Task3-TEFA

CRUD 2 features with Golang

Nama anggota kelompok:

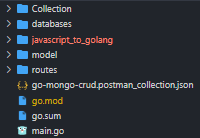
Helsa Nesta Dhaifullah / 5025201005

Naily Khairya

We chose Helsa’s previous task, it is e-learning app, and we create CRUD course and CRUD material with Golang language, Gin framework, and MongoDB as the database. We also try to implement hash table in our crud source code.

You can see the complete source code in (link github)

* Structure project file

Collection is to handle the collection-fetching functionality.

Databases is config for connect with our database in MongoDB.

Model is for the data struct of course and material for database.

Routes is for the controller of CRUD.

Main.go is the main program.

* Connect Go to MongoDB

Text

Description automatically generated

The ConnectDB function establishes a connection and returns a new MongoDB Client object. But first, you need setup the mongoDB, and copy-paste the MongoDB URl to connect Golang with the database.

* Create Database collection

Text

Description automatically generated

This function gets the Collection from the MongoDB database. The database name, in this case, is myGoappDB, with Posts as its collection.

* Database model

Text

Description automatically generated

We have 5 components in Course, such as ID, CourseName (string), Description (string), CreatedAt and UpdatedAt.

* Create POST Endpoint in Go

Text

Description automatically generated

The function of the CreateCourse method is to create a new data course in the MongoDB database using the Gin framework. The course data to be created will be taken from the JSON request sent by the client. Then, the data will be stored in a folder and will be included in a collection in the MongoDB database.

By using a map, the data can be easily accessed and manipulated by key rather than having to access each field in the coursePayload struct. It also allows for flexibility in the future if additional fields need to be added or removed.

* Create GET ALL courses Endpoint in Go

Text

Description automatically generated

This function of the GetAllCourses method is to handle a GET request to retrieve all courses. It then iterates through the cursor and stores the course data in a map using the make function to create a new map, and then appends the data to a slice of maps called courses. The slice of maps can be returned as a JSON response to the client.

* Create GET ONE courses Endpoint in Go

Text

Description automatically generated

This code defines a function that retrieves a single course from a MongoDB database based on its ID. The courseID parameter is retrieved from the URL path using the Param() method of the gin.Context object. A variable of type model.Course is created to store the result of the database query, and the ID string is converted to a MongoDB ObjectID. The FindOne() method is then called on the courseCollection object to retrieve the course with the specified ID, and the result is stored in the previously created variable.

* Create PUT update courses Endpoint in Go

Text

Description automatically generated

The code is an endpoint function to update a course in a MongoDB collection. The code then gets the courseID parameter from the request and decodes it into an ObjectId. The edited map is created with the fields that can be edited in the course document, with the updatedAt field being set to the current time. The courseCollection.UpdateOne method is then called, passing in the filter and update documents created from the ObjectId and edited map. The result of the update is returned and stored in the res map.

* Create DELETE courses Endpoint in Go

Text

Description automatically generated

The code defines a function named DeleteCourse that receives a courseID as a parameter from the URL path and converts it into an object ID. The function then deletes the document with the given ID from the MongoDB collection.

* Routes in main.go
* Benchmark test
* **Result of the test**

results from tests with 10, 100, and 1000 usecase data between crud without applying a hash map and with a hash table / map.

|  |  |  |
| --- | --- | --- |
| N | No Hash Table / map | With Hash Table / map |
| 10 | 2.913 second | 2.88 second |
| 100 | A screenshot of a computer  Description automatically generated  4.57 second | A screenshot of a computer  Description automatically generated with medium confidence  3.03 second |
| 1000 | Graphical user interface, application  Description automatically generated  6.32 second | A screenshot of a computer  Description automatically generated with medium confidence  3.06 second |