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. Implementasikan fungsi generate token JWT dari praktikum kedalam fungsi !
    a. Buat fungsi generate token JWT secara manual, jangan menggunakan library yang ada!

```
// Fungsi untuk meng-encode data ke base64 tanpa padding
func base64Encode(data []byte) string {
    return base64.RawURLEncoding.EncodeToString(data)
}

// Fungsi untuk membuat signature menggunakan HMAC-SHA256
func CreateSignature(header, payload, secret string) string {
    h := hmac.New(sha256.New, []byte(secret))
    h.Write([]byte(header + "." + payload))
    return base64Encode(h.Sum(nil))
}

// Fungsi untuk generate token JWT
func GenerateJWT(username string) (string, error) {
    // Header
    header := map[string]string{"alg": "HS256", "typ": "JWT"}
    headerJSON, _ := json.Marshal(header)
    headerEncoded := base64Encode(headerJSON)

    // Payload
    payload := map[string]interface{}{
        "username": username,
        "exp":      time.Now().Add(time.Hour * 1).Unix(), // Expiry 1
jam
    }
    payloadJSON, _ := json.Marshal(payload)
    payloadEncoded := base64Encode(payloadJSON)

    // Signature
    secret := "your_secret_key" // Ganti dengan secret key yang aman
    signature := CreateSignature(headerEncoded, payloadEncoded, secret)

    // Token JWT (Header.Payload.Signature)
    token := fmt.Sprintf("%s.%s.%s", headerEncoded, payloadEncoded,
signature)
    return token, nil
}
```

    b. Fungsi generate token akan memberi return berupa token JWT

```
// Fungsi untuk generate token JWT
func GenerateJWT(username string) (string, error) {
    // Header
```

```
    header := map[string]string{"alg": "HS256", "typ": "JWT"}
    headerJSON, _ := json.Marshal(header)
    headerEncoded := base64Encode(headerJSON)

    // Payload
    payload := map[string]interface{}{
        "username": username,
        "exp":      time.Now().Add(time.Hour * 1).Unix(), // Expiry 1
jam
    }
    payloadJSON, _ := json.Marshal(payload)
    payloadEncoded := base64Encode(payloadJSON)

    // Signature
    secret := "your_secret_key" // Ganti dengan secret key yang aman
    signature := CreateSignature(headerEncoded, payloadEncoded, secret)

    // Token JWT (Header.Payload.Signature)
    token := fmt.Sprintf("%s.%s.%s", headerEncoded, payloadEncoded,
signature)
    return token, nil
}
```

