**Laporan Tugas Akhir Pemrograman Berbasis Objek**

**A logo with a globe and a planet

Description automatically generated**

Oleh:

**Fahmi Abdurrahman**

**5230411318**

**Kelas F / IX**

**Dosen Pengampu : Muhammad Henri Syuhada S.T.,M.T**

**Program Studi Informatika**

**Fakultas Sains dan Teknologi**

**Universitas Teknologi Yogyakarta**

**2024**

**UML Class Diagram**

Gambar diatas menunjukkan bahwa:

1. Satu reservasi melibatkan satu tamu dan satu kamar.

Di dalam tabel Reservasi mengandung atribut id\_reservasi (Primary Key), id\_tamu (Foreign Key), id\_kamar (Foreign Key), serta atribut tambahan seperti check\_in, check\_out, dan total\_harga. Tabel ini juga memiliki fungsi-fungsi utama seperti Validasi, tambah\_reservasi, ubah\_reservasi, dan hapus\_reservasi.

1. Satu kamar dapat digunakan oleh banyak reservasi.

Menunjukkan bahwa kamar memiliki hubungan banyak ke satu dengan reservasi. Atribut yang terdapat pada tabel Kamar meliputi id\_kamar (Primary Key), tipe\_kamar, harga\_per\_malam, dan status. Fungsi utama untuk pengelolaan data kamar meliputi tambah\_kamar, ubah\_kamar, dan hapus\_kamar.

1. Satu tamu dapat melakukan banyak reservasi.

Tabel Tamu mengandung atribut seperti id\_tamu (Primary Key), nama\_tamu, dan kontak. Fungsi utama dalam tabel ini adalah untuk pengelolaan tamu, seperti tambah\_tamu, ubah\_tamu, dan hapus\_tamu.

**Screenshot Aplikasi**

**Query Database**

Berikut penjelasan dari gambar tersebut:

a. Tabel kamar

* Gambar menunjukkan bahwa tabel kamar memiliki struktur dengan kolom:
  + id\_kamar sebagai *Primary Key* untuk mengidentifikasi setiap kamar secara unik.
  + tipe\_kamar menyimpan jenis kamar dengan tipe data VARCHAR (panjang maksimum 50 karakter).
  + harga\_per\_malam menyimpan biaya kamar per malam dengan tipe data NUMERIC (10 digit total dengan 2 digit desimal).
  + status menunjukkan status kamar (misalnya, "tersedia" atau "terisi") dengan tipe data VARCHAR (panjang maksimum 20 karakter).

b. Tabel tamu

* Tabel tamu memiliki struktur dengan kolom:
  + id\_tamu sebagai *Primary Key* untuk mengidentifikasi tamu secara unik.
  + nama\_tamu menyimpan nama tamu dengan tipe data VARCHAR (panjang maksimum 100 karakter).
  + kontak menyimpan nomor kontak tamu dengan tipe data VARCHAR (panjang maksimum 20 karakter).

c. Tabel reservasi

* Tabel reservasi memiliki struktur dengan kolom:
  + id\_reservasi sebagai *Primary Key* untuk mengidentifikasi reservasi secara unik.
  + id\_tamu sebagai *Foreign Key* yang merujuk ke id\_tamu di tabel tamu, menandakan bahwa setiap reservasi dilakukan oleh tamu tertentu.
  + id\_kamar sebagai *Foreign Key* yang merujuk ke id\_kamar di tabel kamar, menunjukkan kamar yang dipesan.
  + check\_in dan check\_out menyimpan tanggal masuk dan keluar dengan tipe data DATE.
  + total\_harga menyimpan total biaya reservasi dengan tipe data NUMERIC (10 digit total dengan 2 digit desimal).

d. Relasi antar tabel

* Gambar ini menunjukkan hubungan antar entitas:
  + Satu tamu dapat memiliki beberapa reservasi.
  + Satu reservasi terkait dengan satu kamar tertentu, dan kamar tersebut hanya dapat dipesan oleh satu reservasi pada satu waktu tertentu.

**Source Code dan Penjelasannya**

**main.py**

from kivy.app import App

from kivy.uix.button import Button

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.label import Label  # Impor Label

from kamar import KamarApp

from tamu import TamuApp

from reservasi import ReservasiApp

class MainApp(App):

    def build(self):

        self.layout = BoxLayout(orientation='vertical')

        # Label header

        self.header = Label(text="Menu Utama", size\_hint=(1, 0.1))

        self.layout.add\_widget(self.header)

