Databases

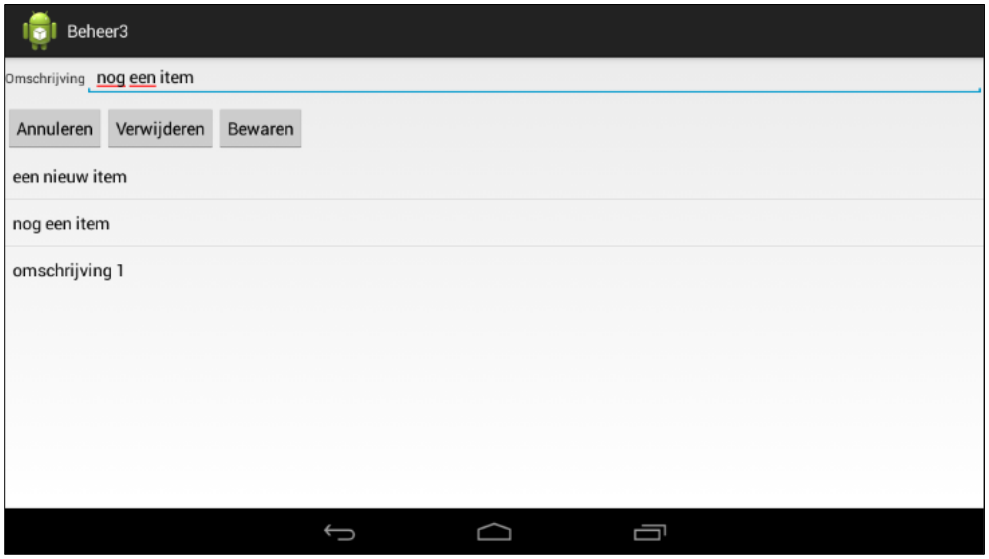
In Android kan je een SQLite database gebruiken om gegevens in op te slaan. Je kan ook een verbinding maken met een MySQL database die zich op het internet bevindt.

Android heeft een "SQLiteOpenHelper" klasse waarin alle standaard methodes om met een SQLite database te werken zijn gedefinieerd. Je kan een database helper klasse aanmaken die afgeleid is van de "SQLiteOpenHelper" klasse.

In jouw app voorzie je de nodige invulvelden en knoppen om de gegevens toe te voegen, te wijzigen en te verwijderen. Daarnaast kan je ook de nodige functies maken om 1 record of alle records uit een database tabel op te halen en te tonen in jouw app.

Het is een goed idee om per database tabel een aparte klasse te definiëren om de inhoud van de velden in te bewaren. Een object van die klasse kan dan eenvoudig doorgegeven worden aan de database functies in de database helper klasse.

In het voorbeeld wordt een database tabel "items" opgevuld met per item een naam. Onder de invulvelden wordt ook een lijst getoond van de reeds bestaande records (in een "ListView").



