

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

08/06/2024

# SMS Tracker App

**Capturer et stocker les Messages de Numéros  
Spécifiques**

Cours d'android Natif dispensé par Monsieur DEKU Kodjo Parfait

Promotion 2023-2024

ESGIS -TOGO

Several thin, curved, wavy lines in shades of blue and grey on the left side of the page.

**BAKO Bouchara**

ETUDIANTE EN MASTER 1 ARCHITECTURE DES LOGICIELS

## Table des matières

I-Introduction .....	2
II-Présentation de l'application .....	2
A - Présentation des outils utilisés .....	2
1- IDE Android studio .....	2
2- Langage Kotlin .....	2
B- Aspect fonctionnel.....	3
1- Ajouter un contact.....	4
2- Voir la liste des contacts.....	5
3- Voir les messages .....	6
C- Aspect technique.....	7
1- Gestion de contacts et messages .....	7
2- Gestion de la capture et stockage des messages .....	7
III - Quelques intérêts d'une telle application .....	10
IV- Conclusion.....	11
V- Liens consultés et utiles .....	11

## I-Introduction

Ce document est rédigé dans le but de présenter notre projet basé sur une application de capture de messages que nous avons dénommé « **SMS TRACKER APP** » ; projet qui permet de capturer, stocker et lire les messages des contacts enregistrés dans l'application et qui constitue notre examen de validation de l'unité d'enseignement d'Android Natif.

## II-Présentation de l'application

Pour cette partie nous parlerons essentiellement des outils utilisés et de la présentation de notre application.

### A - Présentation des outils utilisés

Pour la réalisation de notre projet nous avons eu besoin de l'environnement de Développement Intégré Android **Studio**, du langage de programmation **kotlin** et de quelques recherches faites dont les liens seront mentionnés un peu plus loin.

#### 1- IDE Android studio



Android Studio est l'environnement de développement intégré (IDE) officiel pour le développement d'applications Android. Basé sur IntelliJ IDEA de JetBrains, il offre une multitude d'outils et de fonctionnalités spécifiquement conçus pour faciliter la création, le test et le déploiement d'applications Android. Quelques liens utiles pour en apprendre davantage :

- [Site Officiel d'Android Studio](#)
- [Guide de l'Utilisateur Android Studio](#)
- [Documentation de Google sur Android Studio](#)