        # Tombol-tombol menu utama

        self.kelola\_kamar\_button = Button(text="Kelola Kamar", size\_hint=(1, 0.3), on\_press=self.open\_kamar)

        self.kelola\_tamu\_button = Button(text="Kelola Tamu", size\_hint=(1, 0.3), on\_press=self.open\_tamu)

        self.kelola\_reservasi\_button = Button(text="Kelola Reservasi", size\_hint=(1, 0.3), on\_press=self.open\_reservasi)

        self.layout.add\_widget(self.kelola\_kamar\_button)

        self.layout.add\_widget(self.kelola\_tamu\_button)

        self.layout.add\_widget(self.kelola\_reservasi\_button)

        return self.layout

    def open\_kamar(self, instance):

        self.stop()  # Menutup aplikasi utama

        KamarApp().run()  # Membuka aplikasi Kelola Kamar

    def open\_tamu(self, instance):

        self.stop()  # Menutup aplikasi utama

        TamuApp().run()  # Membuka aplikasi Kelola Tamu

    def open\_reservasi(self, instance):

        self.stop()  # Menutup aplikasi utama

        ReservasiApp().run()  # Membuka aplikasi Kelola Reservasi

if \_\_name\_\_ == "\_\_main\_\_":

    MainApp().run()

Penjelasan Kode main.py:

1. Impor Modul
   * kivy.app, kivy.uix.button, kivy.uix.boxlayout, dan kivy.uix.label digunakan untuk membangun antarmuka aplikasi Kivy.
   * KamarApp, TamuApp, dan ReservasiApp diimpor dari file terpisah untuk memuat fitur kelola kamar, tamu, dan reservasi.
2. Kelas MainApp
   * build Method:
     + Membuat layout utama dengan orientasi vertikal menggunakan BoxLayout.
     + Menambahkan label header ("Menu Utama") dan tombol-tombol untuk navigasi ke menu "Kelola Kamar", "Kelola Tamu", dan "Kelola Reservasi".
   * Fungsi Navigasi:
     + open\_kamar, open\_tamu, dan open\_reservasi digunakan untuk menghentikan aplikasi utama (self.stop()) dan membuka aplikasi sesuai menu yang dipilih (KamarApp, TamuApp, atau ReservasiApp).
3. Eksekusi Aplikasi
   * Program dijalankan dengan memanggil MainApp().run(), yang memulai tampilan menu utama aplikasi.

**kamar.py**

import psycopg2

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.label import Label

from kivy.uix.button import Button

from kivy.uix.scrollview import ScrollView

from kivy.uix.gridlayout import GridLayout

from eksporexcel import export\_kamar\_to\_excel  # Import fungsi ekspor dari eksporexcel.py

# Koneksi ke database PostgreSQL

def connect\_db():

    try:

        conn = psycopg2.connect(

            dbname="reservasihotelfahmi", user="postgres", password="1qaz2wsx", host="localhost", port="5432"

        )

        return conn

    except Exception as e:

        print(f"Error connecting to database: {e}")

        return None

class KamarApp(App):

    def build(self):

        self.conn = connect\_db()

        if not self.conn:

            print("Database connection failed.")

            return

        self.cursor = self.conn.cursor()

        self.layout = BoxLayout(orientation='vertical')

        # Tombol kembali ke menu utama

        self.back\_button = Button(text="Kembali ke Menu Utama", size\_hint=(1, 0.1), on\_press=self.back\_to\_main)

        self.layout.add\_widget(self.back\_button)

        # Tombol untuk ekspor ke Excel

        self.export\_button = Button(text="Ekspor ke Excel", size\_hint=(1, 0.1), on\_press=self.export\_to\_excel)

        self.layout.add\_widget(self.export\_button)

        # Input form untuk operasi CRUD

        self.input\_layout = BoxLayout(orientation='horizontal', size\_hint=(1, None), height=50)

        self.input\_label\_id = Label(text="ID Kamar", size\_hint=(0.3, 1))

        self.input\_text\_id = TextInput(size\_hint=(0.7, 1), hint\_text="auto jika tambah")

        self.input\_label\_type = Label(text="Tipe Kamar", size\_hint=(0.3, 1))

        self.input\_text\_type = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_price = Label(text="Harga Per Malam", size\_hint=(0.4, 1))

        self.input\_text\_price = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_status = Label(text="Status", size\_hint=(0.3, 1))

        self.input\_text\_status = TextInput(size\_hint=(0.7, 1))

        self.input\_layout.add\_widget(self.input\_label\_id)

        self.input\_layout.add\_widget(self.input\_text\_id)

        self.input\_layout.add\_widget(self.input\_label\_type)

        self.input\_layout.add\_widget(self.input\_text\_type)

        self.input\_layout.add\_widget(self.input\_label\_price)

        self.input\_layout.add\_widget(self.input\_text\_price)

        self.input\_layout.add\_widget(self.input\_label\_status)

        self.input\_layout.add\_widget(self.input\_text\_status)

        # Tombol CRUD

        self.add\_button = Button(text="Tambah Kamar", on\_press=self.add\_kamar)

        self.update\_button = Button(text="Perbarui Kamar", on\_press=self.update\_kamar)

        self.delete\_button = Button(text="Hapus Kamar", on\_press=self.delete\_kamar)

        self.layout.add\_widget(self.input\_layout)

        self.layout.add\_widget(self.add\_button)

        self.layout.add\_widget(self.update\_button)

        self.layout.add\_widget(self.delete\_button)

