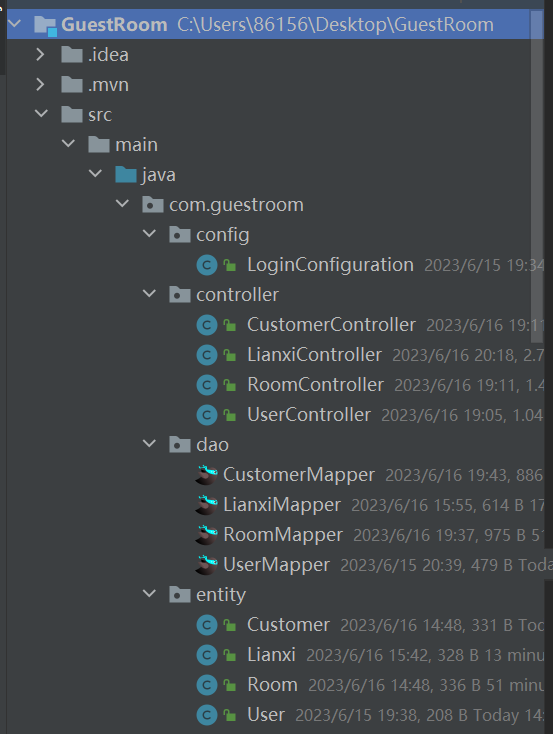
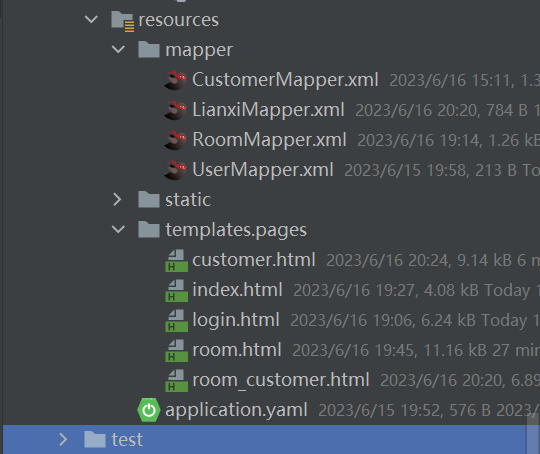
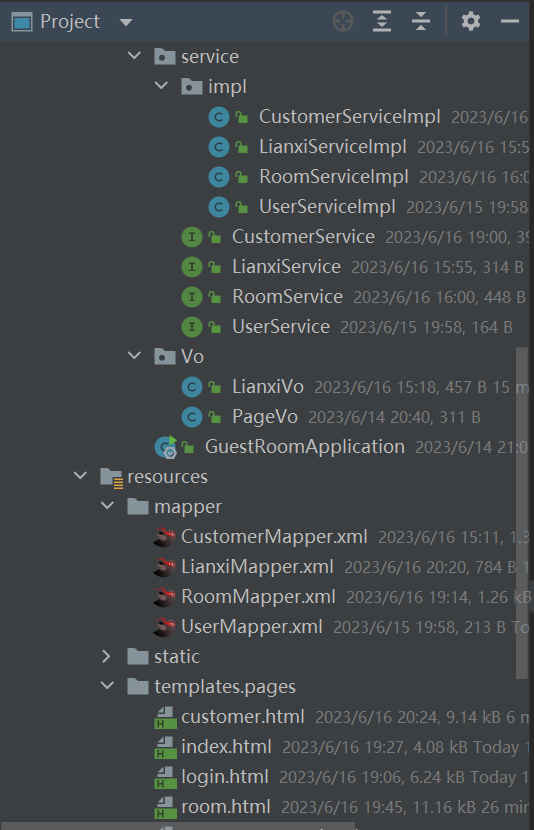
# 酒店客房管理系统系统实现

# 系统功能介绍

客房管理系统实现了管理员的登录和注册功能，实现了对客房的管理，可以模糊查询和分页查询客房的基本信息，可以新增客房信息，可以修改客房信息。实现了对客户的管理，客户信息的模糊查询和分页查询，客户信息的新增以及客户信息的修改，还有客房出租管理，客房可以被出租给用户，记录出租记录和时间，被出租的房间无法再次出租，等到房屋被退房之后才可以再次出租，同时记录用户的退房时间。

