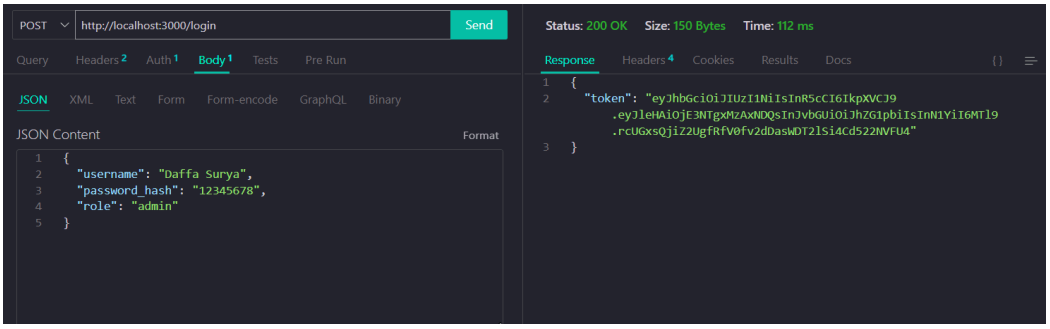
Nama: Daffa Surya Arrayan
Kelas: Pemrograman Backend Lanjutan (Praktikum)
Nim: 434231042

Aunthentikasi Sederhana menggunakan JWT dan Role Based Access Control

❖ Login sebagai (Admin)

```go
func (h *AuthHandler) Login(w http.ResponseWriter, r *http.Request) {
    var req models.User
    if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
        http.Error(w, "Invalid input", http.StatusBadRequest)
        return
    }

    user, err := h.repo.GetByUsername(req.Username)
    if err != nil {
        http.Error(w, "Invalid credentials", http.StatusUnauthorized)
        return
    }

    if bcrypt.CompareHashAndPassword([]byte(user.Password), []byte(req.Password)) != nil {
        http.Error(w, "Invalid credentials", http.StatusUnauthorized)
        return
    }

    claims := jwt.MapClaims{
        "sub":  user.ID,
        "role": user.Role,
        "exp":  time.Now().Add(time.Hour * 24).Unix(),
    }

    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    secret := os.Getenv("JWT_SECRET")
    t, _ := token.SignedString([]byte(secret))

    json.NewEncoder(w).Encode(map[string]string{"token": t})
}
```

Output:



❖ Get Data alumni (admin)

```go
func (r *alumniPostgres) FindAll() ([]models.Alumni, error) {
    rows, err := r.db.Query("SELECT id, nama, email FROM alumni")
    if err != nil {
        return nil, err
    }
    defer rows.Close()

    var list []models.Alumni
    for rows.Next() {
        var a models.Alumni
        rows.Scan(&a.ID, &a.Nama, &a.Email)
        list = append(list, a)
    }
    return list, nil
}
```

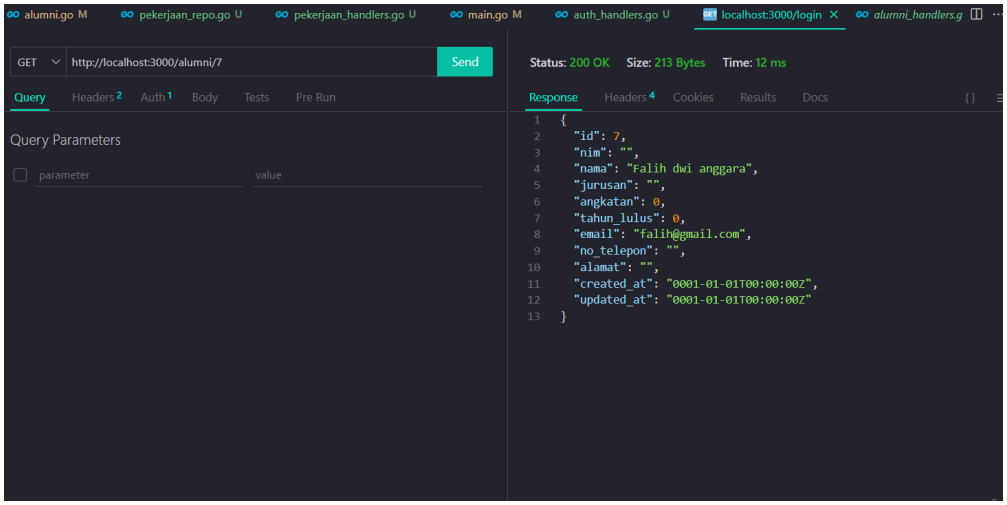
Output:


```json
[
  {
    "id": 7,
    "nim": "",
    "nama": "Falih dwi anggara",
    "jurusan": "",
    "angkatan": 0,
    "tahun_lulus": 0,
    "email": "falih@gmail.com",
    "no_telepon": "",
    "alamat": "",
    "created_at": "0001-01-01T00:00:00Z",
    "updated_at": "0001-01-01T00:00:00Z"
  },
  {
    "id": 6,
    "nim": "",
    "nama": "Diva Dwi berenza",
    "jurusan": "",
    "angkatan": 0,
    "tahun_lulus": 0,
    "email": "",
    "no_telepon": "",
```