        # Menampilkan daftar data kamar

        self.data\_layout = GridLayout(cols=1, size\_hint\_y=None)

        self.data\_layout.bind(minimum\_height=self.data\_layout.setter('height'))

        self.scroll\_view = ScrollView()

        self.scroll\_view.add\_widget(self.data\_layout)

        self.layout.add\_widget(self.scroll\_view)

        self.load\_data()

        return self.layout

    def load\_data(self):

        try:

            self.cursor.execute("SELECT \* FROM kamar")

            rows = self.cursor.fetchall()

            self.data\_layout.clear\_widgets()

            for row in rows:

                item = Button(text=f"ID: {row[0]} - Tipe: {row[1]} - Harga: {row[2]} - Status: {row[3]}")

                self.data\_layout.add\_widget(item)

        except Exception as e:

            print(f"Error loading data: {e}")

    def add\_kamar(self, instance):

        tipe = self.input\_text\_type.text

        harga = self.input\_text\_price.text

        status = self.input\_text\_status.text

        if tipe and harga and status:

            try:

                # Menambahkan kamar dengan tipe, harga, dan status

                self.cursor.execute("INSERT INTO kamar (tipe\_kamar, harga\_per\_malam, status) VALUES (%s, %s, %s)",

                                    (tipe, harga, status))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_type.text = ''

                self.input\_text\_price.text = ''

                self.input\_text\_status.text = ''

            except Exception as e:

                print(f"Error adding kamar: {e}")

        else:

            print("Tipe Kamar, Harga Per Malam, dan Status tidak boleh kosong")

    def update\_kamar(self, instance):

        id\_kamar = self.input\_text\_id.text

        tipe = self.input\_text\_type.text

        harga = self.input\_text\_price.text

        status = self.input\_text\_status.text

        if id\_kamar and tipe and harga and status:

            try:

                # Memperbarui tipe\_kamar, harga\_per\_malam, dan status berdasarkan id\_kamar

                self.cursor.execute("UPDATE kamar SET tipe\_kamar = %s, harga\_per\_malam = %s, status = %s WHERE id\_kamar = %s",

                                    (tipe, harga, status, id\_kamar))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_id.text = ''

                self.input\_text\_type.text = ''

                self.input\_text\_price.text = ''

                self.input\_text\_status.text = ''

            except Exception as e:

                print(f"Error updating kamar: {e}")

        else:

            print("ID Kamar, Tipe Kamar, Harga Per Malam, dan Status tidak boleh kosong")

    def delete\_kamar(self, instance):

        id\_kamar = self.input\_text\_id.text

        if id\_kamar:

            try:

                self.cursor.execute("DELETE FROM kamar WHERE id\_kamar = %s", (id\_kamar,))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_id.text = ''

                self.input\_text\_type.text = ''

                self.input\_text\_price.text = ''

                self.input\_text\_status.text = ''

            except Exception as e:

                print(f"Error deleting kamar: {e}")

        else:

            print("ID Kamar tidak boleh kosong")

    def back\_to\_main(self, instance):

        from main import MainApp  # Import MainApp di dalam fungsi untuk menghindari circular import

        self.stop()  # Menutup aplikasi Kelola Kamar

        MainApp().run()  # Kembali ke aplikasi utama

    def export\_to\_excel(self, instance):

        export\_kamar\_to\_excel()  # Panggil fungsi ekspor dari eksporexcel.py

if \_\_name\_\_ == "\_\_main\_\_":

    KamarApp().run()

Penjelasan Kode kamar.py:

1. Impor Modul
   * psycopg2: Untuk menghubungkan aplikasi ke database PostgreSQL.
   * kivy (BoxLayout, TextInput, Label, Button, ScrollView, GridLayout): Untuk membangun antarmuka aplikasi.
   * export\_kamar\_to\_excel: Fungsi untuk mengekspor data kamar ke file Excel.
2. Fungsi connect\_db
   * Membuat koneksi ke database PostgreSQL dengan parameter dbname, user, password, host, dan port.
   * Mengembalikan objek koneksi jika berhasil, atau mencetak pesan error jika gagal.
3. Kelas KamarApp
   * build Method:
     + Membuat layout utama dengan tombol "Kembali ke Menu Utama" dan "Ekspor ke Excel".
     + Membuat input form untuk menambah, memperbarui, dan menghapus data kamar, termasuk kolom ID Kamar, Tipe Kamar, Harga Per Malam, dan Status.
     + Membuat daftar data kamar menggunakan ScrollView dan GridLayout.
   * Fungsi load\_data:
     + Mengambil data kamar dari database dan menampilkan setiap kamar dalam daftar dengan format tombol.
4. Operasi CRUD
   * add\_kamar:
     + Menambah data kamar ke database dengan INSERT INTO.
     + Membersihkan input setelah data ditambahkan.
   * update\_kamar:
     + Memperbarui data kamar berdasarkan ID Kamar dengan UPDATE.
     + Membersihkan input setelah data diperbarui.
   * delete\_kamar:
     + Menghapus data kamar berdasarkan ID Kamar dengan DELETE.
     + Membersihkan input setelah data dihapus.
5. Navigasi ke Menu Utama
   * back\_to\_main:
     + Menghentikan aplikasi Kelola Kamar dan kembali ke aplikasi utama (MainApp).
6. Ekspor ke Excel
   * export\_to\_excel:
     + Memanggil fungsi export\_kamar\_to\_excel dari file eksporexcel.py untuk mengekspor data kamar ke file Excel.
7. Eksekusi Aplikasi
   * Program dijalankan dengan memanggil KamarApp().run(), yang memulai antarmuka untuk mengelola data kamar.

