNom_prenom_visiteur;
    private Spinner spinNom_prenom_medecin_visite;

    private PraticienDAO praticienAcces = new PraticienDAO( ct: this);
    private VisiteurDAO visiteurAcces = new VisiteurDAO( ct: this);
    private MotifDAO motifAcces = new MotifDAO( ct: this);
    private RapportDAO rapportAcces = new RapportDAO( ct: this);

    private EditText date_visite_valeur;
    private EditText bilan_valeur;
    private Button BtnAjouter;

    private Integer lIDMotif;
    private String lIDVisiteur;
    private Integer lIDPraticien;

    private ArrayList<Motif> listeMotif;
    private ArrayList<Praticien> listePraticien;
    private ArrayList<Visiteur> listeVisiteur;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rapport_visite_ajouter);

        //lien avec le fichier XML
        date_visite_valeur=(EditText) findViewById(R.id.date_visite_valeur);
        bilan_valeur=(EditText) findViewById(R.id.bilan_valeur);
        BtnAjouter=(Button) findViewById(R.id.ajouterR);
        spinMotif = (Spinner) findViewById(R.id.spinMotif);
        spinNom_prenom_visiteur = (Spinner) findViewById(R.id.spinNom_prenom_visiteur);
        spinNom_prenom_medecin_visite = (Spinner) findViewById(R.id.spinNom_prenom_medecin_visite);

        //création des adapteurs
        ArrayAdapter<String> spinMotifsAdapter = new ArrayAdapter<~>
                (this.getBaseContext(),android.R.layout.simple_spinner_item);
        ArrayAdapter<String> spinVisiteursAdapter = new ArrayAdapter<~>
                (this.getBaseContext(),android.R.layout.simple_spinner_item);
        ArrayAdapter<String> spinPraticiensAdapter = new ArrayAdapter<~>
                (this.getBaseContext(),android.R.layout.simple_spinner_item);

        //récupère les différentes listes nécessaires
        listeMotif = motifAcces.getMotifs();
        listePraticien = praticienAcces.getPraticiens();
        listeVisiteur = visiteurAcces.getVisiteurs();
```

```
//remplit l'adapteur
for(int i=0;i<listePraticien.size();i++){
    spinPraticiensAdapter.add(listePraticien.get(i).getNom()+" "+listePraticien.get(i).getPrenom());
}
//applique l'adapteur au spinner
spinNom_prenom_medecin_visite.setAdapter(spinPraticiensAdapter);

//remplit l'adapteur
for(int i=0;i<listeMotif.size();i++){
    spinMotifsAdapter.add(listeMotif.get(i).getLibelle());
}
//applique l'adapteur au spinner
spinMotif.setAdapter(spinMotifsAdapter);

//remplit l'adapteur
for(int i=0;i<listeVisiteur.size();i++){
    spinVisiteursAdapter.add(listeVisiteur.get(i).getNom()+" "+listeVisiteur.get(i).getPrenom());
}
//applique l'adapteur au spinner
spinNom_prenom_visiteur.setAdapter(spinVisiteursAdapter);

//gestion de la valeur du spinner
spinMotif.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {

        //récupération de l'id du motif sélectionné
        lIDMotif=listeMotif.get(arg2).getId();

    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {

    }
});

//gestion de la valeur du spinner
spinNom_prenom_visiteur.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {

        //récupération de l'id du visiteur sélectionné
        lIDVisiteur=listeVisiteur.get(arg2).getId();

    }
```

```
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {

    }
});
//gestion de la valeur du spinner
spinNom_prenom_medecin_visite.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {

        //récupération de l'id du praticien sélectionné
        lIDPraticien=listePraticien.get(arg2).getId();

    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {

    }
});
//gestion du bouton 'Ajouter'
BtnAjouter.setOnClickListener((v) → {

        //teste si les EditText sont vide, si oui message d'erreur sinon ajout nouveau rapport
        if(date_visite_valeur.getText().toString().equals("") || bilan_valeur.getText().toString().equals("")){
            Toast.makeText(getApplicationContext(), text: "Veuillez saisir les informations manquantes.",
                    Toast.LENGTH_LONG).show();
        }else{
            //récupération du fuur id du nouveau rapport
            Integer idMax= rapportAcces.getIdMax()+1;

            //création nouveau rapport
            Rapport nouveauRapport= new Rapport(idMax, date_visite_valeur.getText().toString(), bilan_valeur
                    .getText().toString(), lIDPraticien, lIDVisiteur, lIDMotif);

            //insertion dans la base grâce à la fonction prévue
            long ok = rapportAcces.addRapport(nouveauRapport);

            //si l'insertion a échoué, reste sur la page et message d'errreur sinon retour
            // à l'accueil et message de confirmation
            if(ok==-1){
                Toast.makeText(getApplicationContext(), text: "Une erreur est survenue," +
                                " merci de contacter le service applicatif.",
                        Toast.LENGTH_LONG).show();
            }else{
                Intent intent = new Intent(v.getContext(), Accueil.class);
                Toast.makeText(getApplicationContext(), text: "L'ajout a bien été effectué.",
                        Toast.LENGTH_LONG).show();
                startActivity(intent);
            }
```

```
            }

        });
    }
}
```

L'ACTIVITE « FORMRAPPORTMODIFIER »

```
public class formRapportModifier extends AppCompatActivity{

    //Variables

    private Spinner spinMotif;
    private TextView nom_prenom_visiteur;
    private TextView nom_prenom_medecin_visite;

    private MotifDAO motifAcces = new MotifDAO( ct: this);
    private RapportDAO rapportAcces = new RapportDAO( ct: this);

    private EditText date_visite_valeur;
    private EditText bilan_valeur;
    private Button BtnModifier;

    private Integer lIDMotif;

    private ArrayList<Motif> listeMotif;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rapport_visite_modifier);

        //lien avec le fichier XML
        date_visite_valeur=(EditText) findViewById(R.id.date_visite_valeur);
        bilan_valeur=(EditText) findViewById(R.id.bilan_valeur);
        nom_prenom_visiteur = (TextView) findViewById(R.id.nom_prenom_visiteur_valeur);
        nom_prenom_medecin_visite = (TextView) findViewById(R.id.nom_prenom_medecin_visite_valeur);
        BtnModifier=(Button) findViewById(R.id.modifierR);
        spinMotif = (Spinner) findViewById(R.id.spinMotif);

        //recupération des valeurs et remplissage des EditText
        date_visite_valeur.setText(getIntent().getStringExtra( name: "date"));
        bilan_valeur.setText(getIntent().getStringExtra( name: "bilan"));
        lIDMotif=getIntent().getIntExtra( name: "motif",  defaultValue: -1);
        nom_prenom_visiteur.setText(getIntent().getStringExtra( name: "visiteur"));
        nom_prenom_medecin_visite.setText(getIntent().getStringExtra( name: "praticien"));

        //création de l'adapteur
        ArrayAdapter<String> spinMotifsAdapter = new ArrayAdapter<~>
                (this.getBaseContext(),android.R.layout.simple_spinner_item);

        //récupèration liste des motifs
        listeMotif = motifAcces.getMotifs();
```

```
//remplit l'adapteur avec le motif dans le rapport
for(int i=0;i<listeMotif.size();i++){
    if(lIDMotif==listeMotif.get(i).getId()){
        spinMotifsAdapter.add(listeMotif.get(i).getLibelle());
    }
}
//remplit l'adapteur avec tous les autres motifs
for(int i=0;i<listeMotif.size();i++){
    if(lIDMotif!=listeMotif.get(i).getId()){
        spinMotifsAdapter.add(listeMotif.get(i).getLibelle());
    }
}
//applique l'adapteur au spinner
spinMotif.setAdapter(spinMotifsAdapter);


