Lab8 实验报告

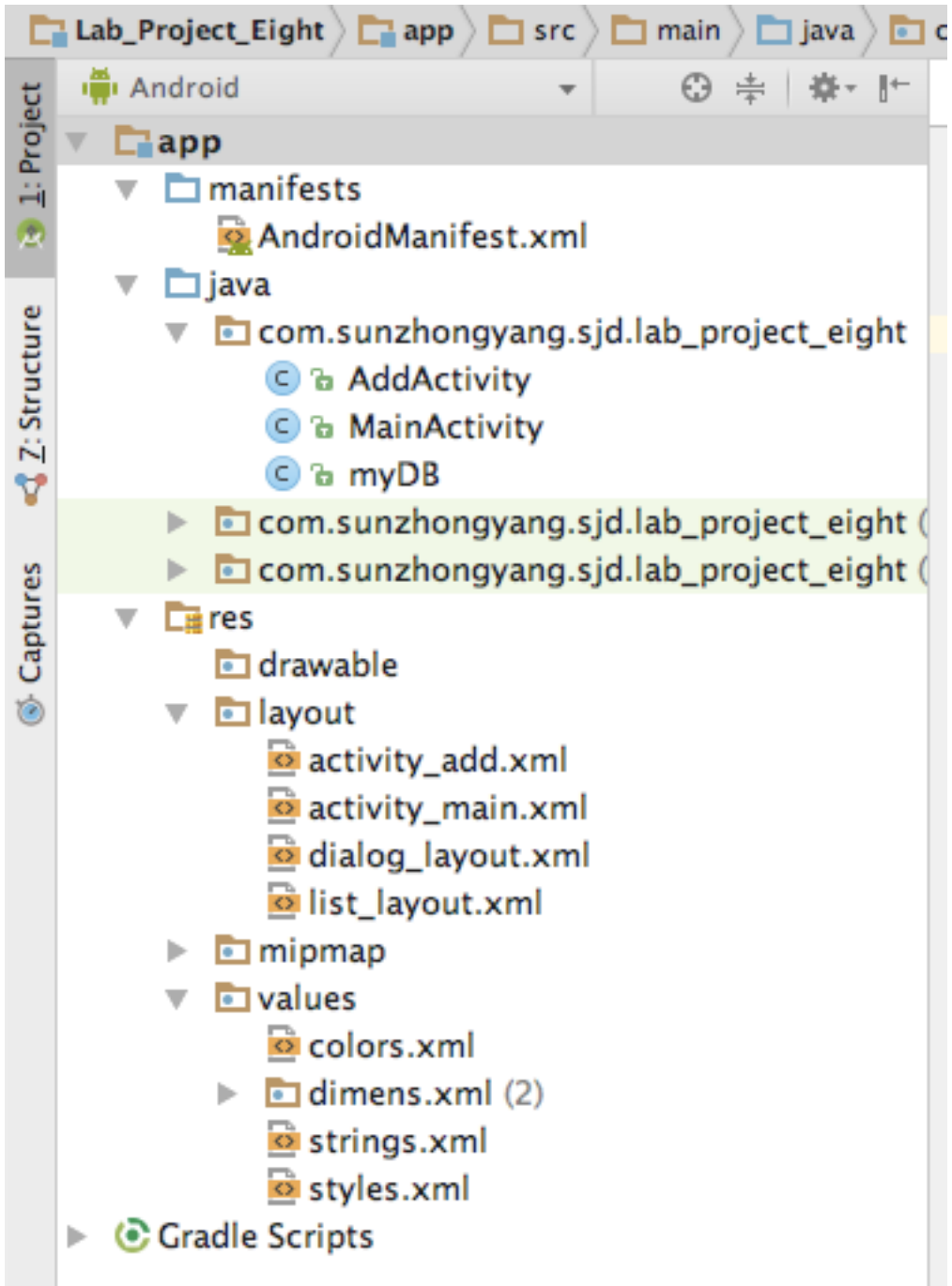
13331233 孙中阳

参考资料

《第一行代码 Android》 郭霖 著
《Lab8实验文档》

实验步骤

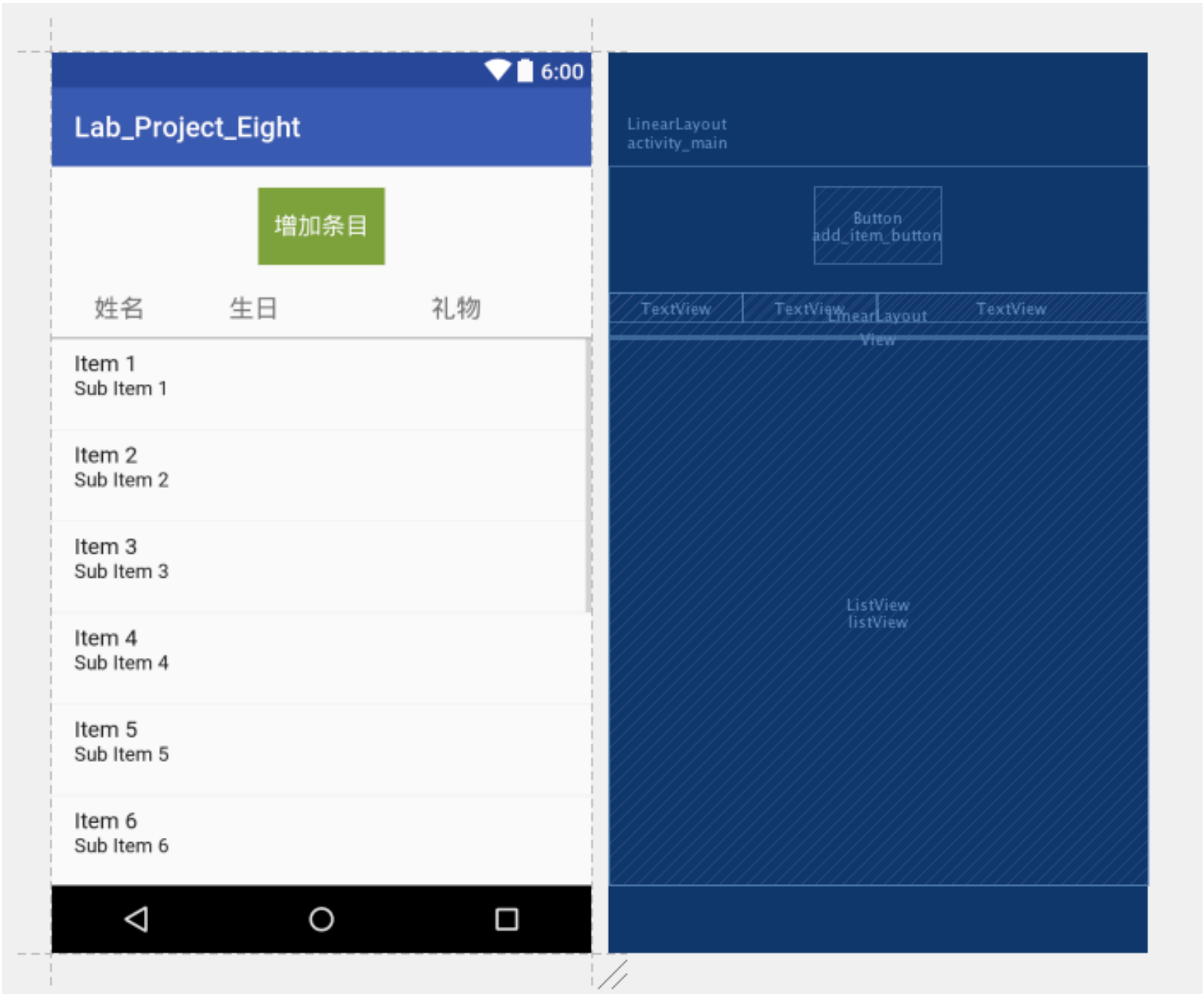
本次试验较上一次实验难度有所提升，代码量有所加大。主要需要熟悉 `SQLite` 及 `ContentProvider` 的使用并使用新的方法控制对话框UI等
图为项目结构