#### 2- Langage Kotlin

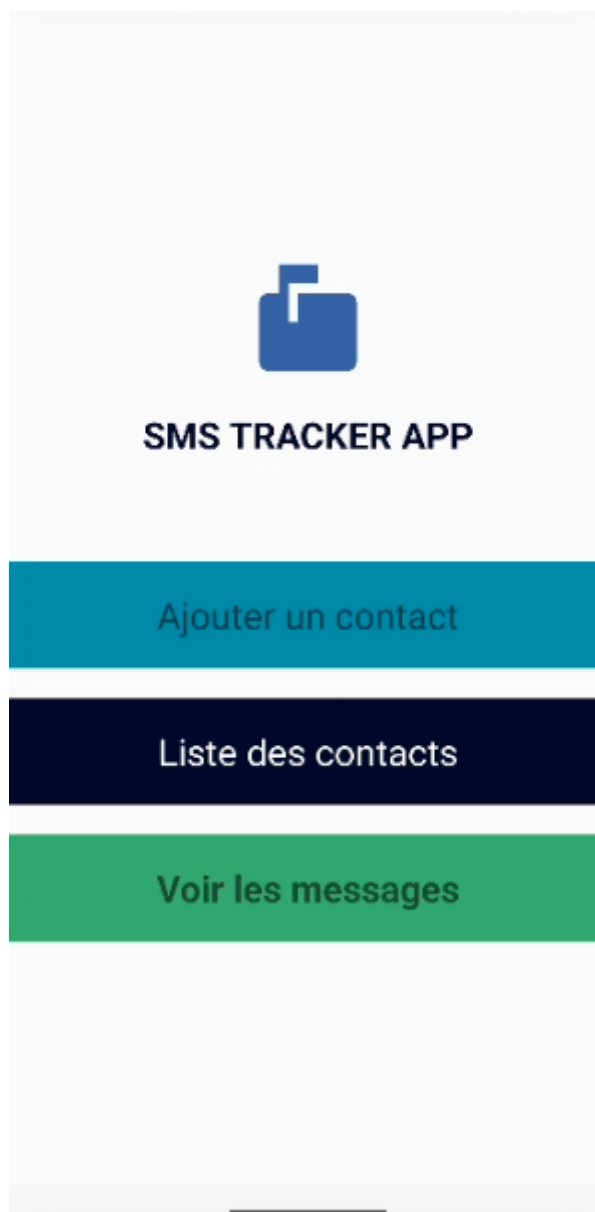


Kotlin est un langage de programmation moderne, concis, et sûr, développé par JetBrains. Conçu pour interopérer pleinement avec Java, il est particulièrement populaire dans le développement Android, mais est également utilisé pour d'autres types de développement, y compris les applications serveur, les applications web, et les applications de bureau. Quelques liens utiles pour en apprendre davantage :

- [Documentation Officielle de Kotlin](#)
- [Kotlin for Android Developers](#)
- [Cours en ligne sur Kotlin](#)

## B- Aspect fonctionnel

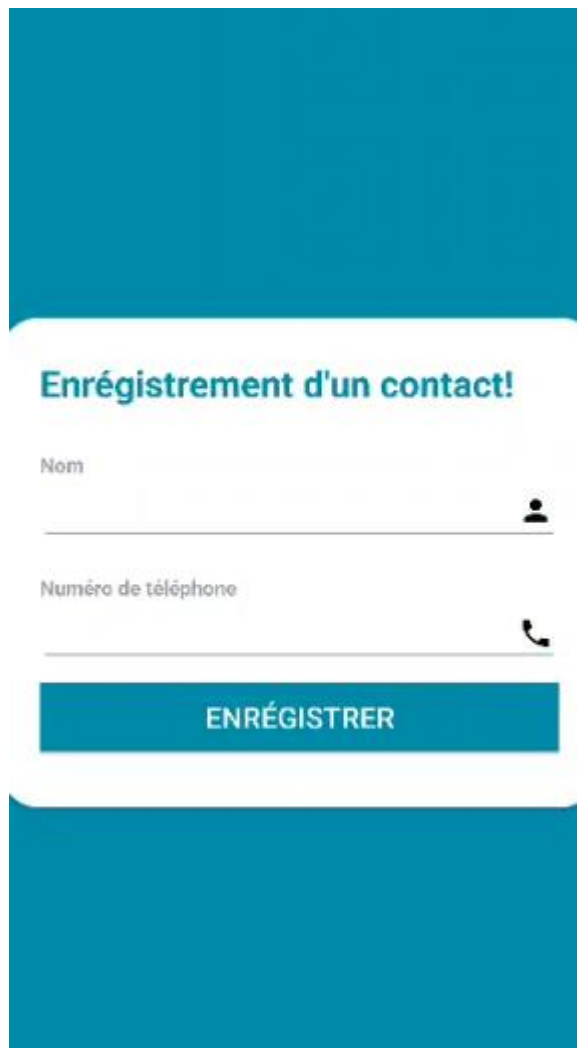
Notre application « **SMS TRACKER APP** » a pour but de capturer des messages entrants des différents contacts enregistrés dans notre application. Ainsi lorsque nous lançons notre application nous avons trois choix possibles qui sont soit : « **Ajouter un contact** », « **Voir la liste des contacts** » et en dernière option « **Voir les messages** »



***Figure 1 : Page d'accueil***

**1- Ajouter un contact**

Comme son nom l'indique, ce menu permet d'enregistrer des contacts dans notre application. Au clic du menu « Ajouter un contact » nous avons l'interface qui se présente à nous comme suit :



***Figure 2 : Page d'ajout de contacts***

On saisit donc le nom et le numéro de téléphone puis on enregistre. Ce numéro enregistré dans notre application, dorénavant aura ses messages reçus, enregistrés non seulement dans l'application par défaut de sms de notre téléphone mais également dans notre application « SMS TRACKER APP » .

## 2- Voir la liste des contacts

Ce menu nous permet de consulter la liste des contacts enregistrés dans notre application. Au clic du menu « **Voir la liste des contacts** » nous avons l'interface qui se présente à nous comme suit :



***Figure 3 :** Page de visualisation de contacts*

### 3- Voir les messages

Ce menu nous permet de consulter l'ensemble des messages envoyés par les contacts préalablement enregistrés dans notre application. Au clic du menu « **Voir les messages** » nous avons l'interface qui se présente à nous comme suit :



***Figure 4 : Page de visualisation des messages***

### C- Aspect technique

#### 1- Gestion de contacts et messages

Pour ce faire nous avons eu à utiliser une base **sqlite** contenant la table **contacts** (regroupant les attributs tels que l'id, le nom, le numéro et la date de création du contact) pour l'enregistrement des contacts et la table **messages** (regroupant les attributs tels que l'id, le contact, le corps du message) pour l'enregistrement des messages.

#### 2- Gestion de la capture et stockage des messages

Cette partie constitue la partie importante de notre application. Pour sa mise en place des configurations sont essentielles.



- Il faut au préalable que des permissions de réception et de lecture de sms soient octroyées. Pour cela dans notre fichier '**AndroidManifest.xml**' il faudra mettre ceci

```
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

- Ensuite créer une classe implémentant le BroadcastReceiver pour gérer les messages SMS entrants. Ce récepteur sera déclenché chaque fois qu'un nouveau SMS est reçu. Pour notre cas nous l'avons nommé **SMSReceiver**. Le contenu de cette classe est le suivi :

```
class SMSReceiver : BroadcastReceiver() {

    lateinit var ma: MessageActivity

    override fun onReceive(context: Context, intent: Intent) {
        // Récupère les extras de l'intent, contenant les données du message SMS.
        val bundle = intent.extras
        // Crée une instance de DBHelper pour gérer la base de données.
        val dbHelper = DBHelper(context)
        // Ouvre la base de données en mode lecture/écriture.
        val db = dbHelper.writableDatabase

        // Vérifie que le bundle n'est pas null.
        if (bundle != null) {
            // Récupère le tableau de PDU contenant les données brutes du SMS.
            val pdus = bundle["pdus"] as Array<*>
            // Itère sur chaque PDU pour traiter chaque message SMS.
            for (pdu in pdus) {
                // Crée un objet SmsMessage à partir du PDU.
                val message = SmsMessage.createFromPdu(pdu as ByteArray)
                // Récupère le numéro de téléphone de l'expéditeur.
                val phoneNumber = message.originatingAddress
                println("*****phoneNumber " + phoneNumber)
                val messageBody = message.messageBody // Récupère le corps du message SMS.
```

```

// Requête pour vérifier si le numéro de téléphone existe dans les contacts.
val cursor = db.query( table: "contacts", columns: null, selection: "contact=?", arrayOf(phoneNumber),
groupBy: null, having: null, orderBy: null)
println("*****HELLO * ${cursor.count}")
// Si le curseur contient au moins un résultat, le contact existe.
if (cursor.moveToFirst()) {
    println("*****HELLO01111")
    // Récupère le nom du contact à partir de la base de données locale.
    val contactName = cursor.getString(cursor.getColumnIndexOrThrow( columnName: "name"))

    // Crée un objet ContentValues pour stocker les données du message.
    val values = ContentValues().apply { this: ContentValues
        // Ajoute le numéro de téléphone de l'expéditeur.
        put("contact_phone", phoneNumber)
        // Ajoute le nom du contact.
        put("contact_name", contactName)
        // Ajoute le corps du message.
        put("message", messageBody)
    }
    // Insère les données du message dans la table messages.
    db.insert( table: "messages", nullColumnHack: null, values)
    // Affiche une notification Toast pour informer l'utilisateur de la réception du message.
    Toast.makeText(context, text: "Message reçu de $contactName", Toast.LENGTH_SHORT).show()
}
cursor.close() // Ferme le curseur pour libérer les ressources.
}

```

- Enregistrer le BroadcastReceiver (**SMSReceiver**) dans le fichier AndroidManifest.xml en faisant comme suit :

```

<receiver
    android:name=".SMSReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>

```

- Il faut aussi demander les autorisations SMS à l'exécution et pour cela nous avons écrit ces lignes dans notre MainActivity :

```
// Surcharge de la méthode onRequestPermissionsResult pour gérer les résultats des demandes d'autorisation
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    // Appel de la méthode de la superclasse pour maintenir le comportement par défaut
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)

    // Vérifie si la demande d'autorisation correspond au code de requête spécifié (1 dans ce cas)
    if (requestCode == 1) {
        // Vérifie si les résultats des autorisations ne sont pas vides et si la première autorisation est accordée
        if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            // Affiche un message indiquant que les permissions ont été accordées
            Toast.makeText(context: this, text: "Permissions accordées", Toast.LENGTH_SHORT).show()
        } else {
            // Affiche un message indiquant que les permissions ont été refusées
            Toast.makeText(context: this, text: "Permissions refusées", Toast.LENGTH_SHORT).show()
        }
    }
}
```

### III - Quelques intérêts d'une telle application

Créer une application mobile qui capture et enregistre les messages entrants peut présenter plusieurs intérêts et avantages dans divers domaines. Voici quelques-uns des principaux intérêts :

- Gestion et Organisation Personnelle

Archivage de Messages Importants : Permet d'enregistrer automatiquement les messages importants pour une consultation ultérieure sans risque de les perdre.

Filtrage de Contenu : Facilite le tri et la gestion des messages entrants, en les classant par catégories ou par expéditeurs spécifiques.

- Sécurité et Surveillance

Suivi Parental : Les parents peuvent surveiller les communications de leurs enfants pour s'assurer qu'ils ne sont pas exposés à des contenus inappropriés ou à des dangers en ligne.

Sécurité Personnelle : Enregistre les messages pour fournir des preuves en cas de harcèlement, de menaces, ou d'autres activités suspectes.

- Utilisation Professionnelle

Service Client : Les entreprises peuvent utiliser une telle application pour enregistrer les communications avec les clients, assurant ainsi un suivi précis des demandes et des réponses.

Conformité Légale : Dans certains secteurs, il est nécessaire de conserver des enregistrements de toutes les communications pour des raisons de conformité réglementaire.

- Automatisation et Productivité

Réponses Automatiques : Peut-être configurée pour envoyer des réponses automatiques basées sur le contenu des messages reçus.

Rappels et Notifications : Enregistre et rappelle les messages importants ou les événements liés aux messages reçus.

- Personnalisation et Accessibilité

Personnalisation : Permet aux utilisateurs de configurer des alertes spécifiques pour certains contacts ou types de messages.

Accessibilité : Facilite l'accès aux messages pour les utilisateurs ayant des besoins spéciaux en fournissant des options d'écoute ou de lecture automatique des messages.

## IV- Conclusion

Voilà en somme présenté notre projet de capture de sms des différents contacts enregistrés dans notre application « SMS TRACKER APP ».

## V- Liens consultés et utiles

- <https://www.c-sharpcorner.com/article/crud-operations-in-android-sqlite-kotlin/>
- <https://androidexample.com/>
- <https://www.kotlinsos.com/android/>
- <https://www.apriorit.com/>