Final Project FlyAway Specifications

Class: Become a Back-end Expert

Student: Hector Alarcon

Application: LockedMe.com

Date: 10/1/2021

FlyAway

Contents

Project Description: 3

Java technologies utilized: 3

Unique Selling Points 3

Sprint breakdown..... 4

Program Details: 5

GitHub: 5

Project Description:

- This project focused on making a web application

Java technologies utilized:

- Access modifiers by using the private keyword for methods and variables that the application will need but the user won't interact with.
- Static keywords for self-references since these are properties of the LockedMe app.
- Scanner was used for handling user input and closed using the finalize keyword at the end of the main method to make sure it closes properly.

Possible enhancements

- Once the app is available for more heavy-duty files, consider switch the array of files to an Array list to keep the fetching times constant.
- More modularity such as breaking down the second menu further into classes will help in future projects where these similar methods are used.

Unique Selling Points

This excellent application provides the following features to the customer:

- Ease of use, all the menus have user interaction information displayed on the console describing all the options available to the user.
- Security, all of the class variables are set to private, and resources closed and only accessible to the class itself. Only letting the main method access the multiple resources.
- Bullet proof, sentinel values are used for input validation and making sure the user is aware of possible mistakes during inputting data.

Sprint breakdown

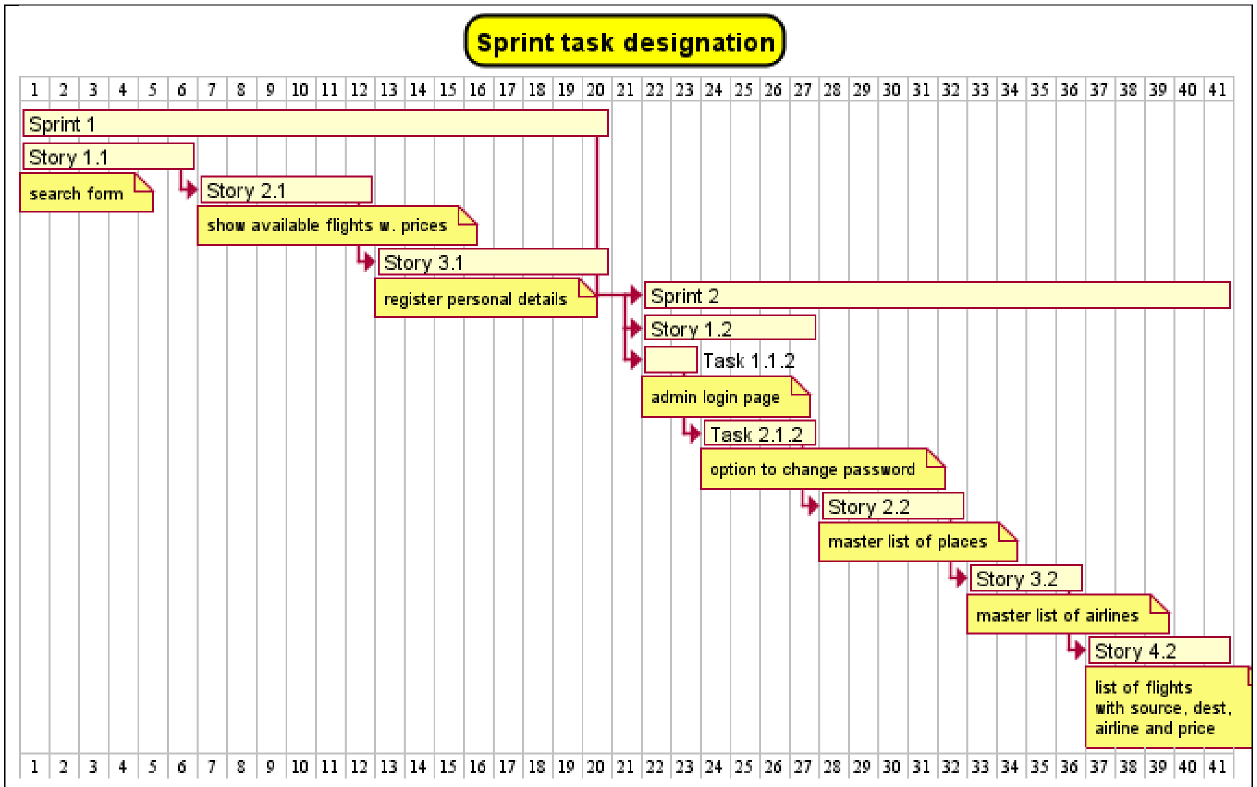


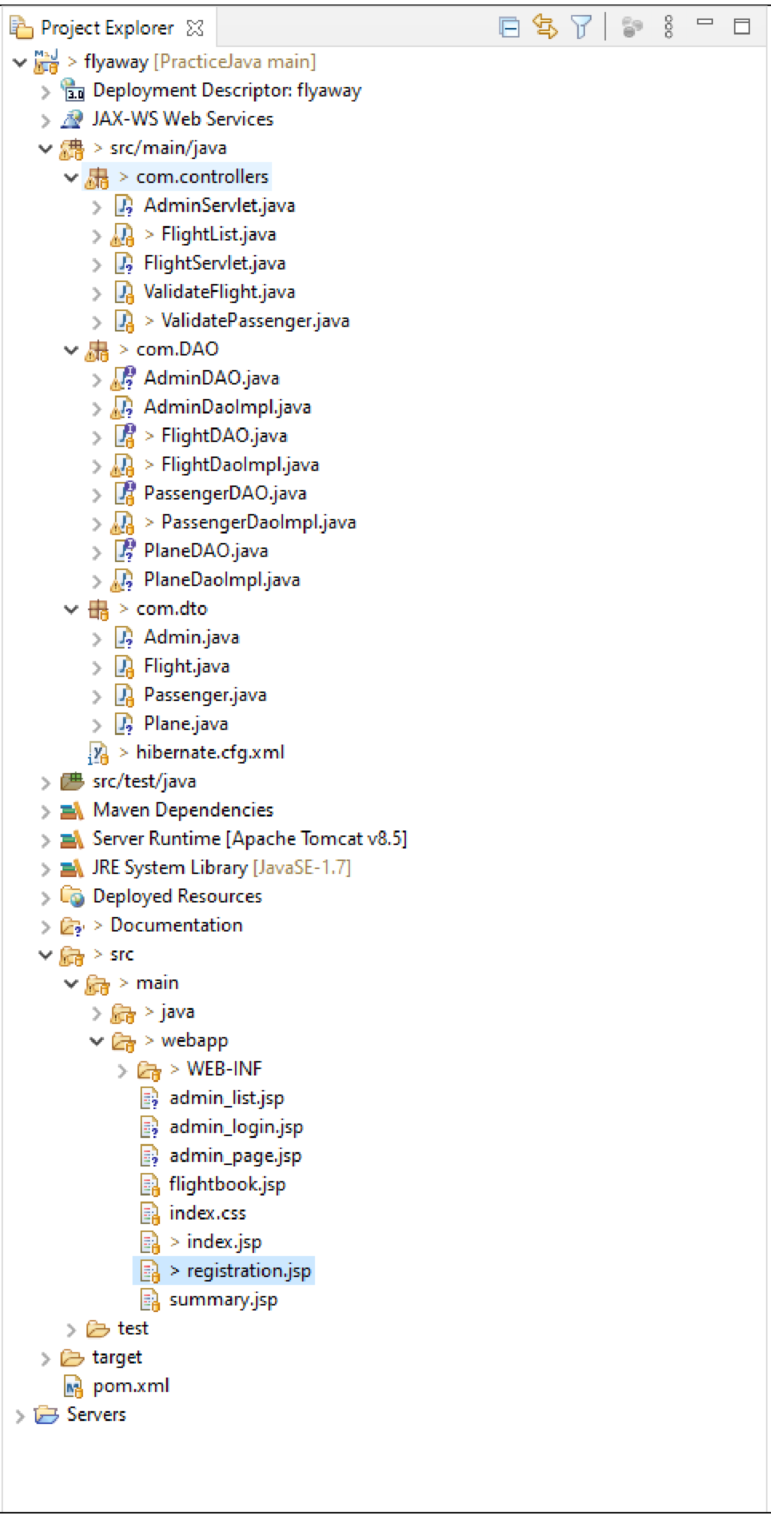
Figure 1 PlantUML Sprint Schedule

Program Details:

GitHub:

https://github.com/ProgrammedPeinado/PracticeJava/tree/main/Simplilearn/Final%20Projects/flyaway

Project Folder



```
Project Explorer
> flyaway [PracticeJava main]
  Deployment Descriptor: flyaway
  JAX-WS Web Services
  > src/main/java
    > com.controllers
      AdminServlet.java
      > FlightList.java
      FlightServlet.java
      ValidateFlight.java
      > ValidatePassenger.java
    > com.DAO
      AdminDAO.java
      AdminDaoImpl.java
      > FlightDAO.java
      > FlightDaoImpl.java
      PassengerDAO.java
      > PassengerDaoImpl.java
      PlaneDAO.java
      PlaneDaoImpl.java
    > com.dto
      Admin.java
      Flight.java
      Passenger.java
      Plane.java
    > hibernate.cfg.xml
  src/test/java
  Maven Dependencies
  Server Runtime [Apache Tomcat v8.5]
  JRE System Library [JavaSE-1.7]
  Deployed Resources
  > Documentation
  > src
    > main
      > java
      > webapp
        > WEB-INF
        admin_list.jsp
        admin_login.jsp
        admin_page.jsp
        flightbook.jsp
        index.css
        > index.jsp
        > registration.jsp
        summary.jsp