**tamu.py**

import psycopg2

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.label import Label

from kivy.uix.button import Button

from kivy.uix.scrollview import ScrollView

from kivy.uix.gridlayout import GridLayout

from eksporexcel import export\_tamu\_to\_excel

# Koneksi ke database PostgreSQL

def connect\_db():

    try:

        conn = psycopg2.connect(

            dbname="reservasihotelfahmi", user="postgres", password="1qaz2wsx", host="localhost", port="5432"

        )

        return conn

    except Exception as e:

        print(f"Error connecting to database: {e}")

        return None

class TamuApp(App):

    def build(self):

        self.conn = connect\_db()

        if not self.conn:

            print("Database connection failed.")

            return

        self.cursor = self.conn.cursor()

        self.layout = BoxLayout(orientation='vertical')

        # Tombol kembali ke menu utama

        self.back\_button = Button(text="Kembali ke Menu Utama", size\_hint=(1, 0.1), on\_press=self.back\_to\_main)

        self.layout.add\_widget(self.back\_button)

        # Tombol untuk ekspor ke Excel

        self.export\_button = Button(text="Ekspor ke Excel", size\_hint=(1, 0.1), on\_press=self.export\_to\_excel)

        self.layout.add\_widget(self.export\_button)

        # Input form untuk operasi CRUD

        self.input\_layout = BoxLayout(orientation='horizontal', size\_hint=(1, None), height=50)

        self.input\_label\_id = Label(text="ID Tamu", size\_hint=(0.3, 1))

        self.input\_text\_id = TextInput(size\_hint=(0.7, 1), hint\_text="auto jika tambah")

        self.input\_label\_name = Label(text="Nama Tamu", size\_hint=(0.3, 1))

        self.input\_text\_name = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_contact = Label(text="Kontak Tamu", size\_hint=(0.3, 1))

        self.input\_text\_contact = TextInput(size\_hint=(0.7, 1))

        self.input\_layout.add\_widget(self.input\_label\_id)

        self.input\_layout.add\_widget(self.input\_text\_id)

        self.input\_layout.add\_widget(self.input\_label\_name)

        self.input\_layout.add\_widget(self.input\_text\_name)

        self.input\_layout.add\_widget(self.input\_label\_contact)

        self.input\_layout.add\_widget(self.input\_text\_contact)

        # Tombol CRUD

        self.add\_button = Button(text="Tambah Tamu", on\_press=self.add\_tamu)

        self.update\_button = Button(text="Perbarui Tamu", on\_press=self.update\_tamu)

        self.delete\_button = Button(text="Hapus Tamu", on\_press=self.delete\_tamu)

        self.layout.add\_widget(self.input\_layout)

        self.layout.add\_widget(self.add\_button)

        self.layout.add\_widget(self.update\_button)

        self.layout.add\_widget(self.delete\_button)

        # Menampilkan daftar data

        self.data\_layout = GridLayout(cols=1, size\_hint\_y=None)

        self.data\_layout.bind(minimum\_height=self.data\_layout.setter('height'))

        self.scroll\_view = ScrollView()

        self.scroll\_view.add\_widget(self.data\_layout)

        self.layout.add\_widget(self.scroll\_view)

        self.load\_data()

        return self.layout

    def load\_data(self):

        try:

            self.cursor.execute("SELECT \* FROM tamu")

            rows = self.cursor.fetchall()

            self.data\_layout.clear\_widgets()

            for row in rows:

                item = Button(text=f"ID: {row[0]} - Nama: {row[1]} - Kontak: {row[2]}")

                self.data\_layout.add\_widget(item)

        except Exception as e:

            print(f"Error loading data: {e}")

    def add\_tamu(self, instance):

        name = self.input\_text\_name.text

        contact = self.input\_text\_contact.text

        if name and contact:

            try:

                # Menambahkan tamu dengan nama dan kontak

                self.cursor.execute("INSERT INTO tamu (nama\_tamu, kontak) VALUES (%s, %s)", (name, contact))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_name.text = ''

                self.input\_text\_contact.text = ''

            except Exception as e:

                print(f"Error adding tamu: {e}")

        else:

            print("Nama Tamu dan Kontak tidak boleh kosong")

    def update\_tamu(self, instance):

        id\_tamu = self.input\_text\_id.text

        name = self.input\_text\_name.text

        contact = self.input\_text\_contact.text

        if id\_tamu and name and contact:

            try:

                # Memperbarui nama\_tamu dan kontak berdasarkan id\_tamu

                self.cursor.execute("UPDATE tamu SET nama\_tamu = %s, kontak = %s WHERE id\_tamu = %s",

