

**Nama** : Hanif Muhammad Zhafran Sutisna

**Kelas / NPM** : 3IA05 / 50422650

**Mata Praktikum** : Lepkom Golang for Intermediate

---

## **Activity 7 - Integrasi Database MySQL pada Golang**

### **Bagian 1**

1. Jelaskan mengapa pentingnya sebuah program terintegrasi dengan database!

**Jawab:**

Program yang terintegrasi dengan database penting karena:

- **Menyimpan Data Secara Permanen**

Database memungkinkan program menyimpan data secara persisten sehingga tetap tersedia meskipun aplikasi dimatikan atau restart.

- **Akses Data yang Efisien**

Database dirancang untuk menyimpan, mengelola, dan mengakses data secara terstruktur dan efisien, memungkinkan operasi seperti pencarian, pembaruan, dan penghapusan data dengan mudah.

- **Mendukung Skalabilitas**

Database yang terintegrasi memungkinkan aplikasi menangani volume data yang besar dengan tetap menjaga performa.

- **Keamanan Data**

Database menyediakan fitur seperti kontrol akses, enkripsi, dan audit log untuk melindungi data dari akses yang tidak sah.

- **Konsistensi Data**

Database menjamin integritas data melalui mekanisme seperti transaksi, sehingga data tetap konsisten meskipun terjadi kegagalan sistem.

- **Kolaborasi Antar Komponen Aplikasi**

Integrasi database memungkinkan berbagai komponen aplikasi berbagi data dengan cara yang terstandarisasi.

2. Dalam konteks pengembangan REST API dengan Golang, jelaskan mengapa pemilihan struktur project (project structure) yang tepat itu penting dan sebutkan minimal 3 folder/package yang umumnya ada dalam struktur project Golang!

**Jawab:**

Pemilihan struktur proyek yang tepat dalam pengembangan REST API dengan Golang penting karena:

- **Kemudahan Pemeliharaan**

Struktur yang terorganisir mempermudah pengembang untuk menambahkan fitur baru, memperbaiki bug, atau menghapus komponen yang tidak diperlukan.

- **Keterbacaan Kode**

Struktur yang baik membuat kode lebih mudah dipahami oleh tim pengembang, termasuk anggota baru.

- **Mendukung Skalabilitas**

Struktur yang modular memungkinkan aplikasi dikembangkan untuk memenuhi kebutuhan yang lebih kompleks atau menambahkan fitur tanpa mempengaruhi komponen lain.

- **Fasilitasi Pengujian**

Dengan struktur yang baik, pengujian unit dan integrasi dapat dilakukan dengan lebih terisolasi dan efisien.

- **Kesesuaian dengan Konvensi Golang**

Golang memiliki filosofi penulisan kode yang sederhana dan minimalis. Struktur yang sesuai memudahkan adopsi framework atau library pihak ketiga.

Folder/Package Umum dalam Struktur Project Golang:

1) **cmd:**

Berisi entry point aplikasi, biasanya berupa file seperti main.go untuk menginisialisasi server REST API.

2) **internal:**

Berisi logika aplikasi dan kode yang tidak diekspos untuk digunakan di luar proyek. Contohnya, kode domain atau business logic.

3) **pkg:**

Berisi kode yang dapat digunakan ulang atau diimpor dalam proyek lain, seperti utility atau helper.

4) **config:**

Menyimpan konfigurasi aplikasi seperti variabel environment.

5) **models:**

Berisi definisi struktur data atau model yang digunakan untuk representasi database atau data API.

6) **routes:**

Mengatur routing untuk endpoint REST API.

7) **handlers** atau **controllers:**

Berisi fungsi untuk menangani request dan mengembalikan response.

8) **services:**

Berisi logika bisnis yang menghubungkan antara controllers dan repository.

