上海电力大学

数据库原理课程设计



**题　　目: 选题七 民航订票管理系统**

**学生姓名： 俞俊伟**

**学 号： 20170026 班 级： 2017221**

**院　　系：　　 计算机科学与技术学院**

**专业班级：　 软件工程**

**2020 年 1 月 9 日**

2. **需求分析**

系统功能是在实际开发设计过程中经过调研、分析用户需求，和用户一起共同确定下来，是系统为满足用户需求所应完成的功能。本课程设计模拟一个小型民航定票管理系统。

航空客运订票的业务活动包括；查询航线、客票预订和办理退票等。试设计一个航空客运订票系统，以使上述业务可以借助计算机来完成。

包含信息如下：

（1）航空公司信息，如南方航空、东方航空。

（2）每条航线所涉及的信息有：起点、终点 、航班号、飞机型号、起飞时间、到达时间、乘员定额（头等舱、经济舱）、里程、属于哪个航空公司、标准价格等。

（3）每种飞机型号（如波音747）都有固定的座位表。

（4）每个航班涉及的信息有：航班号、日期、 已订票的客户名单及仓位（头等舱、经济舱）。

（5）顾客信息及顾客购买机票的信息。

（2）系统能实现的操作和功能如下：

①查询航线：根据旅客提出的起点、终点站名及航班日期输出下列信息：航班号、飞机型号、余票额、价格等。机票的价格是浮动的，请自行确定一个价格浮动策略。

②承办订票业务：根据客户提出的要求（航班号、日期）查询该航班票额情况，若尚有余票，则为客户办理订票手续。顾客能自动按航空公司积累已飞行的里程，里程达到一定的数量，可以升级为银卡、金卡等客户，购票时可以享受不同的打折优惠。

③承办退票业务：根据客户提供的情况（日期、航班），为客户办理退票手续。

说明：请参考携程网的机票购买功能。

1. **概念结构设计**

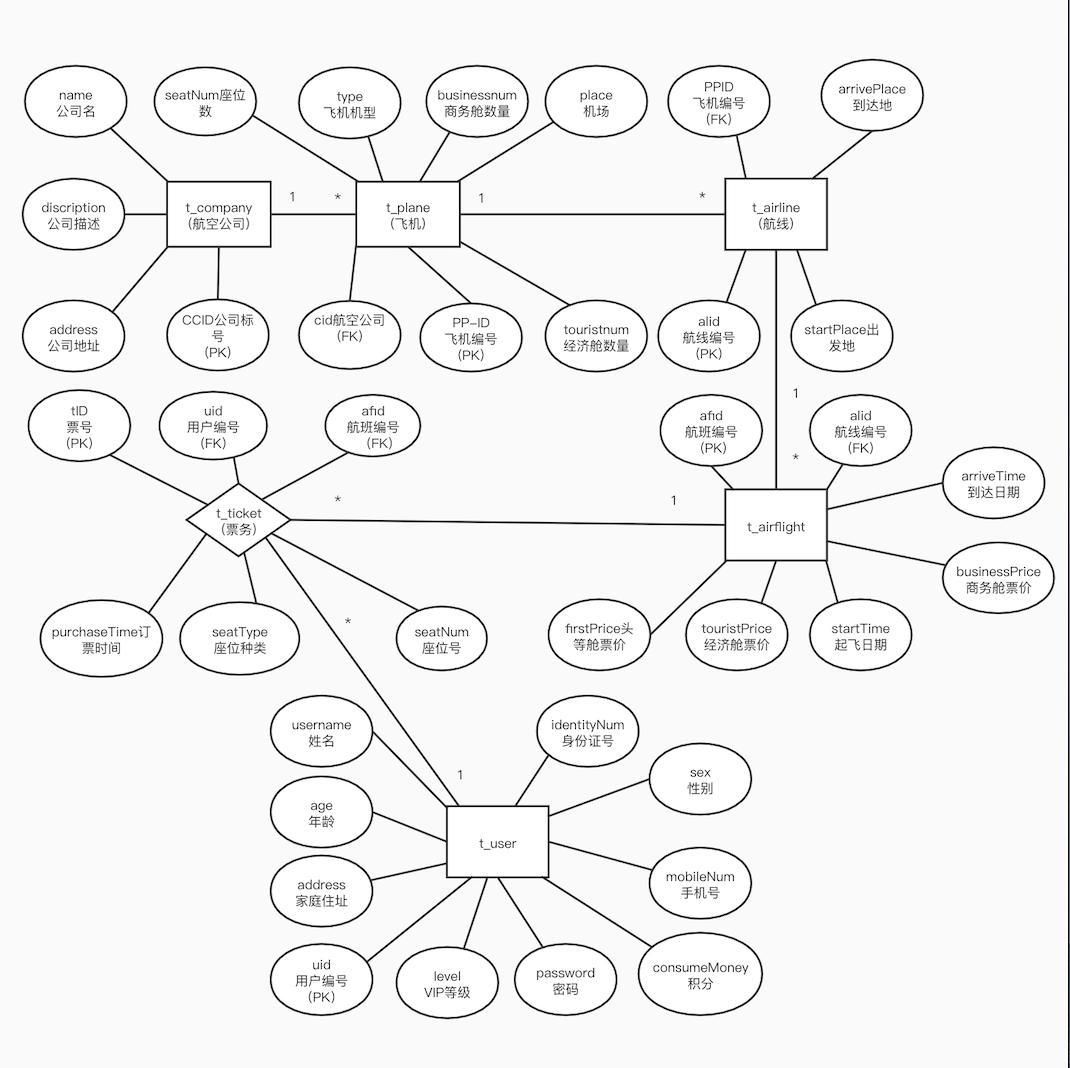
本系统主要的架构如下：

数据库：Micro Soft SQL Server 2017

后台：SpringBoot，Lombok，Log4j，Spring，SpringMVC，Spring-aop，Spring-starter-web

前端：thymeleaf，Semantic-UI

调试工具：IntelJ IDEA，Spring-dev-tools



航天公司(t\_company): ccid,address,discription,name 主键ccid

飞机(t\_plane): ppid,type,businessnum,place,touristnum,seat\_num,ccid 主键: ppid 外键: ccid

