

一、实验概况

实验时间：2019 年 9 月 18（周三）晚上 18:30-21:30，共 4 课时

实验目的：掌握设计、开发、测试、发布、调试经典三层架构软件的基本方法、工具和流程，理解层次体系结构风格基本原理、结构和特点。掌握设计系统时的“高内聚低耦合”的思想。

背景及要求：

[综述研究背景：概述本项工作的研究或观察的理论基础，给出简明的理论或研究背景，一定要列举重要的相关文献。若可能指出存在问题：说明为什么要做这项工作；阐述研究目的：说明有别于他人的“主意”（此红色字体一条不做强行要求）。]

三层架构就是将整个业务应用划分为：表现层、业务逻辑层、数据访问层。区分层次的目的即为了“高内聚低耦合”的思想，在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构，三层架构软件系统为用户的数据传输、提取、储存创造了便利条件。在应用数据时，信息划分架构开发项目，对各层次之间的工作职责进行清晰规划，这样就降低了系统的维护风险。

具体任务如下：

结合课堂上讲授的“一个简单的用户信息查询程序三层逻辑架构”原理，参考以下链接中给出的 C# 代码，<http://www.codeproject.com/Articles/36847/Three-Layer-Architecture-in-C-NET>，完成：

1.结合以下示例数据库或自选其他相当规模数据集，使用 Java 设计实现一个三层架构的业务数据分析系统。各逻辑层的功能如下：

表现层：包含输入、查询相关的控件以及数据图表的展示（如百分图，折线图）；

业务逻辑层：数据处理、数据分析（不少于三项统计分析功能）、数据查询；

数据访问层：负责数据库的访问，主要职责为打开、关闭数据库、构建 SQL 查询、返回查询结果。

附：SqlServer 示例数据库 Northwind

<https://www.cnblogs.com/mahongbiao/p/3764782.html>

附：上证 1999-2016 的某公司股票走势数据，构建其业务数据分析系统。

2. 修改程序以适应三个逻辑层的分布式部署，要求三个逻辑分层分别部署于客户机（本机或手机）、AWS EC2 应用服务器和 AWS RDS 数据库服务器上（即多层 C/S 架构），部署完成后能通过公网 IP 访问该系统。

3.使用 RSA 或 Visio 等建模工具构建软件架构模型（UML 图），要求：

（1）画出逻辑分层结构图；

（2）画出每个逻辑层中所包含的核心构件（此处为类）；

（3）画出每个逻辑层中构件（类）之间的关系，且要细化到聚合(Aggregation)、组成 (Composition) 关系并给出重数（如 1:1,1:*）；

（4）画出系统部署结构图。

二、实验设计(给出你的实习内容的设计方案，可根据实际情况调整条目)

2.1 系统需求

技术环境需求:

- ①使用 java 技术
- ②使用 AWS EC2 和 AWS RDS

功能需求:

表现层:

- ①能够进行请求查询，请求开盘价、收盘价、换手率、成交总额的相关数据
- ②能够接收业务层的数据，显示成折线图、柱状图

业务层:

- ①能够处理表现层传达的请求
- ②向数据层请求数据并接收
- ③对数据层的数据进行处理（如：取平均值），处理完后发送给表现层

数据层:

- ①接收业务层的请求数据并进行相应
- ②能够进行数据查询

约束（自定义的通讯协议）

客户端：消息分为三段，第一段指出是查询类型，如：某一天、某一月、平均值，第二段是查询种类，如：查询开盘价、收盘价等，第三段是查询的日期，三段信息使用“#”连接，其中第一段和第二段的信息对照表如下：

