

Nama: Daffa Surya Arrayan

Kelas: Pemrograman Backend Lanjutan (Praktikum)

Nim: 434231042

Aunthentikasi Sederhana menggunakan JWT dan Role Based Access Control

❖ Login sebagai (Admin)

```
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:

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/login`. The request body is a JSON object containing login credentials for an admin user. The response is a 200 OK status with a JSON body containing a JWT token.

Request:

```
POST http://localhost:3000/login
{
  "username": "Daffa Surya",
  "password_hash": "12345678",
  "role": "admin"
}
```

Response:

```
200 OK
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MTg0MzAwMDQsInR5bGUiOiJhZG1pbGlzInM1YiI6MT19.rcUGxsQjZ2UgFrFv0fv2dBaMDT2LSi4CdS22WVU4"
}
```

❖ Get Data alumni (admin)

```
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:

```
[
  {
    "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)

```
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:

The screenshot shows a web browser with the URL `http://localhost:3000/alumni/7` and a `GET` method. The response status is `200 OK`, size is `213 Bytes`, and time is `12 ms`. The response body is a JSON object:

```
{
  "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"
}
```

❖ Create alumni (admin)

```
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,$7)")
    return err
}
```

Output:

The screenshot shows a web browser with the URL `http://localhost:3000/alumni` and a `POST` method. The response status is `200 OK`, size is `249 Bytes`, and time is `17 ms`. The response body is a JSON object:

```
{
  "id": 0,
  "nim": "41231233",
  "nama": "Tito Muhammad Gafa",
  "jurusan": "Teknik Mesin",
  "angkatan": 2022,
  "tahun_lulus": 2025,
  "email": "tito@gmail.com",
  "no_telepon": "0812752893",
  "alamat": "",
  "created_at": "0001-01-01T00:00:00Z",
  "updated_at": "0001-01-01T00:00:00Z"
}
```

❖ Update alumni (admin)

```
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:

PUT http://localhost:3000/alumni/10 Send

Status: 200 OK Size: 35 Bytes Time: 19 ms

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{  
  "nama": "Tito Gafa",  
  "jurusan": "Teknik Metalurgi",  
  "no_telepon": "08127252672"  
}
```

Response Headers 4 Cookies Results Docs

```
1 {  
2   "message": "Updated successfully"  
3 }
```

❖ Delete alumni (admin)

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

Output:

DELETE http://localhost:3000/alumni/10 Send

Status: 200 OK Size: 35 Bytes Time: 10 ms

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
{  
  "nama": "Tito Gafa",  
  "jurusan": "Teknik Metalurgi",  
  "no_telepon": "08127252672"  
}
```

Response Headers 4 Cookies Results Docs

```
1 {  
2   "message": "Deleted successfully"  
3 }
```

❖ Login sebagai (User)

```
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:

The screenshot displays the REST Client interface. On the left, a POST request is configured with the URL `http://localhost:3000/login`. The request body is in JSON format, containing the following data:

```
{
  "username": "Fahad Usman",
  "password_hash": "12345678",
  "role": "user"
}
```

The response status is 200 OK, with a size of 149 Bytes and a time of 111 ms. The response body is also in JSON format, containing a token:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1eHAiOiJlbnR5cCI6IkpXVCJ9.eyJ1eHAiOiJlbnR5cCI6IkpXVCJ9"
}
```

❖ Get Alumni (User)

```
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:

GET http://localhost:3000/alumni Status: 200 OK Size: 412 Bytes Time: 24 ms

Query Headers 2 Auth 1 Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

1

```
1 [
2   {
3     "id": 7,
4     "nim": "",
5     "nama": "Falih dwi anggara",
6     "jurusan": "",
7     "angkatan": 0,
8     "tahun_lulus": 0,
9     "email": "falih@gmail.com",
10    "no_telepon": "",
11    "alamat": "",
12    "created_at": "0001-01-01T00:00:00Z",
13    "updated_at": "0001-01-01T00:00:00Z"
14  },
15  {
16    "id": 6,
17    "nim": "",
18    "nama": "Diva Dwi berenza",
19    "jurusan": "",
20    "angkatan": 0,
21    "tahun_lulus": 0,
```