# 文件组织结构





# 数据库设计

表3.1 user表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 备注（含主、外键等约束） |
| id | 编号 | int | 主键 |
| username | 账号 | varchar |  |
| password | 密码 | varchar |  |

表3.2 customer表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 备注（含主、外键等约束） |
| id | 编号 | int | 主键 |
| name | 姓名 | varchar | 唯一约束 |
| sex | 性别 | varchar |  |
| age | 年龄 | int |  |
| phone | 电话 | varchar |  |
| card | 身份证 | varchar |  |

表3.3 room表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 备注（含主、外键等约束） |
| id | 编号 | int | 主键 |
| roomname | 房间名称 | varchar |  |
| type | 房间类型 | varchar |  |
| size | 房间大小 | varchar |  |
| state | 房间状态 | int |  |

表3.4 lianxi表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 说明 | 类型 | 备注（含主、外键等约束） |
| id | 编号 | int | 主键 |
| cid | 客户编号 | int | 外键 |
| rid | 房间编号 | int | 外键 |
| inTime | 入住时间 | datetime |  |
| outTime | 退房时间 | datetime |  |

# 功能实现

#### 4.1系统用到的主要配置文件

（1）application.yaml文件内容

|  |
| --- |
| server:  port: 8080 spring:  application:  name: guestroom  datasource:  username: root  password: 123456  driver-class-name: com.mysql.jdbc.Driver  url: jdbc:mysql://localhost:3306/guestroom?&serverTimezone=GMT%2B8&characterEncoding=utf-8&zeroDateTimeBehavior=convertToNull&useSSL=false  name: druidDataSource  type: com.alibaba.druid.pool.DruidDataSource  mybatis:  type-aliases-package: com.guestroom.entity  mapper-locations: mapper/\*.xml  #MybatisLog4g打印SqL语句  configuration:  log-impl: org.apache.ibatis.logging.stdout.StdOutImpl |

（2）LoginConfiguration.java类内容

|  |
| --- |
| package com.guestroom.config;   import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry; import org.springframework.web.servlet.config.annotation.ViewControllerRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  /\*\*  \* 拦截器配置类  \*/ @Configuration public class LoginConfiguration implements WebMvcConfigurer {  @Override  public void addViewControllers(ViewControllerRegistry registry) {  registry.addViewController("/").setViewName("/pages/login.html");  }   @Override  public void addResourceHandlers(ResourceHandlerRegistry registry) {  registry.addResourceHandler("/static/\*\*").addResourceLocations("classpath:/static/");  registry.addResourceHandler("/templates/\*\*").addResourceLocations("classpath:/templates/");  registry.addResourceHandler("/pages/\*\*").addResourceLocations("classpath:/templates/pages/");   }  } |

#### 4.2 用户管理模块设计（实现登录和注册功能）

4.2.1持久化类（删去getter和setter方法）

|  |
| --- |
| @Data @ToString public class User {  private Integer id;  private String username;  private String password; } |

4.2.2映射器接口文件

|  |
| --- |
| @Mapper public interface UserMapper {  @Select("select \* from user where username=#{username} and password=#{password}")  User login(User user);   @Insert("insert into user(username,password) values(#{username},#{password})")  int register(User user); } |

4.2.3映射文件

|  |
| --- |
| <?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="com.guestroom.dao.UserMapper"> </mapper> |

4.2.4 Service层（可选）

|  |
| --- |
| （1）接口  public interface UserService {  User login(User user);  int register(User user); }  （2）实现类  @Service public class UserServiceImpl implements UserService {  @Autowired  private UserMapper userMapper;   @Override  public User login(User user) {  return userMapper.login(user);  }   @Override  public int register(User user) {  return userMapper.register(user);  } } |

4.2.5控制器类

|  |
| --- |
| @RestController @RequestMapping("user") public class UserController {  @Autowired  private UserService userService;   @RequestMapping("login")  public boolean login(@RequestBody User user){  User user1 = userService.login(user);  return user1==null?false:true;  }   @RequestMapping("register")  public boolean register(@RequestBody User user){  int register;  try {  register = userService.register(user);  }catch (Exception e){  return false;  }  return register==1?true:false;  }   } |

