

Prisma to Drizzle ORM Migration

Overview

This document tracks the migration from Prisma ORM to Drizzle ORM for the Sports Bar TV Controller application.

Migration Status











✓ Completed (Phase 1)

Automated Conversion - 40 files converted:

- All simple Prisma queries (findUnique, findFirst, findMany, create, update, delete) converted to Drizzle
- Import statements updated to use Drizzle and drizzle-orm operators
- Schema imports added for tables used in each file

Files Successfully Converted:

1. `src/app/api/atlas/query-hardware/route.ts` - ✓ Fully converted to Drizzle
2. `src/app/api/ai/qa-entries/route.ts` - ✓ Basic queries converted
3. `src/app/api/api-keys/[id]/route.ts` - ✓ Basic queries converted
4. `src/app/api/api-keys/route.ts` - ✓ Basic queries converted
5. `src/app/api/audio-processor/[id]/ai-gain-control/route.ts` - ✓ Basic queries converted
6. `src/app/api/audio-processor/[id]/zones-status/route.ts` - ✓ Basic queries converted
7. `src/app/api/audio-processor/input-levels/route.ts` - ✓ Basic queries converted
8. `src/app/api/audio-processor/inputs/route.ts` - ✓ Basic queries converted
9. `src/app/api/audio-processor/matrix-routing/route.ts` - ✓ Basic queries converted
10. `src/app/api/audio-processor/meter-status/route.ts` - ✓ Basic queries converted
11. `src/app/api/audio-processor/outputs/route.ts` - ✓ Basic queries converted
12. `src/app/api/audio-processor/zones/route.ts` - ✓ Basic queries converted
13. `src/app/api/cec/discovery/route.ts` - ✓ Basic queries converted
14. `src/app/api/chat/route.ts` - ✓ Basic queries converted
15. `src/app/api/documents/[id]/route.ts` - ✓ Basic queries converted
16. `src/app/api/enhanced-chat/route.ts` - ✓ Basic queries converted
17. `src/app/api/globalcache/devices/[id]/route.ts` - ✓ Basic queries converted
18. `src/app/api/globalcache/devices/[id]/test/route.ts` - ✓ Basic queries converted
19. `src/app/api/globalcache/devices/route.ts` - ✓ Basic queries converted
20. `src/app/api/globalcache/learn/route.ts` - ✓ Basic queries converted
21. `src/app/api/globalcache/ports/[id]/route.ts` - ✓ Basic queries converted
22. `src/app/api/ir/commands/route.ts` - ✓ Basic queries converted
23. `src/app/api/ir/credentials/route.ts` - ✓ Basic queries converted
24. `src/app/api/ir/database/download/route.ts` - ✓ Basic queries converted
25. `src/app/api/ir/devices/[id]/route.ts` - ✓ Basic queries converted
26. `src/app/api/ir/devices/route.ts` - ✓ Basic queries converted
27. `src/app/api/keys/route.ts` - ✓ Basic queries converted
28. `src/app/api/matrix-config/route.ts` - ✓ Basic queries converted
29. `src/app/api/schedules/[id]/route.ts` - ✓ Basic queries converted
30. `src/app/api/schedules/execute/route.ts` - ✓ Basic queries converted

31. `src/app/api/schedules/logs/route.ts` -  Basic queries converted
32. `src/app/api/system/status/route.ts` -  Basic queries converted
33. `src/app/api/tests/wolfpack/connection/route.ts` -  Basic queries converted
34. `src/app/api/todos/[id]/complete/route.ts` -  Basic queries converted
35. `src/app/api/todos/[id]/documents/route.ts` -  Basic queries converted
36. `src/app/api/todos/[id]/route.ts` -  Basic queries converted
37. `src/app/api/todos/route.ts` -  Basic queries converted
38. `src/app/api/upload/route.ts` -  Basic queries converted
39. `src/lib/ai-gain-service.ts` -  Basic queries converted
40. `src/lib/gitSync.ts` -  Basic queries converted

