Go Fiber dan MongoDB

Tujuan Pembelajaran

- Memahami cara membuat REST API dengan Go Fiber dan MongoDB
- Mengimplementasikan Clean Architecture dalam Go
- Memisahkan business logic, data access, dan presentation layer
- Menggunakan Repository Pattern untuk akses database
- Menerapkan Dependency Injection
- Membuat aplikasi yang scalable dan maintainable

Install Dependencies

```
# MongoDB driver official
go get go.mongodb.org/mongo-driver/mongo
go get go.mongodb.org/mongo-driver/bson
go get go.mongodb.org/mongo-driver/bson/primitive
```

Environment Configuration

```
MONGODB_URI=mongodb://localhost:27017
DATABASE_NAME=namadatabase
COLLECTION_NAME=namacolection
PORT=3000
```

Model layer

Berisi struktur data (struct) yang memetakan data antara aplikasi Go dan dokumen MongoDB.

```
package model

import "go.mongodb.org/mongo-driver/bson/primitive"

// User merepresentasikan dokumen pengguna di MongoDB
type User struct {
    // ID menggunakan primitive.ObjectID dari MongoDB dan diabaikan
('omitempty')
    // jika kosong saat marshal (misalnya, saat membuat pengguna baru).
    ID    primitive.ObjectID `bson:"_id,omitempty" json:"id,omitempty"`
    Name  string             `bson:"name" json:"name"`
    Email string             `bson:"email" json:"email"`
```

```
    Age   int                `bson:"age" json:"age"`
}
```

Layer Repository

Berisi antarmuka dan implementasi untuk operasi CRUD (Create, Read, Update, Delete) langsung ke MongoDB.

```
package repository

import (
    "context"
    "go-fiber-mongo-architecture/app/model"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/bson/primitive"
    "go.mongodb.org/mongo-driver/mongo"
)

// IUserRepository mendefinisikan operasi basis data (query) untuk entitas
User.
// Ini adalah kontrak yang diikuti oleh implementasi MongoDB.
type IUserRepository interface {
    CreateUser(ctx context.Context, user *model.User) (*model.User, error)
    FindUserByID(ctx context.Context, id string) (*model.User, error)
    FindAllUsers(ctx context.Context) ([]model.User, error)
}

// UserRepository implementasi IUserRepository menggunakan MongoDB.
type UserRepository struct {
    collection *mongo.Collection
}

// NewUserRepository membuat instance baru dari UserRepository dan
mengaitkannya dengan koleksi MongoDB.
func NewUserRepository(db *mongo.Database) IUserRepository {
    return &UserRepository{
        collection: db.Collection("users"), // Ganti 'users' dengan nama
koleksi Anda
    }
}

// CreateUser menyimpan pengguna baru ke MongoDB.
func (r *UserRepository) CreateUser(ctx context.Context, user *model.User)
(*model.User, error) {
    // Pastikan ID tidak disetel saat InsertOne
    user.ID = primitive.NilObjectID
```

```
    result, err := r.collection.InsertOne(ctx, user)
    if err != nil {
        return nil, err
    }
    // Dapatkan ID yang baru dibuat dari hasil insert
    user.ID = result.InsertedID.(primitive.ObjectID)
    return user, nil
}

// FindUserByID mengambil pengguna berdasarkan ID string.
func (r *UserRepository) FindUserByID(ctx context.Context, id string)
(*model.User, error) {
    // Konversi ID string menjadi primitive.ObjectID
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return nil, err // ID tidak valid
    }

    var user model.User
    filter := bson.M{"_id": objID}

    err = r.collection.FindOne(ctx, filter).Decode(&user)
    if err != nil {
        if err == mongo.ErrNoDocuments {
            return nil, nil // Dokumen tidak ditemukan, bukan error fatal
        }
        return nil, err
    }
    return &user, nil
}

// FindAllUsers mengambil semua pengguna dari koleksi.
func (r *UserRepository) FindAllUsers(ctx context.Context) ([]model.User,
error) {
    cursor, err := r.collection.Find(ctx, bson.M{})
    if err != nil {
        return nil, err
    }
    defer cursor.Close(ctx) // Pastikan kursor ditutup

    var users []model.User
    if err = cursor.All(ctx, &users); err != nil {
        return nil, err
    }
    return users, nil
}
```