//gestion de la valeur du spinner
spinMotif.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
    @Override
    public void onItemSelected(AdapterView<?> arg0, View arg1,
                               int arg2, long arg3) {

        //cherche l'id du motif sélectionné
        for(int i=0;i<listeMotif.size();i++){
            if(listeMotif.get(arg2).getId()==listeMotif.get(i).getId()){
                lIDMotif=listeMotif.get(arg2).getId();
            }
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {

    }
});
```

```
    //gestion du bouton 'Modifier'
    BtnModifier.setOnClickListener((v) → {

            //teste si les EditText sont vide, si oui message d'erreur sinon modification rapport
            if(date_visite_valeur.getText().toString().equals("") || bilan_valeur.getText().toString().equals("")){
                Toast.makeText(getApplicationContext(), text: "Veuillez saisir les informations manquantes.", Toast.LENGTH_LONG).show();
            }else{

                //création nouveau rapport
                Rapport nouveauRapport= new Rapport(getIntent().getIntExtra( name: "idRapport",  defaultValue: -1),
                        date_visite_valeur.getText().toString(), bilan_valeur.getText().toString(), getIntent().getIntExtra
                        ( name: "idPraticien", defaultValue: -1), getIntent().getStringExtra( name: "idVisiteur"), lIDMotif);

                //modification dans la base grâce à la fonction prévue
                long ok = rapportAcces.updateRapport(nouveauRapport);

                //si la modification a échoué, reste sur la page et message d'errreur sinon retour
                // à l'accueil et message de confirmation
                if(ok==-1){
                    Toast.makeText(getApplicationContext(), text: "Une erreur est survenue, merci de contacter le service applicatif.",
                            Toast.LENGTH_LONG).show();
                }else{
                    Intent intent = new Intent(v.getContext(), Accueil.class);
                    Toast.makeText(getApplicationContext(), text: "La modification a bien été effectué.", Toast.LENGTH_LONG).show();
                    startActivity(intent);
                }
            }

    });
}
}
```

LA CLASSE « FAMILLE »

```java
package com.example.dimicailledura.medicapp.modele;

/**
 * Created by bast.dubroca on 14/03/2019.
 */

public class Famille {
    private String id;
    private String libelle;

    //constructeur de la classe Famille
    public Famille(String id, String libelle) {
        this.id = id;
        this.libelle = libelle;
    }

    //ensemble des getters et setters
    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getLibelle() { return libelle; }

    public void setLibelle(String libelle) { this.libelle = libelle; }
}
```

LA CLASSE « FAMILLEDAO »

```java
package com.example.dimicailledura.medicapp.modele;

import ...

/**
 * Created by bast.dubroca on 14/03/2019.
 */
public class FamilleDAO {

    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    //constructeur permettant de créer le lien avec la base
    public FamilleDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base, factory: null, version); }

    //fonction permattant de renvoyer un tableau d'objets de la classe Famille
    public ArrayList<Famille> getFamilles(){
        //création du curseur contenant les tuples de la table Famille
        Cursor curseur;
        String req = "select * from Famille";
        curseur = accesBD.getReadableDatabase().rawQuery(req, selectionArgs: null);
        //appel de la fonction cursorToFamillesArrayList en mettant en parametre le curseur créé auparavant
        return cursorToFamillesArrayList(curseur);
    }

    //fonction qui permet de renvoyer un tableau d'objets famille et qui prend en parametre un curseur contenant des tuples de famille
    private ArrayList<Famille> cursorToFamillesArrayList(Cursor curseur){
        ArrayList<Famille> lesFamilles = new ArrayList<~>();
        String idF;
        String libelleF;

        //parcours de chaque tuple du curseur
        curseur.moveToFirst();
        while (!curseur.isAfterLast()){
            idF = curseur.getString( columnIndex: 0);
            libelleF = curseur.getString( columnIndex: 1);
            //création de l'objet Famille avec les données du tuple et ajout de cet objet dans le tableau
            lesFamilles.add(new Famille(idF,libelleF));
            curseur.moveToNext();
        }
        return lesFamilles;
    }
}
```

LA CLASSE « MEDICAMENT »

```
public class Medicament {
    private String id;
    private String nomCommercial;
    private String composition;
    private String effets;
    private String contreIndications;
    private String idFamille;

    //constructeur de la classe Medicament
    public Medicament(String id, String nomCommercial, String composition, String effets,
                      String contreIndications, String idFamille) {
        this.id = id;
        this.nomCommercial = nomCommercial;
        this.composition = composition;
        this.effets = effets;
        this.contreIndications = contreIndications;
        this.idFamille = idFamille;
    }

    //ensemble des getters et setters

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getNomCommercial() { return nomCommercial; }

    public void setNomCommercial(String nomCommercial) { this.nomCommercial = nomCommercial; }

    public String getComposition() { return composition; }

    public void setComposition(String composition) { this.composition = composition; }

    public String getEffets() { return effets; }

    public void setEffets(String effets) { this.effets = effets; }

    public String getContreIndications() { return contreIndications; }

    public void setContreIndications(String contreIndications) {
        this.contreIndications = contreIndications;
    }

    public String getIdFamille() { return idFamille; }

    public void setIdFamille(String idFamille) { this.idFamille = idFamille; }
}
```

LA CLASSE « MEDICAMENTDAO »

```
public class MedicamentDAO {
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    public MedicamentDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base,  factory: null, version); }

    public ArrayList<Medicament> getMedicaments(String famille){
        Cursor curseur;
        String req = "select * from Medicament where idFamille=(select id from famille where libelle='"+famille+"');";
        curseur = accesBD.getReadableDatabase().rawQuery(req, selectionArgs: null);
        return cursorToMedicamentsArrayList(curseur);
    }

    //fonction qui permet de renvoyer un tableau d'objets Medicament et qui prend en parametre un curseur contenant des tuples de Medicament
    private ArrayList<Medicament> cursorToMedicamentsArrayList(Cursor curseur){
        ArrayList<Medicament> lesMedicaments = new ArrayList<~>();
        String idM;
        String nomCommercialM;
        String compositionM;
        String effetsM;
        String contreIndicationsM;
        String idFamille;

        //parcours de chaque tuple du curseur
        curseur.moveToFirst();
        while (!curseur.isAfterLast()){
            idM = curseur.getString( columnIndex: 0);
            nomCommercialM = curseur.getString( columnIndex: 1);
            compositionM = curseur.getString( columnIndex: 2);
            effetsM = curseur.getString( columnIndex: 3);
            contreIndicationsM = curseur.getString( columnIndex: 4);
            idFamille = curseur.getString( columnIndex: 5);
            //création de l'objet Medicament avec les données du tuple et ajout de cet objet dans le tableau
            lesMedicaments.add(new Medicament(idM,nomCommercialM,compositionM,effetsM,contreIndicationsM,idFamille));
            curseur.moveToNext();
        }
        return lesMedicaments;
    }
```

```
    //fonction permettant de supprimer un medicament dans la base à partir d'un objet medicament passé en paramètre
    public long deleteMedicament(Medicament unMedicament){
        SQLiteDatabase bd = accesBD.getWritableDatabase();
        String id=unMedicament.getId();
        //la fonction va renvoyer le nombre de tuple supprimé ou -1 si il y a une erreur dans la suppression
        long ret=bd.delete( table: "Medicament", whereClause: "id='"+id+"'", whereArgs: null);
        return ret;
        }
}
```

