**Laporan Hasil Project Backend Praktikum**



Nama : Muhamad Ghandi Nur Setiawan

Nim : 434221014

Kelas : C-1

**Universitas Airlangga**

**Surabaya**

**2024**

**Hasil Project Backend Praktikum**

Tugas :

1. Silahkan anda buka soal UTS dan hasil disain MongoDb anda. Perhatikan hal berikut:

* Tabel Role hanya memiliki 2 nilai, yaitu admin dan civitas.

*// colection role*

use**(**'unairsatu\_v2'**)**;

db.getCollection**(**'role'**)**.updateMany**(**

    {},

    {

**$set:** {

**created\_by:** ObjectId**(**'674039236461fc1488d67fec'**)**,

**updated\_by:** ObjectId**(**'674039236461fc1488d67fec'**)**

        }

    }

**)**;

use**(**'unairsatu\_v2'**)**;

db.getCollection**(**'role'**)**.insertMany**([**

    {

**"nm\_role":** "Admin",

**created\_at:** new Date**()**,

**created\_by:** 1,

**updated\_at:** new Date**()**,

**updated\_by:** 1

    },

    {

**"nm\_role":** "civitas",

**created\_at:** new Date**()**,

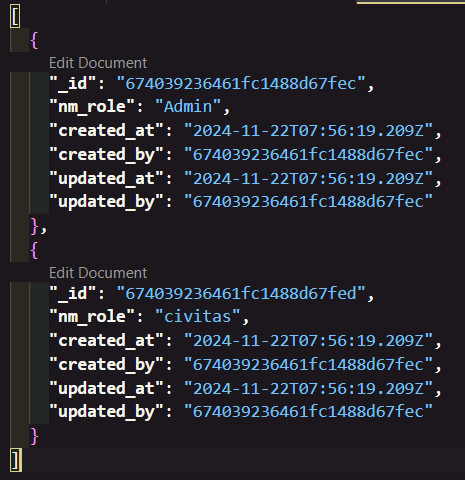
**created\_by:** 1,

**updated\_at:** new Date**()**,

**updated\_by:** 1

    }

**])**;



* Tabel Jenis\_user memiliki nilai: Mahasiswa, Dosen, Tendik, KPS, Dekanat, Ketua\_Unit dan Pimpinan\_univ

*// collection jenis users*

use**(**'unairsatu\_v2'**)**;

db.getCollection**(**'jenis\_user'**)**.insertMany**([**

    {

**"nm\_jenis\_user":** "Mahasiswa",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9985'**)**,  *// aplikasi cyber campus*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9986'**)**,  *// aplikasi pusba elpt*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    },

    {

**"nm\_jenis\_user":** "Dosen",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9985'**)**,  *// aplikasi cyber campus*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9987'**)**,  *// aplikasi dosen*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9988'**)**,  *// aplikasi dosen v3*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

**]**

    },

    {

**"nm\_jenis\_user":** "Tendik",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9985'**)**,  *// aplikasi cyber campus*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9989'**)**,  *// aplikasi e-office*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    },

    {

**"nm\_jenis\_user":** "KPS",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9985'**)**,  *// aplikasi cyber campus*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9989'**)**,  *// aplikasi e-office*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    },

    {

**"nm\_jenis\_user":** "Dekanat",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9989'**)**,  *// aplikasi e-office*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998d'**)**,  *// aplikasi simba*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    },

    {

**"nm\_jenis\_user":** "Ketua\_Unit",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9989'**)**,  *// aplikasi e-office*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998d'**)**,  *// aplikasi simba*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    },

    {

**"nm\_jenis\_user":** "Pimpinan\_univ",

**"templates":** **[**

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9984'**)**,  *// aplikasi email unair*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9985'**)**,  *// aplikasi cyber campus*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e9989'**)**,  *// aplikasi e-office*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998c'**)**,  *// aplikasi dashboard*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998d'**)**,  *// aplikasi simba*