❖ Get Data by ID (admin)

```go
func (r *alumniPostgres) FindByID(id int) (*models.Alumni, error) {
    var a models.Alumni
    err := r.db.QueryRow("SELECT id, nama, email FROM alumni WHERE id=$1", id).
        Scan(&a.ID, &a.Nama, &a.Email)
    if err != nil {
        return nil, err
    }
    return &a, nil
}
```

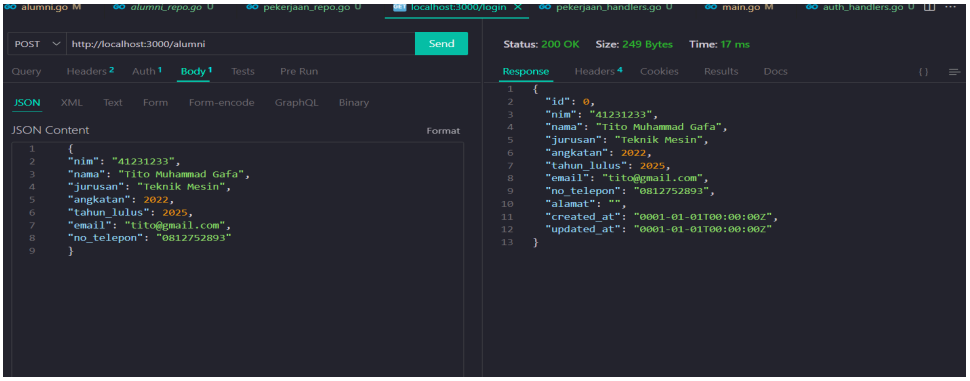
Output:



❖ Create alumni (admin)

```go
func (r *alumniPostgres) Create(a *models.Alumni) error {
    _, err := r.db.Exec("INSERT INTO alumni(nim, nama, jurusan, angkatan, tahun_lulus, email, no_telepon) VALUES($1,$2,$3,$4,$5,$6,
    return err
}
```

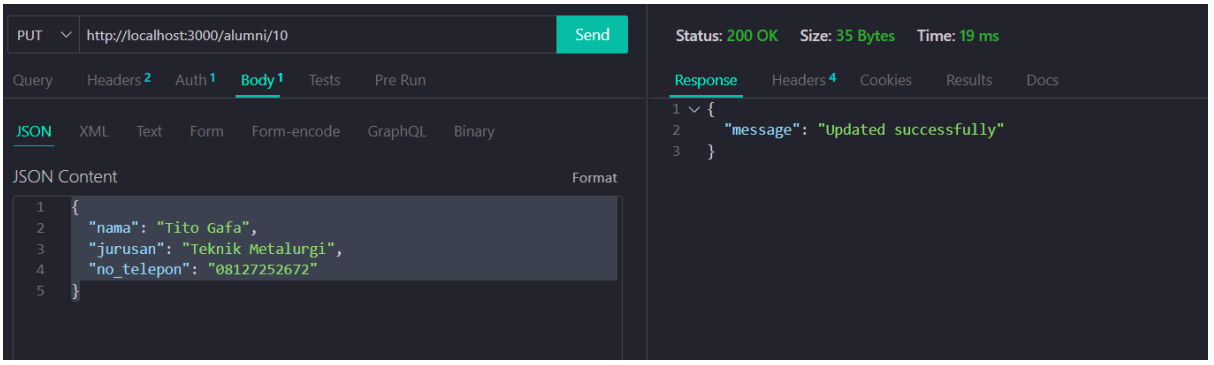
Output:



❖ Update alumni (admin)

```go
func (r *alumniPostgres) Update(id int, a *models.Alumni) error {
    _, err := r.db.Exec("UPDATE alumni SET nama=$1, jurusan=$2, no_telepon=$3 WHERE id=$4", a.Nama, a.Jurusan, a.No_telp, id)
    return err
}
```

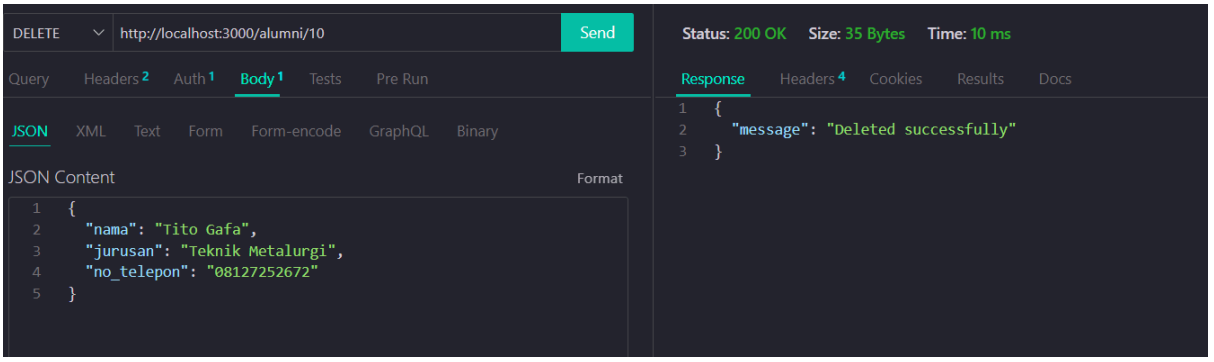
Output:



❖ Delete alumni (admin)

```go
func (r *alumniPostgres) Delete(id int) error {
    _, err := r.db.Exec("DELETE FROM alumni WHERE id=$1", id)
    return err
}
```

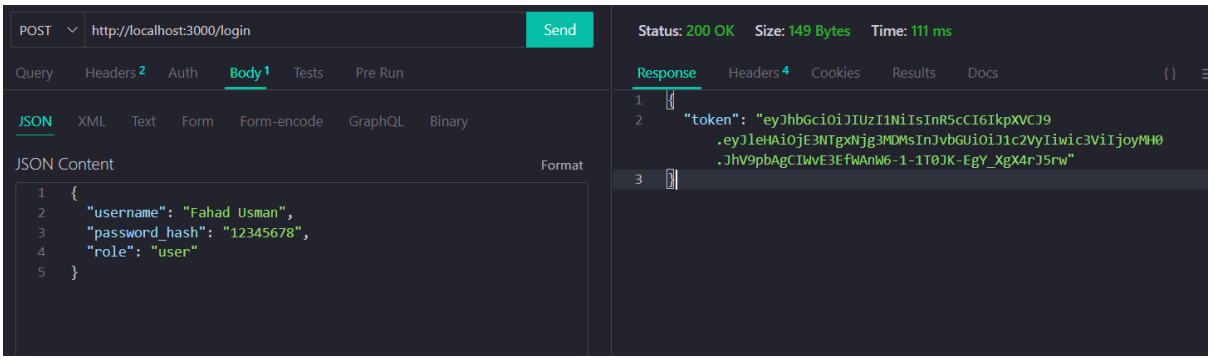
Output:



❖ Login sebagai (User)

```go
func (h *AuthHandler) Login(w http.ResponseWriter, r *http.Request) {
    var req models.User
    if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
        http.Error(w, "Invalid input", http.StatusBadRequest)
        return
    }

    user, err := h.repo.GetByUsername(req.Username)
    if err != nil {
        http.Error(w, "Invalid credentials", http.StatusUnauthorized)
        return
    }

    if bcrypt.CompareHashAndPassword([]byte(user.Password), []byte(req.Password)) != nil {
        http.Error(w, "Invalid credentials", http.StatusUnauthorized)
        return
    }

    claims := jwt.MapClaims{
        "sub":  user.ID,
        "role": user.Role,
        "exp":  time.Now().Add(time.Hour * 24).Unix(),
    }

    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    secret := os.Getenv("JWT_SECRET")
    t, _ := token.SignedString([]byte(secret))

    json.NewEncoder(w).Encode(map[string]string{"token": t})
}
```

