

LAPORAN PRAKTIKUM
PEMOGRAMAN BACKEND



Dosen Pengajar :
Bapak Moh. Nasrul Aziz S. Kom., M. Kom.,

Disusun Oleh :
Noor Fariha Fathmawaty
434231014
D4 Teknik Informatika C5

Migrasi MongoDB

Link Github: <https://github.com/noorfarihaf11/golang.git>

1. GET / project_name/pekerjaan - Ambil semua data pekerjaan alumni (Admin dan User)

pekerjaan_repo.go

```
func GetAllJobs(db *mongo.Database) ([]model.PekerjaanAlumni, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
    defer cancel()

    cur, err := db.Collection("pekerjaan_alumni").Find(ctx, bson.M{"is_deleted": false})
    if err != nil {
        return nil, err
    }
    defer cur.Close(ctx)

    var jobs []model.PekerjaanAlumni
    err = cur.All(ctx, &jobs)
    return jobs, err
}
```

memakai Find() dengan filter {is_deleted: false} untuk menampilkan data aktif saja.

pekerjaan_service.go

```
func GetAllJobService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err != nil {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Token tidak valid"})
    }

    jobs, err := repository.GetAllJobs(db)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"success": false, "message": err.Error()})
    }
    if len(jobs) == 0 {
        return c.Status(404).JSON(fiber.Map{"success": false, "message": "Data kosong"})
    }

    return c.JSON(fiber.Map{"success": true, "data": jobs})
}
```

memvalidasi token JWT, lalu memanggil fungsi repository.

pekerjaan_alumni.go (model)

```
type PekerjaanAlumni struct {
    ID primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    AlumniID primitive.ObjectID `bson:"alumni_id" json:"alumni_id"`
    AlumniIDStr string `bson:"-" json:"alumni_id_str,omitempty" // bantu parsing dari JSON`
    NamaPerusahaan string `bson:"nama_perusahaan" json:"nama_perusahaan"`
    PosisiJabatan string `bson:"posisi_jabatan" json:"posisi_jabatan"`
    BidangIndustri string `bson:"bidang_industri" json:"bidang_industri"`
    LokasiKerja string `bson:"lokasi_kerja" json:"lokasi_kerja"`
    GajiRange string `bson:"gaji_range" json:"gaji_range"`
    TanggalMulaiKerja time.Time `bson:"tanggal_mulai_kerja" json:"tanggal_mulai_kerja"`
    TanggalSelesaikerja *time.Time `bson:"tanggal_selesai_kerja,omitempty" json:"tanggal_selesai_kerja,omitempty"`
    StatusPekerjaan string `bson:"status_pekerjaan" json:"status_pekerjaan"`
    DeskripsiPekerjaan string `bson:"deskripsi_pekerjaan" json:"deskripsi_pekerjaan"`
    CreatedAt time.Time `bson:"created_at" json:"created_at"`
    UpdatedAt time.Time `bson:"updated_at" json:"updated_at"`
    IsDeleted bool `bson:"is_deleted" json:"is_deleted"`
}
```

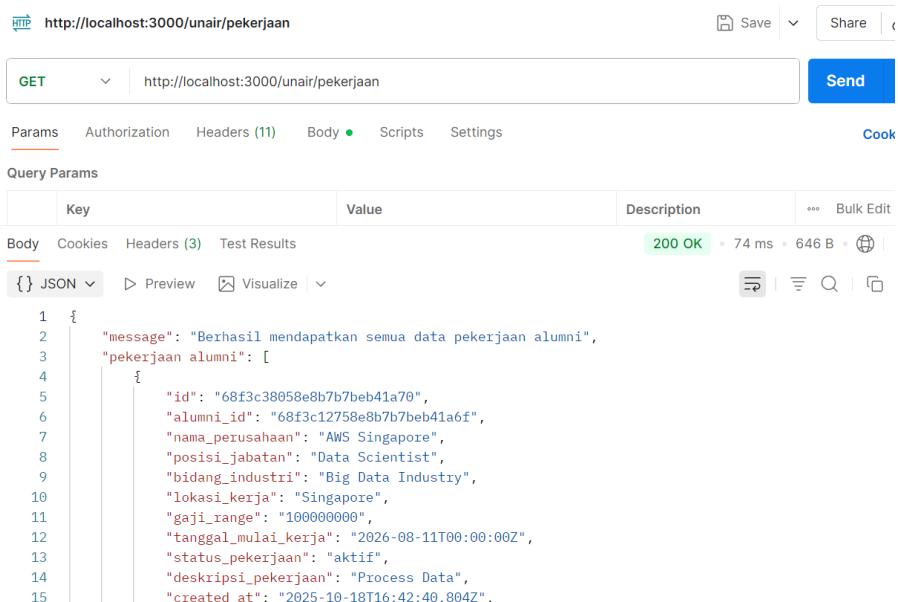
mendefinisikan struktur dokumen dalam koleksi MongoDB.

