Módulo de Gestión de Clientes y Facturación

📋 Descripción General

Este módulo implementa la gestión completa de clientes y facturación automática para el MVP de Rental Management. Las facturas se generan automáticamente al confirmar una reserva y se pueden descargar en formato PDF.

🎯 Funcionalidades Implementadas

1. Gestión de Clientes

Características:

- ✅ Lista de clientes con búsqueda por nombre, email, teléfono o documento
- ✅ Crear nuevos clientes con validación de datos
- ✅ Ver detalle de cliente con historial completo de reservas
- ✅ Editar información del cliente
- ✅ Eliminar clientes (solo si no tienen reservas)
- ✅ Estadísticas: Total de clientes, clientes con reservas, total de reservas

Campos del Cliente:

- Nombre completo (requerido)
- Email (requerido, único por tenant)
- Teléfono (requerido)
- Tipo de documento: DNI, NIE, Pasaporte, Licencia de Conducir (requerido)
- Número de documento (requerido)
- Dirección (opcional)
- Ciudad (opcional)
- País (opcional)
- Notas (opcional)

2. Gestión de Facturación

Características:

- ✅ Lista de facturas con filtros por estado (Pendiente, Pagada, Cancelada)
- ✅ Ver detalle de factura con información completa
- ✅ Actualizar estado de factura (Pendiente → Pagada/Cancelada)
- ✅ Generación automática de factura al confirmar reserva
- ✅ Descarga de facturas en formato PDF
- ✅ Estadísticas financieras detalladas

Estados de Factura:

- **PENDING**: Factura pendiente de pago
- **PAID**: Factura pagada
- **CANCELLED**: Factura cancelada

- **REFUNDED**: Factura reembolsada

Generación Automática:

Cuando se confirma una reserva (POST `/api/bookings/{id}/confirm`):
1. Se actualiza el estado de la reserva a CONFIRMED
2. Se genera un número de factura único (formato: `INV-YYYY-NNNN`)
3. Se crea la factura con estado PENDING
4. Se genera automáticamente el PDF de la factura
5. El PDF se guarda en `/public/invoices/tenant-{tenantId}/`

3. Generación de PDFs

Características:

- ✅ PDF profesional con diseño limpio
- ✅ Información completa del tenant (nombre, ubicación)
- ✅ Datos del cliente (nombre, email, teléfono, documento)
- ✅ Detalles de la reserva (item, fechas, precios)
- ✅ Cálculo de totales (precio base + depósito)
- ✅ Estado de pago y fechas relevantes
- ✅ Descarga directa desde la interfaz

4. Dashboard Actualizado

Nuevas Métricas Financieras:

- **Ingresos Totales**: Suma de todas las facturas
- **Ingresos Pagados**: Suma de facturas con estado PAID
- **Ingresos Pendientes**: Suma de facturas con estado PENDING
- **Total Clientes**: Cantidad total de clientes

Nueva Sección:

- **Últimas Facturas Generadas**: Muestra las 5 últimas facturas con:
- Número de factura
- Cliente y item asociado
- Monto
- Estado visual con badge

Quick Actions Actualizados:

- Acceso rápido a módulo de Clientes
- Acceso rápido a módulo de Facturas
- Enlaces para crear nuevos clientes
- Ver todas las facturas

📁 Estructura de Archivos

```
rental_management/
├── app/
│   ├── api/
│   │   ├── customers/
│   │   │   ├── route.ts                    # GET (listar), POST (crear)
│   │   │   └── [id]/
│   │   │       └── route.ts                # GET, PUT, DELETE
│   │   ├── invoices/
│   │   │   ├── route.ts                    # GET (listar)
│   │   │   └── [id]/
│   │   │       ├── route.ts                # GET, PUT
│   │   │       └── pdf/
│   │   │           └── route.ts            # GET (descargar PDF)
│   │   └── bookings/
│   │       └── [id]/
│   │           └── confirm/
│   │               └── route.ts            # POST (confirmar + facturar)
│   ├── customers/
│   │   ├── page.tsx                        # Lista de clientes
│   │   ├── new/
│   │   │   └── page.tsx                    # Crear cliente
│   │   └── [id]/
│   │       ├── page.tsx                    # Ver detalle
│   │       └── edit/
│   │           └── page.tsx                # Editar cliente
│   ├── invoices/
│   │   ├── page.tsx                        # Lista de facturas
│   │   └── [id]/
│   │       └── page.tsx                    # Ver detalle
│   └── dashboard/
│       └── page.tsx                        # Dashboard con métricas
├── components/
│   ├── customers/
│   │   ├── CustomerList.tsx                # Tabla de clientes
│   │   ├── CustomerForm.tsx                # Formulario crear/editar
│   │   └── CustomerCard.tsx                # Vista detalle
│   └── invoices/
│       ├── InvoiceList.tsx                 # Tabla de facturas
│       ├── InvoiceCard.tsx                 # Vista detalle
│       └── InvoiceStatusBadge.tsx          # Badge de estado
├── lib/
│   └── pdf-generator.ts                    # Generador de PDFs
├── prisma/
│   ├── schema.prisma                       # Modelos Customer e Invoice
│   └── seed.ts                             # Datos de ejemplo con facturas
└── public/
    └── invoices/                           # PDFs generados
        └── tenant-{tenantId}/
```

🔌 API Endpoints

Clientes

`GET /api/customers`

Lista todos los clientes del tenant con opción de búsqueda.