❖ Create alumni (User)

```
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,$7) RETURNING id")
    return err
}
```

POST http://localhost:3000/alumni Status: 403 Forbidden Size: 10 Bytes Time: 8 ms

Query Headers 2 Auth 1 Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nim": "434231052",
3   "nama": "Fahad Usman",
4   "jurusan": "Teknik informatika",
5   "angkatan": 2022,
6   "tahun_lulus": 2026,
7   "email": "Fahad@gmail.com",
8   "no_telepon": "0822678282"
9 }
```

1 Forbidden

❖ Get Data by ID (User)

```
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:

The screenshot shows a REST client interface with a GET request to `http://localhost:3000/alumni/7` and its response. The response is a JSON object containing user details for ID 7.

Request:

- Method: GET
- URL: `http://localhost:3000/alumni/7`
- Status: 200 OK
- Size: 213 Bytes
- Time: 5 ms

Response (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"  
}
```

❖ Delete Data Alumni (User)

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

Output:

DELETE http://localhost:3000/alumni/7 Send

Status: 403 Forbidden Size: 10 Bytes Time: 9 ms

Query Headers 2 **Auth 1** Body 1 Tests Pre Run

None Basic **Bearer** OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NTgxNzMyMDYsInVjbGUiOiI1c2Vylwiwicz3ViljoyMH0.YidKWBnK3tTeEFnhGtsQ6K_e9p3KdnA4NUzAVovmlA
```

Token Prefix Bearer

Response Headers 5 Cookies Results Docs {}

1 Forbidden

❖ Update alumni (User)

```
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:

PUT http://localhost:3000/alumni/7 Send

Status: 403 Forbidden Size: 10 Bytes Time: 8 ms

Query Headers 2 Auth 1 **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "nama": "Daffa Surya",  
3   "jurusan": "Teknik Informatika",  
4   "no_telepon": "08126281022"  
5 }
```

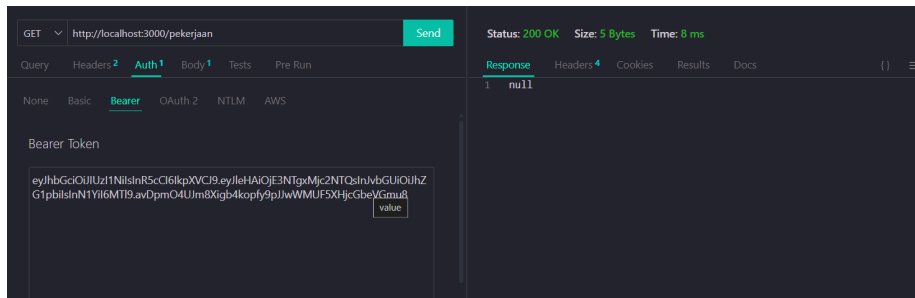
Response Headers 5 Cookies Results Docs {}

1 Forbidden

❖ Get Pekerjaan alumni (admin)

```
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  
}
```

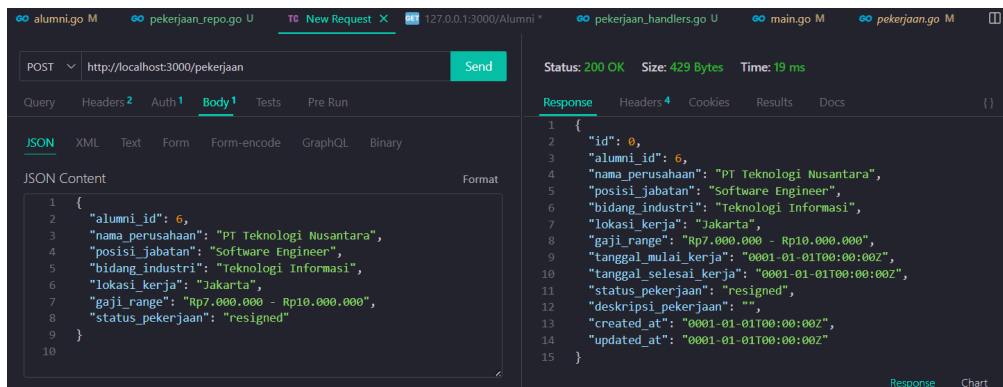

Output:



❖ Create Pekerjaan Alumni (Admin)

```
func (r *pekerjaanPostgres) Create(p *models.Pekerjaan) error {
    _, err := r.db.Exec("INSERT INTO pekerjaan_alumni(alumni_id,nama_perusahaan, posisi_jabatan, bidang_industri, lokasi_kerja, gaji_range, tanggal_mulai_kerja, tanggal_selesai_kerja, status_pekerjaan, deskripsi_pekerjaan, created_at, updated_at) VALUES ($1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12)",
        p.Alumni_ID, p>Nama_Perusahaan, p.Posisi_jabatan, p.Bidang_industri, p.Lokasi_kerja, p.Gaji_range, p.Tanggal_mulai_kerja, p.Tanggal_selesai_kerja, p.Status_pekerjaan, p.Deskripsi_pekerjaan, p.Created_at, p.Updated_at)
    return err
}
```

Output:



❖ Update Pekerjaan Alumni (Admin)

```
func (r *pekerjaanPostgres) Update(id int, p *models.Pekerjaan) error {
    _, err := r.db.Exec("UPDATE pekerjaan_alumni SET nama_perusahaan=$1, posisi_jabatan=$2, bidang_industri=$3, lokasi_kerja=$4, gaji_range=$5, tanggal_mulai_kerja=$6, tanggal_selesai_kerja=$7, status_pekerjaan=$8, deskripsi_pekerjaan=$9, created_at=$10, updated_at=$11 WHERE alumni_id=$12",
        p>Nama_Perusahaan, p.Posisi_jabatan, p.Bidang_industri, p.Lokasi_kerja, p.Gaji_range, p.Tanggal_mulai_kerja, p.Tanggal_selesai_kerja, p.Status_pekerjaan, p.Deskripsi_pekerjaan, p.Created_at, p.Updated_at, p.Alumni_ID)
    return err
}
```

Output:

PUT ▼ http://localhost:3000/pekerjaan/12 Send

Status: 200 OK Size: 35 Bytes Time: 15 ms

Query Headers **2** Auth **1** **Body **1**** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nama_perusahaan": "PT Shopee indonesia",
3   "posisi_jabatan": "Cloud Engineer",
4   "gaji_range": "Rp10.000.000 - Rp15.000.000"
5 }
6
```

Response Headers **4** Cookies Results Docs {} ≡

```
1 {
2   "message": "Updated successfully"
3 }
```

Copy

❖ Delete Data Pekerjaan alumni (Admin)

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

Output:

DELETE ▼ http://localhost:3000/pekerjaan/12 Send

Status: 200 OK Size: 35 Bytes Time: 27 ms

Query Headers **2** **Auth **1**** Body **1** Tests Pre Run

None Basic **Bearer** OAuth 2 NTLM AWS

Bearer Token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiJlE3NTgxMjM0IiwiaXNjbGUiOiJhZG1pbGlzInN1YiI6MTI5LjFKdU7ArFdm_KHuchXyWXA5RjXDJuf16YG4cltXdXs
```

Response Headers **4** Cookies Results Docs {} ≡

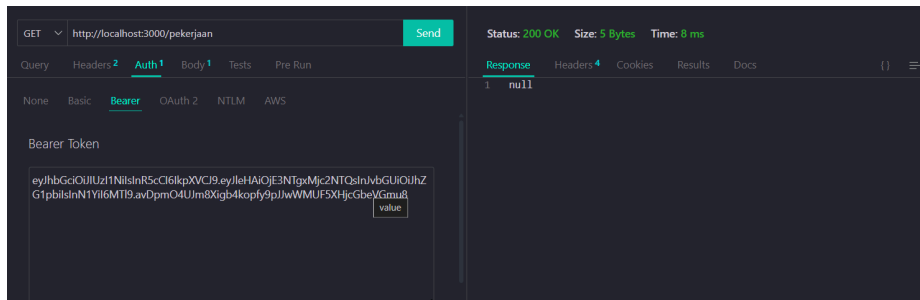
```
1 {
2   "message": "Deleted successfully"
3 }
```

❖ Get Pekerjaan alumni (User)

```
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
}
```