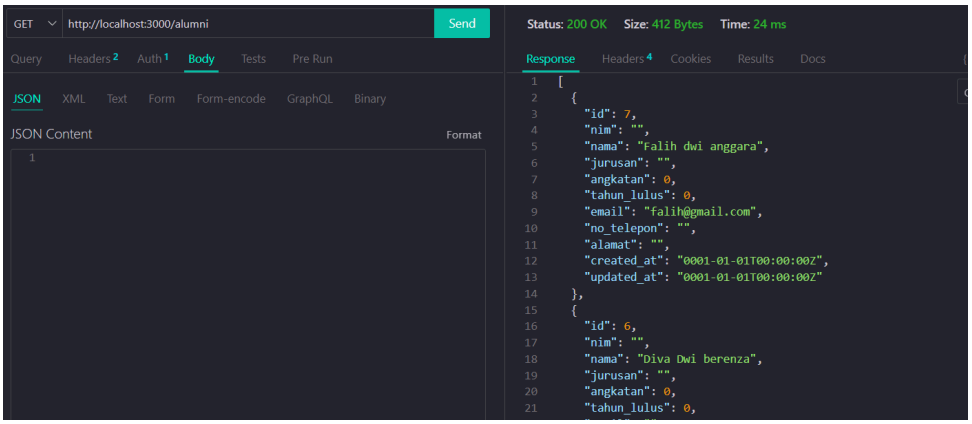
Output:



❖ Get Alumni (User)

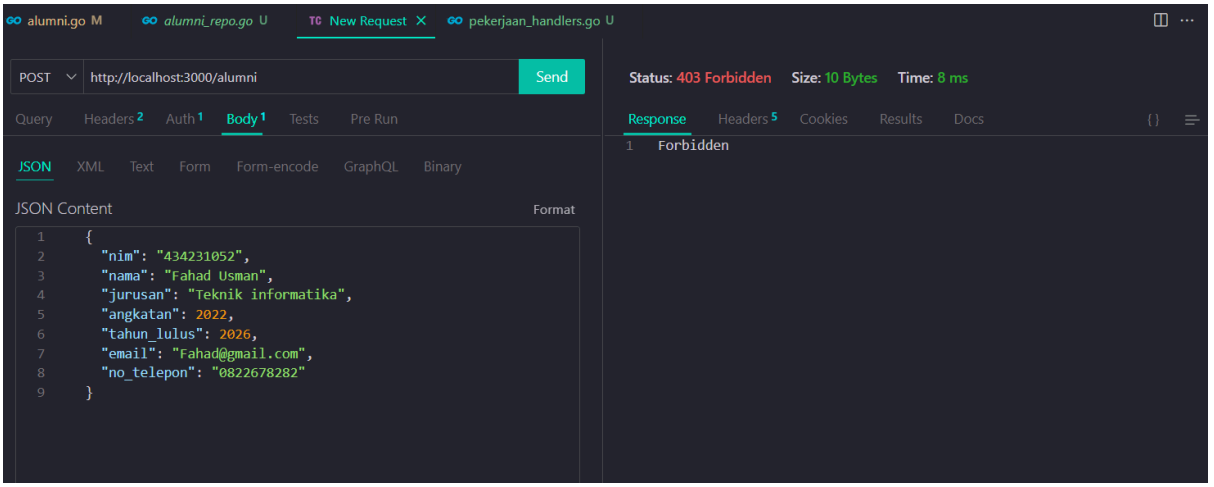
```go
func (r *alumniPostgres) FindAll() ([]models.Alumni, error) {
    rows, err := r.db.Query("SELECT id, nama, email FROM alumni")
    if err != nil {
        return nil, err
    }
    defer rows.Close()

    var list []models.Alumni
    for rows.Next() {
        var a models.Alumni
        rows.Scan(&a.ID, &a.Nama, &a.Email)
        list = append(list, a)
    }
    return list, nil
}
```

Output:



❖ Create alumni (User)

