RAMASTOP



BENITO Anthony

THABET Jihad

THIRUCHELVAM Parthipan

TRAING Sereypich

Lycée Jean Rostand - Année 2015-2016

sommaire

PRÉSENTATION GÉNÉRALE 1

CRÉATION ET LECTURE DES CARTES 3

GÉNÉRATION DES CARTES 3

Lecture des codes QR 5

APPLICATION ANDROID 9

Fonctionnement de l'application 9

Les outils nécessaires au développement 10

Développement 11

Main 12

La classe Enfant 13

EnfantsBDD 13

Saisir manuellement 16

Afficher les enfants descendants 17

Transférer un fichier 18

Informations utiles 22

Difficultés rencontrées 23

Webographie 23

# PRÉSENTATION GÉNÉRALE

Le projet « Ramastop » est un projet ayant pour objectif de de permettre à la mairie de Villepinte d'automatiser sa gestion du ramassage scolaire.

Actuellement, le comptage des enfants se fait par pointage sur une feuille Excel, ce pointage est réalisé par l'un des responsables du ramassage scolaire, qui se trouve dans le bus. Le pointage est effectué chaque matin et chaque soir, sur une feuille différente, sauf le mercredi où le 2eme pointage a lieu à midi puisque les enfants n'ont pas école l'après-midi. Le pointage sert à effectuer des statistiques : le taux de remplissage du bus chaque matin, chaque soir ainsi que le taux de remplissage mensuel du bus puisque les statistiques sont actualisées à la fin de chaque mois. Il existe actuellement deux ramassage scolaires à Villepinte : celui du Vert-Galant et Paul Langevin.

Le projet « Ramastop » consiste à permettre aux responsables du ramassage scolaire de remplacer leurs feuilles par une tablette sous Android, sur laquelle il n'aurait qu'à scanner la carte de chaque enfant, qui contiendra un code QR. L'inscription des élèves se fera au moyen d'un site web, sur lequel il sera possible de modifier ou supprimer un enfant qui n'utiliserais plus le bus de ramassage scolaire par exemple. L'application Android, conformément au cahier des charges donnera donc la possibilité à la personne située dans le bus de scanner le code QR d'un élève, mais également de pouvoir effectuer le pointage manuellement dans le cas où l'élève aurait oublié sa carte, chaque scannage mettra automatiquement à jour la base de données contenant les statistiques. Le responsable aura ainsi la possibilité de consulter un certain nombre d'informations concernant l'enfant (nom, prénom, école, numéro de téléphone des parents, arrêt où il descend, photo), de consulter la liste des enfants descendant à chaque arrêt etc. Il sera également possible de mettre à jour la base de données de la tablette en utilisant le site web et vice-versa.

Le projet a été reparti en quatre parties :

- La création de la page web

- La gestion de la base de données

- Le développement du générateur de cartes ainsi et du lecteur des codes QR

- Le développement de l'application Android

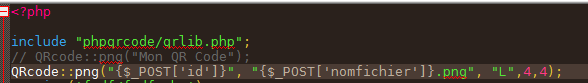
# CRÉATION ET LECTURE DES CARTES

## GÉNÉRATION DES CARTES

La première étape de ma partie consiste à générer les cartes sous la forme d'un fichier pdf, ces cartes doivent contenir un code QR, et les données suivantes :

* Nom
* Prénom
* École
* Arrêt où descend l’enfant
* Année scolaire
* Si l’enfant rentre seul
* S'il utilise le ramassage le matin, le midi ou le soir.

Les données sont enregistrées au moyen d'une page web contenant ce formulaire :

L'identifiant est la donnée qui sera contenue dans le code QR, et qui permettra après lecture d'aller chercher dans la base de données, l'élève avec l'identifiant qui correspond, pour avoir accès à ses données. Le fait de ne mettre qu'un chiffre dans le code QR et non pas directement les données que l'on veut voir affichée dans le code QR permet d’accélérer la vitesse du scan car le code est moins lourd, ce qui est beaucoup plus pratique dans le cadre d'un bus avec une trentaine d'enfants à scanner. Le code QR est généré grâce à la librairie phpqrcode, qui permet d'enregistrer un code QR au format png contenant la donnée que l'on veut y mettre, de donner un titre à l'image et de définir sa taille.

