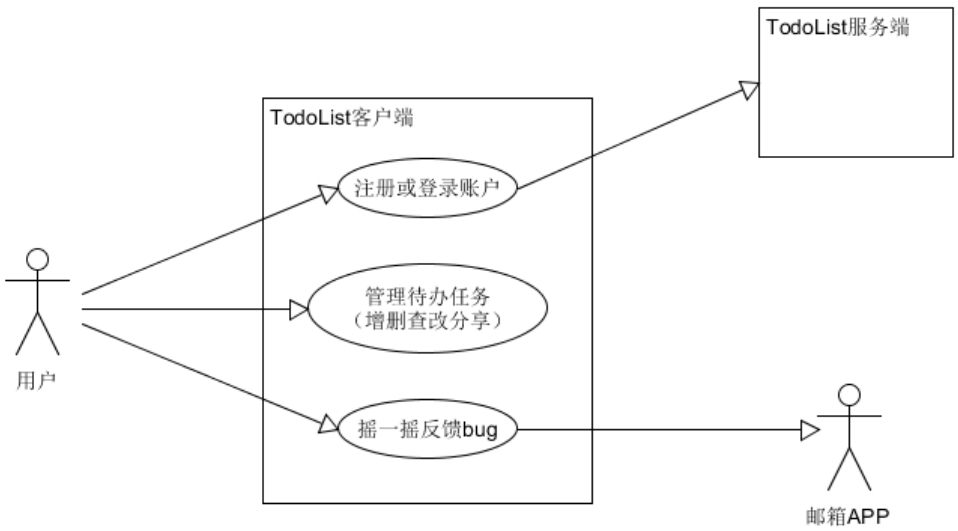
软件设计文档

技术选型

本项目的基本需求是客户端能在主流 Android 设备上运行，因此客户端选择使用 Android Studio 进行开发，使用 Java 和 Xml 语言，数据库使用 SQLite，最低支持 Android4.4.4 版本。

客户端架构设计

用例图

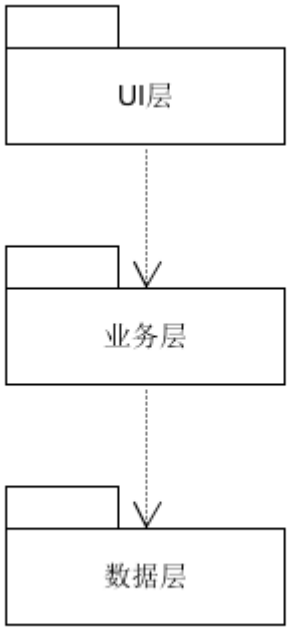


TodoList 客户端的需求

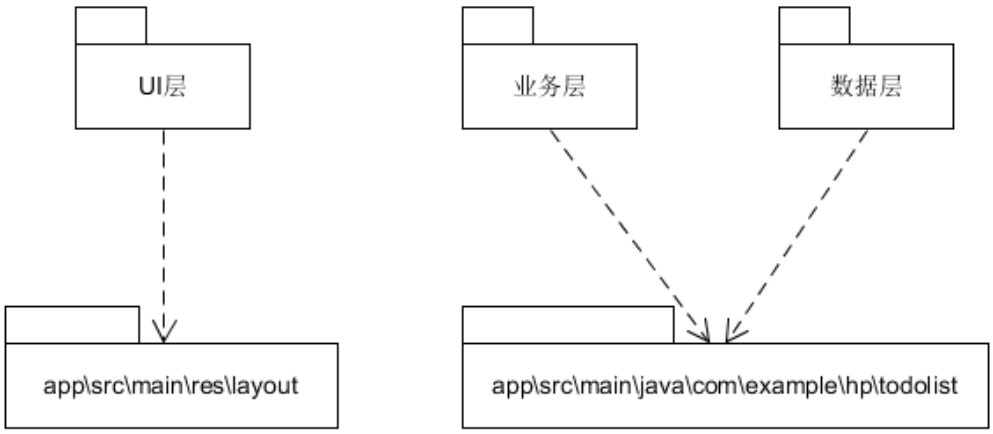
非功能需求			功能需求
约束	运行期质量属性	开发期质量属性	注册或登录账户
使用 Java 和 xml 语言进行开发 最低支持 Android4.4.4 版本的 Android 系统设备 可调用服务端 API	高性能 鲁棒性	模块间低耦合 易测试	管理待办任务（增删查改分享） 任务定时提醒 桌面小插件 摇一摇反馈 bug

