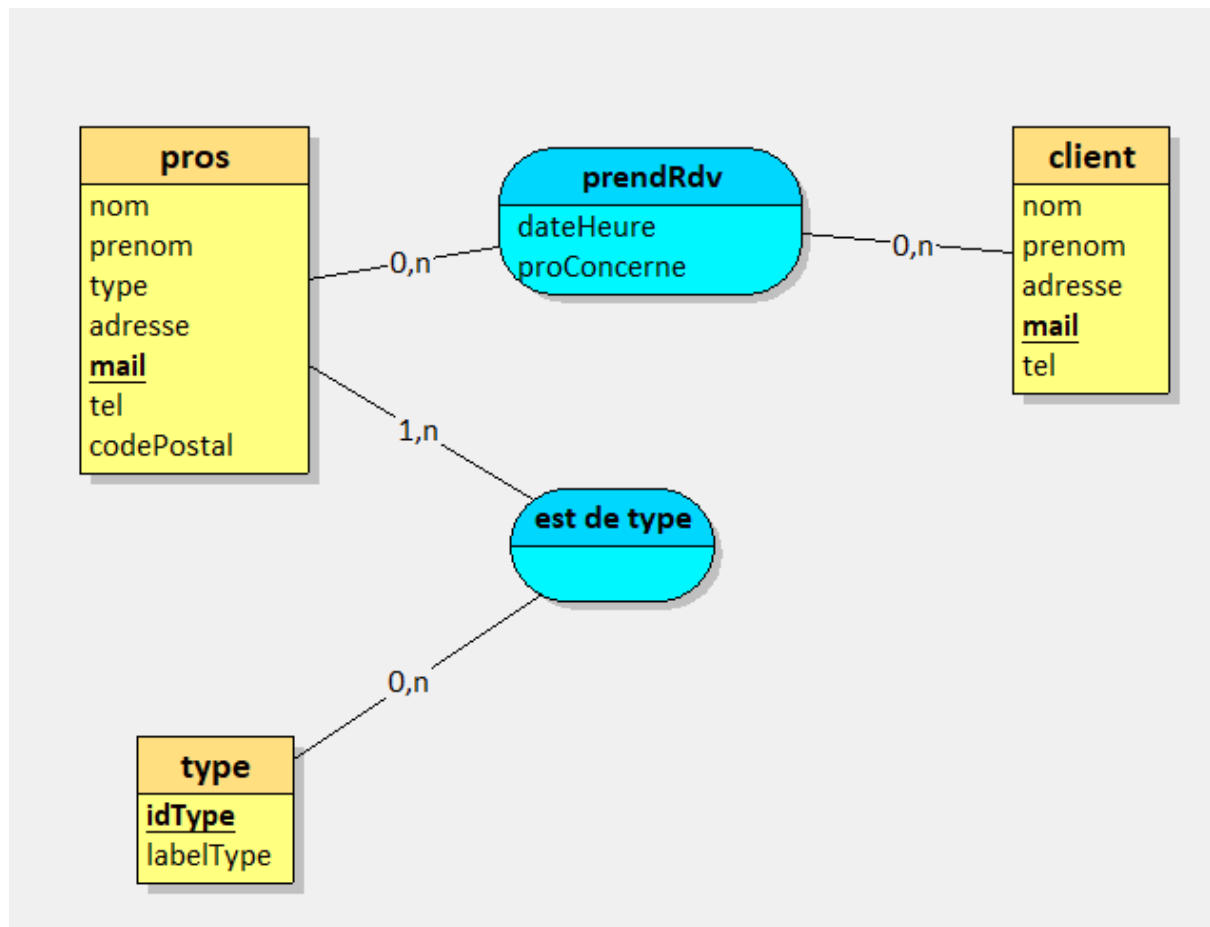


AUGEIX Adrien



Maquette de l'app :

