Solución al Error “Tenant No Encontrado”

📋 Diagnóstico del Problema

Causa Raíz

La aplicación Rental Management utiliza un sistema multi-tenant que identifica el tenant a través de un header HTTP `x-tenant-subdomain`. Este header **no se estaba estableciendo correctamente** cuando se accedía a través de la URL de Vercel:

```
https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app
```

Flujo del Error

1. Usuario accede a la URL de Vercel
2. El middleware no detectaba ningún subdomain válido
3. No se establecía el header `x-tenant-subdomain`
4. NextAuth intentaba autenticar sin tenant → **“Tenant no encontrado”**

Tenants Existentes en la Base de Datos

```
✅ 4 Tenants encontrados en Neon:

1. Demo Rental Company
   - Subdomain: demo
   - ID: 151f121d-f319-4a95-8817-d257f7975c3e

2. Test Company
   - Subdomain: test
   - ID: 8fe1763a-7ec6-4c77-a3e1-748a0a640026

3. Scooters Madrid
   - Subdomain: scooters-madrid
   - ID: 2a26f13d-a3dd-4071-9934-8d01ee35ed79

4. Boats Marbella
   - Subdomain: boats-marbella
   - ID: 22369218-b459-4678-b365-1607e58f351c
```

✅ Solución Implementada

1. Modificación del Middleware (`middleware.ts`)

Se agregó lógica de detección de tenant con múltiples estrategias de fallback:

```
/**
 * Prioridad de detección de tenant:
 * 1. Parámetro de query: ?tenant=xxx
 * 2. Subdomain: demo.example.com → "demo"
 * 3. Variable de entorno: DEFAULT_TENANT_SUBDOMAIN
 * 4. Fallback por defecto: "demo"
 */
function getTenantSubdomain(req: NextRequest): string {
  // 1. Query parameter
  const tenantParam = req.nextUrl.searchParams.get('tenant');
  if (tenantParam) {
    return tenantParam;
  }

  // 2. Subdomain
  const hostname = req.headers.get('host') || '';
  const parts = hostname.split('.');

  if (parts.length >= 3 && parts[0] !== 'www') {
    if (!hostname.includes('vercel.app') || !hostname.includes('-')) {
      return parts[0];
    }
  }

  // 3. Environment variable or 4. Fallback
  return process.env.DEFAULT_TENANT_SUBDOMAIN || 'demo';
}
```

El middleware ahora:
- ✅ Establece el header `x-tenant-subdomain` en **todas las requests**
- ✅ Funciona para rutas públicas (`/login` , `/api/auth`) y protegidas
- ✅ Permite especificar tenant por query parameter: `?tenant=test`
- ✅ Usa “demo” como fallback por defecto

2. Nueva Variable de Entorno

Se agregó soporte para:

```
DEFAULT_TENANT_SUBDOMAIN=demo
```

🚀 Pasos para Desplegar la Solución

Paso 1: Hacer Push de los Cambios

```
cd /home/ubuntu/rental_management

# Los cambios ya están en commit local:
git log --oneline -1
# 7a2e21c Fix: Add tenant detection middleware with fallback to 'demo'

# Hacer push (requiere autenticación)
git push origin main
```

Si el push falla (problema de autenticación):
1. Ve a: https://github.com/jovaik/rental-management
2. Usa GitHub Desktop o CLI autenticado
3. O sube los archivos manualmente:
- `middleware.ts`
- `.env.example`

Paso 2: Configurar Variable de Entorno en Vercel

Opción A: Desde la Dashboard de Vercel

1. Ve a: https://vercel.com/jovaiks-projects/rental-management
2. Click en **Settings** → **Environment Variables**
3. Agregar nueva variable:
   `Name: DEFAULT_TENANT_SUBDOMAIN`
   `Value: demo`
   `Environment: Production, Preview, Development (todos)`
4. Click **Save**

Opción B: Desde Vercel CLI

```
vercel env add DEFAULT_TENANT_SUBDOMAIN
# Cuando pregunte el valor, escribe: demo
# Seleccionar: Production, Preview, Development
```

Paso 3: Redeploy en Vercel

Opción A: Deploy Automático

Si Vercel está conectado a GitHub, el push automáticamente triggereará un deploy.