La génération du fichier PDF se fait grâce à la librairie fpdf, qui est une librairie permettant de créer un fichier pdf en php, et surtout de le mettre en page, ce qui nous permet de créer des cartes qui ressemblent à celles-ci :

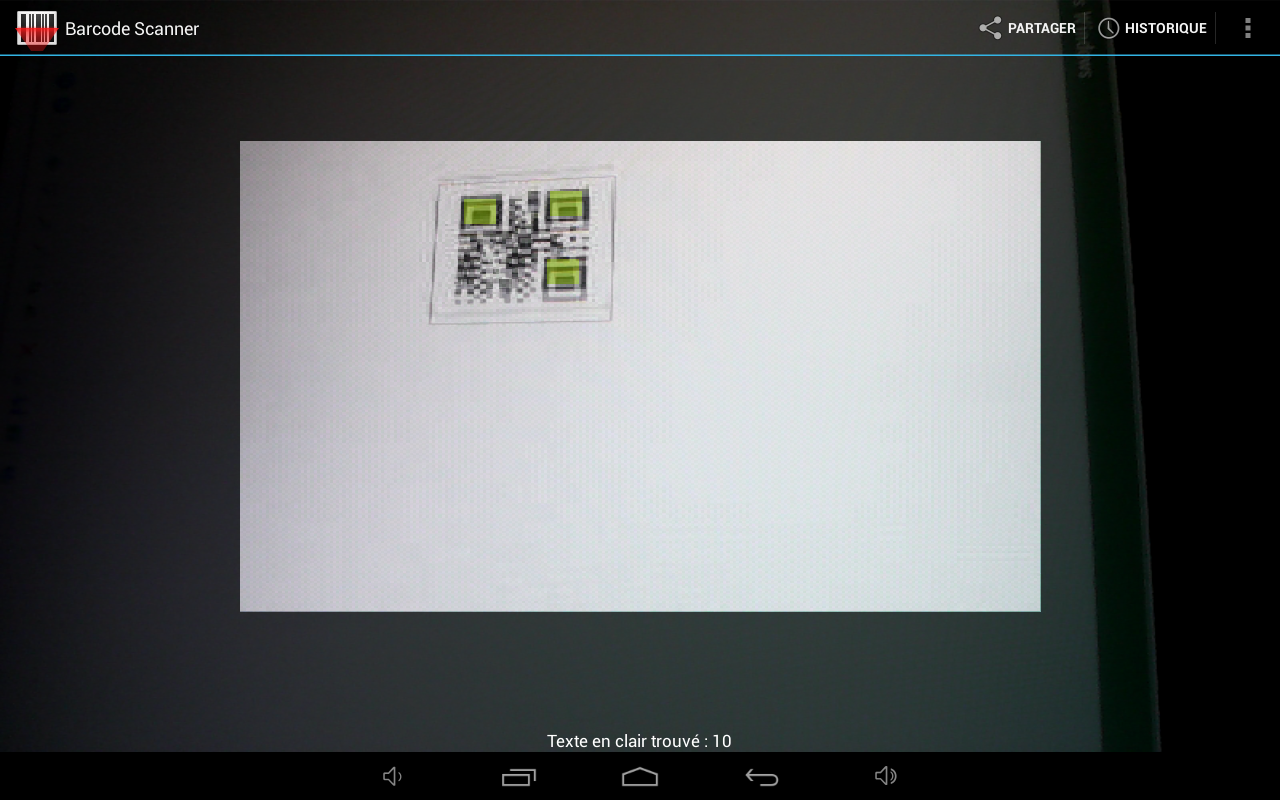
Une fois le fichier pdf mis en page, et les cartes contenant les données entrées dans le formulaire, il ne reste plus qu'à imprimer, découper les cartes, coller la photo de l'élève et les plastifier.