            },

            {

**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998a'**)**,  *// aplikasi vpn unair*

            },

            {

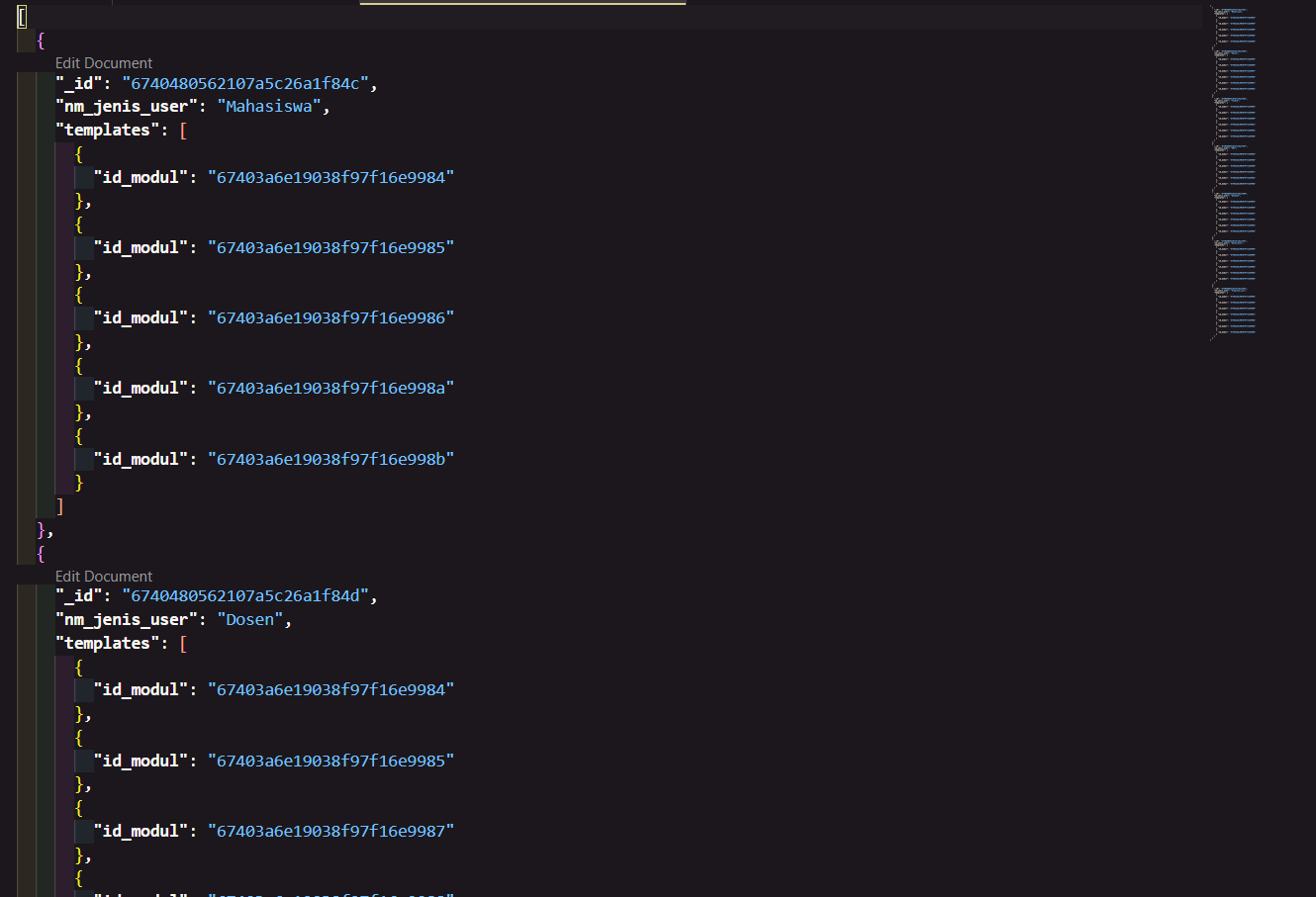
**"id\_modul":** ObjectId**(**'67403a6e19038f97f16e998b'**)**,  *// aplikasi helpdesk unair*

            }

**]**

    }

**])**;



1. Sesuaikan struktur collection user anda dan modelnya, sehingga sesuai dengan disain MongoDB anda

User.go

package models

import (

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

)

