SISTEM INPUT DATA NAMA DAN GAJI KARYAWAN

OLEH

Rifqi Favian Hibatullah (2103015030)

Dafa Setyo Nugroho (2103015027)

DOSEN PENGAMPU

Ade Davy Wiranata S.Kom., M.Kom

UNIVERSITAS Dr.HAMKA

FAKULTAS TEKNIK INFORMATIKA

2022

KATA PENGANTAR

Segala puji bagi Allah SWT sebab karena limpahan rahmat serta anugerah dari-Nya kami dapat menyelesaikan tugas akhir kami dengan judul "Sistem Input Data Nama Dan Gaji Karyawan" ini.

Tidak lupa Shalawat dan salam selalu kita haturkan kepada Nabi Muhammad SAW yang telah menyampaikan petunjukan Allah SWT untuk kita semua sebagai umatnya, yang mana beliau menyampaikan Syariah agama Islam yang sempurna dan merupakan satu-satunya karunia paling besar bagi seluruh alam semesta.

Sistem input data karyawan dan gaji ini dibuat untuk memudahkan pengelolaan data karyawan dan penggajian di perusahaan. Dengan sistem ini, penginputan data karyawan dan penggajian akan lebih cepat dan akurat. Sistem ini juga memungkinkan pengelolaan laporan karyawan dan gaji secara mudah dan efisien. Kami berharap sistem ini dapat membantu perusahaan dalam mengelola data karyawan dan gaji dengan lebih baik.

Selanjutnya dengan kerendahan hati kami meminta kritik dan saran dari pembaca untuk topik ini supaya selanjutnya bisa kami revisi kembali. Karena penulis menyadari, bahwa tugas akhir yang telah kami buat ini masih memiliki banyak kekurangan.

Tidak lupa kami ucapkan terimakasih yang sebanyak-banyaknya kepada semua pihak yang telah mendukung serta membantu kami selama proses penyelesaian tugas akhir ini hingga rampungnya tugas akhir ini.

Demikianlah yang dapat kami sampaikan, Semoga makalah ini mampu memberikan manfaat kepada setiap pembacanya.

1.0 Profil Sistem

Perusahaan di era digital saat ini semakin meningkatkan efisiensi dan efektivitas dalam pengelolaan data karyawan dan gaji. Namun, masih banyak perusahaan yang mengelola data karyawan dan gaji secara manual, seperti mencatat data karyawan pada buku karyawan dan mengelola penggajian dengan cara menghitung gaji secara manual. Hal ini dapat menyebabkan kesalahan dalam penginputan data dan kesulitan dalam pengelolaan laporan karyawan dan gaji.

Oleh karena itu, dibutuhkan sebuah sistem yang dapat membantu perusahaan dalam mengelola data karyawan dan gaji dengan lebih baik. Sistem input data karyawan dan gaji ini akan mempermudah penginputan data karyawan dan penggajian, serta memudahkan pengelolaan laporan karyawan dan gaji. Dengan sistem ini, perusahaan dapat meningkatkan efisiensi dan efektivitas dalam pengelolaan data karyawan dan gaji, sekaligus mengurangi risiko kesalahan dalam penginputan data.

1.1 Metode Pembuatan

1. Pertama-tama, import library Tkinter dan mysql.connector untuk digunakan dalam pembuatan aplikasi. Tkinter digunakan untuk membuat GUI, sedangkan mysql.connector digunakan untuk menghubungkan aplikasi dengan database MySQL.

```
import tkinter as tk
import mysql.connector
```

2. Selanjutnya, buat Class Employee yang digunakan untuk menyimpan data karyawan. Class ini memiliki dua properti, yaitu name dan salary, yang digunakan untuk menyimpan nama dan gaji karyawan.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
```

3. Buat Class EmployeeList untuk menghandle koneksi dan interaksi dengan database MySQL. Class ini memiliki constructor yang digunakan untuk menghubungkan aplikasi dengan database MySQL dengan menggunakan informasi host, user, password, dan nama database. Selain itu, kelas ini juga memiliki beberapa method seperti :

    - Method add_employee : digunakan untuk menambahkan objek Employee baru kedalam list employees dan menambahkan record baru ke dalam tabel employees pada database MySQL.
    - Method delete_employee : digunakan untuk menghapus objek Employee dari list employees dan menghapus record yang sesuai dari tabel employees pada database MySQL.

- Method update_employee : digunakan untuk memperbarui objek Employee yang ada di dalam list employees dan memperbarui record yang sesuai di tabel employees pada database MySQL.
- Method get_employee : digunakan untuk mengambil objek Employee dari list employees pada index tertentu.