## Lecture des codes QR

Le développer d'une application capable de scanner les codes QR que nous pourrions ensuite intégrer à l'application « Ramastop » s'est fait en JAVA grâce à Android Studio, qui est un logiciel permettant d'émuler un système Android sur son PC, de générer des fichiers .apk, qui sont les fichiers installable sous Android, l'équivalent des .exe de Windows.

L'application permettant de scanner utilise l'application « Barcode Scanner », développée par Zxing, cette application est un projet open source, ce qui signifie que son code source peut être facilement consultable sur Internet, cette méthode nécessite cependant que l'application soit installée sur la tablette de l'utilisateur, ainsi l'application que je devais développer n'était pas le scanner en lui même, mais ce qui permet de récupérer la donnée scannée sous la forme d'une chaîne de caractères pour la rendre utilisable par le reste de l'application « Ramastop ». Le fonctionnement de l'application est très simple :

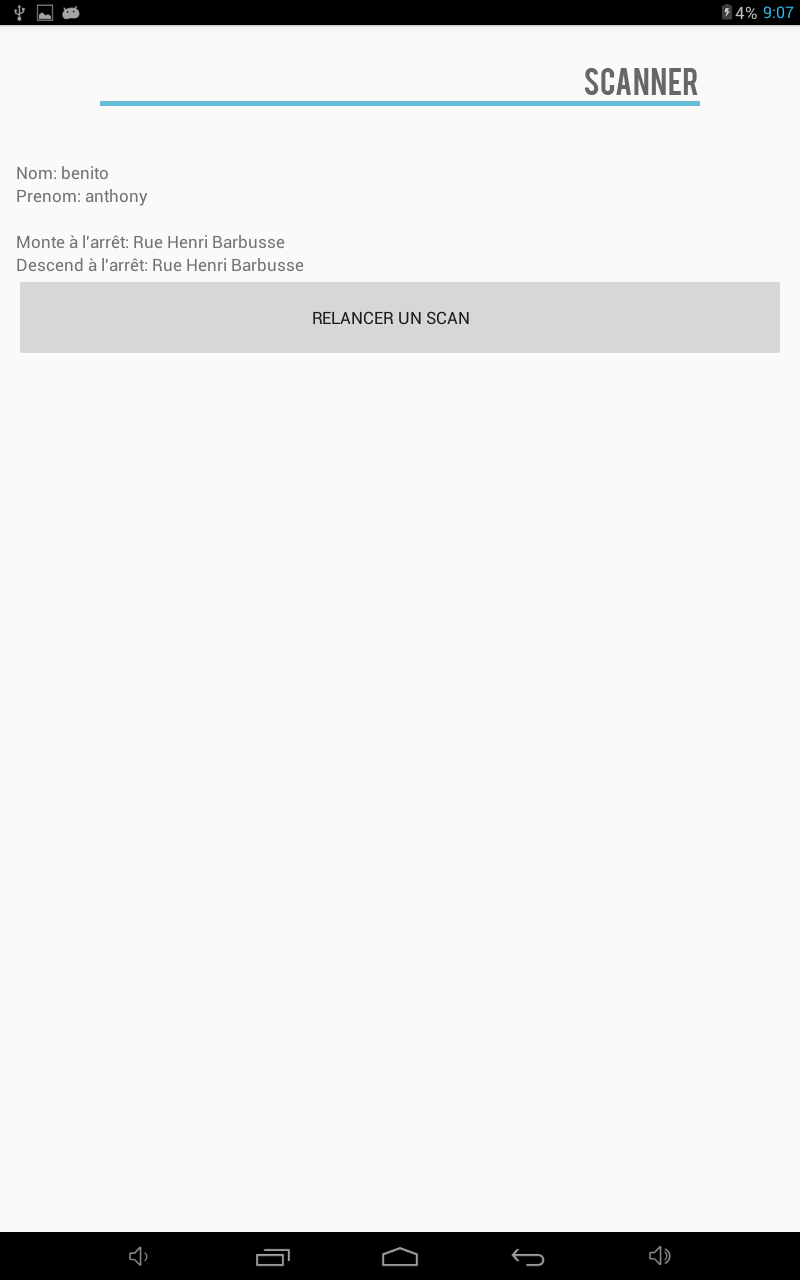
Avant d'être intégrée à l'application principale l'interface est un semble bouton scan, dont le clic va ouvrir l'application Barcode Scanner de Zxing.