                                    (name, contact, id\_tamu))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_id.text = ''

                self.input\_text\_name.text = ''

                self.input\_text\_contact.text = ''

            except Exception as e:

                print(f"Error updating tamu: {e}")

        else:

            print("ID Tamu, Nama Tamu, dan Kontak tidak boleh kosong")

    def delete\_tamu(self, instance):

        name = self.input\_text\_name.text

        if name:

            try:

                self.cursor.execute("DELETE FROM tamu WHERE nama\_tamu = %s", (name,))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_name.text = ''

                self.input\_text\_contact.text = ''

            except Exception as e:

                print(f"Error deleting tamu: {e}")

        else:

            print("Nama Tamu tidak boleh kosong")

    def back\_to\_main(self, instance):

        from main import MainApp  # Import MainApp di dalam fungsi untuk menghindari circular import

        self.stop()  # Menutup aplikasi Kelola Tamu

        MainApp().run()  # Kembali ke aplikasi utama

    def export\_to\_excel(self, instance):

        export\_tamu\_to\_excel()  # Panggil fungsi ekspor dari eksporexcel.py

if \_\_name\_\_ == "\_\_main\_\_":

    TamuApp().run()

Penjelasan Kode tamu.py:

1. Impor Modul
   * psycopg2: Untuk menghubungkan aplikasi ke database PostgreSQL.
   * kivy (BoxLayout, TextInput, Label, Button, ScrollView, GridLayout): Untuk membangun antarmuka aplikasi.
   * export\_tamu\_to\_excel: Fungsi untuk mengekspor data tamu ke file Excel.
2. Fungsi connect\_db
   * Membuat koneksi ke database PostgreSQL dengan parameter dbname, user, password, host, dan port.
   * Mengembalikan objek koneksi jika berhasil, atau mencetak pesan error jika gagal.
3. Kelas TamuApp
   * build Method:
     + Membuat layout utama dengan tombol "Kembali ke Menu Utama" dan "Ekspor ke Excel".
     + Membuat input form untuk menambah, memperbarui, dan menghapus data tamu, termasuk kolom ID Tamu, Nama Tamu, dan Kontak Tamu.
     + Membuat daftar data tamu menggunakan ScrollView dan GridLayout.
   * Fungsi load\_data:
     + Mengambil data tamu dari database dan menampilkan setiap tamu dalam daftar dengan format tombol.
4. Operasi CRUD
   * add\_tamu:
     + Menambah data tamu ke database dengan INSERT INTO.
     + Membersihkan input setelah data ditambahkan.
   * update\_tamu:
     + Memperbarui data tamu berdasarkan ID Tamu dengan UPDATE.
     + Membersihkan input setelah data diperbarui.
   * delete\_tamu:
     + Menghapus data tamu berdasarkan nama tamu dengan DELETE.
     + Membersihkan input setelah data dihapus.
5. Navigasi ke Menu Utama
   * back\_to\_main:
     + Menghentikan aplikasi Kelola Tamu dan kembali ke aplikasi utama (MainApp).
6. Ekspor ke Excel
   * export\_to\_excel:
     + Memanggil fungsi export\_tamu\_to\_excel dari file eksporexcel.py untuk mengekspor data tamu ke file Excel.
7. Eksekusi Aplikasi
   * Program dijalankan dengan memanggil TamuApp().run(), yang memulai antarmuka untuk mengelola data tamu.

**reservasi.py**

import psycopg2

from kivy.app import App

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.label import Label

from kivy.uix.button import Button

from kivy.uix.scrollview import ScrollView

from kivy.uix.gridlayout import GridLayout

from eksporexcel import export\_reservasi\_to\_excel  # Import fungsi ekspor dari eksporexcel.py

# Koneksi ke database PostgreSQL

def connect\_db():

    try:

        conn = psycopg2.connect(

            dbname="reservasihotelfahmi", user="postgres", password="1qaz2wsx", host="localhost", port="5432"

        )

        return conn

    except Exception as e:

        print(f"Error connecting to database: {e}")

        return None

class ReservasiApp(App):

    def build(self):

        self.conn = connect\_db()

        if not self.conn:

            print("Database connection failed.")

            return

        self.cursor = self.conn.cursor()

        self.layout = BoxLayout(orientation='vertical')

        # Tombol kembali ke menu utama

        self.back\_button = Button(text="Kembali ke Menu Utama", size\_hint=(1, 0.1), on\_press=self.back\_to\_main)

        self.layout.add\_widget(self.back\_button)

        # Tombol untuk ekspor ke Excel

        self.export\_button = Button(text="Ekspor ke Excel", size\_hint=(1, 0.1), on\_press=self.export\_to\_excel)

        self.layout.add\_widget(self.export\_button)