01	查询单天
02	查询月
03	查询平均值

第一段信息对照表：表 1

01	查找开盘价
02	查找收盘价
03	查找换手率
04	查找成交总额

第二段信息对照表：表 2

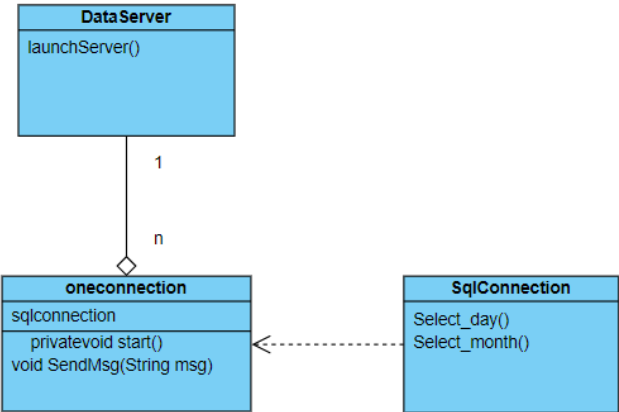
例子：01#01#2010-05-10，查询 2010 年 5 月 10 号的开盘价

服务器：消息分为两段，第一段指出发送的信息类型（与表 1 相同），如：单天、某月、平均值，第二段是查询结果，消息间使用“#”连接

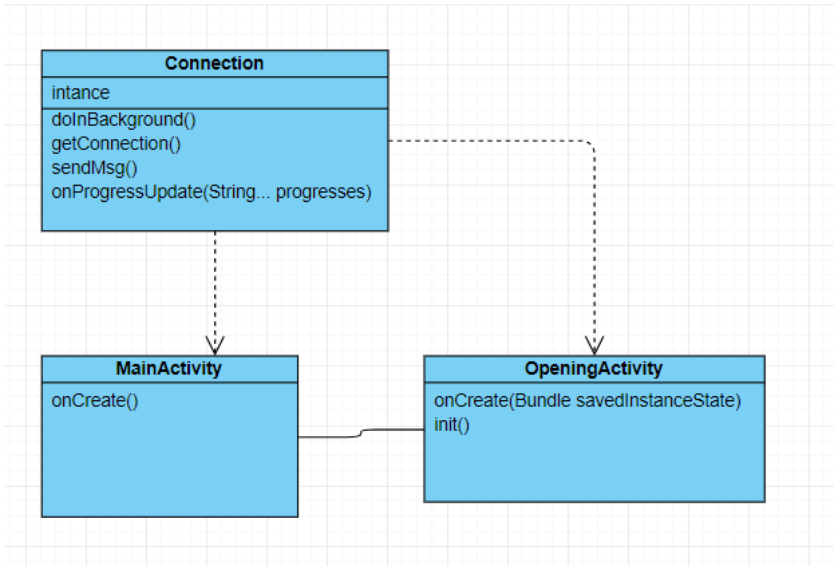
例子：01#8.03，查询单天的结果为 8.03

2.2 架构设计

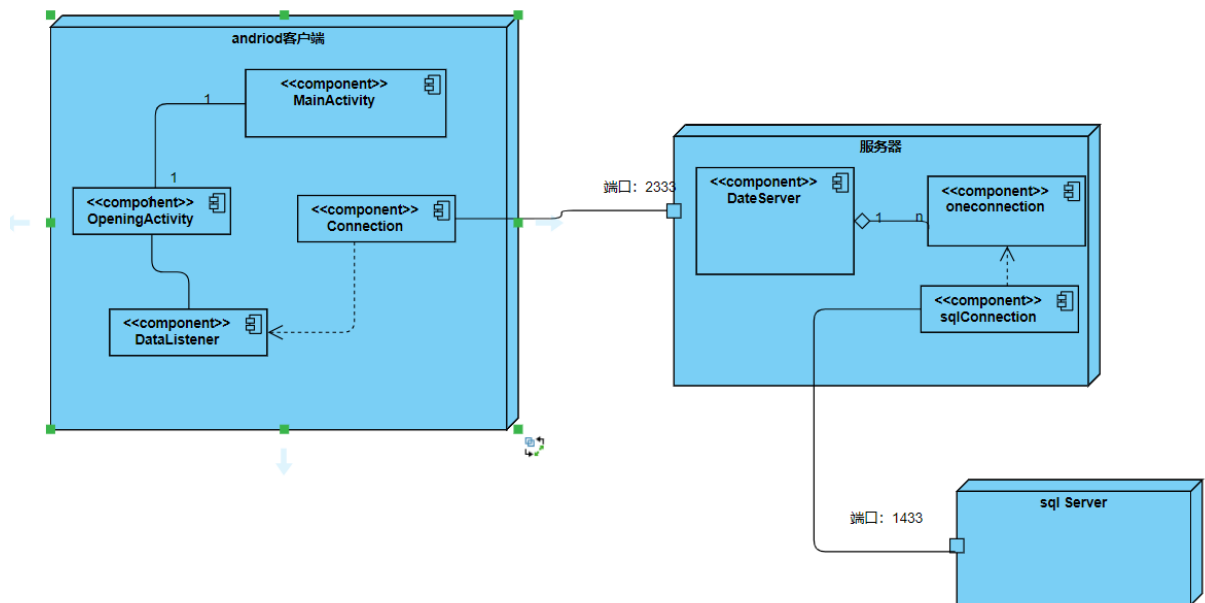
服务器类图:



客户端类图:



部署结构图:



表现层：android 客户端，部署在 android 手机上，其中构件 Connection 通过套接字连接服务器。

业务逻辑层：服务器部署在 aws 的 ec2 上，开放了端口 2333，能够让客户端连接，并且通过 jdbc 连接 sql server 数据库。

数据层：sql server 服务器，部署在 aws 的 RDS 上，能够让业务逻辑层连接，进行数据查询。

构件介绍：

客户端：

构件名字	功能描述
Connection	是客户端的连接类，使用套接字，负责客户端和服务器的连接，负责和服务进行交互，接收和发送消息，采用单例模式，因为整个客户端程序只需要一个连接，
MainActivity	是负责选取对那种信息进行查询，如：到底是选择开盘价信息查询还是收盘价信息查询由 MainActivity 决定，是程序的入口