Er zijn ook 3 knoppen voorzien: "Annuleren", "Verwijderen" en "Bewaren". In de "onClickListener()" functies van de "Verwijderen" en "Bewaren" knoppen worden de database acties uitgevoerd. Met de knop "Annuleren" wordt het invulveld "Omschrijving" leeggemaakt.

```
package com.android.beheer3;

import java.util.ArrayList;
import java.util.List;
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.AdapterView;
```

```
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.AdapterView.OnItemClickListener;

public class MainActivity extends Activity {
  private Button btncancel;
  private Button btndelete;
  private Button btnsave;
  private ListView lvitems;
  private EditText etnaam;
  private DBHelper objDbHelper;
  private int rowid;
  private Item objitem;
  ArrayList<String> itemsarray;
  ArrayList<String> itemsids;
  ArrayAdapter<String> itemsadapter;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    objDbHelper = new DBHelper(this);

    etnaam = (EditText) findViewById(R.id.editText1);

    lvitems = (ListView) findViewById(R.id.listView1);
    itemsadapter = getitems();
    lvitems.setAdapter(itemsadapter);
    lvitems.setOnItemClickListener(new OnItemClickListener() {
      public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        opvullenvelden(Integer.valueOf(itemsids.get(position)));
      }
    });

    btncancel = (Button) findViewById(R.id.button1);
    btndelete = (Button) findViewById(R.id.button2);
    btnsave = (Button) findViewById(R.id.button3);
    btncancel.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        etnaam.setText("");
        etnaam.requestFocus();
        rowid = 0;
      }
    });
    btndelete.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
         objitem = new Item(rowid,etnaam.getText().toString());
         objDbHelper.deleteitem(objitem);
         refreshscreen();
       }
    });
    btnsave.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
        if (rowid == 0) {
          // TODO: testen of velden niet leeg zijn en geen quotes bevatten
          objitem = new Item(etnaam.getText().toString());
          objDbHelper.createitem(objitem);
          refreshscreen();
        } else {
          // TODO: testen of velden quotes bevatten of dat de record reeds bestaat
```

```
            objitem = new Item(rowid,etnaam.getText().toString());
            objDbHelper.updateitem(objitem);
            refreshscreen();
          }
        }
      });
  }

   public ArrayAdapter<String> getitems() {
      DBHelper db = new DBHelper(this);
      List<Item> items = db.haalAlleitems();
      itemsarray = new ArrayList<String>();
      itemsids = new ArrayList<String>();
      for (Item test : items) {
          itemsarray.add(test.getnaam());
          itemsids.add(String.valueOf(test.getid()));
      }
      ArrayAdapter<String> itemsadapter = new ArrayAdapter<String>(this,
                                          android.R.layout.simple_list_item_1, itemsarray);
      return itemsadapter;
   }
   public void opvullenvelden(int id) {
      DBHelper db = new DBHelper(this);
      Item test = db.haalitem(id);
      rowid = id;
      etnaam.setText(test.getnaam());
   }
   public void refreshscreen() {
      etnaam.setText("");
      etnaam.requestFocus();
      itemsadapter = getitems();
      lvitems.setAdapter(itemsadapter);
      rowid = 0;
   }
}
```

```
package com.android.beheer3;

public class Item {
  int mid;
  String mnaam;

  public Item() {
  }

  public Item(int id, String naam) {
    this.mid = id;
    this.mnaam = naam;
  }
  public Item(String naam) {
    this.mnaam = naam;
  }
  public int getid() {
    return this.mid;
  }
  public String getnaam() {
    return this.mnaam;
  }
  public void setid(int id) {
    this.mid = id;
  }
  public void setnaam(String naam) {
```

```
    this.mnaam = naam;
  }
}
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >
   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="50dp"
      android:orientation="horizontal" >
      <TextView
         android:id="@+id/textView2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Omschrijving" />
      <EditText
         android:id="@+id/editText1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:ems="10" >
         <requestFocus />
      </EditText>
   </LinearLayout>
   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="50dp"
      android:layout_gravity="center_horizontal"
      android:orientation="horizontal" >
      <Button
         android:id="@+id/button1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="left"
         android:text="Annuleren" />
      <Button
         android:id="@+id/button2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="center_horizontal"
         android:text="Verwijderen" />
      <Button
         android:id="@+id/button3"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_gravity="right"
         android:gravity="center_vertical|center_horizontal|right"
         android:text="Bewaren" />
   </LinearLayout>
   <ListView
      android:id="@+id/listView1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >
   </ListView>
</LinearLayout>
```

Database helper klasse

In de voorbeeld klasse hieronder worden eerst een aantal constanten gedefinieerd. Met deze werkwijze kan je gemakkelijk de wijzigingen aan de database tabel structuur aanpassen in de code.

Uiteraard moet je op verschillende plaatsen de code aanpassen als je velden aan de database tabel structuur hebt toegevoegd.

De "DBHelper()" constructor functie maakt een verbinding met de SQLite database.

Met de "execSQL()" functie worden de data definitie SQL statements uitgevoerd.

Met de speciale functies "insert()", "update()" en "delete()" van de database helper klasse worden de SQL statements insert, update en delete uitgevoerd.