On procède donc au scan grâce à Barcode Scanner et on peut voir que l'id contenue dans ce code QR est « 10 »

Cette valeur nous est ensuite retournée, avec le format du code que l'on a scanné (l'application de Zxing permet en effet de scanner des codes QR mais également plusieurs formats de codes barre) .

Après avoir été intégrée à l'application Ramastop, le résultat retourné par le scan n'est plus l'id « 10 » mais les informations concernant l'élève dont l'Id est « 10 » dans la base de donnée de l'application, ce qui met également à jour ses statistiques (nombre de présences).



# APPLICATION ANDROID

Cette application changera les habitudes et la façon de vérifier la présence d’un enfant dans le bus. Le responsable devra simplement scanner la carte d’un enfant montant dans le bus. L’application facilitera alors la mise à jour des données qui était à ce jour faite sur un fichier Excel.

## Fonctionnement de l'application

Le développement de cette application a pour objectif de faciliter la validation des enfants montant dans le bus en scannant le QR Code imprimé sur la carte de chaque enfant. Le scannage se fait à l'aide de la tablette. Les informations de l'enfant scanné sont donc ensuite affichées à l'écran comme le nom, le prénom, l'arrêt où il doit monter et descendre... L'application donnera la possibilité d'afficher l'ensemble des enfants inscrits à ce service de ramassage. Elle affichera au responsable la liste des enfants qui devront descendre à l'arrêt choisi. Elle pourra rechercher un enfant si ce dernier n'a pas sa carte et mettra à jour la base de données en le mettant présent dans le bus. Des informations utiles, avec des numéros de téléphone ou adresses seront disponibles pour les responsables. Un menu donnera la possibilité au responsable de stocker les informations de la base de données dans un fichier de type JSON, qui pourra être récupéré, et inversement, c'est-à-dire mettre à jour la base de données de l'application grâce à un fichier JSON.

## Les outils nécessaires au développement

Le développement de l’application nécessite plusieurs outils, voici la liste de ce dont nous avons besoin :

* Android Studio

Le développement de l'application se déroulera entièrement sur ce logiciel. Il est conseillé d'utiliser ce logiciel pour développer sous Android mais Eclipse aurait pu être aussi utilisé. Ce logiciel offre donc au développeur un large choix, plusieurs supports sont disponibles comme le téléphone, la tablette ou encore la montre connectée, plusieurs appareils virtuels sont à disposition afin d'avoir une vue globale de son application... Le logiciel est entièrement gratuit et est disponible à tous.

* SQLite

SQLite est un système de gestion de base de données, comme MySQL dont il diffère par le regroupement en un seul fichier de toutes les tables stockées dans la base de données. Ces bases de données sont donc facilement portables d’un appareil à un autre. La base de données se comporte presque exactement de la même manière qu’une base de données MySQL ; le langage de requête SQL restant, le même.

* Tablette Android

Une tablette sous Android est nécessaire pour avoir le vrai rendu de l'application. Il est mieux de pouvoir tester son application sur une tablette réelle qu'une machine virtuel d'Android Studio qui a parfois des problèmes d'affichage alors que la tablette de ne les auront pas. Une version 4.2 Jelly Bean d'Android est par contre nécessaire pour pouvoir faire fonctionner l'application.

## Développement

Les commentaires du code seront enlevés afin d’avoir plus de lisibilité.

### Main



Cette activité sera la première lancée lorsque l'application démarrera. Elle permet à l'utilisateur de choisir ce qu'il souhaite faire. Cette activité repose essentiellement sur le switch / case. Il permet de choisir selon l'id, de faire une activité. Chaque image envoie vers une autre activité.

On déclare avant les variables en indiquant les différentes images et on utilise la fonction setOnClickListener qui va rendre les éléments cliquables.

