# Burning airlines – requirements

Admin

An admin can create planes on the /airplanes page. ROUTE – RESTFUL stuff

A plane has a name, rows, and columns. MODEL

When a new plane is saved the page should show a seating diagram. Like Tic Tac Toe visuals VISUAL and SITE FLOW

An admin can create flights on the /flights page.

A flight has a flight number, origin, destination, date, and plane. MODEL

When a new flight is saved, the page should show a list with the newest date at the top of the list and the number of available seats on the plane. CALCULATION RUBY OR BACKBONE???

User

A user can search for flights on the /search page. The search page should have an input form for the origin and destination.

When a user creates a search, the page should show a list of flights with the newest date at the top, and include the number of available seats on the flight.

When the user selects a flight, we should go to the show page ie, /flights/3.

The plane show page should show a seating diagram with available seats and seats that have been selected, with their names.

A user can select a seat.

When a seat is saved, the available seats on the /flights page should update.

Server side reservation conflicts, and client/server race conditions involving reservations, can be avoided by having a validator in app/models/reservation.rb.

# Models

There are models for Airplane, Flight, User, and Reservation.

An Airplane has many Flights and a Flight belongs to an Airplane.

A User has many Reservations and a Reservation belongs to a User.

A Flight has many Reservations and a Reservation belongs to a Flight.

The Reservation table is a join table between Users and Flights, which have a many-to-many relationship through Reservations.

The Airplane model has rows and columns to determine the configuration of the plane; the Reservation model has a row and a column for a particular seat.

Application flow and page

# Entry page

User selects whether Choose whether admin or user

# Admin flow

Choose either add a flight or add a plane – *persistent navigation for an admin* – always add these

## Planes - URL /airplanes page.

Design questions

* rails page or backbone to view pages

### Add a plane – CREATE Method – save to DB

*Input form conncected to the DB. Needs to modify the DB when admin presses save. Save to DB – backbone functionality*



### Admin clicks **save** – sees the seats on the

Save the data to the DB

DISPLAY the layout of the page – look back at Tic Tac Toe plane – render a BACKBONE VIEW on the page to show this?

To user display of layout, add plane name as well as type and confirm have added it

**

SEAT FUNCTIONs

How do I tell if a seat is available or not? Do I need to have something in the DB model for available?

row= [1,2,3,4]

column = [A,B,C,D,E,F, G]

need to loop through rows and columns and create a new array that concatenates each row and column

seat = row number + row name

seat = [A1,A2,A3,A4, B1,B2,B3,B4]

function – how many seats in total = number of rows \* number of columns

is seat available function??

on click add:

seat to a new array – seat taken array? OR have a status on a seat? CHANGE COLOUR OF SEAT WHAT APPROACH

if create a seat take

# flights

## Admin add a flight

1. form to enter the data
2. button to submit the form

**

1. On click submit:
   1. SAVE the form to DB
   2. RENDER a list of all flights

**

## View after flight has been added

* a list of all flights with the newest date at the top of the list
* the number of available seats on the plane. SEAT AVAILABILITY FUNCTION

# USER FLOW

*User entry pathway*

*User log in*

## Find flight - /search page

A user can search for flights on the /search page.

The search page should have an input form for the origin and destination.

Form for search – connected to the DB to see what flights are available

**

### Search results – link to flights –adding the NUMBER of seats available

When a user creates a search, the page should show a list of flights with the newest date at the top, and include the number of available seats on the flight. CALCULATION + POLLING FOR CHANGES TO THE NUMBER OF SEATS and the actual seats

**

function – how many seats

function – what seats

## FIND SEATS

See what seats are available \_VIEW

Seat availability must be updated dynamically –FUNCTION WHAT SEATS

Select a seat – EVENT ON CLICK - as soon as clicked needs to UPDATE THE DB

**

*Once have selected a seat need user to:*

* *view the details*
* *confirm and book.*

# SITEMAP or

Sitemap – all one page using Backbone

Outline of different views and the routes/urls

Outline 2 different users pathways through the site – site flow

Site URLs and CRUD

|  |  |  |  |
| --- | --- | --- | --- |
| Who | RESTFUL architecture |  | Page |
| Admin | CREATE | create planes – by admin | /airplanes |
| Admin | CREATE | create flights – by admin | /flights page. |
| Admin | SHOW | list of all flights | ??? |
|  |  |  |  |
| User | GET??? | user search for flights | /search |
| User | SHOW | User selects a flight | /flights/3 |

GET

POST

PUT

DELETE

DESTROY

CREATE

SHOW

# Data model - rails

## Models

1. Airplane:

*The Airplane model has rows and columns to determine the configuration of the plane; the Reservation model has a row and a column for a particular seat.*

* number of rows
* number of columns

seat name = row number + column number

seat status

|  |  |  |
| --- | --- | --- |
| Column | A | B |
| 1 | Seat 1A | 1B |
| 2 | 2A | 2B |
| 3 | 3A | 3B |
| 4 | 4A | 4B |
| 5 | 5A | 6B |

1. Flight
2. User
   1. user type – admin or traveller
3. Reservation. Join table in its own right with:

* reservation id;
* reservation date – *can you add additional things to a join table??*
* user id
* flight id;
* seat number – selected row and seat

Server side reservation conflicts, and client/server race conditions involving reservations, can be avoided by having a validator in app/models/reservation.rb.

## Associations

An Airplane *has many* Flights and a Flight belongs to an Airplane.

A User *has many* Reservations and a Reservation *belongs to* a User.

A Flight *has many* Reservations and a Reservation *belongs to* a Flight.

The Reservation table is a *join table* between Users and Flights, which have a *many-to-many* relationship through Reservations.

# Steps

## Create relationship model

## Create the appropriate tables

# Create the crud for each of the tables in the datamodel - rails

Goal

|  |  |  |
| --- | --- | --- |
| CRUD element | Model | User view |
| Create |  |  |
|  |  |  |
| UPDATE |  |  |
|  |  |  |
| READ |  |  |
|  |  |  |
| Destroy |  |  |
|  |  |  |

# Create the user password and log-in model – rails

# Create the admin password and log-in model - rails

# Populating the database - rails

# Displaying the data - backbone

## Models

Map the backbone models from Ruby into backbone

Map the backbone models into collections

Create the backbone models

Create the backbone collections.

## Views

2 views via backbone?