技术报告

1.CleanText 控件，这是一个自定义的 EditText，通过继承 EditText 类，然后重写一些函数，并添加自定义函数，主要实现的效果是，当用户点击输入框输入数据时，在输入框最右边会出现一个“X”，当用户点击这个“X”时，会清除所有已经输入的数据。

2.帧动画。查看 UI 之后发现我开发的页面涉及到网络请求，所以采用帧动画实现在网络请求等待返回结果时的等待动画，主要是运动帧动画技术，收集了十张连续的图片，设置合适的间隔循环播放这些图片，以实现等待响应的动画效果。

3.Retrofit 网络通信框架。基于性能、易用性以及代码可拓展性角度考虑，我采用谷歌推荐的 Retrofit 网络请求框架，这个框架学习起来比较费劲，因为网上的中文博客大多讲的肤浅，所以只能查看英文文档以及源码学习，花费了较多时间。该框架首先需要定义一个网络接口，在该接口中声明请求的方式（GET、POST 等），以及发起请求时调用的函数以及需要传递的参数，具体形式如下：

```
public interface RetrofitInterface {

    @FormUrlEncoded
    @POST("login")
    Call<LoginRetro> login(@Field("username") String username, @Field("password") String password);

    @GET("getunit")
    Call<UnitRetro> getUnit(@Query("id")int id);

    @GET("getdepartment")
    Call<DepartmentRetro> getDepartment(@Query("id")int id);
}
```

具体使用，如下：

```
Retrofit retrofit = new Retrofit.Builder().baseUrl(IP).addConverterFactory(GsonConverterFactory.create()).client(client).build();
RetrofitInterface service = retrofit.create(RetrofitInterface.class);
Call<LoginRetro> call = service.login(username, password);
call.enqueue(new Callback<LoginRetro>() {
    @Override
    public void onResponse(Call<LoginRetro> call, Response<LoginRetro> response) {

        if (response == null)
            Log.d("response", "空的");
        else {

            if (response.body().status == 1) {

                animationDrawable.stop();
                imgView.setVisibility(View.INVISIBLE);

                CurrentUser.getInstance(context).saveLoginInfo(username,response.headers().get("cookie"),response.body().user.id,response.
                getCompany(response.body().user.companyID);
                getApartment(response.body().user.appartmentID);
                Toast.makeText(Login.this, "登录成功！", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(Login.this, MainActivity.class);
                startActivity(intent);

            } else {
                animationDrawable.stop();
                imgView.setVisibility(View.INVISIBLE);
                Toast.makeText(Login.this, "登录失败！", Toast.LENGTH_SHORT).show();
            }
        }

    }

    @Override
    public void onFailure(Call<LoginRetro> call, Throwable t) {
        animationDrawable.stop();
        imgView.setVisibility(View.INVISIBLE);
        Log.d("异常出现", t.getMessage());
        Toast.makeText(Login.this, "出现异常，登录失败！", Toast.LENGTH_SHORT).show();
    }
});
```

这其中涉及一个 GsonConverterFactory 转换器，就是将服务端返回的 gson 自动转换为我定义的类型，比如说这里的 LoginRetro 模板，转换器会将返回数据与模板中相同的变量名进行匹配并转换，最后返回一个模板的对象。模板代码如下：

```
public class LoginRetro {
        public int status;
        public UserInfo user;
        public class UserInfo {
            public int id;
            public int authority;
            public int sex;
            public String driverType;
            public String identify;
            public String phone;
            public String photoURL;
            public String address;
            public int companyID;
            public int appartmentID;
            public int jobNo;

        }
}
```

最后总结一下 Retrofit 框架：1)其本质上就是对 okHttp 的封装，使用面向接口的方式进行网络请求。2)使用注解和动态代理，极大的简化网络请求的步骤。3)通过转换器自动转换服务端返回的数据,极大地减少了在数据处理方面的工作量。Retrofit 提供几种默认的转换器，例如 GsonConverterFactory，XMLConverterFactory 等.