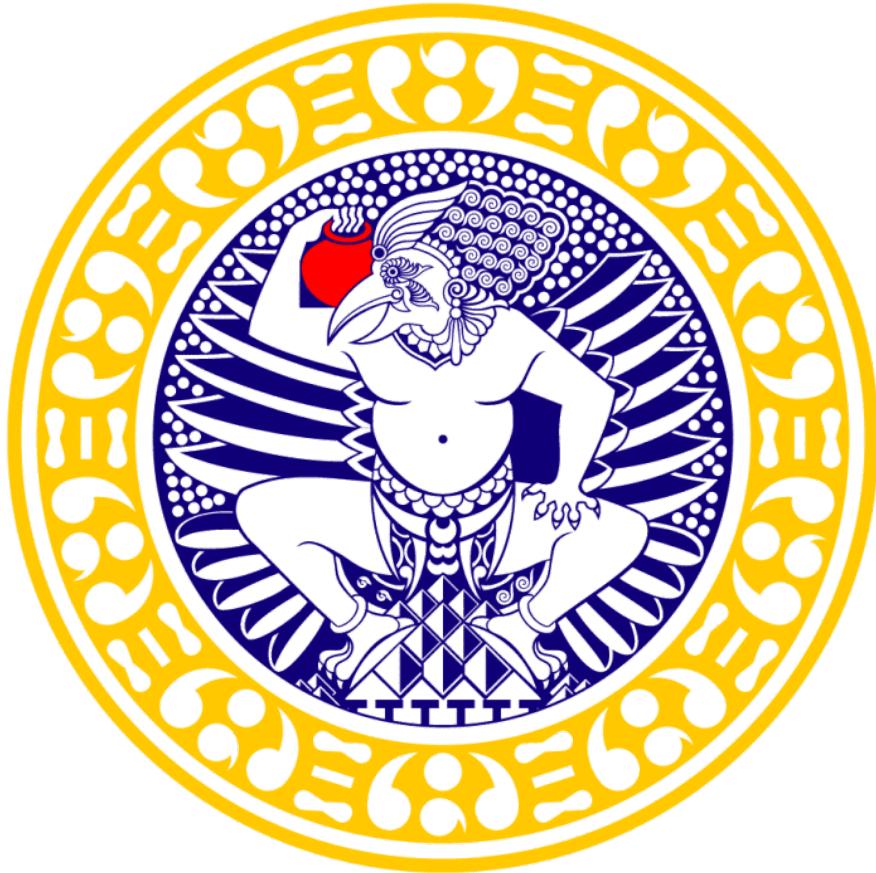


LAPORAN UTS PEMROGRAMAN BACKEND LANJUT PRAKTIKUM

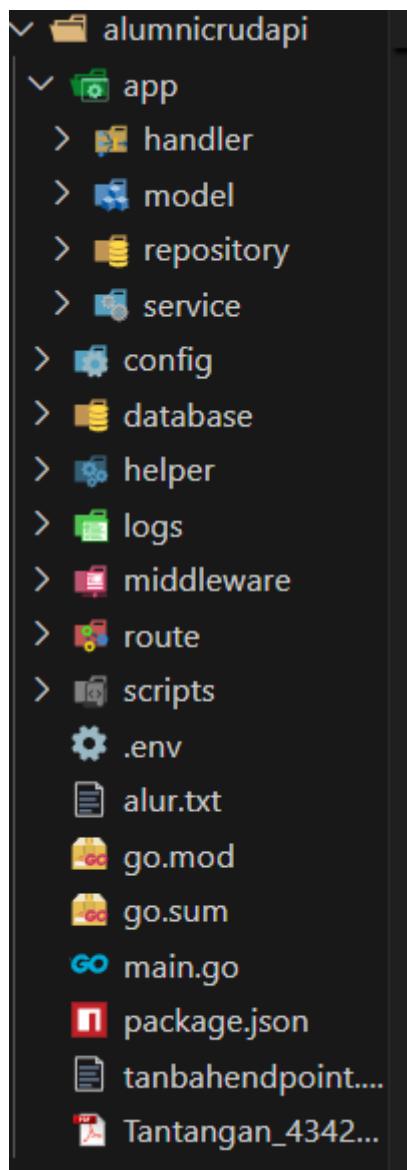


Disusun Oleh:

Pedja Rafsanjani (434231099)

**FAKULTAS VOKASI
UNIVERSITAS AIRLANGGA
2025**

STRUKTUR



Handler/pekerjaan_handler.go

```
func (h *PekerjaanHandler) ListTrash(c *fiber.Ctx) error {
    username := c.Locals("username").(string)
    role := c.Locals("role").(string)
    userID := c.Locals("user_id").(int)

    page, _ := strconv.Atoi(c.Query("page", "1"))
    limit, _ := strconv.Atoi(c.Query("limit", "10"))
    search := c.Query("search", "")

    log.Printf("User %s (%s) accessing GET /pekerjaan/trash
page=%d limit=%d", username, role, page, limit)
```

```
        list, err := h.pekerjaanService.ListTrash(search, page,
limit, userID, role)
        if err != nil {
            return helper.ErrorResponse(c,
fiber.StatusInternalServerError, "Failed to get trash data")
        }

        return c.JSON(fiber.Map{
            "success": true,
            "message": "Trash data retrieved successfully",
            "data":     list,
        })
    }

func (h *PekerjaanHandler) RestorePekerjaan(c *fiber.Ctx) error {
    username := c.Locals("username").(string)
    role := c.Locals("role").(string)
    userID := c.Locals("user_id").(int)
    id, err := strconv.Atoi(c.Params("id"))
    if err != nil {
        return helper.ErrorResponse(c, fiber.StatusBadRequest,
"Invalid ID parameter")
    }

    log.Printf("User %s (%s) restoring pekerjaan ID %d",
username, role, id)

    if err := h.pekerjaanService.RestorePekerjaan(id, userID,
role); err != nil {
        switch err.Error() {
        case "pekerjaan not found":
            return helper.ErrorResponse(c, fiber.StatusNotFound,
"Pekerjaan not found in trash")
        case "access denied: you can only restore your own
pekerjaan":
            return helper.ErrorResponse(c, fiber.StatusForbidden,
"Access denied: you can only restore your own pekerjaan")
        default:
            return helper.ErrorResponse(c,
fiber.StatusInternalServerError, "Failed to restore pekerjaan")
        }
    }
}
```

```

        return helper.SuccessResponse(c, "Pekerjaan restored
successfully", nil)
    }

func (h *PekerjaanHandler) HardDeletePekerjaan(c *fiber.Ctx)
error {
    username := c.Locals("username").(string)
    role := c.Locals("role").(string)
    userID := c.Locals("user_id").(int)
    id, err := strconv.Atoi(c.Params("id"))
    if err != nil {
        return helper.ErrorResponse(c, fiber.StatusBadRequest,
"Invalid ID parameter")
    }

    log.Printf("User %s (%s) hard-deleting pekerjaan ID %d",
username, role, id)

    if err := h.pekerjaanService.HardDeletePekerjaan(id, userID,
role); err != nil {
        if err == sql.ErrNoRows {
            // not found or not in trash/unauthorized
            return helper.ErrorResponse(c, fiber.StatusNotFound,
"Pekerjaan not found in trash or access denied")
        }
        return helper.ErrorResponse(c,
fiber.StatusInternalServerError, "Failed to hard delete
pekerjaan")
    }

    return helper.SuccessResponse(c, "Pekerjaan permanently
deleted", nil)
}

