LAPORAN PRAKTIKUM

PEMOGRAMAN BACKEND

Dosen Pengajar :

Bapak Moh. Nasrul Aziz S. Kom., M. Kom.,

Disusun Oleh :

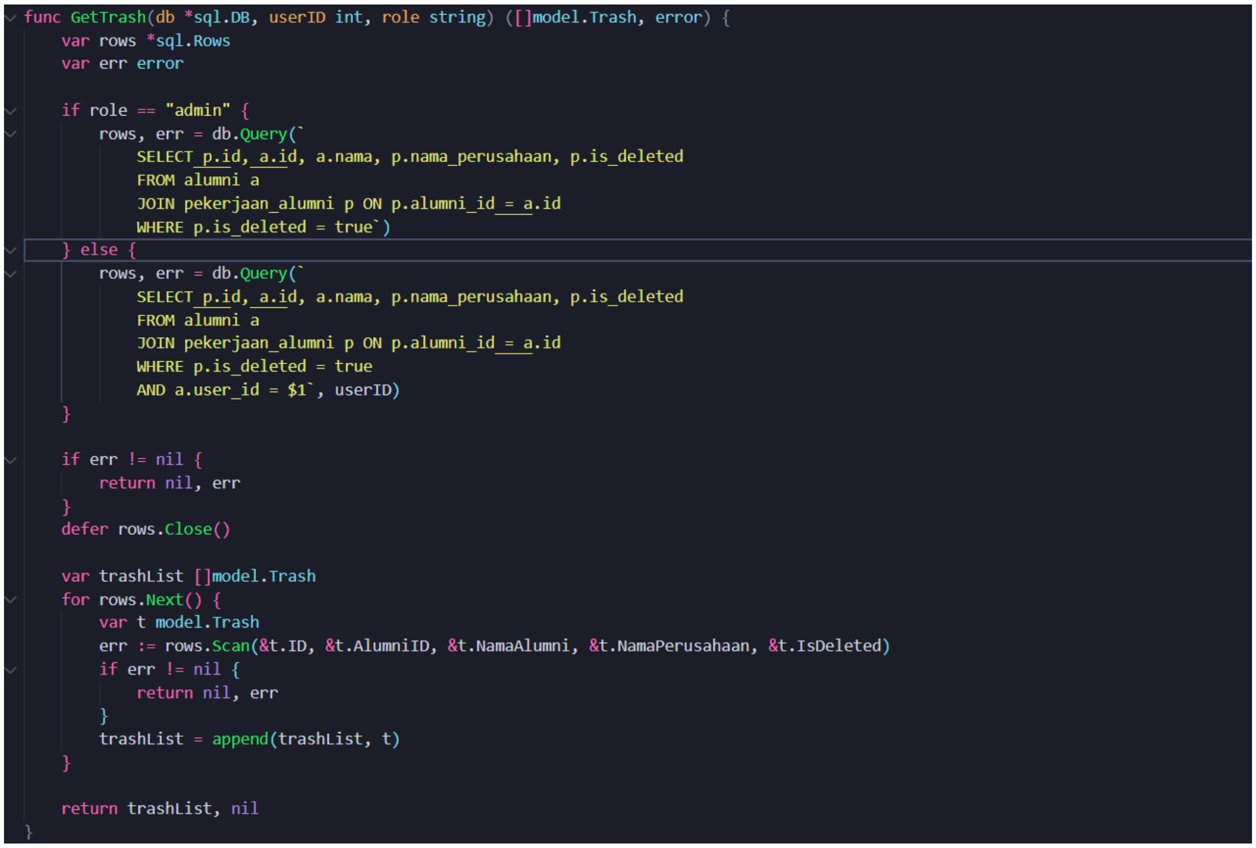
Noor Fariha Fathmawaty

434231014

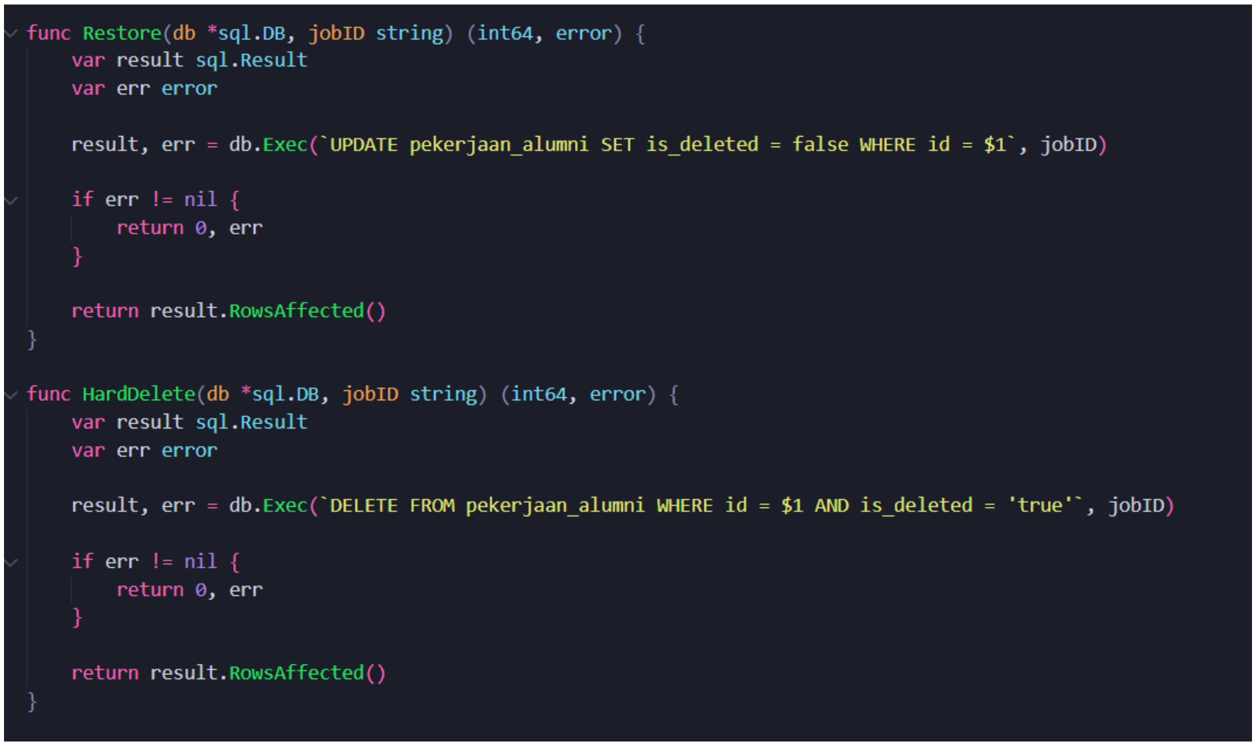
D4 Teknik Informatika C5