4.2.6视图层

|  |
| --- |
| 提供模块主要运行界面截图 |
|  |

#### 4.3 客房模块设计（模块1）

4.3.1持久化类（删去getter和setter方法）

|  |
| --- |
| @Data @ToString public class Room implements Serializable {   private Integer id;   private String roomname;    private String type;    private String size;   private Integer state;   } |

4.3.2映射器接口文件

|  |
| --- |
| @Mapper public interface RoomMapper{   List<Room> list(PageVo pageVo);   int total(String keyword);   @Select("select \* from room where id=#{id}")  Room getOneById(Integer id);   @Insert("insert into room values(#{id},#{roomname},#{type},#{size},0);")  int add(Room room);   @Update("update room set state=1 where id=#{rid} ")  void updateById(Integer rid);   @Update("update room set state=0 where id=#{rid} ")  void updateByIdOut(Integer rid);   @Update("update room set roomname=#{roomname},type=#{type},size=#{size} where id=#{id} ")  int update(Room room); } |

4.3.3映射文件

|  |
| --- |
| <?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="com.guestroom.dao.RoomMapper">   <resultMap id="BaseResultMap" type="com.guestroom.entity.Room">  <id property="id" column="id" jdbcType="INTEGER"/>  <result property="roomname" column="roomname" jdbcType="VARCHAR"/>  <result property="type" column="type" jdbcType="VARCHAR"/>  <result property="size" column="size" jdbcType="VARCHAR"/>  <result property="state" column="state" jdbcType="INTEGER"/>  </resultMap>  <select id="list" resultType="com.guestroom.entity.Room">  select \* from room  <where>  <if test="keyword!='' and keyword!=null ">  and roomname like "%"#{keyword}"%"  </if>  </where>  limit #{pageNum},#{pageSize}  </select>  <select id="total" resultType="java.lang.Integer">  select count(\*) from room  <where>  <if test="keyword!='' and keyword!=null ">  and roomname like "%"#{keyword}"%"  </if>  </where>  </select> </mapper> |

4.3.4 Service层（可选）

|  |
| --- |
| （1）接口  public interface RoomService{  List<Room> list(PageVo pageVo);   int total(String keyword);   Room getOneById(Integer id);   int add(Room room);   void updateById(Integer rid);   void updateByIdOut(Integer rid);   int update(Room room); }  （2）实现类  @Service public class RoomServiceImpl implements RoomService {  @Autowired  private RoomMapper roomMapper;  @Override  public List<Room> list(PageVo pageVo) {  return roomMapper.list(pageVo);  }   @Override  public int total(String keyword) {  return roomMapper.total(keyword);  }   @Override  public Room getOneById(Integer id) {  return roomMapper.getOneById(id);  }   @Override  public int add(Room room) {  return roomMapper.add(room);  }   @Override  public void updateById(Integer rid) {  roomMapper.updateById(rid);  }   @Override  public void updateByIdOut(Integer rid) {  roomMapper.updateByIdOut(rid);  }   @Override  public int update(Room room) {  return roomMapper.update(room);  } } |

4.3.5控制器类

|  |
| --- |
| @RestController @RequestMapping("pages/room") public class RoomController {  @Autowired  private RoomService roomService;   @RequestMapping("list")  public Map<String,Object> list(@RequestBody PageVo pageVo){  pageVo.setPageNum( ( pageVo.getPageNum() -1 ) \* pageVo.getPageSize() );   List<Room> list = roomService.list(pageVo);  int total = roomService.total(pageVo.getKeyword());   Map<String,Object> map = new HashMap<>();  map.put("list",list);  map.put("total",total);  return map;  }   //新增客房  @RequestMapping("add")  public boolean add(@RequestBody Room room){  return roomService.add(room)==1?true:false;  }   //修改客房  @RequestMapping("update")  public boolean update(@RequestBody Room room){  return roomService.update(room)==1?true:false;  }  } |

4.3.6视图层

|  |
| --- |
| 提供模块主要运行界面截图 |
|  |

#### 4.4 客户模块设计（模块2）

4.4.1持久化类（删去getter和setter方法）

|  |
| --- |
| @Data @ToString public class Customer {  private Integer id;  private String name;  private String sex;  private Integer age;  private String phone;  private String card;  } |

