# Article API

# Solution

The solution is based on <https://ffxblue.github.io/interview-tests/test/article-api/>

* Language tool: Golang v1.11.1
* Development Environment: Windows, VSCode
* Using 3rd-party Package:

1. github.com/ant0ine/go-json-rest

Based on http package but it has more convenient approaches to write restful route handlers, requests and responses.

1. github.com/mattn/go-sqlite3

As spec described, we need to store and query data.

Using a in-memory structure can do this work, but the data is not persistent. When server is down, all stored data will lose.

Or we can choose a SQL or NOSQL database. Some databases like mongodb and mysql require much disk space and need to be run in background. SQLite is light-weight and based on file, not network service, so I choose it for querying.

1. github.com/jmoiron/sqlx

As database/sql do not support complex type like structs to fetch queried data, using sqlx is a good alternative to write raw query.

# Source Structure

|  |  |
| --- | --- |
|  |  |
| handler | Handling client request by invoking services.  Marshaling package and transforming to services’ input format |
| services | Basic functions include database |
| server | Initializing and mapping route to handler |
| vendor | 3rd party packages |
| main.go | Application entrance to load server |

# Logic

AppServer

Database

API Handler

REQ

Response

# Database

As SQL doesn’t support structure nesting, the article part and tag part need to be separated to different tables (articles and articleTag).

services/database/models.go support struct definition

and services/database/database.go support tables creating.

Date field do not use time.Tm struct because we need Date without Time, using a string type would be more easy to compare.

Creating an article needs to Insert articles table first, get the id and then insert to articleTag table.

Get an article also need 2 queries and joint the results together.

Query by tag and date need a combined query to find article IDs first, and then use the IDs to get distinct tags, finally compose them into a response structure.

All the transaction need to be protected and synchronized, so I use a locker in handler.

# Future

These article APIs are not complete. We also need APIs to update and delete article and tags.

If the articles have authors, we also need tables to reserve the relations between article\_id and author\_id.

And if allow user comments, there will be relations between comments table and articles table.