**实 验 报 告**

**课程名称：** **移动应用开发**

**学 院： 计算机科学与工程学院**

**专 业：计算机科学与技术** **班 级： 18级1班**

**姓 名：韩晨晨 学 号：201801060104**

**年 月 日**

**山 东 科 技 大 学 教 务 处 制**

**实 验 报 告**

**页**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **组 别** |  | **姓 名** | **韩晨晨** | **同组实验者** |  |
| **实验项目**  **名称** | **BookStore大作业** | | **实验日期** | **1.1** | |
| **教师评语** |  | | | | |
| **实验成绩：** | | | **指导教师（签名）：**  **年 月 日** | | |
| 1. **app功能**   （1）项目连接云端服务器，实现将数据存储云端。  （2）支持多用户。用户可注册，注册后可登录。    （3）用户登录后呈现图书的列表，可以根据书籍姓、作者、价格区间等信息进行筛选。    （4）点击某本图书后可以显示图书的详情（详情内容自定义）。在详情页面可以将该图书加入购物车。    （4）提供结算功能，根据购物车中的图书及数量，显示总额。点击后进行支付。    （5） 提供书籍评论功能，显示每位用户对于不同书籍进行评论，并通过与服务器的交互查看其它用户的评论。    (6) 分类页面可以根据书籍类别显示各类书籍。    （7）我的页面可以查看用户信息，订单，修改签名等功能。     1. **系统设计**   **数据库设计**  **User表**（表示用户信息）  uid 用户编号  uname 用户姓名  upassword 用户密码  umoney 用户余额  utype 用户类型  qian 用户签名；  **Book表**（书籍信息）  bid 书籍编号  bname 书名  bwriter 作者  bdetail 详细信息  bvalue 价格  xiao 销量  btype书籍类型；  **Bill表**（订单表）  oid 订单编号  uid 用户编号  bid 书籍编号  time 订单生成时间  **Remark表**（评论信息）  rid 评论编号  uid 用户编号  bid 书籍编号  comment 评论内容  **系统架构**  （1） 服务器端和android端进行两个项目的编写。  （2） 服务器端搭建基于SSM框架的web项目，在服务器通过mysql进行数据的存储，通过Mybatis在web项目里对数据库进行操作，并实现与android端进行的数据交互，将数据传输给android，将android传输过来的数据进行存储。  （3） android完成前端页面的实现，并通过OkHttp框架进行与服务器的数据交互。  **小组分工**  韩晨晨：进行数据库设计。服务器端代码书写，完成在android端与服务器端进行数据交互的接口，根据android传来信息对服务器端数据库进行修改或返回给android端数据。云服务器的搭建。  胡浩然：购物车界面的实现。在通过Litepal架构进行对android原生数据库的操作，将购物车的信息存入数据库和读出，实现购物车页面的各种功能的实现，对用户余额、书籍销量、订单等通过网络进行对应修改。  袁康裕：分类界面的实现，从服务器端读取书籍信息，通过不同类型的选择，在listview中展示该类别的书籍。商品详情界面的实现，可以查看对应书籍信息，实现加入购物车，购买等功能。评论功能的实现。  刘建豪：我的界面的实现，完成根据登录的用户展示改用户表信息，用户个性签名的修改。订单功能的实现，对于购买的书籍生成响应订单，并在我的页面中可以展示。  樊睿：商品浏览界面的实现，展示所有的书籍，可以通过模糊搜索对应的书名，根基商品价格、销量进行排序，根据作者、书籍类别、价值区间进行筛选。   1. **个人实现** 2. 服务器端基于SSM框架的web项目搭建      1. mysql各类表的创建      1. web中通过Mybatis实现对数据的操作。（以book表为例）   *<?***xml version="1.0" encoding="UTF-8"** *?>* **<!DOCTYPE mapper  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd"*>*** <**mapper namespace="cn.sdust.dao.BookDao"**>  <**sql id="book"**>  select bid,bname,bwriter,bdetail,bvalue,xiao,btype from book  </**sql**>  <**select id="selectBookbyName" resultType="cn.sdust.pojo.Book"**>  select *\** from book  where bname=#{bname}  </**select**>   <**select id="selectBookbyValue" resultType="cn.sdust.pojo.Book"**>  select *\** from book  where bvalue=#{bvalue}  </**select**>   <**select id="selectBookbyId" resultType="cn.sdust.pojo.Book"**>  select *\** from book  where bid=#{bid}  </**select**>   <**select id="selectBookbyWriter" resultType="cn.sdust.pojo.Book"**>  select *\** from book  where bwriter=#{bwriter}  </**select**>   <**select id="selectBookbyType" resultType="cn.sdust.pojo.Book"**>  select *\** from book  where btype = #{btype}  </**select**>   <**select id="selectBook" resultType="cn.sdust.pojo.Book"**>  select *\** from book  </**select**>  <**select id="selectBookbyItem" resultType="cn.sdust.pojo.Book"**>  select *\** from book where 1 = 1  <**if test='bookName!= null and bookName != "" '**>  AND bname LIKE CONCAT(CONCAT('%',#{bookName}),'%');  </**if**>  <**if test='writer!= null and writer != ""'**>  AND bwriter = #{writer}  </**if**>  <**if test='type!= null and type != ""'**>  AND btype = #{type}  </**if**>  <**if test= "minValue!= null and minValue !=-1"**>  AND bvalue >= #{minValue}  </**if**>  <**if test= "maxValue!