```

Repository/pekerjaan_repository.go

```
1 func (r *pekerjaanRepository) ListTrashAdmin(search string, limit, offset int) ([]model.PekerjaanAlumni, error) {
2     q := ` 
3         SELECT id, alumni_id, nama_perusahaan, posisi_jabatan, bidang_industri, lokasi_kerja,
4             gaji_range, tanggal_mulai_kerja, tanggal_selesai_kerja, status_pekerjaan,
5             deskripsi_pekerjaan, is_deleted, deleted_at, deleted_by, created_at, updated_at
6         FROM pekerjaan_alumni
7         WHERE is_deleted = TRUE
8             AND ($1 = '' OR nama_perusahaan ILIKE '%'||$1|| '%' OR posisi_jabatan ILIKE '%'||$1|| '%' OR bidang_industri ILIKE '%'||$1|| '%' OR lokasi_kerja ILIKE '%'||$1|| '%')
9         ORDER BY deleted_at DESC NULLS LAST
10        LIMIT $2 OFFSET $3
11    `
12    rows, err := database.DB.Query(q, search, limit, offset)
13    if err != nil {
14        return nil, err
15    }
16    defer rows.Close()
17
18    var out []model.PekerjaanAlumni
19    for rows.Next() {
20        var p model.PekerjaanAlumni
21        if err := rows.Scan(
22            &p.ID, &p.AlumniID, &p.NamaPerusahaan, &p.PosisiJabatan, &p.BidangIndustri, &p.LokasiKerja,
23            &p.GajiRange, &p.TanggalMulaiKerja, &p.TanggalSelesaiKerja, &p.StatusPekerjaan,
24            &p.DeskripsiPekerjaan, &p.IsDeleted, &p.DeletedAt, &p.DeletedBy, &p.CreatedAt, &p.UpdatedAt,
25        ); err != nil {
26            return nil, err
27        }
28        out = append(out, p)
29    }
30    return out, nil
31 }
32
33 func (r *pekerjaanRepository) ListTrashUser(userID int, search string, limit, offset int) ([]model.PekerjaanAlumni, error) {
34     q := ` 
35         SELECT pa.id, pa.alumni_id, pa.nama_perusahaan, pa.posisi_jabatan, pa.bidang_industri, pa.lokasi_kerja,
36             pa.gaji_range, pa.tanggal_mulai_kerja, pa.tanggal_selesai_kerja, pa.status_pekerjaan,
37             pa.deskripsi_pekerjaan, pa.is_deleted, pa.deleted_at, pa.deleted_by, pa.created_at, pa.updated_at
38         FROM pekerjaan_alumni pa
39         JOIN alumni a ON a.id = pa.alumni_id
40         WHERE pa.is_deleted = TRUE
41             AND a.user_id = $1
42             AND ($2 = '' OR pa.nama_perusahaan ILIKE '%'||$2|| '%' OR pa.posisi_jabatan ILIKE '%'||$2|| '%' OR pa.bidang_industri ILIKE '%'||$2|| '%' OR pa.lokasi_kerja ILIKE '%'||$2|| '%')
43         ORDER BY pa.deleted_at DESC NULLS LAST
44        LIMIT $3 OFFSET $4
45    `
46    rows, err := database.DB.Query(q, userID, search, limit, offset)
47    if err != nil {
48        return nil, err
49    }
50    defer rows.Close()
51
52    var out []model.PekerjaanAlumni
53    for rows.Next() {
54        var p model.PekerjaanAlumni
55        if err := rows.Scan(
56            &p.ID, &p.AlumniID, &p.NamaPerusahaan, &p.PosisiJabatan, &p.BidangIndustri, &p.LokasiKerja,
57            &p.GajiRange, &p.TanggalMulaiKerja, &p.TanggalSelesaiKerja, &p.StatusPekerjaan,
58            &p.DeskripsiPekerjaan, &p.IsDeleted, &p.DeletedAt, &p.DeletedBy, &p.CreatedAt, &p.UpdatedAt,
59        ); err != nil {
60            return nil, err
61        }
62        out = append(out, p)
63    }
64    return out, nil
65 }
66
67 func (r *pekerjaanRepository) Restore(id int) error {
68     q := ` 
69         UPDATE pekerjaan_alumni
70             SET is_deleted = FALSE, deleted_at = NULL, deleted_by = NULL, updated_at = NOW()
71         WHERE id = $1 AND is_deleted = TRUE
72    `
73    res, err := database.DB.Exec(q, id)
74    if err != nil {
75        return err
76    }
77    n, _ := res.RowsAffected()
78    if n == 0 {
79        return sql.ErrNoRows
80    }
81    return nil
82 }
83
84 func (r *pekerjaanRepository) HardDeleteAdmin(id int) error {
85     q := `DELETE FROM pekerjaan_alumni WHERE id = $1 AND is_deleted = TRUE`
86     res, err := database.DB.Exec(q, id)
87     if err != nil {
88         return err
89     }
90     n, _ := res.RowsAffected()
91     if n == 0 {
92         return sql.ErrNoRows
93     }
94     return nil
95 }
96
97 func (r *pekerjaanRepository) HardDeleteUser(id int, userID int) error {
98     q := ` 
99         DELETE FROM pekerjaan_alumni pa
100         USING alumni a
101         WHERE pa.id = $1
102             AND pa.is_deleted = TRUE
103             AND a.id = pa.alumni_id
104             AND a.user_id = $2
105    `
106    res, err := database.DB.Exec(q, id, userID)
107    if err != nil {
108        return err
109    }
110    n, _ := res.RowsAffected()
111    if n == 0 {
112        return sql.ErrNoRows
113    }
114    return nil
115 }
116 }
```