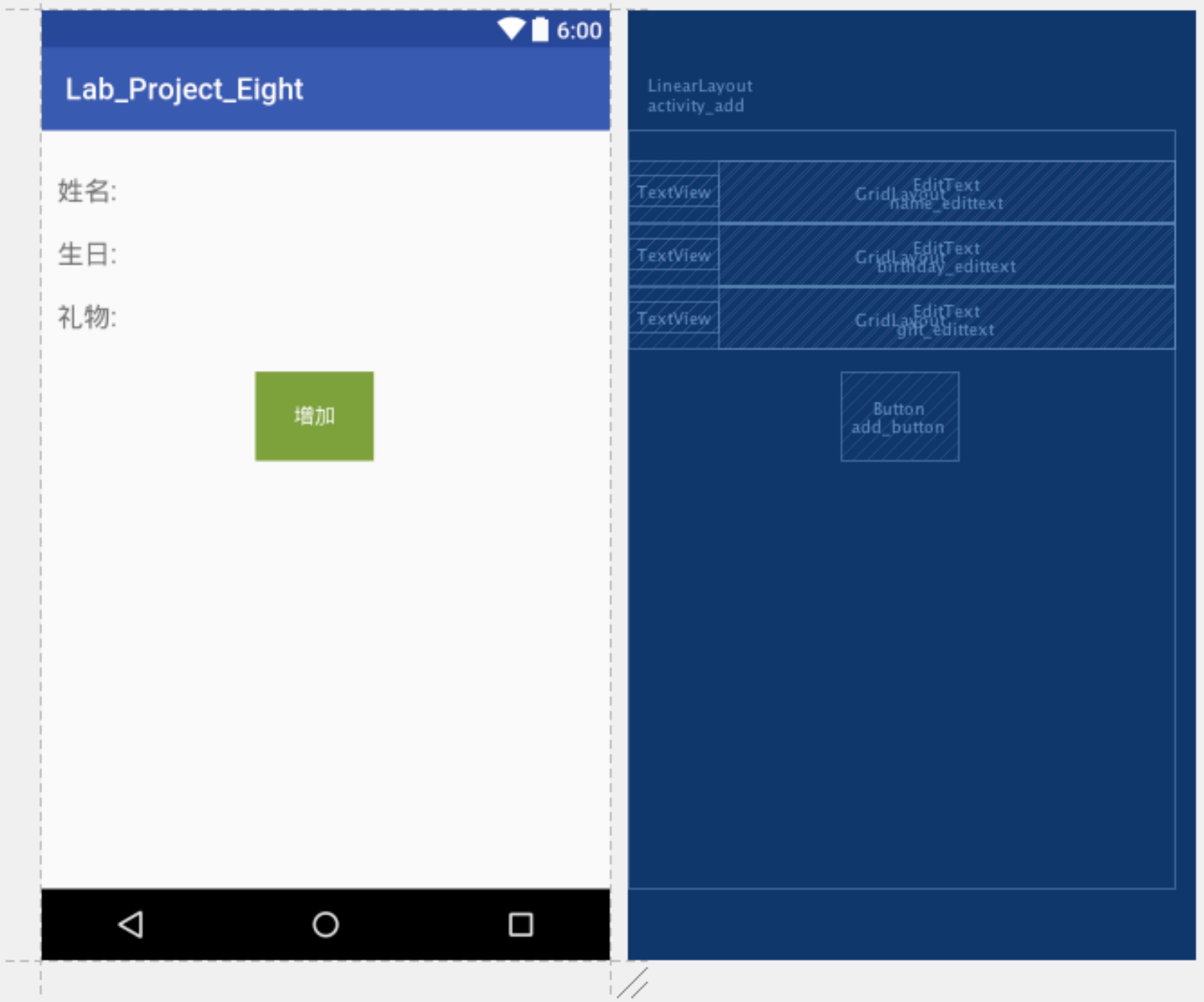
从实验文档看，需要实现的目标主要有

1、使用 `SQLite` 保存姓名生日等有关信息
2、使用 `ContentProvider` 根据姓名获取联系人电话信息 3、使用 `LayoutInflater` 类实现对话框布局的设置