type Moduls struct {

**ModulID**  primitive.ObjectID `json:"modul\_id" bson:"modul\_id"`

**NmModul**  string             `json:"nm\_modul" bson:"nm\_modul"`

**KetModul** string             `json:"ket\_modul" bson:"ket\_modul"`

**Alamat**   string             `json:"alamat" bson:"alamat"`

**GbrIcon**  string             `json:"gbr\_icon" bson:"gbr\_icon"`

}

type User struct {

**ID**            primitive.ObjectID `json:"id" bson:"\_id,omitempty"`

**Username**      string             `json:"username" bson:"username"`

**Nm\_user**       string             `json:"nm\_user" bson:"nm\_user"`

**Pass**          string             `json:"pass" bson:"pass"`

**Email**         string             `json:"email" bson:"email"`

**Role\_aktif**    primitive.ObjectID `json:"role\_aktif" bson:"role\_aktif"`

**Created\_at**    primitive.DateTime `json:"created\_at" bson:"created\_at"`

**Updated\_at**    primitive.DateTime `json:"updated\_at" bson:"updated\_at"`

**Created\_by**    primitive.ObjectID `json:"created\_by" bson:"created\_by"`

**Updated\_by**    primitive.ObjectID `json:"updated\_by" bson:"updated\_by"`

**AuthKey**       string             `json:"auth\_key" bson:"auth\_key"`

**Jenis\_kelamin** int                `json:"jenis\_kelamin" bson:"jenis\_kelamin"`

**Photo**         string             `json:"photo" bson:"photo"`

**Phone**         string             `json:"phone" bson:"phone"`

**Token**         string             `json:"token" bson:"token"`

**Id\_jenis\_user** primitive.ObjectID `json:"id\_jenis\_user" bson:"id\_jenis\_user"`

**Pass\_2**        string             `json:"pass\_2" bson:"pass\_2"`

**Moduls**        []Moduls           `json:"moduls" bson:"moduls"`

}

Jenis\_user.go

package models

import (

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

)

*// Template represents a single module template for a user type.*

type Template struct {

**IDModul** primitive.ObjectID `bson:"id\_modul" json:"id\_modul"`

}

*// JenisUser represents a type of user with associated templates.*

type JenisUser struct {

**ID**          primitive.ObjectID  `bson:"\_id" json:"id"`

**NmJenisUser** string              `bson:"nm\_jenis\_user" json:"nm\_jenis\_user"`

**Templates**   []Template          `bson:"templates" json:"templates"`

}

Kategori.go

package models

import (

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

)

*// Kategori represents the structure of the 'kategori' collection in MongoDB*

type Kategori struct {

**ID**        primitive.ObjectID `bson:"\_id,omitempty" json:"id"`

**NmKategori** string            `bson:"nm\_kategori" json:"nm\_kategori"`

}

Modul.go

package models

import (

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

    "time"

)

*// Modul represents the structure of a module document in MongoDB*

type Modul struct {

**ID**           primitive.ObjectID `bson:"\_id" json:"id"`

**IDKategori**   primitive.ObjectID `bson:"id\_kategori" json:"id\_kategori"`

**NmModul**      string             `bson:"nm\_modul" json:"nm\_modul"`

**KetModul**     string             `bson:"ket\_modul" json:"ket\_modul"`

**IsAktif**      string             `bson:"is\_aktif" json:"is\_aktif"`

**Alamat**       string             `bson:"alamat" json:"alamat"`

**Urutan**       int                `bson:"urutan" json:"urutan"`

**GbrIcon**      string             `bson:"gbr\_icon" json:"gbr\_icon"`

**CreatedAt**    time.Time          `bson:"created\_at" json:"created\_at"`

**CreatedBy**    primitive.ObjectID `bson:"created\_by" json:"created\_by"`

**UpdatedAt**    time.Time          `bson:"updated\_at" json:"updated\_at"`

**UpdatedBy**    primitive.ObjectID `bson:"updated\_by" json:"updated\_by"`

**Icon**         string             `bson:"icon" json:"icon"`

}

Role.go

package models

import (

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

    "time"

)

*// Role represents the structure of the 'role' collection in MongoDB*

type Role struct {

**ID**        primitive.ObjectID `bson:"\_id,omitempty" json:"id"`

**NmRole**    string             `bson:"nm\_role" json:"nm\_role"`

**CreatedAt** time.Time          `bson:"created\_at" json:"created\_at"`

**CreatedBy** primitive.ObjectID `bson:"created\_by" json:"created\_by"`

**UpdatedAt** time.Time          `bson:"updated\_at" json:"updated\_at"`

**UpdatedBy** primitive.ObjectID `bson:"updated\_by" json:"updated\_by"`

}

1. Buatlah 2 middleware yang memiliki parameter:
2. checkRole(role string) → memeriksa, apakah role yang digunakan sesuai dengan nilai role pada parameter