= null and maxValue !=-1"**>  AND #{maxValue} >= bvalue  </**if**>  </**select**>   <**update id="addXiao"**>  update book  set xiao=#{value}  where bid=#{bid}  </**update**> </**mapper**>   1. 在controller层实现对android端发送请求的处理。   @Controller **public class** UserController {  @Resource  **private** UserDao **userDao**;   @Resource  **private** BookDao **bookDao**;   @Resource  **private** RemarkDao **remarkDao**;   @Resource  **private** BillDao **billDao**;   @RequestMapping(**"/login"**)  @ResponseBody  **public** User Login(@RequestParam(**"username"**) String username,  @RequestParam(**"password"**) String password)  {  User user = **userDao**.selectUserByname(username);  **if**(user.getPassword().equals(password))  **return** user;  **else** {  User user1=**null**;  **return** user1;  }  }  @RequestMapping(**"/register"**)  @ResponseBody  **public int** Register(@RequestParam(**"username"**) String username,  @RequestParam(**"password"**) String password)  {  User user= **new** User();  user.setUname(username);  user.setPassword(password);  user.setUmoney(0);  user.setUtype(0);  User user1 = **userDao**.selectUserByname(username);  **if**(username==**null**)  **return** 0;  **else** {  **userDao**.saveUser(user);  **return** 1;  }  }   @RequestMapping(**"/getbook"**)  @ResponseBody  **public** List<Book> getBook(@RequestParam(**"serchItem"**) String data)  {  *//System.out.println(data);* JSONObject js = JSONObject.*parseObject*(data);  SerchItem si = JSONObject.*toJavaObject*(js,SerchItem.**class**);  *//System.out.println(si.getBookName());  //System.out.println(si.getMaxValue());* List<Book> list = **bookDao**.selectBookbyItem(si);  *// System.out.println(list.size());* **return** list;  }  @RequestMapping(**"/updateqian"**)  @ResponseBody  **public** User getBook(@RequestParam(**"uid"**) String uid,  @RequestParam(**"qian"**) String qian)  {  User user = **userDao**.selectUserByid(Integer.*parseInt*(uid));  **userDao**.UpdateQian(Integer.*parseInt*(uid),qian);  user.setQian(qian);  **return** user;  }  @RequestMapping(**"/getabook"**)  @ResponseBody  **public** Book getABook(@RequestParam(**"uid"**) String uid)  {  Book book = **bookDao**.selectBookbyId(Integer.*parseInt*(uid));  **return** book;  }    @RequestMapping(**"/postremark"**)  @ResponseBody  **public void** PostRemark(@RequestParam(**"remark"**) String remark)  {  JSONObject js = JSONObject.*parseObject*(remark);  Remark rr = JSONObject.*toJavaObject*(js,Remark.**class**);   **remarkDao**.insertRemark(rr);  }  @RequestMapping(**"/getuser"**)  @ResponseBody  **public** User getUser(@RequestParam(**"uid"**) String uid)  {  User user = **userDao**.selectUserByid(Integer.*parseInt*(uid));  **return** user;  }  @RequestMapping(**"/getping"**)  @ResponseBody  **public** List<Remark> getPing(@RequestParam(**"bid"**) String bid)  {  List<Remark> list = **remarkDao**.getRemarkById(Integer.*parseInt*(bid));  **return** list;  }  @RequestMapping(**"/getding"**)  @ResponseBody  **public** List<Bill> getDing(@RequestParam(**"uid"**) String uid)  {  List<Bill> list = **billDao**.getBillByid(Integer.*parseInt*(uid));  **return** list;  }   @RequestMapping(**"/insertbill"**)  @ResponseBody  **public void** insertBill(@RequestParam(**"bill"**) String data)  {  JSONObject js = JSONObject.