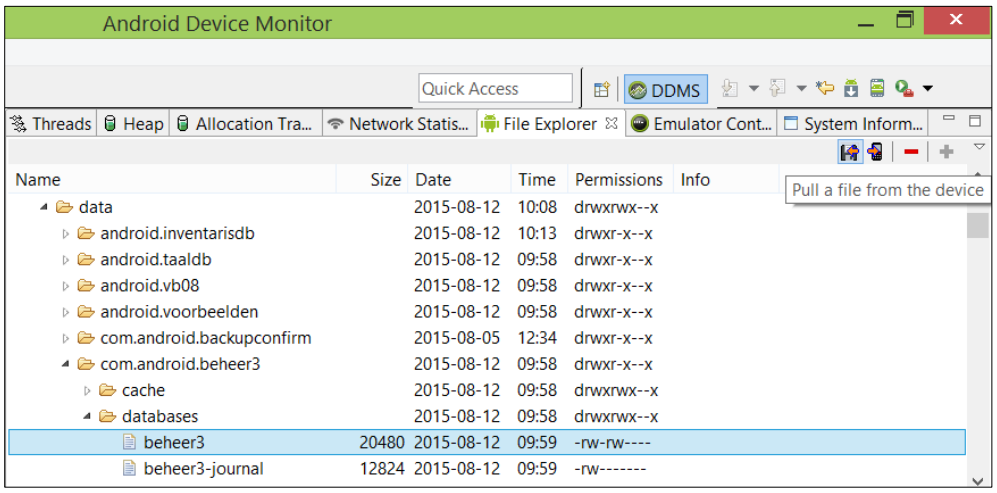
Met de functies "query()" en "rawQuery()" worden de SQL select statements uitgevoerd. De resultaten van de SQL select statements worden in een "Cursor" object bijgehouden. Met de functies "moveToFirst()" en "moveToNext()" van het "Cursor" object worden de records in de cursor (of recordset) afgelopen.

```
package com.android.beheer3;

import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

   // database
   private static final String DB_NAME = "beheer3";
   private static final int DB_VERSION = 1;
   // tabellen
   private static final String TABLE_items = "items";
   public static final String KEY_ID = "id";
   public static final String FLD_NAAM = "naam";

   private static final String CREATE_ITEMS =
         "create table items (id integer primary key autoincrement, "
                                                   + "naam text not null);";

  public DBHelper(Context context) {
     super(context, DB_NAME, null, DB_VERSION);
   }
   @Override
   public void onCreate(SQLiteDatabase db) {
     db.execSQL(CREATE_ITEMS);
   }
   @Override
   public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
     db.execSQL("DROP TABLE IF EXISTS " + TABLE_items);
     onCreate(db);
   }
   // item functies
   public void createitem(Item item) {
     SQLiteDatabase db = this.getWritableDatabase();
```

```
        ContentValues values = new ContentValues();
        values.put(FLD_NAAM, item.getnaam());
        db.insert(TABLE_items, null, values);
        db.close();
    }
    public int updateitem(Item item) {
      int result;
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(FLD_NAAM, item.getnaam());
        result = db.update(TABLE_items, values, KEY_ID + " = ?",
                                                 new String[] { String.valueOf(item.getid()) });
        db.close();
        return result;
    }
    public void deleteitem(Item item) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_items, KEY_ID + " = ?",
                                                 new String[] { String.valueOf(item.getid()) });
        db.close();
    }

    Item haalitem(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.query(TABLE_items, new String[] { KEY_ID,
                                                             FLD_NAAM }, KEY_ID + "=?",
                                        new String[] { String.valueOf(id) }, null, null, null, null);
        if (cursor != null) cursor.moveToFirst();
        Item item = new Item(Integer.parseInt(cursor.getString(0)),
                                                   cursor.getString(1));
        cursor.close();
        db.close();
        return item;
    }
   public List<Item> haalAlleitems() {
        List<Item> itemList = new ArrayList<Item>();
        String selectQuery = "SELECT  * FROM " + TABLE_items + " ORDER BY naam";
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        if (cursor.moveToFirst()) {
            do {
              Item item = new Item();
              item.setid(Integer.parseInt(cursor.getString(0)));
              item.setnaam(cursor.getString(1));
              itemList.add(item);
            } while (cursor.moveToNext());
        }
        cursor.close();
        db.close();
        return itemList;
    }
}
```

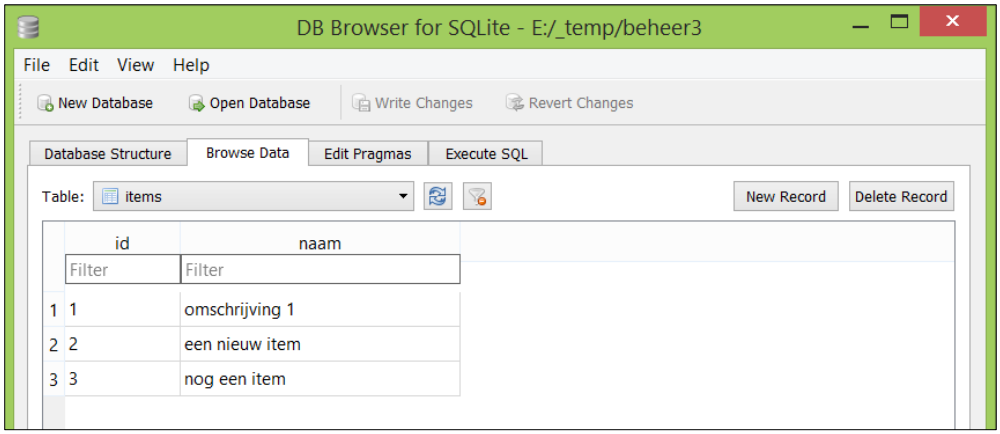
Als je de inhoud van een SQLite database wil ophalen dan kan je via de Android Device Monitor het "File Explorer" venster oproepen.
In de map "data/data/com.android.beheer3/databases" vind je het SQLite bestand ("beheer3") terug. Bovenaan rechts in dit venster kan je op de