```
//fonction permettant d'ajouter un medicament dans la base à partir d'un objet medicament passé en paramètre
public long addMedicament(Medicament unMedicament){
    SQLiteDatabase bd = accesBD.getWritableDatabase();
    ContentValues value = new ContentValues();
    value.put("id", unMedicament.getId());
    value.put("nomCommercial", unMedicament.getNomCommercial());
    value.put("composition",unMedicament.getComposition());
    value.put("effets",unMedicament.getEffets());
    value.put("contreIndications",unMedicament.getContreIndications());
    value.put("idFamille",unMedicament.getIdFamille());
    //la fonction va renvoyer le nombre de tuple inséré ou -1 si il y a une erreur dans l'ajout
    Long ret = bd.insert( table: "medicament",  nullColumnHack: null, value);
    return ret;
}

//fonction permettant de modifier un medicament dans la base à partir d'un objet medicament passé en paramètre
public int updateMedicament(Medicament unMedicament){
    SQLiteDatabase bd = accesBD.getWritableDatabase();

    ContentValues value = new ContentValues();
    value.put("id", unMedicament.getId());
    value.put("nomCommercial", unMedicament.getNomCommercial());
    value.put("composition",unMedicament.getComposition());
    value.put("effets",unMedicament.getEffets());
    value.put("contreIndications",unMedicament.getContreIndications());
    value.put("idFamille",unMedicament.getIdFamille());
    //la fonction va renvoyer le nombre de tuple modifié ou -1 si il y a une erreur dans la modification
    int ret = bd.update( table: "medicament", value, whereClause: "id='"+unMedicament.getId()+"'", whereArgs: null);
    return ret;
}

//fonction qui renvoie true si l'id du médicament est déjà présent dans la base et false sinon
public boolean testIdPresent(Medicament unMedicament){
    Cursor curseur;
    String id=unMedicament.getId();
    String req = "select count(*) from Medicament where id='"+id+"';";
    curseur = accesBD.getReadableDatabase().rawQuery(req, selectionArgs: null);
    curseur.moveToFirst();
    boolean existe=false;
    if (curseur.getInt( columnIndex: 0)>=1){
        existe=true;
    }
    return existe;
}
```

LA CLASSE « MOTIF»

```
public class Motif {


    private Integer id;
    private String libelle;


    //constructeur
    public Motif(Integer id, String libelle) {
        this.id = id;
        this.libelle = libelle;
    }


    //Getter et Setter
    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public String getLibelle() { return libelle; }

    public void setLibelle(String libelle) { this.libelle = libelle; }
}
```

LA CLASSE « MOTIFDAO»

```
//classe MotifDAO
public class MotifDAO {

    ///lien avec la base
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    //connexion à la base
    public MotifDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base,  factory: null, version); }


    //renvoi tous les motifs
    public ArrayList<Motif> getMotifs(){
        ArrayList<Motif> lesMotifs=new ArrayList<~>();
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Motif;", selectionArgs: null);
        return cursorToMotifArrayList(curseur);
    }

    //renvoi un motif quand on lui donne l'id
    public Motif getMotif(Integer id){
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Motif where id="+id+";", selectionArgs: null);
        curseur.moveToFirst();
        return new Motif(curseur.getInt( columnIndex: 0),curseur.getString( columnIndex: 1));
    }

    //crée un tableau de Motif à partir d'un curseur
    private ArrayList<Motif> cursorToMotifArrayList(Cursor curseur){
        ArrayList<Motif> listeMotif = new ArrayList<~>();
        Integer id;
        String libelle;

        curseur.moveToFirst();
        while (!curseur.isAfterLast()){
            id = curseur.getInt( columnIndex: 0);
            libelle = curseur.getString( columnIndex: 1);
            listeMotif.add(new Motif(id,libelle));
            curseur.moveToNext();
        }

        return listeMotif;
    }


}
```

LA CLASSE « PRATICIEN»

```
public class Praticien {
    /*
        Classe métier permettant de matérialiser un praticien de la BDD
        Contenu : Constructeur et assesseurs en consulation et en modification
     */

    private Integer id;
    private String nom;
    private String prenom;
    private String adresse;
    private Integer tel;
    private String specialiteComplementaire;
    private Integer departement;
    private Integer codeRole;

    public Praticien(Integer id, String nom, String prenom, String adresse, Integer tel,
                     String specialiteComplementaire, Integer departement, Integer codeRole) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
        this.adresse = adresse;
        this.tel = tel;
        this.specialiteComplementaire = specialiteComplementaire;
        this.departement = departement;
        this.codeRole = codeRole;
    }

    public Integer getId() { return id; }
    public void setId(Integer id) { this.id = id; }
    public String getNom() { return nom; }

    public void setNom(String nom) { this.nom = nom; }

    public String getPrenom() { return prenom; }

    public void setPrenom(String prenom) { this.prenom = prenom; }

    public String getAdresse() { return adresse; }

    public void setAdresse(String adresse) { this.adresse = adresse; }
```

```
    public Integer getTel() { return tel; }

    public void setTel(Integer tel) { this.tel = tel; }

    public String getSpecialiteComplementaire() { return specialiteComplementaire; }

    public void setSpecialiteComplementaire(String specialiteComplementaire) {
        this.specialiteComplementaire = specialiteComplementaire;
    }

    public Integer getDepartement() { return departement; }

    public void setDepartement(Integer departement) { this.departement = departement; }

    public Integer getCodeRole() { return codeRole; }

    public void setCodeRole(Integer codeRole) { this.codeRole = codeRole; }

}
```

LA CLASSE « PRATICIENDAO»

```
public class PraticienDAO {
    /*
        Classe permettant d'accéder aux données des praticiens de la BDD
     */
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    public PraticienDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base, factory: null, version); }


    public long addPraticien(Praticien unPraticien){
        /*
            Fonction  qui ajoute un praticen dans la BDD
            retourne -1 si l'odre n'a pas fonctionné et le nombre de tuples insérés sinon
        */
        long ret;
        //Accès à la BDD en mode écriture
        SQLiteDatabase bd = accesBD.getWritableDatabase();
        ContentValues value = new ContentValues();
        value.put("id", dernierId());
        value.put("nom", unPraticien.getNom());
        value.put("prenom",unPraticien.getPrenom());
        value.put("adresse",unPraticien.getAdresse());
        value.put("tel",unPraticien.getTel());
        value.put("specialiteComplementaire",unPraticien.getSpecialiteComplementaire());
        value.put("departement",unPraticien.getDepartement());
        value.put("codeRole",unPraticien.getCodeRole());
        //Exécution de l'ordre SQL d'insertion
        ret = bd.insert( table: "praticien",  nullColumnHack: null, value);
        return ret;
    }


    public long deletePraticien(Praticien unPraticien){
        /*
            Fonction  qui surpprime un praticien de la BDD
            retourne un praticien
        */
        SQLiteDatabase bd = accesBD.getWritableDatabase();
        long ret=bd.delete( table: "praticien", whereClause: "id="+unPraticien.getId(), whereArgs: null);
        return ret;

    }

}
```

```
public boolean ExistePraticien(Praticien unPraticien){
    /*
        Fonction  qui vérifie si un praticen existe dans la BDD
        retourne True si le praticien existe et False sinon
    */
    boolean exister=false;
    //Accès à la BDD en mode lecture
    SQLiteDatabase bd = accesBD.getReadableDatabase();
    ArrayList<Praticien> lesPraticiens= getPraticiens();
    for (Praticien lePraticien: lesPraticiens) {
        if(unPraticien.getNom().equals(lePraticien.getNom())){
            if(unPraticien.getPrenom().equals(lePraticien.getPrenom())){
                if(unPraticien.getTel().equals(lePraticien.getTel())){
                    exister=true;
                }
            }
        }
    }
    return exister;
}


public int updatePraticien(Praticien unPraticien){
    /*
        Fonction  qui met à jour un praticen dans la BDD
        retourne -1 si l'odre n'a pas fonctionné et le nombre de tuples modifiés sinon
    */
    //Accès à la BDD en mode écritue
    SQLiteDatabase bd = accesBD.getWritableDatabase();

