🔧 SCRIPTS DE CONTINUIDAD - Sistema ERP

🚀 COMANDOS DE SETUP INICIAL

1. Configuración del Entorno

```
#!/bin/bash
# Script: setup_environment.sh

# Navegar al proyecto
cd /home/ubuntu/sistema_erp_completo/app

# Verificar Node.js y Yarn
echo "Verificando dependencias del sistema..."
node --version
yarn --version

# Instalar dependencias del proyecto
echo "Instalando dependencias del proyecto..."
yarn install

# Generar cliente de Prisma
echo "Generando cliente de Prisma..."
npx prisma generate

# Aplicar schema a la base de datos
echo "Aplicando schema de base de datos..."
npx prisma db push

# Cargar datos de prueba
echo "Cargando datos de prueba..."
yarn prisma db seed

echo "✅ Setup completo. El proyecto está listo para desarrollar."
```

2. Verificación del Estado del Proyecto

```
#!/bin/bash
# Script: verify_project.sh

cd /home/ubuntu/sistema_erp_completo/app

echo "🔍 Verificando estado del proyecto..."

# Verificar TypeScript
echo "Compilando TypeScript..."
npx tsc --noEmit
if [ $? -eq 0 ]; then
    echo "✅ TypeScript: Sin errores"
else
    echo "❌ TypeScript: Errores encontrados"
fi

# Verificar build
echo "Probando build de producción..."
yarn build
if [ $? -eq 0 ]; then
    echo "✅ Build: Exitoso"
else
    echo "❌ Build: Falló"
fi

# Verificar conexión a BD
echo "Verificando conexión a base de datos..."
npx prisma db push --preview-feature
if [ $? -eq 0 ]; then
    echo "✅ Base de datos: Conectada"
else
    echo "❌ Base de datos: Error de conexión"
fi

echo "📊 Verificación completa."
```

🗄️ SCRIPTS DE BASE DE DATOS

3. Backup de Base de Datos

```
#!/bin/bash
# Script: backup_database.sh

# Configuración
BACKUP_DIR="/home/ubuntu/sistema_erp_completo/backups"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="erp_backup_$DATE.sql"

# Crear directorio de backups si no existe
mkdir -p $BACKUP_DIR

# Realizar backup
echo "🗄️ Creando backup de la base de datos..."
pg_dump $DATABASE_URL > "$BACKUP_DIR/$BACKUP_FILE"

if [ $? -eq 0 ]; then
    echo "✅ Backup creado exitosamente: $BACKUP_FILE"
    echo "📍 Ubicación: $BACKUP_DIR/$BACKUP_FILE"
else
    echo "❌ Error al crear el backup"
fi

# Limpiar backups antiguos (mantener solo los últimos 10)
echo "🧹 Limpiando backups antiguos..."
ls -t $BACKUP_DIR/erp_backup_*.sql | tail -n +11 | xargs rm -f
echo "✅ Limpieza completada"
```

4. Restaurar Base de Datos

```
#!/bin/bash
# Script: restore_database.sh
# Uso: ./restore_database.sh backup_file.sql

if [ -z "$1" ]; then
    echo "❌ Error: Especifica el archivo de backup"
    echo "Uso: $0 backup_file.sql"
    exit 1
fi

BACKUP_FILE="$1"

if [ ! -f "$BACKUP_FILE" ]; then
    echo "❌ Error: El archivo $BACKUP_FILE no existe"
    exit 1
fi

echo "⚠️  ADVERTENCIA: Esto restaurará la base de datos desde $BACKUP_FILE"
echo "Todos los datos actuales se perderán."
read -p "¿Continuar? (y/N): " -n 1 -r
echo

if [[ $REPLY =~ ^[Yy]$ ]]; then
    echo "🔄 Restaurando base de datos..."
    psql $DATABASE_URL < "$BACKUP_FILE"

    if [ $? -eq 0 ]; then
        echo "✅ Restauración completada exitosamente"

        # Regenerar cliente de Prisma
        cd /home/ubuntu/sistema_erp_completo/app
        npx prisma generate
        echo "✅ Cliente de Prisma regenerado"
    else
        echo "❌ Error durante la restauración"
    fi
else
    echo "❌ Restauración cancelada"
fi
```

5. Migración de Desarrollo

```bash
#!/bin/bash
# Script: create_migration.sh
# Uso: ./create_migration.sh "nombre-de-la-migracion"

if [ -z "$1" ]; then
    echo "❌ Error: Especifica el nombre de la migración"
    echo "Uso: $0 'nombre-de-la-migracion'"
    exit 1
fi

MIGRATION_NAME="$1"

cd /home/ubuntu/sistema_erp_completo/app

echo "🔄 Creando migración: $MIGRATION_NAME"

# Crear la migración
npx prisma migrate dev --name "$MIGRATION_NAME"

if [ $? -eq 0 ]; then
    echo "✅ Migración creada exitosamente"

    # Regenerar cliente
    npx prisma generate
    echo "✅ Cliente de Prisma actualizado"

    # Verificar que el proyecto compile
    echo "🔍 Verificando compilación..."
    npx tsc --noEmit

    if [ $? -eq 0 ]; then
        echo "✅ Compilación exitosa"
    else
        echo "❌ Errores de compilación encontrados"
    fi
else
    echo "❌ Error al crear la migración"
fi
```

