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. Buatlah API untuk :
   a. Select 1 user berdasarkan ID nya
      Controller userController

```
// Get User by ID
func GetUserOne(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID"})
    }

    var user models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&user)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"User not found"})
    }

    return c.Status(http.StatusOK).JSON(user)
}
```

      Route

```
    // get user by id
    Users.Get("/getUser/:id", controllers.GetUserOne)
```

   b. Update data user berdasarkan ID nya
      Controller UserController

```
// Update User by ID
func UpdateUser(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
```

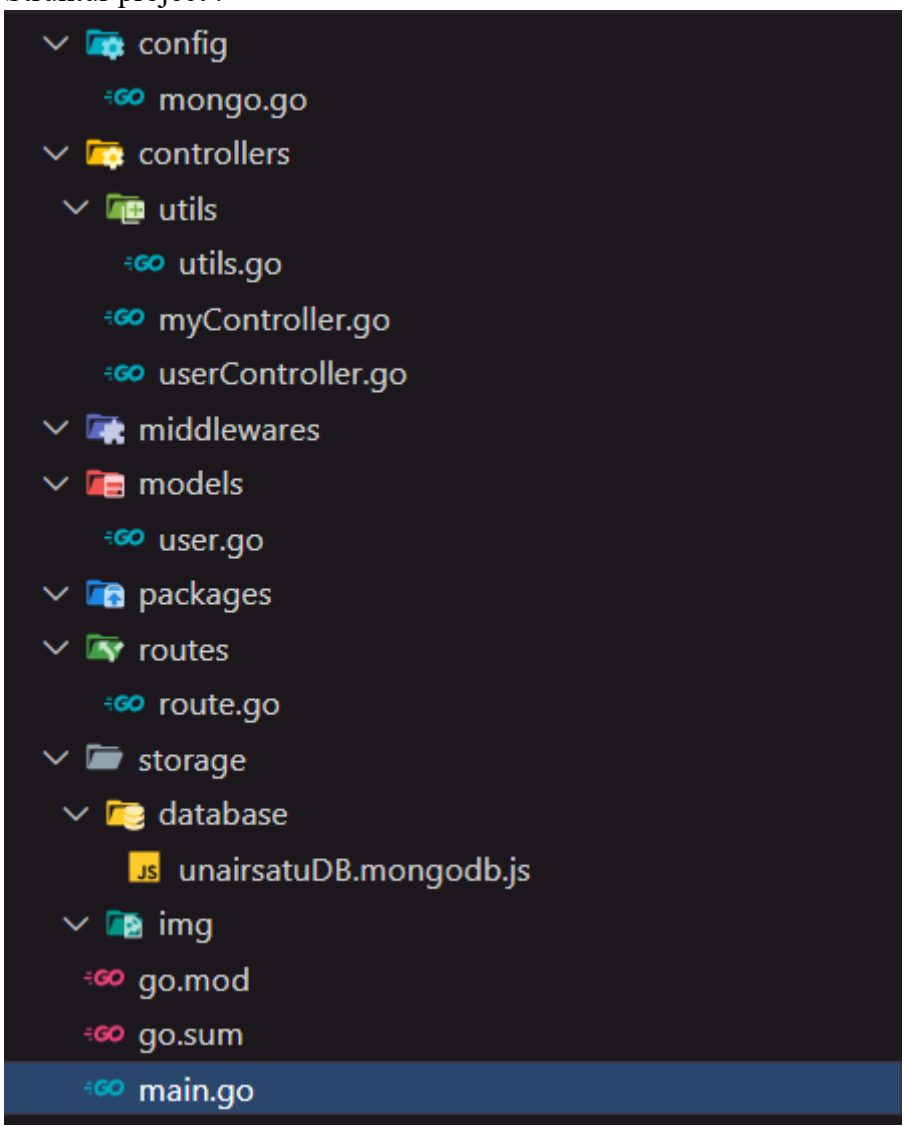
```
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID"})
    }

    // Menerima data pengguna yang baru dari request body
    var user models.User
    if err := c.BodyParser(&user); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Membuat update data
    update := bson.M{
        "username":      user.Username,
        "nm_user":       user.Nm_user,
        "pass":          user.Pass,
        "email":         user.Email,
        "role_aktif":    user.Role_aktif,
        "jenis_kelamin": user.Jenis_kelamin,
        "photo":         user.Photo,
        "phone":         user.Phone,
        "pass_2":       user.Pass_2,
    }

    // Melakukan update
    result, err := userCollection.UpdateOne(ctx, bson.M{"_id": userID},
bson.M{"$set": update})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Jika tidak ada dokumen yang terpengaruh, pengguna tidak
ditemukan
    if result.MatchedCount == 0 {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"User not found"})
    }

    // Mengambil pengguna yang diperbarui untuk dikembalikan
    var updatedUser models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&updatedUser)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
```

```
    }

    return c.Status(http.StatusOK).JSON(updatedUser)
}
```