Accueil

ajouter client

prendre rdv

afficher planning

rechercher client

ajout client

| nom

| prenom

▼ type

| adresse

| code postal

| email

| numero tel

ajouter client

prendre rdv

rechercher client

choisir date

choisir heure



pro concerné

ajouter client



code postal

liste

code :

```
package com.tokard.gestionmedecin

import android.annotation.SuppressLint
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity : AppCompatActivity() {
    @SuppressLint("MissingInflatedId") // dit à mon IDE de
    supprimer le warning "MissingInflatedId"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // récupère les boutons
        val ajoutClientButton =
            findViewById<Button>(R.id.ajoutClientButton)
        var prendreRdvButton = findViewById<Button>(R.id.rdvButton)
```

```

        var affPlanningButton =
findViewById<Button>(R.id.affPlanningButton)
        var rechercheButton =
findViewById<Button>(R.id.rechercheButton)

        supportActionBar?.hide() // cache la barre d'action

        ajoutClientButton.setOnClickListener {
            val intent = Intent(this, AjoutClient::class.java)
            startActivity(intent)
        }

        prendreRdvButton.setOnClickListener {
            val intent = Intent(this, PrendreRdv::class.java)
            startActivity(intent)
        }

        affPlanningButton.setOnClickListener {
            val intent = Intent(this, AffPlanning::class.java)
            startActivity(intent)
        }

        rechercheButton.setOnClickListener {
            val intent = Intent(this, RechercheClient::class.java)
            startActivity(intent)
        }
    }
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:text="AJOUTER CLIENT"
            android:layout_width="match_parent"
            android:layout_height="100dp"

```

```

        android:id="@+id/ajoutClientButton"
        style="@style/Widget.AppCompat.Button"
    />

    <Button
        android:text="PRENDRE RDV"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:id="@+id/rdvButton"
        style="@style/Widget.AppCompat.Button"
    />

    <Button
        android:text="AFF. PLANNING"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:id="@+id/affPlanningButton"
        style="@style/Widget.AppCompat.Button"
    />

    <Button
        android:text="RECHER. CLIENT"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:id="@+id/rechercheButton"
        style="@style/Widget.AppCompat.Button"
    />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="150dp"
        app:srcCompat="@drawable/gsblogo"
        android:id="@+id/imageView3"
        android:paddingTop="10dp"/>

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```

AffPlanning

```

package com.tokard.gestionmedecin

import android.annotation.SuppressLint
import androidx.appcompat.app.AppCompatActivity

```

```

import android.os.Bundle
import android.widget.AdapterView
import android.widget.CalendarView
import android.widget.DatePicker
import android.widget.ListView

class AffPlanning : AppCompatActivity() {
    @SuppressWarnings("Range")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_aff_planning)
        supportActionBar?.title = "Affichage du planning"; // met le
        titre de l'activité dans la barre d'action

        val calendarPick =
        findViewById<CalendarView>(R.id.calendarPick)
        val listeRDV = findViewById<ListView>(R.id.listeRDV)

        val date = java.util.Calendar.getInstance() // récupère la
        date actuelle
        val dayOfMonthStart =
        date.get(java.util.Calendar.DAY_OF_MONTH)
        val monthStart = date.get(java.util.Calendar.MONTH)
        val yearStart = date.get(java.util.Calendar.YEAR)

        getInfoDate("$dayOfMonthStart/${monthStart+1}/${yearStart}")

        calendarPick.setOnDateChangeListener { _, year, month,
        dayOfMonth -> // quand on change la date dans le calendrier
            getInfoDate("$dayOfMonth/${month+1}/${year}")
        }

    }

    @SuppressWarnings("Range")
    fun getInfoDate(date : String) {

        val maBd = BD(this)
        val listeRDV = findViewById<ListView>(R.id.listeRDV)

        var rdv = maBd.selectFromDay(date)

        var liste = ArrayList<String>() // liste des rdv qu'on
        pourra transmettre à l'adapter
        while (rdv.moveToNext()) {
            val heure = rdv.getString(rdv.getColumnIndex("heure"))
            val nom = rdv.getString(rdv.getColumnIndex("nom"))

```

```

        val prenom =
rdv.getString(rdv.getColumnIndex("prenom"))
        val codePostal =
rdv.getString(rdv.getColumnIndex("codePostal"))

        liste.add("$heure - $nom $prenom ($codePostal)")
    }

    var adapter = ArrayAdapter(this,
android.R.layout.simple_list_item_1, liste)
    listeRDV.adapter = adapter
}
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".AjoutClient"
    android:scrollbarStyle="insideInset">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <CalendarView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/calendarPick"
            />

        <ListView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/listeRDV"
            tools:layout_editor_absoluteX="0dp"