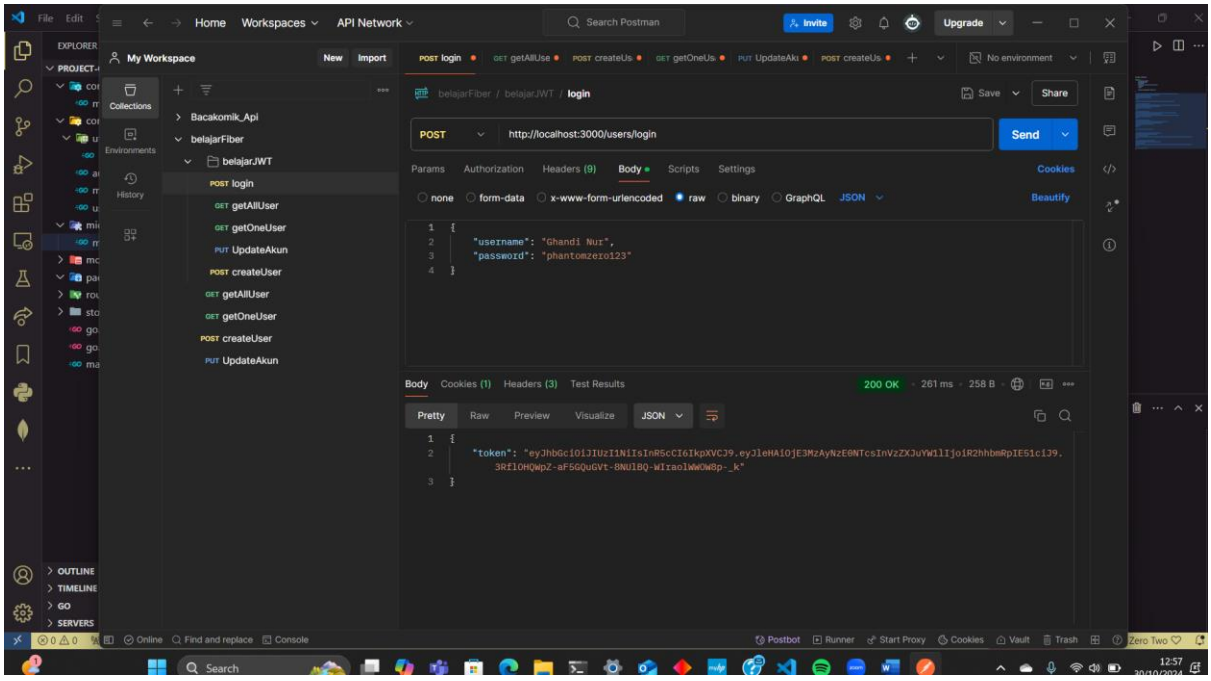
c. Buatlah sebuah API login dengan input username dan password. Gunakan comparison sederhana dengan melakukan query ke collection yang memiliki username dan password yang sama
   i. Jika hasil query menghasilkan sebuah document, gunakan username dari document tersebut untuk memanggil fungsi untuk generate token dan menggunakan token tersebut sebagai output API login
   ii. Jika query tidak menghasilkan, berikan output not found
   iii. Gunakan return http code yang sesuai!

```
func Login(c *fiber.Ctx) error {
    var input struct {
        Username string `json:"username"`
        Password string `json:"password"`
    }
    if err := c.BodyParser(&input); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Bad request"})
    }

    // Cek username dan password di database
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    var user bson.M
```

```
    err := config.GetCollection("users").FindOne(ctx,
bson.M{"username": input.Username, "pass":
input.Password}).Decode(&user)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"Not found"})
    }

    // Generate token JWT
    token, err := utils.GenerateJWT(input.Username)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to generate token"})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"token": token})
}
```

```
package routes

import (
    "project-crud/controllers"
    "project-crud/middlewares" // Pastikan untuk mengimpor middleware
    "github.com/gofiber/fiber/v2"
)

func RouteApp(app *fiber.App) {
    api := app.Group("/api")
    api.Get("/", controllers.HomeFunc)

    // User tanpa autentikasi
    // api.Post("/users/login", controllers.Login) // Pastikan login
berada di luar grup yang menggunakan middleware

    // User dengan autentikasi
    Users := app.Group("/users")
    // login
    Users.Post("/login", controllers.Login)

    Users.Use(middleware.AuthMiddleware) // Semua route dalam grup ini
akan memerlukan token

    Users.Post("/createUser", controllers.CreateUser)
    Users.Get("/getAllUser", controllers.GetUsers)
    Users.Get("/getUser/:id", controllers.GetUserOne)
    Users.Put("/updateUser/:id", controllers.UpdateUser)
}
```

Output Postman :