### La classe Enfant

On crée une classe Enfant qui contiendra les informations d’un enfant, c’est-à-dire :

* Nom
* Prénom
* Le nombre de présence dans le bus
* L’arrêt où il monte
* L’arrêt où il doit descendre
* QR Code

### EnfantsBDD

Cette classe est très importante car c’est elle qui aura la charge de créer et traiter la base de données SQLite à l’aide de fonctions. Nous détaillerons les points importants de cette classe :

La classe hérite de SQLiteOpenHelper. Tout d’abord, nous déclarons les noms de la base de données, de la table, ainsi que de chaque attribut de l’enfant que nous avons vu précédemment. Ces variables sont static afin qu’elles soient utilisables partout dans la classe et le final permet de déclarer une constante, le nom de la variable est alors écrit en majuscule.

**public void** onCreate(SQLiteDatabase db)  
{String CREATE\_CONTACTS\_TABLE = **"CREATE TABLE "** + ***TABLE\_ENFANTS*** + **"("** + ***KEY\_ID*** + **" INTEGER PRIMARY KEY,"** + ***KEY\_NAME*** + **" TEXT,"** + ***KEY\_PRENOM*** + **" TEXT,"**+***KEY\_PRESENT***+ **" INTEGER,"**+***KEY\_ARRET\_MONTER***+**" INTEGER,"** +***KEY\_ARRET\_DESCENDRE***+**" INTEGER"** + **")"**;

db.execSQL(CREATE\_CONTACTS\_TABLE);  
}

La fonction onCreate s’occupera de créer la base de données. On met dans une variable, la commande pour créer la table puis on l’exécute ensuite

**public void** addEnfant(Enfant enfant)  
{SQLiteDatabase db = **this**.getWritableDatabase();ContentValues values = **new** ContentValues();  
 values.put(***KEY\_NAME***, enfant.getNom());values.put(***KEY\_PRENOM***, enfant.getPrenom());values.put(***KEY\_PRESENT***, enfant.getPresent());  
 values.put(***KEY\_ARRET\_MONTER***, enfant.getArret\_id\_monter());  
 values.put(***KEY\_ARRET\_DESCENDRE***, enfant.getArret\_id\_descendre());

db.insert(***TABLE\_ENFANTS***, **null**, values);  
 db.close();  
}

Cette fonction servira à ajouter un enfant, qui sera mis en paramètre, dans la base de données. Nous déclarons et affectons à une variable *db* de type SQLiteDatabase, this.getWritableDatabase, ceci permet d’accéder à un objet SQLiteDatabase en écriture.

L'objet ContentValues permet de définir des clés/valeurs. La clé représente l'identifiant de la colonne de la table et la valeur représente le contenu de l'enregistrement dans cette colonne. ContentValues peut être utilisé pour insertions et mises à jour des enregistrements de la base de données.

On insère ensuite à la base de données les valeurs que l’on a récupérées en indiquant le nom de la table.

**public** Enfant getEnfant(**int** id)  
{  
 SQLiteDatabase db = **this**.getReadableDatabase();  
Cursor cursor = db.query(***TABLE\_ENFANTS***, **new** String[]{***KEY\_ID***,  
 ***KEY\_NAME***, ***KEY\_PRENOM***,***KEY\_PRESENT***, ***KEY\_ARRET\_MONTER***, ***KEY\_ARRET\_DESCENDRE***}, ***KEY\_ID*** + **"=?"**,  
 **new** String[]{String.*valueOf*(id)}, **null**, **null**, **null**, **null**);  
 **if** (cursor != **null**)  
 cursor.moveToFirst();  
Enfant enfantFromBdd = **new** Enfant(Integer.*parseInt*(cursor.getString(0)),  
 cursor.getString(1), cursor.getString(2),Integer.*parseInt*(cursor.getString(3)),  
 Integer.*parseInt*(cursor.getString(4)),Integer.*parseInt*(cursor.getString(5)));**return** enfantFromBdd;  
}

