



# Módulo de Reservas (Bookings) - Documentación Completa

---



## Índice

---

1. [Descripción General](#)
  2. [Características Implementadas](#)
  3. [Arquitectura del Módulo](#)
  4. [API Endpoints](#)
  5. [Componentes de UI](#)
  6. [Páginas](#)
  7. [Flujo de Estados](#)
  8. [Sistema de Disponibilidad](#)
  9. [Cálculo de Precios](#)
  10. [Instalación y Configuración](#)
  11. [Testing](#)
  12. [Próximos Pasos](#)
- 



## Descripción General

---

El módulo de Reservas (Bookings) es el **core del negocio** del sistema Rental Management. Permite a los tenants gestionar el ciclo de vida completo de las reservas de sus items, desde la creación hasta la finalización.

### Funcionalidades Principales

- **CRUD completo** de reservas
  - **Verificación de disponibilidad** en tiempo real
  - **Gestión de estados** del ciclo de vida
  - **Cálculo automático de precios** basado en días
  - **Vista de calendario** visual
  - **Dashboard integrado** con estadísticas
  - **Sistema multi-tenant** con aislamiento de datos
-

# Características Implementadas

## 1. Gestión de Reservas

Característica	Descripción	Estado
Crear reserva	Formulario completo con validación	✓
Editar reserva	Actualización con verificación de disponibilidad	✓
Cancelar reserva	Cambio de estado a CANCELLED	✓
Ver detalle	Vista completa con toda la información	✓
Listar reservas	Tabla con filtros y búsqueda	✓

## 2. Estados de Reserva

PENDING

↔

CONFIRMED

↔

IN\_PROGRESS

↔

COMPLETED

↓

CANCELLED (desde cualquier estado excepto COMPLETED)

Estado	Descripción	Acciones Disponibles
PENDING	Reserva creada, pendiente de confirmación	Confirmar, Cancelar
CONFIRMED	Reserva confirmada, ítem bloqueado	Iniciar, Cancelar
IN_PROGRESS	Reserva en curso, ítem entregado	Completar
COMPLETED	Reserva finalizada, ítem devuelto	-
CANCELLED	Reserva cancelada	-

## 3. Verificación de Disponibilidad

- ✓ **Verificación automática** al seleccionar fechas
- ✓ **Detección de conflictos** con otras reservas
- ✓ **Exclusión de reservas canceladas/completadas**
- ✓ **Feedback visual** en tiempo real

## 4. Cálculo de Precios

Días = (endDate - startDate) en días  
 Precio Total = Días ☐ Precio Base del Item  
 Depósito Sugerido = Precio Total ☐ 20%

- ☒ **Cálculo automático** basado en fechas
- ☒ **Ajuste manual** permitido
- ☒ **Sugerencia de depósito** (20% por defecto)

## 5. Dashboard Integrado

- ☒ **Estadísticas de reservas** por estado
- ☒ **Ingresos totales y pendientes**
- ☒ **Próximas reservas** (7 días)
- ☒ **Accesos rápidos** a crear/ver reservas

## Arquitectura del Módulo

### Estructura de Archivos

```

rental_management/
├── app/
│   ├── api/
│   │   ├── bookings/
│   │   │   ├── route.ts # GET, POST
│   │   │   ├── [id]/
│   │   │   │   ├── route.ts # GET, PUT, DELETE
│   │   │   │   ├── confirm/route.ts # POST
│   │   │   │   ├── start/route.ts # POST
│   │   │   │   ├── complete/route.ts # POST
│   │   │   │   ├── check-availability/route.ts # POST
│   │   │   │   └── stats/route.ts # GET
│   │   ├── bookings/
│   │   │   ├── page.tsx # Lista de reservas
│   │   │   ├── new/page.tsx # Crear reserva
│   │   │   ├── [id]/page.tsx # Detalle de reserva
│   │   │   └── calendar/page.tsx # Vista de calendario
│   ├── components/
│   │   ├── bookings/
│   │   │   ├── BookingList.tsx # Tabla de reservas
│   │   │   ├── BookingForm.tsx # Formulario crear/editar
│   │   │   ├── BookingCard.tsx # Vista de detalle
│   │   │   ├── BookingCalendar.tsx # Calendario visual
│   │   │   └── AvailabilityChecker.tsx # Verificador de disponibilidad
│   ├── prisma/
│   │   └── schema.prisma # Modelo Booking definido
  
```

## API Endpoints

---

### 1. GET /api/bookings

Lista todas las reservas del tenant con filtros opcionales.