Service/Pekerjaan_service.go

```
func (s *pekerjaanService) ListTrash(search string, page, limit
int, userID int, role string) ([]model.PekerjaanAlumni, error) {
    if page < 1 {
        page = 1
    }
    if limit < 1 || limit > 100 {
        limit = 10
    }
    offset := (page - 1) * limit

    if role == "admin" {
        return s.pekerjaanRepo.ListTrashAdmin(search, limit,
offset)
    }
    return s.pekerjaanRepo.ListTrashUser(userID, search, limit,
offset)
}

func (s *pekerjaanService) RestorePekerjaan(id int, userID int,
role string) error {
    // check ownership for non-admin
    if role != "admin" {
        ownerID, err := s.pekerjaanRepo.GetOwnerUserID(id)
        if err != nil {
            if err == sql.ErrNoRows {
                return errors.New("pekerjaan not found")
            }
            return err
        }
        if ownerID != userID {
            return errors.New("access denied: you can only
restore your own pekerjaan")
        }
    }
    return s.pekerjaanRepo.Restore(id)
}

func (s *pekerjaanService) HardDeletePekerjaan(id int, userID
int, role string) error {
    if role == "admin" {
        return s.pekerjaanRepo.HardDeleteAdmin(id)
    }
}
```

```

    // user only own
    return s.pekerjaanRepo.HardDeleteUser(id, userID)
}

```

Route.go

```

package route

import (
    "alumni-crud-api/app/handler"
    "alumni-crud-api/middleware"

    "github.com/gofiber/fiber/v2"
)

func SetupRoutes(fiberApp *fiber.App, alumniHandler
*handler.AlumniHandler, pekerjaanHandler
*handler.PekerjaanHandler, authHandler *handler.AuthHandler) {
    fiberApp.Get("/", func(c *fiber.Ctx) error {
        return c.JSON(fiber.Map{
            "success": true,
            "message": "Alumni CRUD API Server",
            "version": "1.0.0",
            "endpoints": fiber.Map{
                "auth": fiber.Map{
                    "POST /alumni-crud-api/auth/login": "Login
to get access token",
                    "GET /alumni-crud-api/auth/profile": "Get
user profile (requires auth)",
                },
                "alumni": fiber.Map{
                    "GET /alumni-crud-api/alumni": "Get
all alumni (requires auth)",
                    "GET /alumni-crud-api/alumni/:id": "Get
alumni by ID (requires auth)",
                    "POST /alumni-crud-api/alumni": "Create
new alumni (admin only)",
                    "PUT /alumni-crud-api/alumni/:id": "Update
alumni (admin only)",
                    "DELETE /alumni-crud-api/alumni/:id": "Delete
alumni (admin only)",
                },
                "pekerjaan": fiber.Map{

