🚀 GUÍA DE CONTINUACIÓN DEL DESARROLLO

🎯 CÓMO CONTINUAR DESARROLLANDO

1. Acceso al Proyecto

```
# Ubicación del proyecto
cd /home/ubuntu/sistema_erp_completo/app

# Instalar dependencias (si es necesario)
yarn install

# Ejecutar en desarrollo
yarn dev

# Compilar para producción
yarn build
```

2. Credenciales para Testing

```
Admin Principal:
- Email: admin@sistema.com
- Password: 123456

Usuarios de Prueba:
- gestor1@sistema.com / password123
- vendedor1@sistema.com / password123
- analista@sistema.com / password123
```

3. Base de Datos

```
# Ejecutar migraciones
cd /home/ubuntu/sistema_erp_completo/app
npx prisma generate
npx prisma db push

# Cargar datos de prueba
yarn prisma db seed
```

🏗️ PRÓXIMO MÓDULO RECOMENDADO: PRODUCTOS

Razón de la Recomendación:

El módulo de **Productos** es fundamental para completar el ciclo de ventas y debe implementarse antes que otros módulos.

Plan de Implementación:

Paso 1: Completar el Modelo de Datos

```
// Agregar al schema.prisma
model Categoria {
  id          String    @id @default(cuid())
  nombre      String    @unique
  descripcion String?
  isActive    Boolean   @default(true)
  productos   Producto[]
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt
}

model Producto {
  id            String    @id @default(cuid())
  codigo        String    @unique
  nombre        String
  descripcion   String?
  precio        Float     @default(0)
  costo         Float     @default(0)
  stock         Int       @default(0)
  stockMinimo   Int       @default(0)
  unidadMedida  String?
  categoriaId   String?
  categoria     Categoria? @relation(fields: [categoriaId], references: [id])
  isActive      Boolean   @default(true)
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt

  // Relaciones con ventas
  ventasItems   VentaItem[]
}
```

Paso 2: Crear APIs

```
// /api/productos
// - GET: Listar productos con filtros
// - POST: Crear nuevo producto

// /api/productos/[id]
// - GET: Obtener producto específico
// - PUT: Actualizar producto
// - DELETE: Eliminar producto

// /api/productos/categorias
// - GET: Listar categorías
// - POST: Crear categoría
```

Paso 3: Componentes UI

```
// Componentes a crear:
// - ProductosList.tsx
// - ProductoCard.tsx
// - ProductoForm.tsx
// - ProductoModal.tsx
// - CategoriaSelect.tsx
// - StockAlert.tsx
```

Paso 4: Funcionalidades Clave

1. **CRUD completo de productos**
2. **Gestión de categorías**
3. **Control de stock básico**
4. **Alertas de stock mínimo**
5. **Búsqueda y filtros avanzados**
6. **Importación desde CSV**

🎨 ALTERNATIVAS DE DESARROLLO

Opción A: Completar Dashboard

Tiempo estimado: 2-3 horas
- Implementar gráficos con Chart.js
- Crear widgets dinámicos
- Mostrar KPIs principales
- Métricas en tiempo real

Opción B: Mejorar Módulo de Clientes

Tiempo estimado: 1-2 horas
- Historial de pagos por cliente
- Vista de mapa con geolocalización
- Exportación de reportes
- Comunicación integrada mejorada

Opción C: Sistema de Reportes

Tiempo estimado: 3-4 horas
- Generador de reportes dinámico
- Exportación PDF/Excel
- Reportes programados
- Dashboard de reportes

🔧 COMANDOS ÚTILES

Desarrollo:

```
# Ejecutar servidor de desarrollo
yarn dev

# Verificar tipos de TypeScript
yarn type-check

# Ejecutar linter
yarn lint

# Construir para producción
yarn build
```

Base de Datos:

```
# Ver estado de la base de datos
npx prisma studio

# Generar cliente de Prisma
npx prisma generate

# Aplicar cambios al schema
npx prisma db push

# Crear migración
npx prisma migrate dev

# Cargar datos de prueba
npx prisma db seed
```

Testing:

```
# Verificar que el proyecto compila
yarn build

# Probar en servidor local
yarn start
```

📁 ESTRUCTURA DE ARCHIVOS PARA NUEVOS MÓDULOS

```
app/
├── [nuevo-modulo]/
│   ├── page.tsx          # Página principal
│   ├── layout.tsx        # Layout del módulo
│   ├── loading.tsx       # Estado de carga
│   └── components/       # Componentes específicos
│       ├── Lista.tsx
│       ├── Formulario.tsx
│       └── Modal.tsx
├── api/
│   └── [nuevo-modulo]/
│       ├── route.ts      # GET, POST
│       └── [id]/
│           └── route.ts  # GET, PUT, DELETE
└── components/
    └── [nuevo-modulo]/   # Componentes reutilizables
```

🎯 MÉTRICAS DE ÉXITO

Para el próximo módulo:

- [ ] CRUD completo implementado
- [ ] APIs funcionando correctamente
- [ ] UI/UX consistente con el diseño actual
- [ ] Validaciones completas
- [ ] Manejo de errores robusto
- [ ] Responsive design

- [ ] Testing básico completado

Estimación de tiempo por módulo:

- **Módulo simple (ej: Categorías):** 2-3 horas
- **Módulo medio (ej: Productos):** 4-6 horas
- **Módulo complejo (ej: Ventas):** 6-8 horas

📞 RECOMENDACIONES FINALES

1. **Priorizar funcionalidad sobre perfecta estética**
2. **Mantener consistencia con componentes existentes**
3. **Implementar validaciones robustas**
4. **Probar cada funcionalidad antes de continuar**
5. **Documentar cambios importantes**

🎊 **¡El proyecto está en excelente estado para continuar el desarrollo!**