Route

```
    // update user by id
    Users.Put("/updateUser/:id", controllers.UpdateUser)
```

2. Test dan screenshot hasilnya. Yang anda screenshot
   Struktur project :



   a. Code anda
      Controller :

```
package controllers

import (
    "context"
```

```
    "net/http"
    "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"

    "project-crud/config"
    "project-crud/controllers/utils"
    "project-crud/models"
)

var userCollection *mongo.Collection = config.GetCollection("users")

// Create User
func CreateUser(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    var user models.User
    if err := c.BodyParser(&user); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Parse id_jenis_user dari string ke ObjectID
    idJenisUser, err :=
primitive.ObjectIDFromHex(user.Id_jenis_user.Hex())
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid id_jenis_user format"})
    }
    user.Id_jenis_user = idJenisUser

    loc, err := time.LoadLocation("Asia/Jakarta")
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    user.Created_at = primitive.NewDateTimeFromTime(time.Now().In(loc))

    // Generate token acak
    token, err := utils.GenerateRandomString(32)
    if err != nil {
```

```
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to generate token"})
    }

    newUser := models.User{
        ID:            primitive.NewObjectID(),
        Username:      user.Username,
        Nm_user:       user.Nm_user,
        Pass:          user.Pass,
        Email:         user.Email,
        Role_aktif:    user.Role_aktif,
        Created_at:    primitive.NewDateTimeFromTime(time.Now()),
        Jenis_kelamin: user.Jenis_kelamin,
        Photo:         user.Photo,
        Phone:         user.Phone,
        Token:         token,
        Id_jenis_user: user.Id_jenis_user,
        Pass_2:        user.Pass_2,
    }

    _, errIns := userCollection.InsertOne(ctx, newUser)
    if errIns != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
errIns.Error()})
    }

    return c.Status(http.StatusCreated).JSON(newUser)
}

// Get All Users
func GetUsers(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    var users []models.User
    cursor, err := userCollection.Find(ctx, bson.M{})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    if err = cursor.All(ctx, &users); err != nil {
```

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

    return c.Status(http.StatusOK).JSON(users)
}

// Get User by ID
func GetUserOne(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID"})
    }

    var user models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&user)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"User not found"})
    }

    return c.Status(http.StatusOK).JSON(user)
}

// Update User by ID
func UpdateUser(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID"})
    }

    // Menerima data pengguna yang baru dari request body
    var user models.User
    if err := c.BodyParser(&user); err != nil {
```

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

    // Membuat update data
    update := bson.M{
        "username":      user.Username,
        "nm_user":       user.Nm_user,
        "pass":          user.Pass,
        "email":         user.Email,
        "role_aktif":    user.Role_aktif,
        "jenis_kelamin": user.Jenis_kelamin,
        "photo":         user.Photo,
        "phone":         user.Phone,
        "pass_2":       user.Pass_2,
    }

    // Melakukan update
    result, err := userCollection.UpdateOne(ctx, bson.M{"_id": userID},
bson.M{"$set": update})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Jika tidak ada dokumen yang terpengaruh, pengguna tidak
ditemukan
    if result.MatchedCount == 0 {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"User not found"})
    }

    // Mengambil pengguna yang diperbarui untuk dikembalikan
    var updatedUser models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&updatedUser)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    return c.Status(http.StatusOK).JSON(updatedUser)
}
```