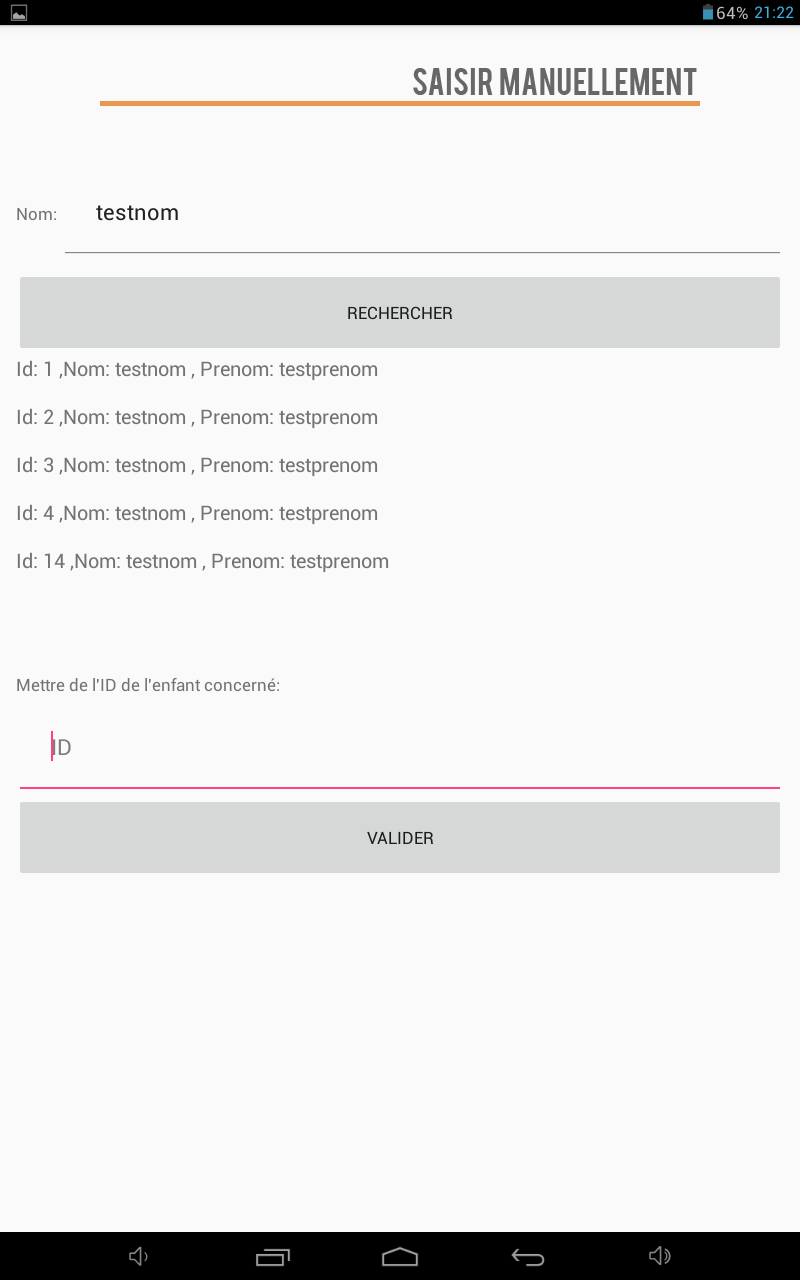
La fonction permet de renvoyer les informations d’un enfant. Nous déclarons et affectons à une variable *db* de type SQLiteDatabase, this.getReadableDatabase, ceci permet d’accéder à un objet SQLiteDatabase en lecture cette fois-ci. Nous créons une variable de type Cursor car une requête retourne un objet de type Cursor. Un curseur représente le résultat d’une requête. Dans cette variable on demande à récupérer toute les informations et on choisit cet enfant s’ils ont le même ID. On crée alors un nouvel enfant qui va avoir les informations de l'enfant trouvé dans la base de données. A la fin, on renvoie l’enfant.