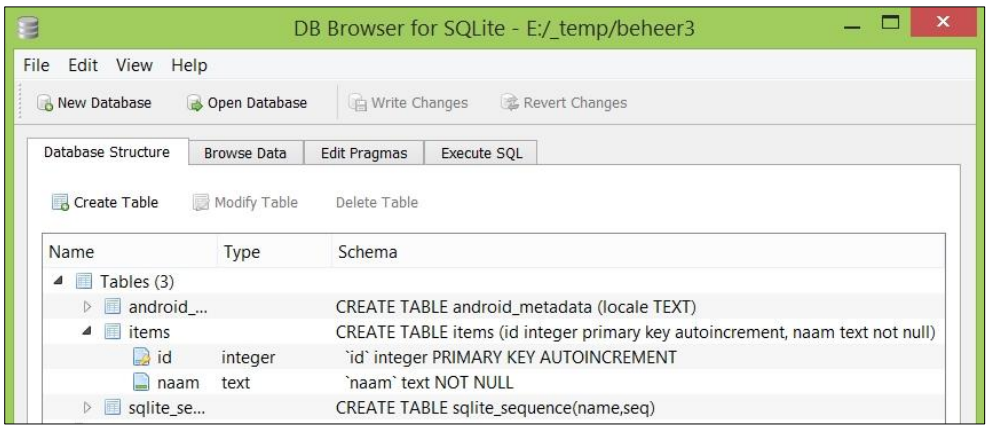
knop klikken ("Pull a file from the device") om het bestand op de harde schijf te kopiëren.



Met de SQLite database browser tool kan je het gekopieerde bestand bekijken en aanpassen. Let op: het gekopieerde bestand heeft geen extensie!

http://portableapps.com/apps/development/sqlite_database_browser_portable





JSONParser klasse

Met de JavaScript Object Notatie ("JSON") kan je gegevens in een MySQL database ophalen, toevoegen, wijzigen of verwijderen.

In de Java code van Android kan je een "JSONParser" klasse schrijven die de informatie van en naar het "JSON" object beheert.

In het voorbeeld worden voor elke bewerking in de database een "AsyncTask" (achtergrond taak) uitgevoerd. Met de "makeHttpRequest()" functie van het "JSONParser" object worden de PHP scripts uitgevoerd die de SQL statements in de MySQL database laat uitvoeren.

De inhoud van het JSON object ziet er als volgt uit:

```
Lijst: {"success":1,"Lokaal":[{"LokaalNaam":"F306","LokaalID":"1","LokaalIndelingUrl":"nog in te vullen"},{"LokaalNaam":"F205","LokaalID":"2","LokaalIndelingUrl":"nog in te vullen"}]}

Insert: {"message":"Record toegevoegd","success":1}
```

De array bevat een index "success" en een index "message" als er een bericht moet doorgegeven worden zoals bvb. "Record toegevoegd" ("success":1) of "Geen Lokaal gevonden" ("success":0).