```

```

        "GET /alumni-crud-api/pekerjaan":  

    "Get all pekerjaan (requires auth)",  

        "GET /alumni-crud-api/pekerjaan/:id":  

    "Get pekerjaan by ID (requires auth)",  

        "GET  

/alumni-crud-api/pekerjaan/alumni/:alumni_id": "Get pekerjaan by  

alumni ID (admin only)",  

        "POST /alumni-crud-api/pekerjaan":  

    "Create new pekerjaan (admin only)",  

        "PUT /alumni-crud-api/pekerjaan/:id":  

    "Update pekerjaan (admin only)",  

        "DELETE /alumni-crud-api/pekerjaan/:id":  

    "Delete pekerjaan (admin only)",  

        "PATCH  

/alumni-crud-api/pekerjaan/:id/soft-delete": "Soft delete  

pekerjaan (admin or owner)",  

        "GET /alumni-crud-api/pekerjaan/trash":  

    "List trash (admin: all, user: own)",  

        "PATCH  

/alumni-crud-api/pekerjaan/:id/restore": "Restore from trash  

(admin or owner)",  

        "DELETE  

/alumni-crud-api/pekerjaan/:id/hard-delete": "Hard delete from  

trash (admin or owner)",  

    },  

},  

})  

})  

fiberApp.Get("/health", func(c *fiber.Ctx) error {  

    return c.JSON(fiber.Map{  

        "success": true,  

        "status": "healthy",  

        "message": "Server is running properly",  

    })  

})  

api := fiberApp.Group("/alumni-crud-api")  

auth := api.Group("/auth")
auth.Post("/login", authHandler.Login)  

protected := api.Group("", middleware.AuthRequired())

```

```
protected.Get("/auth/profile", authHandler.GetProfile)

alumni := protected.Group("/alumni")
alumni.Get("/", alumniHandler.GetAllAlumni)
// Admin and User can access
alumni.Get("/:id", alumniHandler.GetAlumniByID)
// Admin and User can access
alumni.Post("/", middleware.AdminOnly(),
alumniHandler.CreateAlumni)      // Admin only
alumni.Put("/:id", middleware.AdminOnly(),
alumniHandler.UpdateAlumni)      // Admin only
alumni.Delete("/:id", middleware.AdminOnly(),
alumniHandler.DeleteAlumni) // Admin only

pekerjaan := protected.Group("/pekerjaan")
pekerjaan.Get("/", pekerjaanHandler.GetAllPekerjaan)
// Register static routes before dynamic parameter routes so
they don't get treated as :id
// Allow authenticated users to access their own trash;
service filters admin vs user
pekerjaan.Get("/trash", pekerjaanHandler.ListTrash)
pekerjaan.Get("/alumni/:alumni_id", middleware.AdminOnly(),
pekerjaanHandler.GetPekerjaanByAlumniID)
pekerjaan.Get("/:id", pekerjaanHandler.GetPekerjaanByID)
pekerjaan.Post("/", middleware.AdminOnly(),
pekerjaanHandler.CreatePekerjaan)
pekerjaan.Put("/:id", middleware.AdminOnly(),
pekerjaanHandler.UpdatePekerjaan)
pekerjaan.Delete("/:id", middleware.AdminOnly(),
pekerjaanHandler.DeletePekerjaan)
pekerjaan.Patch("/:id/soft-delete",
pekerjaanHandler.SoftDeletePekerjaan)
pekerjaan.Patch("/:id/restore",
pekerjaanHandler.RestorePekerjaan)
pekerjaan.Delete("/:id/hard-delete",
pekerjaanHandler.HardDeletePekerjaan)
}
```