团队开发，使用 GitHub 管理项目			

逻辑视图



开发视图



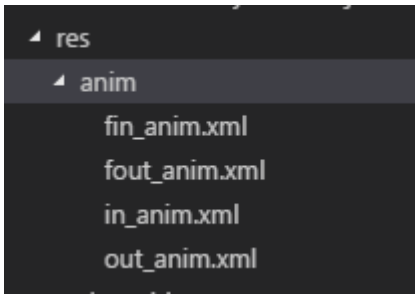
模块划分设计

客户端：

注册登录模块
数据库模块
任务定时提醒模块
传感器模块
任务详情编辑模块
任务时间设置模块

软件设计技术

Material Design 滑动动画的最佳实践



侧滑栏和 Fragment 的最佳实践
MainActivity'.java

```
mDrawerLayout = (DrawerLayout) findViewById(R.id.mainDrawerLayout);
ActionBarDrawerToggle mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout, toolbar,
        R.string.drawer_open, R.string.drawer_close);
mDrawerToggle.syncState();//初始化状态
mDrawerLayout.setDrawerListener(mDrawerToggle);

NavigationView mNavigationView = (NavigationView) findViewById(R.id.navigation_view);
getSupportFragmentManager().beginTransaction().replace(R.id.frame_content, new taskFragment()).commit();
```

```
mNavigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(MenuItem menuItem) {
        switch (menuItem.getItemId()) {
              case R.id.login:
                  Intent intent1 = new Intent(MainActivity.this, LoginActivity.class);
                  startActivity(intent1);
                  break;
            case R.id.search:
                Intent intent = new Intent(MainActivity.this, SearchActivity.class);
                startActivity(intent);
                  overridePendingTransition(R.anim.in_anim, R.anim.out_anim);
                  getSupportFragmentManager().beginTransaction().replace(R.id.frame_content, new searchFragment()).commit();
                  mToolbar.setVisibility(View.INVISIBLE);
                break;
            case R.id.collection:
                getSupportFragmentManager().beginTransaction().replace(R.id.frame_content, new collectionFragment()).commit();
                toolbar.setTitle(R.string.collect_box);
                break;
            case R.id.today_task:
                getSupportFragmentManager().beginTransaction().replace(R.id.frame_content, new taskFragment()).commit();
                toolbar.setTitle(R.string.today);
                break;
        }
        menuItem.setChecked(true);
        mDrawerLayout.closeDrawers();
        return true;
    }
});
```

使用 Handler 更新 UI

因为在 fragment 中有两个 listview 分别显示未完成任务和已完成任务，要实现点击一个任务的 checkbox，使任务在原 listview 中被删除并出现在另一个 listview 中，我采用了 handler 更新 UI 机制：在 fragment 中定义一个 handler 处理 UI 的更新，并将 handler 传入自定义 adapter，自定义 adapter 处理 checkbox 的点击事件，checkbox 被点击则开启一个新线程，发送 message 给 handler。

MyLVAdapter.java

```
finishCB.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                String title = tasks.get("titles").get(position);
                String date = tasks.get("dates").get(position);
                Intent intent = new Intent();
                intent.putExtra("isfinished", finished);
                intent.putExtra("title", title);
                intent.putExtra("date", date);
                Message message = new Message();
                message.obj = intent;
                handler.sendMessage(message);
            }
        }).start();
    }
```

collectionFragment.java

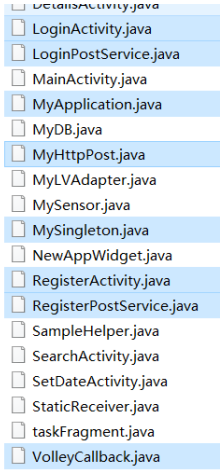
```
Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        Intent intent = (Intent) msg.obj;
        boolean isfinished = intent.getBooleanExtra("isfinished", true);
        String title = intent.getStringExtra("title");
        String date = intent.getStringExtra("date");
        if (isfinished) {
            myDB.unfinishThis(title);
            unfinished.get("titles").add(title);
            unfinished.get("dates").add(date);
            finished.get("titles").remove(finished.get("titles").indexOf(title));
            finished.get("dates").remove(finished.get("dates").indexOf(date));
        } else {
            myDB.finishThis(title);
            finished.get("titles").add(title);
            finished.get("dates").add(date);
            unfinished.get("titles").remove(unfinished.get("titles").indexOf(title));
            unfinished.get("dates").remove(unfinished.get("dates").indexOf(date));
        }
        finishedAdapter.notifyDataSetChanged();
        unfinishedAdapter.notifyDataSetChanged();

        if (finishedAdapter.getCount() == 0) {
            divider.setVisibility(View.GONE);
        } else {
            divider.setVisibility(View.VISIBLE);
        }
    }
};
```