In de index "Lokaal" worden de gegevens van de lokalen geplaatst in het formaat "veldnaam":"inhoud".

```
package android.inventarisdb;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;
```

```
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class LokaalFragment extends Fragment {
   // vanuit de AVD in Android is het loopback adres 10.0.2.2 (wampserver) of 10.0.2.2:8080
(uwamp)

   private static String url_lijst =
                          "http://10.0.2.2:8080/inventaris_DB/lokaal/lokaal_lijst.php";
   private static String url_select =
                         "http://10.0.2.2:8080/inventaris_DB/lokaal/lokaal_select.php";
   private static String url_insert =
                         "http://10.0.2.2:8080/inventaris_DB/lokaal/lokaal_insert.php";
   private static String url_update =
                        "http://10.0.2.2:8080/inventaris_DB/lokaal/lokaal_update.php";
   private static String url_delete =
                         "http://10.0.2.2:8080/inventaris_DB/lokaal/lokaal_delete.php";

/*
   private static String url_lijst =
                                 "http://<jouw domein>/inventaris_DB/lokaal/lokaal_lijst.php";
   private static String url_select =
                                "http://<jouw domein>/inventaris_DB/lokaal/lokaal_select.php";
   private static String url_insert =
                                "http:// <jouw domein>//inventaris_DB/lokaal/lokaal_insert.php";
   private static String url_update =
                               "http:// <jouw domein>//inventaris_DB/lokaal/lokaal_update.php";
   private static String url_delete =
                                "http:// <jouw domein>//inventaris_DB/lokaal/lokaal_delete.php";
*/

   private static final String TAG_FRAGMENT = "LokaalFragment";
   private static final String TAG_SUCCESS = "success";
   private static final String TAG_TABEL = "Lokaal";
   private static final String TAG_LOKAALID = "LokaalID";
   private static final String TAG_LOKAALNAAM = "LokaalNaam";
   private static final String TAG_LOKAALINDELINGURL = "LokaalIndelingUrl";
   private String lokaalnaam;
   private EditText etlokaalnaam;
   private String lokaalindelingurl;
   private EditText etlokaalindelingurl;
   private ListView lv;
   private SimpleAdapter lvadapter;
   private String rowid;
   JSONParser jParser = new JSONParser();
   ArrayList<HashMap<String, String>> tabelList;
   JSONArray jarrtabel = null;
   JSONObject jobjtabel;

   public LokaalFragment() {
   }
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        setHasOptionsMenu(true);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_lokaal, container, false);
    }
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        try {
            getActivity().getWindow().setSoftInputMode(
                    WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
            etlokaalnaam = (EditText) getActivity().findViewById(R.id.et_lokaalnaam);
            etlokaalindelingurl = (EditText) getActivity().findViewById(R.id.et_lokaalindelingurl);
            lv = (ListView) getActivity().findViewById(R.id.lv_tabel);
            refreshscreen();
            lv.setOnItemClickListener(lvitemclick);
        } catch (Exception e) {
            Toast.makeText(getActivity(), e.toString(), Toast.LENGTH_SHORT).show();
            Log.v(TAG_FRAGMENT + " actCre", e.toString());       }
    }

    AdapterView.OnItemClickListener lvitemclick = new AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view, int position,
                                long id) {
            TextView txtpkid = (TextView) view.findViewById(R.id.pkid);
            fillfields(txtpkid.getText().toString());
        }
    };
    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        inflater.inflate(R.menu.scholen_menu, menu);
        super.onCreateOptionsMenu(menu, inflater);
    }
    @SuppressWarnings("finally")
    public boolean onOptionsItemSelected(MenuItem mnitem) {
        try {
            switch (mnitem.getItemId()) {
                case R.id.menu_settings:
                    return false;
                case R.id.menu_toevoegen:
                    insert_tabel();
                    return true;
                case R.id.menu_wijzigen:
                    update_tabel();
                    return true;
                case R.id.menu_verwijderen:
                    delete_tabel();
                    return true;
                case R.id.menu_ongedaan_maken:
                    refreshscreen();
                    return true;
                case R.id.menu_endapp:
                    return false;
                default:
                    return super.onOptionsItemSelected(mnitem);
            }
        } catch (Exception e) {
            Log.v(TAG_FRAGMENT + " options", e.toString());
        } finally {
            return true;
        }
    }
```

```
private boolean validatie() {
    if (etlokaalnaam.getText().toString() == null ||
                                        etlokaalnaam.getText().toString().trim().length() == 0) {
        Toast.makeText(getActivity(), "Lokaalnaam is leeg!", Toast.LENGTH_SHORT).show();
        return false;
    }
   return true;
}
public void fillfields(String id) {
    haalrecord(id);
}
public void refreshscreen() {
    haalrecordlijst();
    etlokaalnaam.setText("");
    etlokaalindelingurl.setText("");
    etlokaalnaam.requestFocus();
    rowid = "0";
}
public void haalrecord(String id) {
    rowid = id;
    new tabelSelect().execute();
}
public void haalrecordlijst() {
    tabelList = new ArrayList<HashMap<String, String>>();
    new tabelLijst().execute();
}
public void insert_tabel() {
    if (rowid == "0") {
        if (validatie()) {
            lokaalnaam = etlokaalnaam.getText().toString();
            lokaalindelingurl = etlokaalindelingurl.getText().toString();
            new tabelinsert().execute();
            refreshscreen();
        }
    }
}
public void update_tabel() {
    if (rowid == "0") {
        if (validatie()) {
            lokaalnaam = etlokaalnaam.getText().toString();
            lokaalindelingurl = etlokaalindelingurl.getText().toString();
            new tabelinsert().execute();
            refreshscreen();
        }
    } else {
        if (validatie()) {
            lokaalnaam = etlokaalnaam.getText().toString();
            lokaalindelingurl = etlokaalindelingurl.getText().toString();
            new tabelupdate().execute();
            refreshscreen();
        }
    }
}