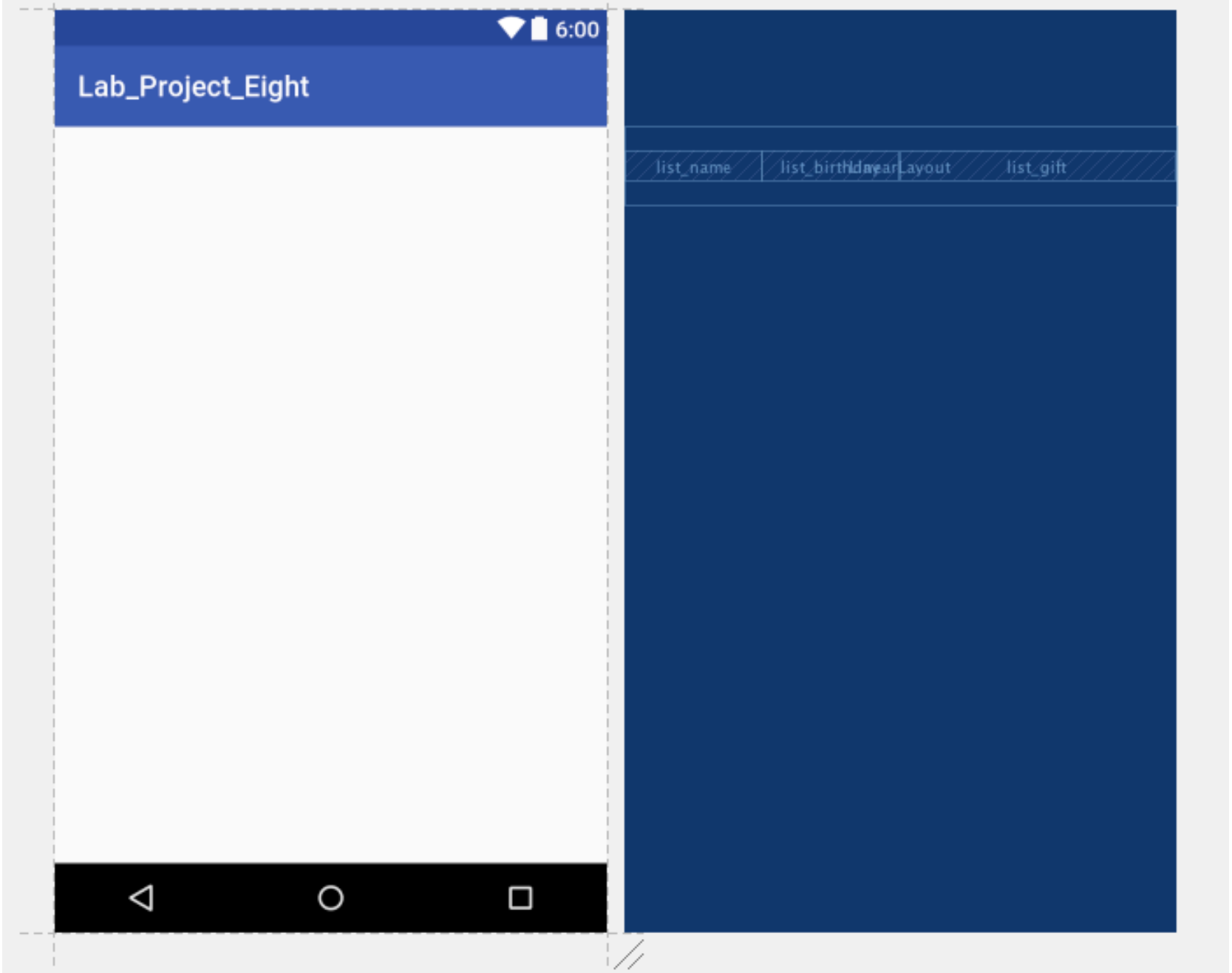
图为主界面布局



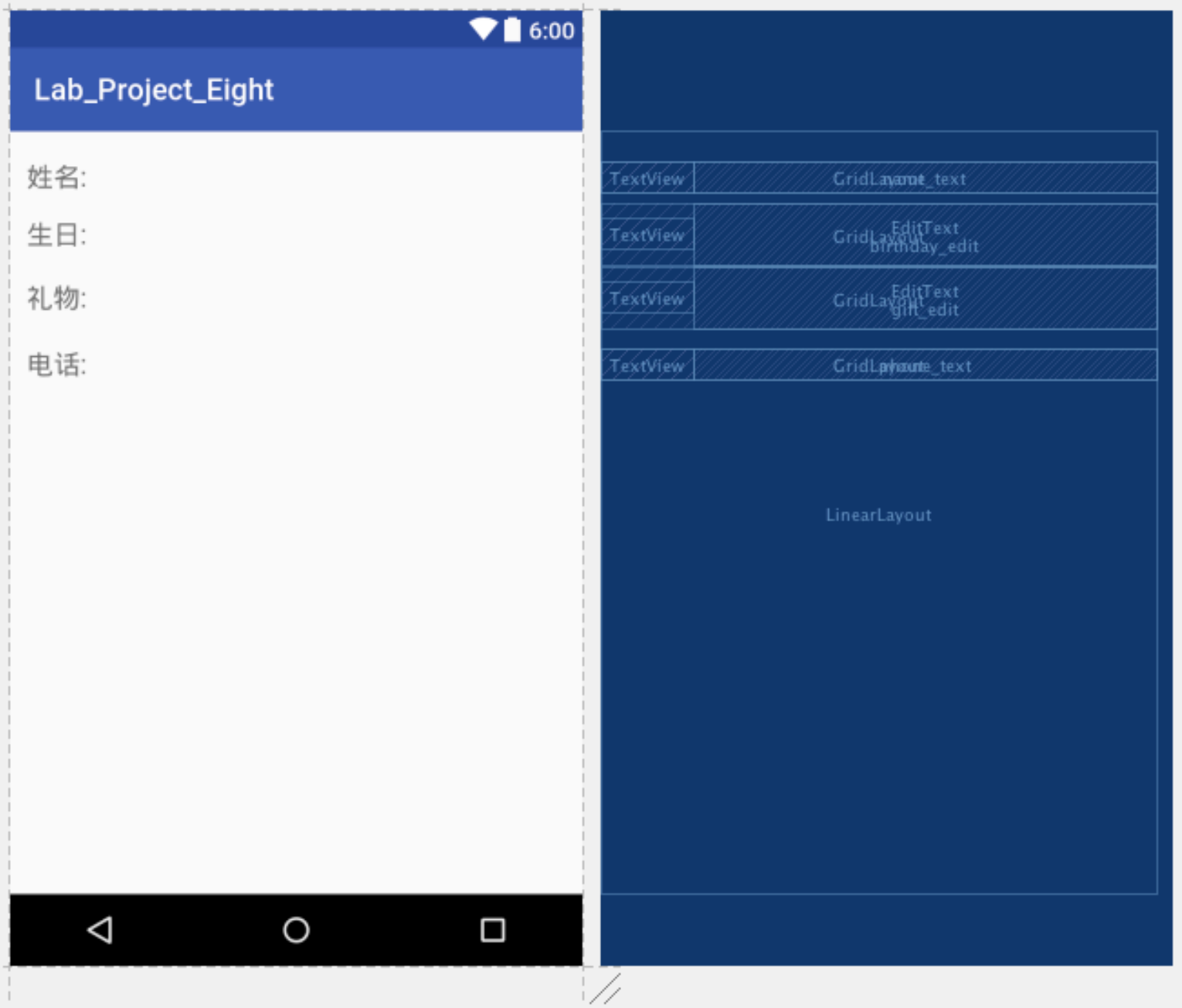
图为增添信息页面布局



图为每个 `ListView` 的布局


6:00
Lab_Project_Eight
list_name
list_birthday
linear_layout
list_gift

图为对话框的布局



接下来考虑实现
本次开发主要涉及三个方面：

1、通过 `SQLite` 保存姓名生日等有关信息

`SQLite` 是一款非常轻型的数据库，适用范围非常广泛，之前在UWP的开发中曾经使用过，基本的SQL语句和操作方法都比较熟悉，相比之下Android端的操作较UWP还要更简便些

首先需要为使用的数据库编写一个工厂类，实现初始化并返回一个可写数据库的功能

```
public class myDB extends SQLiteOpenHelper
{
    //设置数据库的基本信息
    private static final String DB_NAME = "database";
    private static final String TABLE_NAME = "birthday";
    private static final int DB_VERSION = 1;

    //默认初始化
    public myDB(Context context)
    {
        super(context, DB_NAME, null, DB_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase)
    {
        //创建数据库
        String CREATE_TABLE = "CREATE TABLE if not exists " + TABLE_NAME + "
 (_id INTEGER PRIMARY KEY,name TEXT,birth TEXT,gift TEXT)";
        sqLiteDatabase.execSQL(CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int il)
    {

    }

    public SQLiteDatabase  get_db()
    {
        //返回一个可写的数据库
        SQLiteDatabase db = getWritableDatabase();
        return db;
    }
}
```

数据库的基本信息参照文档有关参数，文档里没有提到的是需要为这个类添加一个默认构造函数，经查询得以解决。 `onCreate()` 部分执行SQL语句在没有建立表的情况下建立一个表，最后再实现一个返回可写数据库的函数供各个活动调用即可