*// Middleware to check role*

func CheckRole(role string) func(http.Handler) http.Handler {

    return func(next http.Handler) http.Handler {

        return http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

*// Get Authorization header*

            authHeader := r.Header.Get("Authorization")

            if authHeader == "" {

                http.Error(w, "Authorization header missing", http.StatusUnauthorized)

                return

            }

*// Decode token (assuming it's base64-encoded JSON)*

            parts := strings.Split(authHeader, " ")

            if len(parts) != 2 || parts[0] != "Bearer" {

                http.Error(w, "Invalid Authorization header format", http.StatusUnauthorized)

                return

            }

            token := parts[1]

            data, err := base64.StdEncoding.DecodeString(token)

            if err != nil {

                http.Error(w, "Failed to decode token", http.StatusUnauthorized)

                return

            }

*// Parse token data*

            var payload map[string]interface{}

            err = json.Unmarshal(data, &payload)

            if err != nil {

                http.Error(w, "Failed to parse token", http.StatusUnauthorized)

                return

            }

*// Check role*

            if payload["role"] != role {

                http.Error(w, "Unauthorized role", http.StatusForbidden)

                return

            }

            next.ServeHTTP(w, r)

        })

    }

}

1. checkJenis\_user(ju string) → memeriksa, apakah jenis user yang digunakan sesuai dengan jenis\_user pada parameter

*// Middleware to check jenis\_user*

func CheckJenisUser(ju string) func(http.Handler) http.Handler {

    return func(next http.Handler) http.Handler {

        return http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

*// Get Authorization header*

            authHeader := r.Header.Get("Authorization")

            if authHeader == "" {

                http.Error(w, "Authorization header missing", http.StatusUnauthorized)

                return

            }

*// Decode token (assuming it's base64-encoded JSON)*

            parts := strings.Split(authHeader, " ")

            if len(parts) != 2 || parts[0] != "Bearer" {

                http.Error(w, "Invalid Authorization header format", http.StatusUnauthorized)

                return

            }

            token := parts[1]

            data, err := base64.StdEncoding.DecodeString(token)

            if err != nil {

                http.Error(w, "Failed to decode token", http.StatusUnauthorized)

                return

            }

*// Parse token data*

            var payload map[string]interface{}

            err = json.Unmarshal(data, &payload)

            if err != nil {

                http.Error(w, "Failed to parse token", http.StatusUnauthorized)

                return

            }

*// Check jenis\_user*

            if payload["jenis\_user"] != ju {

                http.Error(w, "Unauthorized jenis\_user", http.StatusForbidden)

                return

            }

            next.ServeHTTP(w, r)

        })

    }

}

1. Buatlah group route untuk Admin dan gunakan middleware pada point 3.a.
2. Pindahkan semua route terkait dengan CRUD pada users ke group ini

    admin := app.Group("/admin")

    admin.Use(middleware.AuthMiddleware) *// Middleware diterapkan di seluruh grup Admin*

*// Routes CRUD Users*

    admin.Post("/createUser", controllers.CreateUser)

    admin.Get("/getAllUser", controllers.GetUsers)

    admin.Get("/getUser/:id", controllers.GetUserOne)

    admin.Put("/updateUser/:id", controllers.UpdateUser)

    admin.Put("/changePassword/:id", controllers.ChangePassword)

    admin.Put("/uploadPhoto/:id", controllers.UploadPhoto)

    admin.Delete("/deleteUser/:id", controllers.DeleteUser)

1. Buat fungsi untuk CRUD collection modul. Modul adalah collection yang berisi semua opsi aplikasi pada halaman unairsatu

package controllers

import (

    "context"

    "net/http"

    "project-crud\_baru/models"

    "time"

    "github.com/gofiber/fiber/v2"

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

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

    "go.mongodb.org/mongo-driver/mongo"

)

*// MongoDB collection reference*

var modulCollection \*mongo.Collection

*// Initialize the MongoDB collection*

func InitModulCollection(db \*mongo.Database) {

    modulCollection = db.Collection("modul")

}

*// Create a new module*

func CreateModul(c \*fiber.Ctx) error {

    var modul models.Modul

*// Parse the request body*

    if err := c.BodyParser(&modul); err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid request body"})

    }

*// Set additional fields*

    modul.ID = primitive.NewObjectID()

    modul.CreatedAt = primitive.NewDateTimeFromTime(time.Now())

    modul.UpdatedAt = primitive.NewDateTimeFromTime(time.Now())

*// Insert the module into the database*

    \_, err := modulCollection.InsertOne(context.Background(), modul)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to create modul"})

    }

    return c.Status(http.StatusCreated).JSON(modul)

}

*// Get all modules*

func GetAllModul(c \*fiber.Ctx) error {

    var modules []models.Modul

*// Retrieve all documents from the modul collection*

    cursor, err := modulCollection.Find(context.Background(), bson.M{})