route

```
func JobRoutes(api fiber.Router, db *mongo.Database) {
    job := api.Group("/unair/pekerjaan", middleware.AuthRequired())

    job.Get("/", func(c *fiber.Ctx) error {
        return service.GetAllJobService(c, db)
    })
}
```

Test di postman



The screenshot shows a Postman request to `http://localhost:3000/unair/pekerjaan`. The response is a 200 OK status with a JSON body:

```
1 {  
2   "message": "Berhasil mendapatkan semua data pekerjaan alumni",  
3   "pekerjaan_alumni": [  
4     {  
5       "id": "68f3c38058e8b7b7beb41a70",  
6       "alumni_id": "68f3c12758e8b7b7beb41a6f",  
7       "nama_perusahaan": "AWS Singapore",  
8       "posisi_jabatan": "Data Scientist",  
9       "bidang_industri": "Big Data Industry",  
10      "lokasi_kerja": "Singapore",  
11      "gaji_range": "100000000",  
12      "tanggal_mulai_kerja": "2026-08-11T00:00:00Z",  
13      "status_pekerjaan": "aktif",  
14      "deskripsi_pekerjaan": "Process Data",  
15      "created_at": "2025-10-18T16:42:40.804Z",  
16    }  
17  ]  
18}
```

Menampilkan seluruh daftar pekerjaan alumni dalam format JSON.

2. GET / project_name/pekerjaan/:id - Ambil data pekerjaan berdasarkan ID (Admin dan User)

pekerjaan_repo.go

```
func GetJobByID(db *mongo.Database, id string) (*model.PekerjaanAlumni, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return nil, fmt.Errorf("ID tidak valid: %v", err)
    }

    var job model.PekerjaanAlumni
    err = db.collection("pekerjaan_alumni").
        FindOne(ctx, bson.M{"_id": objID, "is_deleted": false}).
        Decode(&job)

    if err == mongo.ErrNoDocuments {
        return nil, nil
    }
    return &job, err
}
```

mengubah parameter id menjadi ObjectId MongoDB, lalu mencari data dengan _id tersebut.

Pekerjaan_service.go

```
func GetJobByIDService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err != nil {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Token tidak valid"})
    }

    job, err := repository.GetJobByID(db, c.Params("id"))
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"success": false, "message": err.Error()})
    }
    if job == nil {
        return c.Status(404).JSON(fiber.Map{"success": false, "message": "Pekerjaan tidak ditemukan"})
    }

    return c.JSON(fiber.Map{"success": true, "data": job})
}
```

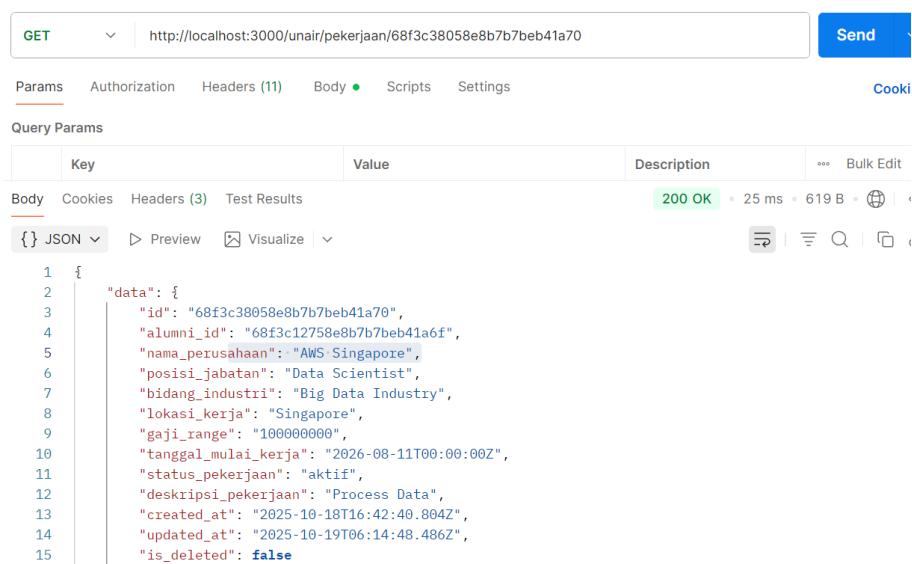
memvalidasi token dan menampilkan hasil dalam response JSON.