Trash, Restore dan Hard Delete

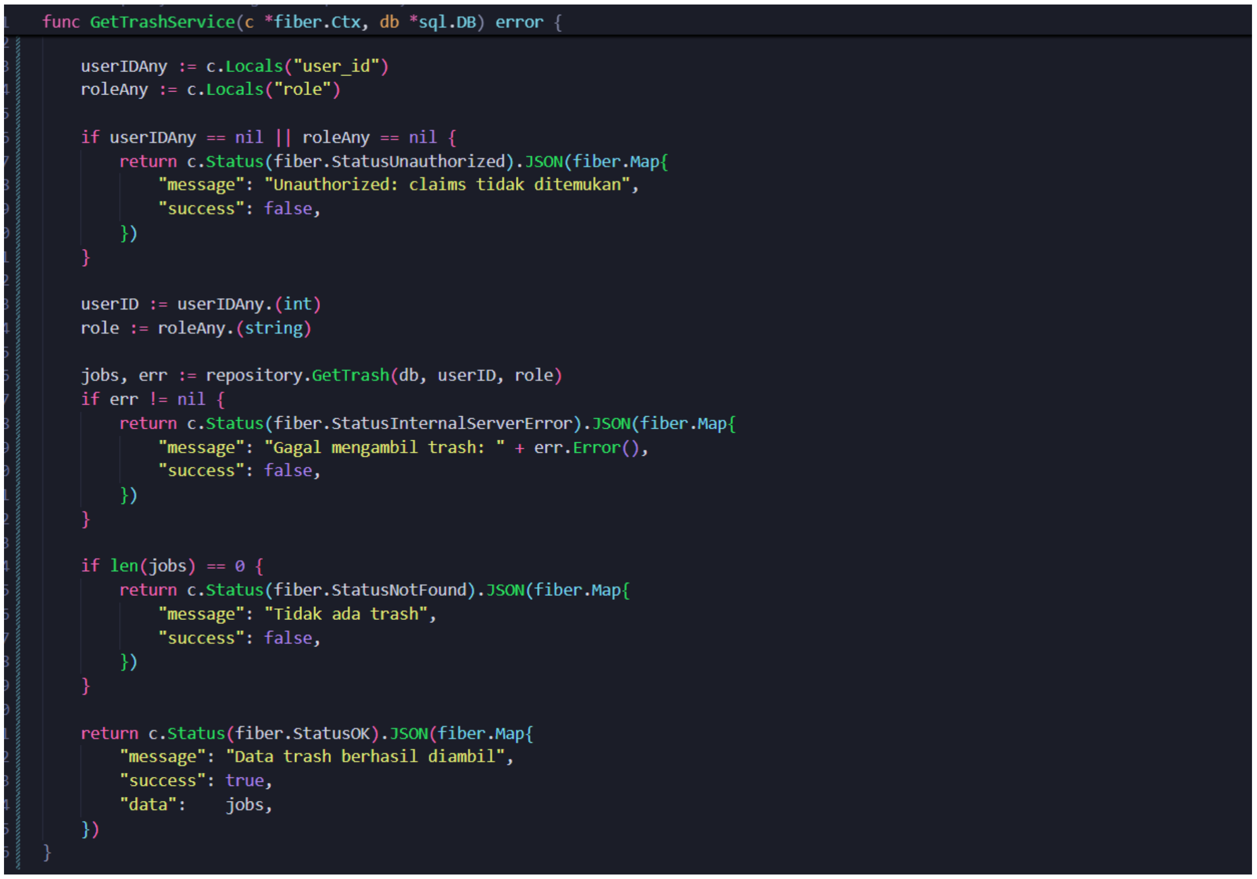
LINK GITHUB: https://github.com/noorfarihaf11/golang



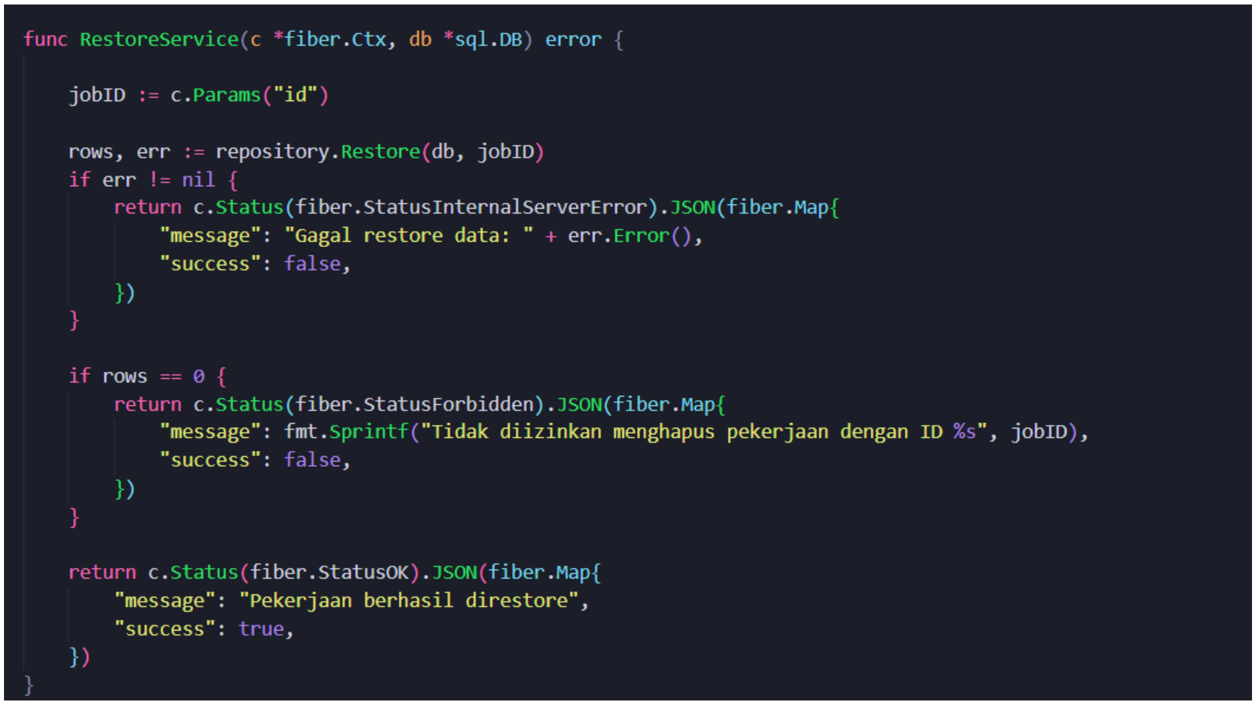
```go
func GetTrash(db *sql.DB, userID int, role string) ([]model.Trash, error) {
    var rows *sql.Rows
    var err error

    if role == "admin" {
        rows, err = db.Query(`
            SELECT p.id, a.id, a.nama, p.nama_perusahaan, p.is_deleted
            FROM alumni a
            JOIN pekerjaan_alumni p ON p.alumni_id = a.id
            WHERE p.is_deleted = true`)
    } else {
        rows, err = db.Query(`
            SELECT p.id, a.id, a.nama, p.nama_perusahaan, p.is_deleted
            FROM alumni a
            JOIN pekerjaan_alumni p ON p.alumni_id = a.id
            WHERE p.is_deleted = true
            AND a.user_id = $1`, userID)
    }

    if err != nil {
        return nil, err
    }
    defer rows.Close()

    var trashList []model.Trash
    for rows.Next() {
        var t model.Trash
        err := rows.Scan(&t.ID, &t.AlumniID, &t.NamaAlumni, &t.NamaPerusahaan, &t.IsDeleted)
        if err != nil {
            return nil, err
        }
        trashList = append(trashList, t)
    }

    return trashList, nil
}
```



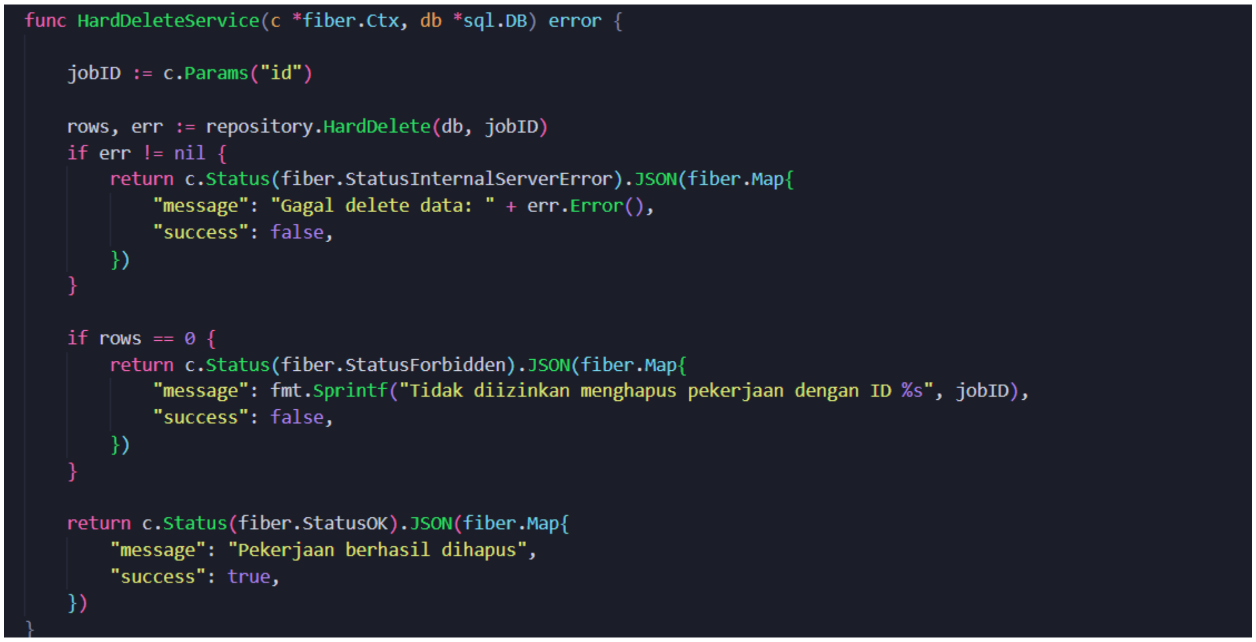
```go
func Restore(db *sql.DB, jobID string) (int64, error) {
    var result sql.Result
    var err error

    result, err = db.Exec(`UPDATE pekerjaan_alumni SET is_deleted = false WHERE id = $1`, jobID)

    if err != nil {
        return 0, err
    }

    return result.RowsAffected()
}

func HardDelete(db *sql.DB, jobID string) (int64, error) {
    var result sql.Result
    var err error

    result, err = db.Exec(`DELETE FROM pekerjaan_alumni WHERE id = $1 AND is_deleted = 'true'`, jobID)

    if err != nil {
        return 0, err
    }

    return result.RowsAffected()
}
```



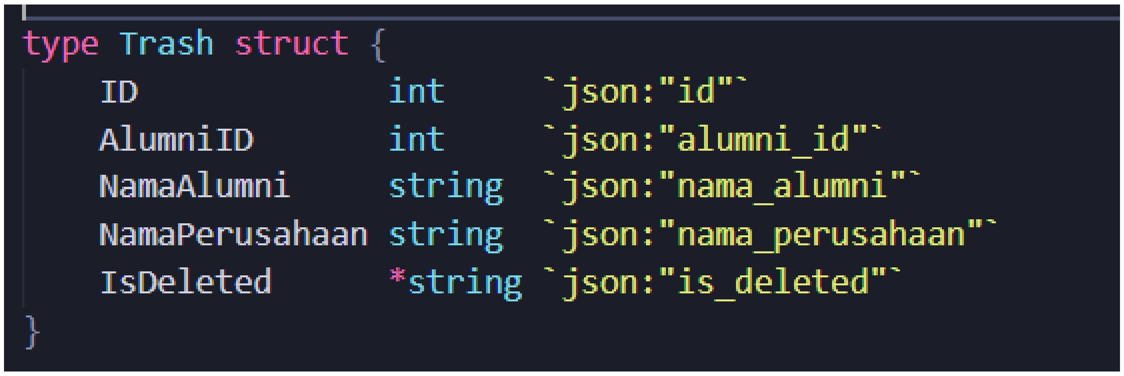
```go
func GetTrashService(c *fiber.Ctx, db *sql.DB) error {

    userIDAny := c.Locals("user_id")
    roleAny := c.Locals("role")

    if userIDAny == nil || roleAny == nil {
        return c.Status(fiber.StatusUnauthorized).JSON(fiber.Map{
            "message": "Unauthorized: claims tidak ditemukan",
            "success": false,
        })
    }

    userID := userIDAny.(int)
    role := roleAny.(string)

    jobs, err := repository.GetTrash(db, userID, role)
    if err != nil {
        return c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
            "message": "Gagal mengambil trash: " + err.Error(),
            "success": false,
        })
    }

    if len(jobs) == 0 {
        return c.Status(fiber.StatusNotFound).JSON(fiber.Map{
            "message": "Tidak ada trash",
            "success": false,
        })
    }

    return c.Status(fiber.StatusOK).JSON(fiber.Map{
        "message": "Data trash berhasil diambil",
        "success": true,
        "data":    jobs,
    })
}
```