1. MEMBUAT ENDPOINT TRASH

a. Admin

The screenshot shows the Postman application interface. At the top, it says "New Collection / login". On the right, there are "Save", "Share", and "Send" buttons. The URL in the address bar is "http://localhost:3000/alumni-crud-api/auth/login". Below the address bar, the method is set to "POST". The "Body" tab is selected, showing the raw JSON payload:

```
1 {  
2   "username": "admin",  
3   "password": "password"  
4 }
```

On the right side, there are "Schema" and "Beautify" buttons. The "Cookies" section is also visible. Below the body, the "Body" tab is selected again, followed by "Cookies", "Headers (3)", and "Test Results". The status bar at the bottom shows "200 OK" with a response time of "86 ms" and a size of "528 B". There are "Save Response" and other icons on the far right.

The "Body" tab is expanded, showing the full JSON response:

```
1 {  
2   "success": true,  
3   "message": "Login berhasil",  
4   "data": {  
5     "user": {  
6       "id": 1,  
7       "username": "admin",  
8       "email": "admin@university.com",  
9       "role": "admin",  
10      "created_at": "2025-09-16T13:31:16.668373Z",  
11      "updated_at": "2025-09-16T13:31:16.668373Z"  
12    },  
13    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJlc2VyeTlKljoxLC1c2YbmFZSIImFkbWluIiwicm9sZSI6ImFkbWluIiwizXhwIjoxNzU5OTg3MjI1LCJpYXQiOje3NTk5MDA4MjV9.  
738yp3Hcw4nYdBl4VS-zXZIpasbE3yC1TremhpHByxA"  
14  }  
15 }
```

Admin login terlebih dahulu dan mendapat token

The screenshot shows the Postman interface with the following details:

- URL:** http://localhost:3000/alumni-crud-api/pekerjaan/trash
- Method:** GET
- Authorization:** Bearer Token (Token field contains a redacted token)
- Body:** JSON response (shown below)
- Headers:** (7 items listed)
- Test Results:** 200 OK (7 ms, 1.17 KB)

```
{} {"id": 417, "alumni_id": 15, "nama_perusahaan": "PT. Indofood", "posisi_jabatan": "ERP Specialist", "bidang_industri": "Food & Beverage", "lokasi_kerja": "Jakarta", "gaji_range": "-9-9 juta", "tanggal_mulai_kerja": "2024-01-15T00:00:00Z", "tanggal_selesai_kerja": null, "status_pekerjaan": "aktif", "deskripsi_pekerjaan": "Mengimplementasikan sistem ERP untuk operasional", "is_deleted": true, "deleted_at": "2025-10-08T11:31:51.458945+07:00", "created_at": "2025-09-19T08:25:49.516832Z", "updated_at": "2025-09-26T08:11:54.069544Z"} { } 
```

Admin dapat menampilkan semua trash yang ada, trash sendiri adalah data pekerjaan yang sudah di soft delete

b. User

POST http://localhost:3000/alumni-crud-api/auth/login

Body (raw) JSON

```
{
  "username": "syahren",
  "password": "password"
}
```

200 OK

```
{
  "success": true,
  "message": "Login berhasil",
  "data": {
    "user": {
      "id": 4,
      "username": "syahren",
      "email": "syahren@gmail.com",
      "role": "user",
      "created_at": "2025-10-08T09:59:44.449285Z",
      "updated_at": "2025-10-08T09:59:44.449285Z"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJsb2dpbi5nbWFpbC5jb20iLCJ9.eyJ1c2VyX2lkIjoiLCJ1c2VybmttZSI6InNSYhYzW4iLCJyb2xlijoidXNlciiSImV4cCI6MTc1OTk4Nzg2MSwiZWFOIjoxNzU5OTAxNDYxfQ.DeuRjhEmza380q2hI89R1DvzhXh7jqeiIAa15r_geU"
  }
}
```

