Data Models & Database

Relevant source files

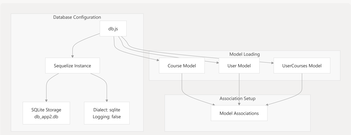
This document covers the data models and database configuration for Project 2: Course Platform (Proy2_Cursos). It details the Sequelize ORM setup, model definitions, and database relationships that support the course management and user subscription functionality.

For authentication and middleware implementation details, see Authentication & Middleware. For testing and development setup, see Development & Testing.

Database Configuration

The course platform uses SQLite as its database engine with Sequelize ORM for data modeling and relationship management. The database configuration is centralized in the `db.js` file which establishes the connection, loads models, and defines associations.

Sequelize Setup



The Sequelize instance is configured with SQLite dialect and points to a database file located at `../db/db_app2.db`. Logging is disabled by default but can be enabled for debugging SQL queries.

Sources: Proy2_Cursos/config/db.js 5-9

Data Models

The platform defines three core models that represent the main entities in the course management system.

User Model

The `User` model represents platform users with authentication credentials and administrative privileges.

Field	Type	Constraints	Description
`id`	INTEGER	PRIMARY KEY, AUTO INCREMENT	Auto-generated unique identifier
`name`	STRING	NOT NULL	User's display name
`user`	STRING	NOT NULL, UNIQUE	Username for authentication
`password`	STRING	NOT NULL	User's password (stored as hash)
`isAdmin`	BOOLEAN	NOT NULL, DEFAULT false	Administrative privileges flag

Sources: Proy2_Cursos/models/user.js 4-25

Course Model

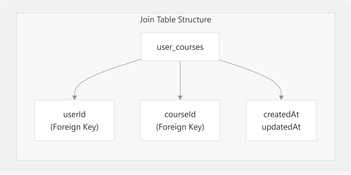
The `Course` model represents educational courses that can be created and managed by users.

Field	Type	Constraints	Description
`id`	INTEGER	PRIMARY KEY, AUTO INCREMENT	Auto-generated unique identifier
`title`	STRING	NOT NULL	Course title
`description`	TEXT	NOT NULL	Detailed course description
`category`	STRING	NOT NULL	Course category classification
`is_public`	BOOLEAN	NOT NULL, DEFAULT true	Public visibility flag
`image`	BLOB	NULL	Course image data
`userId`	INTEGER	FOREIGN KEY	Reference to course creator

Sources: Proy2_Cursos/models/course.js 6-36

UserCourses Model

The `user_courses` model serves as a join table implementing the many-to-many relationship between users and courses for subscription tracking.

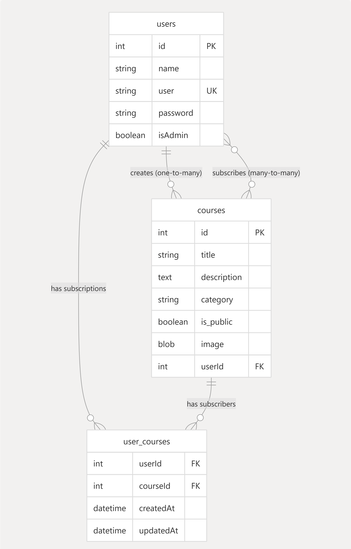


Sources: Proy2_Cursos/config/db.js 22

Model Relationships

The course platform implements a complex relationship structure supporting both course creation and user subscriptions.

Relationship Diagram



Association Implementation

The relationships are implemented through Sequelize associations in the database configuration:

Many-to-Many Subscriptions:

- Users can subscribe to multiple courses
- Courses can have multiple subscribed users
- Implemented via `belongsToMany` through `user_courses` join table

One-to-Many Course Creation:

- Users can create multiple courses
- Each course belongs to one creator user
- Implemented via `belongsTo` and `hasMany` associations

Direct Join Table Access:

- Explicit associations to `user_courses` enable direct queries on subscription data
- Both `users` and `courses` have `hasMany` relationships to `user_courses`

Sources: Proy2_Cursos/config/db.js 29-44

Database Module Export

The database configuration exports a comprehensive `db` object containing:

Export	Description
`db.Sequelize`	Sequelize constructor class
`db.sequelize`	Configured Sequelize instance
`db.users`	User model with associations
`db.courses`	Course model with associations
`db.user_courses`	UserCourses join table model

This structure enables other parts of the application to access both the database connection and all configured models with their relationships intact.

Sources: Proy2_Cursos/config/db.js 12-47