Implementación del Sistema de Autenticación Multi-Tenant

✅ Completado

Se ha implementado exitosamente un sistema completo de autenticación multi-tenant con NextAuth.js para Rental Management.

📦 Componentes Implementados

1. Configuración de NextAuth.js

Archivo: `/app/api/auth/[...nextauth]/route.ts`
- ✅ CredentialsProvider configurado para login con email/password
- ✅ Prisma Adapter para gestión de sesiones
- ✅ Validación de tenant en el proceso de autorización
- ✅ Contraseñas hasheadas con bcryptjs (12 rounds)
- ✅ JWT con tenant_id y role incluidos en la sesión

2. Tipos TypeScript Extendidos

Archivo: `/types/next-auth.d.ts`
- ✅ Session extendida con: `id`, `tenant_id`, `role`, `email`, `name`
- ✅ User extendida con los mismos campos
- ✅ JWT extendido para mantener consistencia
- ✅ Type-safety completo en toda la aplicación

3. Utilidades de Autenticación

Archivo: `/lib/auth.ts`
- ✅ `getServerSession()`: Wrapper para obtener sesión en server components
- ✅ `requireAuth()`: Proteger rutas, redirige a /login si no autenticado
- ✅ `requireRole()`: Verificar permisos por rol(es)
- ✅ `getTenantFromSession()`: Extraer tenant_id de la sesión
- ✅ `hasRole()`: Verificar si el usuario tiene un rol específico
- ✅ `isAdminOrOwner()`: Helper para verificar roles administrativos

4. Proveedor de Contexto de Tenant

Archivo: `/components/providers/tenant-provider.tsx`
- ✅ Context de React para datos del tenant
- ✅ Obtiene tenant desde API basado en subdominio
- ✅ Estados de carga y error manejados
- ✅ Hook `useTenant()` para acceso fácil

5. API Routes

Registro de Usuario

Archivo: `/app/api/auth/register/route.ts`
- ✅ Validación con Zod schema

- ✅ Verificación de email único por tenant
- ✅ Hash de contraseñas con bcryptjs
- ✅ Primer usuario del tenant se convierte en OWNER
- ✅ Usuarios subsiguientes son OPERATOR por defecto
- ✅ Manejo completo de errores

Información del Tenant

Archivo: `/app/api/tenant/current/route.ts`
- ✅ Endpoint para obtener información del tenant actual
- ✅ Basado en subdominio del request

6. Páginas de Autenticación

Login

Archivo: `/app/login/page.tsx`
- ✅ Formulario con validación usando react-hook-form + Zod
- ✅ Manejo de errores de autenticación
- ✅ Diseño responsive con Tailwind CSS
- ✅ Link a página de registro
- ✅ Redirección automática a dashboard tras login exitoso

Registro

Archivo: `/app/register/page.tsx`
- ✅ Formulario completo con validación
- ✅ Confirmación de contraseña
- ✅ Validación en tiempo real
- ✅ Diseño responsive
- ✅ Link a página de login
- ✅ Redirección a login tras registro exitoso

Dashboard

Archivo: `/app/dashboard/page.tsx`
- ✅ Página protegida que requiere autenticación
- ✅ Muestra información del usuario y tenant
- ✅ Confirmación de sistema multi-tenant funcionando

Unauthorized

Archivo: `/app/unauthorized/page.tsx`
- ✅ Página de error para accesos no autorizados
- ✅ Link de regreso al dashboard

7. Componentes de UI

Navbar

Archivo: `/components/layout/navbar.tsx`
- ✅ Logo del tenant (si está configurado)
- ✅ Nombre del usuario logueado
- ✅ Badge con rol del usuario
- ✅ Botón de logout funcional
- ✅ Enlaces a secciones principales
- ✅ Estado de carga mientras verifica sesión

Session Provider

Archivo: `/components/providers/session-provider.tsx`
- ✅ Wrapper de NextAuth SessionProvider
- ✅ Integrado en layout raíz

8. Middleware Mejorado

Archivo: `/middleware.ts`
- ✅ Detección de subdominios mantenida
- ✅ Verificación de autenticación con NextAuth JWT
- ✅ Redirección a /login para rutas protegidas
- ✅ Redirección a /dashboard para usuarios autenticados en /login
- ✅ Lista de rutas públicas configurada
- ✅ Manejo de callbackUrl para redirección post-login

9. Layout Raíz Actualizado

Archivo: `/app/layout.tsx`
- ✅ SessionProvider envuelve toda la aplicación
- ✅ TenantProvider proporciona contexto de tenant
- ✅ Navbar incluido globalmente
- ✅ Metadata actualizado

10. Scripts de Base de Datos

Seed Script

Archivo: `/prisma/seed.ts`
- ✅ Crea dos tenants de prueba (demo y test)
- ✅ Crea usuarios con diferentes roles para testing
- ✅ Contraseñas hasheadas
- ✅ Instrucciones de prueba incluidas
- ✅ Configurado en package.json

Scripts NPM Agregados

- ✅ `npm run db:migrate` - Ejecutar migraciones
- ✅ `npm run db:seed` - Poblar base de datos con datos de prueba
- ✅ `npm run db:studio` - Abrir Prisma Studio
- ✅ `npm run db:reset` - Resetear base de datos

