SETUP_LOCAL_BFF.md - Guía de Implementación Local del BFF

Proyecto: Sistema ERP B2B Móvil - Backend-For-Frontend (BFF)
Directorio: `erp-system/app/`
Versión: v1.0
Actualizado: 24 de agosto de 2025

Esta guía te permitirá configurar y ejecutar localmente el Backend-For-Frontend (BFF) del Sistema ERP B2B Móvil de forma aislada para verificar su funcionalidad.

---

1. Pre-requisitos de Software

1.1 Software Obligatorio

- **Node.js v18.0+** - Descargar desde nodejs.org (https://nodejs.org/)
- **npm v8.0+** - Incluido con Node.js
- **PostgreSQL v13.0+** - Descargar desde postgresql.org (https://www.postgresql.org/download/)
- **Git** - Para clonar el repositorio

1.2 Verificación de Versiones

```
# Ejecutar estos comandos para verificar las instalaciones
node --version    # Debe mostrar v18.0.0 o superior
npm --version     # Debe mostrar v8.0.0 o superior
psql --version    # Debe mostrar PostgreSQL 13.0 o superior
git --version     # Cualquier versión reciente
```

1.3 Herramientas Recomendadas

- **Postman** o **Thunder Client** (VS Code) - Para probar endpoints
- **TablePlus** o **pgAdmin** - Para inspeccionar la base de datos
- **VS Code** con extensiones TypeScript y Prisma

---

2. Guía de Configuración de la Base de Datos

2.1 Crear Usuario y Base de Datos PostgreSQL

Para sistemas Unix/Linux/macOS:

```
-- Conectar como superusuario de PostgreSQL
sudo -u postgres psql

-- Crear usuario dedicado
CREATE USER erp_user WITH PASSWORD 'erp_secure_password_2025';

-- Crear base de datos
CREATE DATABASE erp_b2b_mobile_local
  WITH OWNER erp_user
  ENCODING 'UTF8'
  LC_COLLATE='en_US.UTF-8'
  LC_CTYPE='en_US.UTF-8';

-- Otorgar permisos completos
GRANT ALL PRIVILEGES ON DATABASE erp_b2b_mobile_local TO erp_user;

-- Salir de psql
\q
```

Para Windows (usando Command Prompt como administrador):

```
-- Conectar a PostgreSQL
psql -U postgres

-- Ejecutar los mismos comandos SQL de arriba
CREATE USER erp_user WITH PASSWORD 'erp_secure_password_2025';
CREATE DATABASE erp_b2b_mobile_local WITH OWNER erp_user;
GRANT ALL PRIVILEGES ON DATABASE erp_b2b_mobile_local TO erp_user;
\q
```

2.2 Verificar Conexión

```
# Probar conexión con el nuevo usuario
psql -h localhost -U erp_user -d erp_b2b_mobile_local

# Si funciona, deberías ver algo como:
# erp_b2b_mobile_local=>
# Salir con \q
```

---

3. Configuración del Entorno (.env)

3.1 Ubicación del Archivo

El archivo `.env` debe crearse en la raíz del proyecto BFF:

```
erp-system/app/.env
```