#### Query Parameters:

```
?status=PENDING           // Filtrar por estado
?search=cliente            // Buscar por item, cliente o email
?itemId=xxx                // Filtrar por item específico
?customerId=xxx            // Filtrar por cliente específico
```

#### Response:

```
{
  "bookings": [
    {
      "id": "xxx",
      "itemId": "xxx",
      "customerId": "xxx",
      "startDate": "2024-01-01T10:00:00Z",
      "endDate": "2024-01-05T10:00:00Z",
      "totalPrice": 150.00,
      "deposit": 30.00,
      "status": "CONFIRMED",
      "notes": "...",
      "item": { "name": "...", "type": "...", "basePrice": 30 },
      "customer": { "name": "...", "email": "...", "phone": "..." }
    }
  ],
  "stats": {
    "total": 10,
    "pending": 2,
    "confirmed": 3,
    "inProgress": 1,
    "completed": 4,
    "cancelled": 0
  }
}
```

---

### 2. POST /api/bookings

Crea una nueva reserva.

#### Request Body:

```
{
  "itemId": "xxx",
  "customerId": "xxx",
  "startDate": "2024-01-01T10:00:00Z",
  "endDate": "2024-01-05T10:00:00Z",
  "totalPrice": 150.00,
  "deposit": 30.00,
  "notes": "Cliente VIP"
}
```

#### Validaciones:

- ☒ Item existe y pertenece al tenant
- ☒ Cliente existe y pertenece al tenant
- ☒ Fecha de fin posterior a fecha de inicio
- ☒ Item disponible en fechas seleccionadas

### 3. GET /api/bookings/[id]

Obtiene una reserva específica con toda la información relacionada.

#### Response:

```
{
  "id": "xxx",
  "item": {
    "id": "xxx",
    "name": "Honda PCX 125",
    "type": "VEHICLE",
    "basePrice": 30.00,
    "photos": ["..."],
    "attributes": { "licensePlate": "1234-ABC" }
  },
  "customer": {
    "id": "xxx",
    "name": "John Smith",
    "email": "john@example.com",
    "phone": "+34 600 111 222",
    "documentType": "PASSPORT",
    "documentNumber": "AB123456"
  },
  "startDate": "2024-01-01T10:00:00Z",
  "endDate": "2024-01-05T10:00:00Z",
  "totalPrice": 150.00,
  "deposit": 30.00,
  "status": "CONFIRMED",
  "notes": "..."
}
```

### 4. PUT /api/bookings/[id]

Actualiza una reserva existente.

#### Request Body (campos opcionales):

```
{
  "itemId": "xxx",
  "customerId": "xxx",
  "startDate": "2024-01-01T10:00:00Z",
  "endDate": "2024-01-05T10:00:00Z",
  "totalPrice": 150.00,
  "deposit": 30.00,
  "status": "CONFIRMED",
  "notes": "..."}
}
```

**Validaciones:**

- ☒ Si cambian fechas, verifica disponibilidad
- ☒ Excluye la reserva actual de la verificación

**5. DELETE /api/bookings/[id]**

Cancela una reserva (no la elimina, cambia estado a CANCELLED).

**Response:**

```
{
  "message": "Reserva cancelada correctamente",
  "booking": { ... }
}
```

**6. POST /api/bookings/[id]/confirm**

Confirma una reserva pendiente (PENDING → CONFIRMED).