OpeningActivity	是 MainActivity 进入之后的活动，负责具体的信息查询，如：查询一天还是查询一个月，是用折线图表示还是柱状图。并且最终结果在这上面显示。
DataListener	是 Connection 中的监听，负责有信息传递过来时，就使视图更新。

服务器：

构件名字	功能描述
DataServer	最外层的应用层，是和客户端交互的构件
oneconnection	中间的处理层，由 DataServer 监听到连接后就交由 oneconnection 进行正真处理，并且调用 SqlConnection 的数据库连接查询信息
SqlConnection	和数据库交互的连接层，使用 jdbc 连接，连接后将数据传给 oneconnection 处理

2.3 接口设计

服务器端：

DataServer:

接口名字	参数	返回值	描述
public void launchServer()			负责开启服务器，然后开始监听

Oneconnection:

接口名字	参数	返回值	描述
private void start()			开始 oneconnection 的接收请求，然后做出相应的处理
void SendMsg(String msg)	Msg: 需要发送的信息		将处理好的数据发送给

SqlConnection

接口名字	参数	返回值	描述
public String Select_day(int index,String date)	Index: 第几行数据 Date: 查询的日期	String 类型的结果	负责使用 jdbc 然后进行某一天的查询
public String Select_month(int index,String date)	Index: 第几行数据 Date: 查询的日期	String 类型的结果	负责使用 jdbc 然后进行某一月数据查询

客户端：

Connection:

接口名字	参数	返回值	描述
public static Connection getConnection()		得到类 Connection 的实例（也 是唯一的实 例）	因为类 Connection 是 单例模式，使用 getConnection （）获得实例
Void setListener (DataListener listener)	DataListener listener		设置监听，当有 数据传来时可 以通过监听器 通知活动改变 视图
protected Boolean doInBackground(Void... voids)	protected void onProgressUpdate(String... progresses)	返回关闭是 否成功	背后执行任务， 负责一直处于 接收消息的状 态
protected void onProgressUpdate(String... progresses)	Progresses: 字符串数组，市 场 doInBackground 中接收的 数据		用于更新信息， 通知 listener 更新视图

MainActivity:

接口名字	参数	返回值	描述
public void onClick(View v)	View v		设置各个按钮的点击事 件，并且通过这个接口 进入 OpeningActivity

OpeningActivity:

接口名字	参数	返回值	描述
private void init()			初始化各种控件

DataListener:

接口名字	参数	返回值	描述
Void onDay(String result);	Result 为查询某 一天的结果		得到 connection 提供 的查询某一天的结果， 然后进行视图更新
void onMonth(String result);	Result 为查询某 一月的结果		得到 connection 提供 的查询某一月的结果， 然后进行视图更新
void onAvg(String	Result 为查询某		得到 connection 提供

result);	一月平均值的结 果		的查询某一月平均值的 结果，然后进行视图更 新

三、实验过程

3.1 软件实现

系统开发：

- ①Android 程序开发
- ②套接字编程
- ③Linux 服务器搭建
- ④sql 语句使用
- ⑤jdbc 使用

3.2 实验环境

[包括使用的硬件、软件（服务器/客户端操作系统，服务器、数据库、虚拟机等支撑软件）、实验场景（若有必要的话简单描述一下实验场景情况并给出照片）]

硬件：lenovoR720

软件：windows10、Eclipse、Andriod Studio、Xshell、WinSCP、Sql Server、AWS EC2、AWS RDS

3.3 实验步骤

andriod 客户端：

- ①直接运行程序，界面会出现四个选择按钮（见实习结果展示图），接着可以选择查询类型，选择查询“开盘价”。
- ②选择查询类型之后，会进入一个集体查询的界面，在想要查询的输入框输入日期，点击查询即可。
在查询某一天的输入框输入日期数据：2010-05-10，点击查询；在查询某一月的输入框，输入日期数据：2014-03，点击查询；在查询平均值的输入框，输入日期数据：2015-05，点击查询。
- ③查询某月的数据时，可以在菜单栏可以选择是使用折线图还是柱状图展示。
在查询某一月的输入框，输入日期数据：2014-03，点击菜单栏选择折线图（折线图是默认），点击查询；在查询某一月的输入框，输入日期数据：2014-03，点击菜单栏选择柱状图，点击查询。