9) **repository:**

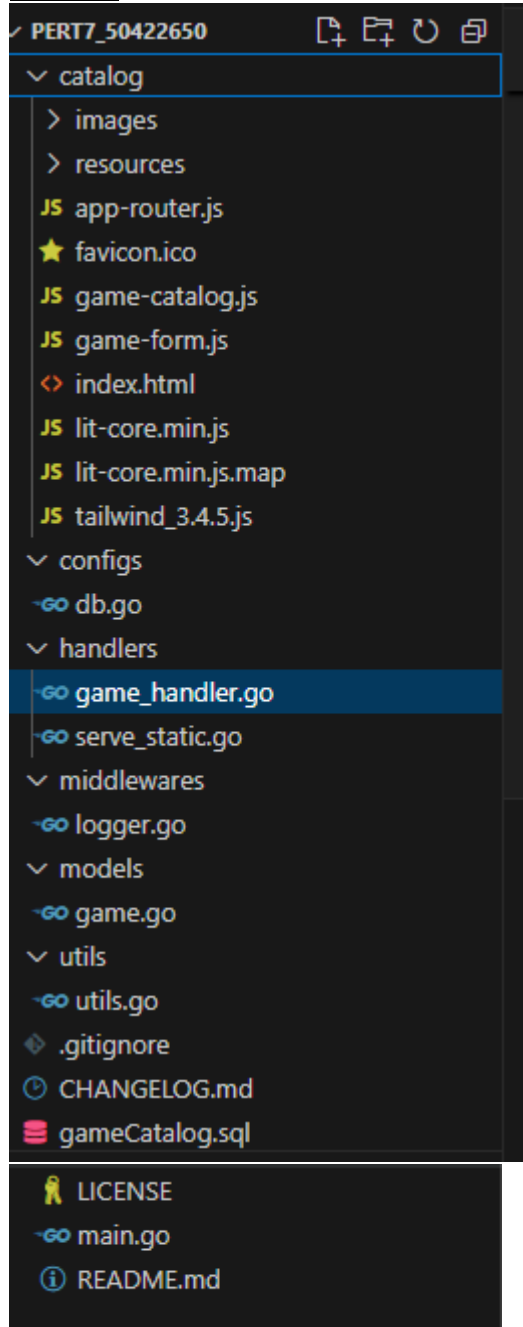
Berisi logika untuk berinteraksi dengan database, seperti query SQL atau ORM.

Struktur yang disesuaikan dengan kebutuhan memastikan aplikasi REST API dengan Golang dapat berkembang secara efisien.

## **Bagian 2**

1. Buatlah struktur direktori proyek seperti dibawah ini!

**Jawab:**



2. Pada file **main.go**, import library yang diperlukan dan buatlah function **main()** yang akan menginisialisasi dan menjalankan server pada port xxxx (empat digit terakhir npm). Fungsi ini harus dapat melakukan koneksi ke database, menangani kemungkinan kegagalan koneksi ke database, menyiapkan file server untuk konten statis dari folder 'catalog', serta mendaftarkan route yang diperlukan untuk API games. Pastikan juga untuk mengimplementasikan middleware logging untuk mencatat semua request yang masuk ke server. Perhatikan bahwa koneksi database harus ditutup dengan benar ketika server berhenti beroperasi!

**Jawab:**

```
GO main.go

package main

import (
    "fmt"
    "log"
    "net/http"
    "pert7_50422650/configs"
    "pert7_50422650/handlers"
    "pert7_50422650/middlewares"

    _ "github.com/go-sql-driver/mysql"
)

func main() {
    PORT := 2650 // 4 Digit Terakhir NPM

    configs.ConnectDB()
    if configs.DB == nil {
        log.Fatal("Database connection failed")
    }
    defer func() {
        if err := configs.DB.Close(); err != nil {
            log.Printf("Error closing database connection: %v", err)
        }
    }()
}
```

```

○ ○ ○ GO main.go

mux := http.NewServeMux()
fileServer := http.FileServer(http.Dir("catalog"))
mux.Handle("/", func(w http.ResponseWriter, r *http.Request) {
    handlers.ServeStaticFile(w, r, "catalog", fileServer)
})
mux.HandleFunc("/api/games/", handlers.HandleGames)
mux.HandleFunc("/api/games", handlers.HandleGames)

// Apply middleware
loggedMux := middlewares.LogRequestHandler(mux)

fmt.Printf("Server berjalan di http://localhost:%d\n", PORT)
log.Fatal(http.ListenAndServe(fmt.Sprintf(":%d", PORT), loggedMux))
}

```

3. Pada file **configs/db.go**, import library yang diperlukan dan buat fungsi ConnectDB() yang akan menginisialisasi koneksi database

**Jawab:**

```

○ ○ ○ GO db.go

package configs

import (
    "database/sql"
    "log"
    "time"

    _ "github.com/go-sql-driver/mysql"
)

var DB *sql.DB

func ConnectDB() {
    var err error

    DB, err = sql.Open("mysql", "root@tcp(127.0.0.1:3306)/steam_catalog")
    if err != nil {
        log.Fatal("Failed to open database connection:", err)
    }

    DB.SetMaxOpenConns(10)
    DB.SetMaxIdleConns(5)
    DB.SetConnMaxLifetime(time.Minute * 5)
}