3.2 Plantilla Completa del Archivo .env

```
# ================================
# CONFIGURACIÓN DE BASE DE DATOS
# ================================
DATABASE_URL="postgresql://erp_user:erp_secure_password_2025@localhost:5432/
erp_b2b_mobile_local"

# ================================
# CONFIGURACIÓN DE AUTENTICACIÓN
# ================================
# Clave secreta para firmar tokens JWT (generar una única y segura)
JWT_SECRET="tu-jwt-secret-super-seguro-de-al-menos-32-caracteres-aqui-2025"

# Clave secreta para NextAuth (generar una única y segura)
NEXTAUTH_SECRET="tu-nextauth-secret-super-seguro-de-al-menos-32-caracteres-aqui-2025"

# URL base de la aplicación (para desarrollo local)
NEXTAUTH_URL="http://localhost:3000"

# ================================
# CONFIGURACIÓN DE NOTIFICACIONES PUSH (OPCIONAL)
# ================================
# Firebase Cloud Messaging - Requerido solo si usas notificaciones push
FIREBASE_PROJECT_ID="tu-proyecto-firebase-id"
FIREBASE_PRIVATE_KEY="-----BEGIN PRIVATE KEY-----\nTU_PRIVATE_KEY_AQUI_ENCODED\n-----
END PRIVATE KEY-----\n"
FIREBASE_CLIENT_EMAIL="firebase-adminsdk-xxxxx@tu-proyecto.iam.gserviceaccount.com"

# ================================
# CONFIGURACIÓN DE ALMACENAMIENTO (OPCIONAL)
# ================================
# Almacenamiento de archivos - Para subida de comprobantes de pago
STORAGE_PROVIDER="local"  # 'local' | 's3' | 'gcs'
STORAGE_LOCAL_PATH="./uploads"
STORAGE_PUBLIC_URL="http://localhost:3000/uploads"

# Para AWS S3 (opcional)
# AWS_ACCESS_KEY_ID="tu-aws-access-key"
# AWS_SECRET_ACCESS_KEY="tu-aws-secret-key"
# AWS_S3_BUCKET="tu-bucket-name"
# AWS_S3_REGION="us-east-1"

# ================================
# CONFIGURACIÓN DE DESARROLLO
# ================================
NODE_ENV="development"
PORT=3000
```

3.3 Explicación de Variables Críticas

Variable	Propósito	¿Obligatoria?
`DATABASE_URL`	Conexión a PostgreSQL	✅ Sí
`JWT_SECRET`	Firma de tokens de autenticación	✅ Sí
`NEXTAUTH_SECRET`	Encriptación de sesiones	✅ Sí
`FIREBASE_*`	Notificaciones push móviles	❌ Opcional
`STORAGE_*`	Almacenamiento de archivos	❌ Opcional

3.4 Generar Secretos Seguros

```
# Generar secretos aleatorios seguros (ejecutar dos veces)
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"

# Usar la salida para JWT_SECRET y NEXTAUTH_SECRET
```

---

4. Pasos de Instalación y Ejecución

4.1 Navegar al Directorio del Proyecto

```
# Asumir que ya tienes el código fuente descargado
cd erp-system/app
```

4.2 Instalar Dependencias

```
# Instalar todas las dependencias del proyecto
npm install

# Verificar que se instalaron correctamente
npm list --depth=0
```

4.3 Configurar Prisma y Base de Datos

```
# Generar el cliente de Prisma
npx prisma generate

# Aplicar el schema a la base de datos (primera vez)
npx prisma db push

# Alternativa: Usar migraciones (recomendado para producción)
# npx prisma migrate dev --name init
```

4.4 Poblar Base de Datos con Datos de Prueba

```
# Ejecutar el script de seed
npx prisma db seed
```

Salida esperada del seed:

```
🌱 Iniciando seed de la base de datos...
👤 Creando usuario admin...
🏷️ Creando tipos de cliente...
📂 Creando categorías...
📏 Creando unidades de medida...
🏬 Creando sucursales...
🏪 Creando almacenes...
📦 Creando productos...
👨‍💼 Creando vendedores...
👥 Creando clientes demo...
💰 Creando precios...
📊 Creando inventario...

✅ Seed completado exitosamente!

🔑 Credenciales de prueba creadas:
- Admin: john@doe.com / johndoe123
- Cliente Demo 1: alexander.godoy@example.com / demo123
- Cliente Demo 2: maria.gonzalez@example.com / demo123
```

4.5 Iniciar el Servidor de Desarrollo

```
# Iniciar el servidor
npm run dev

# El servidor debería iniciarse en http://localhost:3000
```

Salida esperada:

```
▲ Next.js 14.2.28
- Local:        http://localhost:3000
- Environments: .env

✓ Starting...
✓ Ready in 2.3s
```

---

5. Verificación Funcional (Pruebas de “Humo”)

5.1 Verificar que el Servidor Responde

```
# Probar que el servidor principal responde
curl -i http://localhost:3000

# Respuesta esperada: Status 200 con HTML del dashboard
```