Model :

```go
package models

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

// User represents a user in MongoDB
type User struct {
    ID           primitive.ObjectID `bson:"_id,omitempty"`
    Username     string             `json:"username"`
    Nm_user      string             `json:"nm_user"`
    Pass         string             `json:"pass"`
    Email        string             `json:"email"`
    Role_aktif   int                `json:"role_aktif"`
    Created_at   primitive.DateTime `json:"created_at"`
    Jenis_kelamin int               `json:"jenis_kelamin"`
    Photo        string             `json:"photo"`
    Phone        string             `json:"phone"`
    Token        string             `json:"token"`
    Id_jenis_user primitive.ObjectID `json:"id_jenis_user"`
    Pass_2       string             `json:"pass_2"`
}
```

Utils

```go
package utils

import (
    "crypto/rand"
    "encoding/hex"
)

// Fungsi untuk menghasilkan string acak dengan panjang tertentu
func GenerateRandomString(length int) (string, error) {
    bytes := make([]byte, length/2)
    if _, err := rand.Read(bytes); err != nil {
        return "", err
    }
    return hex.EncodeToString(bytes), nil
}
```

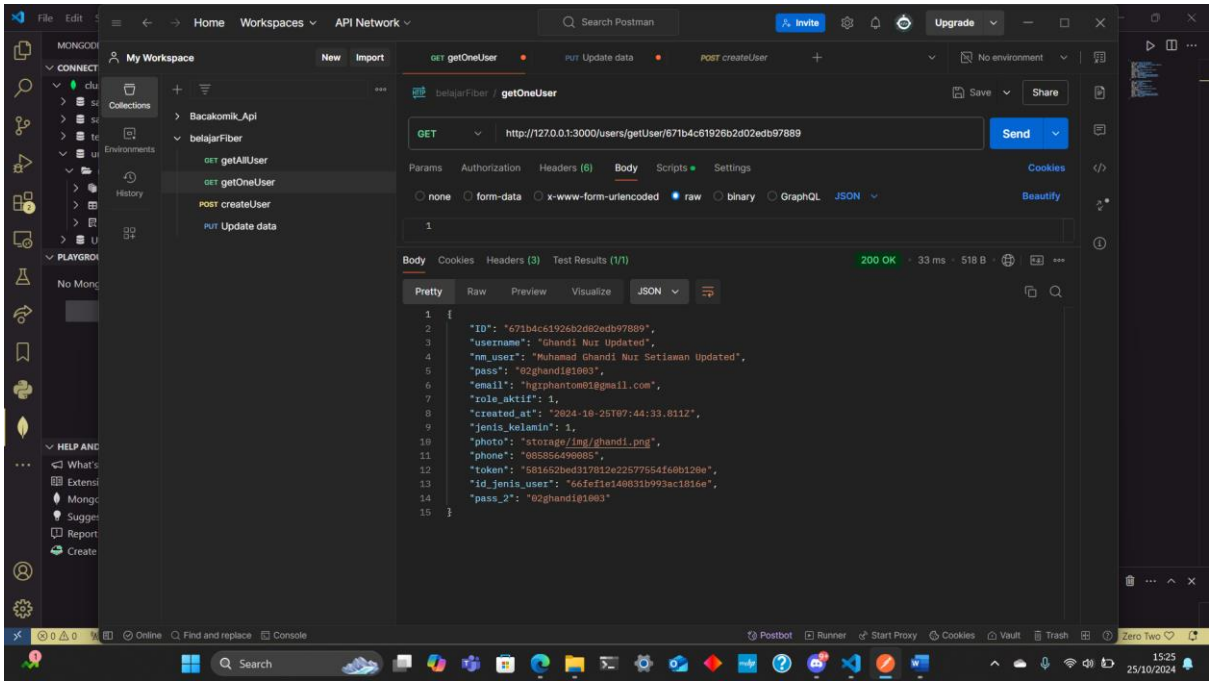
Main :

```go
package main

import (
    "project-crud/routes"
    "github.com/gofiber/fiber/v2"
```