Layer Service

Berisi logika bisnis, validasi, dan orkestrasi data sebelum memanggil repository

```
package service

import (
    "context"
    "errors"
    "go-fiber-mongo-architecture/app/model"
    "go-fiber-mongo-architecture/app/repository"
)

// IUserService mendefinisikan operasi logika bisnis untuk entitas User.
type IUserService interface {
    CreateUser(ctx context.Context, user *model.User) (*model.User, error)
    GetUserByID(ctx context.Context, id string) (*model.User, error)
    GetAllUsers(ctx context.Context) ([]model.User, error)
}

// UserService implementasi IUserService.
type UserService struct {
    repo repository.IUserRepository // Ketergantungan pada Repository
}

// NewUserService membuat instance baru dari UserService.
func NewUserService(repo repository.IUserRepository) IUserService {
    return &UserService{repo: repo}
}

// CreateUser memvalidasi data dan meneruskannya ke repository.
func (s *UserService) CreateUser(ctx context.Context, user *model.User)
(*model.User, error) {
    // Contoh Logika Bisnis / Validasi Data
    if user.Name == "" {
        return nil, errors.New("nama tidak boleh kosong")
    }
    if user.Email == "" || user.Age <= 0 {
        return nil, errors.New("email dan usia (harus > 0) harus diisi dengan
benar")
    }
    // Jika semua validasi lolos, panggil repository
    return s.repo.CreateUser(ctx, user)
}

// GetUserByID mengambil pengguna dan menangani kasus jika tidak ditemukan.
```

```go
func (s *UserService) GetUserByID(ctx context.Context, id string)
(*model.User, error) {
    // Logika Bisnis: Cek apakah ID valid sebelum ke DB (sudah dilakukan di
repo, tapi bisa ditambahkan di sini juga)
    user, err := s.repo.FindUserByID(ctx, id)
    if err != nil {
        return nil, err
    }
    if user == nil {
        return nil, errors.New("pengguna dengan ID tersebut tidak ditemukan")
    }
    return user, nil
}

// GetAllUsers mengambil semua pengguna.
func (s *UserService) GetAllUsers(ctx context.Context) ([]model.User, error) {
    // Logika Bisnis: Bisa menambahkan pagination atau filter di sini
    users, err := s.repo.FindAllUsers(ctx)
    if err != nil {
        return nil, err
    }
    return users, nil
}
```