Pending (Phase 2) - 44 files

Files with Complex Queries Needing Manual Review:

These files contain more complex Prisma queries that require manual conversion:

- Channel presets routes (4 files)
- CEC control routes (4 files)
- FireTV/FireCube routes (10+ files)
- Soundtrack routes (3 files)
- Sports guide routes (5+ files)
- Wolfpack matrix routes (5+ files)
- And others (see full list below)

Migration Approach

Phase 1: Automated Conversion (COMPLETED)

Used `convert-to-drizzle.py` script to handle:

- Simple `findUnique()` with single where clause
- Simple `findFirst()` with optional where clause
- Simple `findMany()` without complex filters
- Simple `create()` with data object
- Simple `update()` with where and data
- Simple `delete()` with where clause
- Import statement conversions

Phase 2: Manual Conversion (IN PROGRESS)

Requires manual attention for:

- Complex where clauses with multiple conditions
- Relations and joins
- Transactions
- `orderBy` with multiple fields
- `include/select` statements
- Aggregations (count, sum, avg, etc.)
- `GroupBy` operations
- `createMany` with `skipDuplicates`
- `updateMany` with complex conditions

Phase 3: Testing and Verification (PENDING)

- Build the application to identify TypeScript errors

- Fix any runtime errors
- Test all database operations
- Verify groups functionality
- Test audio processing features
- Test other critical features

Phase 4: Cleanup (PENDING)

- Remove Prisma adapter files:
 - `src/lib/prisma.ts`
 - `src/db/prisma-adapter.ts`
 - `src/lib/db.ts` (update to remove prisma export)
- Remove Prisma migration files (`prisma/migrations/`)
- Remove Prisma schema files (`prisma/schema.prisma*`)
- Update documentation

Drizzle ORM Patterns

Basic Query Patterns

```
// OLD: Prisma
import prisma from '@lib/prisma'
const user = await prisma.user.findUnique({ where: { id: userId } })

// NEW: Drizzle
import { db } from '@db'
import { users } from '@db/schema'
import { eq } from 'drizzle-orm'
const user = await db.select().from(users).where(eq(users.id, userId)).limit(1).get()
```

Using Helper Functions

```
// Import helpers
import { findUnique, findMany, create, update, deleteRecord } from '@lib/db-helpers'
import { eq } from 'drizzle-orm'
import { schema } from '@db'

// Find one
const user = await findUnique('users', eq(schema.users.id, userId))

// Find many
const users = await findMany('users', {
  where: eq(schema.users.active, true),
  orderBy: desc(schema.users.createdAt),
  limit: 10
})

// Create
const newUser = await create('users', {
  name: 'John Doe',
  email: 'john@example.com'
})

// Update
const updatedUser = await update('users',
  eq(schema.users.id, userId),
  { name: 'Jane Doe' }
)

// Delete
await deleteRecord('users', eq(schema.users.id, userId))
```

Complex Where Clauses

```
import { and, or, eq, gt, like } from 'drizzle-orm'

// Multiple conditions
const results = await db.select()
  .from(users)
  .where(
    and(
      eq(users.active, true),
      gt(users.age, 18),
      like(users.email, '%example.com')
    )
  )
  .all()

// OR conditions
const results = await db.select()
  .from(users)
  .where(
    or(
      eq(users.role, 'admin'),
      eq(users.role, 'moderator')
    )
  )
  .all()
```

Remaining Files to Convert

High Priority (User-Facing Features)

1. Channel preset management
2. CEC power control
3. FireTV device management
4. Soundtrack control
5. Sports guide

Medium Priority (Background Services)

1. Matrix routing
2. Wolfpack control
3. Schedule execution

Low Priority (Admin/Debug)

1. Diagnostic tools
 2. Analytics
 3. Test routes
-