11. Documentación

Guía de Autenticación

Archivo: `/AUTH_GUIDE.md`
- ✅ Explicación completa de la arquitectura
- ✅ Flujos de autenticación documentados
- ✅ Guía de roles y permisos
- ✅ Ejemplos de código
- ✅ Instrucciones de testing local
- ✅ Troubleshooting
- ✅ Consideraciones de seguridad

Resumen de Implementación

Archivo: `/IMPLEMENTATION_SUMMARY.md` (este archivo)

🔐 Características de Seguridad

1. **Aislamiento Multi-Tenant**
   - Usuarios solo pueden autenticarse en su tenant específico
   - Validación de tenant en cada login
   - Session incluye tenant_id para todas las operaciones

2. **Contraseñas Seguras**
   - Hasheadas con bcryptjs (12 rounds)
   - Nunca almacenadas en texto plano
   - Validación de mínimo 8 caracteres

3. **Sesiones JWT**
   - Firmadas con NEXTAUTH_SECRET
   - Cookies httpOnly
   - Incluyen información esencial: id, tenant_id, role, email

4. **Protección de Rutas**
   - Middleware verifica autenticación en todas las rutas protegidas
   - Server components pueden usar `requireAuth()` y `requireRole()`
   - Redirección automática a login

🎯 Roles Implementados

El sistema soporta 5 roles (definidos en Prisma):
- `SUPER_ADMIN` - Acceso a todos los tenants (futuro)
- `OWNER` - Dueño del tenant, acceso completo
- `ADMIN` - Administrador, gestión completa del tenant
- `OPERATOR` - Operador, operaciones del día a día
- `MECHANIC` - Mecánico, mantenimiento de items

🧪 Datos de Prueba

Tenant Demo (subdomain: demo)

```
- owner@demo.com / password123 (OWNER)
- admin@demo.com / password123 (ADMIN)
- operator@demo.com / password123 (OPERATOR)
```

Tenant Test (subdomain: test)

```
- owner@test.com / password123 (OWNER)
```

🚀 Cómo Probar

1. Configurar Base de Datos

```
# En /home/ubuntu/rental_management

# Ejecutar migraciones
npm run db:migrate

# Poblar con datos de prueba
npm run db:seed
```

2. Configurar Hosts Locales

Agregar a `/etc/hosts` (Linux/Mac):

```
127.0.0.1  demo.localhost
127.0.0.1  test.localhost
```

O usar ngrok para subdominios reales.

3. Iniciar Servidor

```
npm run dev
```

4. Probar Flujo Completo

1. Abrir `http://demo.localhost:3000`
2. Sistema redirige a `/login`
3. Hacer click en “crear una cuenta nueva”
4. Registrar un nuevo usuario
5. Login con las credenciales
6. Ver dashboard con información del tenant y usuario

5. Verificar Aislamiento Multi-Tenant

1. Abrir `http://test.localhost:3000`
2. Intentar login con credenciales del tenant “demo”
3. Debe fallar con “Credenciales inválidas”
4. Login con credenciales del tenant “test”
5. Éxito - confirma aislamiento funciona

📝 Próximos Pasos Sugeridos

1. **Recuperación de Contraseña**
   - Implementar email de reset
   - Tokens temporales con expiración

2. **Autenticación 2FA**
   - TOTP con Google Authenticator
   - SMS backup

3. **Invitaciones de Usuario**
   - Owners/Admins pueden invitar usuarios
   - Tokens de invitación con expiración
   - Setup de contraseña en primer login

4. **Logs de Auditoría**
   - Registrar todos los logins
   - Registrar cambios de permisos
   - Dashboard de actividad de usuarios

5. **Permisos Granulares**
   - Más allá de roles, permisos específicos
   - Por módulo (items, bookings, etc.)
   - Configurables por tenant

6. **OAuth Providers**
   - Google Sign-In
   - Microsoft Azure AD
   - GitHub (para desarrollo)

🔧 Tecnologías Utilizadas

- **Next.js 16** - Framework React con App Router
- **NextAuth.js v4** - Sistema de autenticación
- **Prisma 5.x** - ORM y gestión de base de datos
- **PostgreSQL** - Base de datos relacional
- **bcryptjs** - Hash de contraseñas
- **React Hook Form** - Gestión de formularios
- **Zod** - Validación de schemas
- **Tailwind CSS** - Estilos y diseño responsive
- **TypeScript** - Type safety

✨ Resumen

El sistema de autenticación multi-tenant está completamente funcional con:
- ✅ Login/Registro de usuarios
- ✅ Aislamiento por tenant
- ✅ Control de acceso por roles
- ✅ Protección de rutas
- ✅ UI completa y responsive
- ✅ Datos de prueba listos
- ✅ Documentación exhaustiva
- ✅ Build exitoso sin errores TypeScript

Estado: ✅ LISTO PARA DESARROLLO Y PRUEBAS

Desarrollado para Rental Management - Sistema SaaS Multi-Tenant
Fecha: Noviembre 27, 2025