航线(t\_airline): alid,start\_place,arrive\_place,ppid 主键:alid 外键:ppid

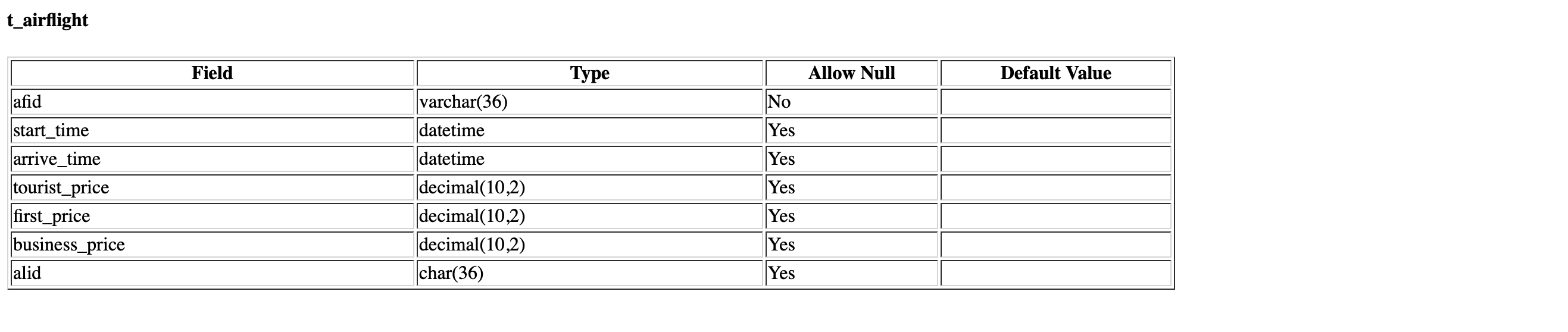
航班(t\_airflight): afid,first\_price,start\_time,business\_price,arrive\_time,alid 主键: afid 外键: alid

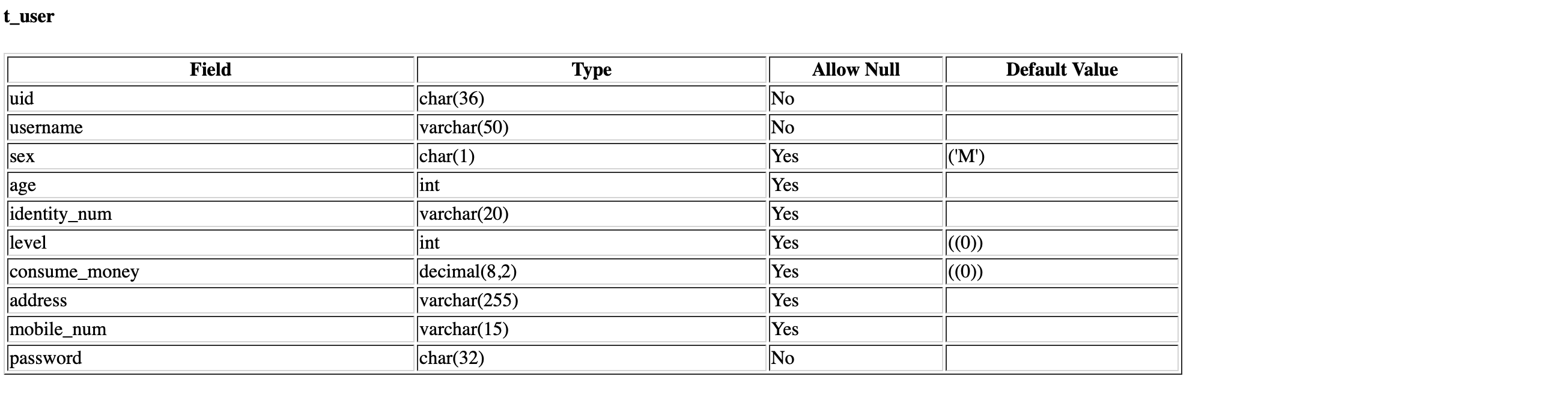
票务(t\_ticket): tid,purchase\_time,seat\_type,afid,uuid 主键:tid 外键:afid,uuid

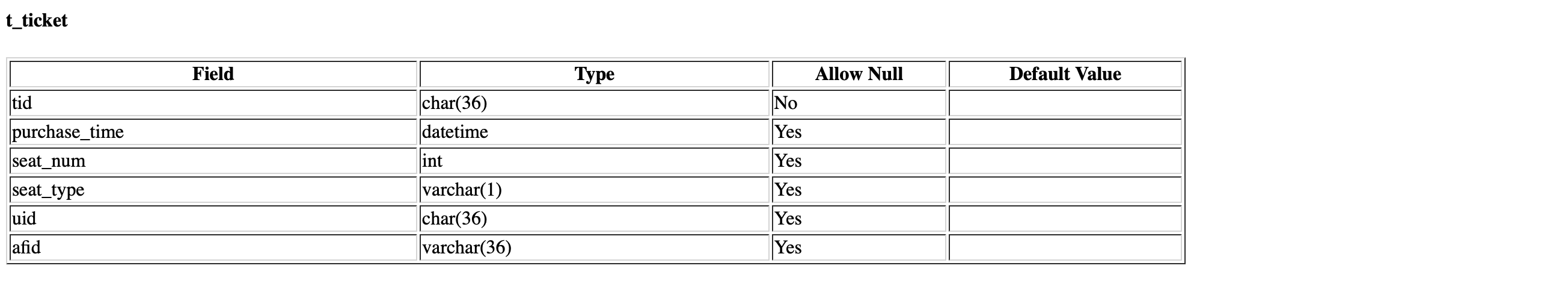
用户(t\_user): uid,username,age,address,identityNum,sex,mobileNum,password,level,password 主键:uid

