1. Primero, creemos los modelos necesarios:

```
php artisan make:model Route -m
php artisan make:model Ticket -m
php artisan make:model Booking -m
php artisan make:model Ship -m
```

2. Definir las migraciones:

`database/migrations/xxxx_create_ships_table.php`:

```
public function up()
{
    Schema::create('ships', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->integer('capacity');
        $table->text('description')->nullable();
        $table->timestamps();
    });
}
```

`database/migrations/xxxx_create_routes_table.php`:

```
public function up()
{
    Schema::create('routes', function (Blueprint $table) {
        $table->id();
        $table->foreignId('ship_id')->constrained()->onDelete('cascade');
        $table->string('origin');
        $table->string('destination');
        $table->decimal('price', 8, 2);
        $table->time('departure_time');
        $table->time('arrival_time');
        $table->timestamps();
    });
}
```

`database/migrations/xxxx_create_tickets_table.php`:

```php
public function up()
{
    Schema::create('tickets', function (Blueprint $table) {
        $table->id();
        $table->foreignId('route_id')->constrained()->onDelete('cascade');
        $table->decimal('price', 8, 2);
        $table->date('travel_date');
        $table->integer('available_seats');
        $table->timestamps();
    });
}
```

`database/migrations/xxxx_create_bookings_table.php`:

```php
public function up()
{
    Schema::create('bookings', function (Blueprint $table) {
        $table->id();
        $table->foreignId('user_id')->constrained()->onDelete('cascade');
        $table->foreignId('ticket_id')->constrained()->onDelete('cascade');
        $table->integer('number_of_seats');
        $table->decimal('total_price', 8, 2);
        $table->enum('status', ['pending', 'confirmed', 'cancelled']);
        $table->string('payment_id')->nullable();
        $table->timestamps();
    });
}
```

3. Definir los modelos con sus relaciones:

`app/Models/Ship.php`:

```php
class Ship extends Model
{
    protected $fillable = ['name', 'capacity', 'description'];

    public function routes()
    {
        return $this->hasMany(Route::class);
    }
}
```

`app/Models/Route.php`:

```php
class Route extends Model
{
    protected $fillable = [
        'ship_id',
        'origin',
        'destination',
        'price',
        'departure_time',
        'arrival_time'
    ];

    public function ship()
    {
        return $this->belongsTo(Ship::class);
    }

    public function tickets()
    {
        return $this->hasMany(Ticket::class);
    }
}
```

`app/Models/Ticket.php`:

```php
class Ticket extends Model
{
    protected $fillable = [
        'route_id',
        'price',
        'travel_date',
        'available_seats'
    ];

    public function route()
    {
        return $this->belongsTo(Route::class);
    }

    public function bookings()
    {
        return $this->hasMany(Booking::class);
    }
}
```

`app/Models/Booking.php`:

```php
class Booking extends Model
{
    protected $fillable = [
        'user_id',
        'ticket_id',
        'number_of_seats',
        'total_price',
        'status',
        'payment_id'
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function ticket()
    {
        return $this->belongsTo(Ticket::class);
    }
}
```

4. Crear los controladores:

```
php artisan make:controller RouteController --resource
php artisan make:controller TicketController --resource
php artisan make:controller BookingController --resource
```

5. Definir las rutas en `routes/web.php`:

```php
Route::middleware(['auth'])->group(function () {
    Route::resource('routes', RouteController::class);
    Route::resource('tickets', TicketController::class);
    Route::resource('bookings', BookingController::class);

    Route::get('/search', [TicketController::class, 'search'])->name('tickets.search');
    Route::post('/bookings/{booking}/cancel', [BookingController::class, 'cancel'])->name('bookings.cancel');
    Route::get('/my-bookings', [BookingController::class, 'myBookings'])->name('bookings.my-bookings');
});
```

6. Para integrar pagos, podemos usar Stripe. Primero instalar el paquete:

```
composer require stripe/stripe-php
```

7. Crear un servicio de pago:

`app/Services/PaymentService.php`:

```php
namespace App\Services;

use Stripe\Stripe;
use Stripe\Charge;

class PaymentService
{
    public function __construct()
    {
        Stripe::setApiKey(config('services.stripe.secret'));
    }

    public function processPayment($amount, $token)
    {
        return Charge::create([
            'amount' => $amount * 100, // Stripe usa centavos
            'currency' => 'usd',
            'source' => $token,
        ]);
    }
}
```

Este es un esquema básico del sistema. Necesitarás también:

1. Crear las vistas correspondientes en `resources/views/`
2. Implementar la lógica de los controladores
3. Añadir validaciones
4. Configurar los middlewares necesarios
5. Implementar el sistema de búsqueda
6. Crear las políticas de acceso
7. Implementar notificaciones por email
8. Añadir tests