**Validación:**

- ☒ Solo se pueden confirmar reservas en estado PENDING

**7. POST /api/bookings/[id]/start**

Inicia una reserva confirmada (CONFIRMED → IN\_PROGRESS).

**Validación:**

- ☒ Solo se pueden iniciar reservas en estado CONFIRMED
- ☒ Actualiza el estado del ítem a RENTED

**8. POST /api/bookings/[id]/complete**

Completa una reserva en progreso (IN\_PROGRESS → COMPLETED).

**Validación:**

- ☒ Solo se pueden completar reservas en estado IN\_PROGRESS
- ☒ Actualiza el estado del ítem a AVAILABLE

---

## 9. POST /api/bookings/check-availability

Verifica si un item está disponible en fechas específicas.

### Request Body:

```
{
  "itemId": "xxx",
  "startDate": "2024-01-01T10:00:00Z",
  "endDate": "2024-01-05T10:00:00Z",
  "excludeBookingId": "xxx" // Opcional, para edición
}
```

### Response:

```
{
  "available": true,
  "conflicts": [],
  "message": "Item disponible en las fechas seleccionadas"
}
```

### Si hay conflictos:

```
{
  "available": false,
  "conflicts": [
    {
      "id": "xxx",
      "startDate": "2024-01-03T10:00:00Z",
      "endDate": "2024-01-07T10:00:00Z",
      "customer": { "name": "María García" }
    }
  ],
  "message": "Item no disponible. Hay reservas que se solapan."
}
```

---

## 10. GET /api/bookings/stats

Obtiene estadísticas completas de reservas.

### Response:

```
{
  "stats": {
    "total": 10,
    "pending": 2,
    "confirmed": 3,
    "inProgress": 1,
    "completed": 4,
    "cancelled": 0,
    "totalRevenue": 1500.00,
    "pendingRevenue": 850.00
  },
  "upcomingBookings": [
    {
      "id": "xxx",
      "startDate": "2024-01-02T10:00:00Z",
      "item": { "name": "...", "photos": [...] },
      "customer": { "name": "..." },
      "totalPrice": 150.00
    }
  ],
  "monthlyData": {
    "2024-01": { "count": 5, "revenue": 750.00 },
    "2024-02": { "count": 8, "revenue": 1200.00 }
  }
}
```

## Componentes de UI

### 1. BookingList ( components/bookings/BookingList.tsx )

Componente principal que muestra la tabla de reservas con filtros.

#### Props:

```
{
  initialBookings?: BookingWithRelations[];
  initialStats?: BookingStats;
}
```

#### Características:

- ☒ Tabla responsive con todas las reservas
- ☒ Filtros por estado (pending, confirmed, etc.)
- ☒ Búsqueda por item, cliente o email
- ☒ Estadísticas en cards (total, por estado)
- ☒ Acciones según estado (confirmar, iniciar, completar, cancelar)
- ☒ Actualización automática tras acciones

### 2. BookingForm ( components/bookings/BookingForm.tsx )

Formulario completo para crear nuevas reservas.

**Características:**

- ☒ Selector de item (solo disponibles)
- ☒ Selector de cliente con link para crear nuevo
- ☒ Date picker para fechas de inicio y fin
- ☒ **Verificación automática de disponibilidad**
- ☒ **Cálculo automático de precio** (días × basePrice)
- ☒ Sugerencia de depósito (20%)
- ☒ Campo de notas
- ☒ Validación con Zod y react-hook-form

**Flujo:**

1. Usuario selecciona item
2. Usuario selecciona cliente
3. Usuario selecciona fechas
4. Sistema verifica disponibilidad automáticamente
5. Sistema calcula precio basado en días
6. Usuario puede ajustar precio y depósito
7. Usuario crea la reserva

**3. BookingCard ( components/bookings/BookingCard.tsx )**

Vista detallada de una reserva individual.