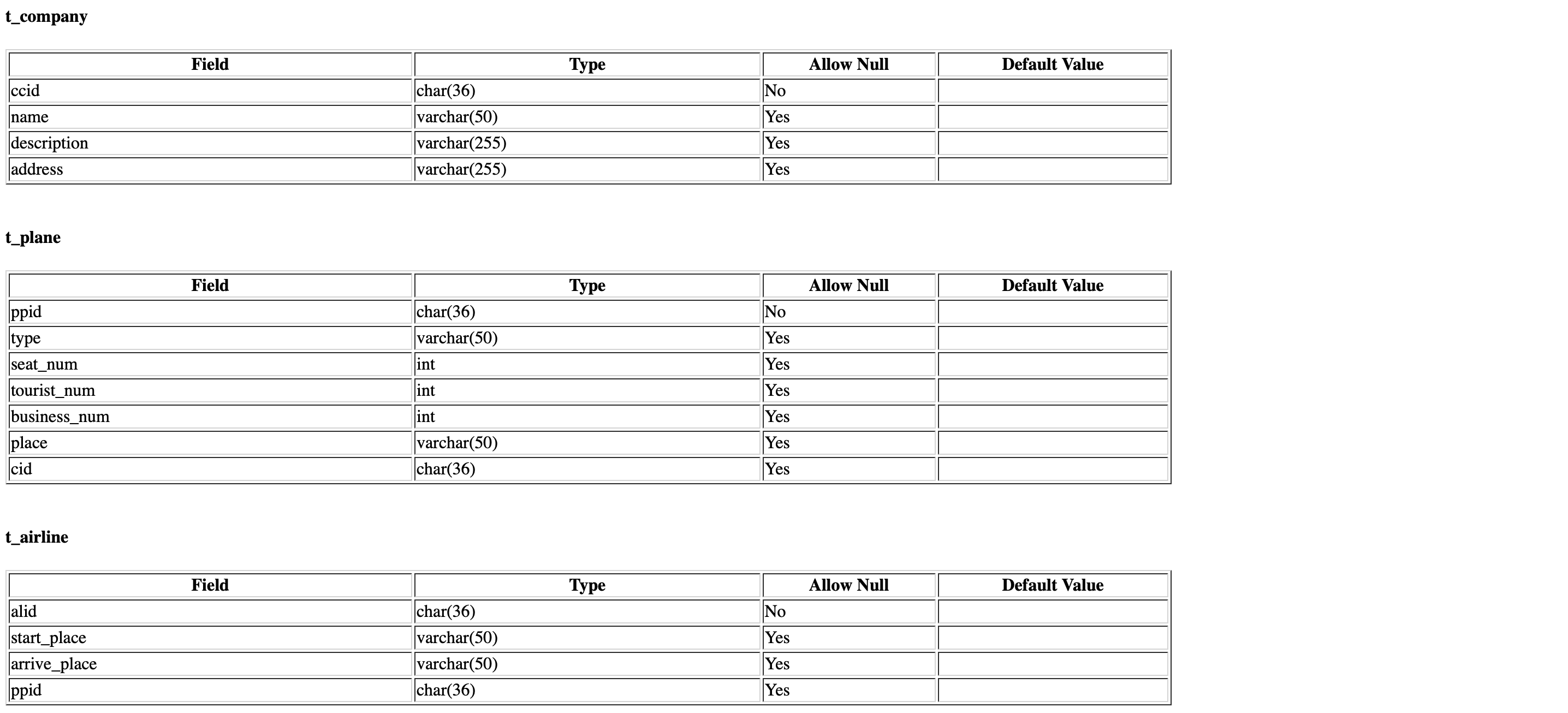
**三、逻辑结构设计**

**（**1）模式设计：按系统整体E-R模型，写出关系模式；并利用数据字典加以描述。（如每个关系模式有何属性、属性的类型、属性值的长度、是否可取空值、是否为主码、有何约束条件等信息）









**四、数据库的物理设计**

为了提高在表中搜索元组的速度，在实际实现的时候应该基于某些属性建立索引。给出所建立的索引。（包括按哪些表中哪些属性按升序还是降序来创建索引及相应的T-SQL的创建语句）。

本系统中使用频繁的我们也设计了索引，见sql语句

1. **数据库设计实现及运行**

（1）数据库的创建

T-SQL语句：

create database airplane-ticket-system;

（2）数据表的创建（可同时在创建时定义相关的约束）

T-SQL语句：

**DROP TABLE IF EXISTS** t\_user;  
**CREATE TABLE** t\_user(  
 **[uid] CHAR**(36) **NOT NULL PRIMARY KEY**,  
 **[username] VARCHAR**(50) **not null**,  
 **[sex] CHAR**(1) **default 'M' check** (**[sex] in** (**'M'**,**'F'**)),  
 **[age] INT**,  
 **[identity\_num] VARCHAR**(20),  
 **[level] INT default** 0,  
 **[consume\_money] decimal**(8,2) **default** 0,  
 **[address] VARCHAR**(255),  
 **[mobile\_num] VARCHAR**(15),  
 **[password] CHAR**(32) **NOT NULL**,  
);  
**DROP TABLE IF EXISTS** t\_company;  
**CREATE TABLE** t\_company(  
 **ccid CHAR**(36) **NOT NULL PRIMARY KEY**,  
 **[name] VARCHAR**(50),  
 **[description] VARCHAR**(255),  
 **[address] VARCHAR**(255)  
)  
**DROP TABLE IF EXISTS** t\_plane;  
**CREATE TABLE** t\_plane(  
 **ppid CHAR**(36) **NOT NULL PRIMARY KEY**,  
 **[type] VARCHAR**(50),  
 **seat\_num INT**,  
 **tourist\_num INT**,  
 **business\_num INT**,  
 **place VARCHAR**(50),  
 **cid CHAR**(36) **FOREIGN KEY REFERENCES** t\_company(**ccid**)  
)  
**DROP TABLE IF EXISTS** t\_airline;  
**CREATE TABLE** t\_airline(  
 **alid CHAR**(36) **NOT NULL PRIMARY KEY**,  
 **startPlace VARCHAR**(50),  
 **arrivePlace VARCHAR**(50),  
 **ppid CHAR**(36) **FOREIGN KEY REFERENCES** t\_plane(**ppid**)  
)  
**DROP TABLE IF EXISTS** t\_airflight;  
**CREATE TABLE** t\_airflight(  
 **afid VARCHAR** (36) **NOT NULL PRIMARY KEY**,  
 **start\_time DATETIME**,  
 **arrive\_time DATETIME**,  
 **tourist\_price DECIMAL**(10,2),  
 **first\_price DECIMAL**(10,2),  
 **business\_price DECIMAL**(10,2),  
 **alid CHAR**(36) **FOREIGN KEY REFERENCES** t\_airline(**alid**)  
)  
**ALTER TABLE** t\_airflight **ADD CONSTRAINT** ck\_airflight **check** (**afid like 'G[0-9][0-9][0-9][0-9]'**)  
**DROP TABLE IF EXISTS** t\_ticket;  
**CREATE TABLE** t\_ticket(  
 **tid CHAR**(36) **NOT NULL PRIMARY KEY**,  
 **purchase\_time DATETIME**,  
 **seat\_num INT**,  
 **seat\_type VARCHAR**(1),  
 **[uid] CHAR**(36) **FOREIGN KEY REFERENCES** t\_user(**[uid]**),  
 **afid VARCHAR** (36) **FOREIGN KEY REFERENCES** t\_airflight(**afid**)  
 );

（3）索引的创建

**CREATE INDEX** plane\_index **ON** t\_plane(**seat\_num**,**tourist\_num**,**business\_num**)

**create index** airline\_index **on** t\_airline(**start\_place**,**arrive\_place**)

**create index** airflight\_index **on** t\_airflight(**start\_time**,**arrive\_time**)

**CREATE INDEX** user\_index **ON** t\_user(**username**)

1. 实图的创建

**CREATE VIEW** today\_ticket\_simple **as  
 select** today\_airflight.**afid** afid,  
 T.**start\_place** start\_place,  
 T.**arrive\_place** arrive\_place,  
 today\_airflight.**start\_time** start\_time,  
 today\_airflight.**arrive\_time** arrive\_time  
 **from** (**select afid**,**alid**,**start\_time**,**arrive\_time from** t\_airflight **where** *DATEDIFF*(**dd**, **start\_time**, *GETDATE*())=0) today\_airflight,t\_airline T  
 **where** today\_airflight.**alid** = T.**alid**