    ContentValues value = new ContentValues();
    value.put("id", unPraticien.getId());
    value.put("nom", unPraticien.getNom());
    value.put("prenom",unPraticien.getPrenom());
    value.put("adresse",unPraticien.getAdresse());
    value.put("tel",unPraticien.getTel());
    value.put("specialitecomplementaire",unPraticien.getSpecialiteComplementaire());
    value.put("departement",unPraticien.getDepartement());
    value.put("codeRole",unPraticien.getCodeRole());
    //Exécution de l'ordre SQL de mise à jour
    int ret = bd.update( table: "praticien", value, whereClause: "id = "+unPraticien.getId(), whereArgs: null);

    return ret;
}
```

```
public ArrayList<Praticien> getPraticiensParRole(Integer unIdRole){
    /*
        Fonction  qui récupère une collection de praticiens en fonction du role dans BDD
        retourne une collection de praticiens
    */
    Cursor curseur;
    String req = "select * from praticien where codeRole ="+unIdRole+" order by nom ASC;";
    curseur = accesBD.getReadableDatabase().rawQuery(req, selectionArgs: null);
    return cursorToPraticienArrayList(curseur);
}


private ArrayList<Praticien> cursorToPraticienArrayList(Cursor curseur){
    /*
        Fonction  qui récupère un curseur et implémente les informations liées aux praticiens en fonction du contenu du curseur
        retourne une collection de praticiens
    */
    ArrayList<Praticien> listePraticien = new ArrayList<~>();
    Integer id;
    String nom;
    String prenom;
    String adresse;
    Integer tel;
    String specialiteComplementaire;
    Integer departement;
    Integer codeRole;


    curseur.moveToFirst();
    while (!curseur.isAfterLast()){
        id = curseur.getInt( columnIndex: 0);
        nom = curseur.getString( columnIndex: 1);
        prenom = curseur.getString( columnIndex: 2);
        adresse = curseur.getString( columnIndex: 3);
        tel = curseur.getInt( columnIndex: 4);
        specialiteComplementaire = curseur.getString( columnIndex: 5);
        departement = curseur.getInt( columnIndex: 6);
        codeRole = curseur.getInt( columnIndex: 7);
        listePraticien.add(new Praticien(id,nom,prenom,adresse,tel,specialiteComplementaire,departement,codeRole));
        curseur.moveToNext();
    }

    return listePraticien;
}
```

```
public Integer dernierId(){
    /*
        Fonction  qui récupère l'id le plus grand des praticens et lui ajoute +1 dans la BDD
        retourne l'id le plus petit id disponible pour les praticiens
    */
    Cursor curseur;
    //Accès à la BDD en mode lecture et éxécution de l'odre sql de consultation
    curseur = accesBD.getReadableDatabase().rawQuery( sql: "select max(id)+1 from praticien;", selectionArgs: null);
    curseur.moveToFirst();
    Integer lastId=curseur.getInt( columnIndex: 0);
    return lastId;
}


public ArrayList<Praticien> getPraticiens(){
    /*
        Fonction  qui récupère l'ensemble des praticiens de la BDD
        retourne une collection de praticiens
    */
    ArrayList<Praticien> lesPraticiens=new ArrayList<~>();
    Cursor curseur;
    curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from praticien;", selectionArgs: null);
    return cursorToPraticienArrayList(curseur);
}

public Praticien getPraticien(Integer id){
    /*
        Fonction  qui récupère un praticien en fonction de son id dans BDD
        retourne un praticien
    */
    Cursor curseur;
    curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Praticien where id="+id+";", selectionArgs: null);
    curseur.moveToFirst();
    return new Praticien(curseur.getInt( columnIndex: 0),curseur.getString( columnIndex: 1),curseur.getString( columnIndex: 2),
curseur.getString( columnIndex: 3),curseur.getInt( columnIndex: 4),curseur.getString( columnIndex: 5),curseur.getInt( columnIndex: 6),curseur.getInt( columnIndex: 7));
}
```

LA CLASSE « RAPPORT»

```
public class Rapport {
    private Integer idR;
    private String date;
    private String bilan;
    private Integer idPraticien;
    private String idVisiteur;
    private Integer idMotif;


    public Rapport(Integer idR, String date, String bilan, Integer idPraticien, String idVisiteur, Integer idMotif) {
        this.idR = idR;
        this.date = date;
        this.bilan = bilan;
        this.idPraticien = idPraticien;
        this.idVisiteur = idVisiteur;
        this.idMotif = idMotif;
    }

    public Integer getIdR() { return idR; }

    public void setIdR(Integer idR) { this.idR = idR; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }

    public String getBilan() { return bilan; }

    public void setBilan(String bilan) { this.bilan = bilan; }

    public Integer getIdPraticien() { return idPraticien; }

    public void setIdPraticien(Integer idPraticien) { this.idPraticien = idPraticien; }

    public String getIdVisiteur() { return idVisiteur; }

    public void setIdVisiteur(String idVisiteur) { this.idVisiteur = idVisiteur; }

    public Integer getIdMotif() { return idMotif; }

    public void setIdMotif(Integer idMotif) { this.idMotif = idMotif; }
}
```

LA CLASSE « RAPPORTDAO»

```
public class RapportDAO {
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    public RapportDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base, factory: null, version); }


    public long addRapport(Rapport unRapport){
        long ret;
        SQLiteDatabase bd = accesBD.getWritableDatabase();

        ContentValues value = new ContentValues();
        value.put("id", unRapport.getIdR());
        value.put("date", unRapport.getDate());
        value.put("bilan", unRapport.getBilan());
        value.put("idPraticien", unRapport.getIdPraticien());
        value.put("idVisiteur", unRapport.getIdVisiteur());
        value.put("idMotif", unRapport.getIdMotif());
        ret = bd.insert( table: "Rapport",  nullColumnHack: null, value);

        return ret;
    }

    public ArrayList<Rapport> getRapports(){
        ArrayList<Rapport> lesRapport=new ArrayList<~>();
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Rapport;", selectionArgs: null);
        return cursorToRapportArrayList(curseur);
    }

    public int getIdMax(){
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select max(id) from Rapport;", selectionArgs: null);
        curseur.moveToFirst();
        return curseur.getInt( columnIndex: 0);
    }

    public ArrayList<Rapport> getRapport(Integer idPraticien){
        ArrayList<Rapport> lesRapport=new ArrayList<~>();
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Rapport where idPraticien="+idPraticien+";", selectionArgs: null);
        return cursorToRapportArrayList(curseur);
    }
```

```
private ArrayList<Rapport> cursorToRapportArrayList(Cursor curseur){
    ArrayList<Rapport> listeRapport = new ArrayList<~>();
    Integer idR;
    String date;
    String bilan;
    Integer idPraticien;
    String idVisiteur;
    Integer idMotif;

    curseur.moveToFirst();
    while (!curseur.isAfterLast()){
        idR = curseur.getInt( columnIndex: 0);
        date = curseur.getString( columnIndex: 1);
        bilan = curseur.getString( columnIndex: 2);
        idPraticien = curseur.getInt( columnIndex: 3);
        idVisiteur = curseur.getString( columnIndex: 4);
        idMotif = curseur.getInt( columnIndex: 5);

        listeRapport.add(new Rapport(idR,date,bilan,idPraticien,idVisiteur,idMotif));
        curseur.moveToNext();
    }

    return listeRapport;
}

public int updateRapport(Rapport unRapport){
    SQLiteDatabase bd = accesBD.getWritableDatabase();