**Props:**

```
{
  booking: BookingWithRelations;
  onUpdate?: () => void;
}
```

**Características:**

- ☒ Header con ID y estado de la reserva
- ☒ Información completa del item con foto
- ☒ Información completa del cliente
- ☒ Fechas de reserva con duración
- ☒ Desglose financiero (precio/día, días, depósito, total)
- ☒ Notas de la reserva
- ☒ **Botones de acción según estado actual**

**4. BookingCalendar ( components/bookings/BookingCalendar.tsx )**





Vista de calendario visual usando `react-big-calendar`.

**Características:**

- ☒ Vista mensual, semanal y diaria
- ☒ Eventos coloreados por estado
- ☒ Click en evento → redirige a detalle
- ☒ Click en slot vacío → redirige a crear reserva con fechas prellenadas
- ☒ Leyenda de colores

- ☒ Localización en español
- ☒ Solo muestra reservas activas (excluye canceladas)

#### Colores por Estado:

-  **Amarillo** - Pendiente
-  **Azul** - Confirmada
-  **Verde** - En Progreso
-  **Gris** - Completada

## 5. AvailabilityChecker ( components/bookings/AvailabilityChecker.tsx )

Componente standalone para verificar disponibilidad.

#### Props:

```
{
  itemId: string;
  excludeBookingId?: string;
}
```

#### Características:

- ☒ Inputs para fechas de inicio y fin
- ☒ Botón para verificar disponibilidad
- ☒ Feedback visual (verde/rojo)
- ☒ Lista de reservas en conflicto si no disponible

## Páginas

### 1. /bookings - Lista de Reservas

Página principal del módulo con tabla de reservas.

#### Características:

- ☒ Server-side rendering con datos iniciales
- ☒ Botones de acción rápida (Nueva Reserva, Ver Calendario)
- ☒ Integración con BookingList

### 2. /bookings/new - Nueva Reserva

Formulario para crear una nueva reserva.

#### Características:

- ☒ Formulario completo con BookingForm
- ☒ Validación en tiempo real
- ☒ Redirección a detalle tras crear

### 3. `/bookings/[id]` - Detalle de Reserva

Vista detallada de una reserva específica.

#### Características:

- ☒ Muestra toda la información con BookingCard
- ☒ Botones de acción según estado
- ☒ Server-side rendering

### 4. `/bookings/calendar` - Vista de Calendario

Calendario visual de todas las reservas.

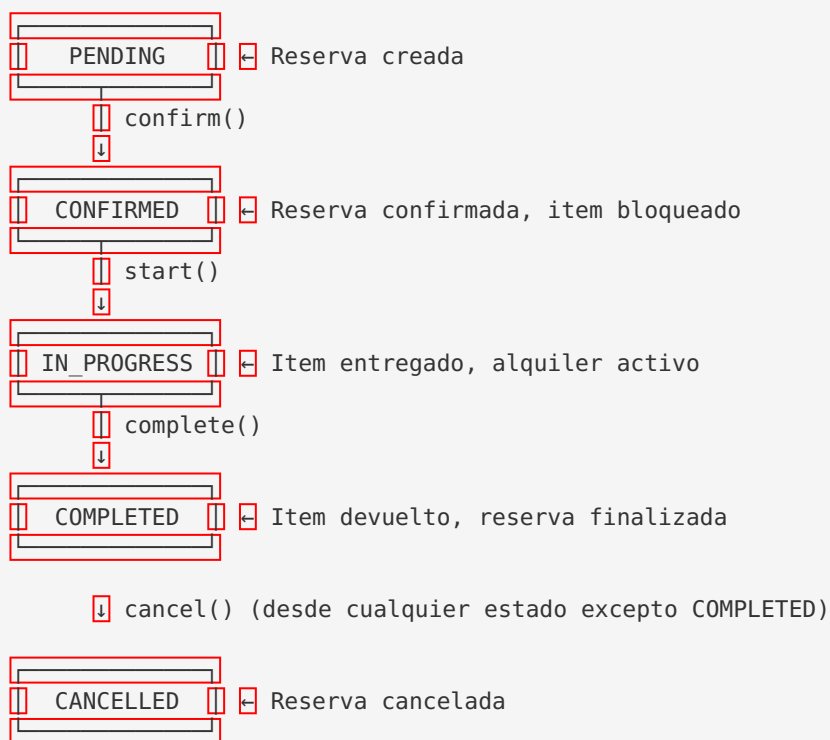
#### Características:

- ☒ Vista interactiva con BookingCalendar
- ☒ Botón para crear nueva reserva



## Flujo de Estados

### Diagrama de Flujo



## Acciones por Estado

Estado Actual	Acciones Disponibles	Estado Siguiente
PENDING	confirm(), cancel()	CONFIRMED, CANCELLED
CONFIRMED	start(), cancel()	IN_PROGRESS, CANCELLED
IN_PROGRESS	complete()	COMPLETED
COMPLETED	-	-
CANCELLED	-	-

## ✓ Sistema de Disponibilidad

### Lógica de Verificación

Para determinar si un item está disponible en unas fechas específicas, se buscan reservas que cumplan **todas** las siguientes condiciones:

```
WHERE itemId = :itemId
  AND status NOT IN ('CANCELLED', 'COMPLETED')
  AND (
    (startDate <= :endDate AND endDate >= :startDate)
  )
```

### Casos de Solapamiento

```
Reserva Existente:  |=====|
Fechas Solicitadas:  |=====|  ✗ CONFLICTO

Reserva Existente:  |=====|
Fechas Solicitadas:  |=====|  ✓ OK

Reserva Existente:  |=====|
Fechas Solicitadas:  |=====|  ✓ OK

Reserva Existente:  |=====|
Fechas Solicitadas:  |=====|  ✗ CONFLICTO
```

### Exclusiones

Al editar una reserva existente, se excluye la propia reserva de la verificación:

```
WHERE itemId = :itemId
  AND id != :currentBookingId
  AND status NOT IN ('CANCELLED', 'COMPLETED')
  ...
```

## Cálculo de Precios

### Fórmula Base

```
const diffTime = Math.abs(endDate.getTime() - startDate.getTime());
const days = Math.ceil(diffTime / (1000 * 60 * 60 * 24));
const totalPrice = days * item.basePrice;
const suggestedDeposit = totalPrice * 0.2; // 20%
```

### Ejemplo Práctico

Item: Honda PCX 125  
 Precio Base: €30/día  
 Fecha Inicio: 01/01/2024  
 Fecha Fin: 05/01/2024  
 Días: 4  
 Precio Total: 4 x €30 = €120  
 Depósito Sugerido: €120 x 20% = €24

### Ajuste Manual

Los usuarios pueden ajustar manualmente:

- ☒ Precio total (para descuentos, ofertas especiales, etc.)
- ☒ Depósito (según políticas del negocio)



## Instalación y Configuración

### 1. Instalar Dependencias

```
cd /home/ubuntu/rental_management
npm install react-big-calendar date-fns
```

### 2. Migrar Base de Datos

El modelo `Booking` ya está definido en el schema de Prisma. Si aún no has migrado:

```
npx prisma migrate dev --name add_bookings
```

### 3. Ejecutar Seed

Para poblar la base de datos con datos de ejemplo:

```
npx prisma db seed
```

Esto creará:

- ☒ 4 clientes de ejemplo para cada tenant
- ☒ 6-10 reservas de ejemplo en diferentes estados
- ☒ Reservas pasadas, actuales y futuras

## 4. Iniciar Servidor de Desarrollo

```
npm run dev
```

## 5. Acceder al Módulo

```
http://demo.localhost:3000/bookings
```

### Credenciales de prueba:

- Email: `owner@demo.com`

- Password: `password123`



## Testing

### Testing Manual - Checklist

#### 1. Crear Reserva

- ☐ Crear reserva con fechas válidas
- ☐ Verificar cálculo automático de precio
- ☐ Verificar sugerencia de depósito (20%)
- ☐ Intentar crear reserva con ítem no disponible
- ☐ Intentar crear reserva con fecha fin antes de fecha inicio
- ☐ Crear reserva con notas