public void delete_tabel() {
    AlertDialog.Builder alert = new AlertDialog.Builder(getActivity());
    alert.setTitle(R.string.deletedialog_title);
    alert.setMessage(etlokaalnaam.getText().toString());
    alert.setPositiveButton(R.string.deletedialogok, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int whichButton) {
            lokaalnaam = etlokaalnaam.getText().toString();
            lokaalindelingurl = etlokaalindelingurl.getText().toString();
            new tabeldelete().execute();
            refreshscreen();
```

```
        }
    });
    alert.setNegativeButton(R.string.deletedialogcancel, new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int whichButton) {
            // geen delete
        }
    });
    alert.show();
}
class tabelLijst extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    protected String doInBackground(String... args) {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        JSONObject json = jParser.makeHttpRequest(url_lijst, "POST", params);
        Log.d("Lokaal Lijst: ", json.toString());
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                jarrtabel = json.getJSONArray(TAG_TABEL);
                for (int i = 0; i < jarrtabel.length(); i++) {
                    JSONObject c = jarrtabel.getJSONObject(i);
                    String LokaalID = c.getString(TAG_LOKAALID);
                    String LokaalNaam = c.getString(TAG_LOKAALNAAM);
                    String LokaalIndelingUrl = c.getString(TAG_LOKAALINDELINGURL);
                    HashMap<String, String> map = new HashMap<String, String>();
                    map.put(TAG_LOKAALID, LokaalID);
                    map.put(TAG_LOKAALNAAM, LokaalNaam);
                    map.put(TAG_LOKAALINDELINGURL, LokaalIndelingUrl);
                    tabelList.add(map);
                }
            } else {
                // geen records
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
        getActivity().runOnUiThread(new Runnable() {
            public void run() {
                lvadapter = new SimpleAdapter(
                        getActivity(), tabelList,
                        R.layout.list_item, new String[] { TAG_LOKAALID,
                        TAG_LOKAALNAAM},
                        new int[] { R.id.pkid, R.id.lvtv });
                // updating listview
                lv.setAdapter(lvadapter);
            }
        });
    }
}
class tabelSelect extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    protected String doInBackground(String... args) {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
```

```
        params.add(new BasicNameValuePair("LokaalID", rowid));
        JSONObject json = jParser.makeHttpRequest(url_select, "POST", params);
        Log.d("Select: ", json.toString());
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                JSONArray LokaalObj = json.getJSONArray(TAG_TABEL);
                jobjtabel = LokaalObj.getJSONObject(0);
            } else {
                // geen record
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
        if (jobjtabel != null) {
            getActivity().runOnUiThread(new Runnable() {
                public void run() {
                    etlokaalnaam = (EditText) getActivity().findViewById(R.id.et_lokaalnaam);
                    etlokaalindelingurl = (EditText)
getActivity().findViewById(R.id.et_lokaalindelingurl);
                    try {
                        etlokaalnaam.setText(jobjtabel.getString(TAG_LOKAALNAAM));
                        etlokaalindelingurl.setText(jobjtabel.getString(
                                                        TAG_LOKAALINDELINGURL));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    }
}
class tabelinsert extends AsyncTask<String, String, String> {
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }
    protected String doInBackground(String... args) {
        List<NameValuePair> params = new ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("LokaalNaam", lokaalnaam));
        params.add(new BasicNameValuePair("LokaalIndelingUrl", lokaalindelingurl));
        JSONObject json = jParser.makeHttpRequest(url_insert, "POST", params);
        Log.d("Insert: ", json.toString());
        try {
            int success = json.getInt(TAG_SUCCESS);
            if (success == 1) {
                // status = "inserted";
            } else {
                // geen record
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }
    protected void onPostExecute(String file_url) {
    }
}
class tabelupdate extends AsyncTask<String, String, String> {
    @Override
```

```
        protected void onPreExecute() {
            super.onPreExecute();
        }
        protected String doInBackground(String... args) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("LokaalID", rowid));
            params.add(new BasicNameValuePair("LokaalNaam", lokaalnaam));
            params.add(new BasicNameValuePair("LokaalIndelingUrl", lokaalindelingurl));
            JSONObject json = jParser.makeHttpRequest(url_update, "POST", params);
            Log.d("Update: ", json.toString());
            try {
                int success = json.getInt(TAG_SUCCESS);
                if (success == 1) {
                    // status = "updated";
                } else {
                    // geen record
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
            return null;
        }
        protected void onPostExecute(String file_url) {
        }
    }
    class tabeldelete extends AsyncTask<String, String, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }
        protected String doInBackground(String... args) {
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("LokaalID", rowid));
            params.add(new BasicNameValuePair("LokaalNaam", lokaalnaam));
            params.add(new BasicNameValuePair("LokaalIndelingUrl", lokaalindelingurl));
            JSONObject json = jParser.makeHttpRequest(url_delete, "POST", params);
            Log.d("Delete: ", json.toString());
            try {
                int success = json.getInt(TAG_SUCCESS);
                if (success == 1) {
                    // status = "deleted";
                } else {
                    // geen record
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
            return null;
        }
        protected void onPostExecute(String file_url) {
        }
    }
}
```

```
package android.inventarisdb;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;
```