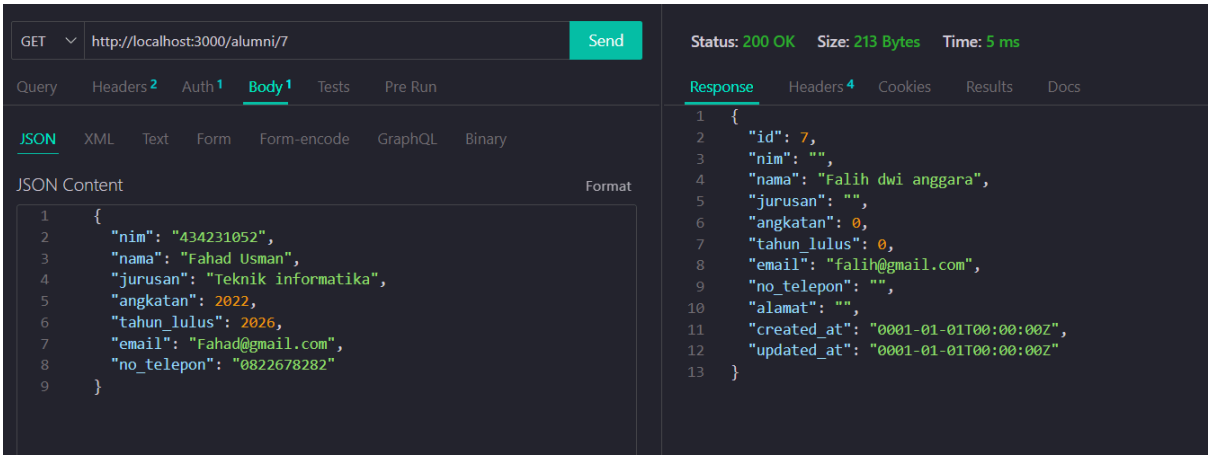
```go
func (r *alumniPostgres) Create(a *models.Alumni) error {
    _, err := r.db.Exec("INSERT INTO alumni(nim, nama, jurusan, angkatan, tahun_lulus, email, no_telepon) VALUES($1,$2,$3,$4,$5,$6,
    return err
}
```



❖ Get Data by ID (User)

```go
func (r *alumniPostgres) FindByID(id int) (*models.Alumni, error) {
    var a models.Alumni
    err := r.db.QueryRow("SELECT id, nama, email FROM alumni WHERE id=$1", id).
        Scan(&a.ID, &a.Nama, &a.Email)
    if err != nil {
        return nil, err
    }
    return &a, nil
}
```

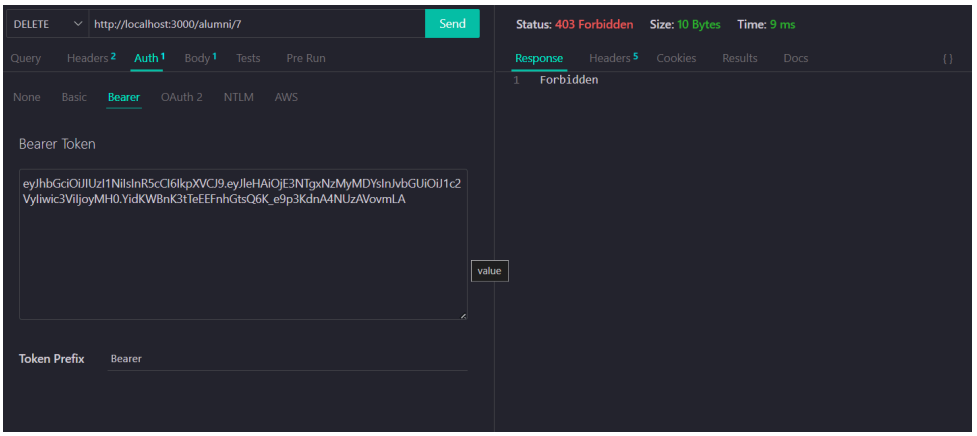
Output:



❖ Delete Data Alumni (User)

```go
func (r *alumniPostgres) Delete(id int) error {
    _, err := r.db.Exec("DELETE FROM alumni WHERE id=$1", id)
    return err
}
```

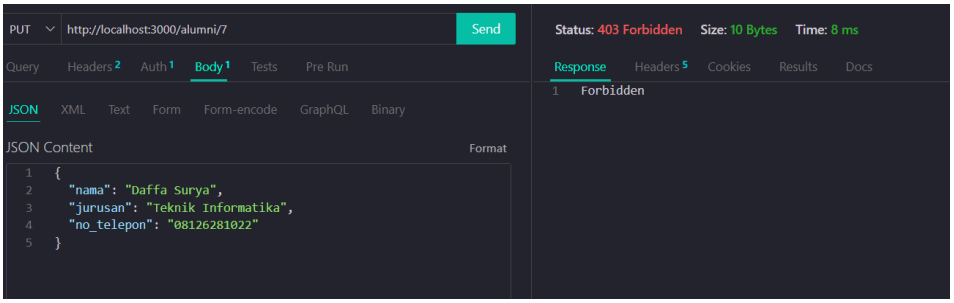
Output:



❖ Update alumni (User)

```go
func (r *alumniPostgres) Update(id int, a *models.Alumni) error {
    _, err := r.db.Exec("UPDATE alumni SET nama=$1, jurusan=$2, no_telepon=$3 WHERE id=$4", a.Nama, a.Jurusan, a.No_telp, id)
    return err
}
```

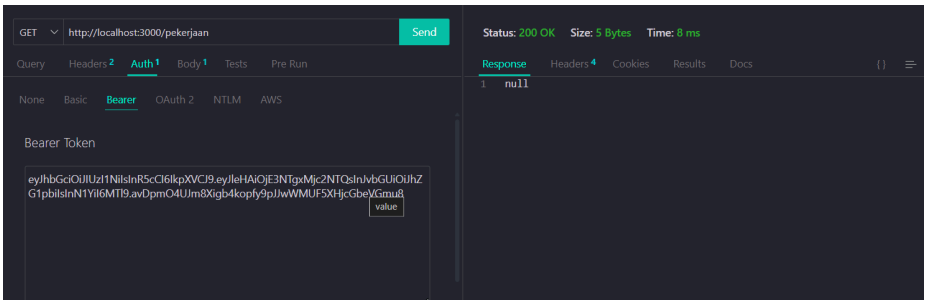
Output:



❖ Get Pekerjaan alumni (admin)

```go
func (r *pekerjaanPostgres) FindAllPekerjaan() ([]models.Pekerjaan, error) {
    rows, err := r.db.Query("SELECT id, nama_perusahaan, posisi_jabatan FROM pekerjaan_alumni")
    if err != nil {
        return nil, err
    }
    defer rows.Close()

    var list []models.Pekerjaan
    for rows.Next() {
        var a models.Pekerjaan
        rows.Scan(&a.ID, &a.Nama_Perusahaan, &a.Posisi_jabatan)
        list = append(list, a)
    }
    return list, nil
}
```

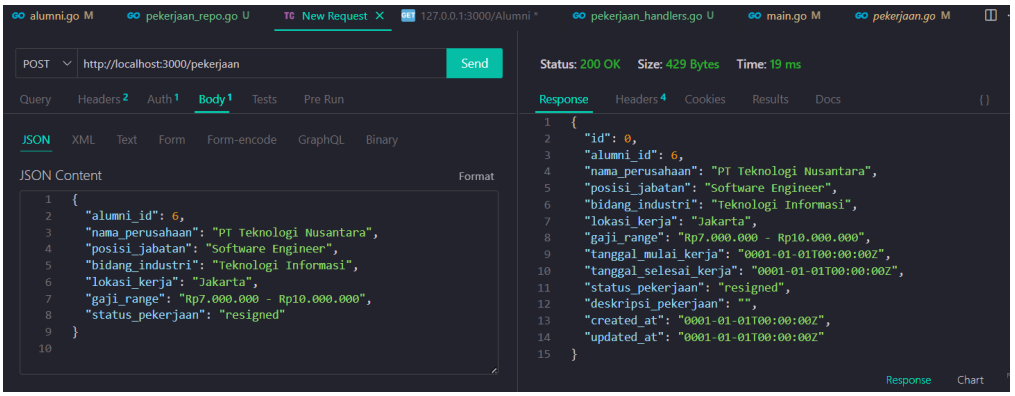
Output:



❖ Create Pekerjaan Alumni (Admin)

```go
func (r *pekerjaanPostgres) Create(p *models.Pekerjaan) error {
    _, err := r.db.Exec("INSERT INTO pekerjaan_alumni(alumni_id,na
        p.Alumni_ID, p.Nama_Perusahaan, p.Posisi_jabatan, p.Bidang
    return err
}
```