在 `MainActivity` 中，首先需要获取数据库

```
//获得一个数据库
myDB helper = new myDB(this);
final SQLiteDatabase current_db = helper.get_db();
```

`MainActivity` 初始化过程中需要使用到数据库信息的只有 `ListView` 初始化这一部分，所以我们首先需要执行语句获得一个有表内所有信息的游标

```
//根据数据库获取一个有数据库全部内容的游标
Cursor full_cursor = current_db.rawQuery("select * from birthday", null);
```

然后用游标中的信息填充用于初始化 `ListView` 的几个数组

```
//将游标中数据库内的数据添加到数组中
while (full_cursor.moveToNext())
{
    names.add(full_cursor.getString(1));
    birthdays.add(full_cursor.getString(2));
    gifts.add(full_cursor.getString(3));
}
```

最后用数组初始化 `ListView` 即可

`MainActivity` 中另一个需要用到数据库的部分是短按 `ListView` 条目弹出对话框对有关信息的更改，这里主要需要用 `ContentValues` 实现数据库中数据的更新

```
//更新数据库
ContentValues cv = new ContentValues();
cv.put("birth", birthday_information);
cv.put("gift", gift_information);

String tmp = data.get(count).get("names").toString();
String[] args = {tmp};

current_db.update("birthday", cv, "name=?", args);
```

还有长按删除，这里用了一个语句

```
current_db.execSQL("delete from birthday where name='" + data.get(count).get
("names").toString() + "'");
```

删除和更新后都要改变 `ListView` ，否则界面不会自动更改

另外一个需要使用数据库的活动是 `addActivity` ，这个活动需要向数据库里增添条目，同时保证不会增添名字重复的条目进去

检测是否重复:

```
//如果名字重复
while (full_cursor.moveToNext())
{
    if(name_content.equals(full_cursor.getString(1)))
    {
        Toast.makeText(AddActivity.this, "名字重复啦,请核查", Toast.LENGTH_SHOR
T).show();
        return;
    }
}
```

游标和数据库之前已经获取好了

增添同样是使用 `ContentValues()`

```
//通过ContentValues在数据库中增加新内容
ContentValues cv = new ContentValues();
cv.put("name", name_content);
cv.put("birth", birthday_content);
cv.put("gift", gift_content);
current_db.insert("birthday", null, cv);
```

另外 `ListView` 的构造和初始化，以及界面UI的设计等之前的实验已有描述，本次不多赘述

2、通过 `ContentProvider` 根据姓名获取联系人电话信息

这部分本来是想通过文档中的方法得到，但是略有问题，于是上网查询后改了改

```
//获取某个联系人的电话号码
public String getPhoneNumber(String name)
{
    //默认为"无"
    String result = "无";
    ContentResolver contentResolver = this.getContentResolver();
    Cursor cursor = contentResolver.query(android.provider.ContactsContract.
Contacts.CONTENT_URI,
            null, null, null, null);
    while(cursor.moveToNext())
    {
        if(name.equals(cursor.getString(cursor.getColumnIndex(android.provid
er.ContactsContract.Contacts.DISPLAY_NAME))))
        {
            result = "";
            String id = cursor.getString(cursor.getColumnIndex(android.provi
der.ContactsContract.Contacts._ID));
            Log.i("t", id);
            Cursor c = getContentResolver().query(ContactsContract.CommonDat
aKinds.Phone.CONTENT_URI, null, ContactsContract.CommonDataKinds.Phone.CONTA
CT_ID + " = " + id, null, null);

            //要考虑到有多个电话号码的情况
            while (c.moveToNext())
            {
                result += c.getString(c.getColumnIndex(ContactsContract.Comm
onDataKinds.Phone.NUMBER)) + "   ";
            }
        }
    }

    return result;
}
```

过程比较简单，首先用一个 `contentResolver` 获取包含全部联系人信息的游标，然后逐一核实游标中的名字信息是否和需要查询的联系人相同，如果相同则根据这个联系人的ID获取电话号码。

后来想了想每次获取一个游标不是很好地方法，联系人列表还是比较稳定的，可以在每次活动初始化时获得一个游标，然后之后每次使用这个游标即可，每次获取会对资源有些影响

3、通过 `LayoutInflater` 类实现对话框布局的设置
这部分主要靠文档

```
//为弹出的对话框载入布局
LayoutInflater factory = LayoutInflater.from(MainActivity.this);
View v = factory.inflate(R.layout.dialog_layout, null);

final int count = i;

//获取页面控件也设置数据库及获取输入
```

```
TextView name_text = (TextView) v.findViewById(R.id.name_text);
final EditText birthday_text = (EditText) v.findViewById(R.id.birthday_edit)
;
final EditText gift_text = (EditText) v.findViewById(R.id.gift_edit);
final TextView phone_text = (TextView) v.findViewById(R.id.phone_text);

//显示被点击条目的名字
name_text.setText(data.get(count).get("names").toString());
birthday_text.setText(data.get(count).get("birthdays").toString());
gift_text.setText(data.get(count).get("gifts").toString());

//设置编辑框中默认的初始内容
Editable text = birthday_text.getText();
birthday_text.setSelection(text.length());

Editable text2 = gift_text.getText();
gift_text.setSelection(text2.length());

//获取并设置该联系人的电话
String phone_number = getPhoneNumber(data.get(count).get("names").toString()
);
phone_text.setText(phone_number);

//构造一个有布局的对话框
AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
builder.setView(v);
builder.setTitle("此处应为表情?");
builder.setPositiveButton("是", new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialogInterface, int j)
    {
        //获取有关输入
        String birthday_information = birthday_text.getText().toString();
        String gift_information = gift_text.getText().toString();

        //更新数据库
        ContentValues cv = new ContentValues();
        cv.put("birth", birthday_information);
        cv.put("gift", gift_information);

        String tmp = data.get(count).get("names").toString();
        String[] args = {tmp};

        current_db.update("birthday", cv, "name=?", args);

        //更新ListView
        data.get(count).put("gifts", gift_information);
        data.get(count).put("birthdays", birthday_information);
        sAdapter.notifyDataSetChanged();
    }
```

```
});
//点击取消按钮则提示取消按钮被按下
builder.setNegativeButton("否", new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialogInterface, int j)
    {

    }
});
builder.create();
builder.show();
```

唯一需要注意的是，获取控件的时候需要使用之前构造的 `View` 获取

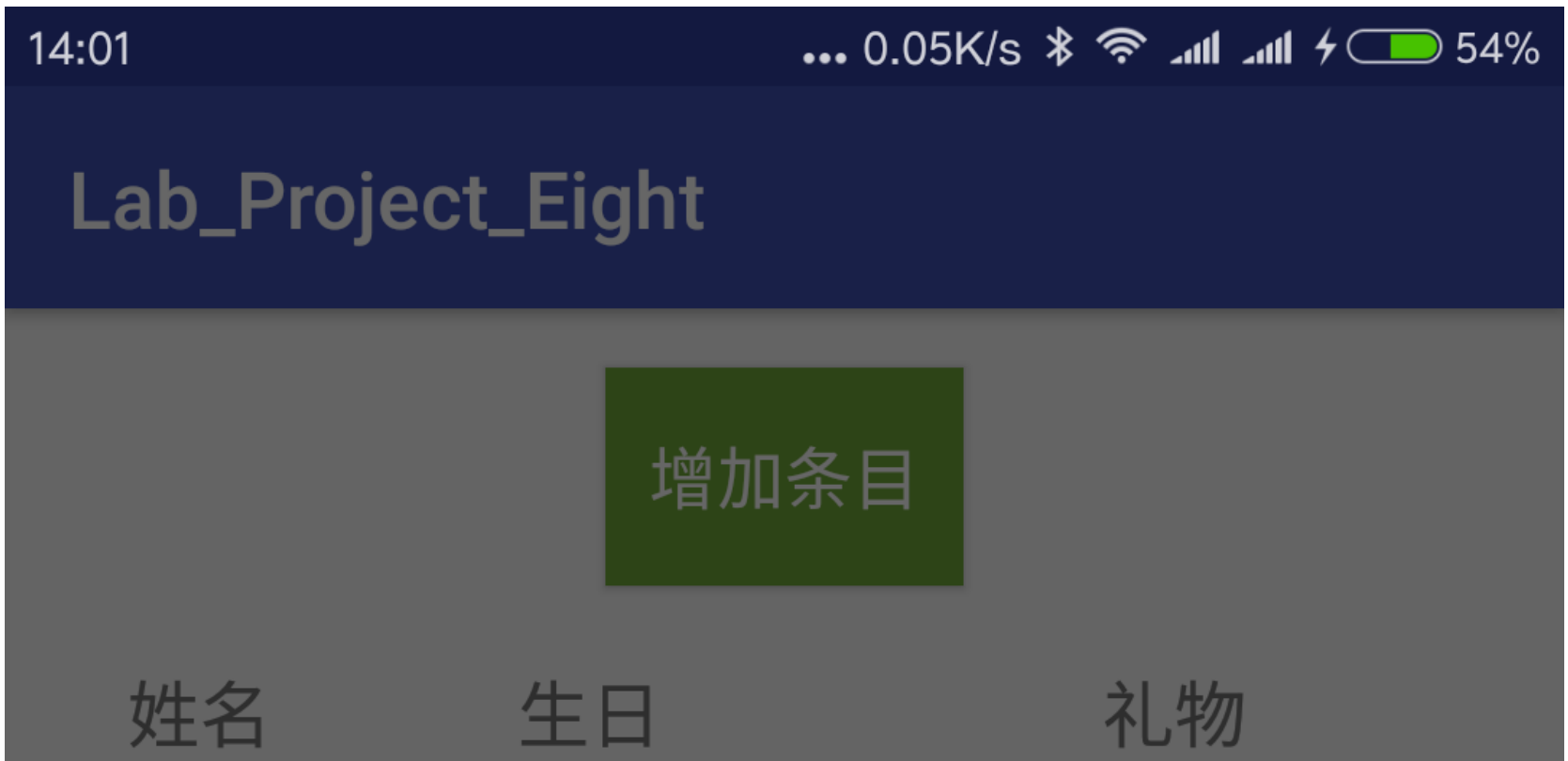
```
TextView name_text = (TextView) v.findViewById(R.id.name_text);
```