```

```
GO db.go

err = DB.Ping()
if err != nil {
    log.Fatal("Failed to ping database:", err)
}

_, err = DB.Query("SELECT 1 FROM games LIMIT 1")
if err != nil {
    log.Fatal("Failed to query games table:", err)
}

log.Println("Successfully connected to database")
}
```

4. Pada file **models/game.go**, buatlah struct Game dengan lima field: ID, Nama, Developer, Genre, dan Harga

**Jawab:**

```
GO game.go

package models

type Game struct {
    ID      int    `json:"id_game"`
    Nama    string `json:"nama_game"`
    Developer string `json:"developer"`
    Genre   string `json:"genre"`
    Harga   float64 `json:"harga"`
}
```

5. Pada file **utils/utils.go**, buatlah fungsi RespondJSON untuk mengirim response dalam format JSON. Fungsi ini menerima `http.ResponseWriter` dan data `interface{}` sebagai parameter

**Jawab:**

```
utils.go

package utils

import (
    "encoding/json"
    "net/http"
)

func RespondJSON(w http.ResponseWriter, data interface{}) {
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(data)
}
```

6. Pada file **handlers/serve\_static.go**, buatlah fungsi ServeStaticFile untuk melayani file statis dari folder catalog. Fungsi ini harus dapat menangani path URL, melakukan pengecekan file, dan mengarahkan kembali ke halaman utama jika file tidak ditemukan. Pastikan request ke path `/api/` tidak dilayani oleh file server

**Jawab:**

```
serve_static.go

package handlers

import (
    "net/http"
    "os"
    "path/filepath"
    "strings"
)

func ServeStaticFile(w http.ResponseWriter, r *http.Request, baseDir string, fileServer
http.Handler) {
    path := strings.TrimPrefix(r.URL.Path, "/")
    fullPath := filepath.Join(baseDir, path)

    if strings.HasPrefix(r.URL.Path, "/api/") {
        http.NotFound(w, r)
        return
    }

    _, err := os.Stat(fullPath)
    if err != nil {
        http.Redirect(w, r, "/", http.StatusTemporaryRedirect)
        return
    }
    fileServer.ServeHTTP(w, r)
}
```



7. Pada file **handlers/game\_handler.go**, buatlah API endpoint untuk operasi CRUD pada data game. Buatlah fungsi utama `HandleGames` yang akan mengarahkan request ke fungsi yang sesuai berdasarkan method HTTP. Implementasikan juga fungsi `handleGetGames` untuk mengambil data game (baik semua game maupun game spesifik), `handlePostGame` untuk menambah game baru, `handleUpdateGame` untuk memperbarui data game, dan `handleDeleteGame` untuk menghapus game. Pastikan setiap fungsi menangani error dengan tepat dan memberikan response yang sesuai

**Jawab:**

```
package handlers

import (
    "database/sql"
    "encoding/json"
    "net/http"
    "pert7_50422650/configs"
    "pert7_50422650/models"
    "pert7_50422650/utils"
    "strconv"
)

func HandleGames(w http.ResponseWriter, r *http.Request) {
    path := r.URL.Path
    var id int
    var err error

    if len(path) > len("/api/games/") {
        id, err = strconv.Atoi(path[len("/api/games/"):])
        if err != nil {
            http.Error(w, "ID tidak valid", http.StatusBadRequest)
            return
        }
    } else {
        id = 0
    }

    switch r.Method {
    case http.MethodGet:
        handleGetGames(w, id)
    case http.MethodPost:
        handlePostGame(w, r)
    case http.MethodPut:
        handleUpdateGame(w, r, id)
    case http.MethodDelete:
```

```

        handleDeleteGame(w, id)
    }
}

func handleGetGames(w http.ResponseWriter, id int) {
    if id == 0 {
        rows, err := configs.DB.Query("SELECT id_game, nama_game, developer,
genre, harga FROM games")
        if err != nil {
            http.Error(w, "Gagal mengambil data game",
http.StatusInternalServerError)
            return
        }
        defer rows.Close()

        var games []models.Game
        for rows.Next() {
            var game models.Game
            err := rows.Scan(&game.ID, &game>Nama, &game.Developer, &game.Genre,
&game.Harga)
            if err != nil {
                http.Error(w, "Gagal memproses data game",
http.StatusInternalServerError)
                return
            }
            games = append(games, game)
        }
        utils.RespondJSON(w, games)
    } else {
        var game models.Game
        err := configs.DB.QueryRow("SELECT id_game, nama_game, developer, genre,
harga FROM games WHERE id_game = ?", id).
            Scan(&game.ID, &game>Nama, &game.Developer, &game.Genre, &game.Harga)
        if err != nil {
            if err == sql.ErrNoRows {
                http.Error(w, "Game tidak ditemukan", http.StatusNotFound)
            } else {
                http.Error(w, "Gagal mengambil data game",
http.StatusInternalServerError)
            }
            return
        }
        utils.RespondJSON(w, game)
    }
}
}