Untuk model masih sama dengan yang PekerjaanAlumni tadi

Route

```
job.Get("/:id", func(c *fiber.Ctx) error {
    return service.GetJobByIDService(c, db)
})
```

Test di postman



GET http://localhost:3000/unair/pekerjaan/68f3c38058e8b7b7beb41a70

Params Headers (11) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Body	Cookies	Headers (3)	Test Results

200 OK 25 ms 619 B

```
1 {  
2   "data": {  
3     "id": "68f3c38058e8b7b7beb41a70",  
4     "alumni_id": "68f3c12758e8b7b7beb41a6f",  
5     "nama_perusahaan": "AWS Singapore",  
6     "posisi_jabatan": "Data Scientist",  
7     "bidang_industri": "Big Data Industry",  
8     "lokasi_kerja": "Singapore",  
9     "gaji_range": "100000000",  
10    "tanggal_mulai_kerja": "2026-08-11T00:00:00Z",  
11    "status_pekerjaan": "aktif",  
12    "deskripsi_pekerjaan": "Process Data",  
13    "created_at": "2025-10-18T16:42:40.804Z",  
14    "updated_at": "2025-10-19T06:14:48.486Z",  
15    "is_deleted": false  
16  }  
17}
```

Menampilkan satu data pekerjaan alumni berdasarkan ID yang diberikan.

3. GET / project_name/pekerjaan/alumni/:alumni_id - Ambil semua pekerjaan berdasarkan alumni (Admin)

Pekerjaan_repo.go

```
func GetJobsByAlumniID(db *mongo.Database, alumniID string) ([]model.PekerjaanAlumni, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 10*time.Second)
    defer cancel()

    aid, err := primitive.ObjectIDFromHex(alumniID)
    if err != nil {
        return nil, fmt.Errorf("ID alumni tidak valid")
    }

    cur, err := db.Collection("pekerjaan_alumni").Find(ctx, bson.M{"alumni_id": aid, "is_deleted": false})
    if err != nil {
        return nil, err
    }
    defer cur.Close(ctx)

    var jobs []model.PekerjaanAlumni
    err = cur.All(ctx, &jobs)
    return jobs, err
}
```

mencari semua dokumen dengan alumni_id yang sesuai.

Pekerjaan_service.go

```
func GetJobsByAlumniIDService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err != nil {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Token tidak valid"})
    }

    id := c.Params("alumni_id")
    jobs, err := repository.GetJobsByAlumniID(db, id)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"success": false, "message": err.Error()})
    }
    if len(jobs) == 0 {
        return c.Status(404).JSON(fiber.Map{"success": false, "message": "Tidak ada pekerjaan"})
    }

    return c.JSON(fiber.Map{"success": true, "data": jobs})
}
```

memastikan token valid, lalu mengembalikan seluruh hasil pekerjaan alumni tersebut.