1. 存储过程

买票：

**create procedure** *purchase* @tid **char**(36),  
 @purchase\_time **datetime**,  
 @seat\_num **int**,  
 @seat\_type **varchar**(50),  
 @uid **char**(36),  
 @afid **char**(36)  
**AS  
 BEGIN  
 insert** t\_ticket **values** (@tid,@purchase\_time,@seat\_num,@seat\_type,@uid,@afid)  
 **end**

机票价格浮动策略:

**create procedure** *price\_add* @percent **decimal**(5,2),  
 @afid **varchar**(50),  
 @type **char**(1)  
**AS  
 begin  
 if**(@type=**'h'**)  
 **begin  
 update** t\_airflight **set first\_price**=**first\_price**\*@percent **where afid**=@afid  
 **end  
 if**(@type=**'b'**)  
 **begin  
 update** t\_airflight **set business\_price**=**business\_price**\*@percent **where afid**=@afid  
 **end  
 if**(@type=**'t'**)  
 **begin  
 update** t\_airflight **set tourist\_price**=**tourist\_price**\*@percent **where afid**=@afid  
 **end  
 end**

1. 触发器

VIP等级触发器

**create trigger** vipTrigger **on** t\_user  
 **for update  
 as  
 declare** @uid **char**(36),@money **decimal**(8,2),@init\_money **decimal**(8,2)  
 **select** @uid=uid,@money=consume\_money **from** inserted  
 **update** t\_user **set level**=(@money)/1000 **where uid**=@uid

1. JDBC运用到的SQL CRUD语句

增加航班

*/\*\*  
 \* Add Airflight  
 \** ***@param airflight*** *\*/***public void** addAirflight(Airflight airflight){  
 Connection conn = DBUtils.*getConnection*();  
 String sql = **"insert t\_airflight values(?,?,?,?,?,?,?)"**;  
 PreparedStatement prep = DBUtils.*getPreparedStatement*(conn,sql);  
 **try** {  
 prep.setString(1,airflight.getAfId());  
 prep.setObject(2,airflight.getStartTime());  
 prep.setObject(3,airflight.getArriveTime());  
 prep.setDouble(4,airflight.getTouristPrice());  
 prep.setDouble(5,airflight.getFirstPrice());  
 prep.setDouble(6,airflight.getBusinessPrice());  
 prep.setString(7,airflight.getAlid());  
 prep.execute();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
}

查找航班个数  
*/\*\*  
 \* count the airflight for the unique algorithm  
 \** ***@param afid*** *\** ***@return*** *\*/***public int** countAirflight(String afid){  
 Connection conn = DBUtils.*getConnection*();  
 String sql = **"select** *count***(afid) from t\_airflight where afid = '"** + afid +**"'"**;  
 **int** count = 0;  
 **try** {  
 PreparedStatement prep = conn.prepareStatement(sql);  
 ResultSet rs = prep.executeQuery();  
 rs.next();  
 count = rs.getInt(1);  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** count;  
}  
计算当天航班个数  
*/\*\*  
 \* caculate intraday count of airflight  
 \** ***@return*** *\*/***public int** countAirflight(){  
 Connection conn = DBUtils.*getConnection*();  
 String sql = **"select** *count***(afid) from t\_airflight where DATEDIFF(dd, start\_time, GETDATE())=0"**;  
 **int** count = 0;  
 **try** {  
 PreparedStatement prep = conn.prepareStatement(sql);  
 ResultSet rs = prep.executeQuery();  
 rs.next();  
 count = rs.getInt(1);  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** count;  
}

增加航线  
 */\*\*  
 \* add airline method  
 \** ***@param airline*** *\*/* **public void** addAirline(Airline airline){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*execute*(stmt,**"insert t\_airline values('"**+  
 airline.getAlid()+**"','"**+  
 airline.getStartPlace()+**"','"**+  
 airline.getArrivePlace()+**"','"**+  
 airline.getPPID()+**"')"**);  
 }  
根据开始地点，最终地点获取航线  
 */\*\** ***\* get airline method  
 \* @param startPlace start place  
 \* @param arrivePlace arrive place  
 \* @return  
 \*/*****public** Airline getAirline(String startPlace,String arrivePlace){  
 Airline airline = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select \* from t\_airline where start\_place = '"** +  
 startPlace + **"' and arrive\_place='"**+arrivePlace+**"'"**);  
 **try**{  
 **while** (rs.next()){  
 airline = **new** Airline();  
 airline.setAlid(rs.getString(**"alid"**)).  
 setPPID(rs.getString(**"ppid"**)).  
 setStartPlace(rs.getString(**"start\_place"**)).  
 setArrivePlace(rs.getString(**"arrive\_place"**));  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** airline;  
 }  
根据地点获取所有航线

*/\*\*  
 \* get airlines  
 \** ***@param place*** *the start place or arrive place  
 \** ***@return*** *the arraylist contains airlines  
 \*/* **public** ArrayList<Airline> getAirline(String place){  
 ArrayList<Airline> arr = **new** ArrayList<>();  
 Airline airline = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select \* from t\_airline where start\_place = '"** +  
 place + **"'"**);  
 **try**{  
 **while** (rs.next()){  
 airline = **new** Airline();  
 airline.setAlid(rs.getString(**"alid"**)).  
 setPPID(rs.getString(**"ppid"**)).  
 setStartPlace(rs.getString(**"start\_place"**)).  
 setArrivePlace(rs.getString(**"arrive\_place"**));  
 arr.add(airline);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 rs = DBUtils.*executeQuery*(stmt, **"select \* from t\_airline where arrive\_place = '"** +  
 place + **"'"**);  
 **try**{  
 **while** (rs.next()){  
 airline = **new** Airline();  
 airline.setAlid(rs.getString(**"alid"**)).  
 setPPID(rs.getString(**"ppid"**)).  
 setStartPlace(rs.getString(**"start\_place"**)).  
 setArrivePlace(rs.getString(**"arrive\_place"**));  
 arr.add(airline);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** arr;  
 }  
}

增加公司

*/\*\*  
 \* add company  
 \** ***@param company*** *\*/***public void** addCompany(Company company){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*execute*(stmt,**"insert t\_company values ('"** + company.getCCID() +**"','"** +company.getName()+**"','"** +company.getDescription()+**"','"** +company.getAddress()+**"');"**);  
}

通过名字查找公司  
*/\*\*  
 \* get company by name  
 \** ***@param name*** *\** ***@return*** *the company object  
 \*/***public** Company getCompanyByName(String name){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 Company company = **null**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select \* from t\_company where name = '"** + name + **"'"**);  
 **try**{  
 **while** (rs.next()){  
 company = **new** Company(rs.getString(**"ccid"**),  
 rs.getString(**"name"**),  
 rs.getString(**"description"**),  
 rs.getString(**"address"**));  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 **return** company;  
}

增加飞机

*/\*\*  
 \* add plane  
 \** ***@param plane*** *\*/***public void** addPlane(Plane plane){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*execute*(stmt,**"insert t\_plane values('"**+plane.getPPID()+**"',"** +  
 **"'"**+plane.getType()+**"',"**+plane.getSeatNum()+**","** +  
 plane.getTouristNum()+**","**+  
 plane.getBusinessnum()+**",'"**+  
 plane.getPlace()+**"','"**+  
 plane.getCid()+**"')"**);  
}  
查找飞机  
*/\*\*  
 \* get plane  
 \** ***@param place*** *\** ***@return*** *\*/***public** Plane getPlane(String place){  
 Plane plane = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select \* from t\_plane where place = '"** + place + **"'"**);  
 **try**{  
 **while** (rs.next()){  
 plane = **new** Plane();  
 plane.setPPID(rs.getString(**"ppid"**)).  
 setType(rs.getString(**"type"**)).  
 setSeatNum(rs.getInt(**"seat\_num"**)).  
 setTouristNum(rs.getInt(**"tourist\_num"**)).  
 setBusinessnum(rs.getInt(**"business\_num"**)).  
 setPlace(rs.getString(**"place"**)).  
 setCid(rs.getString(**"cid"**));  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** plane;  
}  
通过afid查找飞机  
*/\*\*  
 \* get plane by air flight  
 \** ***@param afid*** *\** ***@return*** *\*/***public** Plane getPlaneByAfid(String afid){  
 Plane plane = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select \* from (\n"** +  
 **" select ppid\n"** +  
 **" from (\n"** +  
 **" select alid\n"** +  
 **" from t\_airflight\n"** +  
 **" where afid = '"**+  
 afid+**"'\n"** +  
 **" ) A,\n"** +  
 **" t\_airline\n"** +  
 **" where t\_airline.alid = A.alid\n"** +  
 **" ) B,t\_plane where B.ppid=t\_plane.ppid"**);  
 **try**{  
 **while** (rs.next()){  
 plane = **new** Plane();  
 plane.setPPID(rs.getString(**"ppid"**)).  
 setType(rs.getString(**"type"**)).  
 setSeatNum(rs.getInt(**"seat\_num"**)).  
 setTouristNum(rs.getInt(**"tourist\_num"**)).  
 setBusinessnum(rs.getInt(**"business\_num"**)).  
 setPlace(rs.getString(**"place"**)).  
 setCid(rs.getString(**"cid"**));  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** plane;  
}

通过afid查找ticket detail细节

*/\*\*  
 \* get ticket detail by afid  
 \** ***@param AfId*** *\** ***@return*** *\*/***public** TicketDetails getTicketDetailByAfId(String AfId){  
 TicketDetails ticketDetails = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt,  
 **"select afid,t\_airline.alid,start\_time,arrive\_time,\n"** +  
 **" tourist\_price,first\_price,business\_price,start\_place,arrive\_place from\n"** +  
 **"(select \* from t\_airflight where afid = '"** + AfId + **"') info,t\_airline\n"** +  
 **"where info.alid=t\_airline.alid"**);  
 **try**{  
 **while** (rs.next()){  
 ticketDetails = **new** TicketDetails();  
 ticketDetails.setAfid(rs.getString(**"afid"**)).  
 setAlid(rs.getString(**"alid"**)).  
 setStartTime(rs.getString(**"start\_time"**).substring(0,16)).  
 setArriveTime(rs.getString(**"arrive\_time"**).substring(0,16)).  
 setTouristPrice(rs.getDouble(**"tourist\_price"**)).  
 setFirstPrice(rs.getDouble(**"first\_price"**)).  
 setBusinessPrice(rs.getDouble(**"business\_price"**)).  
 setStartPlace(rs.getString(**"start\_place"**)).  
 setArrivePlace(rs.getString(**"arrive\_place"**));  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 **return** ticketDetails;  
}  
得到ticket的数量  
*/\*\*  
 \* get ticket count  
 \** ***@param seatType*** *\** ***@param afid*** *\** ***@return*** *\*/***public int** getTicketCount(String seatType,String afid){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 ResultSet rs = DBUtils.*executeQuery*(stmt, **"select count(tid) from t\_ticket where seat\_type='"**+  
 seatType+**"' and afid='"**+  
 afid+**"'"**);  
 **try**{  
 rs.next();  
 **return** rs.getInt(1);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** 0;  
}

增加票务

*/\*\*  
 \* add ticket  
 \** ***@param ticket*** *\*/***public void** addTicket(Ticket ticket){  
 Connection conn = DBUtils.*getConnection*();  
 String sql = **"insert t\_ticket values (?,?,?,?,?,?)"**;  
 PreparedStatement ps =DBUtils.*getPreparedStatement*(conn,sql);  
 **try** {  
 ps.setString(1,ticket.getTID());  
 ps.setObject(2, DateUtils.*getString*(ticket.getPurchaseTime()));  
 ps.setInt(3,ticket.getSeatNum());  
 ps.setString(4,ticket.getSeatType());  
 ps.setString(5,ticket.getUid());  
 ps.setString(6,ticket.getAfid());  
 ps.execute();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
}  
取出票务  
*/\*\*  
 \* get ticket num  
 \** ***@param afid*** *\** ***@return*** *\*/***public** Integer getTicketNum(String afid) {  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql =**"select count(tid) from t\_ticket where afid='"**+afid+**"'"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 rs.next();  
 **return** rs.getInt(1);  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** 0;  
}  
通过uid和afid取票  
*/\*\*  
 \* get ticket by uid and Afid  
 \** ***@param uid*** *\** ***@param afid*** *\** ***@return*** *\*/***public** Ticket searchTicketByUidAndAfid(String uid, String afid) {  
 Ticket ticket = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql =**"select \* from t\_ticket where t\_ticket.uid='"**+uid+**"' and t\_ticket.afid='"**+afid+**"'"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 **while** (rs.next()){  
 ticket = **new** Ticket();  
 ticket.setTID(rs.getString(**"tid"**)).  
 setPurchaseTime(rs.getTime(**"purchase\_time"**)).  
 setSeatNum(rs.getInt(**"seat\_num"**)).setSeatType(rs.getString(**"seat\_type"**)).  
 setUid(rs.getString(**"uid"**)).setAfid(rs.getString(**"afid"**));  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** ticket;  
}  
得到票务