**public** List<Enfant> getAllEnfants()  
{  
 List<Enfant> enfantList = **new** ArrayList<Enfant>();String selectQuery = **"SELECT \* FROM "** + ***TABLE\_ENFANTS***;  
  
 SQLiteDatabase db = **this**.getWritableDatabase();  
 Cursor cursor = db.rawQuery(selectQuery, **null**);  
**if** (cursor.moveToFirst()) {  
 **do** {  
 Enfant enfant = **new** Enfant();  
 enfant.setId(Integer.*parseInt*(cursor.getString(0)));  
 enfant.setNom(cursor.getString(1));  
 enfant.setPrenom(cursor.getString(2));  
 enfant.setPresent(Integer.*parseInt*(cursor.getString(3)));  
 enfant.setArret\_id\_monter(Integer.*parseInt*(cursor.getString(4)));  
 enfant.setArret\_id\_descendre(Integer.*parseInt*(cursor.getString(5)));  
  
 enfantList.add(enfant);  
 } **while** (cursor.moveToNext());  
 }**return** enfantList;  
}

On veut avec cette fonction récupérer tout les enfants de la base de données pour cela, nous créons une liste de type Enfant qui contiendra les enfants. Le résultat de la requête est stocké dans la variable de type Cursor. Nous créons ensuite une boucle qui créera un nouvel Enfant pour chaque enfant de la base de données.

On lui attribue toutes ses informations. Nous retournons pour terminer la liste.

### Saisir manuellement



L’activité sera utilisée si un enfant n’a pas sa carte. Le responsable devra alors saisir le nom de l’enfant en question. Une fois le nom entré, une liste d’enfant s’affichera avec le nom, le prénom et l’ID de l’enfant. Le responsable n’aura plus qu’à entrer l’ID de l’enfant puis valider son choix pour indiquer la présence de ce dernier dans le bus.

### Afficher les enfants descendants



Dans cette activité, on affiche la liste des enfants qui doivent descendre à un arrêt choisi. Une liste des arrêts est affichée à l’utilisateur et doit simplement choisir l’arrêt pour que la liste des enfants apparaisse. On récupère la liste en utilisant la fonction getAllEnfantsParArret() et en paramètre on mettra l’ID de l’arrêt souhaité. Un switch / case donnera l’ID à mettre en paramètre à la fonction.

### Transférer un fichier



Cette activité est une pièce importante de l’application. Elle permet de d’enregistrer dans un fichier JSON, la base de données ; et inversement, mettre à jour sa base de données à partir d’un fichier JSON mis dans la tablette. Nous détaillerons les fonctions importantes de cette activité.

**private void** creerFile()  
{File sdCard = Environment.*getExternalStorageDirectory*();File dir = **new** File (sdCard.getAbsolutePath() + **"/nine/baguettes"**);  
dir.mkdirs();  
**file** = **new** File(dir, **"alk2.json"**);  
 **if**(!**file**.exists()) {  
 **try** {  
  
 JSONArray jsa = **new** JSONArray();  
 JSONObject jso = **new** JSONObject();  
 jso.put(**"nom"**, **"testnom"**);  
 jso.put(**"prenom"**, **"testprenom"**);  
 jso.put(**"present"**, 1);  
 jso.put(**"arret\_monter"**,1);  
 jso.put(**"arret\_descendre"**,1);  
 jsa.put(jso);  
  
 FileWriter f = **new** FileWriter(**file**);  
 BufferedWriter bw = **new** BufferedWriter(f);  
 bw.write(jsa.toString());  
 bw.close();  
  
  
 } **catch** (IOException | JSONException e) {  
 e.printStackTrace();  
 }  
 }  
}

Nous devons d’abord créer le fichier JSON, pour cela on récupère la racine et on l’affecte à la variable. Puis on crée une autre variable de type File et on transmet en paramètre la racine ainsi que le chemin où l’on souhaite écrire le fichier. On crée une dernière fois une variable de type File, qui contiendra le nom de la précédente variable File ainsi que le nom du fichier.

On crée un JSONArray qui contiendra des JSONObject, ces JSONObject auront les noms des attributs ainsi que les valeurs. On utilise ensuite des variables de type BufferedWriter et FileWriter pour écrire dans un fichier.