🚀 SCRIPTS DE DESARROLLO

6. Desarrollo Rápido

```
#!/bin/bash
# Script: dev_start.sh

cd /home/ubuntu/sistema_erp_completo/app

echo "🚀 Iniciando servidor de desarrollo..."

# Verificar que las dependencias estén instaladas
if [ ! -d "node_modules" ]; then
    echo "📦 Instalando dependencias..."
    yarn install
fi

# Verificar cliente de Prisma
echo "🔄 Verificando cliente de Prisma..."
npx prisma generate

# Aplicar cambios de schema si existen
echo "🗄️ Aplicando cambios de schema..."
npx prisma db push

# Iniciar servidor de desarrollo
echo "🌐 Servidor disponible en: http://localhost:3000"
echo "👤 Usuario de prueba: admin@sistema.com / 123456"
yarn dev
```

7. Testing Completo

```
#!/bin/bash
# Script: full_test.sh

cd /home/ubuntu/sistema_erp_completo/app

echo "🧪 Ejecutando suite completa de tests..."

# Test de TypeScript
echo "1️⃣ Verificando tipos TypeScript..."
npx tsc --noEmit
TS_EXIT_CODE=$?

# Test de Linting
echo "2️⃣ Ejecutando linter..."
yarn lint --max-warnings 0
LINT_EXIT_CODE=$?

# Test de Build
echo "3️⃣ Probando build de producción..."
yarn build
BUILD_EXIT_CODE=$?

# Test de conexión a BD
echo "4️⃣ Verificando conexión a base de datos..."
npx prisma db push --preview-feature > /dev/null 2>&1
DB_EXIT_CODE=$?

# Test de seed
echo "5️⃣ Probando datos de prueba..."
yarn prisma db seed > /dev/null 2>&1
SEED_EXIT_CODE=$?

# Resumen de resultados
echo ""
echo "📊 RESUMEN DE TESTS:"
echo "================================"
[ $TS_EXIT_CODE -eq 0 ] && echo "✅ TypeScript" || echo "❌ TypeScript"
[ $LINT_EXIT_CODE -eq 0 ] && echo "✅ Linting" || echo "❌ Linting"
[ $BUILD_EXIT_CODE -eq 0 ] && echo "✅ Build" || echo "❌ Build"
[ $DB_EXIT_CODE -eq 0 ] && echo "✅ Base de datos" || echo "❌ Base de datos"
[ $SEED_EXIT_CODE -eq 0 ] && echo "✅ Datos de prueba" || echo "❌ Datos de prueba"

# Exit code general
TOTAL_ERRORS=$((TS_EXIT_CODE + LINT_EXIT_CODE + BUILD_EXIT_CODE + DB_EXIT_CODE + SEED_
EXIT_CODE))

if [ $TOTAL_ERRORS -eq 0 ]; then
    echo ""
    echo "🎉 ¡Todos los tests pasaron exitosamente!"
    exit 0
else
    echo ""
    echo "⚠️ Algunos tests fallaron. Revisar errores arriba."
    exit 1
fi
```

📊 SCRIPTS DE MONITOREO

8. Estado del Sistema

```
#!/bin/bash
# Script: system_status.sh

cd /home/ubuntu/sistema_erp_completo/app

echo "📊 ESTADO DEL SISTEMA ERP"
echo "=========================="

# Información del proyecto
echo "📁 Proyecto: $(pwd)"
echo "📅 Fecha: $(date)"

# Estado de la aplicación
echo ""
echo "🔧 ESTADO DE LA APLICACIÓN:"
if pgrep -f "next dev" > /dev/null; then
    echo "✅ Servidor de desarrollo: Ejecutándose"
    echo "🌐 URL: http://localhost:3000"
else
    echo "❌ Servidor de desarrollo: Detenido"
fi

# Estado de la base de datos
echo ""
echo "🗄️ ESTADO DE LA BASE DE DATOS:"
if npx prisma db push --preview-feature > /dev/null 2>&1; then
    echo "✅ Conexión a BD: Exitosa"

    # Contar registros principales
    echo "📊 Registros en BD:"
    echo "   - Usuarios: $(npx prisma db seed --preview 2>/dev/null | grep -c "user" |
| echo "N/A")"
    echo "   - Clientes: $(npx prisma db seed --preview 2>/dev/null | grep -c
"client" || echo "N/A")"
else
    echo "❌ Conexión a BD: Error"
fi

# Estado de archivos importantes
echo ""
echo "📁 ARCHIVOS IMPORTANTES:"
[ -f ".env" ] && echo "✅ .env" || echo "❌ .env"
[ -f "prisma/schema.prisma" ] && echo "✅ schema.prisma" || echo "❌ schema.prisma"
[ -f "package.json" ] && echo "✅ package.json" || echo "❌ package.json"

# Espacio en disco
echo ""
echo "💾 ESPACIO EN DISCO:"
df -h . | tail -1 | awk '{print "   Usado: " $3 " de " $2 " (" $5 ")"}'

echo ""
echo "✅ Revisión de estado completada"
```