app:layout_constraintTop_toBottomOf="@+id/datePicker"/>

```

```
        </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

AjoutClient

```
package com.tokard.gestionmedecin

import android.annotation.SuppressLint
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.*

class AjoutClient : AppCompatActivity() {
    @SuppressLint("CutPasteId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_ajout_client)
        supportActionBar?.title = "Ajout d'un client" // met le
        titre de l'activité dans la barre d'action

        val maBD = BD(this)

        val typesProfessionnels = listOf("Généraliste",
        "Pharmacien", "Kiné", "Spécialiste")
        val spinnerType =
        findViewById<AutoCompleteTextView>(R.id.type)
        val adapter = ArrayAdapter(this,
        android.R.layout.simple_spinner_item, typesProfessionnels)

        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
        pdown_item)
        spinnerType.setAdapter(adapter)

        var btnAjouterClient =
        findViewById<Button>(R.id.btnPrendreRdv)
        btnAjouterClient.setOnClickListener {
            // récupère les zones de saisie de texte
```



```

        val nom =
findViewById<EditText>(R.id.editTextNom).text.toString().uppercase(
)
        // met la première lettre du prénom en majuscule
        val prenom =
findViewById<EditText>(R.id.editTextPrenom).text.toString().lowercase().replaceFirstChar {
            if (it.isLowerCase()) it.titlecase(
                java.util.Locale.getDefault()
            ) else it.toString()
        }

        val type =
findViewById<AutoCompleteTextView>(R.id.type).text.toString()
        val adresse =
findViewById<EditText>(R.id.editTextAdresse).text.toString()
        val codePostal =
findViewById<EditText>(R.id.editTextCodePostal).text.toString()
        val email =
findViewById<EditText>(R.id.editTextEmail).text.toString()
        val telephone =
findViewById<EditText>(R.id.editTextTelephone).text.toString()

        // Vérification des champs vides
        if (nom.isEmpty() || prenom.isEmpty() || type.isEmpty()
|| adresse.isEmpty() ||
            codePostal.isEmpty() || email.isEmpty() ||
telephone.isEmpty()
        ) {
            // Affiche un Toast indiquant à l'utilisateur de
remplir tous les champs
            Toast.makeText(this, "Veuillez remplir tous les
champs", Toast.LENGTH_SHORT).show()
        } else {
            // Transmet les données à la méthode enregistrerPro
de la base de données
            maBD.enregistrerPro(nom, prenom, type, adresse,
email, telephone, codePostal)
            // Affiche un Toast indiquant que l'ajout a été
effectué avec succès
            Toast.makeText(this, "Client ajouté avec succès",
Toast.LENGTH_SHORT).show()
            finish()
        }
    }
}
}
}

```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".AjoutClient"
    android:scrollbarStyle="insideInset">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp">

<com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Prénom"
            android:id="@+id/editTextPrenom"/>

        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp">

<com.google.android.material.textfield.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Nom"
            android:id="@+id/editTextNom"/>

        </com.google.android.material.textfield.TextInputLayout>

        <com.google.android.material.textfield.TextInputLayout
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBo  
x.ExposedDropdownMenu"
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="8dp">
```

```
        <AutoCompleteTextView  
            android:text=""  
            android:layout_width="match_parent"  
            android:layout_height="match_parent"  
            android:id="@+id/type"  
            android:inputType="none"  
            android:layout_weight="1"/>
```

```
    </com.google.android.material.textfield.TextInputLayout>
```

```
    <com.google.android.material.textfield.TextInputLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginBottom="8dp">
```

```
<com.google.android.material.textfield.TextInputEditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Adresse"  
    android:inputType="textNoSuggestions"  
    android:id="@+id/editTextAdresse"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
    <com.google.android.material.textfield.TextInputLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_marginBottom="8dp">
```

```
<com.google.android.material.textfield.TextInputEditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Code Postal"  
    android:inputType="number"  
    android:id="@+id/editTextCodePostal"  
    android:textColorHint="#616161"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
    <com.google.android.material.textfield.TextInputLayout
```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">

<com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:id="@+id/editTextEmail"
        android:textColorHint="#616161"/>

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">

<com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Numéro Tel."
        android:inputType="phone"
        android:id="@+id/editTextTelephone"/>

</com.google.android.material.textfield.TextInputLayout>

<Button
        android:text="Ajouter Client"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:id="@+id/btnPrendreRdv"/>

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

PrendreRdv

```

package com.tokard.gestionmedecin

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.*

```

```

import androidx.core.content.ContextCompat

class PrendreRdv : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_prendre_rdv)
        supportActionBar?.title = "Prise de rendez-vous"

        // quand dateButton est cliqué, affiche le calendarPick
        avec une animation
        val dateButton = findViewById<Button>(R.id.dateButton)
        val calendarPick =
        findViewById<CalendarView>(R.id.calendarPick)
        val timePickerRdv =
        findViewById<TimePicker>(R.id.timePickerRdv)
        val heureButton = findViewById<Button>(R.id.heureButton)

        timePickerRdv.setIs24HourView(true) // met le timePicker en
        24h

        calendarPick.visibility = CalendarView.GONE
        timePickerRdv.visibility = TimePicker.GONE

        dateButton.setOnClickListener { // quand on clique sur
        dateButton
            if (calendarPick.visibility == CalendarView.GONE) { //
            gère la visibilité des éléments pour ne pas surcharger l'interface
                calendarPick.visibility = CalendarView.VISIBLE
                timePickerRdv.visibility = TimePicker.GONE
            } else {
                calendarPick.visibility = CalendarView.GONE
            }
        }

        heureButton.setOnClickListener {
            if (timePickerRdv.visibility == TimePicker.GONE) {
                timePickerRdv.visibility = TimePicker.VISIBLE
                calendarPick.visibility = CalendarView.GONE
            } else {
                timePickerRdv.visibility = TimePicker.GONE
            }
        }

        calendarPick.setOnDateChangeListener { _, year, month,
        dayOfMonth -> // quand on change la date dans le calendrier
            var dateSelectionnee = "$dayOfMonth/${month+1}/${year}"
            dateButton.text = dateSelectionnee
        }
    }
}

```