```
class EmployeeList:
    def __init__(self, host, user, password, database):
        self.cnx = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )
        self.cursor = self.cnx.cursor()
        self.employees = []

    def add_employee(self, employee):
        self.employees.append(employee)
        query = "INSERT INTO employees (name, salary) VALUES (%s, %s)"
        self.cursor.execute(query, (employee.name, employee.salary))
        self.cnx.commit()

    def delete_employee(self, employee):
        self.employees.remove(employee)
        query = "DELETE FROM employees WHERE name = %s AND salary = %s"
        self.cursor.execute(query, (employee.name, employee.salary))
        self.cnx.commit()

    def update_employee(self, employee, updated_employee):
        self.delete_employee(employee)
        self.add_employee(updated_employee)
        query = "UPDATE employees SET name = %s, salary = %s WHERE name =
%s AND salary = %s"
        self.cursor.execute(query, (updated_employee.name,
updated_employee.salary, employee.name, employee.salary))
        self.cnx.commit()

    def get_employee(self, index):
        return self.employees[index]
```

4. Buat Class EmployeeGUI yang digunakan untuk menangani pembuatan GUI menggunakan Tkinter. Class ini memiliki constructor yang menerima objek EmployeeList sebagai argumen, sehingga dapat digunakan untuk mengakses data employees dari database MySQL. Class ini juga memiliki method create_widgets yang digunakan untuk membuat frame, label, field, dan button yang akan digunakan dalam GUI, serta method create_employee, read_employee, update_employee, dan delete_employee yang digunakan untuk meng-handle event ketika button diklik.

```
class EmployeeGUI:
    def __init__(self, employee_list):
        self.employee_list = employee_list
        self.root = tk.Tk()
        self.root.title("Employee CRUD")
        self.root.geometry("400x600")

        self.create_widgets()

    def create_widgets(self):
        # Create a frame for the input fields
        input_frame = tk.Frame(self.root)
        input_frame.pack(padx=10, pady=10)

        # Create a label and text field for the employee name
        name_label = tk.Label(input_frame, text="Name:")
        name_label.grid(row=0, column=0)
        self.name_field = tk.Entry(input_frame)
        self.name_field.grid(row=0, column=1)

        # Create a label and text field for the employee salary
        salary_label = tk.Label(input_frame, text="Salary:")
        salary_label.grid(row=1, column=0)
        self.salary_field = tk.Entry(input_frame)
        self.salary_field.grid(row=1, column=1)

        # Create a frame for the buttons
        buttons_frame = tk.Frame(self.root)
        buttons_frame.pack(padx=10, pady=10)

        # Create a "Create" button
        create_button = tk.Button(buttons_frame, text="Create",
command=self.create_employee)
        create_button.grid(row=0, column=0)

        # Create a "Read" button
        read_button = tk.Button(buttons_frame, text="Read",
command=self.read_employee)
        read_button.grid(row=0, column=1)

        # Create an "Update" button
        update_button = tk.Button(buttons_frame, text="Update",
command=self.update_employee)
        update_button.grid(row=0, column=2)