        # Input form untuk operasi CRUD

        self.input\_layout = BoxLayout(orientation='horizontal', size\_hint=(1, None), height=50)

        self.input\_label\_id = Label(text="ID Reservasi", size\_hint=(0.3, 1))

        self.input\_text\_id = TextInput(size\_hint=(0.7, 1), hint\_text="auto jika tambah")

        self.input\_label\_tamu = Label(text="ID Tamu", size\_hint=(0.3, 1))

        self.input\_text\_tamu = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_kamar = Label(text="ID Kamar", size\_hint=(0.3, 1))

        self.input\_text\_kamar = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_check\_in = Label(text="Check-in", size\_hint=(0.3, 1))

        self.input\_text\_check\_in = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_check\_out = Label(text="Check-out", size\_hint=(0.3, 1))

        self.input\_text\_check\_out = TextInput(size\_hint=(0.7, 1))

        self.input\_label\_total = Label(text="Total Harga", size\_hint=(0.3, 1))

        self.input\_text\_total = TextInput(size\_hint=(0.7, 1))

        self.input\_layout.add\_widget(self.input\_label\_id)

        self.input\_layout.add\_widget(self.input\_text\_id)

        self.input\_layout.add\_widget(self.input\_label\_tamu)

        self.input\_layout.add\_widget(self.input\_text\_tamu)

        self.input\_layout.add\_widget(self.input\_label\_kamar)

        self.input\_layout.add\_widget(self.input\_text\_kamar)

        self.input\_layout.add\_widget(self.input\_label\_check\_in)

        self.input\_layout.add\_widget(self.input\_text\_check\_in)

        self.input\_layout.add\_widget(self.input\_label\_check\_out)

        self.input\_layout.add\_widget(self.input\_text\_check\_out)

        self.input\_layout.add\_widget(self.input\_label\_total)

        self.input\_layout.add\_widget(self.input\_text\_total)

        # Tombol CRUD

        self.add\_button = Button(text="Tambah Reservasi", on\_press=self.add\_reservasi)

        self.update\_button = Button(text="Perbarui Reservasi", on\_press=self.update\_reservasi)

        self.delete\_button = Button(text="Hapus Reservasi", on\_press=self.delete\_reservasi)

        self.layout.add\_widget(self.input\_layout)

        self.layout.add\_widget(self.add\_button)

        self.layout.add\_widget(self.update\_button)

        self.layout.add\_widget(self.delete\_button)

        # Menampilkan daftar data reservasi

        self.data\_layout = GridLayout(cols=1, size\_hint\_y=None)

        self.data\_layout.bind(minimum\_height=self.data\_layout.setter('height'))

        self.scroll\_view = ScrollView()

        self.scroll\_view.add\_widget(self.data\_layout)

        self.layout.add\_widget(self.scroll\_view)

        self.load\_data()

        return self.layout

    def load\_data(self):

        try:

            self.cursor.execute("SELECT \* FROM reservasi")

            rows = self.cursor.fetchall()

            self.data\_layout.clear\_widgets()

            for row in rows:

                item = Button(text=f"ID: {row[0]} - Tamu: {row[1]} - Kamar: {row[2]} - Check-in: {row[3]} - Check-out: {row[4]} - Total: {row[5]}")

                self.data\_layout.add\_widget(item)

        except Exception as e:

            print(f"Error loading data: {e}")

    def add\_reservasi(self, instance):

        id\_tamu = self.input\_text\_tamu.text

        id\_kamar = self.input\_text\_kamar.text

        check\_in = self.input\_text\_check\_in.text

        check\_out = self.input\_text\_check\_out.text

        total = self.input\_text\_total.text

        if id\_tamu and id\_kamar and check\_in and check\_out and total:

            try:

                # Menambahkan reservasi

                self.cursor.execute("INSERT INTO reservasi (id\_tamu, id\_kamar, check\_in, check\_out, total\_harga) VALUES (%s, %s, %s, %s, %s)",

                                    (id\_tamu, id\_kamar, check\_in, check\_out, total))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_tamu.text = ''

                self.input\_text\_kamar.text = ''

                self.input\_text\_check\_in.text = ''

                self.input\_text\_check\_out.text = ''

                self.input\_text\_total.text = ''

            except Exception as e:

                print(f"Error adding reservasi: {e}")

        else:

            print("Semua field harus diisi")

    def update\_reservasi(self, instance):

        id\_reservasi = self.input\_text\_id.text

        id\_tamu = self.input\_text\_tamu.text

        id\_kamar = self.input\_text\_kamar.text

        check\_in = self.input\_text\_check\_in.text

        check\_out = self.input\_text\_check\_out.text

        total = self.input\_text\_total.text

        if id\_reservasi and id\_tamu and id\_kamar and check\_in and check\_out and total:

            try:

                # Memperbarui reservasi

                self.cursor.execute("UPDATE reservasi SET id\_tamu = %s, id\_kamar = %s, check\_in = %s, check\_out = %s, total\_harga = %s WHERE id\_reservasi = %s",