2. Middleware
    a. Buatlah sebuah middleware untuk melakukan validasi token JWT

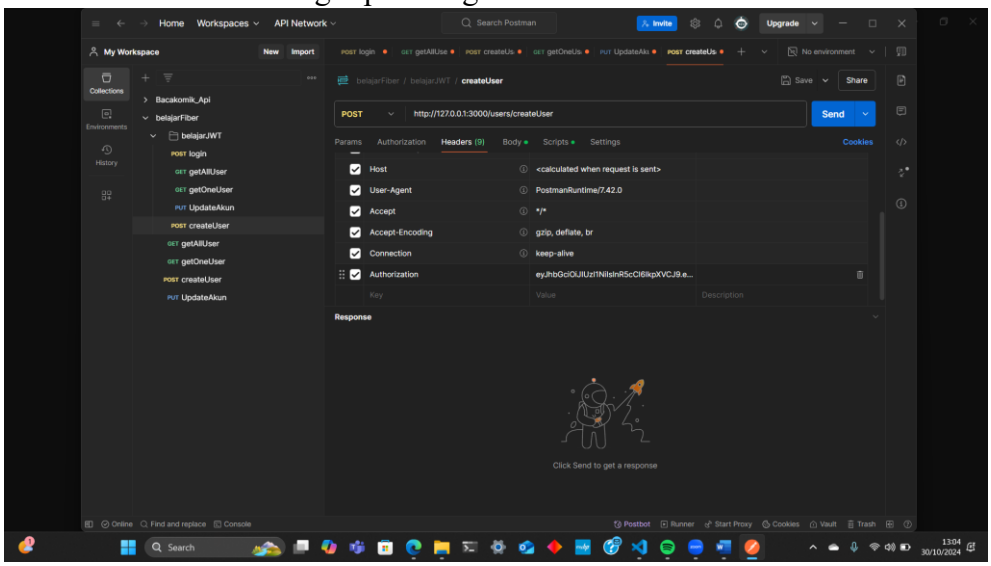
```go
func VerifyJWT(token, secret string) (map[string]interface{}, bool) {
    parts := strings.Split(token, ".")
    if len(parts) != 3 {
        return nil, false
    }

    header, payload, signature := parts[0], parts[1], parts[2]

    // Verify signature
    expectedSignature := utils.CreateSignature(header, payload, secret)
    if signature != expectedSignature {
        return nil, false
    }

    // Decode payload
    payloadJSON, _ := base64.RawURLEncoding.DecodeString(payload)
    var payloadData map[string]interface{}
    if err := json.Unmarshal(payloadJSON, &payloadData); err != nil {
        return nil, false
    }

    // Check expiration
    if exp, ok := payloadData["exp"].(float64); ok {
```

```
        if int64(exp) < time.Now().Unix() {
            return nil, false
        }
    }

    return payloadData, true
}

func AuthMiddleware(c *fiber.Ctx) error {
    token := c.Get("Authorization")
    if token == "" {
        return
c.Status(http.StatusUnauthorized).JSON(fiber.Map{"error": "Missing
token"})
    }

    secret := "your_secret_key"
    _, valid := VerifyJWT(token, secret)
    if !valid {
        return
c.Status(http.StatusUnauthorized).JSON(fiber.Map{"error": "Invalid
token"})
    }

    return c.Next()
}
```

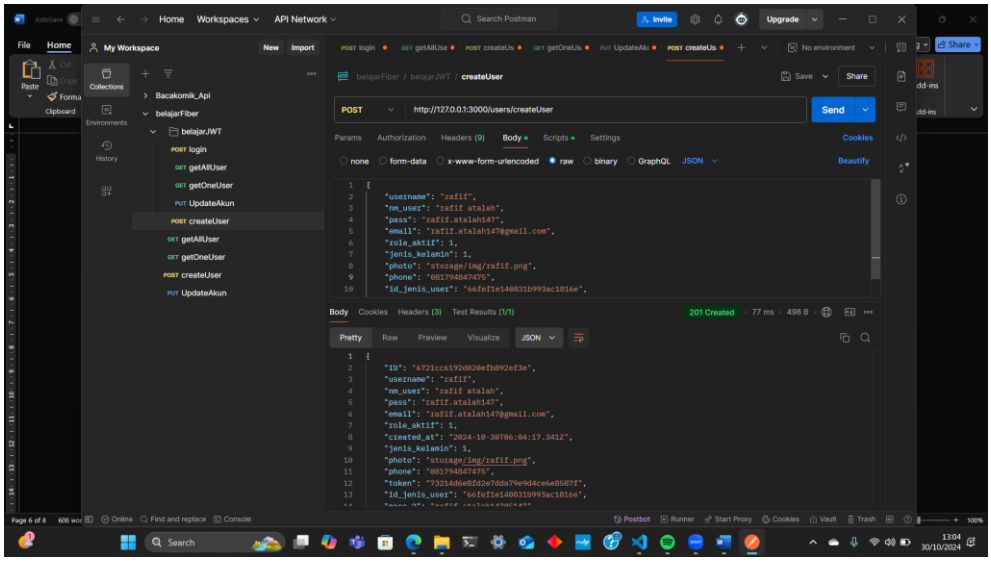
b. Gunakan middleware tersebut pada route untuk:

i. Create user

Menambahkan Token login pada bagian header

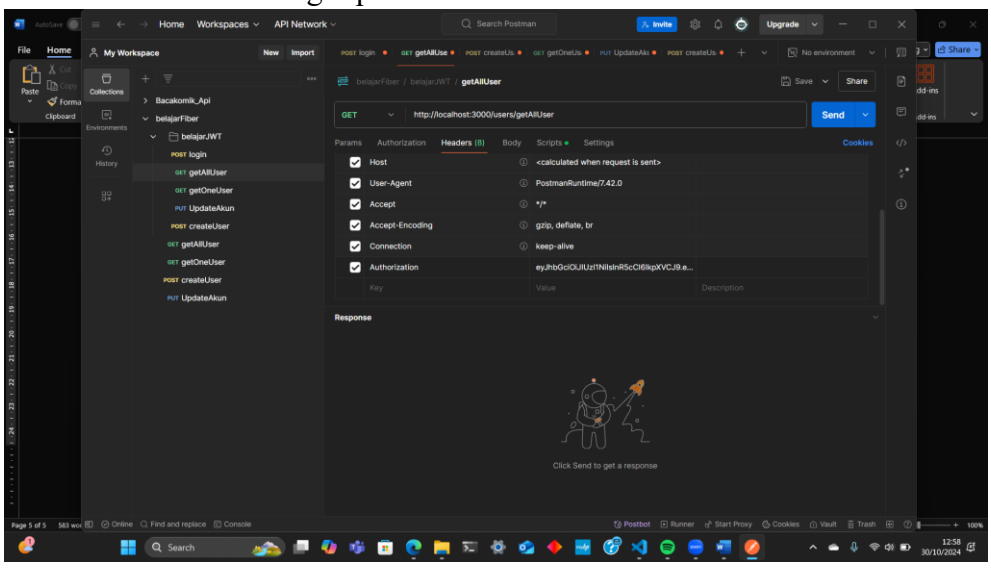


Output Hasil :