Query Parameters:
- `search` (opcional): Busca en nombre, email, teléfono o documento

Response:

```
{
  "customers": [
    {
      "id": "...",
      "name": "Juan Pérez",
      "email": "juan@example.com",
      "phone": "+34 600 000 000",
      "documentType": "DNI",
      "documentNumber": "12345678A",
      "_count": {
        "bookings": 5
      }
    }
  ]
}
```

`POST /api/customers`

Crea un nuevo cliente.

Body:

```
{
  "name": "Juan Pérez",
  "email": "juan@example.com",
  "phone": "+34 600 000 000",
  "documentType": "DNI",
  "documentNumber": "12345678A",
  "address": "Calle Principal 123",
  "city": "Madrid",
  "country": "España",
  "notes": "Cliente VIP"
}
```

`GET /api/customers/{id}`

Obtiene detalle de un cliente con historial de reservas.

`PUT /api/customers/{id}`

Actualiza información de un cliente.

`DELETE /api/customers/{id}`

Elimina un cliente (solo si no tiene reservas).

Facturas

`GET /api/invoices`

Lista todas las facturas del tenant con filtros.

Query Parameters:
- `status` (opcional): PENDING, PAID, CANCELLED, ALL

Response:

```
{
  "invoices": [...],
  "stats": {
    "total": 10,
    "pending": 3,
    "paid": 6,
    "cancelled": 1,
    "totalAmount": 5000.00,
    "paidAmount": 4200.00,
    "pendingAmount": 800.00
  }
}
```


`GET /api/invoices/{id}`

Obtiene detalle de una factura.

`PUT /api/invoices/{id}`

Actualiza el estado de una factura.

Body:


```
{
  "status": "PAID",
  "paidAt": "2024-11-27T10:00:00Z"
}
```


`GET /api/invoices/{id}/pdf`

Descarga el PDF de la factura.

Response: PDF binary file

Confirmar Reserva

`POST /api/bookings/{id}/confirm`

Confirma una reserva y genera la factura automáticamente.

Response:


```
{
  "message": "Reserva confirmada y factura generada correctamente",
  "booking": {...},
  "invoice": {
    "id": "...",
    "invoiceNumber": "INV-2024-0001",
    "amount": 250.00,
    "status": "PENDING"
  }
}
```

🧪 Testing

1. Configurar Base de Datos

Opción A: PostgreSQL Local

```
# Instalar PostgreSQL
sudo apt update
sudo apt install postgresql postgresql-contrib

# Iniciar servicio
sudo service postgresql start

# Crear base de datos y usuario
sudo -u postgres psql
CREATE DATABASE rental_management;
CREATE USER rental_user WITH ENCRYPTED PASSWORD 'rental_password';
GRANT ALL PRIVILEGES ON DATABASE rental_management TO rental_user;
\q
```

Opción B: Base de Datos Cloud (Recomendado)

- Neon (https://neon.tech) - Free tier disponible
- Supabase (https://supabase.com) - Free tier disponible
- Railway (https://railway.app) - Free tier disponible

Actualizar `DATABASE_URL` en `.env` con tu connection string.

2. Ejecutar Migraciones y Seed

```
cd /home/ubuntu/rental_management

# Generar cliente de Prisma
npx prisma generate

# Crear migraciones
npm run db:migrate

# Ejecutar seed (incluye clientes y facturas)
npm run db:seed
```

3. Iniciar Aplicación

```
npm run dev
```

La aplicación estará disponible en `http://localhost:3000`

4. Configurar Subdominios Locales

Añadir a `/etc/hosts`:

```
127.0.0.1 demo.localhost
127.0.0.1 scooters-madrid.localhost
127.0.0.1 boats-marbella.localhost
```

5. Probar Módulos

Credenciales de Prueba:

```
Demo Tenant (subdomain: demo)
- owner@demo.com / password123
- admin@demo.com / password123
- operator@demo.com / password123
```

Flujo de Prueba Completo:

1. **Login**: `http://demo.localhost:3000/login`
   - Email: `owner@demo.com`
   - Password: `password123`