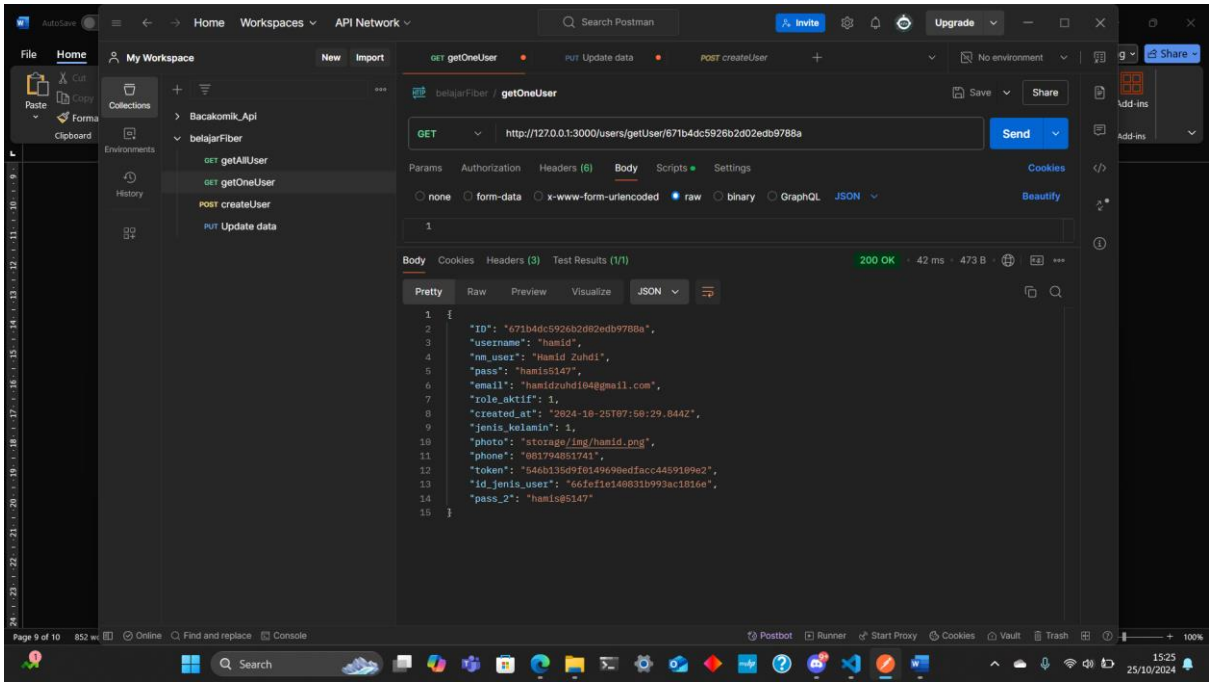
```
)

func main() {
    app := fiber.New()
    routes.RouteApp(app)
    app.Listen(":3000")
}
```

b. Request pada postman
Menampilkan user by id

- User 1 :

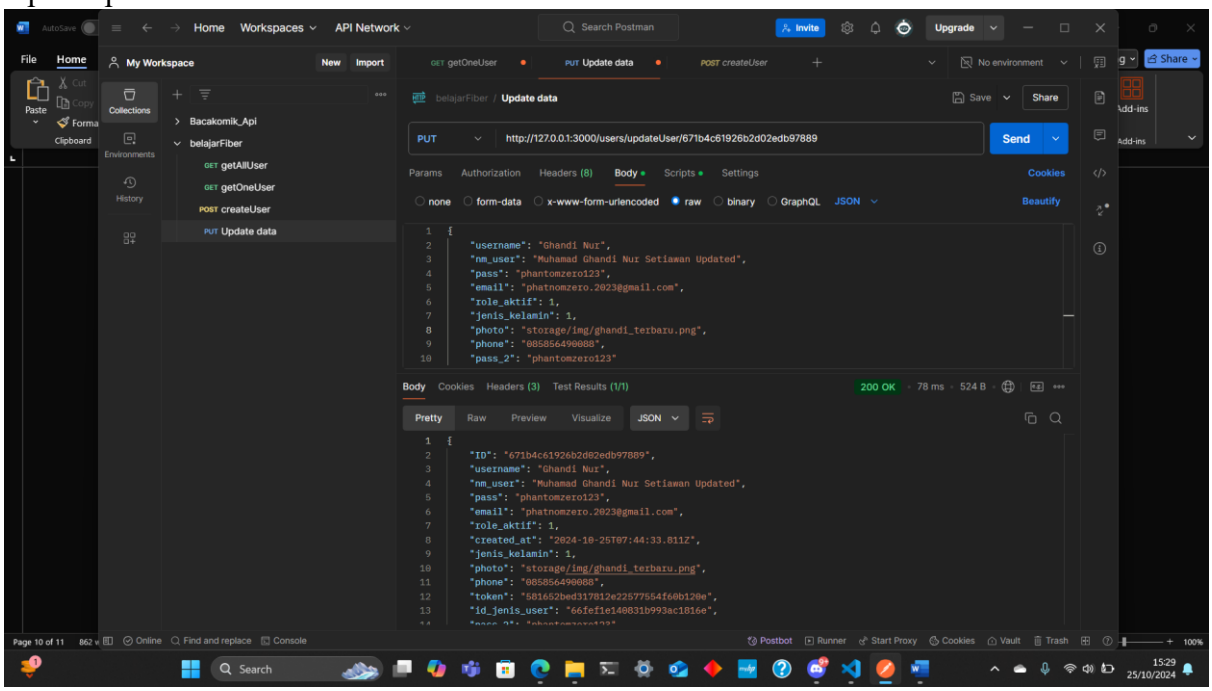


- User 2 :



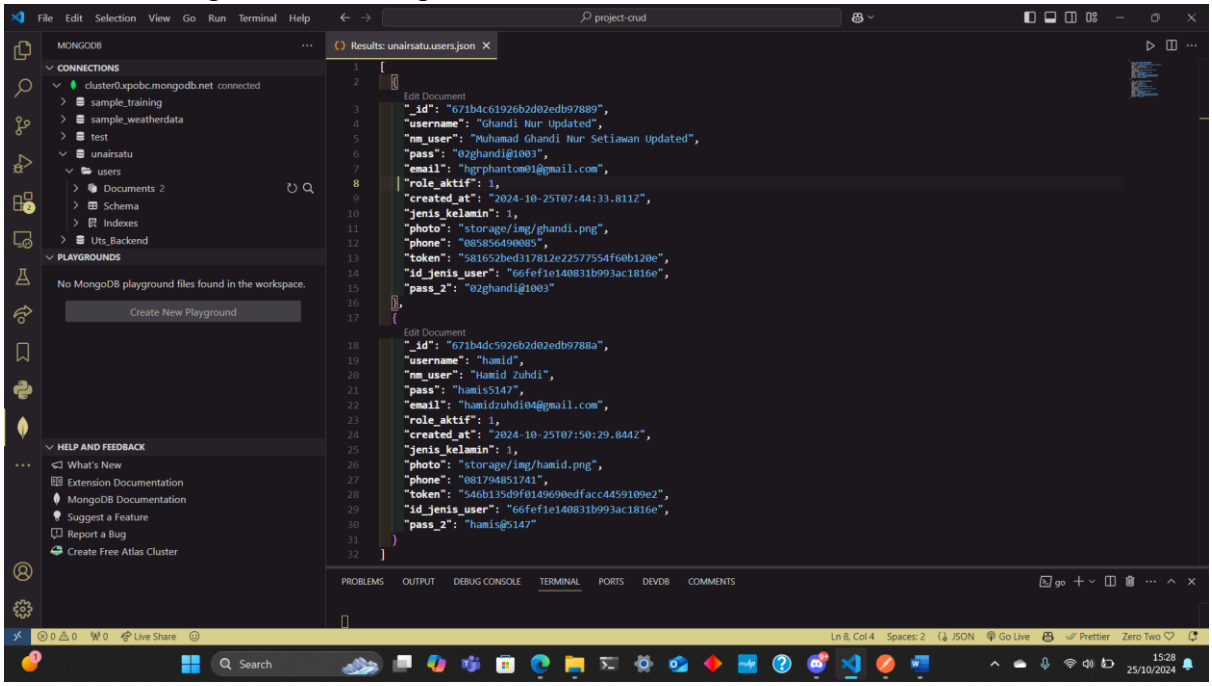
Update user by id

Update pada User 1 :