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to retrieve modules"})

    }

    defer cursor.Close(context.Background())

*// Iterate through the cursor and decode each document*

    for cursor.Next(context.Background()) {

        var modul models.Modul

        if err := cursor.Decode(&modul); err != nil {

            return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to decode modul"})

        }

        modules = append(modules, modul)

    }

    return c.JSON(modules)

}

*// Get a single module by ID*

func GetModulByID(c \*fiber.Ctx) error {

    id := c.Params("id")

*// Convert the string ID to ObjectID*

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    var modul models.Modul

*// Find the modul by ID*

    err = modulCollection.FindOne(context.Background(), bson.M{"\_id": objID}).Decode(&modul)

    if err != nil {

        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "Modul not found"})

    }

    return c.JSON(modul)

}

*// Update a module by ID*

func UpdateModul(c \*fiber.Ctx) error {

    id := c.Params("id")

*// Convert the string ID to ObjectID*

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    var modul models.Modul

*// Parse the request body*

    if err := c.BodyParser(&modul); err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid request body"})

    }

*// Update the UpdatedAt field*

    modul.UpdatedAt = primitive.NewDateTimeFromTime(time.Now())

*// Create an update document*

    update := bson.M{

        "$set": bson.M{

            "id\_kategori": modul.IDKategori,

            "nm\_modul":    modul.NmModul,

            "ket\_modul":   modul.KetModul,

            "is\_aktif":    modul.IsAktif,

            "alamat":      modul.Alamat,

            "urutan":      modul.Urutan,

            "gbr\_icon":    modul.GbrIcon,

            "updated\_at":  modul.UpdatedAt,

            "updated\_by":  modul.UpdatedBy,

            "icon":        modul.Icon,

        },

    }

*// Update the modul in the database*

    \_, err = modulCollection.UpdateOne(context.Background(), bson.M{"\_id": objID}, update)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to update modul"})

    }

    return c.JSON(fiber.Map{"message": "Modul updated successfully"})

}

*// Delete a module by ID*

func DeleteModul(c \*fiber.Ctx) error {

    id := c.Params("id")

*// Convert the string ID to ObjectID*

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

*// Delete the modul from the database*

    \_, err = modulCollection.DeleteOne(context.Background(), bson.M{"\_id": objID})

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to delete modul"})

    }

    return c.JSON(fiber.Map{"message": "Modul deleted successfully"})

}

1. Buat fungsi untuk CRUD aplikasi dari collection modul ke collection template\_modul. Template\_modul ini berisi list aplikasi template pada setiap

package controllers

import (

    "context"

    "net/http"

    "github.com/gofiber/fiber/v2"

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

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

    "go.mongodb.org/mongo-driver/mongo"

    "project-crud\_baru/config"

    "project-crud\_baru/models"

)

var jenisUserCollection \*mongo.Collection = config.GetCollection("jenis\_user")

*// CreateJenisUser handles creating a new JenisUser*

func CreateJenisUser(c \*fiber.Ctx) error {

    var jenisUser models.JenisUser

    if err := c.BodyParser(&jenisUser); err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": err.Error()})

    }

*// Set ID as an ObjectID*

    jenisUser.ID = primitive.NewObjectID()

*// Insert into the database*

    \_, err := jenisUserCollection.InsertOne(context.Background(), jenisUser)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    return c.Status(http.StatusCreated).JSON(fiber.Map{"message": "JenisUser created successfully", "id": jenisUser.ID})

}

*// GetJenisUsers handles retrieving all JenisUser*

func GetJenisUsers(c \*fiber.Ctx) error {

    var jenisUsers []models.JenisUser

    cursor, err := jenisUserCollection.Find(context.Background(), bson.M{})

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    if err := cursor.All(context.Background(), &jenisUsers); err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"data": jenisUsers})

}

*// GetJenisUser handles retrieving a single JenisUser by ID*

func GetJenisUser(c \*fiber.Ctx) error {

    id := c.Params("id")

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    var jenisUser models.JenisUser

    err = jenisUserCollection.FindOne(context.Background(), bson.M{"\_id": objID}).Decode(&jenisUser)

    if err != nil {

        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "JenisUser not found"})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"data": jenisUser})

}

*// UpdateJenisUser handles updating a JenisUser by ID*

func UpdateJenisUser(c \*fiber.Ctx) error {

    id := c.Params("id")

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    var jenisUser models.JenisUser

    if err := c.BodyParser(&jenisUser); err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": err.Error()})