Opción B: Deploy Manual

```
cd /home/ubuntu/rental_management

# Login a Vercel (si no estás autenticado)
npx vercel login

# Deploy a producción
npx vercel --prod
```

🧪 Verificación y Pruebas

1. Acceder a la Aplicación

Una vez desplegado, accede a:

```
https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app
```

2. Pruebas de Autenticación

Intenta hacer login con las credenciales del tenant “demo”:

Usuarios disponibles en el tenant “demo”:

```
Email: demo@example.com (u otro admin del tenant demo)
Password: (según tu seed o configuración)
```

3. Acceso con Diferentes Tenants

Puedes acceder a otros tenants usando el query parameter:

```
# Tenant Test
https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app?tenant=test

# Tenant Scooters Madrid
https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app?tenant=scooters-madrid

# Tenant Boats Marbella
https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app?tenant=boats-marbella
```

🔍 Debugging

Verificar que el Header se Está Estableciendo

Agrega logging temporal en `middleware.ts`:

```
function middleware(request: NextRequest) {
  const tenantSubdomain = getTenantSubdomain(request);
  console.log('🏢 Tenant detectado:', tenantSubdomain);

  // ... resto del código
}
```

Ver logs en Vercel:

```
vercel logs
```

Verificar Tenant en Base de Datos

Ejecutar el script de verificación:

```
cd /home/ubuntu/rental_management
node verify_tenants.js
```

📌 URLs Importantes

- **Aplicación**: https://rental-management-pkjgwm09m-jovaiks-projects.vercel.app

- **Vercel Dashboard**: https://vercel.com/jovaiks-projects/rental-management
- **GitHub Repo**: https://github.com/jovaik/rental-management
- **Neon Dashboard**: https://console.neon.tech

🎯 Próximos Pasos Opcionales

1. Usar un Dominio Personalizado

Si quieres usar subdominios reales (e.g., `demo.tuempresa.com`):

1. Configurar dominio en Vercel
2. Agregar registros DNS wildcard:
   `*.tuempresa.com → CNAME → cname.vercel-dns.com`
3. El middleware automáticamente detectará el subdomain

2. Crear Usuario Admin para el Tenant Demo

Si no existe, crear uno:

```
cd /home/ubuntu/rental_management
node -e "
const { PrismaClient } = require('@prisma/client');
const bcrypt = require('bcryptjs');
const prisma = new PrismaClient();

async function createAdmin() {
  const tenant = await prisma.tenant.findUnique({
    where: { subdomain: 'demo' }
  });

  if (!tenant) {
    console.log('❌ Tenant demo not found');
    return;
  }

  const hashedPassword = await bcrypt.hash('admin123', 10);

  const user = await prisma.user.create({
    data: {
      id: crypto.randomUUID(),
      tenantId: tenant.id,
      email: 'admin@demo.com',
      name: 'Admin Demo',
      password: hashedPassword,
      role: 'OWNER',
      isActive: true,
    }
  });

  console.log('✅ Usuario creado:', user.email);
}

createAdmin().finally(() => prisma.\$disconnect());
"
```

📊 Resumen de Cambios

Archivo	Cambio	Propósito
`middleware.ts`	Agregada función `getTenant-Subdomain()`	Detectar tenant con fallbacks
`middleware.ts`	Modificado middleware principal	Establecer header para todas las requests
`.env.example`	Agregado `DEFAULT_TENANT_SUBDOMAIN`	Documentar nueva variable de entorno

✅ Checklist de Deployment

- [x] Commit creado localmente (7a2e21c)
- [ ] Push a GitHub completado
- [ ] Variable `DEFAULT_TENANT_SUBDOMAIN=demo` configurada en Vercel
- [ ] Deploy a producción ejecutado
- [ ] Verificación: Login funciona en la URL de Vercel
- [ ] Verificación: Acceso con `?tenant=test` funciona

Fecha de Solución: 28 de Noviembre de 2025
Versión: 1.0
Estado: ✅ Código listo, pendiente deployment