    test
  target
  pom.xml
Servers
```

AdminServlet

```
package com.controllers;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.DAO.AdminDAO;
import com.DAO.AdminDaoImpl;
import com.dto.Admin;

/**
 * Servlet implementation class AdminServlet
 */
@WebServlet("/AdminServlet")
public class AdminServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public AdminServlet()
   {
      super();
   }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		response.sendRedirect("admin_login.jsp");
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		AdminDAO adminCheck = new AdminDaoImpl();
		String user = request.getParameter("user");
		String pass = request.getParameter("pass");
		Admin admin = adminCheck.searchAdminByUser(user);
```

```
                try
                {
                        if(user.equals(admin.getUser()) & pass.equals(admin.getPass()));
                        {
                                response.sendRedirect("admin_page.jsp");
                        }
                        {
                                request.setAttribute("loginResult", true);
                                response.sendRedirect("admin_login.jsp");
                        }
                }
                catch(Exception e)
                {
                        System.out.println(e.getMessage());
                }
        }

}
```

FlightList

```
package com.controllers;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.hibernate.SessionFactory;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.DAO.FlightDAO;
import com.DAO.FlightDaoImpl;
import com.dto.Flight;

import net.bytebuddy.description.type.TypeList.Generic;
```

```
/**
 * Servlet implementation class FlightList
 */
public class FlightList extends HttpServlet
{


	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public FlightList() {
      super();
   }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		response.getWriter().append("Served at: ").append(request.getContextPath());
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		String src = request.getParameter("source_country");
		String dest = request.getParameter("destination_country");
		String date = request.getParameter("book_date");
		int pass = Integer.parseInt(request.getParameter("n_pass"));

		PrintWriter out = response.getWriter();
		FlightDAO flightDAO = new FlightDaoImpl();

		try
		{
			List<Flight> res = flightDAO.listFlights(src, dest, date, pass);
			request.setAttribute("flightList", res);
		}
		catch(Exception e)
		{
			System.out.println("\n\nStack Trace:");
			e.printStackTrace();
```

```
				System.out.println("\n\nMessage:"+e.getMessage());
				System.out.println("\n\nThere are no flights matching your search 
criterion.");
			}
			finally
			{
				
				request.getRequestDispatcher("flightbook.jsp").forward(request,response);
				out.close();
			}
			
			}
		

	}
```

