✅ PostgreSQL 18 LÄUFT – UND DEIN USER AUCH!

Alles perfekt:

- Du bist als `postgres` reingekommen.
- `ALTER ROLE` hat geklappt.
- Login als `song_nexus_user` auf `song_nexus_dev` funktioniert:

```
song_nexus_dev=>
```

Genau das wollten wir erreichen.

Zum Beenden von psql:

```
\q
```

Verbindungsdaten für dein Projekt

Das sind jetzt deine **fixen Settings**, die wir im Backend verwenden:

- Host: `localhost`
- Port: `5432`
- Datenbank: `song_nexus_dev`
- User: `song_nexus_user`
- Passwort: `SongNexus2025`

Beispiel-Connection-String (für später im Code)

```
postgres://song_nexus_user:SongNexus2025@localhost:5432/song_nexus_dev
```

oder getrennt als Config-Objekt (z.B. für Node/pg):

```
{
  host: 'localhost',
  port: 5432,
  database: 'song_nexus_dev',
  user: 'song_nexus_user',
```

```
  password: 'SongNexus2025'
}
```

Wie geht’s jetzt im Projekt weiter?

Das hängt vom nächsten Schritt ab, den du angehen willst:

1. **Datenbank-Schema anlegen**
   Tabellen wie `users`, `tracks`, `orders`, `play_stats` etc. in `song_nexus_dev` erstellen.
2. **Backend mit PostgreSQL verbinden**
   In deinem Node/Express-Projekt (oder was du verwendest) die DB-Verbindung mit obigen Daten eintragen.
3. **Erste Test-Query aus dem Code**
   Z.B. `SELECT NOW();` oder `SELECT 1;` aus deinem Backend aufrufen, um die Verbindung zu prüfen.

Schreib bitte kurz:

- Mit welcher Sprache/Lib greifst du auf die DB zu? (z.B. Node.js mit `pg`, irgendwas anderes?)
- Hast du schon ein Projekt-Repo, in dem eine DB-Config-Datei existiert?

Dann gibt es als nächsten Schritt eine **konkrete Anleitung genau für dein Setup** (inkl. Code-Snippet), damit dein Backend heute noch mit PostgreSQL 18 spricht.