**private** String readFile(File f) **throws** IOException  
{  
 FileReader filereader = **new** FileReader(f);  
 BufferedReader br = **new** BufferedReader(filereader);  
 String line;  
 String s = **""**;  
  
 **while** ((line = br.readLine()) != **null**) {  
 s += line;  
 }  
  
 br.close();  
 **return** s;  
}

Cette fonction permet de lire le fichier mis en paramètre. On met dans la variable line, la ligne qu'il y a dans le fichier f grâce au BufferedReader. Lorsque la ligne est finie, on l'ajoute ensuite dans la variable s qui contiendra toutes les lignes.

**private void** writeFile(File f, String s) **throws** IOException  
{  
 FileWriter fw = **new** FileWriter(f);  
 BufferedWriter bw = **new** BufferedWriter(fw);  
 bw.write(s);  
 bw.close();  
}

On transmet en paramètre dans quel fichier nous voulons écrire et la chaîne de caractère à écrire. Puis on utilise la même méthode vue précédemment.

**private void** ajouter\_enfant(String nom\_parametre, String prenom\_parametre, **int** present\_param, **int** arret\_monter\_param, **int** arret\_descendre\_param)  
{**if** (**file**.canRead())  
 {  
 **try** {  
  
 String s = readFile(**file**);

JSONArray jsa = **new** JSONArray(s);JSONObject jso = **new** JSONObject();  
 jso.put(**"nom"**, nom\_parametre);  
 jso.put(**"prenom"**, prenom\_parametre);  
 jso.put(**"present"**, present\_param);  
 jso.put(**"arret\_monter"**,arret\_monter\_param);  
 jso.put(**"arret\_descendre"**,arret\_descendre\_param);  
jsa.put(jso);writeFile(**file**, jsa.toString());  
  
  
 } **catch** (IOException | JSONException e) {  
 e.printStackTrace();  
 }  
 }  
}

La fonction ajoute un enfant à la base de données. Toutes les informations de l’enfant doivent être mis en paramètre. On vérifie d’abord que le fichier peut être lu et ensuite on essaye de récupérer dans une variable String l’ensemble du fichier. Puis nous créons un JSONArray à partir du contenu du fichier en passant en paramètre la variable String. Nous ajoutons ensuite un JSONObject qui contient alors toutes les informations du nouvel enfant dans la base de données. Et on finit par écrire dans le fichier le nouveau JSONArray.

### Informations utiles



Cette partie de l’application contient seulement du texte. Elle permet de mettre à disposition du responsable des numéros qui peuvent être utiles en cas de problème par exemple.

## Difficultés rencontrées

Durant le développement de l’application, plusieurs problèmes et difficultés sont apparus. Le premier est arrivé pour l’installation d’Android Studio. Un problème de droit et de proxy posait des problèmes lors de la phase d’installation malgré que le compte soit celui de l’administrateur. Il a fallu une semaine et de nombreux essais pour réussir à installer le logiciel. Le deuxième était matériel, l’ordinateur fourni pour le projet, était trop juste aux niveaux des caractéristiques car Android Studio demande un minimum de ressource. Les tests et compilations prenaient énormément de temps, cela pouvait prendre plus de 20 minutes. Il était difficile d’avancer correctement dans ces conditions. L’achat d’un ordinateur portable personnel était donc nécessaire pour ma part.

L’utilisation des fonctions spécifiques à Android n’était pas facile pendant les premières heures de développement. Beaucoup de nouveaux termes apparaissaient et ce moment d’adaptation nécessitaient quelques heures. Un passage sur le site d’OpenClassrooms était indispensable pour connaître les bases d’Android et prendre de suite de bonne habitude.

Après des premières heures compliquées, l’application commençait à prendre forme. A force de pratiquer, des habitudes se créaient et permettaient d’avancer plus rapidement et avec plus d’efficacité.

## Webographie

* http://mobilesiri.com
* http://stackoverflow.com
* http://forum.frandroid.com
* http://openclassrooms.com