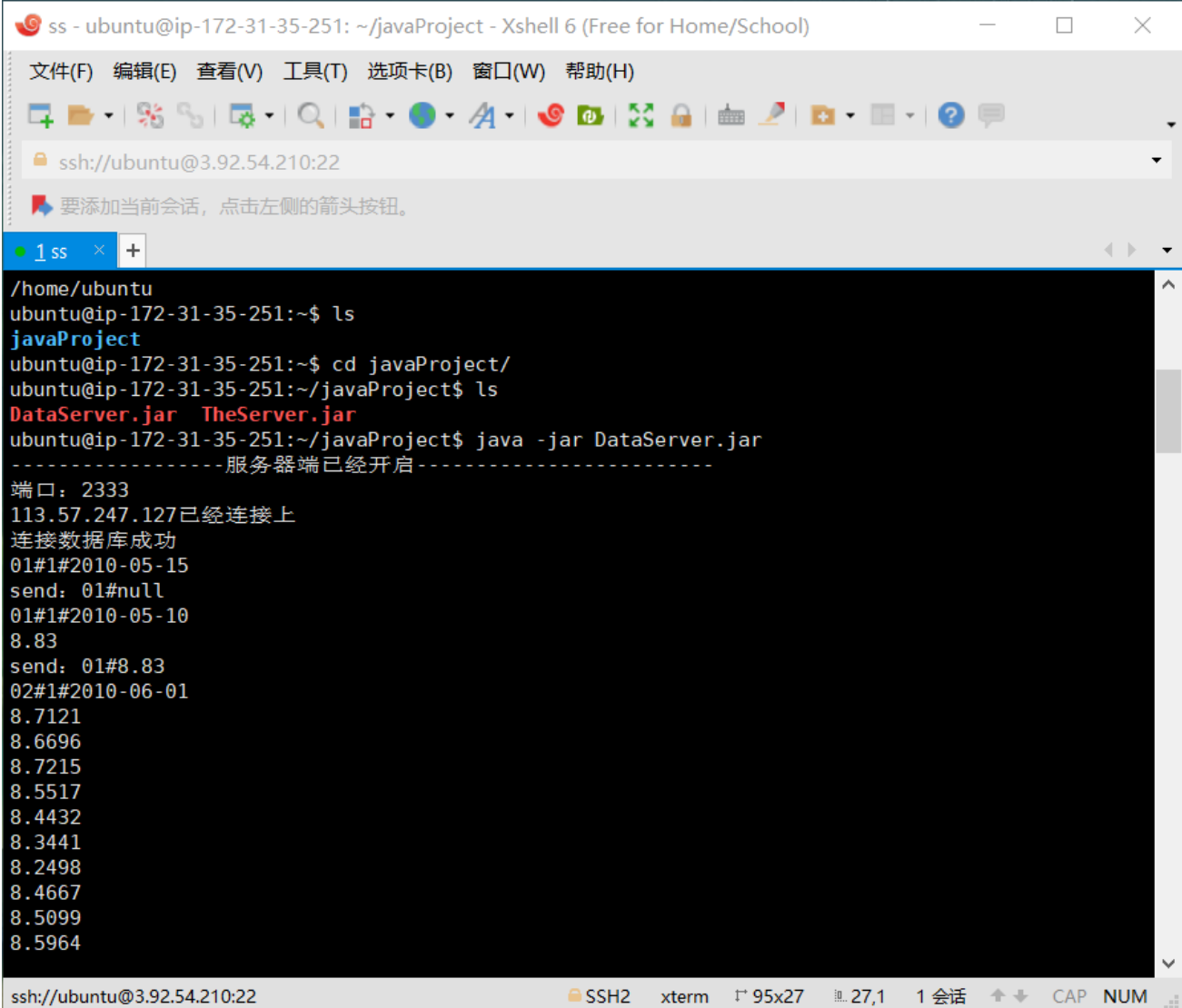
服务器端：

- ①在 EC2 上直接使用命令行 `java -jar DataServer`, 在后面可以选择加入参数 `-p`, 配置端口, 如果不加参数默认开启 2333 端口
- ②退出服务器可以选择杀死进程即可。