Login sebagai user Syahren dan mendapatkan token

GET http://localhost:3000/alumni-crud-api/pekerjaan/trash

Auth Type: Bearer Token

Token: (redacted)

200 OK

```
{
  "data": [
    {
      "id": 416,
      "alumni_id": 14,
      "nama_perusahaan": "PT. Pertamina",
      "posisi_jabatan": "Database Administrator",
      "bidang_industri": "oil & Gas",
      "lokasi_kerja": "Jakarta",
      "gaji_range": "10-13 juta",
      "tanggal_mulai_kerja": "2024-02-15T00:00:00Z",
      "tanggal_selesai_kerja": null,
      "status_pekerjaan": "aktif",
      "deskripsi_pekerjaan": "Mengelola database sistem informasi perusahaan",
      "is_deleted": true,
      "deleted_at": "2025-10-08T12:37:27.091003+07:00",
      "deleted_by": 1
    }
  ]
}
```

User bisa melihat Trashnya sendiri

2. MEMBUAT ENDPOINT UNTUK RESTORE

a. Admin

The screenshot shows a Postman collection named "soft delete dan restore". A PATCH request is made to the URL `http://localhost:3000/alumni-crud-api/pekerjaan/417/restore`. The "Authorization" tab is selected, showing "Bearer Token" and a redacted token. The "Body" tab contains a JSON payload:

```

1 {
2   "success": true,
3   "message": "Pekerjaan restored successfully"
4 }

```

The response is a 200 OK status with 22 ms latency and 168 B size. The response body is identical to the request body.

Admin dapat restore trash yang ada

	<code>id</code> [PK] integer	<code>alumni_id</code> integer	<code>nama_perusahaan</code> character varying (100)	<code>posisi_jabatan</code> character varying (100)	<code>bidang_industri</code> character varying (50)	<code>lokasi_kerja</code> character varying (100)	<code>gaji_range</code> character varying (50)	tar da
1	416	14	PT. Pertamina	Database Administrator	Oil & Gas	Jakarta	10-13 juta	20

Setelah dilakukan restore, saat cek trash maka data yang sudah di restore hilang di trash, namun masih ada di data pekerjaan

b. User

The screenshot shows a Postman collection named "soft delete dan restore". A PATCH request is made to the URL `http://localhost:3000/alumni-crud-api/pekerjaan/416/restore`. The "Authorization" tab is selected, showing "Bearer Token" and a redacted token. The "Body" tab contains a JSON payload:

```

1 {
2   "success": true,
3   "message": "Pekerjaan restored successfully"
4 }

```

The response is a 200 OK status with 8 ms latency and 168 B size. The response body is identical to the request body.

User dapat melakukan restore dirinya sendiri

The screenshot shows a Postman collection named "New Collection / get trash". A GET request is made to `http://localhost:3000/alumni-crud-api/pekerjaan/trash`. The "Authorization" tab is selected, showing "Bearer Token" as the auth type with a token input field containing a redacted token. The response status is 200 OK, with a response time of 5 ms, 182 B, and a green "Save Response" button.

Body (JSON) response:

```
1 {  
2   "data": null,  
3   "message": "Trash data retrieved successfully",  
4   "success": true  
5 }
```

Saat dicek, data yang sudah di restore sudah hilang dari Trash

3. MEMBUAT ENDPOINT UNTUK HARD DELETE

a. Admin

The screenshot shows a Postman collection named "New Collection / hard". A DELETE request is made to `http://localhost:3000/alumni-crud-api/pekerjaan/417/hard-delete`. The "Authorization" tab is selected, showing "Bearer Token" as the auth type with a token input field containing a redacted token. The response status is 200 OK, with a response time of 16 ms, 166 B, and a green "Save Response" button.

