MODULE 4

Nội dung thực hành
- *Hiểu được cơ chế làm việc của Content Provider* - *Sử dụng được các Content Provider được cung cấp sẵn* - *Tạo được Content Provider do người dùng định nghĩa*

Bài tập 1

Mục đích:

- Sử dụng được content provider ContactsContract được cung cấp sẵn.
- Sử dụng cursor
- Các kỹ năng chuyên sâu khác

Yêu cầu:

Viết ứng dụng sổ địa chỉ quản lý các contacts trên thiết bị có thể đồng bộ với sổ địa chỉ của thiết bị.

Hướng dẫn:

Sau đây là các phương thức tương ứng với các hoạt động CRUD của ContactsContract. Sinh viên customize lại cho tương ứng với yêu cầu bài tập

Chèn một contact vào Contact Manager

```
public static boolean createContact(Context ctx, String name, String phone) throws
Exception{
	ContentResolver cr = ctx.getContentResolver();
	Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
				null, null, null, null);
	if (cur.getCount() > 0) {
		while (cur.moveToNext()) {
				String existName =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
				if (existName.contains(name)) {
					Log.i(TAG,"The contact name: " + name + " already exists");
					return false;
				}
		}
	}
	ArrayList<ContentProviderOperation> ops =new
ArrayList<ContentProviderOperation>();
	ops.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_
URI)
	.withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, "accountname@gmail.com")
	.withValue(ContactsContract.RawContacts.ACCOUNT_NAME, "com.google")
	.build());
```

```
	ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)

	.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
				.withValue(ContactsContract.Data.MIMETYPE,

	ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)

	.withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME, name)
				.build());

	ops.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)

	.withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
			.withValue(ContactsContract.Data.MIMETYPE,

	ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
			.withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, phone)
			.withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_HOME)
			.build());
		cr.applyBatch(ContactsContract.AUTHORITY, ops);
	Log.i(TAG, "Created a new contact with name: " + name + " and Phone No: " +
phone);
	return true;
}
```

Kiểm tra một contact có trong Contacts Manager không

```
public static boolean isTheNumberExistsinContacts(Context ctx, String phoneNumber) {
	Cursor cur = null;
	ContentResolver cr = null;
	cr = ctx.getContentResolver();
	cur = cr.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
	if (cur.getCount() > 0) {
		while (cur.moveToNext()) {
			String id =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts._ID));
			String name =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
			Log.i("Names", name);
			if (Integer.parseInt( cur.getString(
cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER))) > 0) {
				// Query phone here. Covered next
				Cursor phones = ctx.getContentResolver().query(
ContactsContract.CommonDataKinds.Phone.CONTENT_URI,null,
ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + id, null, null);
				while (phones.moveToNext()) {
					String phoneNumberX = phones.getString(
phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));
					phoneNumberX = phoneNumberX.replace(" ", "");
					phoneNumberX = phoneNumberX.replace("(", "");
					phoneNumberX = phoneNumberX.replace(")", "");
					if (phoneNumberX.contains(phoneNumber)) {
						phones.close();
						return true;
					}
				}
```

```
                        phones.close();
                    }
                }
            }
            return false;
}
```

Xóa một contact vào Contact Manager

```
public static boolean deleteContactbyPhoneNumber(Context ctx, String phoneNumber)
throws Exception{
      Uri contactUri = Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
Uri.encode(phoneNumber));
      Cursor cur = ctx.getContentResolver().query(contactUri, null, null, null, null);
      if (cur.moveToFirst()) {
            do {
                  String lookupKey = cur.getString( cur.getColumnIndex(
                        ContactsContract.Contacts.LOOKUP_KEY));
                  Uri uri = Uri.withAppendedPath(
                        ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
                  ctx.getContentResolver().delete(uri, null, null);
            } while (cur.moveToNext());
      }
      return false;
}
```

```
public static void deleteContactByName(Context ctx, String name) throws Exception{
      ContentResolver cr = ctx.getContentResolver();
      String where = ContactsContract.Data.DISPLAY_NAME + " = ? ";
      String[] params = new String[] {name};

      ArrayList<ContentProviderOperation> ops = new
ArrayList<ContentProviderOperation>();
      ops.add(ContentProviderOperation.newDelete(ContactsContract.RawContacts.CONTENT_
URI)
            .withSelection(where, params)
            .build());
      cr.applyBatch(ContactsContract.AUTHORITY, ops);
      Log.i(TAG,"Deleted the contact with name '" + name +"'");
}
```