```

        timePickerRdv.setOnTimeChangedListener { _, hourOfDay,
minute ->
            var heureSelectionnee = "$hourOfDay:$minute"
            heureButton.text = heureSelectionnee
        }

        // récupère tout les medecins de la base de données et les
affiche dans le spinner spinnerProConcerne
        val maBD = BD(this)
        val listePro = maBD.listerPro() // hashmap
        val spinnerProConcerne =
findViewById<AutoCompleteTextView>(R.id.proConcerne)

        // quand on change de professionnel, on met la couleur du
texte en noir
        spinnerProConcerne.setOnItemClickListener { _, _, _, _ ->

spinnerProConcerne.setTextColor(ContextCompat.getColor(this,
R.color.black))
        }

        // fais une liste avec les noms des professionnels
// liste pro est tel que "nom prenom" : clé (int)
        val listeNomPro = listePro.keys.toList()
        val adapter = ArrayAdapter(this,
android.R.layout.simple_spinner_item, listeNomPro)

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dro
pdown_item)
        spinnerProConcerne.setAdapter(adapter)

        var btnPrendreRdv = findViewById<Button>(R.id.btnPrendreRdv)

        btnPrendreRdv.setOnClickListener {

            // récupère l'id du professionnel sélectionné
            val nomPro = spinnerProConcerne.text.toString()
            val idPro = listePro[nomPro]

            // récupère la date et l'heure sélectionnée à partir
            val dateRdv = dateButton.text.toString()
            val heureRdv = heureButton.text.toString()

```

```

        // insère les données dans la bdd avec la méthode
enregistrerRdv
        if (idPro != null && dateRdv != "date" && heureRdv !=
"heure") {
            maBD.prendreRdv(dateRdv, heureRdv, idPro)
            Toast.makeText(this, "Rendez-vous pris avec
succès", Toast.LENGTH_SHORT).show()
            finish()
        } else {
            if (dateRdv == "date")
                Toast.makeText(this, "Merci de selectionner une
date", Toast.LENGTH_SHORT).show()
            else if (heureRdv == "heure")
                Toast.makeText(this, "Merci de selectionner une
heure", Toast.LENGTH_SHORT).show()
            else
                Toast.makeText(this, "Merci de selectionner un
client", Toast.LENGTH_SHORT).show()
        }
    }
}
}
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".PrendreRdv"
    android:scrollbarStyle="insideInset">

    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <com.google.android.material.textfield.TextInputLayout

```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBo  
x.ExposedDropDownMenu"
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_marginBottom="8dp">
```

```
    <AutoCompleteTextView
```

```
        android:text="sélectionner un client"  
        android:textColor="#B1B1B1"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:id="@+id/proConcerne"  
        android:inputType="none"  
        android:layout_weight="1"/>
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<Button
```

```
    android:text="date"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/dateButton"  
    android:backgroundTint="#985EE0"/>
```

```
<CalendarView
```

```
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/calendarPick"  
    />
```

```
<Button
```

```
    android:text="heure"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/heureButton"  
    android:backgroundTint="#985EE0"/>
```

```
<TimePicker
```

```
    android:id="@+id/timePickerRdv"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    />
```

```
<!-- Bouton "ajouterClient" -->
```



```

        <Button
            android:id="@+id/btnPrendreRdv"
            android:layout_width="match_parent"
            android:layout_height="80dp"
            android:text="prendre RDV"
        />

    </LinearLayout>
</ScrollView>

```

BD

```

package com.tokard.gestionmedecin

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class BD(context: Context) : SQLiteOpenHelper(context,
    DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        const val DATABASE_NAME = "GestionMedecin.db"
        private const val DATABASE_VERSION = 1

        // Table Professionnel
        private const val TABLE_PROFESSIONNEL = "professionnel"
        private const val COL_ID = "id"
        private const val COL_NOM = "nom"
        private const val COL_PRENOM = "prenom"
        private const val COL_TYPE = "type"
        private const val COL_ADRESSE = "adresse"
        private const val COL_MAIL = "mail"
        private const val COL_TEL = "tel"
        private const val COL_CODEPOSTAL = "codepostal"

        // Table RDV
        private const val TABLE_RDV = "rdv"
        private const val COL_DATE = "date"
        private const val COL_HEURE = "heure"
        private const val COL_ID_PRO = "idPro"
    }

    /**
     * Création des tables
     */
}