9. Deployment a Producción

```
#!/bin/bash
# Script: deploy_production.sh

cd /home/ubuntu/sistema_erp_completo/app

echo "🚀 PREPARANDO DEPLOYMENT A PRODUCCIÓN"
echo "======================================"

# Pre-deployment checks
echo "1️⃣ Ejecutando verificaciones pre-deployment..."

# Test completo
if ! ./scripts/full_test.sh; then
    echo "❌ Tests fallaron. Deployment cancelado."
    exit 1
fi

# Build de producción
echo "2️⃣ Creando build de producción..."
yarn build

if [ $? -ne 0 ]; then
    echo "❌ Build falló. Deployment cancelado."
    exit 1
fi

# Aplicar migraciones de producción
echo "3️⃣ Aplicando migraciones de producción..."
npx prisma migrate deploy

if [ $? -ne 0 ]; then
    echo "❌ Migraciones fallaron. Deployment cancelado."
    exit 1
fi

echo "✅ Proyecto listo para deployment"
echo ""
echo "📋 SIGUIENTE PASOS:"
echo "   1. Configurar variables de entorno de producción"
echo "   2. Configurar dominio y SSL"
echo "   3. Ejecutar: yarn start"
echo ""
echo "🎉 ¡Deployment exitoso!"
```

🔧 SCRIPTS DE UTILIDAD

10. Crear Nuevo Módulo

```
#!/bin/bash
# Script: create_module.sh
# Uso: ./create_module.sh "nombre-del-modulo"

if [ -z "$1" ]; then
    echo "❌ Error: Especifica el nombre del módulo"
    echo "Uso: $0 'nombre-del-modulo'"
    exit 1
fi

MODULE_NAME="$1"
MODULE_PATH="/home/ubuntu/sistema_erp_completo/app/app/$MODULE_NAME"

echo "🏗️ Creando módulo: $MODULE_NAME"

# Crear directorio del módulo
mkdir -p "$MODULE_PATH"

# Crear archivos básicos
cat > "$MODULE_PATH/page.tsx" << EOF
'use client';

import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';

export default function ${MODULE_NAME^}Page() {
  return (
    <div className="space-y-4">
      <div>
        <h1 className="text-3xl font-bold tracking-tight">${MODULE_NAME^}</h1>
        <p className="text-muted-foreground">
          Gestión de ${MODULE_NAME}
        </p>
      </div>

      <Card>
        <CardHeader>
          <CardTitle>Módulo ${MODULE_NAME^}</CardTitle>
        </CardHeader>
        <CardContent>
          <p>Contenido del módulo ${MODULE_NAME}</p>
        </CardContent>
      </Card>
    </div>
  );
}
EOF

# Crear API route
mkdir -p "/home/ubuntu/sistema_erp_completo/app/app/api/$MODULE_NAME"
cat > "/home/ubuntu/sistema_erp_completo/app/app/api/$MODULE_NAME/route.ts" << EOF
import { NextRequest, NextResponse } from 'next/server';
import { getServerSession } from 'next-auth/next';
import { authOptions } from '@/lib/auth';

export async function GET(request: NextRequest) {
  try {
    const session = await getServerSession(authOptions);
    if (!session) {
      return NextResponse.json({ error: 'No autorizado' }, { status: 401 });
    }

    // TODO: Implementar lógica de GET para ${MODULE_NAME}
```

```
    return NextResponse.json({
      message: 'Endpoint ${MODULE_NAME} funcionando',
      data: []
    });
  } catch (error) {
    return NextResponse.json(
      { error: 'Error interno del servidor' },
      { status: 500 }
    );
  }
}

export async function POST(request: NextRequest) {
  try {
    const session = await getServerSession(authOptions);
    if (!session) {
      return NextResponse.json({ error: 'No autorizado' }, { status: 401 });
    }

    // TODO: Implementar lógica de POST para ${MODULE_NAME}

    return NextResponse.json({
      message: 'Elemento ${MODULE_NAME} creado',
      data: {}
    });
  } catch (error) {
    return NextResponse.json(
      { error: 'Error interno del servidor' },
      { status: 500 }
    );
  }
}
EOF

echo "✅ Módulo $MODULE_NAME creado exitosamente"
echo "📁 Ubicación: $MODULE_PATH"
echo "🔗 API: /api/$MODULE_NAME"
echo ""
echo "📋 TODO:"
echo "   1. Agregar el módulo al sidebar de navegación"
echo "   2. Implementar la lógica de las APIs"
echo "   3. Crear componentes específicos del módulo"
echo "   4. Agregar validaciones y permisos"
```

🔧 Scripts listos para acelerar el desarrollo y mantenimiento del sistema