FlightServlet

```
package com.controllers;

import java.io.IOException;
import java.util.List;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.DAO.FlightDAO;
import com.DAO.FlightDaoImpl;
import com.DAO.PlaneDAO;
import com.DAO.PlaneDaoImpl;
import com.dto.Flight;
import com.dto.Plane;

/**
 * Servlet implementation class FlightServlet
 */
@WebServlet("/FlightServlet")
public class FlightServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public FlightServlet() {
      super();
   }

          /**
           * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
           */
          protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
          {
                    String adminOptions = request.getParameter("selection");
                    FlightDAO flightDAO = new FlightDaoImpl();
                    PlaneDAO planeDAO = new PlaneDaoImpl();
                    String source = null; //to identify the parametrized method
                    String dest = null; //
                    List<Flight> res = null;
                    List<Plane> resAir = null;

                    if(adminOptions ==null);
                              adminOptions = "none";

                    switch(adminOptions)
                    {
                              case "listSources":
                              {
                                        res = flightDAO.listFlights(source, dest);
                                        request.setAttribute("List", res);
                                        response.sendRedirect("admin_list.jsp");
                                        break;
                              }
                              case "listAirlines":
                              {
                                        resAir = planeDAO.listAirlines();
                                        request.setAttribute("List", resAir);
                                        response.sendRedirect("admin_list.jsp");
                                        break;
                              }
                              default:
                              {
                                        res = flightDAO.listFlights();
                                        request.setAttribute("List", res);
                                        response.sendRedirect("admin_list.jsp");
                                        break;
                              }
                    }
          }

          /**
```

```
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		doGet(request, response);
	}

}
```

ValidateFlight

```
package com.controllers;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.DAO.FlightDaoImpl;
import com.dto.Flight;



/**
 * Servlet implementation class ValidateUser
 */
public class ValidateFlight extends HttpServlet {
	private static final long serialVersionUID = 1L;
    private FlightDaoImpl flightDao;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public ValidateFlight() {
        super();
        flightDao = new FlightDaoImpl();
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		if (request.getParameter("id") == null)
```

```
			{
				System.out.println("id is null");
				response.getWriter().append("The id was nulled");
			}
			else
			{
				Flight flightID = 
flightDao.searchFlightById(Integer.parseInt(request.getParameter("id")));
				request.setAttribute("flight", flightID);
				RequestDispatcher dispatcher = 
request.getRequestDispatcher("registration.jsp");
		dispatcher.forward(request, response);
			}

	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws 
ServletException, IOException {
		response.getWriter().append("Served at: ").append(request.getContextPath());
	}

}
```

ValidatePassenger

```
package com.controllers;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.DAO.FlightDaoImpl;
import com.DAO.PassengerDaoImpl;
import com.dto.Flight;
import com.dto.Passenger;



/**
 * Servlet implementation class ValidateUser
 */
```

```
public class ValidatePassenger extends HttpServlet {
	private static final long serialVersionUID = 1L;
   private PassengerDaoImpl passDao;
   private FlightDaoImpl flightDao;
   /**
    * @see HttpServlet#HttpServlet()
    */
   public ValidatePassenger() {
      super();
      passDao = new PassengerDaoImpl();
      flightDao = new FlightDaoImpl();
   }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		response.getWriter().append("Served at: ").append(request.getContextPath());
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{

		if (request.getParameter("id") == null)
		{
			System.out.println("id is null");
			response.getWriter().append("The id was nulled");
		}
		else
		{
			Passenger passenger =
passDao.searchPassengerById(Integer.parseInt(request.getParameter("id")));
			Flight flight = flightDao.searchFlightById(passenger.getFlight_id());
			request.setAttribute("passenger", passenger);
			request.setAttribute("flight", flight);
			RequestDispatcher dispatcher =
request.getRequestDispatcher("summary.jsp");
	        dispatcher.forward(request, response);
		}
	}

}
```

AdminDaoImpl

```
package com.DAO;

import java.util.List;

import javax.persistence.TypedQuery;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.dto.Admin;

public class AdminDaoImpl implements AdminDAO
{
private SessionFactory factory;

	public AdminDaoImpl()
	{
		System.out.println("Config about to load");
		StandardServiceRegistry ssr = new
StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
    Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
		factory = meta.getSessionFactoryBuilder().build();
		System.out.println("Config loaded");
	}


	@Override
	public String addAdmin(Admin admin) {
		String administrator = null;
		Session session = factory.openSession();
		Transaction txn = session.beginTransaction();
		administrator = (String) session.save(admin);
		txn.commit();
		session.close();
		return administrator;
	}


	@Override
	public void updateAdmin(String admin, String pass)
	{
		Session session = factory.openSession();
```

```
			Transaction txn = session.beginTransaction();
			Admin administrator = session.get(Admin.class, admin);
			administrator.setPass(pass);
			session.update(administrator);
			txn.commit();
			session.close();
		}

		@Override
		public void deleteAdmin(String admin) {
			Session session  = factory.openSession();
			Transaction txn = session.beginTransaction();
			Admin administrator = session.get(Admin.class, admin);
			session.delete(admin);
			txn.commit();
			session.close();
		}

		@Override
		public Admin searchAdminByUser(String admin) {
			Session session  = factory.openSession();
			Transaction txn = session.beginTransaction();
			String hql = "SELECT ad.user, ad.pass FROM administrators ad "+"WHERE 
ad.user="+admin;
			TypedQuery<Admin> query = session.createQuery(hql);
			Admin administrator = query.getSingleResult();
			return administrator;
		}
}
```