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;
import android.util.Log;

public class JSONParser {

   static InputStream is = null;
   static JSONObject jObj = null;
   static String json = "";

   public JSONParser() {
   }
   public JSONObject makeHttpRequest(String url, String method,
                              List<NameValuePair> params) {
      try {
         if (method == "POST") {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params));
            HttpResponse httpResponse = httpClient.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
         } else if(method == "GET") {
            DefaultHttpClient httpClient = new DefaultHttpClient();
            String paramString = URLEncodedUtils.format(params, "utf-8");
            url += "?" + paramString;
            HttpGet httpGet = new HttpGet(url);
            HttpResponse httpResponse = httpClient.execute(httpGet);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
         }
      } catch (UnsupportedEncodingException e) {
         e.printStackTrace();
      } catch (ClientProtocolException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      } catch (Exception e) {
         Log.e("Get content", "Error = " + e.toString());
      }
      try {
         Log.d("JSON parser url: ", url);
         if (is == null) {
            Log.d("JSON parser fout", "input stream is null");
         }
         BufferedReader reader = new BufferedReader(new InputStreamReader(
               is, "iso-8859-1"), 8);
         StringBuilder sb = new StringBuilder();
         String line = null;
         while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
         }
         is.close();
         Log.d("JSON input stream: ", is.toString());
```

```
            json = sb.toString();
        } catch (Exception e) {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }
        try {
            jObj = new JSONObject(json);
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }
        return jObj;
    }
}
```

Met de PHP scripts wordt een connectie gemaakt met de MySQL database. Met de select, insert, update en delete SQL statements worden de arrays ingevuld en de wijzigingen in de database doorgevoerd.

In het voorbeeld wordt gebruik gemaakt van de "PHP Data Objects" ("PDO") functies.

```
<?php
define('DB_WRITER', "schrijver");
define('DB_WRITER_PASSWORD', <password>);
define('DB_READER', "lezer");
define('DB_READER_PASSWORD', <password>);
define('DB_DATABASE', "inventarisDB");
define('DB_SERVER', "localhost");
?>
```

```
<?php
        // lijst
        $response = array();
        require_once '../includes/db_config.php';
        $host = DB_SERVER;
        $db = DB_DATABASE;
        $user = DB_READER;
        $pass = DB_READER_PASSWORD;
        try {
                $con = new PDO("mysql:host=$host;dbname=$db",$user,$pass);
        }
        catch(PDOException $err) {
          die("Connection error");  //, because: " . $err->getMessage());
        }
        $sql = "SELECT COUNT(*) FROM Lokaal";
        $q = $con->prepare($sql);
        $q->execute();
        if(!$q) {
                die("Execute query error");    //, because: ". $con->errorInfo());
        }
        if ($q) {
                if ($q->fetchColumn() > 0) {
                        $response["Lokaal"] = array();
                        $sql = "SELECT * FROM Lokaal";
                        $q = $con->query($sql) or die("Execute query error");
                        while($row = $q->fetch(PDO::FETCH_ASSOC)){
                                $Lokaal = array();
                                $Lokaal["LokaalID"] = $row["LokaalID"];
                                $Lokaal["LokaalNaam"] = $row["LokaalNaam"];
                                $Lokaal["LokaalIndelingUrl"] = $row["LokaalIndelingUrl"];
                                array_push($response["Lokaal"], $Lokaal);
                        }
                        $response["success"] = 1;
                        echo json_encode($response);
```

```
                } else {
                        $response["success"] = 0;
                        $response["message"] = "Geen Lokaal gevonden";
                        echo json_encode($response);
                }
        } else {
                $response["success"] = 0;
                $response["message"] = "Geen Lokaal gevonden";
                echo json_encode($response);
        }
?>
```

```
<?php
        // select
        $response = array();
        if (isset($_POST['LokaalID'])) {
                require_once '../includes/db_config.php';
                $host = DB_SERVER;
                $db = DB_DATABASE;
                $user = DB_READER;
                $pass = DB_READER_PASSWORD;
                try {
                        $con = new PDO("mysql:host=$host;dbname=$db",$user,$pass);
                }
                catch(PDOException $err) {
                  die("Connection error");  //, because: " . $err->getMessage());
                }
                $LokaalID = $_POST['LokaalID'];
                $sql = "SELECT COUNT(*) FROM Lokaal WHERE LokaalID = :LokaalID";
                $q = $con->prepare($sql);
                $q->execute(array(':LokaalID'=>$LokaalID));
                if(!$q) {
                        die("Execute query error");    //, because: ". $con->errorInfo());
                }
                if ($q) {
                        if ($q->fetchColumn() > 0) {
                                $sql = "SELECT * FROM Lokaal WHERE LokaalID = :LokaalID";
                                $q = $con->prepare($sql);
                                $q->execute(array(':LokaalID'=>$LokaalID));
                                $result = $q->fetch(PDO::FETCH_ASSOC);
                                $Lokaal = array();
                                $Lokaal["LokaalID"] = $result["LokaalID"];
                                $Lokaal["LokaalNaam"] = $result["LokaalNaam"];
                                $Lokaal["LokaalIndelingUrl"] = $result["LokaalIndelingUrl"];
                                $response["success"] = 1;
                                $response["Lokaal"] = array();
                                array_push($response["Lokaal"], $Lokaal);
                                echo json_encode($response);
                        } else {
                                $response["success"] = 0;
                                $response["message"] = "Geen Lokaal gevonden";
                                echo json_encode($response);
                        }
                } else {
                        $response["success"] = 0;
                        $response["message"] = "Geen Lokaal gevonden";
                        echo json_encode($response);
                }
        } else {
                $response["success"] = 0;
                $response["message"] = "Vereiste velden ontbreken";
                echo json_encode($response);
```