*/\*\*  
 \* get ticket  
 \** ***@param uid*** *\** ***@return*** *\*/***public** List<Ticket> getTicketByUid(String uid) {  
 List<Ticket> res = **new** ArrayList<>();  
 Ticket ticket = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql =**"select \* from t\_ticket where t\_ticket.uid='"**+uid+**"'"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 **while** (rs.next()){  
 ticket = **new** Ticket();  
 ticket.setTID(rs.getString(**"tid"**)).  
 setPurchaseTime(rs.getDate(**"purchase\_time"**)).  
 setSeatNum(rs.getInt(**"seat\_num"**)).setSeatType(rs.getString(**"seat\_type"**)).  
 setUid(rs.getString(**"uid"**)).setAfid(rs.getString(**"afid"**));  
 res.add(ticket);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** res;  
}  
通过tid获取票务

*/\*\*  
 \* get ticket by tid  
 \** ***@param tid*** *\** ***@return*** *\*/***public** Ticket getTicketByTid(String tid) {  
 Ticket ticket = **null**;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql = **"select \* from t\_ticket where t\_ticket.tid='"** + tid + **"'"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try** {  
 **while** (rs.next()) {  
 ticket = **new** Ticket();  
 ticket.setTID(rs.getString(**"tid"**)).  
 setPurchaseTime(rs.getDate(**"purchase\_time"**)).  
 setSeatNum(rs.getInt(**"seat\_num"**)).setSeatType(rs.getString(**"seat\_type"**)).  
 setUid(rs.getString(**"uid"**)).setAfid(rs.getString(**"afid"**));  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** ticket;  
}  
移除票务  
**public void** removeTicket(String tid) {  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql = **"delete from t\_ticket where t\_ticket.tid='"** + tid + **"'"**;  
 DBUtils.*execute*(stmt,sql);  
}

页面插件  
*/\*\*  
 \* Page Plugin  
 \* caculate page algorithm  
 \** ***@param page*** *\*/***public** ArrayList<TodayTicketSimple> showAllData(Page page){  
 ArrayList<TodayTicketSimple> arr = **new** ArrayList<>();  
 TodayTicketSimple todayTicketSimple;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql = **"select top "** + page.getPageSize() + **" \* from today\_ticket\_simple\n"** +  
 **"where afid not in (\n"** +  
 **"select top "** + page.getNotInTopNumber() + **" afid from today\_ticket\_simple\n"** +  
 **")"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 **while** (rs.next()){  
 todayTicketSimple = **new** TodayTicketSimple(rs.getString(**"afid"**),  
 rs.getString(**"start\_place"**),  
 rs.getString(**"arrive\_place"**),  
 DateUtils.*getDate*(rs.getString(**"start\_time"**)),  
 DateUtils.*getDate*(rs.getString(**"arrive\_time"**)));  
 arr.add(todayTicketSimple);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** arr;  
}  
得到票务通过开始地址和结束地址  
*/\*\*  
 \* get ticket by start place and arrive place  
 \** ***@param startPlace*** *\** ***@param arrivePlace*** *\** ***@return*** *\*/***public** ArrayList<TodayTicketSimple> getTicketByStartPlaceAndArrivePlace(String startPlace,String arrivePlace){  
 ArrayList<TodayTicketSimple> arr = **new** ArrayList<>();  
 TodayTicketSimple todayTicketSimple;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql = **"select \* from today\_ticket\_simple where start\_place='"**+  
 startPlace+**"' and arrive\_place='"**+arrivePlace+**"'"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 **while** (rs.next()){  
 todayTicketSimple = **new** TodayTicketSimple(rs.getString(**"afid"**),  
 rs.getString(**"start\_place"**),  
 rs.getString(**"arrive\_place"**),  
 DateUtils.*getDate*(rs.getString(**"start\_time"**)),  
 DateUtils.*getDate*(rs.getString(**"arrive\_time"**)));  
 arr.add(todayTicketSimple);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** arr;  
}  
得到票务通过开始地址结束地址和日期

*/\*\*  
 \* get ticket by start place and arrive place and date  
 \** ***@param startPlace*** *\** ***@param arrivePlace*** *\** ***@param date*** *\** ***@return*** *\*/***public** ArrayList<TodayTicketSimple> getTicketByStartPlaceAndArrivePlaceAndDate(String startPlace,String arrivePlace,String date){  
 ArrayList<TodayTicketSimple> arr = **new** ArrayList<>();  
 TodayTicketSimple todayTicketSimple;  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 String sql = **"select \* from today\_ticket\_simple where start\_place='"**+startPlace  
 +**"' and arrive\_place='"**+arrivePlace+**"' and DATEDIFF(dd, start\_time, '"**+date+**"')=0"**;  
 ResultSet rs = DBUtils.*executeQuery*(stmt, sql);  
 **try**{  
 **while** (rs.next()){  
 todayTicketSimple = **new** TodayTicketSimple(rs.getString(**"afid"**),  
 rs.getString(**"start\_place"**),  
 rs.getString(**"arrive\_place"**),  
 DateUtils.*getDate*(rs.getString(**"start\_time"**)),  
 DateUtils.*getDate*(rs.getString(**"arrive\_time"**)));  
 arr.add(todayTicketSimple);  
 }  
 }**catch** (SQLException e){  
 e.printStackTrace();  
 }  
 **return** arr;  
}

通过用户名和密码查找user（用于登陆判断）

*/\*\*  
 \* find user by username and password  
 \** ***@param username*** *\** ***@param password*** *\** ***@return*** *\*/***public** User findByUsernameAndPassword(String username, String password){  
 Connection connection = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(connection);  
 **try** {  
 ResultSet rs = stmt.executeQuery(**"select** *\** **from t\_user where username='"** +username+**"' and password='"**+password+**"'"**);  
 **if**(rs.next()){  
 **return new** User(rs.getString(**"uid"**),  
 rs.getString(**"username"**),  
 rs.getString(**"sex"**).charAt(0),  
 rs.getInt(**"age"**),  
 rs.getString(**"identity\_num"**),  
 rs.getInt(**"level"**),  
 rs.getDouble(**"consume\_money"**),  
 rs.getString(**"address"**),  
 rs.getString(**"mobile\_num"**),  
 rs.getString(**"password"**));  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return null**;  
}  
通过用户名查找user  
*/\*\*  
 \* find user by username*

