

# Database Connection Fix - PR #31

---

## Overview

---

This PR addresses critical database connection issues in the CitaPlanner application deployed on Easypanel with PostgreSQL.

## Issues Fixed

---

### 1. DATABASE\_URL Validation and Parsing

- **Problem:** The application was not properly validating the DATABASE\_URL format before attempting connections
- **Solution:** Added comprehensive URL validation that:
  - Checks for proper PostgreSQL URL format ( `postgresql://` or `postgres://` )
  - Extracts and validates all components (user, password, host, port, database)
  - Masks sensitive information in logs
  - Provides clear error messages for invalid URLs

### 2. Connection Verification

- **Problem:** No proper connection testing before running migrations
- **Solution:** Implemented dual-layer connection verification:
  - **Primary:** Direct PostgreSQL connection using `psql` client
  - **Secondary:** Prisma-based connection verification
- Retry logic with configurable attempts (30 for psql, 5 for Prisma)
- Clear diagnostic messages at each step

### 3. Error Handling and Logging

- **Problem:** Limited error information made debugging difficult
- **Solution:** Enhanced logging system with:
  - Timestamped log entries
  - Multiple log levels (info, success, warning, error, debug)
  - Detailed error messages with troubleshooting steps
  - Log files for migration and generation steps
  - Masked credentials in output

### 4. Migration Execution

- **Problem:** Migrations failing silently or with unclear errors
- **Solution:** Improved migration process:
  - Attempts `prisma migrate deploy` first (production best practice)
  - Falls back to `prisma db push` if migrations don't exist
  - Logs output to files for debugging ( `/tmp/migrate-deploy.log` , `/tmp/db-push.log` )
  - Clear success/failure indicators

## 5. Prisma Client Generation

- **Problem:** Client generation failures not properly detected
- **Solution:** Enhanced generation process:
  - Logs output to `/tmp/prisma-generate.log`
  - Verifies client exists in expected location
  - Provides clear error messages on failure

## 6. PostgreSQL Client Tools

- **Problem:** `psql` not available in Docker image for connection testing
- **Solution:** Added `postgresql-client` package to both base and runner stages

## 7. Script Robustness

- **Problem:** Script could continue with partial failures
- **Solution:**
  - Added `set -euo pipefail` for strict error handling
  - Proper exit codes on critical failures
  - Clear distinction between recoverable warnings and fatal errors

## Database URL Format Support

The application now fully supports the Easypanel PostgreSQL URL format:

```
postgres://postgres:674a351a07db86883d92@cloudmx_citaplanner-db:5432/citaplanner-db?
sslmode=disable
```

### Supported URL Formats:

- `postgresql://user:password@host:port/database`
- `postgres://user:password@host:port/database`
- With query parameters (e.g., `?sslmode=disable`, `?schema=public`)
- Special characters in passwords (properly URL-encoded)

## Changes Made

### Files Modified:

#### 1. `docker-entrypoint.sh`

- Added `validate_database_url()` function
- Added `check_database_connection_psql()` function
- Enhanced `check_database_connection()` with better error handling
- Improved `run_migrations()` with fallback strategy
- Enhanced `generate_prisma_client()` with verification
- Updated `is_database_empty()` with better SQL execution
- Updated `configure_master_password()` with better SQL execution
- Restructured `main()` function with proper error handling
- Added `log_debug()` function for detailed diagnostics
- Changed from `set -e` to `set -euo pipefail` for stricter error handling

## 2. Dockerfile

- Added `postgresql-client` to base stage
- Added `postgresql-client` to runner stage (ensures availability at runtime)






# Testing Recommendations

## 1. Verify DATABASE\_URL Configuration

```
# In Easypanel, ensure DATABASE_URL is set correctly:
postgres://postgres:674a351a07db86883d92@cloudmx_citaplanner-db:5432/citaplanner-db?
sslmode=disable
```

## 2. Check Logs After Deployment

Look for these key log messages:

-  "DATABASE\_URL validada correctamente"
-  "Conexión a PostgreSQL establecida correctamente"
-  "Migraciones aplicadas correctamente"
-  "Cliente Prisma generado correctamente"
-  "Configuración de Master Admin creada correctamente"

## 3. Verify Database Tables

After successful deployment, verify tables exist:

```
❏ dt -- List all tables
SELECT * FROM master_admin_config; -- Verify master admin config
SELECT COUNT(*) FROM users; -- Check if seed data exists
```

# Deployment Steps

1. **Merge this PR** to main branch
2. **Redeploy in Easypanel:**
  - The application will automatically rebuild with the new Dockerfile
  - The enhanced entrypoint script will run
3. **Monitor logs** during startup:
  - Watch for connection validation messages
  - Verify migrations complete successfully
  - Check for master admin configuration
4. **Verify functionality:**
  - Access the application at <https://citaplanner.com>
  - Test master admin panel at <https://citaplanner.com/admin/master>
  - Verify database operations work correctly

# Troubleshooting

## If Connection Fails:

1. **Check DATABASE\_URL format:**

```
bash
```

```
# Should match this pattern:
postgres://USER:PASSWORD@HOST:PORT/DATABASE
```

## 2. Verify PostgreSQL is running:

- Check Easypanel dashboard
- Ensure citaplanner-db service is active

## 3. Check network connectivity:

- Verify internal hostname: `cloudmx_citaplanner-db`
- Verify port: `5432`

## 4. Review logs:

- Look for masked DATABASE\_URL in logs
- Check specific error messages
- Review `/tmp/migrate-deploy.log` if migrations fail

## If Migrations Fail:

### 1. Check migration files exist:

```
bash
ls -la app/prisma/migrations/
```

### 2. Verify schema.prisma is correct:

```
bash
cat app/prisma/schema.prisma
```

### 3. Try manual migration:

```
bash
npx prisma migrate deploy
# or
npx prisma db push
```

## If Master Admin Config Fails:

### 1. Verify table exists:

```
sql
\d master_admin_config
```

### 2. Check for existing config:

```
sql
SELECT * FROM master_admin_config;
```

### 3. Manually insert if needed:

```
sql
INSERT INTO master_admin_config (id, password_hash, created_at, updated_at,
last_password_change)
VALUES ('singleton', '$2b$10$P/AV363LeWhZGK0kkrON3eGmAlkmiTHKuzZzDKCAppFV.0Gzf0Za0',
NOW(), NOW(), NOW())
ON CONFLICT (id) DO NOTHING;
```

## Security Considerations

1. **Password Masking:** All logs mask the database password

2. **Environment Variables:** DATABASE\_URL should be set as environment variable, never hard-coded
3. **Master Password:** Default master password should be changed after first login
4. **SSL Mode:** Consider enabling SSL for production ( `?sslmode=require` )

## Performance Improvements

---

1. **Faster Startup:** Connection verification happens early, preventing wasted initialization
2. **Better Error Recovery:** Clear failure points prevent partial initialization
3. **Efficient Retries:** Configurable retry logic with exponential backoff

## Future Enhancements

---

1. **Health Checks:** Add `/health` endpoint that verifies database connectivity
2. **Migration Status:** Add endpoint to check migration status
3. **Connection Pooling:** Optimize Prisma connection pool settings
4. **Monitoring:** Add metrics for connection attempts and failures

## Related Issues

---

- Fixes database connection failures on Easypanel deployment
- Resolves missing `master_admin_config` table issue
- Addresses migration execution problems
- Improves error diagnostics and logging

## Credits

---

- Database URL format support for Easypanel PostgreSQL
- Enhanced error handling and logging
- Comprehensive connection validation
- Robust migration execution strategy