Liệt kê danh sách các contacts có trong Contacts Manager
(lưu ý chỉnh sửa lại tương ứng với yêu cầu của bạn)

```
public static void displayContacts(Context ctx) {
      ContentResolver cr = ctx.getContentResolver();
      Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
                  null, null, null, null);
      if (cur.getCount() > 0) {
            String numbers="";
            while (cur.moveToNext()) {
                  String id =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts._ID));
                  String name =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
                  if (Integer.parseInt(cur.getString(
      cur.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER))) > 0) {
                        Cursor pCur = cr.query(
      ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
```

```
        ContactsContract.CommonDataKinds.Phone.CONTACT_ID +" = ?",
                                                new String[]{id}, null);
                        while (pCur.moveToNext()) {
                                String phoneNo = pCur.getString(pCur.getColumnIndex(
ContactsContract.CommonDataKinds.Phone.NUMBER));
                                numbers+=phoneNo+";";
                        }
                        pCur.close();
                }
                Log.i(TAG, "Name: " + name + ", Phone No: " + numbers);
        }
    }
}
```

Cập nhật một contact trong Contacts Manager

```
public static void updateContact(Context ctx, String name, String phone) throws
Exception{
    ContentResolver cr = ctx.getContentResolver();
    String where = ContactsContract.Data.DISPLAY_NAME + " = ? AND " +
        ContactsContract.Data.MIMETYPE + " = ? AND " +
        String.valueOf(ContactsContract.CommonDataKinds.Phone.TYPE) + " = ? ";
        String[] params = new String[] {name,
            ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE,
            String.valueOf(ContactsContract.CommonDataKinds.Phone.TYPE_HOME)};

        Cursor phoneCur = new CursorLoader(ctx, ContactsContract.Data.CONTENT_URI,
                null, where, params, null).loadInBackground();

        ArrayList<ContentProviderOperation> ops = new
ArrayList<ContentProviderOperation>();
        if ( (null == phoneCur)  ) {
            createContact(ctx, name, phone);
        } else {

    ops.add(ContentProviderOperation.newUpdate(ContactsContract.Data.CONTENT_URI)
                        .withSelection(where, params)
        .withValue(ContactsContract.CommonDataKinds.Phone.DATA, phone)
                        .build());
        }
        phoneCur.close();
        cr.applyBatch(ContactsContract.AUTHORITY, ops);
        Log.i(TAG, "Updated the phone number of 'Sample Name' to: " + phone);
    }
```

Bài tập 2

Mục đích

- Sử dụng được content provider ContactsContract được cung cấp sẵn.
- Sử dụng cursor
- Các kỹ năng chuyên sâu khác

Yêu cầu

- Viết ứng dụng quản lý calendar và các event trên thiết bị

Hướng dẫn

Lớp AlternativeCalendar này chứa các tác vụ cơ bản cho việc thao tác với calendar như:

- Lấy danh sách các calendar
- Lấy danh sách các event
- Các thao tác CRUD trên các events
- Và các thao tác khác

