Pagination, Sorting & Search Dengan Go Fiber

Tujuan Pembelajaran

- Memahami konsep pagination untuk menangani data dalam jumlah besar
- Mengimplementasikan sorting data berdasarkan kolom tertentu
- Membuat fitur search/pencarian data
- Mengoptimalkan performa query database

1. Pagination

Pagination adalah teknik membagi data besar menjadi bagian-bagian kecil (halaman) untuk:

- Mengurangi beban server dan database
- Mempercepat loading time
- Meningkatkan user experience
- Menghemat bandwidth

2. Sorting

Sorting memungkinkan user mengurutkan data berdasarkan:

- Kolom tertentu (nama, tanggal, ID)
- Arah sorting (ascending/descending)
- Multiple column sorting

3. Search/Filter

Search memungkinkan user mencari data berdasarkan:

- Keyword di multiple kolom
- Filter berdasarkan kriteria spesifik
- Kombinasi filter

1. Persiapan Database

```sql
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT now()
);

-- Dummy data
INSERT INTO users (name, email)
SELECT 'User ' || i, 'user' || i || '@mail.com'
FROM generate_series(1, 2000) s(i);
```

2. Struktur Project

```
fiber-pagination/
├── main.go
├── db/
│   └── db.go
├── routes/
│   └── user_route.go
├── services/
│   └── user_service.go
├── repository/
│   └── user_repository.go
├── models/
│   └── user.go          // Model user
│   └── response.go      // Struct untuk response & meta
```

3. Models untuk response (models/response.go)

```go
package models

// MetaInfo -> informasi pagination & filter
type MetaInfo struct {
    Page   int    `json:"page"`
    Limit  int    `json:"limit"`
    Total  int    `json:"total"`
    Pages  int    `json:"pages"`
    SortBy string `json:"sortBy"`
    Order  string `json:"order"`
    Search string `json:"search"`
}

// UserResponse -> hasil akhir untuk endpoint /users
type UserResponse struct {
```

```
    Data []User   `json:"data"`
    Meta MetaInfo `json:"meta"`
}
```

4. Model (models/user.go)

```
package models

import "time"

// User merepresentasikan 1 record di tabel users
type User struct {
    ID        int       `json:"id"`
    Name      string    `json:"name"`
    Email     string    `json:"email"`
    CreatedAt time.Time `json:"created_at"`
}
```

5. Repository (repository/user_repository.go)

```
package repository

import (
    "database/sql"
    "fmt"
    "log"

    "fiber-pagination/db"
    "fiber-pagination/models"
)

// GetUsersRepo -> ambil data users dari DB
func GetUsersRepo(search, sortBy, order string, limit, offset int)
([]models.User, error) {
    query := fmt.Sprintf(`
        SELECT id, name, email, created_at
        FROM users
        WHERE name ILIKE $1 OR email ILIKE $1
        ORDER BY %s %s
        LIMIT $2 OFFSET $3
    `, sortBy, order)

    rows, err := db.DB.Query(query, "%"+search+"%", limit, offset)
    if err != nil {
        log.Println("Query error:", err)
        return nil, err
    }
```

```
    defer rows.Close()

    var users []models.User
    for rows.Next() {
        var u models.User
        if err := rows.Scan(&u.ID, &u.Name, &u.Email, &u.CreatedAt);
err != nil {
            return nil, err
        }
        users = append(users, u)
    }

    return users, nil
}

// CountUsersRepo -> hitung total data untuk pagination
func CountUsersRepo(search string) (int, error) {
    var total int
    countQuery := `SELECT COUNT(*) FROM users WHERE name ILIKE $1 OR
email ILIKE $1`
    err := db.DB.QueryRow(countQuery, "%"+search+"%").Scan(&total)
    if err != nil && err != sql.ErrNoRows {
        return 0, err
    }
    return total, nil
}
```