四、实验评价

4.1 实验结果

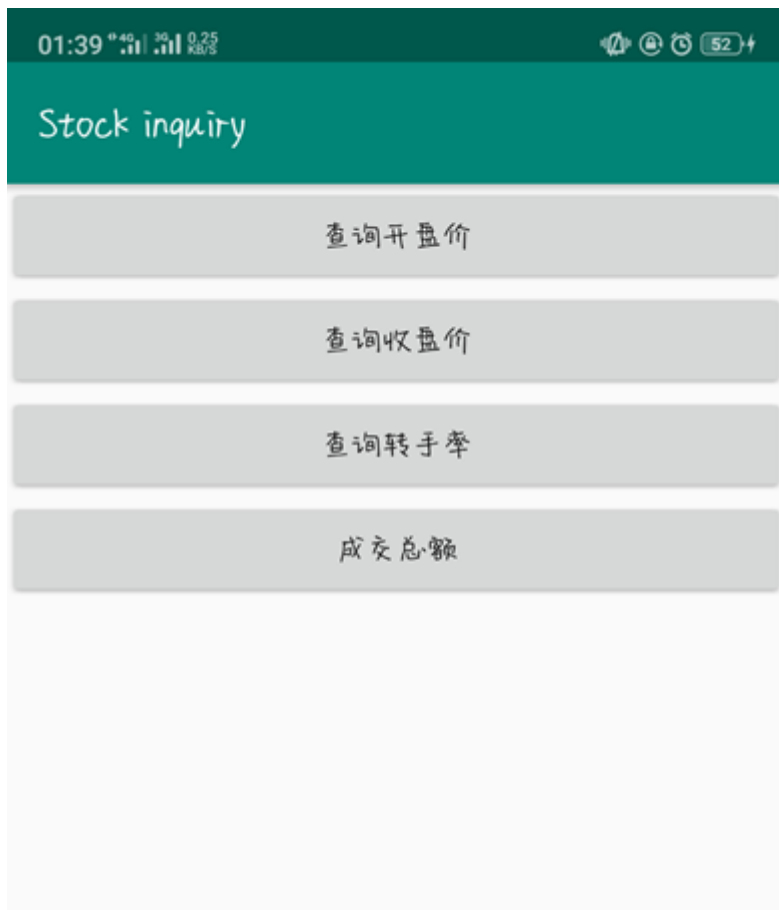
在 EC2 上运行服务器：



The screenshot shows an Xshell 6 terminal window titled "ss - ubuntu@ip-172-31-35-251: ~/javaProject - Xshell 6 (Free for Home/School)". The terminal displays the following commands and output:

```
ssh://ubuntu@3.92.54.210:22
要添加当前会话，点击左侧的箭头按钮。
1 ss x +
/home/ubuntu
ubuntu@ip-172-31-35-251:~$ ls
javaProject
ubuntu@ip-172-31-35-251:~$ cd javaProject/
ubuntu@ip-172-31-35-251:~/javaProject$ ls
DataServer.jar TheServer.jar
ubuntu@ip-172-31-35-251:~/javaProject$ java -jar DataServer.jar
-----服务器端已经开启-----
端口：2333
113.57.247.127已经连接上
连接数据库成功
01#1#2010-05-15
send: 01#null
01#1#2010-05-10
8.83
send: 01#8.83
02#1#2010-06-01
8.7121
8.6696
8.7215
8.5517
8.4432
8.3441
8.2498
8.4667
8.5099
8.5964
ssh://ubuntu@3.92.54.210:22 SSH2 xterm 95x27 27,1 1 会话 CAP NUM
```

