5. Select a database migration tool in your own project

Database ORM and Migration Tool Selection

ORM and Migration Tools Comparison

Tool	Database Support	Migration System	TypeScript Support	Learning Curve	Integration with Next.js
Prisma ORM	PostgreSQL, MySQL, SQLite, MongoDB	Built-in migration system	Excellent (type generation)	Low	Excellent
TypeORM	PostgreSQL, MySQL, SQLite, MSSQL	Built-in migrations	Good	Medium	Good
Sequelize	PostgreSQL, MySQL, SQLite, MSSQL	Built-in migrations	Moderate (requires setup)	Medium	Moderate
Raw SQL + migration tools	Any	Manual/custom tools	None	High	Manual integration

Why Prisma ORM for PostgreSQL

Type Safety: Prisma automatically generates TypeScript types from your database schema, ensuring type safety across the entire application stack from database to frontend.

Migration System: Built-in migration system handles database schema changes automatically with simple commands, eliminating manual SQL migration scripts.

Next.js Integration: Designed to work seamlessly with Next.js applications, providing optimal developer experience and performance.

PostgreSQL Optimization: Excellent support for PostgreSQL features including JSON fields, relations, and advanced querying needed for game data storage.

Schema Definition Example

Prisma uses a schema file (prisma/schema.prisma) to define database structure:

```
// Updated User model for NextAuth compatibility
model User {
  id             String    @id @default(cuid())
  username       String    @unique
  hashedPassword String?   // Make optional for OAuth users
  name           String?   // For NextAuth
  email          String?   @unique // For NextAuth
  image          String?   // For NextAuth
  gamesPlayed     Int      @default(0)
  gamesWon        Int      @default(0)
  totalPoints     Int      @default(0)
  createdAt      DateTime  @default(now())
  updatedAt      DateTime  @updatedAt

  // Your existing relations
  roomsOwned Room[] @relation("RoomOwner")
}
```

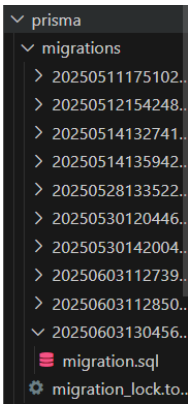
Migration Commands

Initialize Prisma: npx prisma init

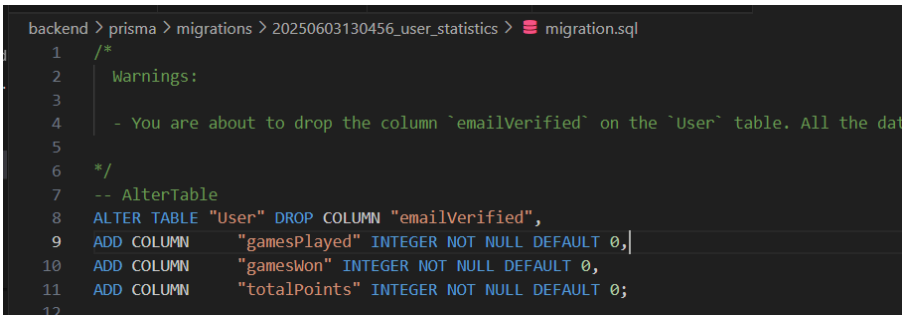
Generate Migration: npx prisma migrate dev --name

Apply Migration to Production: npx prisma migrate deploy

Generate TypeScript Client: npx prisma generate



Example of the migration file:



Migration Workflow

Development Process:

1. Modify *schema.prisma* file with new fields or tables
2. Run *npx prisma migrate dev --name* to generate migration
3. Prisma creates SQL migration files automatically
4. TypeScript types are regenerated automatically
5. Use new types immediately in Next.js application

Production Deployment:

1. Migration files are committed to version control
2. Run *npx prisma migrate deploy* on production server
3. Database schema is updated automatically
4. No manual SQL scripts required

This approach ensures consistent database schemas across all environments while maintaining type safety throughout the application development process.