而不是直接的 `findViewById()`

实验成果截图

初始界面

短按弹出更改对话框




此处应为表情?
姓名: 高伟
生日: 孙中阳
礼物: 13331233
电话: 1-337-390-0822
否
是

增加条目


15:13
0.05K/s
59%
Lab_Project_Eight

姓名: 高伟

生日: 孙中阳

礼物: 13331233

增加

增加条目后点击返回

通过对话框更改信息

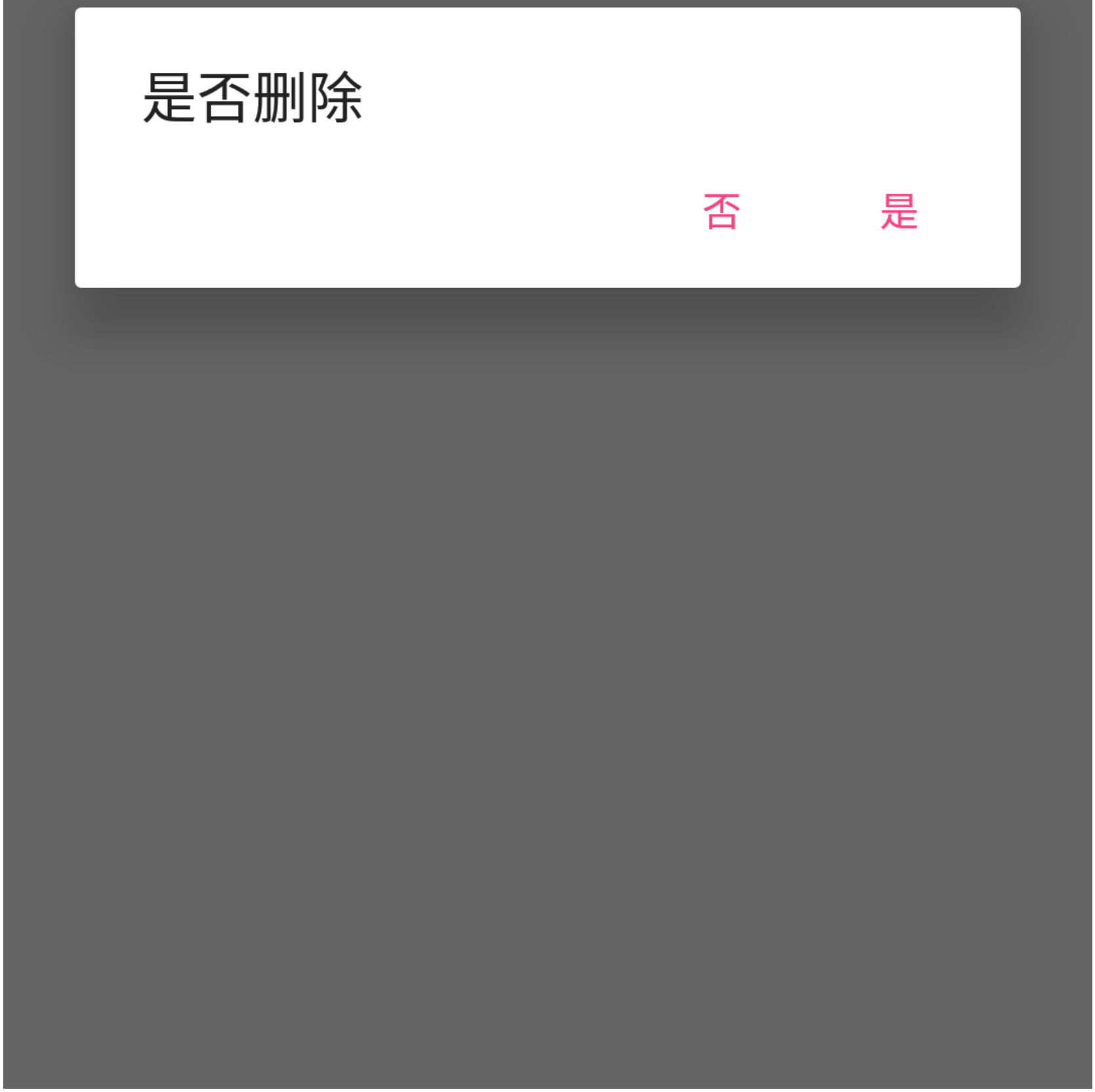

14:01
0.00K/s
54%
Lab_Project_Eight
增加条目
姓名
生日
礼物
此处应为表情?
姓名: 高伟
生日: 孙中阳
礼物:
电话: 1-337-390-0822

否 是

更改后

长按提示是否删除

高伟　　孙中阳



删除后

增加条目

姓名	生日	礼物

实验过程遇到的问题

小问题不断，不过大部分是对具体方法的陌生，实现功能上遇到的问题只有一个，就是如何在 `addActivity` 返回 `MainActivity` 后 `MainActivity` 的 `ListView` 是更新过后的而不是进入 `addActivity` 之前的

最初的思路是点击“增加”按钮后跳转到一个新的 `MainActivity` 页面，后来发现这样做点击手机上的返回键会返回之前的 `addActivity` 甚至能够从 `addActivity` 返回到更早的 `MainActivity` ，也就是说活动栈最下面的活动并没有消失

于是尝试更改 `AndroidManifest.xml` 将 `MainActivity` 改为单例，无奈的是虽然活动栈下面的 `MianActivity` 被清除了，无法通过返回键再次被唤醒，但新打开的 `MainAcitity` 依然是没有更新的版本，这条路也走不通

最后想到每次点返回后，活动栈中的活动被唤醒前会执行 `onResume()` 函数，于是想到把原来 `onCreate()` 中的所有部分除载入布局外等全部拷贝一份到重写的 `onResume()` 中去，这部分包含了对 `ListView` 的更新，理论上能够更新 `MainActivity` 的内容。同时 finish() 掉 `addActivity()` 即可