2. **Dashboard**: Verificar métricas financieras
   - Ingresos totales, pagados y pendientes
   - Total de clientes
   - Últimas facturas generadas

3. **Gestión de Clientes**: `http://demo.localhost:3000/customers`
   - Ver lista de clientes existentes
   - Crear nuevo cliente
   - Ver detalle con historial de reservas
   - Editar información
   - Probar búsqueda

4. **Gestión de Facturas**: `http://demo.localhost:3000/invoices`
   - Ver lista de facturas
   - Filtrar por estado (Todas, Pendientes, Pagadas)
   - Ver detalle de factura
   - Descargar PDF
   - Actualizar estado (Pendiente → Pagada)

5. **Crear Reserva y Confirmar**:
   - Ir a Reservas: `http://demo.localhost:3000/bookings`
   - Crear nueva reserva
   - Confirmar reserva (esto genera factura automáticamente)
   - Verificar que se creó la factura
   - Descargar PDF de la factura

🎨 Características de UI/UX

Componentes Reutilizables

- **CustomerList**: Tabla responsive con búsqueda en tiempo real
- **CustomerForm**: Formulario validado con react-hook-form y Zod
- **CustomerCard**: Vista detallada con historial de reservas
- **InvoiceList**: Tabla con filtros y estadísticas financieras
- **InvoiceCard**: Vista detallada con acciones de estado
- **InvoiceStatusBadge**: Badge visual para estados de factura

Validaciones

- Email único por tenant
- Campos requeridos con mensajes de error claros
- Validación de formato de email
- Prevención de eliminación si hay reservas asociadas

Feedback Visual

- Toast notifications para acciones (éxito/error)
- Loading states durante operaciones
- Estados visuales (disponible, alquilado, pendiente, pagado)
- Badges de colores para estados

📊 Datos de Seed

El seed genera automáticamente:
- **3 tenants** con usuarios y configuración
- **Clientes** para cada tenant
- **Items** de inventario (vehículos, barcos)
- **Reservas** en diferentes estados
- **Facturas automáticas** para reservas confirmadas/completadas

Facturas Generadas:

- Facturas PAID para bookings COMPLETED
- Facturas PENDING para bookings CONFIRMED
- Números únicos por tenant y año
- Fechas de vencimiento (30 días)

🔧 Tecnologías Utilizadas

- **Next.js 16** - Framework React
- **Prisma** - ORM para base de datos
- **PostgreSQL** - Base de datos
- **@react-pdf/renderer** - Generación de PDFs
- **react-hook-form** - Gestión de formularios
- **Zod** - Validación de esquemas
- **date-fns** - Manejo de fechas
- **react-hot-toast** - Notificaciones
- **Tailwind CSS** - Estilos

📦 Dependencias Instaladas

```
{
  "@react-pdf/renderer": "^latest"
}
```

🚀 Próximos Pasos Recomendados

Mejoras Inmediatas:

1. **Configurar AWS S3** para almacenamiento de PDFs (código preparado)
2. **Envío de facturas por email** al confirmar reserva
3. **Recordatorios automáticos** para facturas pendientes
4. **Dashboard de análisis** con gráficos de ingresos

Funcionalidades Futuras:

1. **Pagos online** integración con Stripe/PayPal
2. **Facturas recurrentes** para alquileres a largo plazo
3. **Notas de crédito** para devoluciones
4. **Exportación** de facturas a Excel/CSV
5. **Informes financieros** mensuales/anuales
6. **Multi-moneda** para tenants internacionales
7. **Plantillas personalizables** de facturas por tenant

🐛 Troubleshooting

Error: Database not reachable

Solución: Verificar que PostgreSQL está corriendo y la `DATABASE_URL` es correcta.

Error: Can’t generate PDF

Solución: Verificar que la carpeta `/public/invoices/` existe y tiene permisos de escritura.

Error: Invoice already exists

Solución: Cada booking solo puede tener una factura. Verificar en la base de datos.

Error: Email already exists

Solución: Los emails deben ser únicos por tenant.

📝 Notas de Implementación

Seguridad

- ✅ Todas las rutas requieren autenticación

- ✅ Aislamiento por tenant en todas las operaciones
- ✅ Validación de ownership antes de operaciones
- ✅ Sanitización de inputs con Zod

Performance

- ✅ Índices en campos de búsqueda
- ✅ Paginación preparada (actualmente muestra todos)
- ✅ Queries optimizadas con includes
- ✅ Generación de PDF asíncrona

Preparación para Producción

- ✅ Estructura preparada para AWS S3
- ✅ Variables de entorno configurables
- ✅ Error handling completo
- ✅ Logging de operaciones importantes

👥 Autores

Sistema desarrollado como parte del MVP de Rental Management.

📄 Licencia

Este proyecto es parte del sistema propietario Rental Management.