    ContentValues value = new ContentValues();
    value.put("id", unRapport.getIdR());
    value.put("date", unRapport.getDate());
    value.put("bilan",unRapport.getBilan());
    value.put("idPraticien",unRapport.getIdPraticien());
    value.put("idVisiteur",unRapport.getIdVisiteur());
    value.put("idMotif",unRapport.getIdMotif());
    int ret = bd.update( table: "rapport", value, whereClause: "id="+unRapport.getIdR(), whereArgs: null);

    return ret;
}

public long deleteRapport(int idRapport){
    SQLiteDatabase bd = accesBD.getWritableDatabase();
    long ret=bd.delete( table: "Rapport", whereClause: "id="+idRapport, whereArgs: null);
    return ret;

}
```

LA CLASSE « ROLE»

```
public class Role {
    /*
        Classe métier permettant de matérialiser un rôle de la BDD
        Contenu : Constructeur et assesseurs en consulation et en modification
     */
    private Integer id;
    private String libelle;

    public Role(Integer id, String libelle) {
        this.id = id;
        this.libelle = libelle;
    }

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public String getLibelle() { return libelle; }

    public void setLibelle(String libelle) { this.libelle = libelle; }
}
```

LA CLASSE « ROLEDAO»

```
public class RoleDAO {
    /*
        Classe permettant d'accéder aux données des rôles de la BDD
     */
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    public RoleDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base,  factory: null, version); }



    public ArrayList<Role> getRole(){
        /*
            Fonction  qui récupère l'enssemble des rôles dans la BDD
            retourne une collection de rôles
        */
        ArrayList<Role> lesRoles=new ArrayList<~>();
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from role;", selectionArgs: null);
        return cursorToRoleArrayList(curseur);
    }



    private ArrayList<Role> cursorToRoleArrayList(Cursor curseur){
       /*
            Fonction  qui récupère un curseur et implémente les informations liées aux rôles en fonction du contenu du curseur
            retourne une collection de rôles
        */
        ArrayList<Role> listeRole = new ArrayList<~>();
        Integer id;
        String libelle;

        curseur.moveToFirst();
        while (!curseur.isAfterLast()){
            id = curseur.getInt( columnIndex: 0);
            libelle = curseur.getString( columnIndex: 1);

            listeRole.add(new Role(id,libelle));
            curseur.moveToNext();
        }

        return listeRole;
    }
```

LA CLASSE « VISITEUR»

```
//classe Visiteur

public class Visiteur {
    private String id;
    private String nom;
    private String prenom;
    private String login;
    private String mdp;
    private String adresse;
    private Integer cp;
    private String ville;
    private String dateEmbauche;

    //constructeur
    public Visiteur(String id, String nom, String prenom, String login, String mdp, String adresse,
                    Integer cp, String ville, String dateEmbauche) {
        this.id = id;
        this.nom = nom;
        this.prenom = prenom;
        this.login = login;
        this.mdp = mdp;
        this.adresse = adresse;
        this.cp = cp;
        this.ville = ville;
        this.dateEmbauche = dateEmbauche;
    }

    //Getter et Setter
    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public String getNom() { return nom; }

    public void setNom(String nom) { this.nom = nom; }

    public String getPrenom() { return prenom; }

    public void setPrenom(String prenom) { this.prenom = prenom; }

    public String getLogin() { return login; }

    public void setLogin(String login) { this.login = login; }

    public String getMdp() { return mdp; }
```

```
    public String getMdp() { return mdp; }

    public void setMdp(String mdp) { this.mdp = mdp; }

    public String getAdresse() { return adresse; }

    public void setAdresse(String adresse) { this.adresse = adresse; }

    public Integer getCp() { return cp; }

    public void setCp(Integer cp) { this.cp = cp; }

    public String getVille() { return ville; }

    public void setVille(String ville) { this.ville = ville; }

    public String getDateEmbauche() { return dateEmbauche; }

    public void setDateEmbauche(String dateEmbauche) { this.dateEmbauche = dateEmbauche; }
}
```

LA CLASSE « VISITEURDAO»

```
//classe VisiteurDAO
public class VisiteurDAO {
    private static String base = "medicapp";
    private static int version = 1;
    BD_SQLiteOpenHelper accesBD;

    //lien avec la base
    public VisiteurDAO(Context ct) { accesBD = new BD_SQLiteOpenHelper(ct, base, factory: null, version); }


    //test si un visiteur existe à partir de son login et du mot de passe
    public Visiteur getVisiteur(String login, String mdp){
        Visiteur unVisiteur=null;
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Visiteur " +
                "where login='"+login+"' and mdp='"+mdp+"';", selectionArgs: null);
        curseur.moveToFirst();
        if (curseur.getCount() > 0) {
            unVisiteur= new Visiteur(curseur.getString( columnIndex: 0),curseur.getString( columnIndex: 1)
                    ,curseur.getString( columnIndex: 2),curseur.getString( columnIndex: 3),curseur.getString( columnIndex: 4),
                    curseur.getString( columnIndex: 5),curseur.getInt( columnIndex: 6),curseur.getString( columnIndex: 7),
                    curseur.getString( columnIndex: 8));
        }
        return unVisiteur;
    }

    //renvoi un Visiteur à partir d'un id
    public Visiteur getVisiteur(String id){
        Cursor curseur;
        curseur = accesBD.getReadableDatabase().rawQuery( sql: "select * from Visiteur where id='"+id+"';", selectionArgs: null);
        curseur.moveToFirst();
        return new Visiteur(curseur.getString( columnIndex: 0),curseur.getString( columnIndex: 1),curseur.getString( columnIndex: 2)
                ,curseur.getString( columnIndex: 3),curseur.getString( columnIndex: 4),curseur.getString( columnIndex: 5),
                curseur.getInt( columnIndex: 6),curseur.getString( columnIndex: 7),curseur.getString( columnIndex: 8));
    }

    //renvoi tous les visiteurs
    public ArrayList<Visiteur> getVisiteurs(){
        Cursor curseur;
        String req = "select * from Visiteur";
        curseur = accesBD.getReadableDatabase().rawQuery(req, selectionArgs: null);
        return cursorToVisiteurArrayList(curseur);
    }
```

```
    //crée un tableau de visiteurs à partir d'un curseur
    private ArrayList<Visiteur> cursorToVisiteurArrayList(Cursor curseur){
        ArrayList<Visiteur> listeVisiteur = new ArrayList<~>();
        String id;
        String nom;
        String prenom;
        String login;
        String mdp;
        String adresse;
        Integer cp;
        String ville;
        String dateEmbauche;

        curseur.moveToFirst();
        while (!curseur.isAfterLast()){
            id = curseur.getString( columnIndex: 0);
            nom = curseur.getString( columnIndex: 1);
            prenom = curseur.getString( columnIndex: 2);
            login = curseur.getString( columnIndex: 3);
            mdp = curseur.getString( columnIndex: 4);
            adresse = curseur.getString( columnIndex: 5);
            cp = curseur.getInt( columnIndex: 6);
            ville = curseur.getString( columnIndex: 7);
            dateEmbauche = curseur.getString( columnIndex: 8);
            listeVisiteur.add(new Visiteur(id,nom,prenom,login,mdp,adresse,cp,ville,dateEmbauche));
            curseur.moveToNext();
        }

        return listeVisiteur;
    }


}
```

LA BASE DE DONNEES

```
public class BD_SQLiteOpenHelper extends SQLiteOpenHelper {

    private String creaTablePraticien="create table praticien ( "
            + "     id integer, "
            + "     nom text, "
            + "     prenom text, "
            + "     adresse text, "
            + "     tel integer, "
            + "     specialiteComplementaire text, "
            + "     departement integer, "
            + "     codeRole integer, "
            + "     primary key (id), "
            + "     foreign key(codeRole) references role(id) "
            + " ); ";


    private String creaTableRole = " create table role ( "
            + "     id integer, "
            + "     libelle text, "
            + "     primary key (id) "
            + " ); ";

    private String creaTableVisiteur = " create table visiteur ( "
            + "     id text, "
            + "     nom text, "
            + "     prenom text, "
            + "     login text, "
            + "     mdp text, "
            + "     adresse text, "
            + "     cp integer, "
            + "     ville text, "
            + "     dateEmbauche text, "
            + "     primary key (id) "
            + " ); ";
```