Body (JSON) response:

```
1 {  
2   "success": true,  
3   "message": "Pekerjaan permanently deleted"  
4 }
```

Admin dapat melakukan Hard Delete

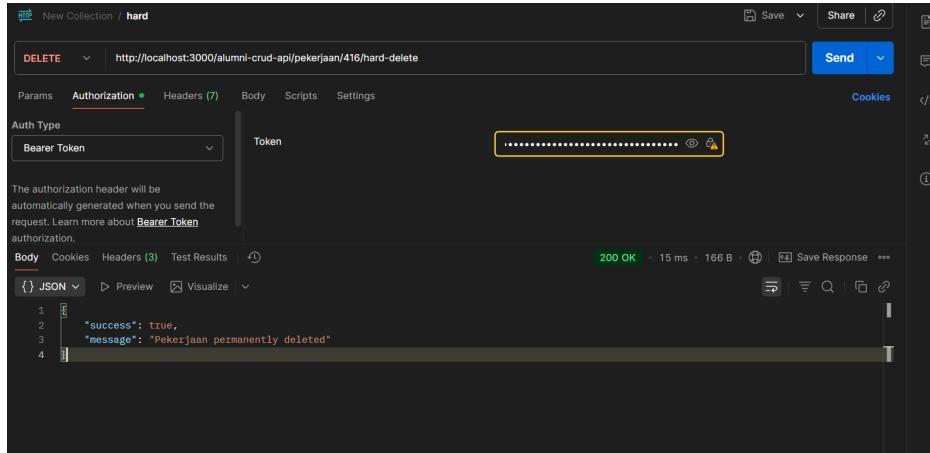
The screenshot shows a Postman collection named "New Collection / get trash". A GET request is made to `http://localhost:3000/alumni-crud-api/pekerjaan/417`. The "Authorization" tab is selected, showing "Bearer Token" as the auth type with a token input field containing a redacted token. The response status is 404 Not Found, with a response time of 8 ms, 175 B, and a green "Save Response" button.

Body (JSON) response:

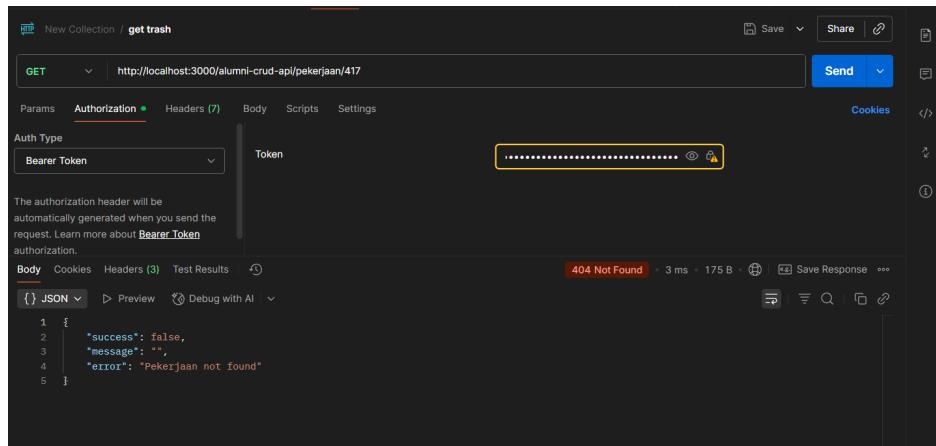
```
1 {  
2   "success": false,  
3   "message": "",  
4   "error": "Pekerjaan not found"  
5 }
```

Pekerjaan sudah dihapus dan hilang dari database

b. User



User dapat melakukan Hard delete



Setelah melakukan hard delete lalu dicek, data pekerjaan sudah hilang permanen dari database