*\** ***@param username*** *\** ***@return*** *\*/***public** User findByUsername(String username){  
 Connection connection = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(connection);  
 **try** {  
 ResultSet rs = stmt.executeQuery(**"select** *\** **from t\_user where username='"** +username+**"'"**);  
 **if**(rs.next()){  
 **return new** User(rs.getString(**"uid"**),  
 rs.getString(**"username"**),  
 rs.getString(**"sex"**).charAt(0),  
 rs.getInt(**"age"**),  
 rs.getString(**"identity\_num"**),  
 rs.getInt(**"level"**),  
 rs.getDouble(**"consume\_money"**),  
 rs.getString(**"address"**),  
 rs.getString(**"mobile\_num"**),  
 rs.getString(**"password"**));  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return null**;  
}  
增加一个新用户（用于注册）  
*/\*\*  
 \* add a new User  
 \** ***@param uid*** *\** ***@param username*** *\** ***@param password*** *\*/***public void** addUser(String uid, String username, String password){  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*execute*(stmt,**"insert t\_user(username,password,uid) values('"**+username+**"','"**+password+**"','"**+uid+**"')"**);  
}  
更新用户信息  
*/\*\*  
 \* 更新User  
 \** ***@param user*** *\*/***public void** updateUser(User user) {  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*executeQuery*(stmt,**"update t\_user set sex='"**+user.getSex()  
 +**"',age="**+user.getAge()  
 +**",identity\_num='"**+user.getIdentityNum()  
 +**"',address='"**+user.getAddress()+**"',mobile\_num='"** +user.getMobileNum()+**"' where uid='"**+user.getUid()+**"'"**);  
}  
更新账户的积分  
*/\*\*  
 \* update money  
 \** ***@param user*** *\*/***public void** updateMoney(User user) {  
 Connection conn = DBUtils.*getConnection*();  
 Statement stmt = DBUtils.*getStatement*(conn);  
 DBUtils.*execute*(stmt,**"update t\_user set consume\_money = "**+user.getConsumeMoney()+**" where uid = '"**+user.getUid()+**"'"**);  
}

**六、系统核心模块以及实现**

1. 登陆功能

首页点击登陆进入登陆界面



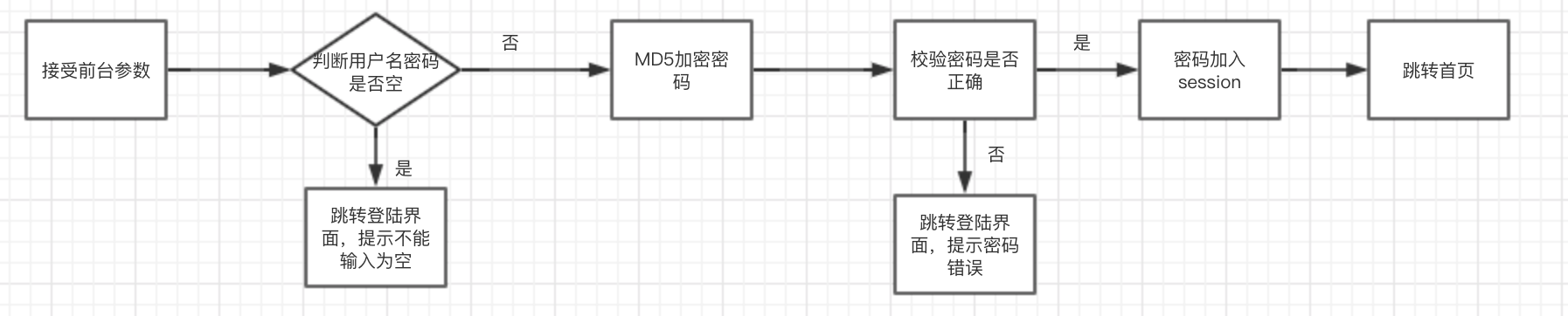
用户名或者密码错误进行提示



密码正确跳转首页



流程图



代码

@PostMapping(**"login/valiLogin"**)  
**public** String login(@RequestParam String username,  
 @RequestParam String password,  
 HttpSession session,  
 RedirectAttributes attributes,  
 Model model) {  
 **if**(username.equals(**""**) || password.equals(**""**)){  
 attributes.addFlashAttribute(**"message"**, **"用户名或密码不能为空"**);  
 **return "redirect:/admin/login"**;  
 }  
 password = MD5Utils.*code*(password);  
 User user = **userService**.findUser(username, password);  
 **if** (user != **null**) {  
 user.setPassword(**null**);  
 session.setMaxInactiveInterval(30\*24\*60\*60);  
 session.setAttribute(**"user"**,user);  
 Page page = **new** Page();  
 page.setPageNumber(**airflightService**.countAirflightService()/10).  
 setPageSize(10).  
 setIndex(0);  
 model.addAttribute(**"airflightcount"**,**airflightService**.countAirflightService());  
 model.addAttribute(**"todayTicket"**,**todayTicketService**.getTodayTicket(page));  
 model.addAttribute(**"page"**,page);  
 **return "/index"**;  
 } **else** {  
 attributes.addFlashAttribute(**"message"**, **"用户名或密码错误"**);  
 **return "redirect:/admin/login"**;  
 }  
}