```

```

func handlePostGame(w http.ResponseWriter, r *http.Request) {
    var game models.Game
    if err := json.NewDecoder(r.Body).Decode(&game); err != nil {
        utils.RespondJSON(w, map[string]string{"error": "Data tidak valid"})
        return
    }

    _, err := configs.DB.Exec(
        "INSERT INTO games (nama_game, developer, genre, harga) VALUES (?, ?, ?, ?)",
        game>Nama, game.Developer, game.Genre, game.Harga)

    if err != nil {
        // This will be caught by your LogRequestHandler middleware
        utils.RespondJSON(w, map[string]string{"error": err.Error()})
        return
    }

    utils.RespondJSON(w, map[string]string{"message": "Game berhasil
ditambahkan!"})
}

func handleUpdateGame(w http.ResponseWriter, r *http.Request, id int) {
    var game models.Game
    if err := json.NewDecoder(r.Body).Decode(&game); err != nil {
        http.Error(w, "Data tidak valid", http.StatusBadRequest)
        return
    }
    result, err := configs.DB.Exec("UPDATE games SET nama_game=?, developer=?,
genre=?, harga=? WHERE id_game=?",
    game>Nama, game.Developer, game.Genre, game.Harga, id)
    if err != nil {
        http.Error(w, "Gagal memperbarui game", http.StatusInternalServerError)
        return
    }
    rowsAffected, _ := result.RowsAffected()
    if rowsAffected == 0 {
        http.Error(w, "Game dengan ID tersebut tidak ditemukan",
http.StatusNotFound)
        return
    }
    utils.RespondJSON(w, map[string]string{"message": "Game berhasil
diperbarui!"})
}

```

```

func handleDeleteGame(w http.ResponseWriter, id int) {
    result, err := configs.DB.Exec("DELETE FROM games WHERE id_game=?", id)
    if err != nil {
        http.Error(w, "Gagal menghapus game", http.StatusInternalServerError)
        return
    }

    rowsAffected, _ := result.RowsAffected()
    if rowsAffected == 0 {
        http.Error(w, "Game dengan ID tersebut tidak ditemukan",
http.StatusNotFound)
        return
    }

    utils.RespondJSON(w, map[string]string{"message": "Game berhasil dihapus!"})
}

```

8. Pada file **middlewares/logger.go**, buatlah fungsi `LogRequestHandler` untuk mencatat setiap request yang masuk ke server. Fungsi ini harus menerima dan meneruskan request ke handler berikutnya sambil melakukan logging method dan path URL request

**Jawab:**

```

package middlewares

import (
    "log"
    "net/http"
)

func LogRequestHandler(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        log.Printf("Incoming request: %s %s", r.Method, r.URL.Path)
        next.ServeHTTP(w, r)
    })
}