FlightDaoImpl

```
package com.DAO;

import java.io.PrintWriter;
import java.util.List;


import javax.persistence.TypedQuery;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.query.Query;
import com.dto.Flight;

public class FlightDaoImpl implements FlightDAO{
	private SessionFactory factory;

	public FlightDaoImpl()
	{
		System.out.println("Config about to load");
		StandardServiceRegistry ssr = new
StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
      Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
		factory = meta.getSessionFactoryBuilder().build();
		System.out.println("Config loaded");
	}


	@Override
	public Integer addFlight(Flight flight) {
		Integer flight_id = null;
		Session session = factory.openSession();
		Transaction txn = session.beginTransaction();
		flight_id = (Integer) session.save(flight);
		txn.commit();
		session.close();
		return flight_id;
	}


	@Override
	public void updateFlight(Integer flightID, int seats) {
		Session session = factory.openSession();
		Transaction txn = session.beginTransaction();

		Flight flight = session.get(Flight.class, flightID);
		System.out.println(flight);
		flight.setSeats(seats);
		System.out.println(flight);

		session.update(flight);
		txn.commit();
		session.close();
	}

	@Override
	public void deleteFlight(Integer flightID) {
		Session session  = factory.openSession();
		Transaction txn = session.beginTransaction();
```

```
			Flight flight = session.get(Flight.class, flightID);
			session.delete(flight);
			txn.commit();
			session.close();
		}

		@Override
		public List<Flight> listFlights(String src, String dest)
		{
			List<Flight> flights = null;
			Session session  = factory.openSession();
			Transaction txn = session.beginTransaction();

			String hql = "SELECT fli.source, fli.destination, FROM Flight AS fli";

			TypedQuery<Flight> query = session.createQuery(hql);
			query.setParameter("source", src);
			query.setParameter("destination", dest);

			flights = query.getResultList();
			session.close();
			return flights;
		}

		@Override
		public List<Flight> listFlights(String src, String dest, String date, int seats)
		{
			List<Flight> flights = null;
			Session session  = factory.openSession();
			Transaction txn = session.beginTransaction();

			String hql = "SELECT fli.id, fli.source, fli.destination, fli.date, fli.seats FROM Flight AS fli
WHERE ((fli.source =:source"+
						") AND (fli.destination =:destination"+
						") AND (fli.seats >=:seats"+
						") AND (fli.date >=:date))";

			TypedQuery<Flight> query = session.createQuery(hql);
			query.setParameter("source", src);
			query.setParameter("destination", dest);
			query.setParameter("seats", seats);
			query.setParameter("date", date);

			flights = query.getResultList();
			session.close();
			return flights;
		}
```

```
	@Override
	public List<Flight> listFlights()
	{
		List<Flight> flights = null;
		Session session  = factory.openSession();
		Transaction txn = session.beginTransaction();

		String hql = "From Flight";

		TypedQuery<Flight> query = session.createQuery(hql);

		flights = query.getResultList();
		session.close();
		return flights;
	}

	@Override
	public Flight searchFlightById(Integer flightID) {
		Session session  = factory.openSession();
		Transaction txn = session.beginTransaction();
		String hql = "FROM Flight = "+ flightID;
		TypedQuery<Flight> query = session.createQuery(hql);
		Flight flight = query.getSingleResult();
		return flight;
	}
}
```

PassengerDaoImpl

```
package com.DAO;

import java.util.List;

import javax.persistence.TypedQuery;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.dto.Passenger;

public class PassengerDaoImpl implements PassengerDAO
{
	private SessionFactory factory;

	public PassengerDaoImpl()
	{
		System.out.println("Config about to load");
		StandardServiceRegistry ssr = new
StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
	Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
		factory = meta.getSessionFactoryBuilder().build();
		System.out.println("Config loaded");
	}


	@Override
	public Integer addPassenger(Passenger passenger)
	{
		Integer pass_id = null;
		Session session = factory.openSession();
		Transaction txn = session.beginTransaction();
		pass_id = (Integer) session.save(passenger);
		txn.commit();
		session.close();
		return pass_id;
	}


	@Override
	public void updatePassenger(Integer passengerID, int seats) {
		Session session = factory.openSession();
		Transaction txn = session.beginTransaction();

		Passenger passenger = session.get(Passenger.class, passengerID);
		System.out.println(passenger);
		passenger.setSeats_purchased(seats);
		System.out.println(passenger);
```

```
                session.update(passenger);
                txn.commit();
                session.close();
        }

        @Override
        public void deletePassenger(Integer passengerID) {
                Session session  = factory.openSession();
                Transaction txn = session.beginTransaction();
                Passenger passenger = session.get(Passenger.class, passengerID);
                session.delete(passenger);
                txn.commit();
                session.close();
        }


        @Override
        public List<Passenger> listPassengers(String src, String dest, String date, int seats)
        {
                List<Passenger> passengers = null;
                Session session  = factory.openSession();
                Transaction txn = session.beginTransaction();

                System.out.println("Source: "+src+
                                                        "\nDestination: "+dest+
                                                        "\nDate: "+date+
                                                        "\nSeats: "+seats);


                String hql = "SELECT fli.source, fli.destination, fli.date, fli.seats FROM Passenger AS fli
WHERE ((fli.source =:source"+
                                                ") AND (fli.destination =:destination"+
                                                ") AND (fli.seats >=:seats"+
                                                ") AND (fli.date >=:date))";


                //System.out.println(sql);
                //Query query = session.createQuery(hql);
                TypedQuery<Passenger> query = session.createQuery(hql);
                query.setParameter("source", src);
                query.setParameter("destination", dest);
                query.setParameter("seats", seats);
                query.setParameter("date", date);

                //TypedQuery<Passenger> query = session.createSQLQuery(sql);
                System.out.println("Query created");
                passengers = query.getResultList();
                System.out.println("Query completed, leaving the method.");
```

```
			session.close();
			return passengers;
		}

		@Override
		public Passenger searchPassengerById(Integer passengerID) {
			Session session  = factory.openSession();
			Transaction txn = session.beginTransaction();
			String hql = "FROM Passenger = "+ passengerID;
			TypedQuery<Passenger> query = session.createQuery(hql);
			Passenger passenger = query.getSingleResult();
			return passenger;
		}
}
```