```

```

    */
    override fun onCreate(db: SQLiteDatabase) {
        val createProfessionnelTable = """
            CREATE TABLE $TABLE_PROFESSIONNEL (
                $COL_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COL_NOM TEXT,
                $COL_PRENOM TEXT,
                $COL_TYPE TEXT,
                $COL_ADRESSE TEXT,
                $COL_MAIL TEXT,
                $COL_TEL TEXT,
                $COL_CODEPOSTAL TEXT
            )
        """.trimIndent()

        val createRdvTable = """
            CREATE TABLE $TABLE_RDV (
                $COL_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COL_DATE TEXT,
                $COL_HEURE TEXT,
                $COL_ID_PRO INTEGER,
                FOREIGN KEY($COL_ID_PRO) REFERENCES
$TABLE_PROFESSIONNEL($COL_ID)
            )
        """.trimIndent()

        db.execSQL(createProfessionnelTable)
        db.execSQL(createRdvTable)
    }

    /**
     * ajout d'un professionnel
     * @param nom
     * @param prenom
     * @param type
     * @param adresse
     * @param mail
     * @param tel
     * @param codePostal
     */
    fun enregistrerPro(
        nom: String,
        prenom: String,
        type: String,
        adresse: String,
        mail: String,
        tel: String,
        codePostal: String
    )

```

```

): Long {
    val values = ContentValues().apply {
        put(COL_NOM, nom)
        put(COL_PRENOM, prenom)
        put(COL_TYPE, type)
        put(COL_ADRESSE, adresse)
        put(COL_MAIL, mail)
        put(COL_TEL, tel)
        put(COL_CODEPOSTAL, codePostal)
    }

    return writableDatabase.insert(TABLE_PROFESSIONNEL, null,
values)
}

/**
 * ajout d'un rendez-vous
 * @param date
 * @param heure
 * @param idPro
 */
fun prendreRdv(date: String, heure: String, idPro: Int): Long {
    val values = ContentValues().apply {
        put(COL_DATE, date)
        put(COL_HEURE, heure)
        put(COL_ID_PRO, idPro)
    }

    return writableDatabase.insert(TABLE_RDV, null, values)
}

/**
 * selection des rendez-vous d'un jour precis
 * @param selectedDate jj/mm/aaaa
 */
fun selectFromDay(selectedDate: String): Cursor {
    val query = """
        SELECT $TABLE_RDV.$COL_DATE as date,
$TABLE_RDV.$COL_HEURE as heure,
        $TABLE_PROFESSIONNEL.$COL_NOM as nom,
$TABLE_PROFESSIONNEL.$COL_PRENOM as prenom,
        $TABLE_PROFESSIONNEL.$COL_CODEPOSTAL as
codePostal
        FROM $TABLE_RDV
        INNER JOIN $TABLE_PROFESSIONNEL ON
$TABLE_RDV.$COL_ID_PRO = $TABLE_PROFESSIONNEL.$COL_ID
    """
    return writableDatabase.query(query, null, null, null, null)
}

```

```

        WHERE $TABLE_RDV.$COL_DATE = ?
        ORDER BY $TABLE_RDV.$COL_HEURE ASC;
    """.trimIndent()
    val selectionArgs = arrayOf(selectedDate)
    return readableDatabase.rawQuery(query, selectionArgs)
}

/**
 * liste des professionnels
 * @return HashMap<String, Int> "nom prenom" -> id
 */
@SuppressLint("Range")
fun listerPro(): HashMap<String, Int> {
    val mapPro = hashMapOf<String, Int>()
    val query = "SELECT * FROM $TABLE_PROFESSIONNEL"
    val cursor = readableDatabase.rawQuery(query, null)
    while (cursor.moveToNext()) {
        val nom =
cursor.getString(cursor.getColumnIndex(COL_NOM))
        val prenom =
cursor.getString(cursor.getColumnIndex(COL_PRENOM))
        val id = cursor.getInt(cursor.getColumnIndex(COL_ID))
        mapPro["$nom $prenom"] = id
    }
    cursor.close()
    return mapPro
}

/**
 * liste des professionnels d'un code postal
 * @param codePostal
 * @return HashMap<String, Int> "nom prenom" -> id
 */
@SuppressLint("Range")
fun listerProFromCodePostal(codePostal: String):
HashMap<String, Int> {
    val mapPro = hashMapOf<String, Int>()
    val query = "SELECT * FROM $TABLE_PROFESSIONNEL WHERE
$COL_CODEPOSTAL = ?"
    val selectionArgs = arrayOf(codePostal)
    val cursor = readableDatabase.rawQuery(query, selectionArgs)
    while (cursor.moveToNext()) {
        val nom =
cursor.getString(cursor.getColumnIndex(COL_NOM))
        val prenom =
cursor.getString(cursor.getColumnIndex(COL_PRENOM))
        val id = cursor.getInt(cursor.getColumnIndex(COL_ID))
        mapPro["$nom $prenom"] = id
    }
}