Route

```
job.Get("/alumni/:alumni_id", middleware.AdminOnly(), func(c *fiber.Ctx) error {
    return service.GetJobsByAlumniIDService(c, db)
})
```

Test di postman

The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: http://localhost:3000/unair/pekerjaan/alumni/68f3c12758e8b7b7beb41a6f
- Headers (11):
 - none
 - form-data
 - x-www-form-urlencoded
 - raw (selected)
 - binary
 - GraphQL
 - JSON (selected)
- Body (raw JSON response):

```
{ "data": [ { "id": "68f3c38058e8b7b7beb41a79", "alumni_id": "68f3c12758e8b7b7beb41a6f", "nama_perusahaan": "AWS Singapore", "posisi_jabatan": "Data Scientist", "bidang_industri": "Big Data Industry", "lokasi_kerja": "Singapore", "gaji_range": "1000000000", "tangal_mulai_kerja": "2026-08-11T00:00:00Z", "status_pekerjaan": "aktif", "deskripsi_pekerjaan": "Process Data", "created_at": "2025-10-18T16:42:40.864Z", "updated_at": "2025-10-19T06:14:48.486Z", "is_deleted": false } ] }
```
- Response status: 200 OK
- Time: 81 ms
- Size: 573 B

Menampilkan daftar pekerjaan berdasarkan satu alumni_id.

4. POST / project_name/pekerjaan - Tambah pekerjaan baru (Admin)

Pekerjaan_repo.go

```
func CreateJob(db *mongo.Database, job *model.PekerjaanAlumni) (*model.PekerjaanAlumni, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    id, err := primitive.ObjectIDFromHex(job.AlumniIDStr)
    if err != nil {
        return nil, fmt.Errorf("alumni_id tidak valid")
    }
    job.AlumniID = id
    job.ID = primitive.NewObjectID()
    job.CreatedAt, job.UpdatedAt = time.Now(), time.Now()
    job.IsDeleted = false

    if _, err := db.collection("pekerjaan_alumni").InsertOne(ctx, job); err != nil {
        return nil, err
    }
    return job, nil
}
```

membuat ObjectId baru, mengisi waktu created_at, updated_at, dan menyimpan data dengan InsertOne().

Pekerjaan_service.go

```
func CreateJobService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err != nil {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Token tidak valid"})
    }

    var job model.PekerjaanAlumni
    if err := c.BodyParser(&job); err != nil {
        return c.Status(400).JSON(fiber.Map{"success": false, "message": "Body tidak valid"})
    }

    res, err := repository.CreateJob(db, &job)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"success": false, "message": err.Error()})
    }

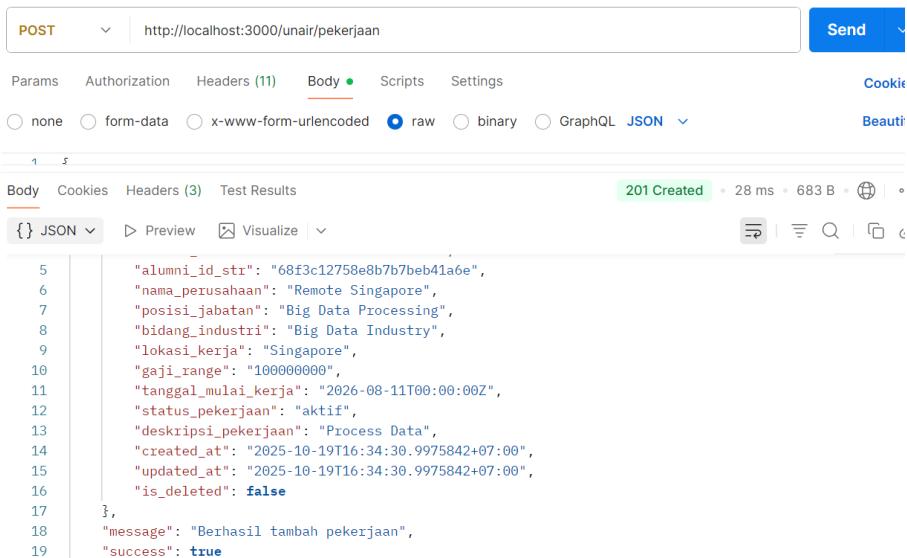
    return c.Status(201).JSON(fiber.Map{"success": true, "message": "Berhasil tambah pekerjaan", "data": res})
}
```

melakukan validasi token dan membaca data dari BodyParser().