PlaneDaoImpl

```
package com.DAO;

import java.util.List;

import javax.persistence.TypedQuery;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import com.dto.Passenger;
import com.dto.Plane;

public class PlaneDaoImpl implements PlaneDAO
{
private SessionFactory factory;

		public PlaneDaoImpl()
		{
			System.out.println("Config about to load");
			StandardServiceRegistry ssr = new
StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();
	Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
			factory = meta.getSessionFactoryBuilder().build();
			System.out.println("Config loaded");
		}
```

```
@Override
public Integer addPlane(Plane plane) {
        Integer plane_id = null;
        Session session = factory.openSession();
        Transaction txn = session.beginTransaction();
        plane_id = (Integer) session.save(plane);
        txn.commit();
        session.close();
        return plane_id;
}


@Override
public void updatePlane(String airline, Integer flightID)
{
        Session session = factory.openSession();
        Transaction txn = session.beginTransaction();

        Plane plane = session.get(Plane.class, airline);
        System.out.println(plane);
        plane.setFlight_id(flightID);
        System.out.println(plane);

        session.update(plane);
        txn.commit();
        session.close();
}

@Override
public void deletePlane(Integer planeID)
{
        Session session  = factory.openSession();
        Transaction txn = session.beginTransaction();
        Plane plane = session.get(Plane.class, planeID);
        session.delete(plane);
        txn.commit();
        session.close();
}

@Override
public List<Plane> listPlanes()
{
        List<Plane> planes = null;
        Session session  = factory.openSession();
        Transaction txn = session.beginTransaction();

        String hql = "From Plane";
```

```
                TypedQuery<Plane> query = session.createQuery(hql);

                planes = query.getResultList();
                session.close();
                return planes;
        }

        @Override
        public List<Plane> listAirlines()
        {
                List<Plane> planes = null;
                Session session  = factory.openSession();
                Transaction txn = session.beginTransaction();

                String hql = "Select Plane.airline From Plane";

                TypedQuery<Plane> query = session.createQuery(hql);

                planes = query.getResultList();
                session.close();
                return planes;
        }
        @Override
        public Plane searchPlaneByID(Integer flightID)
        {
                Session session  = factory.openSession();
                Transaction txn = session.beginTransaction();
                String hql = "FROM Passenger = "+ flightID;
                TypedQuery<Plane> query = session.createQuery(hql);
                Plane plane = query.getSingleResult();
                return plane;
        }
}
```

Admin

```
package com.dto;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="administrators")
```

```
public class Admin
{
	@Id
	@GeneratedValue(strategy=GenerationType.SEQUENCE)
	@Column(name="username")
	private String user;
	@Column(name="password")
	private String pass;

	public String getUser() {
		return user;
	}
	public void setUser(String user) {
		this.user = user;
	}
	public String getPass() {
		return pass;
	}
	public void setPass(String pass) {
		this.pass = pass;
	}

	@Override
	public String toString() {
		return "Admin [user=" + user + ", pass=" + pass + "]";
	}
}
```

Flight

```
package com.dto;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="avail_flights")
public class Flight
{
	@Id
	@GeneratedValue(strategy=GenerationType.SEQUENCE)
	@Column(name="flight_id")
	private int id;
	@Column(name="src_point")
	private String source;
	@Column(name="dest_point")
	private String destination;
	@Column(name="travel_date")
	private String date;
	@Column(name="time_to_board")
	private String time;
	@Column(name="price")
	private double price;
	@Column(name="seat_vacancy")
	private int seats;

	public Flight()
	{
	}

	public Flight(int id, String source, String destination, String date, String time, String price,
String seats)
	{
		super();
		this.id = id;
		this.source = source;
		this.destination = destination;
		this.date = date;
		this.time = time;
		this.price = Double.parseDouble(price);
		this.seats = Integer.parseInt(seats);
	}

	public Flight(int id, String source, String destination, String seats, String date)
	{
		super();
		this.source = source;
		this.destination = destination;
		this.seats = Integer.parseInt(seats);
		this.date = date;
	}

	public Flight(String source, String destination, String seats, String date)
	{
		super();
		this.source = source;
		this.destination = destination;
		this.seats = Integer.parseInt(seats);
		this.date = date;
	}

	public int getId() {
			return id;
		}
		public void setId(int id) {
			this.id = id;
		}
		public String getSource() {
			return source;
		}
		public void setSource(String source) {
			this.source = source;
		}
		public String getDestination() {
			return destination;
		}
		public void setDestination(String destination) {
			this.destination = destination;
		}
		public String getDate() {
			return date;
		}
		public void setDate(String date) {
			this.date = date;
		}
		public String getTime() {
			return time;
		}
		public void setTime(String time) {
			this.time = time;
		}
		public double getPrice() {
			return price;
		}
		public void setPrice(double price) {
			this.price = price;
		}
		public int getSeats() {
			return seats;
		}
		public void setSeats(int seats) {
			this.seats = seats;
		}
		@Override
		public String toString() {
			return "Flight [id=" + id + ", source=" + source + ", destination=" + destination + ",
date=" + date + ", time="
					+ time + ", price=" + price + "]";
		}
}
```