```

```

    }
    cursor.close()
    return mapPro
}

/**
 * liste des codes postaux distincts
 * @return ArrayList<String> liste des codes postaux
 */
@SuppressLint("Range")
fun listerCodesPostal(): ArrayList<String> {
    val listeCodePostal = ArrayList<String>()
    val query = "SELECT DISTINCT $COL_CODEPOSTAL FROM $TABLE_PROFESSIONNEL"
    val cursor = readableDatabase.rawQuery(query, null)
    while (cursor.moveToNext()) {
        val codePostal =
            cursor.getString(cursor.getColumnIndex(COL_CODEPOSTAL))
        listeCodePostal.add(codePostal)
    }
    cursor.close()
    return listeCodePostal
}

/**
 * informations d'un professionnel
 * @param idPro
 * @return HashMap<String, String> "nom" -> nom, "prenom" -> prenom, "type" -> type, "adresse" -> adresse, "mail" -> mail, "tel" -> tel, "codePostal" -> codePostal
 */
@SuppressLint("Range")
fun getInfoPro(idPro: Int): HashMap<String, String> {
    val mapPro = hashMapOf<String, String>()
    val query = "SELECT * FROM $TABLE_PROFESSIONNEL WHERE $COL_ID = ?"
    val selectionArgs = arrayOf(idPro.toString())
    val cursor = readableDatabase.rawQuery(query, selectionArgs)
    while (cursor.moveToNext()) {
        val nom =
            cursor.getString(cursor.getColumnIndex(COL_NOM))
        val prenom =
            cursor.getString(cursor.getColumnIndex(COL_PRENOM))
        val type =
            cursor.getString(cursor.getColumnIndex(COL_TYPE))
        val adresse =
            cursor.getString(cursor.getColumnIndex(COL_ADRESSE))
    }
}

```

```

        val mail =
cursor.getString(cursor.getColumnIndex(COL_MAIL))
        val tel =
cursor.getString(cursor.getColumnIndex(COL_TEL))
        val codePostal =
cursor.getString(cursor.getColumnIndex(COL_CODEPOSTAL))
        mapPro["nom"] = nom
        mapPro["prenom"] = prenom
        mapPro["type"] = type
        mapPro["adresse"] = adresse
        mapPro["mail"] = mail
        mapPro["tel"] = tel
        mapPro["codePostal"] = codePostal
    }
    cursor.close()
    return mapPro
}
}

```

recherche client

```

package com.tokard.gestionmedecin

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.AutoCompleteTextView
import android.widget.ListView
import androidx.core.content.ContextCompat

class RechercheClient : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_recherche_client)
        supportActionBar?.title = "Recherche d'un client"

        val nomListe =
findViewById<AutoCompleteTextView>(R.id.nomListe)
        val CodePostalListe =
findViewById<AutoCompleteTextView>(R.id.CodePostalListe)

        nomListe.setOnItemClickListener { _, _, _, _ -> // quand un
element est selectionné on met le texte en noir
            // sert pas à grand chose mais c'est plus joli
            nomListe.setTextColor(ContextCompat.getColor(this,
R.color.black))
        }
    }
}

```

```

        CodePostalListe.setOnItemClickListener { _, _, _, _ ->
            // pareil
        }

        CodePostalListe.setTextColor(ContextCompat.getColor(this,
            R.color.black))
    }

    // récupère la liste des codes postaux disponibles avec
    listerCodesPostal()
    val maBD = BD(this)
    val listeCodePostal = maBD.listerCodesPostal()
    val adapterCodePostal = ArrayAdapter(this,
        android.R.layout.simple_list_item_1, listeCodePostal)
    CodePostalListe.setAdapter(adapterCodePostal)
    val affListeClients =
        findViewById<ListView>(R.id.listeClients)

    // quand codePostalListe est changé on utilise
    listerProFromCodePostal() pour récupérer les professionnels
    // on affiche les pros "nom prenom" dans nomListe
    CodePostalListe.setOnItemClickListener { _, _, _, _ ->

        CodePostalListe.setTextColor(ContextCompat.getColor(this,
            R.color.black))

        nomListe.setText("Client")
        // on vide affListeClients
        val listeVide = ArrayList<String>()
        nomListe.setTextColor(ContextCompat.getColor(this,
            R.color.grey))

        val hashPro =
            maBD.listerProFromCodePostal(CodePostalListe.text.toString()) //
            le hashmap contient les noms et prénoms et les id "nom prenom" ->
            id

        // on convertit le hashmap en arraylist pour récupérer
        uniquement les noms et prénoms
        val listePro = ArrayList<String>()
        for (i in hashPro) {
            listePro.add(i.key)
        }

        val adapterPro = ArrayAdapter(this,
            android.R.layout.simple_list_item_1, listePro)
        nomListe.setAdapter(adapterPro)
    }

    nomListe.setOnItemClickListener() { _, _, _, _ ->