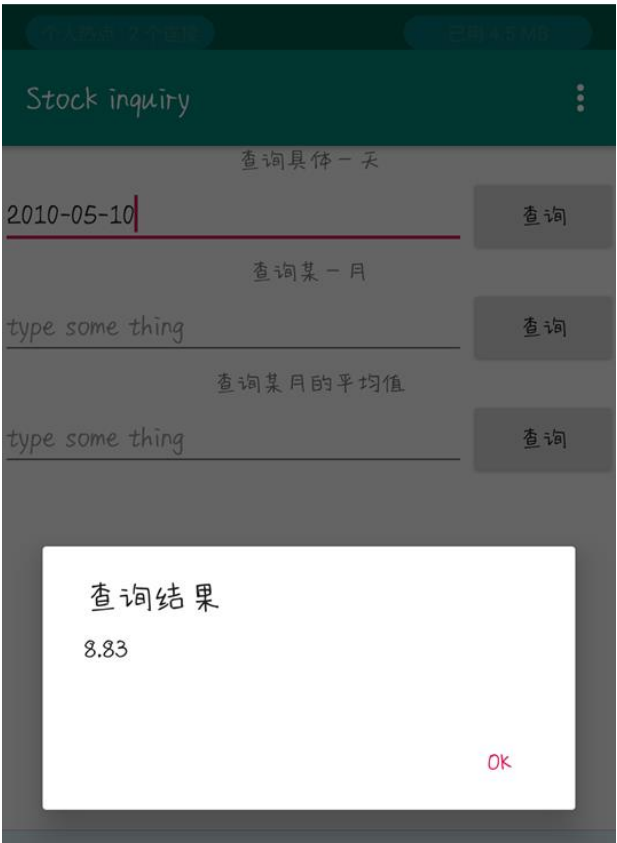
客户端刚刚进入的界面：



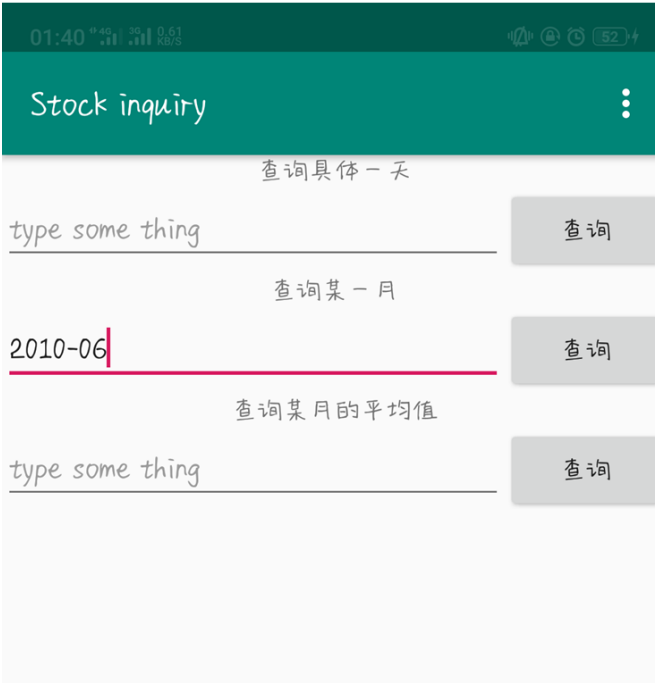
查询某一天的开盘价：



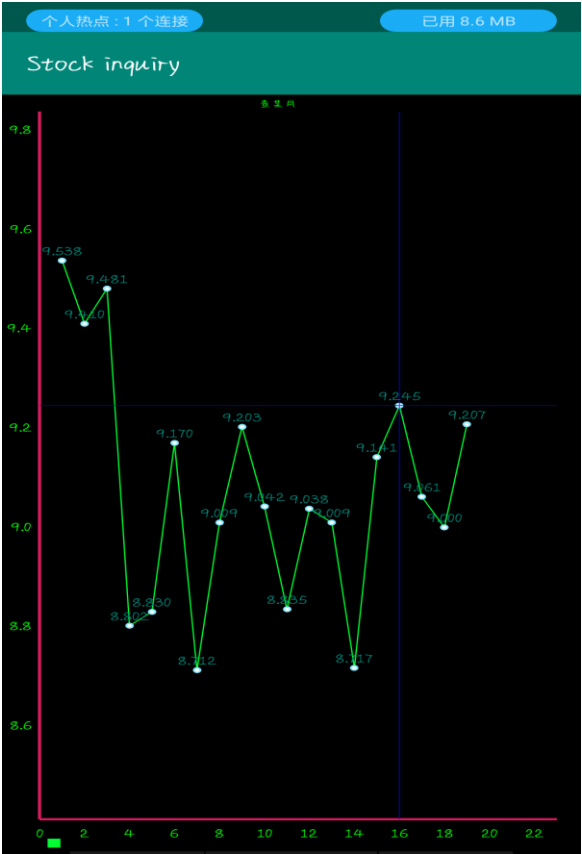
查询结果：



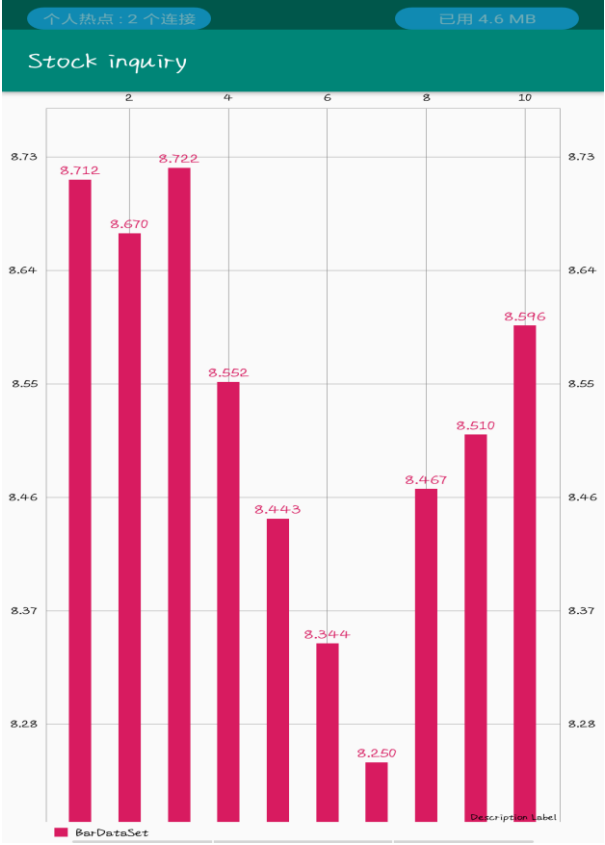
查询某月的开盘价



折线图显示:



柱状图显示:



查询某月的平均值：



4.2 结果分析

实验结果基本达到预期目标，能够查询相应的信息。

局限性：

- ①实验最多只用了两部手机实验，没有使用多部手机连接服务器查看结果，一些潜在并发问题，可能发现不了。
- ②实验没有使用大量数据测试，一些程序上的漏洞可能存在。

五、总结

如：

- (1) [对本次实验报告工作的整体总结。包括：结果要点和结论，本次实习收获和不足。]
 - (2) [给出开发中遇到的问题及解决办法，程序中待解决的问题及改进方向。]
 - (3) [总结性地阐述本实习结果可能的应用前景、局限性及需要进一步深入研究的方向。]