#### 2. Ver Reservas

- ☐ Ver lista completa de reservas
- ☐ Filtrar por estado (pending, confirmed, etc.)
- ☐ Buscar por nombre de ítem
- ☐ Buscar por nombre de cliente
- ☐ Buscar por email de cliente
- ☐ Verificar que las estadísticas son correctas

#### 3. Gestión de Estados

- ☐ Confirmar reserva pendiente
- ☐ Iniciar reserva confirmada
- ☐ Completar reserva en progreso
- ☐ Cancelar reserva desde PENDING
- ☐ Cancelar reserva desde CONFIRMED
- ☐ Intentar cancelar reserva COMPLETED (debería fallar)

#### 4. Disponibilidad

- ☐ Verificar disponibilidad de ítem libre
- ☐ Verificar disponibilidad de ítem con reserva existente
- ☐ Verificar que se muestran los conflictos
- ☐ Editar reserva y verificar que se excluye a sí misma

## 5. Calendario

- [ ] Ver calendario mensual
- [ ] Ver calendario semanal
- [ ] Ver calendario diario
- [ ] Click en evento → redirige a detalle
- [ ] Click en slot vacío → redirige a crear reserva
- [ ] Verificar colores por estado

## 6. Dashboard

- [ ] Verificar estadísticas de reservas
- [ ] Verificar ingresos totales
- [ ] Verificar ingresos pendientes
- [ ] Ver próximas reservas (7 días)
- [ ] Click en accesos rápidos

## 7. Multi-tenant

- [ ] Login con demo tenant → ver solo sus reservas
- [ ] Login con scooters-madrid tenant → ver solo sus reservas
- [ ] Verificar aislamiento de datos



## Próximos Pasos

### Mejoras Sugeridas

#### 1. Módulo de Clientes Completo

- Página de lista de clientes
- Formulario de crear/editar cliente
- Vista de detalle con historial de reservas

#### 2. Notificaciones y Recordatorios

- Email de confirmación de reserva
- Recordatorio 24h antes del inicio
- Notificación de cancelación

#### 3. Pagos e Invoices

- Integración con pasarelas de pago
- Generación automática de facturas
- Registro de pagos y depósitos

#### 4. Contratos Digitales

- Generación de contratos PDF
- Firma digital
- Almacenamiento en la nube

#### 5. Inspecciones de Vehículos

- Formulario de inspección (entrega/devolución)
- Subida de fotos de daños
- Comparativa de estado

## 6. Reportes y Analytics

- Dashboard avanzado con gráficos
- Exportación a Excel/CSV
- Reporte de ingresos por período

## 7. Integración con Calendarios Externos

- Sincronización con Google Calendar
- Exportación a iCal
- Booking widgets para web pública

## 8. Sistema de Reviews

- Clientes pueden dejar reseñas
- Rating de items
- Moderación de comentarios

## 9. Optimizaciones de Disponibilidad

- Cache de disponibilidad
- Búsqueda de items disponibles en rango de fechas
- Sugerencias de fechas alternativas

## 10. Mobile App

- App React Native para operadores
- Gestión de reservas desde móvil
- Notificaciones push









## Soporte

Para preguntas o problemas con el módulo de Reservas:

1. **Consulta la documentación** completa
2. **Revisa el código** en los archivos mencionados
3. **Ejecuta los tests** manuales del checklist
4. **Contacta al equipo de desarrollo** si necesitas ayuda adicional

## Changelog

### v1.0.0 (Nov 2024)

-  Implementación completa del módulo de Reservas
-  API routes con todas las operaciones CRUD
-  Componentes de UI completos y funcionales
-  Vista de calendario con react-big-calendar
-  Sistema de verificación de disponibilidad
-  Integración con dashboard
-  Seed con datos de ejemplo
-  Documentación completa

**Última actualización:** Noviembre 2024

**Versión:** 1.0.0

**Desarrollado para:** Rental Management MVP