这个方法果然是可行的，不过感觉 `onCreate()` 和 `onResume()` 里面的内容几乎一致不是很好，于是尝试把诸如读取数据库之类的操作放在这两个函数外面，减少重复代码，可却发现游标的调用会出现异常...但办法还是有的，经过查询发现创建活动的时候不仅会执行 `onCreate()` ， `onResume()` 也会被执行，也就是说 `onResume()` 不仅仅在从其他活动返回时被执行。这就方便多了，删除掉 `onCreate()` 中和 `onResume()` 重复的部分即可，只保留必要的部分。最终 `onCreate()` 内容如下

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    //不在应用中显示应用名
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    //载入布局
    setContentView(R.layout.activity_main);

}
```

一个没有解决的问题是模拟器中点击 `ListView` 的对象时会因为没有通讯录权限而闪退，不太清楚原生Android授权的原理，所以最后所有截图都是通过小米Note获得的

思考与总结

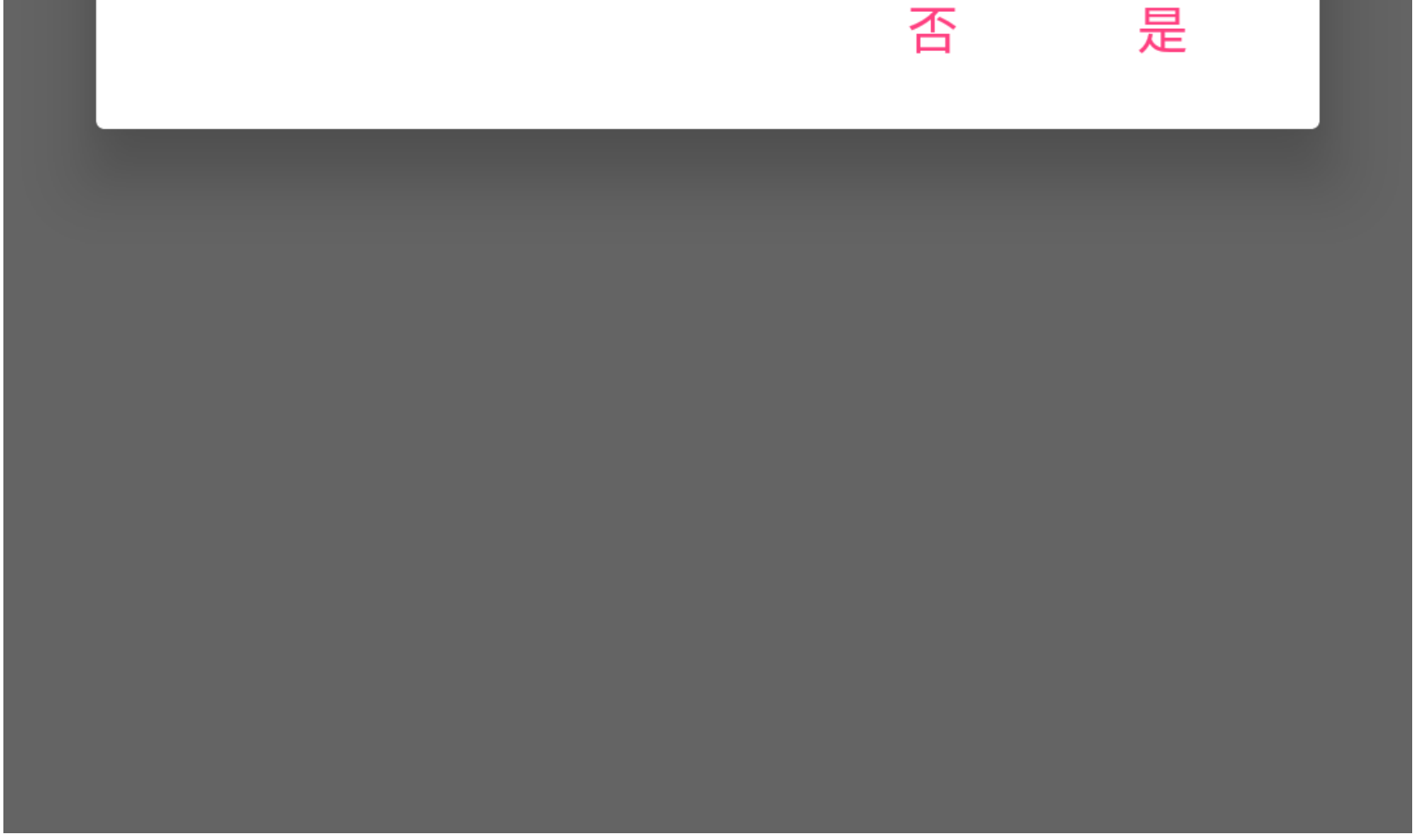
本次试验深入了解了Android开发的有关知识，尤其是和数据库以及 `ContentProvider` 有关的内容。这次实验的内容较大，大部分内容可以通过查阅PPT或作业说明得到解决，只有少部分需要百度。

有一个小的改进，一般情况下 `EditText` 有内容时光标会停留在开始的位置不变，这为之后的编辑造成了小小的困扰。于是我增加了对光标的设置，使得载入后光标自动定位在文本的末尾

```
//设置光标位置为文本结尾处
Editable text = birthday_text.getText();
birthday_text.setSelection(text.length());

Editable text2 = gift_text.getText();
gift_text.setSelection(text2.length());
```

如图所示



原理部分更多的参考了上课的课件。Android较UWP应用范围更广，网上的资料也更多，问题也解决的比较顺利。总的来说比较有收获