- 本次实习是对三层架构的一次练习,在实习当中我逐渐理解了三层架构的优点以及不足，并学会了三层架构的初步使用。三层架构条理清晰，分工明确，能够很简单的就提供服务，但是三层架构还是过于复杂繁琐，像现在的 AWS 服务，给个函数就能提供服务，使用起来比三层架构简单很多，是以后的趋势所在。实习当中我收获了安卓图表的绘制，学习到了 EC2 部署以及加强了数据库相应知识的编写，不足的是安卓客户端界面写的不是特别好看，以及服务器提供的功能也比较少，如果以后有时间一定继续把该填的坑写上。

遇到的问题：

1. android 中不知道怎么画图

解决方式：查找资料，知道安卓有许多画图的方式，然后选择了 MPAndroidChart 这个 jar 包导入，再进行图形的绘制，还要看官方的文档。

2. 将表格数据导入

解决方式：通过 SQL Server 管理工具 Microsoft SQL Server Management Studio 17 连接上数据库后，再通过导入文件本机导入。

3. 服务器部署问题

解决方式：查看 AWS 官网教程，然后自己通过 Xshell 连接上 EC2 的 ubuntu 服务器，再上面安装 java 环境，再通过 Winscp 将自己写的 jar 包导入上去，通过命令行运行，成功的实现了部署。

程序中待解决的问题：

①客户端添加表格显示

②客户端添加饼状图显示

②服务器添加更多的数据处理和查询的功能

最后我觉得此程序的应用前景是写一个 android 端的炒股软件，市面上也有许多相对应的软件。