```

## 9. Tampilan Maria

### Jawab:

```
XAMPP for Windows - mysql -u root

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> -- Membuat database steam_catalog
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS steam_catalog;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> -- Menggunakan database steam_catalog
MariaDB [(none)]> USE steam_catalog;
Database changed
MariaDB [steam_catalog]> -- Membuat tabel games
MariaDB [steam_catalog]> CREATE TABLE IF NOT EXISTS games (
  ->   id_game INT AUTO INCREMENT PRIMARY KEY,
  ->   nama_game VARCHAR(255) NOT NULL,
  ->   developer VARCHAR(255) NOT NULL,
  ->   genre VARCHAR(100) NOT NULL,
  ->   harga DECIMAL(10,2) NOT NULL
  -> );
Query OK, 0 rows affected (0.005 sec)

MariaDB [steam_catalog]> -- Menambahkan beberapa data contoh ke tabel games
MariaDB [steam_catalog]> INSERT INTO games (nama_game, developer, genre, harga) VALUES
  -> ('Cyber Battle', 'Cyber Studios', 'Action', 49.99),
  -> ('Fantasy Quest', 'DreamWorks Games', 'RPG', 59.99),
  -> ('Speed Racer', 'FastLane Devs', 'Racing', 29.99),
  -> ('Puzzle Mania', 'Brainy Games', 'Puzzle', 19.99),
  -> ('Space Adventure', 'Galactic Studio', 'Adventure', 39.99);
Query OK, 5 rows affected (0.038 sec)
Records: 5  Duplicates: 0  Warnings: 0

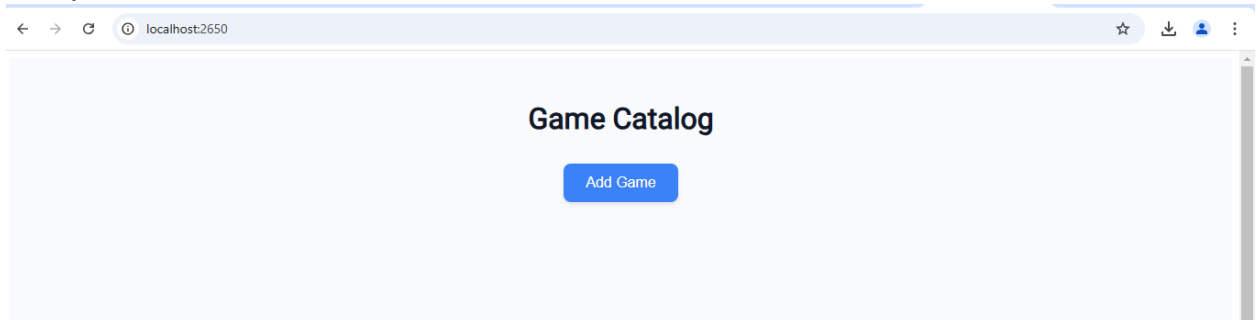
MariaDB [steam_catalog]> -- Menampilkan data dari tabel games
MariaDB [steam_catalog]> SELECT * FROM games;
+----+-----+-----+-----+-----+
| id_game | nama_game | developer | genre | harga |
+----+-----+-----+-----+-----+
| 1 | Cyber Battle | Cyber Studios | Action | 49.99 |
| 2 | Fantasy Quest | DreamWorks Games | RPG | 59.99 |
| 3 | Speed Racer | FastLane Devs | Racing | 29.99 |
| 4 | Puzzle Mania | Brainy Games | Puzzle | 19.99 |
| 5 | Space Adventure | Galactic Studio | Adventure | 39.99 |
+----+-----+-----+-----+-----+
5 rows in set (0.000 sec)

MariaDB [steam_catalog]>
```

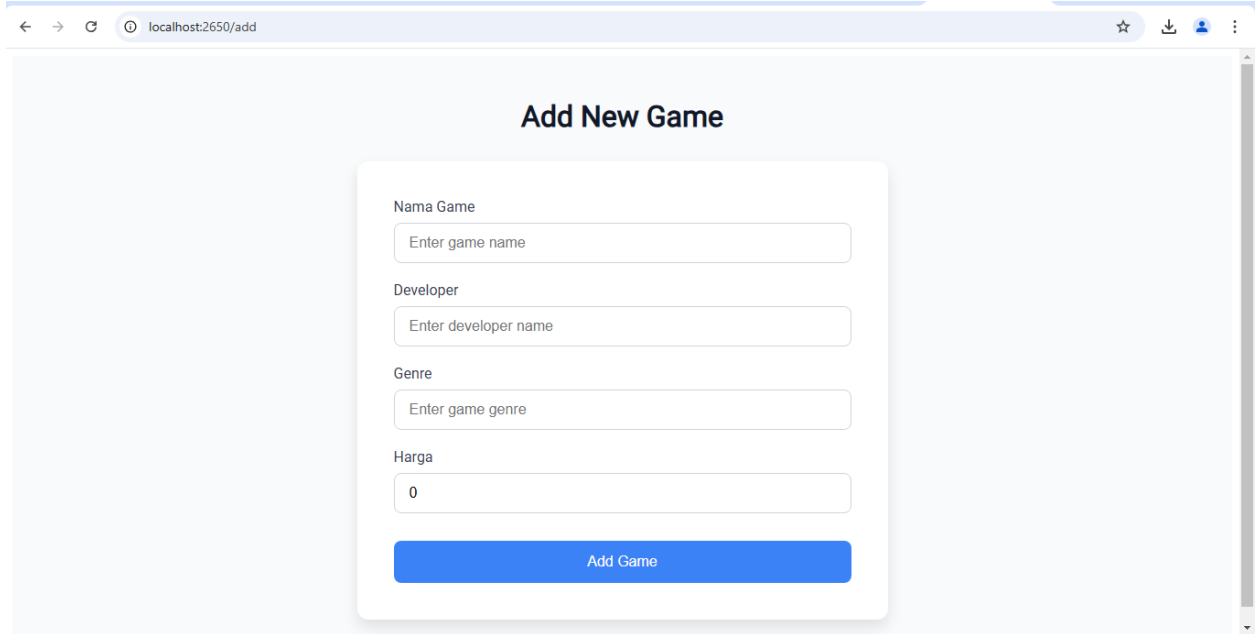
## 10. Tampilkan Outputnya

### Jawab:

- Tampilan Default

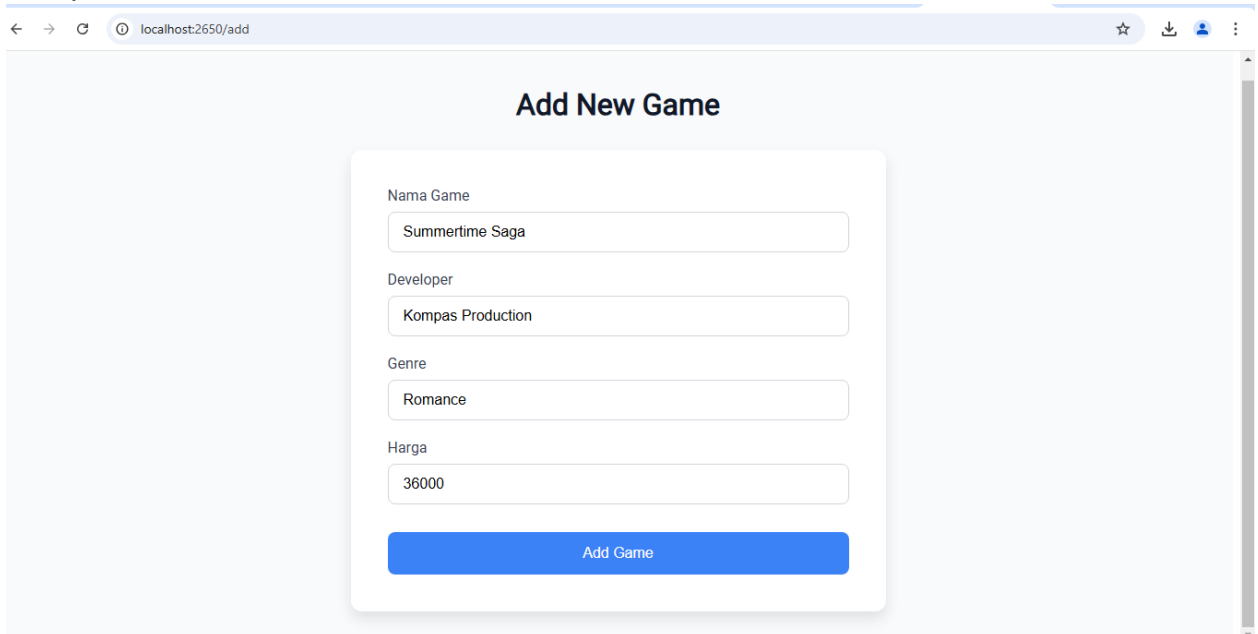


- Tampilan saat mencoba Menambahkan Game



A screenshot of a web browser showing a form titled "Add New Game". The form is centered on a light gray background. It contains four input fields: "Nama Game" with placeholder text "Enter game name", "Developer" with placeholder text "Enter developer name", "Genre" with placeholder text "Enter game genre", and "Harga" with the value "0". Below the input fields is a blue button labeled "Add Game". The browser's address bar shows "localhost:2650/add".

- Tampilan saat Data terisi



A screenshot of the same "Add New Game" form, but now the input fields are filled with data. The "Nama Game" field contains "Summertime Saga", the "Developer" field contains "Kompas Production", the "Genre" field contains "Romance", and the "Harga" field contains "36000". The blue "Add Game" button remains at the bottom. The browser's address bar still shows "localhost:2650/add".

Ada kesalahan dimana Button **Add Game** tidak bisa memproses ke tahap selanjutnya dan bukan kesalahan saya ya :v