    }

*// Update the document in the collection*

    update := bson.M{

        "$set": bson.M{

            "nm\_jenis\_user": jenisUser.NmJenisUser,

            "templates":     jenisUser.Templates,

        },

    }

    \_, err = jenisUserCollection.UpdateOne(context.Background(), bson.M{"\_id": objID}, update)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "JenisUser updated successfully"})

}

*// DeleteJenisUser handles deleting a JenisUser by ID*

func DeleteJenisUser(c \*fiber.Ctx) error {

    id := c.Params("id")

    objID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    \_, err = jenisUserCollection.DeleteOne(context.Background(), bson.M{"\_id": objID})

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "JenisUser deleted successfully"})

}

1. jenis\_user. Silahkan tentukan sendiri, apakah anda akan melakukan secara bulk atau satuan
2. Buat fungsi untuk menampilkan semua modul yang dimiliki oleh seorang user

func GetUserModules(c \*fiber.Ctx) error {

    ctx, cancel := context.WithTimeout(context.Background(), 10\*time.Second)

    defer cancel()

*// Retrieve user ID from URL parameters*

    id := c.Params("id")

    userID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

*// Find user by ID*

    var user models.User

    err = userCollection.FindOne(ctx, bson.M{"\_id": userID}).Decode(&user)

    if err != nil {

*// If user not found, return error message*

        if err == mongo.ErrNoDocuments {

            return c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "User not found"})

        }

*// Other errors (e.g., database issues)*

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

*// Return the user's modules if user found*

    if len(user.Moduls) == 0 {

        return c.Status(http.StatusOK).JSON(fiber.Map{"message": "User has no modules"})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"moduls": user.Moduls})

}

1. Buat fungsi untuk pindah jenis\_user. Hal yang dilakukan ketika pindah user adalah sbb:
   * 1. Tentukan user mana akan dipindahkan atau diberi jenis\_user yang mana
     2. Hapus semua modul yang dimiliki oleh user tersebut
     3. Tambahkan modul baru ke user tersebut, sesuai dengan modul yang ada pada template\_modul pada jenis\_user tersebut

func ChangeJenisUser(c \*fiber.Ctx) error {

    ctx, cancel := context.WithTimeout(context.Background(), 10\*time.Second)

    defer cancel()

*// Retrieve user ID and new jenis\_user ID from URL parameters*

    id := c.Params("id")

    userID, err := primitive.ObjectIDFromHex(id)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid ID format"})

    }

    idJenisUser := c.Params("id\_jenis\_user")

    jenisUserID, err := primitive.ObjectIDFromHex(idJenisUser)

    if err != nil {

        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error": "Invalid jenis\_user ID format"})

    }

*// Find user by ID*

    var user models.User

    err = userCollection.FindOne(ctx, bson.M{"\_id": userID}).Decode(&user)

    if err != nil {

*// If user not found, return error message*

        if err == mongo.ErrNoDocuments {

            return c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "User not found"})

        }

*// Other errors (e.g., database issues)*

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

*// Find jenis\_user by ID*

    jenisUserCollection := config.GetCollection("jenis\_user")

    var jenisUser models.JenisUser

    err = jenisUserCollection.FindOne(ctx, bson.M{"\_id": jenisUserID}).Decode(&jenisUser)

    if err != nil {

*// If jenis\_user not found, return error message*

        if err == mongo.ErrNoDocuments {

            return c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "Jenis\_user not found"})

        }

*// Other errors (e.g., database issues)*

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

*// Remove all existing modules from the user*

    update := bson.M{"$set": bson.M{"moduls": []models.Moduls{}}}

    \_, err = userCollection.UpdateOne(ctx, bson.M{"\_id": userID}, update)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

*// Add new modules to the user based on the jenis\_user's template\_modul*

    update = bson.M{"$set": bson.M{"moduls": jenisUser.Templates}}

    \_, err = userCollection.UpdateOne(ctx, bson.M{"\_id": userID }, update)

    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "Jenis\_user changed successfully"})

}

1. Buat fungsi untuk CUD pada modul yang dimiliki oleh user. Fungsi ini bertujuan untuk menambahkan satu modul ke user tersebut. Terkadang terdapat user yang memiliki privilege khusus untuk mengakses modul tertentu. Contoh, dosen biasa yang ditunjuk sebagai salah satu tim satu data, sehingga dosen tersebut memiliki akses ke satu data dimana sebenarnya satu data hanya diberikan kepada jenis\_user kps, dekanat, ketua\_unit dan pimpinan\_univ