4.3.2映射器接口文件

|  |
| --- |
| @Mapper public interface CustomerMapper{   List<Customer> list(PageVo pageVo);   int total(String keyword);   @Select("select \* from customer where id=#{id}")  Customer getOneById(Integer id);   @Insert("insert into customer values(null,#{name},#{sex},#{age},#{phone},#{card});")  int add(Customer customer);   @Update("update customer set name=#{name},sex=#{sex},age=#{age},phone=#{phone},card=#{card} where id=#{id} ;")  int update(Customer customer);   @Select("select \* from customer")  List<Customer> getAll(); } |

4.4.3映射文件

|  |
| --- |
| <?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="com.guestroom.dao.CustomerMapper">   <resultMap id="BaseResultMap" type="com.guestroom.entity.Customer">  <id property="id" column="id" jdbcType="INTEGER"/>  <result property="name" column="name" jdbcType="VARCHAR"/>  <result property="sex" column="sex" jdbcType="VARCHAR"/>  <result property="age" column="age" jdbcType="INTEGER"/>  <result property="phone" column="phone" jdbcType="VARCHAR"/>  <result property="card" column="card" jdbcType="VARCHAR"/>  </resultMap>   <select id="list" resultType="com.guestroom.entity.Customer">  select \* from customer  <where>  <if test="keyword!='' and keyword!=null ">  and name like "%"#{keyword}"%"  </if>  </where>  limit #{pageNum},#{pageSize}  </select>  <select id="total" resultType="java.lang.Integer">  select count(\*) from customer  <where>  <if test="keyword!='' and keyword!=null ">  and name like "%"#{keyword}"%"  </if>  </where>  </select> </mapper> |

4.4.4 Service层（可选）

|  |
| --- |
| （1）接口  public interface CustomerService {   List<Customer> list(PageVo pageVo);   int total(String keyword);   Customer getOneById(Integer id);   int add(Customer customer);   int update(Customer customer);   List<Customer> getAll(); }  （2）实现类  @Service public class CustomerServiceImpl implements CustomerService {  @Autowired  private CustomerMapper customerMapper;  @Override  public List<Customer> list(PageVo pageVo) {  return customerMapper.list(pageVo);  }   @Override  public int total(String keyword) {  return customerMapper.total(keyword);  }   @Override  public Customer getOneById(Integer id) {  return customerMapper.getOneById(id);  }   @Override  public int add(Customer customer) {  return customerMapper.add(customer);  }   @Override  public int update(Customer customer) {  return customerMapper.update(customer);  }   @Override  public List<Customer> getAll() {  return customerMapper.getAll();  } } |

4.4.5控制器类

|  |
| --- |
| @RestController @RequestMapping("pages/customer") public class CustomerController {  @Autowired  private CustomerService customerService;   //获取所有用户  @RequestMapping("getAll")  public List<Customer> customers (){  return customerService.getAll();  }    @RequestMapping("list")  public Map<String,Object> list(@RequestBody PageVo pageVo){  pageVo.setPageNum( ( pageVo.getPageNum() -1 ) \* pageVo.getPageSize() );   List<Customer> list = customerService.list(pageVo);  int total = customerService.total(pageVo.getKeyword());   Map<String,Object> map = new HashMap<>();  map.put("list",list);  map.put("total",total);  return map;  }   //新增用户  @RequestMapping("add")  public boolean add(@RequestBody Customer customer){  return customerService.add(customer)==1?true:false;  }   //修改用户  @RequestMapping("update")  public boolean update(@RequestBody Customer customer){  return customerService.update(customer)==1?true:false;  }  } |

4.4.6视图层

|  |
| --- |
| 提供模块主要运行界面截图 |
|  |

#### 4.5 ×××模块设计（模块3）

4.5.1持久化类（删去getter和setter方法）

|  |
| --- |
| @Data @ToString public class Lianxi{   private Integer id;  private Integer cid;  private Integer rid;   private Date inTime;  private Date outTime;  } |

4.5.2映射器接口文件

|  |
| --- |
| @Mapper public interface LianxiMapper{   List<Lianxi> list(PageVo pageVo);   int total(String keyword);   @Insert("insert into lianxi(cid,rid,inTime) values(#{cid},#{rid},#{inTime});")  int add(Lianxi lianxi);   @Update("update lianxi set outTime = #{outTime} where id=#{id} ;")  int updateById(Lianxi lianxi); } |