```

```

        nomListe.setTextColor(ContextCompat.getColor(this,
R.color.black))
        // récupère les infos du pro dans un hashmap avec
getInfoPro(id)
        val idPro =
maBD.listerProFromCodePostal(CodePostalListe.text.toString())[nomL
iste.text.toString()]
        val hashPro = maBD.getInfoPro(idPro!!)
        // on ajoute chaque information sur une nouvelle ligne
dans affListeClients
        val listeInfoPro = ArrayList<String>()
        for (i in hashPro) {
            listeInfoPro.add(i.key + " : " + i.value)
        }
        val adapterInfoPro = ArrayAdapter(this,
android.R.layout.simple_list_item_1, listeInfoPro)
        affListeClients.adapter = adapterInfoPro

    }

}
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".AjoutClient"
    android:scrollbarStyle="insideInset">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.google.android.material.textfield.TextInputLayout

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBo
x.ExposedDropdownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">

```



```

        <AutoCompleteTextView
            android:text="Code postal"
            android:textColor="#B1B1B1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/CodePostalListe"
            android:inputType="none"
            android:layout_weight="1"/>
    </com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBo
x.ExposedDropDownMenu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp">

        <AutoCompleteTextView
            android:text="Client"
            android:textColor="#B1B1B1"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/nomListe"
            android:inputType="none"
            android:layout_weight="1"/>
    </com.google.android.material.textfield.TextInputLayout>

    <ListView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/listeClients"
    />

</LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>


```

lien de l'apk :

<https://cdn.discordapp.com/attachments/828942054783975454/1186579138312679484/gestionMedecin.apk?ex=6593c2f0&is=65814df0&hm=2bc91fba74925dcec1a5f9ea71752b88220306f0621488b9536b5e39f41922a0&>

N°	Valeur BD avant test	action	attendu	valeur bd après test	résultat
1	aucune	cliquer sur un des boutons de l'accueil	ouvre le menu correspondant	aucune	ouvre la page
2	aucune	cliquer sur le bouton ajouter client sans remplir d'infos	toast d'erreur	aucune	aff. mess. d'erreur
3	aucune	saisie d'informations valides pour un client	ajout d'un client + toast de validation	image 1	ajout dans la bdd
4	image 1	ajout d'un rdv sans saisie de date ou heure	toast d'erreur	image 1	aff. mess. d'erreur
5	image 1	ajout d'un rdv	toast qui valide, ajout d'un rdv	image 2	ajout d'un rdv
6	image 2	clic dans le calendrier d'affichage du planning	affichage des rdv pour la journée	image 2	affichage des rdv pour la journée
7	image 2	sélection d'un client et d'un code postal dans la liste des CP/Clients de la liste déroulante	affichage des infos du client	image 2	affichage des infos du client

 id	nom	pren...	type	adresse	mail	tel	codepostal
1	Baguette	Petite	Généraliste	rue du oui	puidf@dkbhdfsfr	123156	31200