*parseObject*(data);  Bill order = JSONObject.*toJavaObject*(js, Bill.**class**);  order.setTime(order.getTime().trim());  **billDao**.insertBill(order);  }  @RequestMapping(**"/addxiao"**)  @ResponseBody  **public void** addXiao(@RequestParam(**"bid"**) String data)  {  **int** bid=Integer.*parseInt*(data);  Book book = **bookDao**.selectBookbyId(bid);  **bookDao**.addXiao(bid,book.getXiao()+1);  }  @RequestMapping(**"/jianqian"**)  @ResponseBody  **public void** jianQian(@RequestParam(**"uid"**) String data1,@RequestParam(**"money"**) String data2)  {  **int** uid=Integer.*parseInt*(data1);  **double** money=Double.*parseDouble*(data2);  User user = **userDao**.selectUserByid(uid);  **userDao**.Qianjianjian(uid,user.getUmoney()-money);  } }   1. 实现android端对于服务器端的请求访问。   package com.example.bookstore;  import android.graphics.Bitmap; import android.graphics.BitmapFactory; import com.google.gson.Gson; import com.google.gson.reflect.TypeToken; import java.io.IOException; import java.io.InputStream; import java.util.LinkedList; import java.util.List; import okhttp3.FormBody; import okhttp3.OkHttpClient; import okhttp3.Request; import okhttp3.RequestBody; import okhttp3.Response;  public class HttpUtil {  public static String *IP* = "http://47.110.134.38:8080/SSM\_Demo\_war/";  public static User *MyUser*;  public static Book getABook(int bookid) throws IOException  {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("uid",String.*valueOf*(bookid)).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getabook")  .build();   Response response = client.newCall(request).execute();  String s = response.body().string();  System.*out*.println(s);  Gson gson = new Gson();  Book book = gson.fromJson(s,Book.class);   String url1 =HttpUtil.*IP*+ "/static/image/" + String.*valueOf*(book.getBid()) + "-1.jpg";  OkHttpClient client1 = new OkHttpClient();  Request request1 = new Request.Builder()  .url(url1)  .build();  Response response1 = client1.newCall(request1).execute();  InputStream inputStream1 = response1.body().byteStream();*//得到图片的流* Bitmap bitmap1 = BitmapFactory.*decodeStream*(inputStream1);  book.setPic1(bitmap1);  String url2=HttpUtil.*IP*+ "/static/image/" + String.*valueOf*(book.getBid()) + "-2.jpg";  OkHttpClient client2 = new OkHttpClient();  Request request2 = new Request.Builder()  .url(url2)  .build();  Response response2 = client2.newCall(request2).execute();  InputStream inputStream2 = response2.body().byteStream();*//得到图片的流* Bitmap bitmap2 = BitmapFactory.*decodeStream*(inputStream2);  book.setPic2(bitmap2);   String url3=HttpUtil.*IP*+ "/static/image/" + String.*valueOf*(book.getBid()) + "-3.jpg";  OkHttpClient client3 = new OkHttpClient();  Request request3 = new Request.Builder()  .url(url3)  .build();  Response response3 = client3.newCall(request3).execute();  InputStream inputStream3 = response3.body().byteStream();*//得到图片的流* Bitmap bitmap3 = BitmapFactory.*decodeStream*(inputStream3);  book.setPic3(bitmap3);  return book;  }  public static User getUser(int uid) throws IOException {   OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("uid",String.*valueOf*(uid)).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getuser")  .build();  Response response = client.newCall(request).execute();  String s = response.body().string();  System.*out*.println(s);  Gson gson = new Gson();  User user = gson.fromJson(s,User.class);  return user;  }  public static List<Remark> getRemark(int bid) throws IOException {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("bid",String.*valueOf*(bid)).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getping")  .build();  Response response = client.newCall(request).execute();  String s = response.body().string();  Gson gson = new Gson();  List<Remark> list;  list = gson.fromJson(s,new TypeToken<List<Remark>>(){}.getType());  return list;  }  public static List<Bill> getBill(int uid) throws IOException  {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("uid",String.*valueOf*(uid)).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getding")  .build();  Response response = client.newCall(request).execute();  String s = response.body().string();  Gson gson = new Gson();  List<Bill> list;  list = gson.fromJson(s,new TypeToken<List<Bill>>(){}.getType());  return list;  }   public static List<Book> getBooks(SerchItem serchItem) throws Exception  {  List<Book> bookList = new LinkedList<>();  OkHttpClient client = new OkHttpClient();  Gson gson1 = new Gson();  String jsons= gson1.toJson(serchItem);  RequestBody requestBody = new FormBody.Builder()  .add("serchItem",jsons).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getbook")  .build();  Response response = client.newCall(request).execute();  String s = response.body().string();  Gson gson = new Gson();  System.*out*.println(s);  bookList.clear();  bookList = gson.fromJson(s,new TypeToken<List<Book>>(){}.getType());  return bookList;   }  public static void insertBill(Bill bill)  {  Gson gson = new Gson();  String jsons = gson.toJson(bill);  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("bill", jsons).build();  Request request = new Request.Builder()  .url(HttpUtil.*IP* + "/insertbill")  .post(requestBody)  .build();  try {  Response response = client.newCall(request).execute();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void addxiao(int bid)  {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("bid",String.*valueOf*(bid)).build();  Request request = new Request.Builder()  .url(HttpUtil.*IP* + "/addxiao")  .post(requestBody)  .build();  try {  Response response = client.newCall(request).execute();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void jianqian(int uid,double money)  {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("uid",String.*valueOf*(uid))  .add("money",String.*valueOf*(money))  .build();  Request request = new Request.Builder()  .url(HttpUtil.*IP* + "/jianqian")  .post(requestBody)  .build();  try {  Response response = client.newCall(request).execute();  } catch (IOException e) {  e.printStackTrace();  }  }  public static Book getOnlyBook(int bookid) throws IOException {  OkHttpClient client = new OkHttpClient();  RequestBody requestBody = new FormBody.Builder()  .add("uid" ,String.*valueOf*(bookid)).build();  Request request = new Request.Builder()  .post(requestBody)  .url(HttpUtil.*IP*+"/getabook")  .build();  Response response = client.newCall(request).execute();  String s = response.body().string();  Gson gson = new Gson();  Book book = gson.fromJson(s,Book.class);  return book;  } } | | | | | |