        # Create a "Delete" button
        delete_button = tk.Button(buttons_frame, text="Delete",
command=self.delete_employee)
```

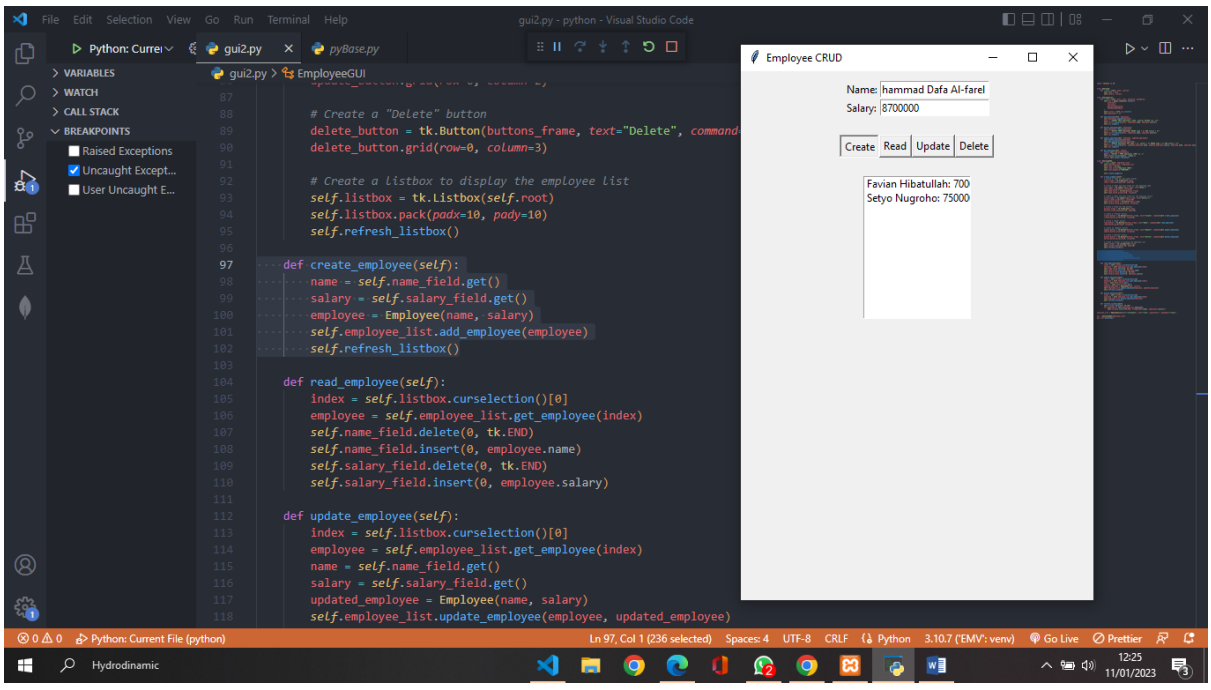
```
        delete_button.grid(row=0, column=3)

        # Create a listbox to display the employee list
        self.listbox = tk.Listbox(self.root)
        self.listbox.pack(padx=10, pady=10)
        self.refresh_listbox()
```

5. Terakhir, jalankan aplikasi dengan membuat objek dari kelas EmployeeGUI dan menjalankan mainloop dari Tkinter.

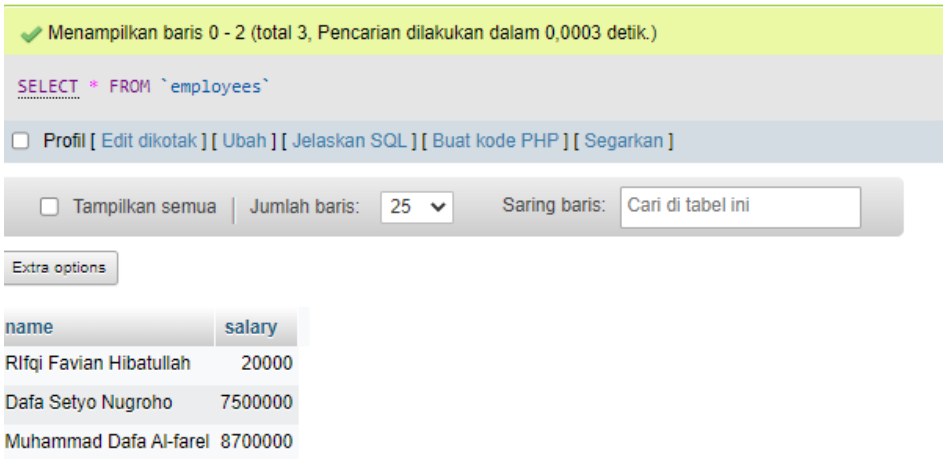
```
employee_list = EmployeeList(host="localhost", user="root", password="",
database="rifqi")