                                    (id\_tamu, id\_kamar, check\_in, check\_out, total, id\_reservasi))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_id.text = ''

                self.input\_text\_tamu.text = ''

                self.input\_text\_kamar.text = ''

                self.input\_text\_check\_in.text = ''

                self.input\_text\_check\_out.text = ''

                self.input\_text\_total.text = ''

            except Exception as e:

                print(f"Error updating reservasi: {e}")

        else:

            print("ID Reservasi, Tamu, Kamar, Check-in, Check-out, dan Total Harga tidak boleh kosong")

    def delete\_reservasi(self, instance):

        id\_reservasi = self.input\_text\_id.text

        if id\_reservasi:

            try:

                self.cursor.execute("DELETE FROM reservasi WHERE id\_reservasi = %s", (id\_reservasi,))

                self.conn.commit()

                self.load\_data()

                self.input\_text\_id.text = ''

                self.input\_text\_tamu.text = ''

                self.input\_text\_kamar.text = ''

                self.input\_text\_check\_in.text = ''

                self.input\_text\_check\_out.text = ''

                self.input\_text\_total.text = ''

            except Exception as e:

                print(f"Error deleting reservasi: {e}")

        else:

            print("ID Reservasi tidak boleh kosong")

    def back\_to\_main(self, instance):

        from main import MainApp  # Import MainApp di dalam fungsi untuk menghindari circular import

        self.stop()  # Menutup aplikasi Kelola Reservasi

        MainApp().run()  # Kembali ke aplikasi utama

    def export\_to\_excel(self, instance):

        export\_reservasi\_to\_excel()  # Panggil fungsi ekspor dari eksporexcel.py

if \_\_name\_\_ == "\_\_main\_\_":

    ReservasiApp().run()

Penjelasan Kode reservasi.py:

1. Impor Modul
   * psycopg2: Untuk menghubungkan aplikasi ke database PostgreSQL.
   * kivy (BoxLayout, TextInput, Label, Button, ScrollView, GridLayout): Untuk membangun antarmuka aplikasi.
   * export\_reservasi\_to\_excel: Fungsi untuk mengekspor data reservasi ke file Excel dari eksporexcel.py.
2. Fungsi connect\_db
   * Membuat koneksi ke database PostgreSQL dengan parameter dbname, user, password, host, dan port.
   * Mengembalikan objek koneksi jika berhasil, atau mencetak pesan error jika gagal.
3. Kelas ReservasiApp
   * build Method:
     + Membuat layout utama dengan tombol "Kembali ke Menu Utama" dan "Ekspor ke Excel".
     + Membuat input form untuk menambah, memperbarui, dan menghapus data reservasi, termasuk kolom ID Reservasi, ID Tamu, ID Kamar, Check-in, Check-out, dan Total Harga.
     + Membuat daftar data reservasi menggunakan ScrollView dan GridLayout.
   * Fungsi load\_data:
     + Mengambil data reservasi dari database dan menampilkan setiap reservasi dalam daftar dengan format tombol.
4. Operasi CRUD
   * add\_reservasi:
     + Menambah data reservasi ke database dengan INSERT INTO.
     + Membersihkan input setelah data ditambahkan.
   * update\_reservasi:
     + Memperbarui data reservasi berdasarkan ID Reservasi dengan UPDATE.
     + Membersihkan input setelah data diperbarui.
   * delete\_reservasi:
     + Menghapus data reservasi berdasarkan ID Reservasi dengan DELETE.
     + Membersihkan input setelah data dihapus.
5. Navigasi ke Menu Utama
   * back\_to\_main:
     + Menghentikan aplikasi Kelola Reservasi dan kembali ke aplikasi utama (MainApp).
6. Ekspor ke Excel
   * export\_to\_excel:
     + Memanggil fungsi export\_reservasi\_to\_excel dari file eksporexcel.py untuk mengekspor data reservasi ke file Excel.
7. Eksekusi Aplikasi
   * Program dijalankan dengan memanggil ReservasiApp().run(), yang memulai antarmuka untuk mengelola data reservasi.

**eksporexcel.py**

import psycopg2

from openpyxl import Workbook

from openpyxl.utils import get\_column\_letter

# Koneksi ke database PostgreSQL

def connect\_db():

    try:

        conn = psycopg2.connect(

            dbname="reservasihotelfahmi", user="postgres", password="1qaz2wsx", host="localhost", port="5432"

        )

        return conn

    except Exception as e:

        print(f"Error connecting to database: {e}")

        return None

# Fungsi untuk mengekspor data tamu ke Excel

def export\_tamu\_to\_excel():

    conn = connect\_db()

    if not conn:

        print("Database connection failed.")

        return

    cursor = conn.cursor()

    # Ambil data tamu dari database

    cursor.execute("SELECT \* FROM tamu")

    rows = cursor.fetchall()

    # Membuat workbook dan sheet baru

    wb = Workbook()

    ws = wb.active

    ws.title = "Tamu"

    # Menambahkan header ke sheet Excel

    columns = ['ID Tamu', 'Nama', 'Kontak']  # Sesuaikan dengan kolom di tabel tamu

    for col\_num, column\_title in enumerate(columns, 1):

        ws[get\_column\_letter(col\_num) + '1'] = column\_title

    # Menambahkan data tamu ke Excel

    for row\_num, row in enumerate(rows, 2):

        for col\_num, cell\_value in enumerate(row, 1):

            ws[get\_column\_letter(col\_num) + str(row\_num)] = cell\_value

    # Menyimpan file Excel

    wb.save("data\_tamu.xlsx")

    print("Data tamu berhasil diekspor ke data\_tamu.xlsx")