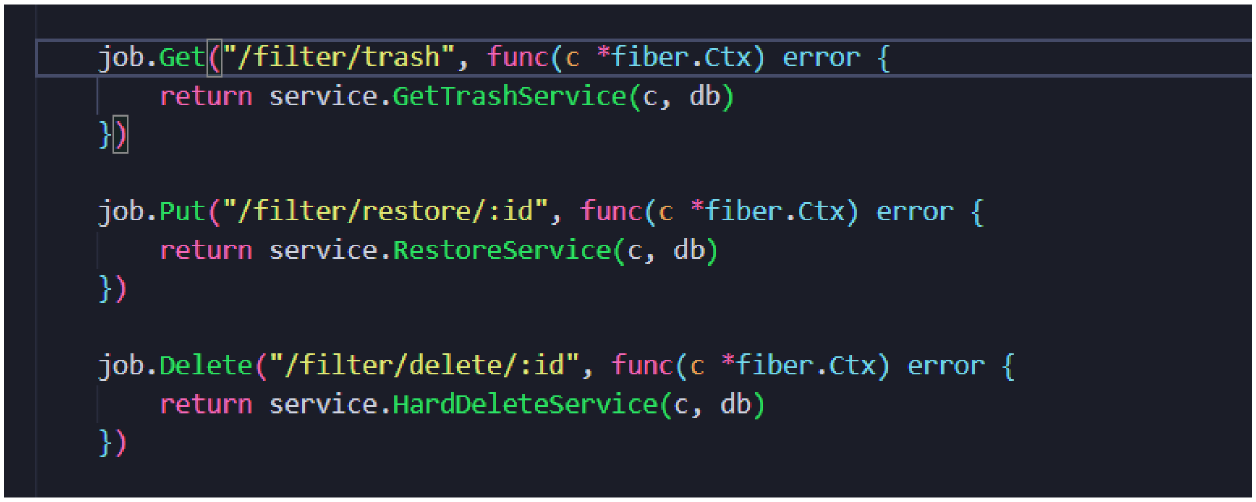
```go
func RestoreService(c *fiber.Ctx, db *sql.DB) error {

    jobID := c.Params("id")

    rows, err := repository.Restore(db, jobID)
    if err != nil {
        return c.Status(fiber.StatusInternalServerError).JSON(fiber.Map{
            "message": "Gagal restore 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 direstore",
        "success": true,
    })
}
```



```go
func HardDeleteService(c *fiber.Ctx, db *sql.DB) 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,
    })
}
```



```go
type Trash struct {
    ID              int     `json:"id"`
    AlumniID        int     `json:"alumni_id"`
    NamaAlumni      string  `json:"nama_alumni"`
    NamaPerusahaan  string  `json:"nama_perusahaan"`
    IsDeleted       *string `json:"is_deleted"`
}
```



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

    job.Put("/filter/restore/:id", func(c *fiber.Ctx) error {
        return service.RestoreService(c, db)
    })

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