gui = EmployeeGUI(employee_list)
gui.root.mainloop()
```

1.2 Pembahasan

1. Logika Fitur Create



Hasil pada database mysql :



Ketika kita menekan tombol Create, Method create_employee akan dijalankan, Method create_employee digunakan untuk menambah data karyawan baru ke dalam aplikasi dan ke dalam tabel employees di database MySQL. Method ini memiliki beberapa langkah dalam logikanya:

1. Mengambil input nama dan gaji karyawan dari field yang diisi oleh user menggunakan method get() dari object Entry pada Tkinter.

```
        name = self.name_field.get()
        salary = self.salary_field.get()
```

2. Membuat objek Employee baru dengan nama dan gaji yang diinputkan.

```
        employee = Employee(name, salary)
```

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
```

3. Menambahkan objek Employee baru ke dalam list employees yang ada di objek EmployeeList.

```
        self.employee_list.add_employee(employee)
```

4. Menambahkan record baru ke dalam tabel employees di database MySQL menggunakan query INSERT INTO dan method execute() dari objek cursor.

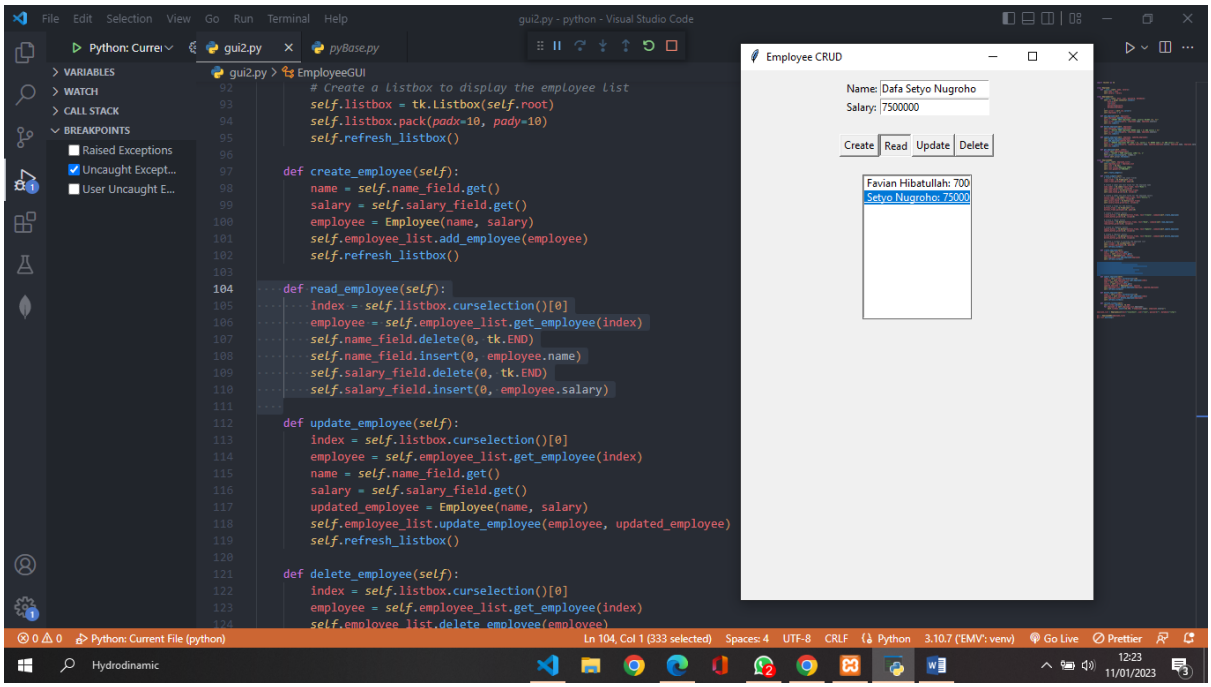
```
    def add_employee(self, employee):
        self.employees.append(employee)
        query = "INSERT INTO employees (name, salary) VALUES (%s, %s)"
        self.cursor.execute(query, (employee.name, employee.salary))
```

5. Melakukan commit untuk mengirim perubahan ke dalam database menggunakan method commit() dari objek MySQL Connector.

```
        self.cnx.commit()
```

Secara keseluruhan, method create_employee digunakan untuk menambahkan data karyawan baru ke dalam aplikasi dan menyimpannya ke dalam database MySQL agar dapat diakses dan digunakan pada waktu yang akan datang.