```
private String creaTableRapport = " create table rapport ( "
        + "     id integer, "
        + "     date text, "
        + "     bilan text, "
        + "     idPraticien integer, "
        + "     idVisiteur text, "
        + "     idMotif integer, "
        + "     primary key (id), "
        + "     foreign key(idPraticien) references praticien(id), "
        + "     foreign key(idVisiteur) references visiteur(id), "
        + "     foreign key(idMotif) references motif(id) "
        + " ); ";

private String creaTableOffrir = " create table offrir ( "
        + "     quantite float, "
        + "     idRapport integer, "
        + "     idMedicament integer, "
        + "     primary key (idRapport, idMedicament), "
        + "     foreign key(idRapport) references rapport(id), "
        + "     foreign key(idMedicament) references medicament(id) "
        + " )";

private String creaTableMedicament = " create table medicament ( "
        + "     id text, "
        + "     nomCommercial text, "
        + "     composition text, "
        + "     effets text, "
        + "     contreIndications text, "
        + "     idFamille text, "
        + "     primary key (id), "
        + "     foreign key(idFamille) references famille(id) "
        + " ); ";

private String creaTableFamille = " create table famille ( "
        + "     id text, "
        + "     libelle text, "
        + "     primary key (id) "
        + " ); ";

private String creaTableMotif = " create table motif ( "
        + "     id integer, "
        + "     libelle text, "
        + "     primary key (id) "
        + " ); ";
```

```
public BD_SQLiteOpenHelper(Context context, String name, CursorFactory factory, int version) {
    super(context, name, factory, version);
    // TODO Auto-generated constructor stub
}

@Override
public void onCreate(SQLiteDatabase db) {
    // TODO Auto-generated method stub

    db.execSQL(creaTableVisiteur);
    db.execSQL(creaTableMotif);
    db.execSQL(creaTableRole);
    db.execSQL(creaTablePraticien);
    db.execSQL(creaTableRapport);
    db.execSQL(creaTableFamille);
    db.execSQL(creaTableMedicament);
    db.execSQL(creaTableOffrir);

    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('a131', 'Aribi', 'Alain', 'a', 'a', '8 rue des Charmes', '46000', 'Cahors', '2005-12-21')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('a17', 'Andre', 'David', 'dandre', 'oppg5', '1 rue Petit', '46200', 'Lalbenque', '1998-11-23')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('a55', 'Bedos', 'Christian', 'cbedos', 'gmhxd', '1 rue Peranud', '46250', 'Montcuq', '1995-01-12')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('a93', 'Tusseau', 'Louis', 'ltusseau', 'ktp3s', '22 rue des Ternes', '46123', 'Gramat', '2000-05-01')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('b13', 'Bentot', 'Pascal', 'pbentot', 'doyw1', '11 allée des Cerises', '46512', 'Bessines', '1992-07-09')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('b16', 'Bioret', 'Luc', 'lbioret', 'hrjfs', '1 Avenue gambetta', '46000', 'Cahors', '1998-05-11')");
    db.execSQL("INSERT INTO visiteur (id, nom, prenom, login, mdp, adresse, cp, ville, dateEmbauche) " +
            "VALUES ('b19', 'Bunisset', 'Francis', 'fbunisset', '4vbnd', '10 rue des Perles', '93100', 'Montreuil', '1987-10-21')");
```

```
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (1,'positif')");
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (2,'Demande du médecin')");
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (3,'recommandation de confrère')");
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (4,'Installation nouvelle')");
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (5,'Visite annuelle')");
db.execSQL("INSERT INTO `motif` (`id`, `libelle`) VALUES (6,'Prise de contact')");

db.execSQL("INSERT INTO `role` (`id`, `libelle`) VALUES (1,'Médecin')");
db.execSQL("INSERT INTO `role` (`id`, `libelle`) VALUES (2,'Pharmacien')");
db.execSQL("INSERT INTO `role` (`id`, `libelle`) VALUES (3,'Infirmier')");
db.execSQL("INSERT INTO `role` (`id`, `libelle`) VALUES (4,'Chef de clinique')");
db.execSQL("INSERT INTO `role` (`id`, `libelle`) VALUES (5,'Hôpital')");

db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (1, 'MARTIN', 'Prosper', '25 rue Anatole France BRIANCON 05100', '0485244174', NULL, 5, 1)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (2, 'BLANC', 'Anne-Lucie', '39 rue des gatinnes BILLIAT 01200', '0356895400', NULL, 1, 2)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (3, 'GARCIA', 'Camille', '58 rue du stade MESSINCOURT 08110', '0365489929', NULL, 8, 3)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (4, 'MARTINEZ', 'Alice', '12 rue des Pigeons JOIGNY-SUR-MEUSE 08700', '0319692016', NULL, 8, 4)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (5, 'MICHEL', 'Vénus', '55 rue du 14 juillet ORCIERES 05170', '0421670911', NULL, 5, 5)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (6, 'ROUX', 'Anne-Lucie', '49 rue des Ormes ATTILLY 02490', '0313817061', NULL, 2, 2)");
db.execSQL("INSERT INTO `praticien` (`id`, `nom`, `prenom`, `adresse`, `tel`, `specialitecomplementaire`, `departement`, `codeRole`) " +
        "VALUES (7, 'FABRE', 'Marie', '78 rue de Poligny YONCQ 08210', '0388716930', NULL, 8, 3)");
```

```
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (1, '2017-01-02', 1, 'visiteannuelle', 1, 'b16')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (2, '2016-07-02', 2, 'Très aimable maintenir un contact régulier', 2, 'a17')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (3, '2016-07-02', 2, 'Trop pressé', 3, 'a93')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (4, '2016-07-02', 3, 'Visite positive', 1, 'a131')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`)" +
        " VALUES (5, '2016-07-02', 3, 'Pas intéressé du tout', 1, 'b19')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (6, '2016-07-02', 4, 'Sur sa réserve', 4, 'b13')");
db.execSQL("INSERT INTO `rapport` (`id`, `date`, `idMotif`, `bilan`, `idPraticien`, `idVisiteur`) " +
        "VALUES (7, '2016-07-02', 5, 'Pas intéressé du tout', 5, 'a131')");

db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('AA', 'Antalgiques en association')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('AAA', 'Antalgiques antipyréques en association')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('AAC', 'Antidépresseur d action centrale')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('AAH', 'Antivertigineux antihistaminique H1')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('ABA', 'Antibiotique antituberculeux')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('ABC', 'Antibiotique antiacnénique local')");
db.execSQL("INSERT INTO `famille` (`id`, `libelle`) VALUES ('ABP', 'Antibiotique de la " +
        "famille des béta-lactamines -pénicilline A-')");
```