4.5.3映射文件

|  |
| --- |
| <?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="com.guestroom.dao.LianxiMapper">   <resultMap id="BaseResultMap" type="com.guestroom.entity.Lianxi">  <id property="id" column="id" jdbcType="INTEGER"/>  <result property="cid" column="cid" jdbcType="INTEGER"/>  <result property="rid" column="rid" jdbcType="INTEGER"/>  </resultMap>   <select id="list" resultType="com.guestroom.entity.Lianxi">  select \* from lianxi  limit #{pageNum},#{pageSize}  </select>  <select id="total" resultType="java.lang.Integer">  select count(\*) from lianxi  </select> </mapper> |

4.5.4 Service层（可选）

|  |
| --- |
| （1）接口  public interface LianxiService{   List<Lianxi> list(PageVo pageVo);   int total(String keyword);   int add(Lianxi lianxi);   int updateById(Lianxi lianxi); }  实现类  @Service public class LianxiServiceImpl implements LianxiService {  @Autowired  private LianxiMapper lianxiMapper;   @Override  public List<Lianxi> list(PageVo pageVo) {  return lianxiMapper.list(pageVo);  }   @Override  public int total(String keyword) {  return lianxiMapper.total(keyword);  }   @Override  public int add(Lianxi lianxi) {  return lianxiMapper.add(lianxi);  }   @Override  public int updateById(Lianxi lianxi) {  return lianxiMapper.updateById(lianxi);  } } |

4.5.5控制器类

|  |
| --- |
| @RestController @RequestMapping("pages/lianxi") public class LianxiController {  @Autowired  private LianxiService lianxiService;  @Autowired  private CustomerService customerService;  @Autowired  private RoomService roomService;  @RequestMapping("list")  public Map<String,Object> list(@RequestBody PageVo pageVo){  pageVo.setPageNum( ( pageVo.getPageNum() -1 ) \* pageVo.getPageSize() );   List<Lianxi> list = lianxiService.list(pageVo);  int total = lianxiService.total(pageVo.getKeyword());   //根据list封装出VO  List<LianxiVo> voList = new ArrayList<>();  for (Lianxi lianxi : list) {  LianxiVo lianxiVo = new LianxiVo();  //拷贝联系  BeanUtils.copyProperties(lianxi,lianxiVo);  //拷贝用户,忽略id  BeanUtils.copyProperties(customerService.getOneById(lianxi.getCid()),lianxiVo,"id");  //拷贝客房,忽略id  BeanUtils.copyProperties(roomService.getOneById(lianxiVo.getRid()),lianxiVo,"id");  if(lianxiVo!=null) voList.add(lianxiVo);  }   Map<String,Object> map = new HashMap<>();  map.put("list",voList);  map.put("total",total);  return map;  }    //入住  @Transactional  @RequestMapping("add")  public boolean add(@RequestBody Lianxi lianxi){  //修改房屋状态  roomService.updateById(lianxi.getRid());  //新增联系  //只有cid和rid,现在设置入住时间  lianxi.setInTime(new Date());  return lianxiService.add(lianxi)==1?true:false;  }    //退房  @Transactional  @RequestMapping("update")  public boolean update(@RequestBody Lianxi lianxi){  System.out.println(lianxi);   //修改房屋状态  roomService.updateByIdOut(lianxi.getRid());  //修改联系  //设置退房时间  lianxi.setOutTime(new Date());  return lianxiService.updateById(lianxi)==1?true:false;  }  } |

4.5.6视图层

|  |
| --- |
| 提供模块主要运行界面截图 |
|  |

# 总结

|  |
| --- |
| 阐述当前实现系统的主要功能、不足及下一步的工作思路。 |
| 当前系统的主要功能有客房信息管理，客户信息管理以及入住退房信息管理，实现了用户，客户，以及客户入住退房等功能，缺点是用户入住房间后无法去设置其他时间段的出租，这样其实是不好的，比如客户今天入住，我今天其实还可以出租明天的房间，这是系统存在的一个很大的问题，无法做到客房预约的功能。 |

# 总结

|  |  |
| --- | --- |
| 评分 |  |