Route

```
job.Post("/", middleware.AdminOnly(), func(c *fiber.Ctx) error {
    return service.CreateJobService(c, db)
})
```

Test di postman



The screenshot shows a Postman interface with a POST request to `http://localhost:3000/unair/pekerjaan`. The response is a 201 Created status with a response body containing JSON data and a success message.

```
201 Created
{
  "alumni_id": "68f3c12758e8b7b7beb41a6e",
  "nama_perusahaan": "Remote Singapore",
  "posisi_jabatan": "Big Data Processing",
  "bidang_industri": "Big Data Industry",
  "lokasi_kerja": "Singapore",
  "gaji_range": "100000000",
  "tanggal_mulai_kerja": "2026-08-11T00:00:00Z",
  "status_pekerjaan": "aktif",
  "deskripsi_pekerjaan": "Process Data",
  "created_at": "2025-10-19T16:34:30.9975842+07:00",
  "updated_at": "2025-10-19T16:34:30.9975842+07:00",
  "is_deleted": false
},
"message": "Berhasil tambah pekerjaan",
"success": true
}
```

Menampilkan pesan “Pekerjaan berhasil ditambahkan” dan data yang baru dibuat.

```
1  {
2    "error": "Akses ditolak. Hanya admin yang diizinkan"
3 }
```

Ada middleware khusus admin, apabila alumni yang mengakses tidak diizinkan.

5. PUT / project_name/pekerjaan/:id - Update data pekerjaan (Admin)

Pekerjaan_repo.go

```
func UpdateJob(db *mongo.Database, id string, data model.PekerjaanAlumni) (*model.PekerjaanAlumni, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return nil, fmt.Errorf("ID tidak valid")
    }
    if data.AlumniIDStr != "" {
        if data.AlumniID, err = primitive.ObjectIDFromHex(data.AlumniIDStr); err != nil {
            return nil, fmt.Errorf("alumni_id tidak valid")
        }
    }

    data.UpdatedAt = time.Now()
    update := bson.M{"$set": bson.M{
        "alumni_id": data.AlumniID, "nama_perusahaan": data.NamaPerusahaan,
        "posisi_jabatan": data.PosisiJabatan, "bidang_industri": data.BidangIndustri,
        "lokasi_kerja": data.LokasiKerja, "gaji_range": data.GajiRange,
        "tanggal_mulai_kerja": data.TanggalMulaiKerja, "tanggal_selesai_kerja": data.TanggalSelesaiKerja,
        "status_pekerjaan": data.StatusPekerjaan, "deskripsi_pekerjaan": data.DeskripsiPekerjaan,
        "updated_at": data.UpdatedAt,
    }}

    if r, err := db.collection("pekerjaan_alumni").updateOne(ctx, bson.M{"_id": objID, "is_deleted": false}, update); err != nil {
        return nil, err
    } else if r.MatchedCount == 0 {
        return nil, fmt.Errorf("data tidak ditemukan")
    }

    var updated model.PekerjaanAlumni
    if err := db.collection("pekerjaan_alumni").FindOne(ctx, bson.M{"_id": objID}).Decode(&updated); err != nil {
        return nil, err
    }
    return &updated, nil
}
```

menggunakan UpdateOne() dengan \$set untuk memperbarui kolom yang diubah.
Pekerjaan_service.go

```
func UpdateJobService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err != nil {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Token tidak valid"})
    }

    var job model.PekerjaanAlumni
    if err := c.BodyParser(&job); err != nil {
        return c.Status(400).JSON(fiber.Map{"success": false, "message": "Body tidak valid"})
    }

    res, err := repository.UpdateJob(db, c.Params("id"), job)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"success": false, "message": err.Error()})
    }

    return c.Status(200).JSON(fiber.Map{"success": true, "message": "Berhasil update pekerjaan", "data": res})
}
```

menerima data dari body dan meneruskannya ke repository.

