ramosmarco95: How is it using database in this app?

GitHub Copilot: I'll help you understand how the database is being used in this real-time Kanban application. Let me examine the database-related files to give you a comprehensive overview.

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/prisma/schema.prisma)

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/src/lib/prisma.ts)

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/src/scripts/seed.ts#1-1), lines 1 to 100

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/src/scripts/seed.ts#100-100), lines 100 to 164

Searched text for `prisma` (`**/apps/server/src/routes/*.ts`), 20 results

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/package.json)

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/prisma/migrations)

Read [](file:///c%3A/Users/ramos/javascript-directories/real-time-kanban/apps/server/prisma/migrations/20251016030148_example1/migration.sql#1-1), lines 1 to 50

Now let me check how the database is configured and connected:

Searched text for `DATABASE_URL` (`**/apps/server/**`), 3 results

Database Usage Overview

This real-time Kanban application uses **Prisma ORM** with **SQLite** as the database. Here's how the database is structured and used:

🗄️ **Database Technology Stack**

- **Database**: SQLite (lightweight, file-based database)

- **ORM**: Prisma (TypeScript-first database toolkit)

- **Database File**: `dev.db` (stored locally in the server directory)

📊 **Database Schema**

The app has 4 main entities with clear relationships:

1. **Users** (`users` table)

   - Stores user information (name, email, avatar)

   - Unique email constraint

2. **Boards** (`boards` table)

   - Kanban boards with title and description

   - One board can have multiple columns

3. **Columns** (`columns` table)

  - Represents Kanban columns (To Do, In Progress, Review, Done)

  - Belongs to a board, has ordering system

  - Cascade delete when board is deleted

4. **Cards** (`cards` table)

  - Individual tasks/cards in the Kanban board

  - Belongs to a column, can be assigned to users

  - Has labels (stored as JSON string), due dates, descriptions

  - Cascade delete when column is deleted

🔗 **Key Relationships**

- **Board → Columns**: One-to-many (boards can have multiple columns)

- **Column → Cards**: One-to-many (columns can have multiple cards)

- **User → Cards**: One-to-many (users can be assigned to multiple cards)

🛠️ **Database Operations**

The app uses Prisma Client for all database operations:

- **Connection**: Managed through prisma.ts with singleton pattern

- **CRUD Operations**: Implemented in route handlers (boards.ts, columns.ts, cards.ts)

- **Migrations**: Managed by Prisma (`prisma/migrations/`)

- **Seeding**: Sample data created via seed.ts

📝 **Database Scripts**

Available npm scripts for database management:

- `pnpm db:migrate` - Run database migrations

- `pnpm db:generate` - Generate Prisma client

- `pnpm db:seed` - Seed database with sample data

- `pnpm db:reset` - Reset database and run migrations

💡 **Notable Features**

- **Ordering System**: Uses float values for flexible card/column ordering

- **Soft References**: Cards can optionally be assigned to users

- **JSON Storage**: Labels are stored as JSON strings in SQLite

- **Cascade Deletes**: Proper cleanup when boards/columns are deleted

- **Development-friendly**: Hot reload support with singleton pattern to prevent connection issues

The database design is well-structured for a Kanban application, supporting real-time updates through the API while maintaining data integrity with proper foreign key constraints.