5.2 Prueba de Endpoint de Pre-Login

```
# Probar verificación de cliente por email
curl -X POST http://localhost:3000/api/auth/mobile/pre-login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "alexander.godoy@example.com"
  }'
```

Respuesta esperada:

```
{
  "success": true,
  "data": {
    "name": "ALEXANDER GODOY CASTELLON",
    "address": "Av. Comercial 789, Ciudad",
    "phone": "70123456",
    "nit": ""
  }
}
```

5.3 Prueba de Endpoint de Login Completo

```
# Probar login completo para obtener tokens
curl -X POST http://localhost:3000/api/auth/mobile/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "alexander.godoy@example.com",
    "password": "demo123",
    "sucursalId": "SUCURSAL_ID_AQUI",
    "almacenId": "ALMACEN_ID_AQUI"
  }'
```

Para obtener los IDs necesarios:

```
# Obtener ID de sucursal
curl http://localhost:3000/api/sucursales

# Obtener ID de almacén (usar sucursalId obtenido arriba)
curl "http://localhost:3000/api/almacenes?sucursalId=SUCURSAL_ID_AQUI"
```

Respuesta esperada del login:

```
{
  "success": true,
  "tokens": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "customer": {
    "id": "...",
    "name": "ALEXANDER GODOY CASTELLON",
    "email": "alexander.godoy@example.com"
  }
}
```

5.4 Prueba de Endpoint Protegido (Productos)

```
# Usar el accessToken obtenido en el paso anterior
export ACCESS_TOKEN="eyJhbGciOiJIUzI1NiIs..."

# Probar endpoint protegido de productos
curl -H "Authorization: Bearer $ACCESS_TOKEN" \
  http://localhost:3000/api/products
```

Respuesta esperada:


```
{
  "success": true,
  "data": {
    "products": [
      {
        "id": "...",
        "name": "Fanta Naranja Vidrio Retornable 190 Ml",
        "sku": "2519",
        "price": "31.20",
        "currency": "BOB"
      }
    ],
    "pagination": {
      "page": 1,
      "totalPages": 1,
      "totalItems": 3
    }
  }
}
```


5.5 Prueba de Endpoint de Creación de Pedido

```
# Crear un pedido de prueba
curl -X POST http://localhost:3000/api/orders \
  -H "Authorization: Bearer $ACCESS_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "items": [
      {
        "productId": "PRODUCT_ID_AQUI",
        "quantity": 2,
        "unitId": "UNIT_ID_AQUI"
      }
    ],
    "observaciones": "Pedido de prueba desde curl"
  }'
```

Respuesta esperada:

```
{
  "success": true,
  "data": {
    "orderId": "...",
    "orderNumber": "ORD-2025-001",
    "total": "62.40",
    "status": "BORRADOR"
  }
}
```

---

6. Verificación Avanzada con Postman

6.1 Colección Postman Recomendada

Crear una nueva colección en Postman con las siguientes requests:

Environment Variables:

```
base_url: http://localhost:3000
access_token: (se actualiza después del login)
```

Request 1: Pre-login

```
POST {{base_url}}/api/auth/mobile/pre-login
Content-Type: application/json

{
  "email": "alexander.godoy@example.com"
}
```

Request 2: Get Sucursales

```
GET {{base_url}}/api/sucursales
```

Request 3: Get Almacenes

```
GET {{base_url}}/api/almacenes?sucursalId={{sucursal_id}}
```

Request 4: Login

```
POST {{base_url}}/api/auth/mobile/login
Content-Type: application/json

{
  "email": "alexander.godoy@example.com",
  "password": "demo123",
  "sucursalId": "{{sucursal_id}}",
  "almacenId": "{{almacen_id}}"
}

# Script para extraer token:
pm.test("Login successful", function () {
    pm.response.to.have.status(200);
    var jsonData = pm.response.json();
    pm.environment.set("access_token", jsonData.tokens.accessToken);
});
```

Request 5: Get Products

```
GET {{base_url}}/api/products
Authorization: Bearer {{access_token}}
```

Request 6: Create Order

```
POST {{base_url}}/api/orders
Authorization: Bearer {{access_token}}
Content-Type: application/json

{
  "items": [
    {
      "productId": "{{product_id}}",
      "quantity": 2,
      "unitId": "{{unit_id}}"
    }
  ],
  "observaciones": "Pedido de prueba desde Postman"
}
```