```
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('3MYC7', 'TRIMYCINE', 'CRT', 'Triamcinolone acétonide + Néomycine + Nystatine', 'Ce médicament est un corticoïde " +
            "à  activité forte ou très forte associé à  un antibiotique et un ant', 'Ce médicament est contre-indiqué en cas d allergie à " +
            " l un des constituants  d infections de la peau')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('ADIMOL9', 'ADIMOL', 'ABP', 'Amoxicilline + Acide clavulanique', 'Ce médicament  plus puissant que les pénicillines " +
            "simples  est utilisé pour traiter des infections b', 'Ce médicament est contre-indiqué en cas d allergie aux pénicillines ou " +
            "aux céphalosporines.')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('AMOPIL7', 'AMOPIL', 'ABP', 'Amoxicilline', 'Ce médicament  plus puissant que les pénicillines simples  est utilisé pour" +
            " traiter des infections b', 'Ce médicament est contre-indiqué en cas d allergie aux pénicillines. Il doit être administré avec pr')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('AMOX45', 'AMOXAR', 'ABP', 'Amoxicilline', 'Ce médicament  plus puissant que les pénicillines simples  est utilisé pour " +
            "traiter des infections b', 'La prise de ce médicament peut rendre positifs les tests de dépistage du dopage.')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('AMOXIG12', 'AMOXI Gé', 'ABP', 'Amoxicilline', 'Ce médicament  plus puissant que les pénicillines simples  est utilisé " +
            "pour traiter des infections b', 'Ce médicament est contre-indiqué en cas d allergie aux pénicillines. Il doit être administré avec pr')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('APATOUX22', 'APATOUX Vitamine C', 'ALO', 'Tyrothricine + Tétracaïne + Acide ascorbique (Vitamine C)', 'Ce médicament est " +
            "utilisé pour traiter les affections de la bouche et de la gorge.', 'Ce médicament est contre-indiqué en cas d allergie à  l un des " +
            "constituants  en cas de phénylcétonur')");
    db.execSQL("INSERT INTO `medicament` (`id`, `nomCommercial`, `idFamille`, `composition`, `effets`, `contreIndications`) " +
            "VALUES ('BACTIG10', 'BACTIGEL', 'ABC', 'Erythromycine', 'Ce médicament est utilisé en application locale pour traiter l acné et les " +
            "infections cutanées bacté', 'Ce médicament est contre-indiqué en cas d allergie aux antibiotiques de la famille des macrolides ou')");

    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (3, '3MYC7', 3)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (4, 'CLAZER6', 4)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (5, 'AMOXIG12', 1)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (6, 'PIRIZ8', 1)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (7, 'DOLRIL7', 2)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (8, 'CARTION6', 3)");
    db.execSQL("INSERT INTO `offrir` (`idRapport`, `idMedicament`, `quantite`) VALUES (9, 'AMOXIG12', 2)");

    Log.d( tag: "log", msg: "base de test cree");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // TODO Auto-generated method stub