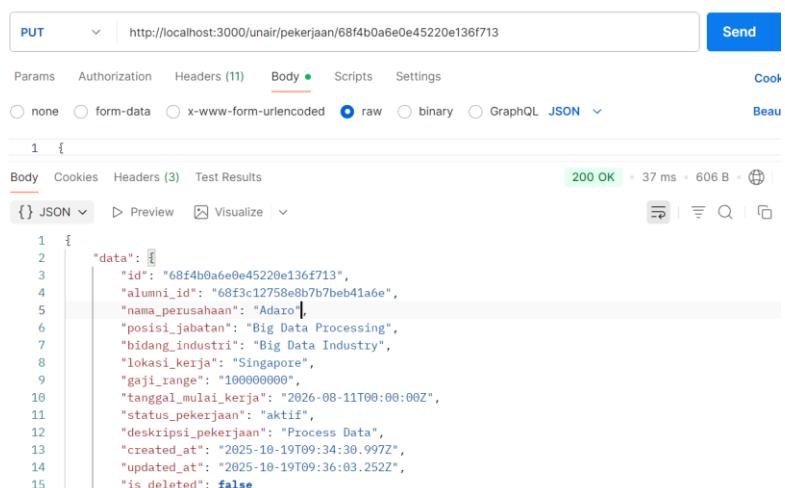
route

```
job.Put("/:id", middleware.AdminOnly(), func(c *fiber.Ctx) error {
    return service.UpdateJobService(c, db)
})
```

Test di postman:

```
1  {
2  |     "error": "Akses ditolak. Hanya admin yang diizinkan"
3 }
```

Ada middleware khusus admin, apabila alumni yang mengakses tidak diizinkan.



```
PUT http://localhost:3000/unair/pekerjaan/68f4b0a6e0e45220e136f713
```

Params Authorization Headers (11) Body Scripts Settings Cook Beau

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2 | "data": {
3 | "id": "68f4b0a6e0e45220e136f713",
4 | "alumni_id": "68f3c12758e8b7b7beb41a6e",
5 | "nama_perusahaan": "Adaxo",
6 | "posisi_jabatan": "Big Data Processing",
7 | "bidang_industri": "Big Data Industry",
8 | "lokasi_kerja": "Singapore",
9 | "gaji_range": "1000000000",
10 | "tanggal_mulai_kerja": "2026-08-11T00:00:00Z",
11 | "status_pekerjaan": "aktif",
12 | "deskripsi_pekerjaan": "Process Data",
13 | "created_at": "2025-10-19T09:34:30.997Z",
14 | "updated_at": "2025-10-19T09:36:03.252Z",
15 | "is_deleted": false
16 }

Menampilkan data pekerjaan yang telah diperbarui.

6. DELETE / project_name/pekerjaan/:id - Hapus data pekerjaan (Admin)

Pekerjaan_repo.go

```
func HardDelete(db *mongo.Database, jobID string) (int64, error) {
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    oid, err := primitive.ObjectIDFromHex(jobID)
    if err != nil {
        return 0, err
    }

    res, err := db.Collection("pekerjaan_alumni").DeleteOne(ctx, bson.M{"_id": oid, "is_deleted": true})
    if err != nil {
        return 0, err
    }

    return res.DeletedCount, nil
}
```

Menghapus menggunakan deleteone untuk id yang diinput dan is_deleted nya true

```
func HardDeleteService(c *fiber.Ctx, db *mongo.Database) error {
    jobID := c.Params("id")

    rows, err := repository.HardDelete(db, jobID)
    if err != nil {
        return c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
            "message": "Gagal delete data: " + err.Error(),
            "success": false,
        })
    }

    if rows == 0 {
        return c.Status(fiber.StatusForbidden).JSON(fiber.Map{
            "message": fmt.Sprintf("Tidak diizinkan menghapus pekerjaan dengan ID %s", jobID),
            "success": false,
        })
    }

    return c.Status(fiber.StatusOK).JSON(fiber.Map{
        "message": "Pekerjaan berhasil dihapus",
        "success": true,
    })
}
```

Memastikan token valid dan memanggil repository

route

```
job.Delete("/:id", middleware.AdminOnly(), func(c *fiber.Ctx) error {
    return service.HardDeleteService(c, db)
})
```

Test di postman

The screenshot shows the Postman interface with a successful API call. The method is set to **DELETE**, the URL is **http://localhost:3000/unair/pekerjaan/delete**, and the response status is **200 OK**. The response body is a JSON object:

```
{ "message": "Pekerjaan berhasil dihapus", "success": true }
```

Berhasil menghapus pekerjaan

```
1  {
2   |   "error": "Akses ditolak. Hanya admin yang diizinkan"
3 }
```

Ada middleware khusus admin, apabila alumni yang mengakses tidak diizinkan.