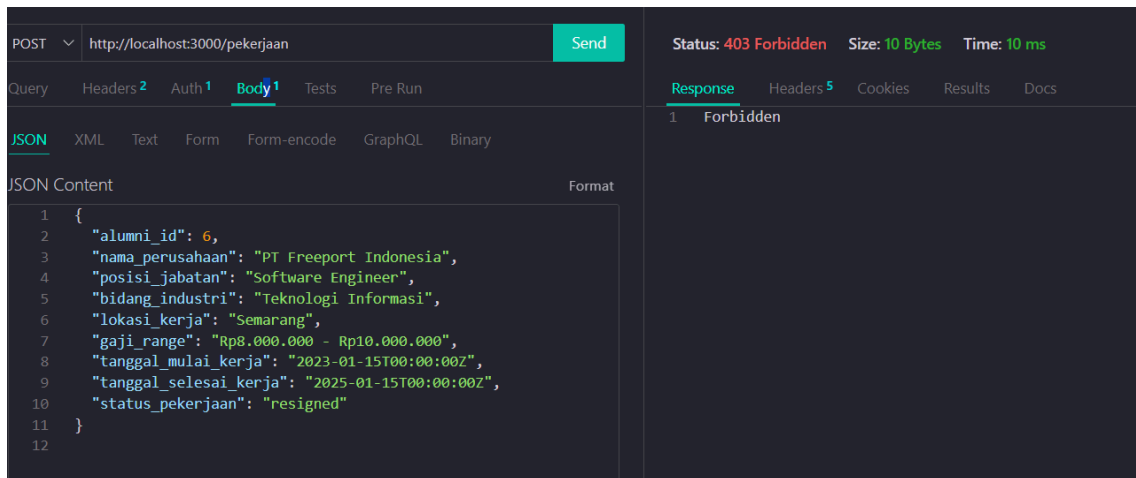
Output:



❖ Post Pekerjaan alumni (User)

```
func (r *pekerjaanPostgres) Create(p *models.Pekerjaan) error {
    _, err := r.db.Exec("INSERT INTO pekerjaan_alumni(alumni_id,nama_perusahaan, posisi_jabatan, bidang_industri, lokasi_kerja, gaji_range, tanggal_mulai_kerja, tanggal_selesai_kerja, status_pekerjaan) VALUES ($1,$2,$3,$4,$5,$6,$7,$8,$9)",
        p.Alumni_ID, p>Nama_Perusahaan, p.Posisi_jabatan, p.Bidang_industri, p.Lokasi_kerja, p.Gaji_range, p.Tanggal_mulai_kerja, p.Tanggal_selesai_kerja, p.Status_pekerjaan)
    return err
}
```

Output:



❖ Update Pekerjaan alumni (User)

```
func (r *pekerjaanPostgres) Update(id int, p *models.Pekerjaan) error {
    _, err := r.db.Exec("UPDATE pekerjaan_alumni SET nama_perusahaan=$1, posisi_jabatan=$2, bidang_industri=$3, lokasi_kerja=$4, gaji_range=$5, tanggal_mulai_kerja=$6, tanggal_selesai_kerja=$7, status_pekerjaan=$8 WHERE alumni_id=$9",
        p>Nama_Perusahaan, p.Posisi_jabatan, p.Bidang_industri, p.Lokasi_kerja, p.Gaji_range, p.Tanggal_mulai_kerja, p.Tanggal_selesai_kerja, p.Status_pekerjaan, p.Alumni_ID)
    return err
}
```

Output:

PUT ⌵ http://localhost:3000/pekerjaan/15 Send

Status: 403 Forbidden Size: 10 Bytes Time: 10 ms

Query Headers ² Auth ¹ Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nama_perusahaan": "PT Freeport Indonesia",
3   "posisi_jabatan": "Software Engineer",
4   "gaji_range": "Rp8.000.000 - Rp10.000.000"
5 }
6
```

Response Headers ⁵ Cookies Results Docs

1 Forbidden

❖ Delete Pekerjaan alumni (User)

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

Output:

DELETE ⌵ http://localhost:3000/pekerjaan/15 Send

Status: 403 Forbidden Size: 10 Bytes Time: 7 ms

Query Headers ² Auth ¹ Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Response Headers ⁵ Cookies Results Docs

1 Forbidden