Bạn cần phải hiệu chỉnh lại theo mục đích của bạn với từng chức năng và yêu cầu khác nhau


```
package vovanhai.wordpress.com.calendardemo;

import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.TimeZone;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.CalendarContract.Attendees;
import android.provider.CalendarContract.Calendars;
import android.provider.CalendarContract.Events;
import android.provider.CalendarContract.Reminders;
import android.support.v4.content.CursorLoader;
import android.util.Log;

public class AlternativeCalendar {
	public void getAllCalendars(Context ctx) {
		String[] projection = new String[] {
					Calendars._ID,
					Calendars.NAME,
					Calendars.ACCOUNT_NAME,
					Calendars.ACCOUNT_TYPE };
		Cursor calCursor =
ctx.getContentResolver().query(Calendars.CONTENT_URI,
					projection, null,//Calendars.VISIBLE + " = 1",
					null, Calendars._ID + " ASC");
```

```
            if (calCursor.moveToFirst()) {
                  do {
                        long id = calCursor.getLong(0);
                        String displayName = calCursor.getString(1);
                        // …
                        Log.i("----------", id+" - "+displayName);
                  } while (calCursor.moveToNext());
            }
      }

      public void getAllEvents(Context ctx) {
            Uri uri = Uri.parse("content://com.android.calendar/events");
            String[] projection = new String[] { "title", "dtstart", "dtend" };
            Cursor cursor = new CursorLoader(ctx, uri, projection, null, null,
"dtstart DESC, dtend DESC")
                        .loadInBackground();
            if (cursor.moveToFirst()) {
                  StringBuilder displayText = new StringBuilder();
                  int l_colTitle = cursor.getColumnIndex(projection[0]);
                  int l_colBegin = cursor.getColumnIndex(projection[1]);
                  int l_colEnd = cursor.getColumnIndex(projection[1]);
                  do {
                        String l_title = cursor.getString(l_colTitle);
                        String l_begin = new
Date(cursor.getLong(l_colBegin)).toString();
                        String l_end = new
Date(cursor.getLong(l_colEnd)).toString();
                        displayText.append(l_title + "\n" + l_begin + "\n" +
l_end + "\n----------------\n");
                  } while (cursor.moveToNext() );
                  Log.i("-----", displayText.toString());
            }

      }

      /*
      public Uri addEvent(Context ctx, String m_selectedCalendarId) {
            ContentValues event = new ContentValues();
            event.put("calendar_id", m_selectedCalendarId);
            event.put("title", "calendar tutorial test");
            event.put("description", "This is a simple test for calendar api");
            event.put("eventLocation", "@home");
            event.put("dtstart", System.currentTimeMillis());
            event.put("dtend", System.currentTimeMillis() + 1800 * 1000);
            event.put("allDay", 0);
            // status: 0~ tentative; 1~ confirmed; 2~ canceled
            event.put("eventStatus", 1);
            // 0~ default; 1~ confidential; 2~ private; 3~ public
            event.put("visibility", 0);
            // 0~ opaque, no timing conflict is allowed; 1~ transparency, allow
            // overlap of scheduling
            event.put("transparency", 0);
            // 0~ false; 1~ true
            event.put("hasAlarm", 1);
            Uri uri = Uri.parse("content://com.android.calendar/events");
            Uri l_uri = ctx.getContentResolver().insert(uri, event);
```

```
            return l_uri;
      }*/

      public long createAllDayEvent(Context ctx, long calendarId, String title,
String description) {
            if (calendarId == -1) {
                  Log.i("----------------------", "no calendar account");
                  return -1;
            }
            Calendar cal = new GregorianCalendar(2016, 03, 05);
            cal.setTimeZone(TimeZone.getTimeZone("UTC"));
            cal.set(Calendar.HOUR, 0);
            cal.set(Calendar.MINUTE, 0);
            cal.set(Calendar.SECOND, 0);
            cal.set(Calendar.MILLISECOND, 0);
            long start = cal.getTimeInMillis();
            ContentValues values = new ContentValues();
            values.put(Events.DTSTART, start);
            values.put(Events.DTEND, start);
            values.put(Events.RRULE,
"FREQ=DAILY;COUNT=20;BYDAY=MO,TU,WE,TH,FR;WKST=MO");
            values.put(Events.TITLE, title);
            values.put(Events.EVENT_LOCATION, "HoChiMinh City");
            values.put(Events.CALENDAR_ID, calendarId);
            values.put(Events.EVENT_TIMEZONE, "Indochina Time");
            values.put(Events.DESCRIPTION, description);
            // reasonable defaults exist:
            values.put(Events.ACCESS_LEVEL, Events.ACCESS_PRIVATE);
            values.put(Events.SELF_ATTENDEE_STATUS, Events.STATUS_CONFIRMED);
            values.put(Events.ALL_DAY, 1);
            values.put(Events.ORGANIZER, "vovanhaiqn@gmail.com");
            values.put(Events.GUESTS_CAN_INVITE_OTHERS, 1);
            values.put(Events.GUESTS_CAN_MODIFY, 1);
            values.put(Events.AVAILABILITY, Events.AVAILABILITY_BUSY);
            Uri uri = ctx.getContentResolver().insert(Events.CONTENT_URI, values);
            long eventId = Long.valueOf(uri.getLastPathSegment());
            return eventId;
      }

      public void addAttendee(Context ctx, String eventId, String name, String
email) {
            // adding an attendee:
            ContentValues values = new ContentValues();
            values.put(Attendees.EVENT_ID, eventId);
            values.put(Attendees.ATTENDEE_TYPE, Attendees.TYPE_REQUIRED);
            values.put(Attendees.ATTENDEE_NAME, name);
            values.put(Attendees.ATTENDEE_EMAIL, email);
            ctx.getContentResolver().insert(Attendees.CONTENT_URI, values);
      }

      public void addReminder(Context ctx, long eventId) {
            // adding a reminder:
            ContentValues values = new ContentValues();
            values.put(Reminders.EVENT_ID, eventId);
            values.put(Reminders.METHOD, Reminders.METHOD_ALERT);
            values.put(Reminders.MINUTES, 30);
```

```
            ctx.getContentResolver().insert(Reminders.CONTENT_URI, values);
      }

      public boolean deleteEvent(Context ctx, long eventId) {
            String[] selArgs = new String[] { eventId + "" };
            int deleted = ctx.getContentResolver().delete(Events.CONTENT_URI,
                        Events._ID + " =? ", selArgs);
            return deleted > 0;
      }

      public boolean updateEvent(Context ctx, long eventId, String title, String
location) {
            ContentValues values = new ContentValues();
            values.put(Events.TITLE, title);
            values.put(Events.EVENT_LOCATION, location);
            String[] selArgs = new String[] { eventId+"" };
            int updated = ctx.getContentResolver().update(Events.CONTENT_URI,
values,
                        Events._ID + " =? ", selArgs);
            return updated > 0;
      }
}
```

Bài tập 3

Mục đích

- Tạo một content provider cho việc cung cấp một danh sách các tài khoản cho các ứng dụng dùng chung.
- Ôn lại thao tác với cơ sở dữ liệu

Yêu cầu

1. Một ứng dụng có tên **AccountProvider** cung cấp danh mục các tài khoản dùng chung. Ứng dụng này dùng một Sqllite database có tên **accountsdb** với một bảng có tên **account**(**accountID** varchar(30), **credential** varchar(30),**role** varchar(20)).
   URI của content provider này là: **content://fit.iuh.edu.vn.AccountProvider/accounts**

2. Một ứng dụng có tên SampleCpConsumer sẽ truy cập dữ liệu từ content provider cung cấp bởi ứng dụng AccountProvider. Thực hiện tất cả các thao tác CRUD từ content provider trên

Hướng dẫn

1. Ứng dụng AccountProvider
   Xây dựng lớp database helper để thao tác với cơ sở dữ liệu

```
package fit.iuh.edu.vn;
import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper{
	static final String DATABASE_NAME = "accountsdb";
	static final String TABLE_NAME = "account";
	static final int DATABASE_VERSION = 1;
	static final String CREATE_DB_TABLE = "CREATE TABLE " + TABLE_NAME
			+ " (accountid VARCHAR(30) PRIMARY KEY, "
			+ " credential VARCHAR(30) NOT NULL, "
			+ " role VARCHAR(20) NOT NULL default guest)";


	public DatabaseHelper(Context context) {
		super(context, DATABASE_NAME, null, DATABASE_VERSION);
	}

	@Override
	public void onCreate(SQLiteDatabase db) {
		db.execSQL(CREATE_DB_TABLE);
		ContentValues values=new ContentValues();
		values.put("accountID", "VovanHai");
		values.put("credential", "123");
		values.put("role", "admin");
		db.insert(TABLE_NAME, null, values);
```

```
	}

	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
		onCreate(db);
	}
}
```

Tạo lớp `AccountProvider` để cung cấp nội dung sẽ cung cấp

```
package fit.iuh.edu.vn;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.provider.Settings.System;
import android.util.Log;

public class AccountProvider extends ContentProvider {
	static final String TAG = "AccountProvider";

	static final String AUTHORITY =
"content://fit.iuh.edu.vn.AccountProvider/accounts";
	static final Uri CONTENT_URI = Uri.parse(AUTHORITY);

	static final String ACCOUNT_ID = "accountID";
	static final String CREDENTIAL = "credential";
	static final String ROLE = "role";
	static final String TABLE_NAME = "account";
	private SQLiteDatabase db;

	@Override
	public boolean onCreate() {
		Context context = getContext();
		DatabaseHelper dbHelper = new DatabaseHelper(context);
		db = dbHelper.getWritableDatabase();
		if (db != null) return true;
		return false;
	}

	@Override
	public String getType(Uri uri) {
		String ret =
getContext().getContentResolver().getType(System.CONTENT_URI);
		Log.d(TAG, "getType returning: " + ret);
		return ret;
	}

	@Override
	public Uri insert(Uri uri, ContentValues values) {
		long rowID = db.insert(TABLE_NAME, "", values);
		if (rowID > 0) {
```

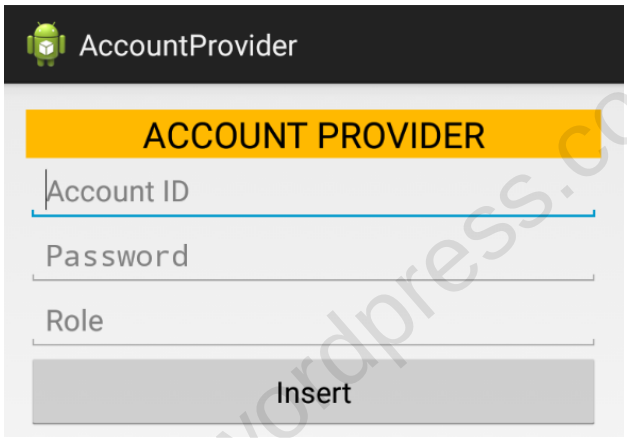
```
                    Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
                    getContext().getContentResolver().notifyChange(_uri, null);
                    return _uri;
             }
             throw new SQLException("Failed to add a record into " + uri);
      }

      @Override
      public int update(Uri uri, ContentValues values, String selection,
String[] selectionArgs) {
             int count = 0;
             count = db.update(TABLE_NAME, values, selection, selectionArgs);
             getContext().getContentResolver().notifyChange(uri, null);
             return count;
      }

      @Override
      public int delete(Uri uri, String selection, String[] selectionArgs) {
             int count = 0;
             count = db.delete(TABLE_NAME, selection, selectionArgs);
             getContext().getContentResolver().notifyChange(uri, null);
             return count;
      }

      @Override
      public Cursor query(Uri uri, String[] projection, String
selection,String[] selectionArgs, String sortOrder) {
             Cursor c = getContext().getContentResolver()
                          .query(uri, projection, selection, selectionArgs,
sortOrder);
             c.setNotificationUri(getContext().getContentResolver(), uri);
             return c;
      }
}
```

Đăng ký provider trong manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="iuh.edu.vn.accountprovider"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="23" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="fit.iuh.edu.vn.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
        <provider xmlns:tools="http://schemas.android.com/tools"
            android:name="fit.iuh.edu.vn.AccountProvider"
            android:authorities="fit.iuh.edu.vn.AccountProvider"
            android:exported="true"
            android:multiprocess="true"
            tools:ignore="ExportedContentProvider">
        </provider>
    </application>
</manifest>
```

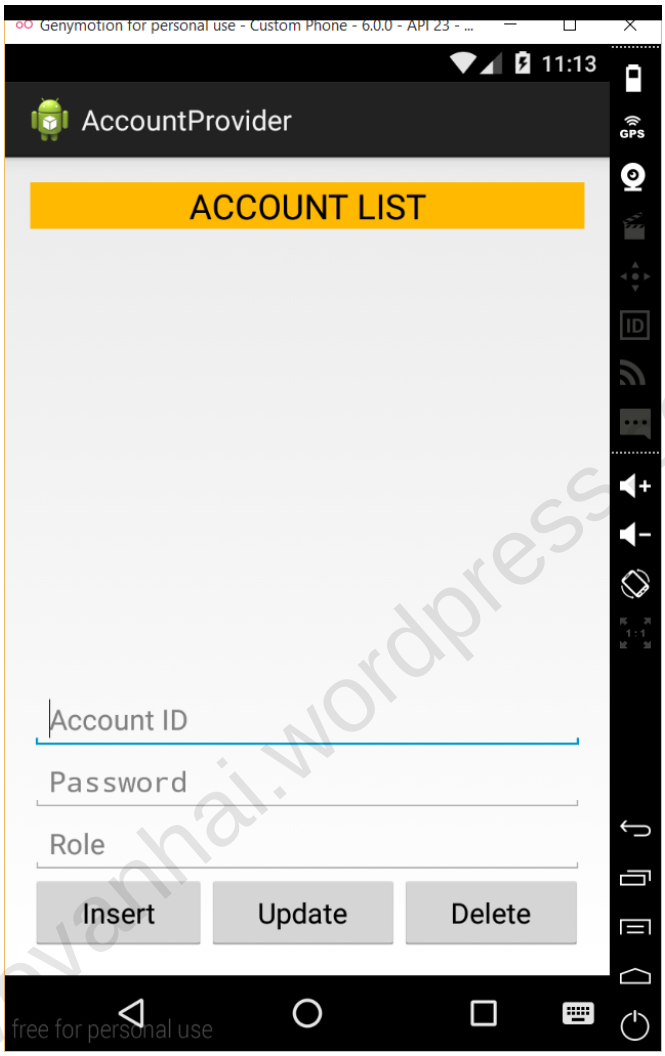
MainActivity dùng để kiểm tra việc thêm một tài khoản vào content provider




```
package fit.iuh.edu.vn;
import android.app.Activity;
import android.content.ContentValues;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import iuh.edu.vn.accountprovider.R;

public class MainActivity extends Activity {
	private EditText edtID,edtPsw,edtRole;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		edtID=(EditText) findViewById(R.id.editText1);
		edtPsw=(EditText) findViewById(R.id.editText2);
		edtRole=(EditText) findViewById(R.id.editText3);
	}
	public void insert(View v){
	Uri uri=Uri.parse("content://fit.iuh.edu.vn.AccountProvider/accounts");
		ContentValues values=new ContentValues();
		values.put("accountid", edtID.getText().toString());
		values.put("credential", edtPsw.getText().toString());
		values.put("role", edtRole.getText().toString());
		getContentResolver().insert(uri, values);
	}
}
```

Triển khai ứng dụng

2. Viết ứng dụng client như mô tả của giao diện bên dưới



Khi ứng dụng khởi động sẽ nạp toàn bộ account lên danh sách.

Khi chọn một account, thông tin chi tiết sẽ hiển thị lên các edit text

Viết các chức năng thêm, cập nhật, xóa các account khỏi content provider