6. Service (services/user_service.go)

```
package services

import (
    "strconv"
    "strings"

    "github.com/gofiber/fiber/v2"
    "fiber-pagination/models"
    "fiber-pagination/repository"
)

// GetUsersService -> service untuk ambil data user dengan
pagination, search, sorting
func GetUsersService(c *fiber.Ctx) error {
    page, _ := strconv.Atoi(c.Query("page", "1"))
    limit, _ := strconv.Atoi(c.Query("limit", "10"))
    sortBy := c.Query("sortBy", "id")
    order := c.Query("order", "asc")
    search := c.Query("search", "")
```

```
    offset := (page - 1) * limit

    // Validasi input
    sortByWhitelist := map[string]bool{"id": true, "name": true,
"email": true, "created_at": true}
    if !sortByWhitelist[sortBy] {
        sortBy = "id"
    }
    if strings.ToLower(order) != "desc" {
        order = "asc"
    }

    // Ambil data dari repository
    users, err := repository.GetUsersRepo(search, sortBy, order,
limit, offset)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"error": "Failed to fetch
users"})
    }

    total, err := repository.CountUsersRepo(search)
    if err != nil {
        return c.Status(500).JSON(fiber.Map{"error": "Failed to count
users"})
    }

    // Buat response pakai model
    response := models.UserResponse{
        Data: users,
        Meta: models.MetaInfo{
            Page:   page,
            Limit:  limit,
            Total:  total,
            Pages:  (total + limit - 1) / limit,
            SortBy: sortBy,
            Order:  order,
            Search: search,
        },
    }

    return c.JSON(response)
}
```

GetUsersService

1. Mengambil Parameter Kueri dari URL 🗂️

Baris-baris kode pertama dalam fungsi ini fokus pada **ekstraksi informasi dari parameter kueri** yang dikirimkan melalui URL permintaan. Parameter kueri adalah bagian dari URL yang mengikuti tanda tanya ?, seperti ?page=2&limit=20.

- page, _ := strconv.Atoi(c.Query("page", "1")):
    - c.Query("page", "1") mengambil nilai dari parameter kueri bernama page. Jika parameter page tidak ada dalam URL, maka nilai **default-nya adalah "1"**.
    - strconv.Atoi() kemudian mencoba mengonversi nilai yang didapat (yang awalnya berupa string) menjadi sebuah **integer** (bilangan bulat).
    - Tanda underscore (_) digunakan untuk **mengabaikan nilai error** yang mungkin dikembalikan oleh strconv.Atoi(). Ini adalah praktik umum jika kita yakin konversi akan berhasil atau jika kita tidak perlu menangani error secara spesifik di sini.
- limit, _ := strconv.Atoi(c.Query("limit", "10")):
    - Serupa dengan page, baris ini mengambil parameter kueri limit. Jika tidak ada, **default-nya adalah "10"**.
    - Nilai tersebut juga dikonversi menjadi integer.
- sortBy := c.Query("sortBy", "id"):
    - Mengambil nilai parameter kueri sortBy. Jika tidak ada, **default-nya adalah "id"**. Nilai ini akan digunakan untuk menentukan kolom mana yang akan dijadikan dasar pengurutan.
- order := c.Query("order", "asc"):
    - Mengambil nilai parameter kueri order. Jika tidak ada, **default-nya adalah "asc"** (ascending/naik). Nilai ini akan menentukan apakah pengurutan dilakukan dari yang terkecil ke terbesar (asc) atau sebaliknya (desc).
- search := c.Query("search", ""):

- Mengambil nilai parameter kueri search. Jika tidak ada, **default-nya adalah string kosong** (""). Nilai ini akan digunakan untuk memfilter hasil berdasarkan kriteria pencarian.

2. Menghitung Offset untuk Paginasi 📏

offset := (page - 1) * limit:

- Baris ini menghitung nilai **offset**. Offset digunakan dalam kueri database untuk menentukan dari data keberapa hasil harus mulai diambil.
- Misalnya, jika page adalah 2 dan limit adalah 10:
  - offset akan menjadi (2 - 1) * 10 = 1 * 10 = 10.
  - Ini berarti kita ingin mengambil data mulai dari item ke-11 (karena offset dimulai dari 0).

7. Route (routes/user_route.go)

```go
package routes

import (
    "github.com/gofiber/fiber/v2"
    "fiber-pagination/services"
)

// UserRoutes -> definisi route untuk user
func UserRoutes(app *fiber.App) {
    app.Get("/users", services.GetUsersService)
}
```

8. Main (main.go)

```go
package main

import (
    "github.com/gofiber/fiber/v2"
    "fiber-pagination/db"
    "fiber-pagination/routes"
)

func main() {
    app := fiber.New()

    // Koneksi DB
```

```
    db.Connect()

    // Daftar routes
    routes.UserRoutes(app)

    // Jalankan server
    app.Listen(":3000")
}
```

Tugas 6

Buat data dummy untuk table alumni dan pekerjaan_alumni kemudian Implementasikan Pagination, Sorting & Search pada tugas yang telah dibuat sebelumnya.

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