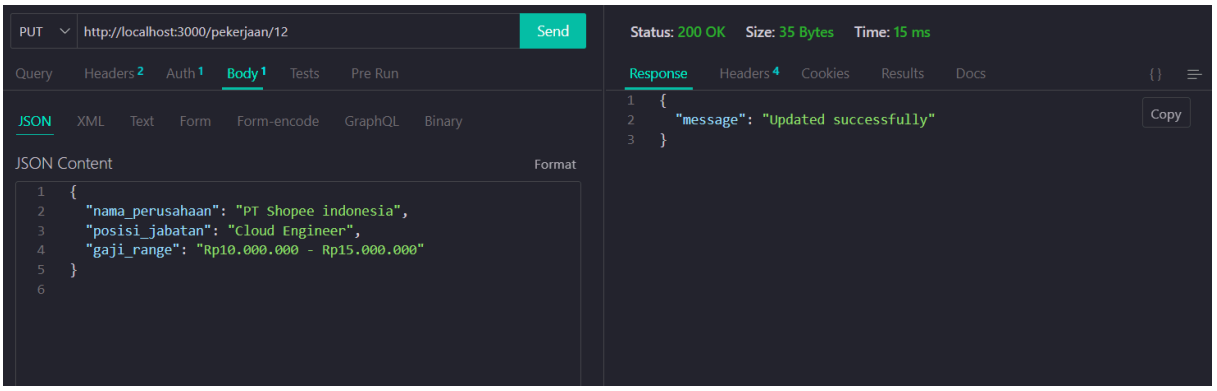
Output:



❖ Update Pekerjaan Alumni (Admin)

```go
func (r *pekerjaanPostgres) Update(id int, p *models.Pekerjaan) error {
    _, err := r.db.Exec("UPDATE pekerjaan_alumni SET nama_perusahaan=$1, pos
    return err
}
```

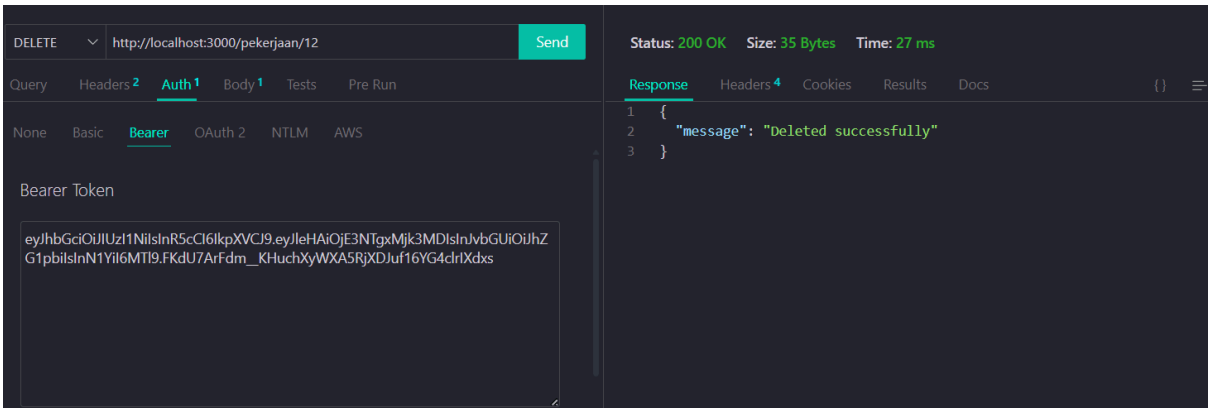
Output:



❖ Delete Data Pekerjaan alumni (Admin)

```go
func (r *pekerjaanPostgres) Delete(id int) error {
    _, err := r.db.Exec("DELETE FROM pekerjaan_alumni WHERE id=$1", id)
    return err
}
```

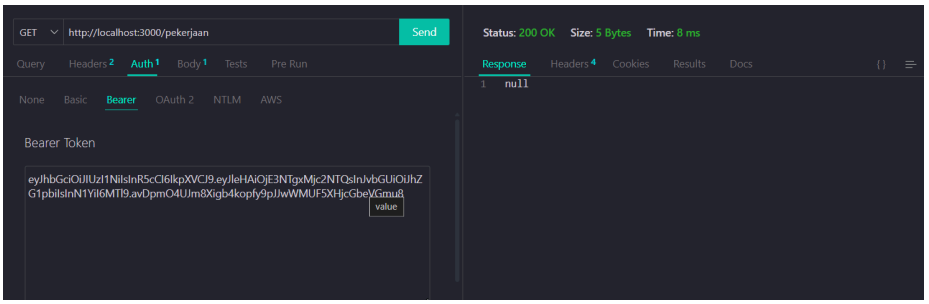
Output:



❖ Get Pekerjaan alumni (User)

```go
func (r *pekerjaanPostgres) FindAllPekerjaan() ([]models.Pekerjaan, error) {
    rows, err := r.db.Query("SELECT id, nama_perusahaan, posisi_jabatan FROM pekerjaan_alumni")
    if err != nil {
        return nil, err
    }
    defer rows.Close()

    var list []models.Pekerjaan
    for rows.Next() {
        var a models.Pekerjaan
        rows.Scan(&a.ID, &a.Nama_Perusahaan, &a.Posisi_jabatan)
        list = append(list, a)
    }
    return list, nil
}
```

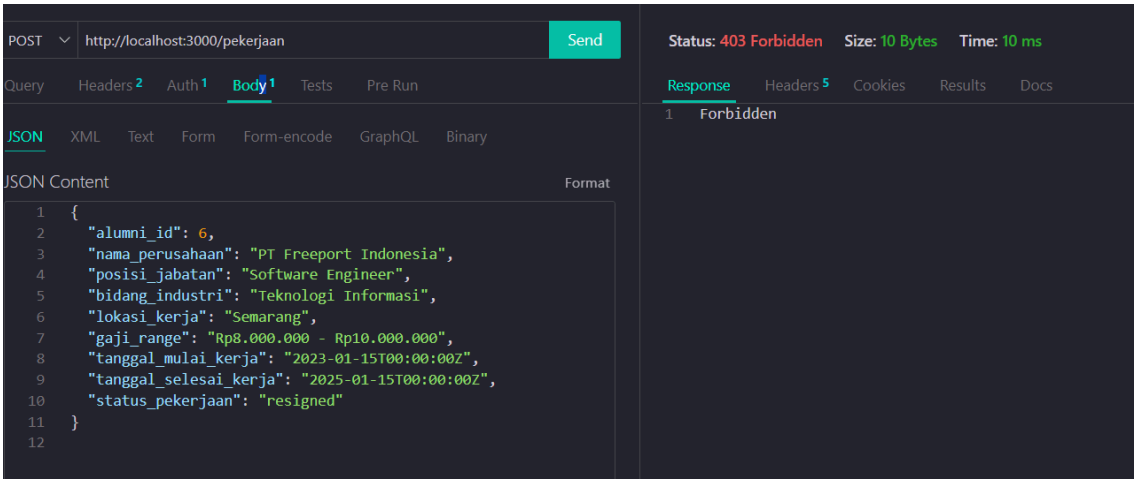
Output:



❖ Post Pekerjaan alumni (User)

```go
func (r *pekerjaanPostgres) Create(p *models.Pekerjaan) error {
    _, err := r.db.Exec("INSERT INTO pekerjaan_alumni(alumni_id,nama
        p.Alumni_ID, p.Nama_Perusahaan, p.Posisi_jabatan, p.Bidang_i
    return err
}
```

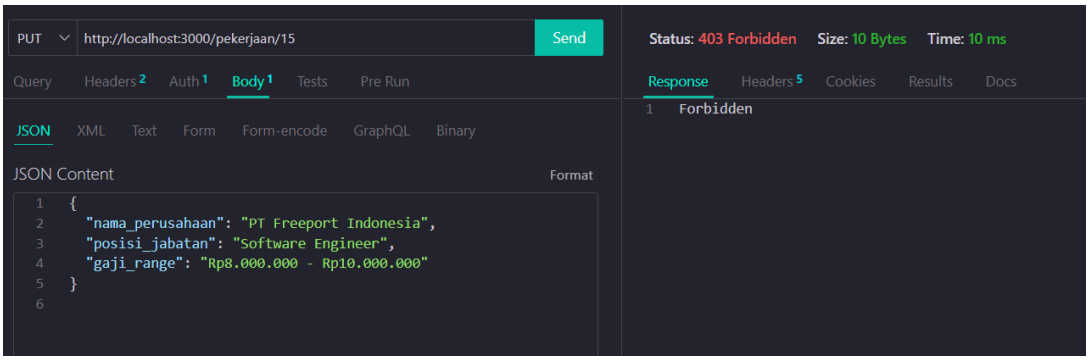
Output:



❖ Update Pekerjaan alumni (User)

```go
func (r *pekerjaanPostgres) Update(id int, p *models.Pekerjaan) error {
    _, err := r.db.Exec("UPDATE pekerjaan_alumni SET nama_perusahaan=$1, posisi_jabatan=$2
    return err
}
```

Output:



❖ Delete Pekerjaan alumni (User)

```go
func (r *pekerjaanPostgres) Delete(id int) error {
    _, err := r.db.Exec("DELETE FROM pekerjaan_alumni WHERE id=$1", id)
    return err
}
```

Output: