Documentation du projet Android "Dépann'tout"

Contexte

"Dépann'tout" est une application mobile conçue pour un auto-entrepreneur qui effectue des dépannages à domicile. L'objectif est de simplifier la gestion des clients et des interventions pour le professionnel en leur offrant une interface efficace et simple .

Fonctionnalité présentée pour l'E6 :

Gestion des clients

Dans le cadre de l'application Dépann'tout, la fonctionnalité principale que je souhaite présenter est la gestion des clients.

Fonctionnalités principales de l'application

L'application "Dépann'tout" propose plusieurs fonctionnalités :
Gestion des clients :

- Créer un client : Ajouter un nouveau client avec les informations nécessaires (identifiant, nom, prénom, téléphone, email, adresse postale).

- Lister les clients : Visualiser tous les clients enregistrés.

- Modifier un client : Mettre à jour les informations d'un client existant.

- Supprimer un client :Supprimer un client (mais pas son intervention, si une intervention est liée à son id, l'intervention n'est pas supprimée) .

Gestion des interventions * :

- Créer une intervention : Ajouter une nouvelle intervention avec les détails (identifiant, date, observation, identifiant du client).

- Lister les interventions : Visualiser toutes les interventions créées.

*Cette partie de gestion des interventions ne sera pas utilisé pour l'E6.

Gestion de la base de données avec SQLite

Pour le stockage des données, l'application utilise SQLite.
Dans un package BDD :

La classe **CreateBdDepannTout :**
Cette classe hérite de SQLiteOpenHelper.
Elle est chargée de créer la base de données BdDepannTout avec les 2 tables nécessaires.
Comme elle hérite de SQLiteHelper, elle comprend un constructeur et les méthodes onCreate et on onUpgrade.

```
public class CreateBdDepannTout extends SQLiteOpenHelper {

    7 usages
    public static final String TABLE_CLIENT = "client";
    5 usages
    public static final String TABLE_INTER = "intervention";
    1 usage
    private static final String CREATE_TABLE_CLIENT =
            "CREATE TABLE " + TABLE_CLIENT + "(" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "nom TEXT NOT NULL, " +
                    "prenom TEXT NOT NULL, " +
                    "adrMail TEXT NOT NULL, " +
                    "numTel TEXT NOT NULL, " +
                    "adrPostale TEXT NOT NULL);";

    1 usage
    private static final String CREATE_TABLE_INTER =
            "CREATE TABLE " + TABLE_INTER + "(" +
                    "id INTEGER PRIMARY KEY AUTOINCREMENT, " +
                    "idCli INTEGER NOT NULL, " +
                    "dateTime TEXT NOT NULL, " +
                    "observation TEXT NOT NULL);";
```

```
// Constructeur (généré automatiquement)
1 usage
public CreateBdDepannTout(@Nullable Context context, @Nullable String name,
                          @Nullable SQLiteDatabase.CursorFactory factory, int version) {
    super(context, name, factory, version);
}
```

```java
/**
 * Création de la base de données si elle n'existe pas
 *
 * @param db base
 */
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE_CLIENT);
    Log.d( tag: "bdd",  msg: "table client créée");

    db.execSQL(CREATE_TABLE_INTER);
    Log.d( tag: "bdd",  msg: "table intervention créée");
}

/**
 * Création d'une nouvelle base en cas de changement de version
 *
 * @param db
 * @param oldVersion
 * @param newVersion
 */
10 usages
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_CLIENT + ";");
    Log.d( tag: "bdd",  msg: "Table " + TABLE_CLIENT + " supprimée");
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_INTER + ";");
    Log.d( tag: "bdd",  msg: "Table " + TABLE_INTER + " supprimée");
    onCreate(db);
}
```

La classe **DAO** (pour : Data Access Object) :
Cette classe permet la connexion à la base de donnée.
Elle contient les méthodes :
open() : pour ouvrir la base de donnée .
close() : pour fermer la base de donnée .

```java
public class DAO {
    1 usage
    private static final int VERSION_BDD = 2;
    1 usage
    private static final String NOM_BDD = "bdDepannTout";
    2 usages
    private CreateBdDepannTout createBd = null;
    5 usages
    private SQLiteDatabase db = null;

    2 usages
    public DAO(Context context){
        createBd = new CreateBdDepannTout(context, NOM_BDD,  factory: null, VERSION_BDD);
        Log.d( tag: "bdd",  msg: "Appel au constructeur de DAO ok, bdd créée");
    }
    2 usages
    public SQLiteDatabase open(){
        if (db == null){
            db = createBd.getWritableDatabase();
            Log.d( tag: "bdd",  msg: "Base de données ouverte");
        } else {
            Log.d( tag: "bdd",  msg: "base de données accessible");
        }
        return db;
    }
    2 usages
    public void close() {
        if(db != null){
            db.close();
            Log.d( tag: "bdd",  msg: "Base de données fermée");
        }
    }
```

La classe **ClientDAO** :
Gère les opérations CRUD (Create, Read, Update, Delete) pour les clients dans l'application. Elle interagit directement avec la base de données SQLite pour ajouter, lire, modifier et supprimer les clients .

Elle est composée :

- D'objets :
- dao : Un objet de la classe DAO (qui gère la connexion à la base de données).

- db : Un objet de type SQLiteDatabase qui représente la base de données ouverte via DAO.

- Méthodes :
- ClientDAO(Context context) : Le constructeur crée une instance de DAO et ouvre la connexion à la base de données via dao.open().

- close() : Ferme la base de données en appelant dao.close().

- readLesClients() : Exécute une requête SQL pour récupérer tous les clients de la base de données et retourne un Cursor avec les résultats.

- create(Client c) : Insère un nouveau client dans la base de données en utilisant les informations de l'objet Client passé en paramètre.

- deleteClient(int idClient) : Supprime un client de la base de données à partir de son id.

- updateClient(int idClient, ...) : Met à jour les informations d'un client dans la base de données avec de nouvelles valeurs.

Insertion : Les données d'un client sont insérées dans la base à l'aide de ContentValues, puis via db.insert().

Lecture : Les données sont lues via une requête SQL avec db.rawQuery(), qui retourne un Cursor contenant les résultats.

Suppression : Un client est supprimé à l'aide de db.delete(), en passant l'ID du client.

Mise à jour (modification) : Les informations d'un client sont mises à jour avec db.update(), en utilisant l'ID comme clé.

```
public class ClientDAO  {
    3 usages
    private DAO dao = null;
    6 usages
    private SQLiteDatabase db = null;

    /**
     * Constructeur
     * @param context
     */

    4 usages
    public ClientDAO(Context context){
        dao = new DAO(context);
        db = dao.open();
    }
```

```
/**
 * Fermeture de la base de données
 */
3 usages
public void close() { dao.close(); }
```

```
public Cursor readLesClients() {
    String reqSql = "Select id as '_id', nom, prenom, adrMail, numTel, adrPostale FROM " + TABLE_CLIENT +";";
    // Execution de la requête
    Cursor c = db.rawQuery(reqSql, selectionArgs: null);
    Log.d( tag: "bdd",  msg: "le curseur contient " + c.getCount() + " lignes");
    return c;
}
```

```java
public long create(Client c) {
    ContentValues values = new ContentValues();
    values.put("nom", c.getNom());
    values.put("prenom", c.getPrenom());
    values.put("adrMail", c.getAdrMail());
    values.put("numTel", c.getNumTel());
    values.put("adrPostale", c.getAdrPostale());
    Log.d( tag: "bdd",  msg: "insert, :" + c);
    return db.insert(TABLE_CLIENT,  nullColumnHack: null, values);
}

1 usage
public void deleteClient(int idClient){
    db.delete( table: "client",  whereClause: "id = ?", new String[]{String.valueOf(idClient)});
    db.close();
}
```

```java
1 usage
public void updateClient(int idClient, String nom, String prenom, String adrMail, String numTel, String adrPostale) {
    ContentValues values = new ContentValues();
    values.put("nom", nom);
    values.put("prenom", prenom);
    values.put("adrMail", adrMail);
    values.put("numTel", numTel);
    values.put("adrPostale", adrPostale);

    db.update( table: "client", values,  whereClause: "id = ?", new String[]{String.valueOf(idClient)});
}
```

La classe **InterventionDAO** :

La classe InterventionDAO sert à gérer les opérations CRUD (Créer, Lire, etc.) pour les interventions dans l'application Dépann'tout. Elle permet de lire les interventions existantes et d'en créer de nouvelles dans la base de données SQLite.

Elle est constitué de :

- dao : un objet de la classe DAO qui sert à gérer la connexion à la base de données.

- db : un objet SQLiteDatabase représentant la base de données ouverte.

- Un constructeur InterventionDAO(Context context) : Crée une instance de DAO et ouvre la connexion à la base de données via dao.open().

- Méthodes :
- close() : Ferme proprement la base de données en appelant dao.close().

- readLesInterventions() : Exécute une requête SQL (SELECT) pour récupérer la liste des interventions. Utilise db.rawQuery() pour exécuter la requête. Retourne un Cursor contenant toutes les interventions avec : id, idCli, dateTime, observation.

- create(Intervention i) : Insère une nouvelle intervention dans la base. Récupère les infos de l'objet Intervention (ID client, date, heure, observation). Utilise ContentValues pour préparer les données, puis db.insert() pour les insérer dans la table intervention.

```
public class InterventionDAO {

    3 usages
    private DAO dao = null;
    3 usages
    private SQLiteDatabase db = null;
    /**
     * Constructeur
     * @param context
     */

    2 usages
    public InterventionDAO(Context context){
        dao = new DAO(context);
        db = dao.open();
    }
```

```java
public InterventionDAO(Context context){
    dao = new DAO(context);
    db = dao.open();
}


/**
 * Fermeture de la base de données
 */

2 usages
public void close() {

    dao.close();
}
```

```java
public Cursor readLesInterventions() {
    String reqSql = "Select id as '_id', idCli as 'idCli', dateTime, observation FROM " +
            CreateBdDepannTout.TABLE_INTER +";";
    // Execution de la requête
    Cursor c = db.rawQuery(reqSql, selectionArgs: null);
    Log.d( tag: "bdd", msg: "le curseur contient " + c.getCount() + " lignes");
    return c;
}
```

```java
public long create(Intervention i ) {
    ContentValues values = new ContentValues();
    values.put("idCli", i.getIdCli());
    values.put("dateTime", i.getDateTime());
    values.put("observation", i.getObservation());
    Log.d( tag: "bdd", msg: "insert, :" + i);
    return db.insert(CreateBdDepannTout.TABLE_INTER, nullColumnHack: null, values);
}
```