RESTful Library Management System

Andrei Petru Stefanie
Group 244

Requirements

Build a library management system enabling its users to perform the following:

- Read, create, update, delete authors
- Search authors by substrings from their name
- Read, create, update, delete books
- Search books by substrings from their title

The purpose of the project is to present the REST architectural style.

Implementation

The system consists of a server and several clients interacting through REST API calls over HTTP.

The clients are based on the command-line interface (CLI) and enable the users to perform the described actions. They follow the <program> <verb> <subject> format - for example `libctl list authors.`

Server

The server is implemented in Go (1.18) and exposes its functionality as a RESTful API.

The system has two resources: **Authors** and **Books**.

An author has the following properties:

- Id (int)
- Name (string)

A book has the following properties:

- Id (int)
- Title (string)
- Publishing year (int)
- Author

The server exposes the following endpoints:

- `GET /v1/authors`
  - Retrieve the list of authors, optionally filtered by an arbitrary substring from their name
- `GET /v1/authors/<id>`
  - Retrieve a certain author
- `POST /v1/authors`
  - Create an author by providing the name

- `DELETE /v1/authors/<id>`
    - Delete the author identified by the *id*
- `PUT /v1/authors/<id>`
    - Update the name of the author identified by the *id*
- `GET /v1/books`
    - Retrieve the list of books and their authors, optionally filtered by an arbitrary substring from their title
- `GET /v1/books/<id>`
    - Retrieve a certain book
- `POST /v1/books`
    - Create a book by providing the title, author id, and publishing year
- `DELETE /v1/books/<id>`
    - Delete the book identified by the *id*
- `PUT /v1/books/<id>`
    - Update the title of the book identified by the *id*

The API is described in the OpenAPI 3.0 format in the file **server-go/spec.yaml.** You can view the specification using tools such as Postman or Swagger Editor.

The server is based on the gin framework which provides features such as routing to define how the server responds to each endpoint and middleware for handling CORS, authentication, logging, etc.

The server respects most REST conventions such as using the expected HTTP status codes (see the OpenAPI Spec), responding with the expected representation for various verbs, and considering the Content-Type header.

Database

The database is MySQL (developed using version 8.0.29) and the server interacts with it through the gorm ORM.

To run the server you have to set the `DB_CONNECTION_STRING` environment variable that instructs the server on how to connect to the database.
E.g. `export DB_CONNECTION_STRING="root:pass@tcp(127.0.0.1:3306)/library?charset=utf8mb4&parseTime=True&loc=Local"`.

The ORM automatically creates the database tables when it starts. These are illustrated in diagram 1.

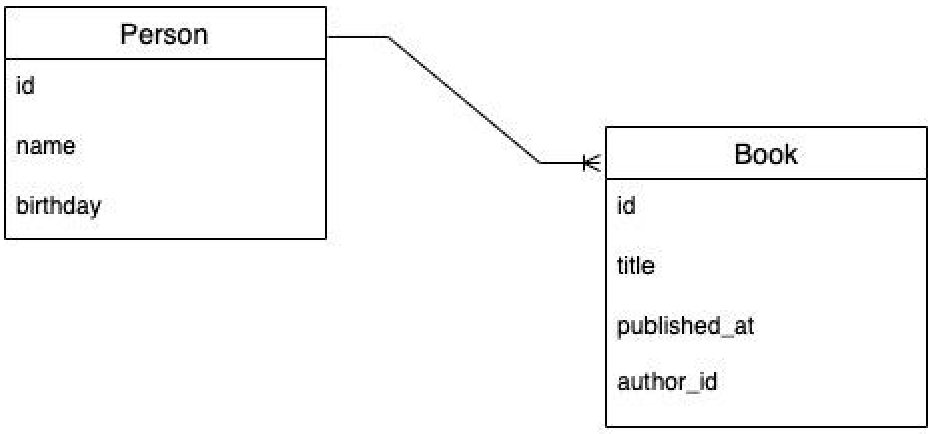

Diagram 1. Database Model

Serialization

Gin also handles serialization/deserialization and leverages the JSON/XML/YAML struct tags.

Running the Server

After filling in the **.env** file, you can run the server with `go run .` or using the provided script, `run.sh` (it uses nodemon to watch for code changes - npm install -g nodemon)

Clients

There are 3 available CLI clients. The positional arguments and options are the same for all of them:

- List authors: `list authors [--query will]`
- Get author: `get authors 1`
- Add author: `add authors --name 'William Shakespeare``
- Delete author: `delete authors 1`
- Update author: `update authors 1 --name 'New Name``
- List books: `list books [--query 'ham']`
- Get book: `get books 1`
- Add book: `add books --title 'Hamlet` --year 1599 --author 1`
- Delete book: `delete books 1`
- Update book: `update books id 1 --title 'New Title``

Go

The Go client is based on the net/http standard library and uses the cobra library to create the CLI.

You can run it with `go run . list books`

C#

The C# client is built with .NET Core 6 and uses the HttpClient standard class for interacting with the server and the CommandLine library for the CLI.

You can run it with `dotnet run list books`

Java

The Java client is built with Java 18 and Gradle. It uses the Jersey library (3.1.0-M3) for interacting with the server and the picocli library for the CLI.

You can run it with `./gradlew run --args=”list books”`

References

- https://github.com/gin-gonic/gin
- https://gorm.io/
- https://github.com/spf13/cobra
- https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/console-webapiclient
- https://github.com/commandlineparser/commandline
- https://picocli.info/
- https://www.baeldung.com/jersey-jax-rs-client