}
```

LA DOCUMENTATION UTILISATEUR

Ceci est la documentation d'utilisation de l'application Android « Medic'App », elle retracera le fonctionnement de celle-ci pour que vous aussi puissiez profiter de toutes ses fonctionnalités.

INTRODUCTION

L'activité commerciale d'un laboratoire pharmaceutique est principalement réalisée par les visiteurs médicaux. En effet, un médicament remboursé par la sécurité sociale n'est jamais vendu directement au consommateur mais prescrit au patient par son médecin. Toute communication publicitaire sur les médicaments remboursés est d'ailleurs interdite par la loi. Il est donc important, pour l'industrie pharmaceutique de promouvoir ses produits directement auprès des praticiens.

Pour cela, nous avons mis au point une application qui permet à un visiteur médical de consulter, ajouter, modifier et supprimer un praticien, un rapport ou encore un médicament.

CONNEXION

Une fois l'application « Medic'App » ouverte, la page de connexion s'ouvre. Elle vous propose de saisir un nom de compte et un mot de passe. Une fois la saisie effectuée, vous devez appuyer sur le bouton « connexion » pour accéder à l'accueil. Si vos informations sont correctes, vous pourrez accéder au reste de l'application.

ACCUEIL

Une fois connecté, la page d'accueil vous propose de choisir entre trois boutons :

- Le bouton « PRATICIENS » : il permet d'accéder à la page de gestion des praticiens,
- le bouton « RAPPORTS DE VISITE » : il permet d'accéder à la page de gestion des rapports de visite,
- le bouton « MÉDICAMENTS » : il permet d'accéder à la page de gestion des médicaments.

PRATICIENS

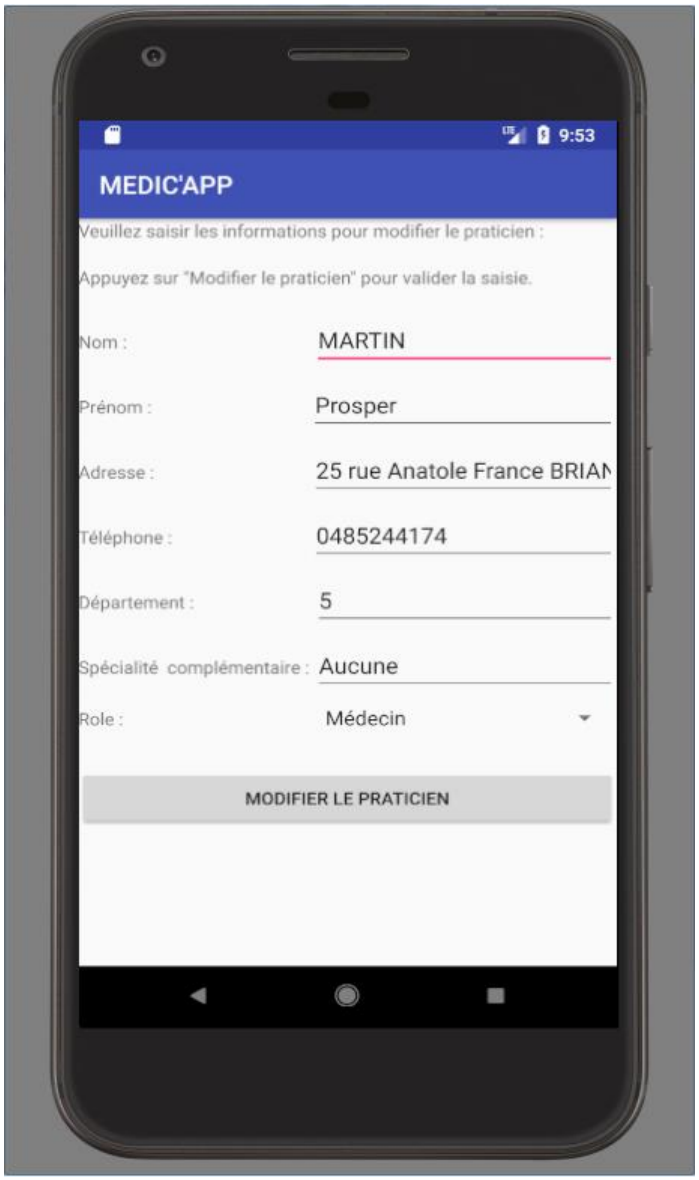
PAGE DE CONSULTATION ET DE GESTION DES PRATICIENS

Une fois sur la page de consultation des praticiens, vous avez la possibilité de consulter les informations d'un praticien en fonction de chaque rôle. De plus, il vous est proposé de choisir entre trois boutons :

- Le bouton « MODIFIER CE PRATICIEN » : il permet d'accéder à la page de modification du praticien sélectionné,
- le bouton « AJOUTER CE PRATICIEN » : il permet d'accéder à la page d'ajout du praticien sélectionné,
- Le bouton « SUPPRIMER CE PRATICIEN » : il permet de supprimer directement le praticien sélectionné.

PAGE DE MODIFICATION D'UN PRATICIEN

Une fois sur la page de modification d'un praticien, vous accéder à un formulaire pré-remplis des informations du praticien précédemment sélectionné. Dès lors, vous pouvez modifier les champs que vous désirez et valider ces changements en appuyant sur le bouton « MODIFIER LE PRATICIEN».



PAGE D'AJOUT D'UN PRATICIEN

Une fois sur la page d'ajout d'un praticien, vous accéder à un formulaire où les informations du nouveau praticien doivent être complétées. Enfin, validez l'ajout en appuyant sur le bouton « AJOUTER LE PRATICIEN».

PAGE DE SUPPRESSION D'UN PRATICIEN

Vous pouvez supprimer le praticien sélectionner depuis la page de consultation de praticiens en appuyant sur le bouton « SUPPRIMER CE PRATICIEN ».

RAPPORTS DE VISITE

PAGE DE CONSULTATION ET DE GESTION DES RAPPORTS DE VISITE

Une fois sur la page de consultation des rapports de visite, vous avez la possibilité de consulter les informations d'un rapport de visite en fonction du praticien et du code rapport sélectionné. De plus, il vous est proposé de choisir entre trois boutons :

- Le bouton « MODIFIER CE RAPPORT » : il permet d'accéder à la page de modification du rapport sélectionné si vous connecté en tant que le visiteur du rapport. Pour pouvoir modifier un rapport, vous devez être connecté en tant que visiteur rédacteur du rapport : Si vous êtes connecté en tant que « Aribi Alain », vous aurez la possibilité de modifier uniquement les rapports dont le nom et le prénom du visiteur est « Aribi Alain » (Vous pourrez alors modifier le rapport du screenshot de gauche mais pas celui de droite),
- le bouton « AJOUTER CE RAPPORT » : il permet d'accéder à la page d'ajout du rapport sélectionné,
- Le bouton « SUPPRIMER CE RAPPORT » : il permet de supprimer directement le rapport sélectionné.

PAGE DE MODIFICATION D'UN RAPPORT DE VISITE

Une fois sur la page de modification d'un rapport de visite, vous accéder à un formulaire pré-remplis des informations du rapport précédemment sélectionné. Dès lors, vous pouvez modifier les champs que vous désirez et valider ces changements en appuyant sur le bouton « SAUVEGARDER LE RAPPORT».

PAGE D'AJOUT D'UN RAPPORT DE VISITE

Une fois sur la page d'ajout d'un rapport de visite, vous accéder à un formulaire où les informations du nouveau rapport de visite doivent être complétées. Enfin, validez l'ajout en appuyant sur le bouton « AJOUTER LE RAPPORT».

PAGE DE SUPPRESSION D'UN RAPPORT DE VISITE

Vous pouvez supprimer le rapport de visite sélectionné depuis la page de consultation de praticiens en appuyant sur le bouton « SUPPRIMER CE RAPPORT ».

MEDICAMENTS

PAGE DE CONSULTATION ET DE GESTION DES MÉDICAMENTS

Une fois sur la page de consultation des médicaments, vous avez la possibilité de consulter les informations d'un médicament en fonction de chaque famille. De plus, il vous est proposé de choisir entre trois boutons :

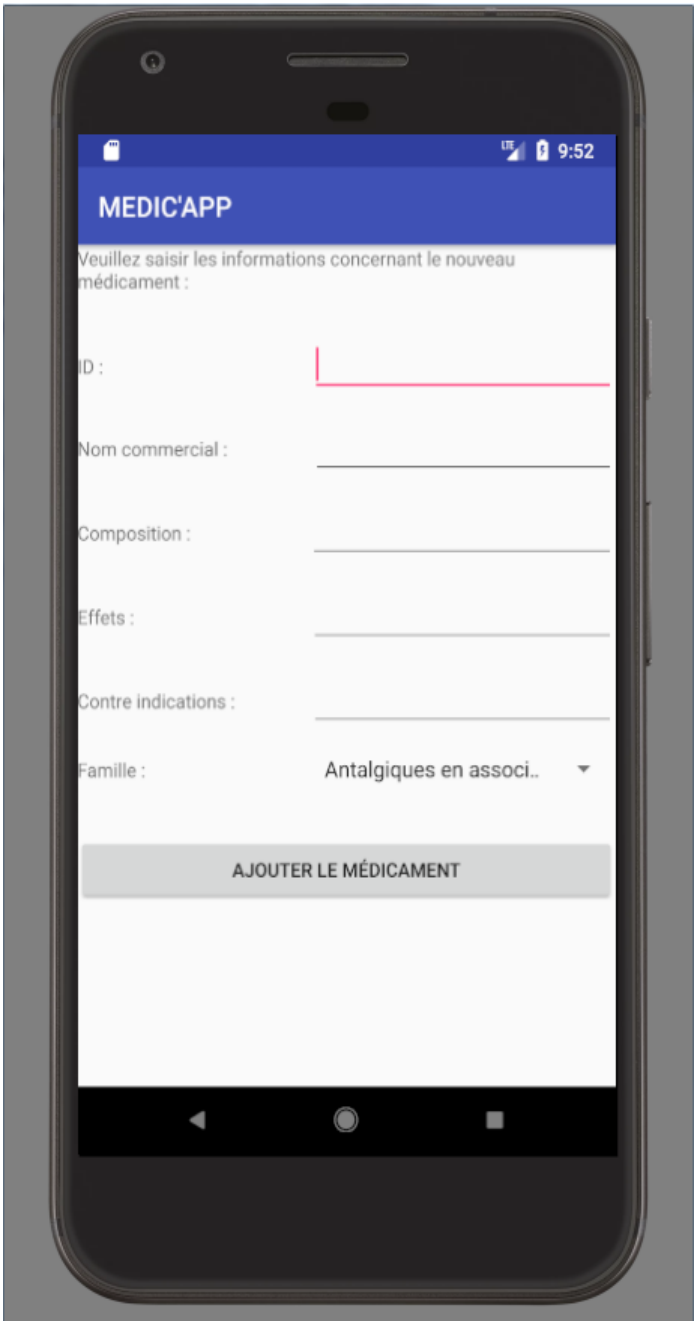
- Le bouton « MODIFIER CE MÉDICAMENT » : il permet d'accéder à la page de modification du médicament sélectionné,
- le bouton « AJOUTER CE MÉDICAMENT » : il permet d'accéder à la page d'ajout du médicament sélectionné,
- Le bouton « SUPPRIMER CE MÉDICAMENT » : il permet de supprimer directement le médicament sélectionné.

PAGE DE MODIFICATION D'UN MÉDICAMENT

Une fois sur la page de modification d'un médicament, vous accéder à un formulaire pré-remplis des informations du médicament précédemment sélectionné. Dès lors, vous pouvez modifier les champs que vous désirez et valider ces changements en appuyant sur le bouton « MODIFIER LE MÉDICAMENT ».

PAGE D'AJOUT D'UN MÉDICAMENT

Une fois sur la page d'ajout d'un médicament, vous accéder à un formulaire où les informations du nouveau médicament doivent être complétées. Enfin, validez l'ajout en appuyant sur le bouton « AJOUTER LE MEDICAMENT».



PAGE DE SUPPRESSION D'UN MÉDICAMENT

Vous pouvez supprimer le médicament sélectionné depuis la page de consultation des médicaments en appuyant sur le bouton « SUPPRIMER CE MEDICAMENT».