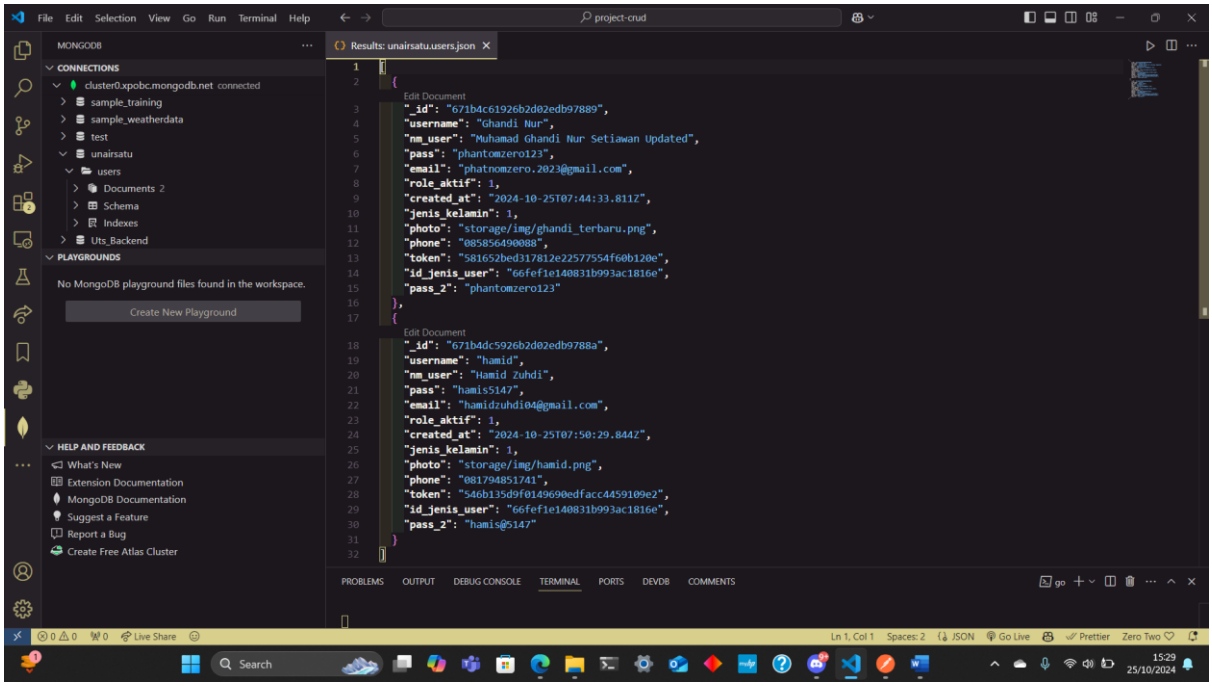
c. Collection pada mongoDB sebelum dan sesudah API dipanggil

Collection MongoDB Setelah Input :



Collection MongoDB Setelah Update :

Update pada user 1 :


File Edit Selection View Go Run Terminal Help
project-crud
MONGODB
CONNECTIONS
cluster0.xpobc.mongodb.net connected
sample_training
sample_weatherdata
test
unairsatu
users
Documents 2
Schema
Indexes
Uts_Backend
PLAYGROUNDS
No MongoDB playground files found in the workspace.
Create New Playground
HELP AND FEEDBACK
What's New
Extension Documentation
MongoDB Documentation
Suggest a Feature
Report a Bug
Create Free Atlas Cluster
Results: unairsatu.users.json
[
{
Edit Document
"_id": "671b4c61926b2d02edb97889",
"username": "Ghandi Nur",
"nm_user": "Muhamad Ghandi Nur Setiawan Updated",
"pass": "phantomzero123",
"email": "phatnomzero.2023@gmail.com",
"role_aktif": 1,
"created_at": "2024-10-25T07:44:33.811Z",
"jenis_kelamin": 1,
"photo": "storage/img/ghandi_terbaru.png",
"phone": "085856490088",
"token": "581652bed317812e22577554f60b120e",
"id_jenis_user": "66fef1e140831b993ac1816e",
"pass_2": "phantomzero123"
},
{
Edit Document
"_id": "671b4dc5926b2d02edb9788a",
"username": "hamid",
"nm_user": "Hamid Zuhdi",
"pass": "hamis5147",
"email": "hamidzuhdi04@gmail.com",
"role_aktif": 1,
"created_at": "2024-10-25T07:50:29.844Z",
"jenis_kelamin": 1,
"photo": "storage/img/hamid.png",
"phone": "081794851741",
"token": "546b135d9f0149690edfacc4459109e2",
"id_jenis_user": "66fef1e140831b993ac1816e",
"pass_2": "hamis@5147"
}
]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS DEVDB COMMENTS
Ln 1, Col 1 Spaces: 2 JSON Go Live Prettier Zero Two
Search
15:29
25/10/2024