```
        }
?>
```

```
<?php
        // insert
        $response = array();
        if (isset($_POST['LokaalNaam']) && isset($_POST['LokaalIndelingUrl'])) {
                require_once '../includes/db_config.php';
                $host = DB_SERVER;
                $db = DB_DATABASE;
                $user = DB_WRITER;
                $pass = DB_WRITER_PASSWORD;
                try {
                        $con = new PDO("mysql:host=$host;dbname=$db",$user,$pass);
                }
                catch(PDOException $err) {
                  die("Connection error");  //, because: " . $err->getMessage());
                }
                $LokaalNaam = $_POST['LokaalNaam'];
                $LokaalIndelingUrl = $_POST['LokaalIndelingUrl'];
                $sql = "INSERT INTO Lokaal (LokaalNaam,LokaalIndelingUrl) VALUES
(:LokaalNaam,:LokaalIndelingUrl)";
                $q = $con->prepare($sql);
                $q->execute(array(':LokaalNaam'=>$LokaalNaam,
                                                ':LokaalIndelingUrl'=>$LokaalIndelingUrl));
                if ($q->rowCount() > 0) {
                        $response["success"] = 1;
                        $response["message"] = "Record toegevoegd";
                        echo json_encode($response);
                } else {
                        $response["success"] = 0;
                        $response["message"] = "Oops! Een fout.";
                        echo json_encode($response);
                }
        } else {
                $response["success"] = 0;
                $response["message"] = "Vereiste velden ontbreken";
                echo json_encode($response);
        }
?>
```

```
<?php
        // update
        $response = array();
        if (isset($_POST['LokaalID']) && isset($_POST['LokaalNaam']) &&
isset($_POST['LokaalIndelingUrl'])) {
                require_once '../includes/db_config.php';
                $host = DB_SERVER;
                $db = DB_DATABASE;
                $user = DB_WRITER;
                $pass = DB_WRITER_PASSWORD;
                try {
                        $con = new PDO("mysql:host=$host;dbname=$db",$user,$pass);
                }
                catch(PDOException $err) {
                  die("Connection error");  //, because: " . $err->getMessage());
                }
                $LokaalID = $_POST['LokaalID'];
                $LokaalNaam = $_POST['LokaalNaam'];
                $LokaalIndelingUrl = $_POST['LokaalIndelingUrl'];
```

```
                $sql = "UPDATE Lokaal SET LokaalNaam = :LokaalNaam, LokaalIndelingUrl =
:LokaalIndelingUrl WHERE LokaalID = :LokaalID";
                $q = $con->prepare($sql);
                $q->execute(array(':LokaalID'=>$LokaalID,
                                                ':LokaalNaam'=>$LokaalNaam,
                                                ':LokaalIndelingUrl'=>$LokaalIndelingUrl));
                if ($q->rowCount() > 0) {
                        $response["success"] = 1;
                        $response["message"] = "Record gewijzigd.";
                        echo json_encode($response);
                } else {
                        $response["success"] = 0;
                        $response["message"] = "Oops! Een fout.";
                        echo json_encode($response);
                }
        } else {
                $response["success"] = 0;
                $response["message"] = "Vereiste velden ontbreken";
                echo json_encode($response);
        }
?>
```

```
<?php
        // delete
        $response = array();
        if (isset($_POST['LokaalID'])) {
                require_once '../includes/db_config.php';
                $host = DB_SERVER;
                $db = DB_DATABASE;
                $user = DB_WRITER;
                $pass = DB_WRITER_PASSWORD;
                try {
                        $con = new PDO("mysql:host=$host;dbname=$db",$user,$pass);
                }
                catch(PDOException $err) {
                  die("Connection error");  //, because: " . $err->getMessage());
                }
                $LokaalID = $_POST['LokaalID'];
                $sql = "DELETE from Lokaal WHERE LokaalID = :LokaalID";
                $q = $con->prepare($sql);
                $q->execute(array(':LokaalID'=>$LokaalID));
                if ($q->rowCount() > 0) {
                        $response["success"] = 1;
                        $response["message"] = "Record verwijderd";
                        echo json_encode($response);
                } else {
                        $response["success"] = 0;
                        $response["message"] = "Oops! Een fout.";
                        echo json_encode($response);
                }
        } else {
                $response["success"] = 0;
                $response["message"] = "Vereiste velden ontbreken";
                echo json_encode($response);
        }
?>
```