Passenger

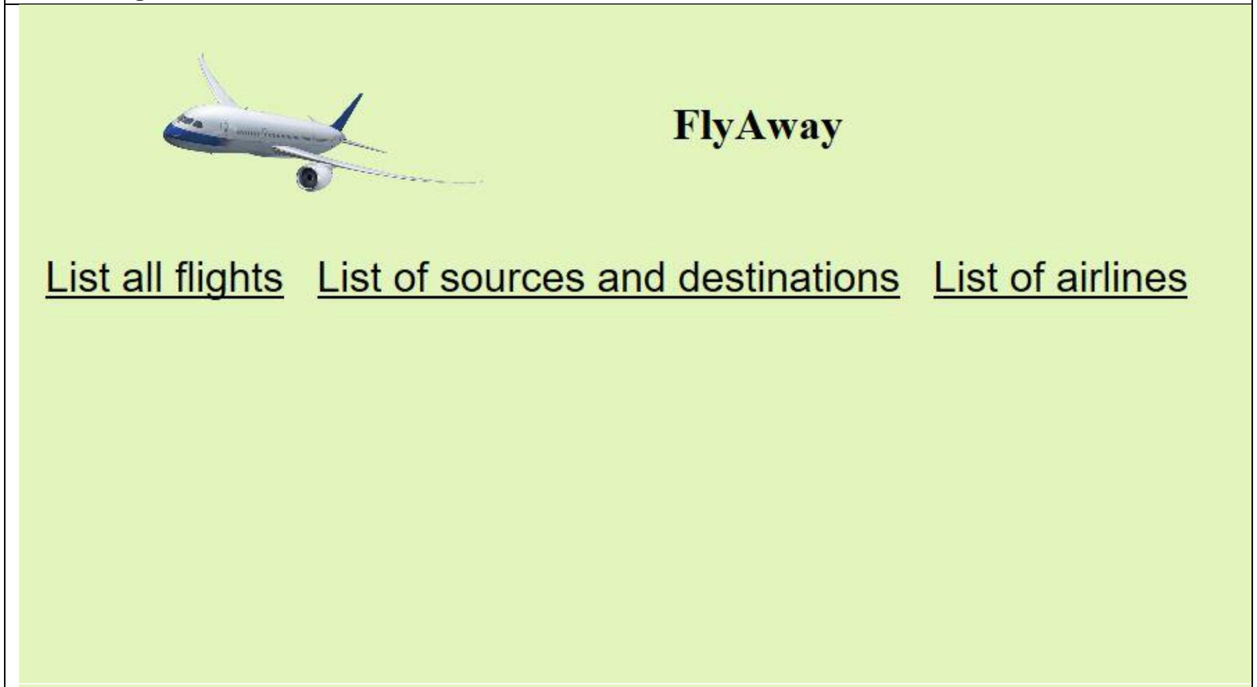
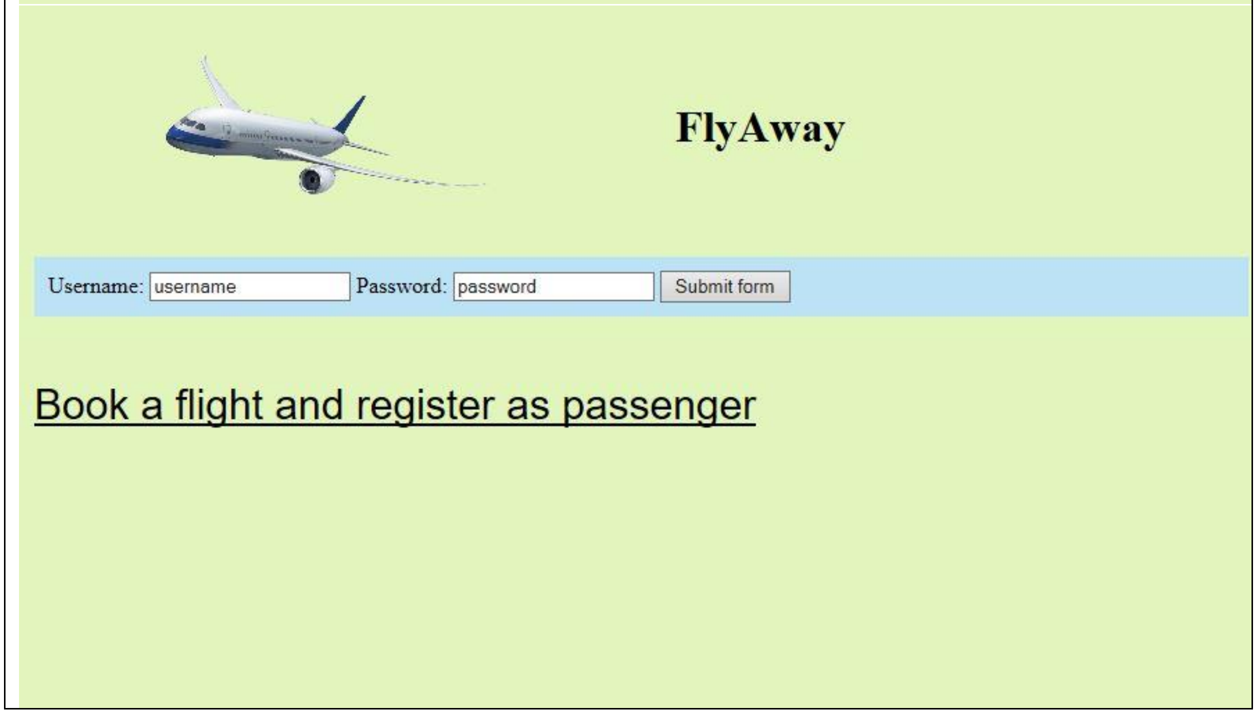
```
package com.dto;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="passengers")
public class Passenger
{
	@Id
	@GeneratedValue(strategy=GenerationType.SEQUENCE)
	@Column(name="passenger_id")
	private int passenger_id;
	@Column(name="flight_id")
	private int flight_id;
	@Column(name="firstname")
	private String firstname;
	@Column(name="lastname")
	private String lastname;
	@Column(name="seats_purchased")
	private int seats_purchased;

	public Passenger()
	{
	}

	public Passenger(String firstname, String lastname, int flight_id, int seats_purchased)
	{
		this.firstname = firstname;
		this.lastname = lastname;
		this.flight_id = flight_id;
		this.seats_purchased = seats_purchased;
	}

	public int getPassenger_id() {
		return passenger_id;
	}
	public void setPassenger_id(int passenger_id) {
		this.passenger_id = passenger_id;
	}
	public int getFlight_id() {
		return flight_id;
	}
```

```
		public void setFlight_id(int flight_id) {
			this.flight_id = flight_id;
		}
		public String getFirstname() {
			return firstname;
		}
		public void setFirstname(String firstname) {
			this.firstname = firstname;
		}
		public String getLastname() {
			return lastname;
		}
		public void setLastname(String lastname) {
			this.lastname = lastname;
		}
		public int getSeats_purchased() {
			return seats_purchased;
		}
		public void setSeats_purchased(int seats_purchased) {
			this.seats_purchased = seats_purchased;
		}

		@Override
		public String toString() {
			return "Passenger [passenger_id=" + passenger_id + ", flight_id=" + flight_id + ",
firstname=" + firstname
					+ ", lastname=" + lastname + ", seats_purchased=" + seats_purchased
+ "]";
		}

}
```

