# Notas de Migración: PostgreSQL → MariaDB

## 🔄 Cambios Realizados

### 1. Schema de Prisma ( `prisma/schema.prisma` )

**Antes (PostgreSQL):**

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

**Después (MariaDB/MySQL):**

```
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}
```

### 2. Tipos de Datos Ajustados

#### Campos String con Índices Únicos

MariaDB/MySQL requiere especificar la longitud para campos indexados:

```
// Agregado @db.VarChar(255) a campos con @unique
email         String?   @unique @db.VarChar(255)
sessionToken  String    @unique @db.VarChar(255)
uniqueToken   String    @unique @db.VarChar(255)
token         String    @unique @db.VarChar(255)
```

#### Campos JSON

- **PostgreSQL**: Usa `JSONB` (binario, optimizado)
- **MariaDB**: Usa `JSON` (texto)

En Prisma, ambos usan el tipo `Json`, pero MariaDB almacena como texto JSON.

**Impacto**: Diferencia mínima de rendimiento. Para la mayoría de casos no es significativo.

### 3. Connection String

**PostgreSQL:**

```
postgresql://usuario:contraseña@host:5432/base_datos
```

**MariaDB/MySQL:**

```
mysql://usuario:contraseña@host:3306/base_datos
```

---

# ⚖️ Diferencias PostgreSQL vs MariaDB

| Característica | PostgreSQL | MariaDB/MySQL |
|---|---|---|
| **JSON** | JSONB (binario) | JSON (texto) |
| **Strings indexados** | Sin límite | Máximo 767 bytes (255 chars UTF8) |
| **Booleanos** | BOOLEAN nativo | TINYINT(1) |
| **Fechas** | TIMESTAMP | DATETIME |
| **Secuencias** | SERIAL | AUTO_INCREMENT |
| **Puerto por defecto** | 5432 | 3306 |

---

# 🛠️ Compatibilidad del Código

## ✅ Sin Cambios Necesarios

El código de la aplicación **NO requiere cambios** porque:

1. **Prisma abstrae las diferencias**: El cliente de Prisma genera el código apropiado para cada base de datos
2. **Tipos compatibles**: Los tipos usados en el schema son compatibles con ambas bases de datos
3. **APIs idénticas**: Las operaciones CRUD son las mismas

## Ejemplo: Sin Cambios en el Código

```
// Este código funciona con PostgreSQL Y MariaDB sin modificaciones
const user = await prisma.user.create({
  data: {
    email: 'admin@easton.cl',
    password: hashedPassword,
    role: 'admin'
  }
});
```

---

# 📊 Rendimiento

## Consultas JSON

**PostgreSQL (JSONB):**

```
-- Indexado y optimizado
SELECT * FROM "SurveyResponse"
WHERE "dynamicAnswers"->>'q1' = '7';
```

**MariaDB (JSON):**

```
-- Funciona pero menos optimizado
SELECT * FROM SurveyResponse
WHERE JSON_EXTRACT(dynamicAnswers, '$.q1') = '7';
```

**Impacto en tu aplicación:** Mínimo. Las consultas JSON son poco frecuentes y el volumen de datos es bajo.

---

# 📝 Comandos de Migración

## Opción A: Base de Datos Nueva (Recomendado)

```
# 1. Configurar DATABASE_URL para MariaDB en .env
# 2. Generar cliente
npx prisma generate

# 3. Crear tablas
npx prisma db push

# 4. Cargar datos iniciales
npm run prisma:seed
```

## Opción B: Migrar Datos Existentes

Si tienes datos en PostgreSQL que quieres migrar:

### 1. Exportar datos de PostgreSQL

```
# Exportar a SQL
pg_dump -U usuario -h host -d easton_surveys > postgres_dump.sql

# O exportar con Prisma Studio a CSV
npx prisma studio
# Exportar cada tabla manualmente
```

### 2. Crear estructura en MariaDB

```
# En tu servidor MariaDB
npx prisma generate
npx prisma db push
```

### 3. Importar datos

**Opción manual (recomendado para datos pequeños):**

```
// script de migración personalizado
import { PrismaClient as PgClient } from '@prisma/client-pg';
import { PrismaClient as MyClient } from '@prisma/client';

const pgClient = new PgClient({
  datasources: { db: { url: 'postgresql://...' } }
});

const myClient = new MyClient({
  datasources: { db: { url: 'mysql://...' } }
});

// Copiar usuarios
const users = await pgClient.user.findMany();
for (const user of users) {
  await myClient.user.create({ data: user });
}

// Copiar clientes
const clients = await pgClient.client.findMany();
for (const client of clients) {
  await myClient.client.create({ data: client });
}

// ... repetir para otras tablas
```

---

# ⚠️ Consideraciones Importantes

## 1. Límite de Longitud de Índices

**MariaDB/MySQL** tiene un límite de 767 bytes para índices (255 caracteres UTF-8).

Por eso agregamos `@db.VarChar(255)` a campos con `@unique`.

**Solución aplicada**: Todos los campos indexados están limitados a 255 caracteres.

## 2. Transacciones

Ambas bases de datos soportan transacciones, pero:

- **PostgreSQL**: Transacciones más robustas (MVCC)
- **MariaDB**: Usa InnoDB (soporta transacciones ACID)

**Tu aplicación**: No usa transacciones explícitas, por lo que no hay impacto.

## 3. Case Sensitivity

- **PostgreSQL**: Case-sensitive por defecto en nombres de tablas/columnas
- **MariaDB**: Case-insensitive en nombres de tablas (depende del SO)

**Prisma maneja esto automáticamente** usando convenciones consistentes.

---

# ✅ Pruebas Post-Migración

Después de migrar, verifica:

### 1. Conexión a la Base de Datos

```
npx prisma db pull
```

Debería mostrar el schema sin errores.

### 2. Operaciones CRUD

```
# Abrir Prisma Studio
npx prisma studio
```

Prueba:
- ✅ Crear registros
- ✅ Leer registros
- ✅ Actualizar registros
- ✅ Eliminar registros

### 3. Funcionalidad de la App

- [ ] Login de administrador
- [ ] Crear cliente
- [ ] Generar link de encuesta
- [ ] Completar encuesta
- [ ] Ver estadísticas
- [ ] Exportar a Excel

---

## 📚 Referencias


- Prisma - MySQL (https://www.prisma.io/docs/concepts/database-connectors/mysql)
- MariaDB Documentation (https://mariadb.com/kb/en/documentation/)
- Next.js Deployment (https://nextjs.org/docs/deployment)


---

## 👍 Resumen

**✅ El código de la aplicación NO requiere cambios**
**✅ Solo cambia el schema de Prisma y las variables de entorno**
**✅ Rendimiento similar para este caso de uso**
**✅ Totalmente funcional con MariaDB/MySQL**