登录注册模块中使用的网络技术是 Volley，其中 Volley 构建请求队列使用了单例模式：

```
43      public static synchronized MySingleton getInstance(Context context) {
44          if (mInstance == null) {
45              mInstance = new MySingleton(context);
46          }
47          return mInstance;
48      }
49
50      public RequestQueue getRequestQueue() {
51          if (mRequestQueue == null) {
52              // getApplicationContext() is key, it keeps you from leaking the
53              // Activity or BroadcastReceiver if someone passes one in.
54              mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());
55          }
56          //mRequestQueue.getCache().clear();
57          return mRequestQueue;
58      }
59
60      public <T> void addToRequestQueue(Request<T> req) {
61          getRequestQueue().add(req);
62      }
63
64      public ImageLoader getImageLoader() {
65          return mImageLoader;
66      }
67  }
```

由于网络通信方面为了实现可扩展性和模块化的设计，客户端的注册和登录模块如下：



所以在 MyHttpPost 中获取到从服务器返回的信息时，采用回调接口的方式接收信息（VolleyCallback）。

具体的实现是（以登录为例）：在 LoginActivity 中获取到登录页面的用户输入的 id 以及登录密码，开启线程（为了避免造成 UI 线程阻塞，需要网络的模块在子线程中实现），将数据发送出去，通过 VolleyCallback 接口返回的数据判断是否登录成功。

```
private void LoginPostThread(final String id, final String password) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            if (!id.equals("")) {
                // 要发送的数据
                Map<String, String> map = new HashMap<>();
                map.put("id", id);
                map.put("password", password);
                Log.d("id", id);
                Log.d("password", password);
                // 发送数据，获取对象
                LoginPostService loginPostService = new LoginPostService();
                loginPostService.send(map, new VolleyCallback() {
                    @Override
                    public void onSuccess(String result) {
                        test.setText(result);

                        // 准备发送消息
                        Message message = new Message();
                        if (result.equals("FAILED")) {
                            message.what = FAILED;
                        } else if (result.equals("SUCCEEDED")) {
                            message.what = SUCCEEDED;
                        } else {
                            message.what = 666;
                        }
                        //Log.d("妈卖逼.....", "" + message.what);
                        handler.sendMessage(message);
                    }
                });
            }
        }
    }).start();
}
```

由于实现了登录和注册的功能，服务器这里进行了分类，所以使用 LoginPostService 来定位

服务器的 servlet(登录)，之后将定位后的 servlet 以及参数传到 MyHttpPost 中实现真正与服务器交互的部分，这里使用 volley 框架实现：

```
public void executeHttpPost(String servlet, final Map<String, String> params, final Voll
    final String baseURL = SERVER + PROJECT + servlet;
    // 创建请求队列
    // 在MySingleton中实现，单例模式

    // 创建一个post请求
    StringRequest stringRequest = new StringRequest(Request.Method.POST, baseURL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    try {
                        String responseMsg = new String(response.getBytes(), "utf-8");
                        Log.i("tag", "MyHttpPost: responseMsg = 我爱你:  " + responseMsg
                        volleyCallback.onSuccess(responseMsg);
                    } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {

        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            return params;
        }
    };

    //stringRequest.setShouldCache(false);

    // 将请求添加到请求队列中
```

至于服务器端，实现比较简单，就是 LoginServlet 和 RegisterServlet 两个接收到客户端发送过来的数据，比如说在登录方面，就是通过 MyService 来访问数据库，执行一条查询语句来判断用户是否存在以及用户和密码是否匹配。注册的话就是想数据库插入一条数据。
MyService 中的注册的插入语句实现：

```java
public static int register(String id, String username, String password) {
    int result = REGISTER_FAILED;
    updateRowCnt = 0;

    // 执行 SQL 插入语句
    String sql = "insert into user_list(`id`, `username`,`password`) values ('"
            + id + "', '" + username + "', '" + password + "')";
    String csql = "select * from user_list where id='" + id +"'" + " limit 1";
    try {
        Connection connect = DBManager.getConnection();
        // 检查id是否已经存在
        preparedStatement2 = connect.prepareStatement(csql);
        if (preparedStatement2.executeQuery().next()) {
            preparedStatement2.close();
            connect.close();
            result = REGISTER_FAILED;
        } else {
            preparedStatement = connect.prepareStatement(sql);
            try{
                updateRowCnt = preparedStatement.executeUpdate();
                // 插入结果
                if(updateRowCnt != 0) {
                    result = REGISTER_SUCCEEDED;
                    System.out.println("id:" + id
                        + " username:" + username //resultSet.getString("username")
                        + " --register");
                }
                preparedStatement.close();
                //resultSet.close();
                connect.close();
            } catch(Exception ee){
                ee.printStackTrace();
            }
        }
    } catch(Exception ee){
        ee.printStackTrace();
    }
    System.out.println("register service result:" + result);
    return result;
}
```

注：此项目为 Android 手机应用程序开发课程项目。