# Fungsi untuk mengekspor data kamar ke Excel

def export\_kamar\_to\_excel():

    conn = connect\_db()

    if not conn:

        print("Database connection failed.")

        return

    cursor = conn.cursor()

    # Ambil data kamar dari database

    cursor.execute("SELECT \* FROM kamar")

    rows = cursor.fetchall()

    # Membuat workbook dan sheet baru

    wb = Workbook()

    ws = wb.active

    ws.title = "Kamar"

    # Menambahkan header ke sheet Excel

    columns = ['ID Kamar', 'Tipe', 'Harga', 'Status']  # Sesuaikan dengan kolom di tabel kamar

    for col\_num, column\_title in enumerate(columns, 1):

        ws[get\_column\_letter(col\_num) + '1'] = column\_title

    # Menambahkan data kamar ke Excel

    for row\_num, row in enumerate(rows, 2):

        for col\_num, cell\_value in enumerate(row, 1):

            ws[get\_column\_letter(col\_num) + str(row\_num)] = cell\_value

    # Menyimpan file Excel

    wb.save("data\_kamar.xlsx")

    print("Data kamar berhasil diekspor ke data\_kamar.xlsx")

# Fungsi untuk mengekspor data reservasi ke Excel

def export\_reservasi\_to\_excel():

    conn = connect\_db()

    if not conn:

        print("Database connection failed.")

        return

    cursor = conn.cursor()

    # Ambil data reservasi dari database

    cursor.execute("SELECT \* FROM reservasi")

    rows = cursor.fetchall()

    # Membuat workbook dan sheet baru

    wb = Workbook()

    ws = wb.active

    ws.title = "Reservasi"

    # Menambahkan header ke sheet Excel

    columns = ['ID Reservasi', 'ID Tamu', 'ID Kamar', 'Check In', 'Check Out', 'Total Harga' ]  # Sesuaikan dengan kolom di tabel reservasi

    for col\_num, column\_title in enumerate(columns, 1):

        ws[get\_column\_letter(col\_num) + '1'] = column\_title

    # Menambahkan data reservasi ke Excel

    for row\_num, row in enumerate(rows, 2):

        for col\_num, cell\_value in enumerate(row, 1):

            ws[get\_column\_letter(col\_num) + str(row\_num)] = cell\_value

    # Menyimpan file Excel

    wb.save("data\_reservasi.xlsx")

    print("Data reservasi berhasil diekspor ke data\_reservasi.xlsx")

Penjelasan Kode eksporexcel.py:

1. Impor Modul
   * psycopg2: Untuk menghubungkan aplikasi ke database PostgreSQL.
   * openpyxl: Digunakan untuk membuat dan mengelola file Excel.
   * get\_column\_letter: Untuk mengonversi nomor kolom menjadi huruf kolom Excel.
2. Fungsi connect\_db
   * Membuat koneksi ke database PostgreSQL dengan parameter dbname, user, password, host, dan port.
   * Mengembalikan objek koneksi jika berhasil, atau mencetak pesan error jika gagal.
3. Fungsi export\_tamu\_to\_excel
   * Membuka koneksi ke database dan mengakses data tamu dari tabel tamu.
   * Membuat file Excel baru, menambahkan sheet dengan nama "Tamu", dan menambahkan header kolom.
   * Memasukkan data tamu yang diambil dari database ke dalam sheet Excel.
   * Menyimpan file Excel dengan nama data\_tamu.xlsx.
4. Fungsi export\_kamar\_to\_excel
   * Membuka koneksi ke database dan mengakses data kamar dari tabel kamar.
   * Membuat file Excel baru, menambahkan sheet dengan nama "Kamar", dan menambahkan header kolom.
   * Memasukkan data kamar yang diambil dari database ke dalam sheet Excel.
   * Menyimpan file Excel dengan nama data\_kamar.xlsx.
5. Fungsi export\_reservasi\_to\_excel
   * Membuka koneksi ke database dan mengakses data reservasi dari tabel reservasi.
   * Membuat file Excel baru, menambahkan sheet dengan nama "Reservasi", dan menambahkan header kolom.
   * Memasukkan data reservasi yang diambil dari database ke dalam sheet Excel.
   * Menyimpan file Excel dengan nama data\_reservasi.xlsx.
6. Eksekusi Fungsi
   * Setiap fungsi ekspor (export\_tamu\_to\_excel, export\_kamar\_to\_excel, export\_reservasi\_to\_excel) akan mengeksekusi proses pengambilan data dari database dan menyimpannya ke dalam file Excel yang sesuai.

**Output Ekspor Excel**

* Hasil output excel menu Kelola Kamar
* Hasil output excel Kelola Tamu
* Hasil output excel Kelola Reservasi