Main layer

```go
package main

import (
    "context"
    "fmt"
    "log"
    "os"
    "time"

    // Impor dari modul lokal yang telah dibuat
    "go-fiber-mongo-architecture/app/model"
    "go-fiber-mongo-architecture/app/repository"
    "go-fiber-mongo-architecture/app/service"

    "github.com/gofiber/fiber/v2"
    "github.com/gofiber/fiber/v2/middleware/logger"
    "go.mongodb.org/mongo-driver/mongo"
    "go.mongodb.org/mongo-driver/mongo/options"
)

// connectDB menangani koneksi ke MongoDB.
```

```go
func connectDB() *mongo.Database {
    // Mengambil URI dari variabel lingkungan, atau menggunakan default lokal
    mongoURI := os.Getenv("MONGO_URI")
    if mongoURI == "" {
        mongoURI = "mongodb://localhost:27017" // Pastikan MongoDB Anda
berjalan di sini
        log.Println("Peringatan: MONGO_URI tidak disetel. Menggunakan
default:", mongoURI)
    }

    clientOptions := options.Client().ApplyURI(mongoURI)
    // Membuat konteks dengan batas waktu
    ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
    defer cancel()

    client, err := mongo.Connect(ctx, clientOptions)
    if err != nil {
        log.Fatalf("Koneksi ke MongoDB gagal: %v", err)
    }

    // Cek koneksi (Ping)
    err = client.Ping(ctx, nil)
    if err != nil {
        log.Fatalf("Ping ke MongoDB gagal: %v", err)
    }

    fmt.Println("🎉 Berhasil terhubung ke MongoDB!")
    return client.Database("mydatabase") // Ganti 'mydatabase' dengan nama DB
Anda
}

func main() {
    // 1. Inisialisasi Database (Hanya sekali)
    db := connectDB()

    // 2. Inisialisasi Arsitektur (Dependency Injection)
    // Repository menerima objek DB
    userRepo := repository.NewUserRepository(db)
    // Service menerima Repository
    userService := service.NewUserService(userRepo)

    // 3. Setup Fiber App
    app := fiber.New()
    app.Use(logger.New()) // Middleware logging untuk melihat permintaan

    // 4. Setup Routes (Handlers)
    api := app.Group("/api/v1")
    userRoutes := api.Group("/users")
```

```
    // POST /api/v1/users -> Membuat pengguna baru
    userRoutes.Post("/", func(c *fiber.Ctx) error {
        user := new(model.User)
        if err := c.BodyParser(user); err != nil {
            // Permintaan buruk (misalnya, JSON tidak valid)
            return c.Status(fiber.StatusBadRequest).JSON(fiber.Map{
                "error": "Permintaan tidak valid",
                "message": "Pastikan body permintaan Anda dalam format JSON
yang benar.",
            })
        }

        ctx, cancel := context.WithTimeout(context.Background(),
5*time.Second)
        defer cancel()

        createdUser, err := userService.CreateUser(ctx, user)
        if err != nil {
            // Menangani error dari layer Service/Repository (misalnya,
validasi gagal, DB error)
            return c.Status(fiber.StatusUnprocessableEntity).JSON(fiber.Map{
                "error":   "Gagal membuat pengguna",
                "message": err.Error(),
            })
        }

        return c.Status(fiber.StatusCreated).JSON(createdUser)
    })

    // GET /api/v1/users -> Mendapatkan semua pengguna
    userRoutes.Get("/", func(c *fiber.Ctx) error {
        ctx, cancel := context.WithTimeout(context.Background(),
5*time.Second)
        defer cancel()

        users, err := userService.GetAllUsers(ctx)
        if err != nil {
            return c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
                "error":   "Gagal mengambil data",
                "message": err.Error(),
            })
        }
        return c.JSON(users)
    })

    // GET /api/v1/users/:id -> Mendapatkan pengguna berdasarkan ID
    userRoutes.Get("/:id", func(c *fiber.Ctx) error {
```

```
        id := c.Params("id")

        ctx, cancel := context.WithTimeout(context.Background(),
5*time.Second)
        defer cancel()

        user, err := userService.GetUserByID(ctx, id)
        if err != nil {
            // Error dari service menunjukkan pengguna tidak ditemukan atau ID
tidak valid
            return c.Status(fiber.StatusNotFound).JSON(fiber.Map{
                "error":   "Gagal mengambil pengguna",
                "message": err.Error(),
            })
        }

        return c.JSON(user)
    })

    // 5. Jalankan Server
    log.Fatal(app.Listen(":3000"))
}
```

Migrasikan CRUD Pekerjaan Alumni dengan MongoDB (Modul 5)

- GET / project_name/pekerjaan - Ambil semua data pekerjaan alumni (Admin dan User)
- GET / project_name/pekerjaan/:id - Ambil data pekerjaan berdasarkan ID (Admin dan User)
- GET / project_name/pekerjaan/alumni/:alumni_id - Ambil semua pekerjaan berdasarkan alumni (Admin)
- POST / project_name/pekerjaan - Tambah pekerjaan baru (Admin)
- PUT / project_name/pekerjaan/:id - Update data pekerjaan (Admin)
- DELETE / project_name/pekerjaan/:id - Hapus data pekerjaan (Admin)