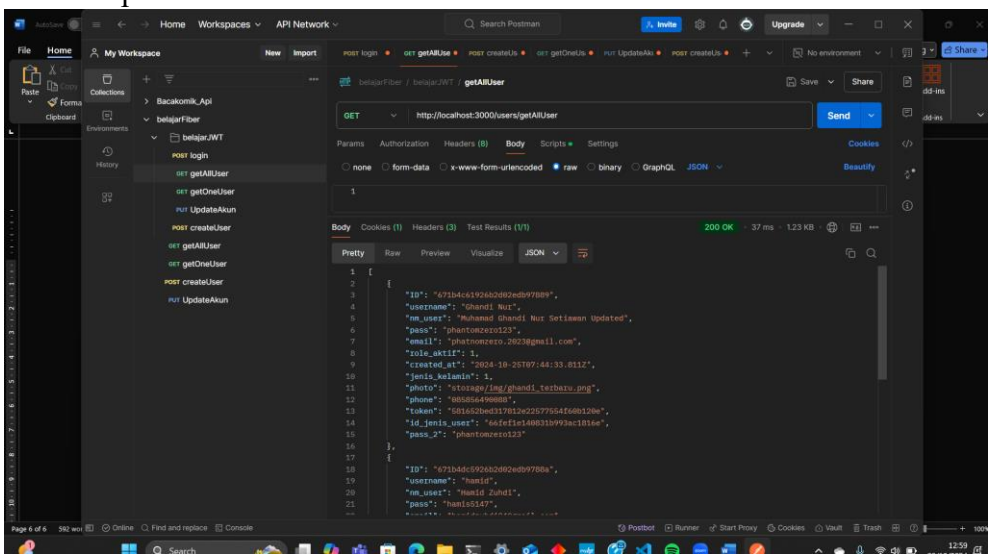


ii. Select all user

Menambahkan token login pada header

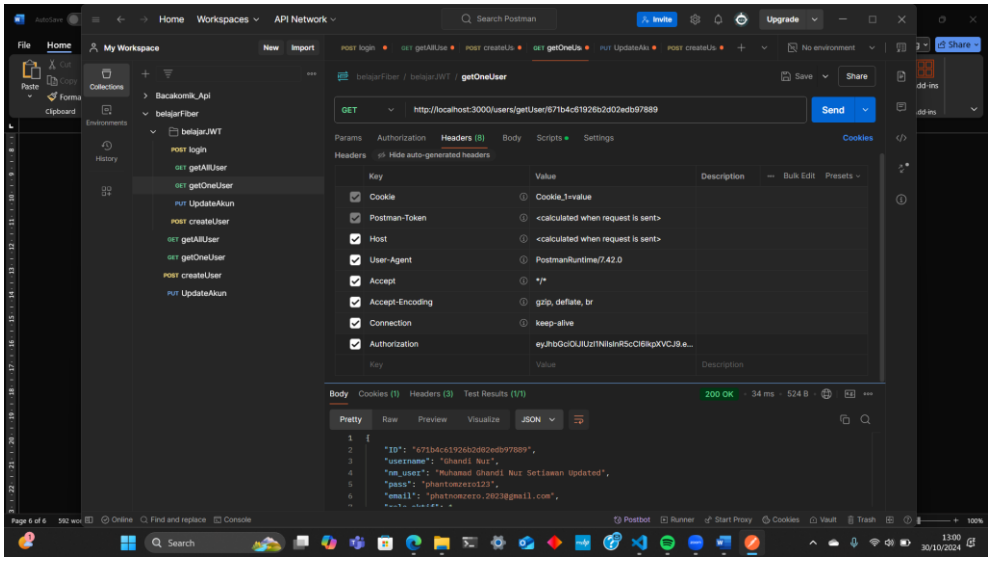


Hasil output :

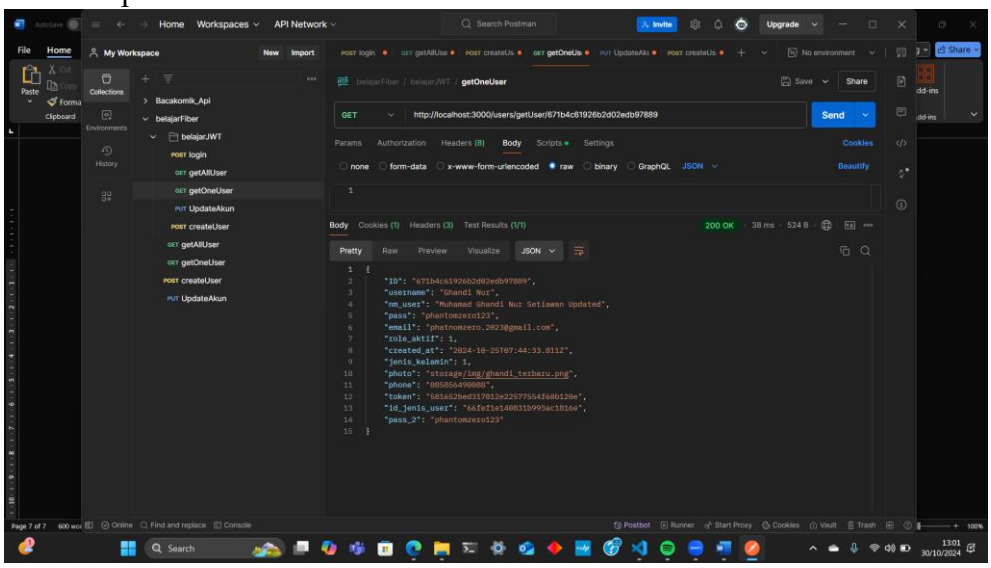


iii. Select satu user

Menambahkan token login pada header

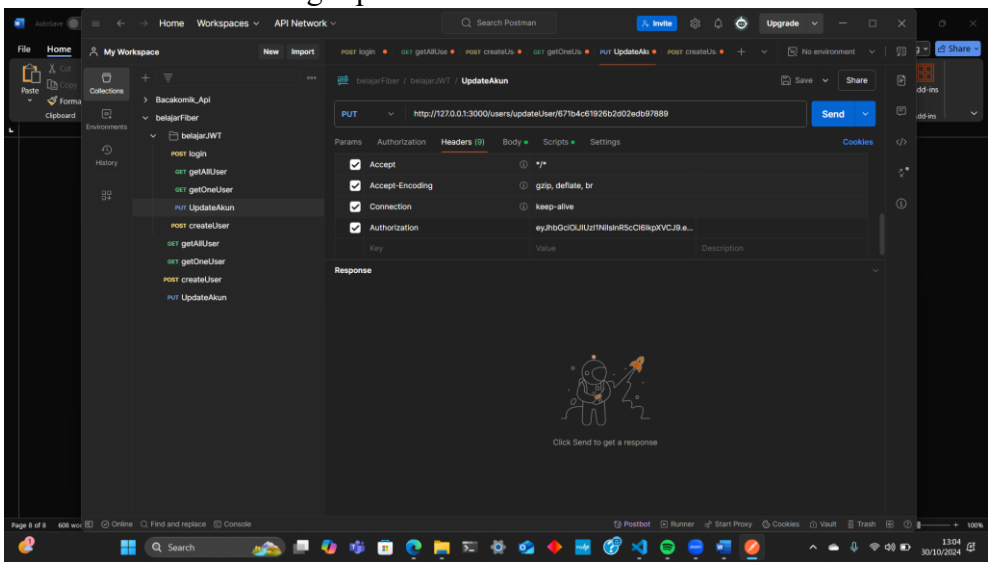


Hasil output :

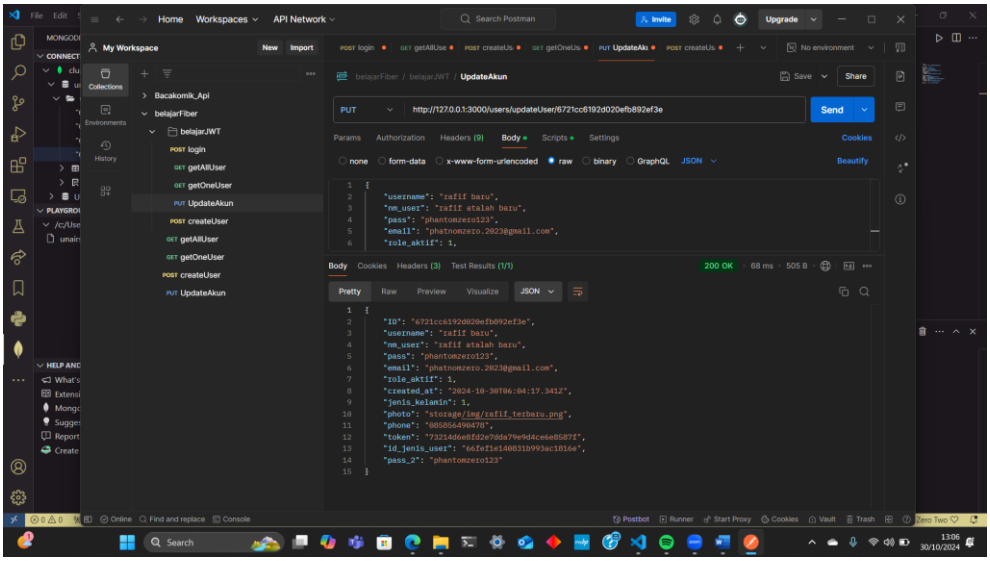


iv. Update user

Menambahkan token login pada header



Hasil output :


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
Bacakomik_Api
belajarFiber
belajarJWT
POST login
GET getAllUser
GET getOneUser
PUT UpdateAkun
POST createUser
GET getAllUser
GET getOneUser
POST createUser
PUT UpdateAkun
belajarFiber / belajarJWT / UpdateAkun
Save
Share
PUT
http://127.0.0.1:3000/users/updateUser/6721cc6192d020efb892ef3e
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "rafif baru",
"nm_user": "rafif atalah baru",
"pass": "phantomzero123",
"email": "phatnomzero.2023@gmail.com",
"role_aktif": 1,
}
Body
Cookies
Headers (3)
Test Results (1/1)
200 OK
68 ms
505 B
Pretty
Raw
Preview
Visualize
JSON
{
"ID": "6721cc6192d020efb892ef3e",
"username": "rafif baru",
"nm_user": "rafif atalah baru",
"pass": "phantomzero123",
"email": "phatnomzero.2023@gmail.com",
"role_aktif": 1,
"created_at": "2024-10-30T06:04:17.341Z",
"jenis_kelamin": 1,
"photo": "storage/img/rafif_terbaru.png",
"phone": "6858664906470",
"token": "73214d6e8fd2e7dda79e9d4ce6e0587f",
"id_jenis_user": "66fef1e140831b993ac1816e",
"pass_2": "phantomzero123"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
Search
1306
30/10/2024