 id	date	heure	idPro
1	14/12/2023	11:10	2

Tests BD

```
package com.tokard.gestionmedecin

import android.content.Context
import androidx.test.core.app.ApplicationProvider
import org.junit.jupiter.api.BeforeEach
import org.junit.jupiter.api.Test
import org.junit.jupiter.api.Assertions.*

class BDTest {

    private lateinit var context: Context
    private lateinit var database: BD

    @BeforeEach
    fun setup() {
        // Utilisez le contexte de l'application pour éviter les
erreurs de contexte
        context = ApplicationProvider.getApplicationContext()
        // Utilisez le contexte de test pour créer une base de
données de test distincte
        database = BD(context)
        // Supprimez la base de données de test existante à chaque
exécution du test
        context.deleteDatabase(BD.DATABASE_NAME)
    }

    @Test
    fun enregistrerPro() {
        val id = database.enregistrerPro("Nom", "Prenom", "Type",
"Adresse", "Mail", "Tel", "CodePostal")
        assertTrue(id > 0, "L'enregistrement du professionnel a
échoué")
    }

    @Test
    fun prendreRdv() {
        // Enregistrez un professionnel pour obtenir son ID
        val idPro = database.enregistrerPro("Nom", "Prenom",
"Type", "Adresse", "Mail", "Tel", "CodePostal")

        // Planifiez un rendez-vous avec l'ID du professionnel
        val idRdv = database.prendreRdv("2023-01-01", "10:00",
idPro.toInt())
        assertTrue(idRdv > 0, "La planification du rendez-vous a
échoué")
    }
}
```

```

@Test
fun listerPro() {
    // Enregistrez quelques professionnels
    database.enregistrerPro("Nom1", "Prenom1", "Type1",
"Adresse1", "Mail1", "Tel1", "CodePostal1")
    database.enregistrerPro("Nom2", "Prenom2", "Type2",
"Adresse2", "Mail2", "Tel2", "CodePostal2")

    // Liste des professionnels
    val mapPro = database.listerPro()
    assertEquals(2, mapPro.size, "Le nombre de professionnels
ne correspond pas")
}

@Test
fun listerProFromCodePostal() {
    // Enregistrez quelques professionnels avec un code postal
spécifique
    database.enregistrerPro("Nom1", "Prenom1", "Type1",
"Adresse1", "Mail1", "Tel1", "CodePostalTest")
    database.enregistrerPro("Nom2", "Prenom2", "Type2",
"Adresse2", "Mail2", "Tel2", "CodePostalTest")

    // Liste des professionnels avec le code postal spécifique
    val mapPro =
database.listerProFromCodePostal("CodePostalTest")
    assertEquals(2, mapPro.size, "Le nombre de professionnels
avec le code postal spécifique ne correspond pas")
}

@Test
fun listerCodesPostal() {
    // Enregistrez quelques professionnels avec des codes
postaux différents
    database.enregistrerPro("Nom1", "Prenom1", "Type1",
"Adresse1", "Mail1", "Tel1", "CodePostal1")
    database.enregistrerPro("Nom2", "Prenom2", "Type2",
"Adresse2", "Mail2", "Tel2", "CodePostal2")

    // Liste des codes postaux distincts
    val listeCodePostal = database.listerCodesPostal()
    assertEquals(2, listeCodePostal.size, "Le nombre de codes
postaux distincts ne correspond pas")
}

@Test
fun getInfoPro() {

```

```
// Enregistrez un professionnel pour obtenir son ID
val idPro = database.enregistrerPro("Nom", "Prenom",
    "Type", "Adresse", "Mail", "Tel", "CodePostal")

// Obtenez les informations du professionnel
val mapPro = database.getInfoPro(idPro.toInt())
assertNotNull(mapPro["nom"], "Les informations du
professionnel ne sont pas disponibles")
}
}
```

AJOUTER CLIENT

PRENDRE RDV

AFF. PLANNING

RECHER. CLIENT



Ajout d'un client

AJOUTER CLIENT

9:30



Prise de rendez-vous

sélectionner un client



DATE



December 2023



S

M

T

W

T

F

S

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

HEURE

PRENDRE RDV

9:31



Prise de rendez-vous

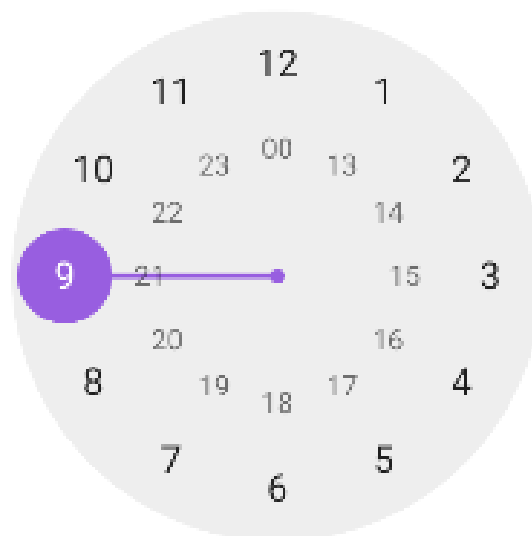
sélectionner un client



DATE

HEURE

09:30



PRENDRE RDV

Affichage du planning



December 2023



S

M

T

W

T

F

S

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

11:10 - tout cawaii Le petit metro (445465)

11:56 - Baguette Petite (31200)

14:20 - tout cawaii Le petit metro (445465)

Recherche d'un client

31600



Flourence Stéfan



mail : emaildefou@gmail.com

adresse : l'adresse de fou

tel : 0785426985

codePostal : 31600

type : Pharmacien

nom : Flourence

prenom : Stéfan