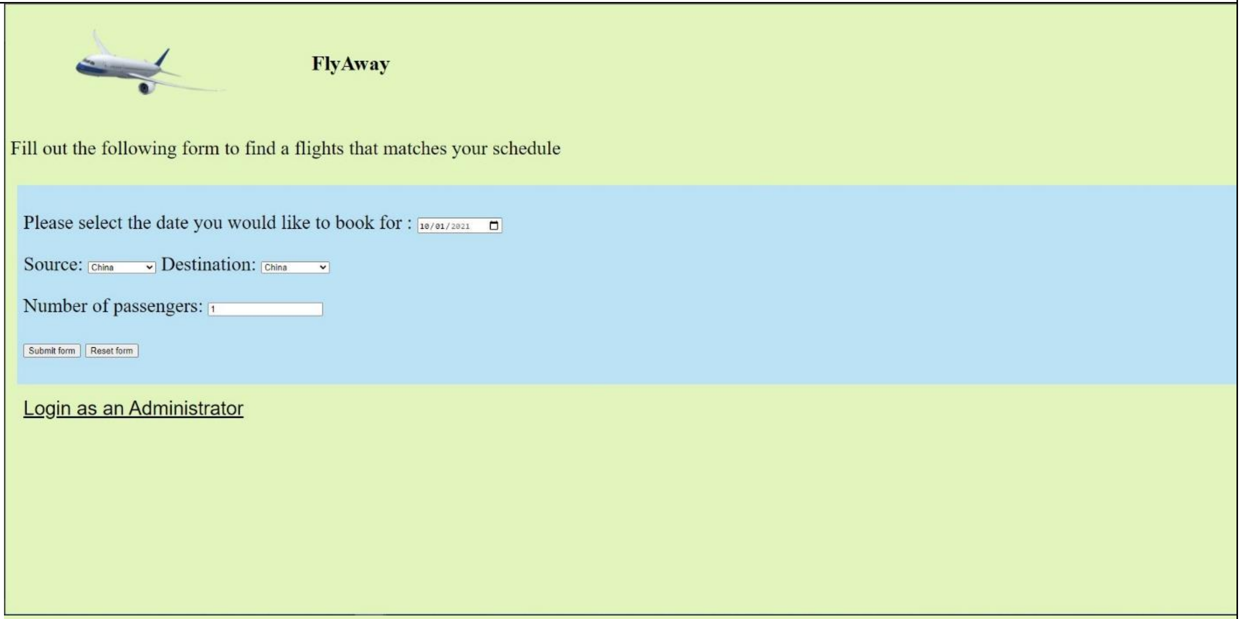
Plane

```
package com.dto;


import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="plane")
public class Plane
{
	@Id
	@GeneratedValue(strategy=GenerationType.SEQUENCE)
	@Column(name="flight_id")
	private int flight_id;
	@Column(name="plane_id")
	private int plane_id;
	@Column(name="airline")
	private String airline;

	public int getFlight_id() {
		return flight_id;
	}
	public void setFlight_id(int flight_id) {
		this.flight_id = flight_id;
	}
	public int getPlane_id() {
		return plane_id;
	}
	public void setPlane_id(int plane_id) {
		this.plane_id = plane_id;
	}
	public String getAirline() {
		return airline;
	}
	public void setAirline(String airline) {
		this.airline = airline;
	}

	@Override
	public String toString() {
		return "Plane [flight_id=" + flight_id + ", plane_id=" + plane_id + ", airline=" + airline +
"]";
	}
}
```

Screenshots

Admin Page
 FlyAway [List all flights](#) [List of sources and destinations](#) [List of airlines](#)  FlyAway Username: username Password: password Submit form [Book a flight and register as passenger](#)

Passenger pages



FlyAway

Fill out the following form to find a flights that matches your schedule

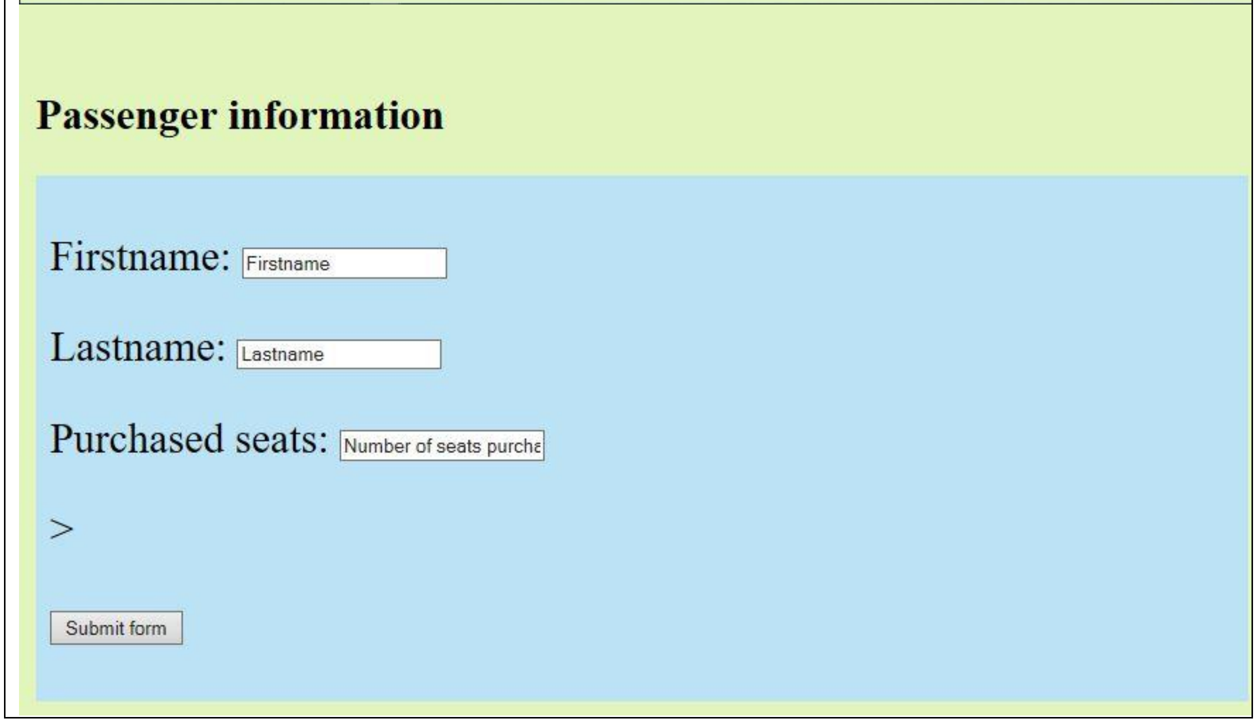
Please select the date you would like to book for : 10/01/2021