---

7. Comandos de Desarrollo Útiles

7.1 Gestión de Base de Datos

```
# Ver el estado de migraciones
npx prisma migrate status

# Crear nueva migración
npx prisma migrate dev --name nombre_migracion

# Resetear BD completamente (¡PELIGROSO!)
npx prisma migrate reset

# Reejecutar seed
npx prisma db seed

# Abrir Prisma Studio (interfaz web para BD)
npx prisma studio
```

7.2 Debugging y Logs

```
# Ejecutar con debug de Prisma
DEBUG="prisma*" npm run dev

# Ejecutar con logs detallados de Next.js
npm run dev -- --debug

# Ver logs en tiempo real (si usas PM2)
pm2 logs
```

7.3 Verificación de Tipos

```
# Compilación TypeScript sin ejecutar
npx tsc --noEmit

# Verificar tipos de Prisma
npx prisma generate && npx tsc --noEmit
```

---

8. Troubleshooting Común

8.1 Error: “Cannot find module @prisma/client”

```
# Solución:
cd erp-system/app
npx prisma generate
npm run dev
```

8.2 Error: “Database does not exist”

```
# Verificar que la BD existe
psql -U erp_user -d erp_b2b_mobile_local -c "SELECT version();"

# Si no existe, crearla:
psql -U postgres -c "CREATE DATABASE erp_b2b_mobile_local OWNER erp_user;"
```

8.3 Error: “JWT malformed” o “Invalid token”

```
# Verificar que JWT_SECRET está configurado
grep JWT_SECRET .env

# Regenerar un nuevo secreto si es necesario
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

8.4 Error: “Port 3000 already in use”

```
# Encontrar proceso usando el puerto
lsof -ti :3000

# Matarlo
kill -9 $(lsof -ti :3000)

# O usar otro puerto
PORT=3001 npm run dev
```

8.5 Error de CORS en requests desde app móvil

Verificar que `next.config.js` incluye la configuración CORS:

```
async headers() {
  return [
    {
      source: '/api/:path*',
      headers: [
        { key: 'Access-Control-Allow-Origin', value: '*' },
        { key: 'Access-Control-Allow-Methods', value: 'GET,POST,PUT,DELETE,OPTIONS' },
        { key: 'Access-Control-Allow-Headers', value: 'Content-Type,Authorization' },
      ],
    },
  ];
}
```

---

9. Estructura de Archivos del BFF

```
erp-system/app/
├── app/
│   ├── api/                          # Endpoints de API
│   │   ├── auth/mobile/              # Autenticación móvil
│   │   ├── products/                 # Catálogo de productos
│   │   ├── orders/                   # Gestión de pedidos
│   │   ├── favorites/                # Productos favoritos
│   │   ├── sucursales/               # Configuración sucursales
│   │   └── almacenes/                # Configuración almacenes
│   ├── globals.css                   # Estilos globales
│   ├── layout.tsx                    # Layout principal
│   └── page.tsx                      # Dashboard del BFF
├── lib/                              # Lógica de negocio
│   ├── auth.ts                       # Autenticación JWT
│   ├── business-logic.ts             # Lógica central
│   └── price-calculator.ts           # Cálculo de precios
├── prisma/
│   └── schema.prisma                 # Schema de base de datos
├── scripts/
│   └── seed.ts                       # Script de datos de prueba
├── types/                            # Tipos TypeScript
├── .env                              # Variables de entorno
├── package.json                      # Dependencias y scripts
└── next.config.js                    # Configuración Next.js
```

🎯 Verificación Final Exitosa:

Si has seguido todos los pasos correctamente, deberías tener:

- ✅ BFF ejecutándose en `http://localhost:3000`
- ✅ Base de datos PostgreSQL configurada y poblada
- ✅ Autenticación JWT funcionando
- ✅ Endpoints API respondiendo correctamente
- ✅ Datos de prueba disponibles
- ✅ Dashboard visible con estadísticas

Para soporte técnico adicional, verificar logs del servidor y base de datos.