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

```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

```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)

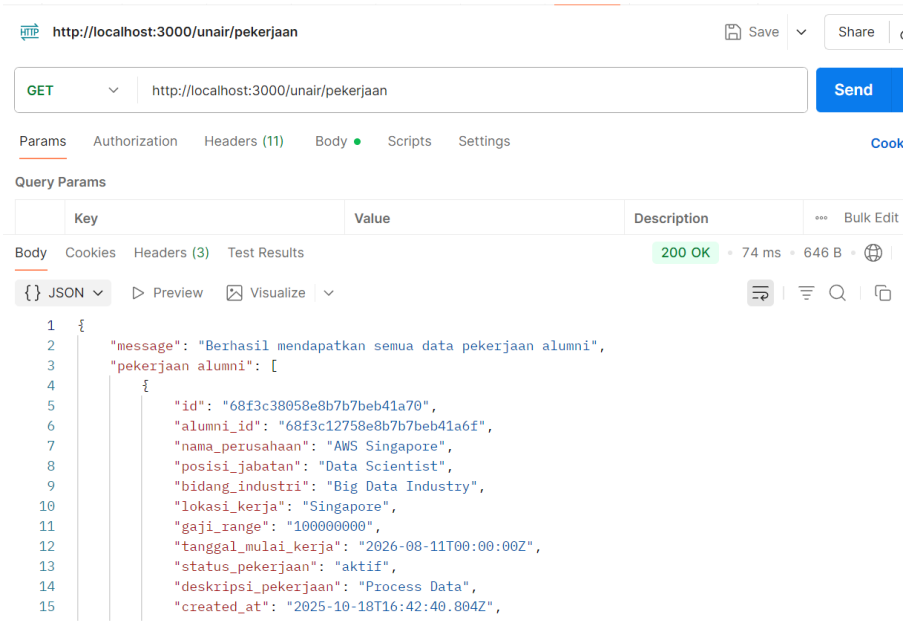
```go
type PekerjaanAlumni struct {
    ID                  primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    AlumniID            primitive.ObjectID `bson:"alumni_id" json:"alumni_id"`
    AlumniIDStr         string             `bson:"-" json:"alumni_id_str,omitempty"` // bantu parsing dari JSON
    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

```go
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



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

```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

```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})
}
```

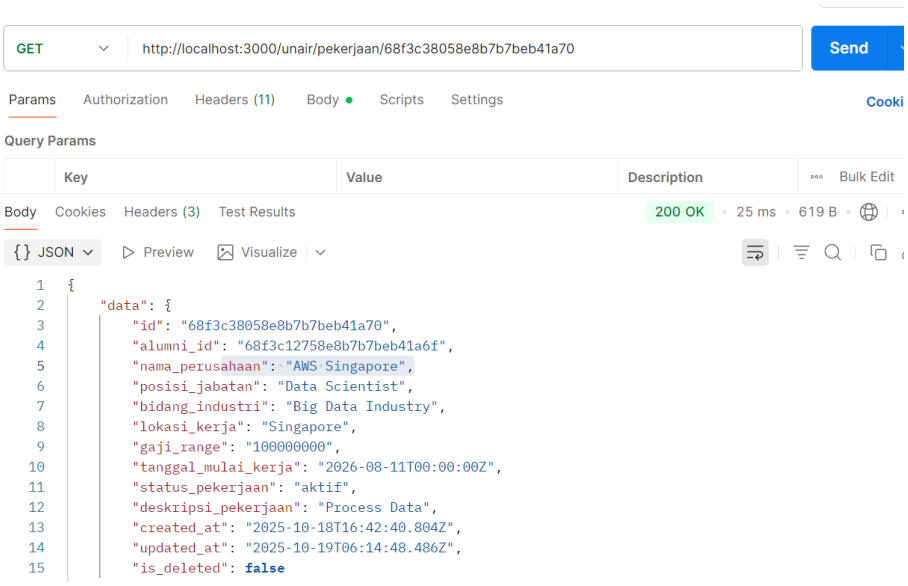
memvalidasi token dan menampilkan hasil dalam response JSON.

Untuk model masih sama dengan yang PekerjaanAlumni tadi

Route

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

Test di postman



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

```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

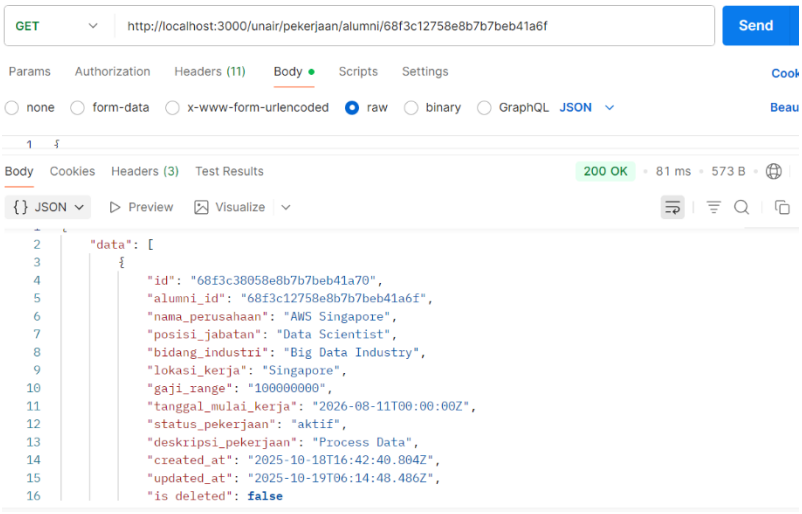
```go
func GetJobsByAlumniIDService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token);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

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

Test di postman



Menampilkan daftar pekerjaan berdasarkan satu alumni_id.

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

Pekerjaan_repo.go

```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

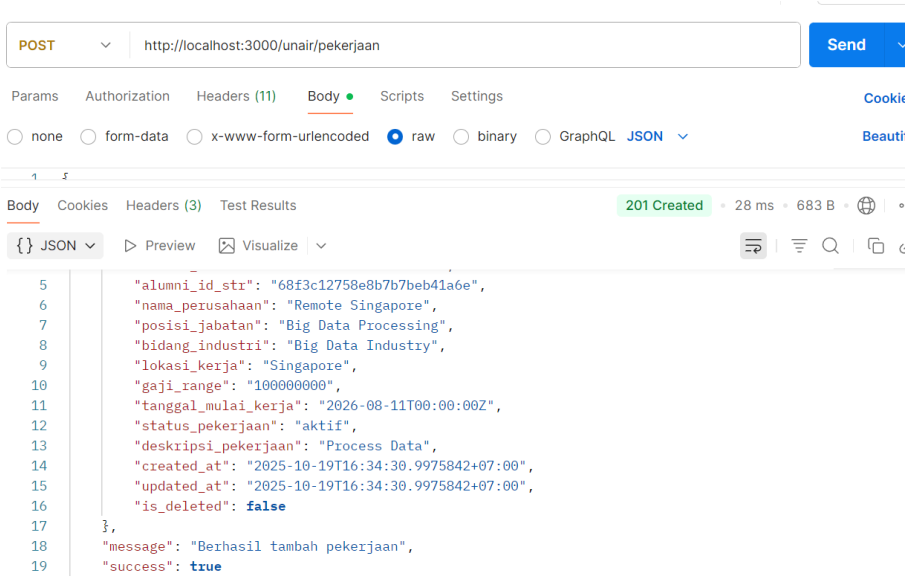
```go
func CreateJobService(c *fiber.Ctx, db *mongo.Database) error {
    token := strings.TrimPrefix(c.Get("Authorization"), "Bearer ")
    if token == "" {
        return c.Status(401).JSON(fiber.Map{"success": false, "message": "Unauthorized"})
    }
    if _, err := utils.ValidateToken(token); err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

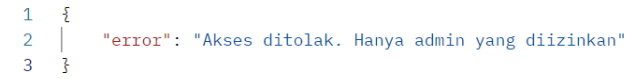
Route

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

Test di postman



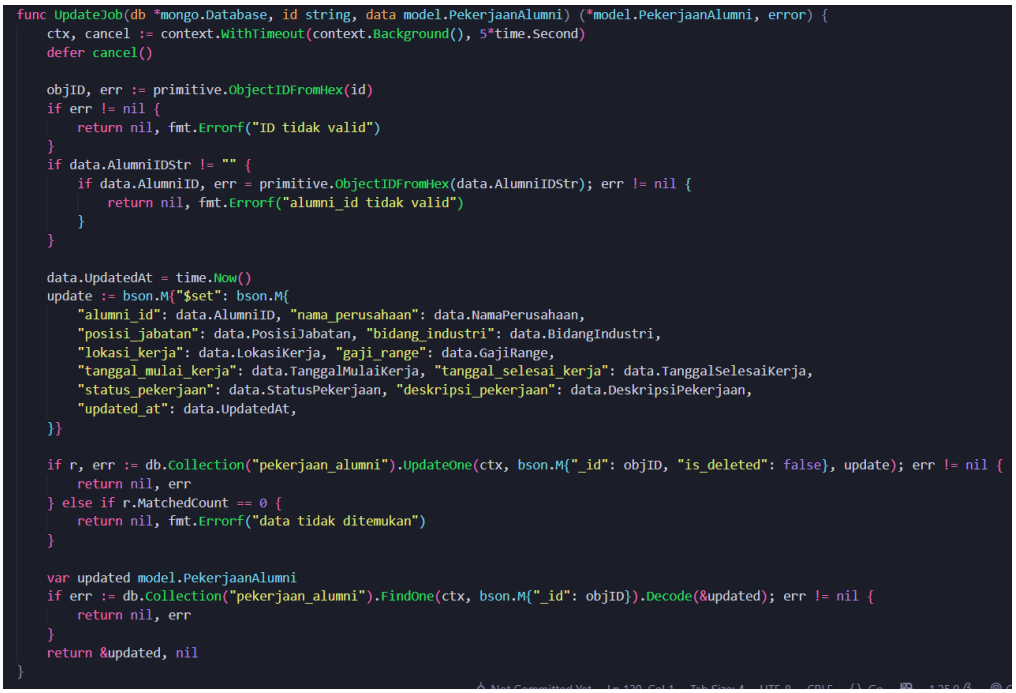
Menampilkan pesan “Pekerjaan berhasil ditambahkan” dan data yang baru dibuat.



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

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

Pekerjaan_repo.go



```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

```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.

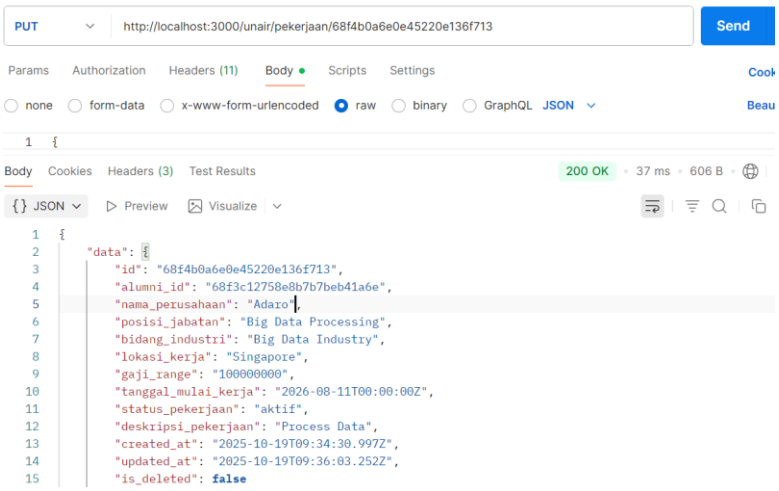
route

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

Test di postman:

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

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



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

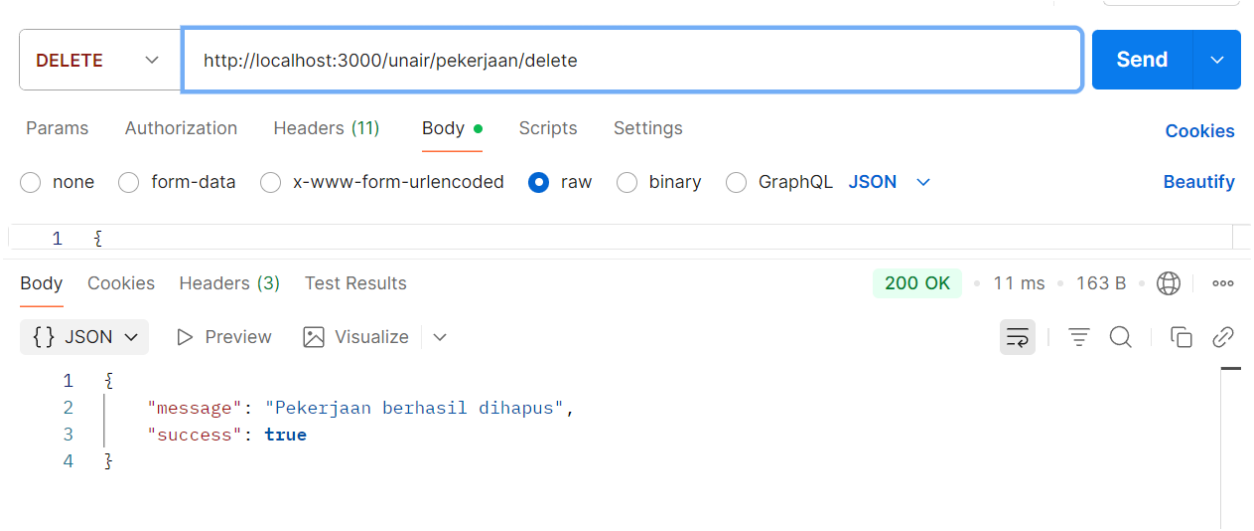
```
func HardDeleteService(c *fiber.Ctx, db *mongo.Database) error {

    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



Berhasil menghapus pekerjaan

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

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