Source: China Destination: China

Number of passengers: 1

Submit form Reset form

Login as an Administrator



Passenger information

Firstname: Firstname

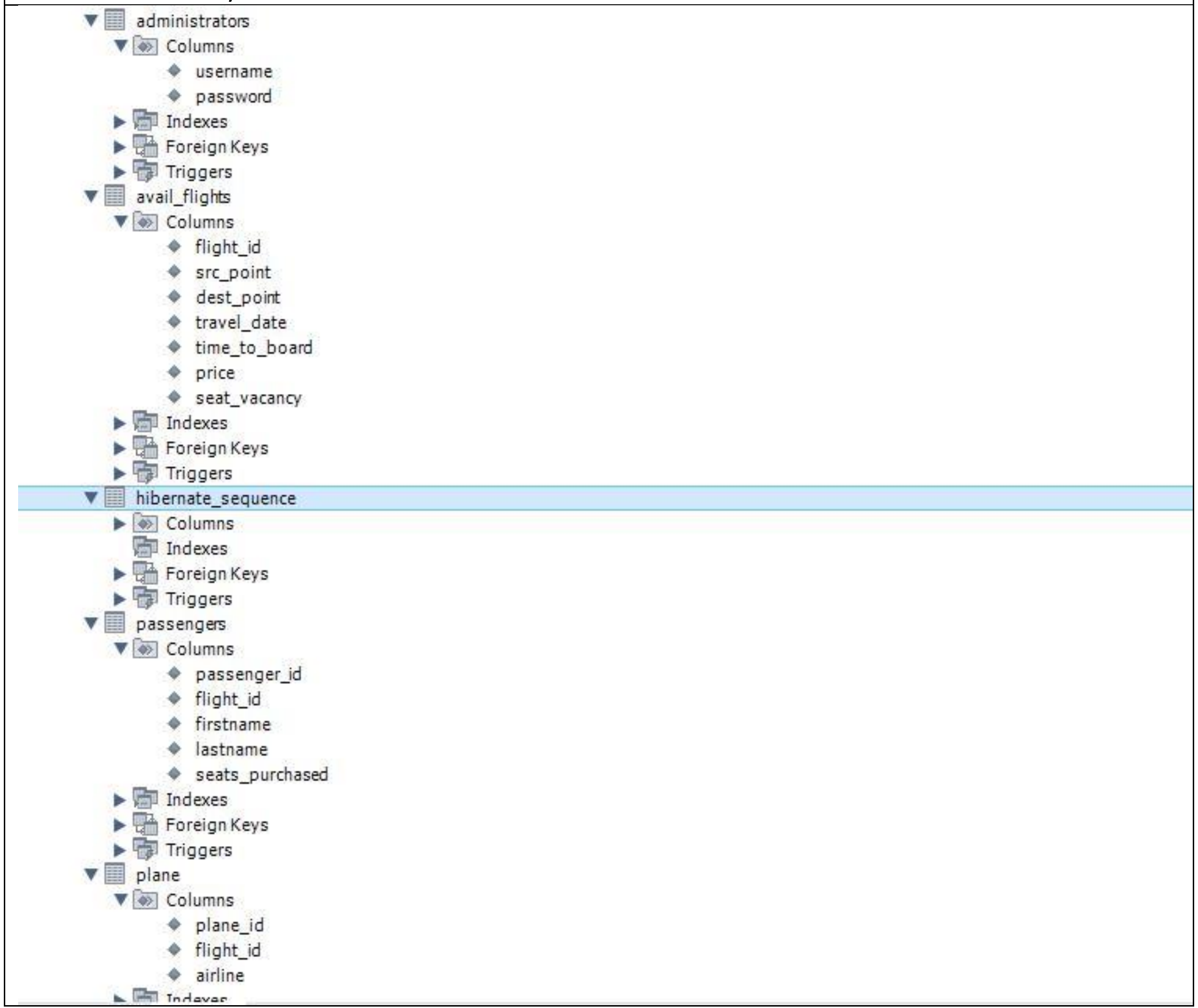
Lastname: Lastname

Purchased seats: Number of seats purcha

>

Submit form

Database hierarchy



```
▼ administrators
    ▼ Columns
        username
        password
    ▶ Indexes
    ▶ Foreign Keys
    ▶ Triggers
▼ avail_flights
    ▼ Columns
        flight_id
        src_point
        dest_point
        travel_date
        time_to_board
        price
        seat_vacancy
    ▶ Indexes
    ▶ Foreign Keys
    ▶ Triggers
▼ hibernate_sequence
    ▶ Columns
      Indexes
    ▶ Foreign Keys
    ▶ Triggers
▼ passengers
    ▼ Columns
        passenger_id
        flight_id
        firstname
        lastname
        seats_purchased
    ▶ Indexes
    ▶ Foreign Keys
    ▶ Triggers
▼ plane
    ▼ Columns
        plane_id
        flight_id
        airline
    ▶ Indexes
```