1. 买票

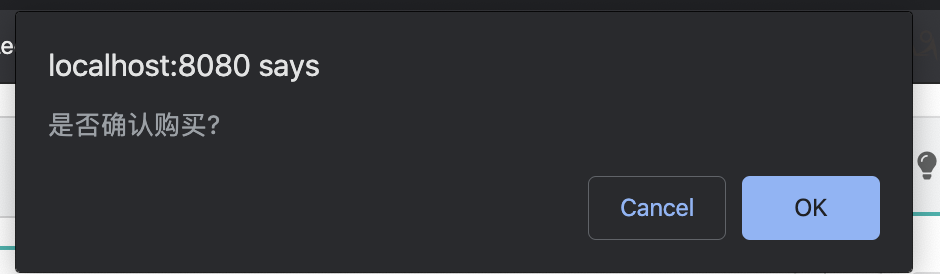
点击购买进入详情界面



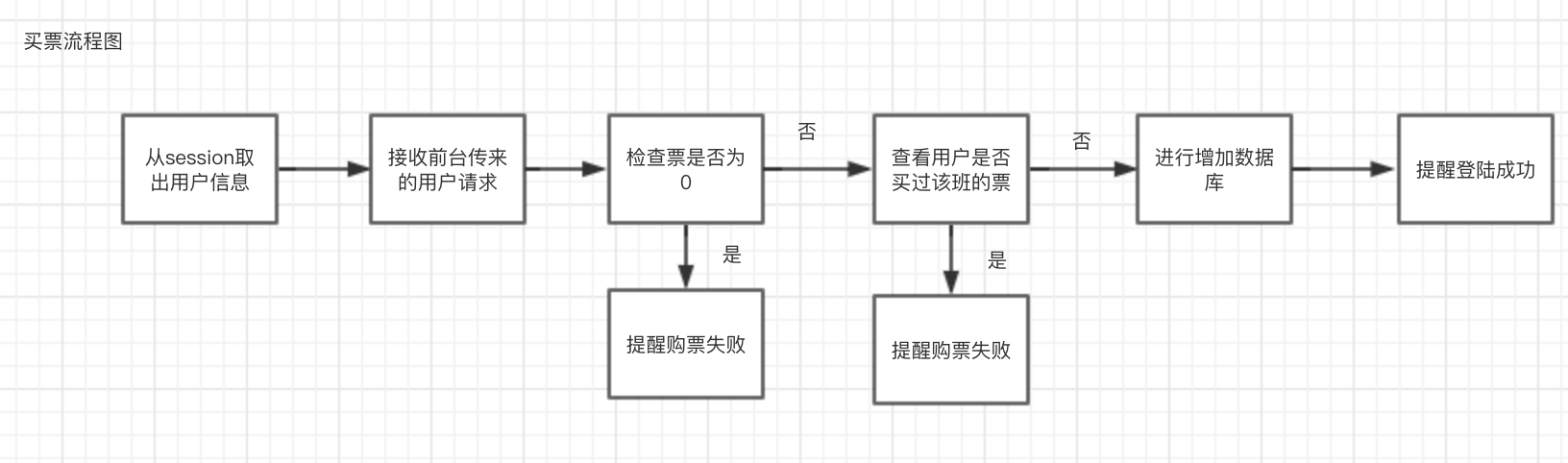
选择想买的票



点击确认购买



流程图



代码

@GetMapping(**"buy/{type}/{afid}"**)  
**public** String buy(@PathVariable String type,  
 @PathVariable String afid,  
 HttpSession httpSession,  
 Model model, RedirectAttributes attributes){  
 User user = (User)httpSession.getAttribute(**"user"**);  
 Ticket ticket = **new** Ticket();  
 TicketDetails ticketDetails = **ticketDetailsService**.getTicketDetails(afid);  
 */\*\*  
 \* release ticket  
 \*/* **if**((type.equals(**"h"**)&&ticketDetails.getFirstRelease()==0)&&  
 (type.equals(**"b"**)&&ticketDetails.getBusinessRelease()==0)&&  
 (type.equals(**"t"**)&&ticketDetails.getTouristRelease()==0)){  
 attributes.addFlashAttribute(**"message"**,**"购买失败"**);  
 **return "redirect:/info/purchaseResult"**;  
 }  
 */\*\*  
 \* one person should have one same ticket  
 \*/* **if**(**ticketService**.getTicketByUidAndAfid(user.getUid(),afid)!=**null**){  
 attributes.addFlashAttribute(**"message"**,**"不能重复购票"**);  
 **return "redirect:/info/purchaseResult"**;  
 }  
 */\*\*  
 \* add consume\_money  
 \*/* **vipService**.consumeMoney(user.getUsername(),afid,type);  
 ticket.setTID(UUID.*randomUUID*().toString()).setPurchaseTime(**new** Date()).  
 setSeatNum(**ticketService**.getTicketNum(afid)+1).setSeatType(type).setUid(user.getUid()).  
 setAfid(afid);  
 **ticketService**.buyTicket(ticket);  
 attributes.addFlashAttribute(**"message"**,**"购买成功"**);  
 **return "redirect:/info/purchaseResult"**;  
}

1. 源码目录清单

src

├── main

│   ├── java

│   │   └── com

│   │       └── littlepage

│   │           └── airplaneticketsystem

│   │               ├── AirplaneTicketSystemApplication.java

│   │               ├── aspect

│   │               │   ├── LogAspect.java

│   │               │   └── RequestLog.java

│   │               ├── controller

│   │               │   ├── Index.java

│   │               │   ├── LoginAndRegister.java

│   │               │   ├── PersonalInfo.java

│   │               │   ├── SearchPlace.java

│   │               │   ├── TestError.java

│   │               │   └── TicketInfo.java

│   │               ├── dao

│   │               │   ├── AirflightRepository.java

│   │               │   ├── AirlineRepository.java

│   │               │   ├── CompanyRepository.java

│   │               │   ├── PlaneRepository.java

│   │               │   ├── TicketDetailsRepository.java

│   │               │   ├── TicketRepository.java

│   │               │   ├── TodayTicketSimpleRepository.java

│   │               │   └── UserRepository.java

│   │               ├── handle

│   │               │   └── ControllerExceptionHandle.java

│   │               ├── intercepter

│   │               │   ├── LoginInterceptor.java

│   │               │   └── WebConfig.java

│   │               ├── pojo

│   │               │   ├── Airflight.java

│   │               │   ├── Airline.java

│   │               │   ├── Company.java

│   │               │   ├── Plane.java

│   │               │   ├── Ticket.java

│   │               │   └── User.java

│   │               ├── service

│   │               │   ├── AirflightService.java

│   │               │   ├── AirflightServiceImpl.java

│   │               │   ├── SearchService.java

│   │               │   ├── SearchServiceImpl.java

│   │               │   ├── TicketDetailsService.java

│   │               │   ├── TicketDetailsServiceImpl.java

│   │               │   ├── TicketNumberService.java

│   │               │   ├── TicketNumberServiceImpl.java

│   │               │   ├── TicketService.java

│   │               │   ├── TicketServiceImpl.java

│   │               │   ├── TodayTicketService.java

│   │               │   ├── TodayTicketServiceImpl.java

│   │               │   ├── UserService.java

│   │               │   ├── UserServiceImpl.java

│   │               │   ├── VipService.java

│   │               │   └── VipServiceImpl.java

│   │               ├── utils

│   │               │   ├── AirflightUUID.java

│   │               │   ├── DBUtils.java

│   │               │   ├── DateUtils.java

│   │               │   ├── MD5Utils.java

│   │               │   ├── Page.java

│   │               │   └── ValidateUtils.java

│   │               └── vojo

│   │                   ├── TicketDetails.java

│   │                   └── TodayTicketSimple.java

│   └── resources

│       ├── application-dev.yml

│       ├── application-pro.yml

│       ├── application.yml

│       ├── logback-spring.xml

│       ├── rebel.xml

│       ├── static

│       │   ├── css

│       │   │   └── stevecss.css

│       │   ├── img

│       │   │   ├── QRcode.png

│       │   │   └── headpic.jpeg

│       │   └── js

│       └── templates

│           ├── \_fragments.html

│           ├── error

│           │   ├── 404.html

│           │   ├── 500.html

│           │   └── error.html

│           ├── index.html

│           ├── login.html

│           ├── modifyPersonalInfo.html

│           ├── myticket.html

│           ├── personalInfo.html

│           ├── register.html

│           ├── result.html

│           └── ticketInfo.html

└── test

    └── java

        └── com

            └── littlepage

                └── airplaneticketsystem

                    ├── AirplaneTicketSystemApplicationTests.java

                    ├── fakedata

                    │   └── Fake.java

                    └── uuid

                        └── UUIDTest.java