2. Logika Fitur Read



Ketika kita menekan tombol read, method read_employee akan dijalankan, Method read_employee digunakan untuk menampilkan data karyawan yang ada di dalam aplikasi dan di database MySQL. Logika dari method ini sebagai berikut:

1. Mengambil input index dari field yang diisi oleh user menggunakan method get() dari object Entry pada Tkinter.

```
        index = self.listbox.curselection()[0]
```

2. Mengambil objek Employee dari list employees pada index yang diinputkan menggunakan method get_employee() dari objek EmployeeList.

```
        employee = self.employee_list.get_employee(index)
```

```
    def get_employee(self, index):
        return self.employees[index]
```

3. Menghapus isi dari field nama dan gaji menggunakan method delete() dari object Entry pada Tkinter.

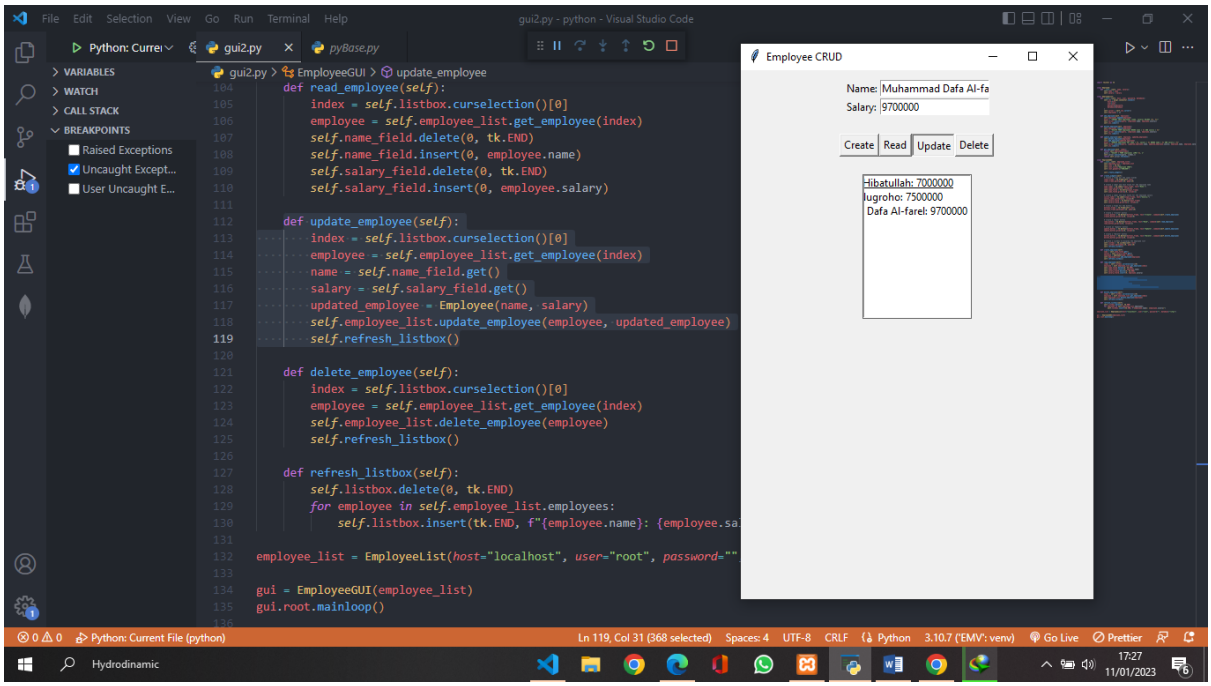
```
        self.name_field.delete(0, tk.END)
        self.salary_field.delete(0, tk.END)
```

4. Memasukkan nama dan gaji dari objek Employee ke dalam field nama dan gaji menggunakan method insert() dari object Entry pada Tkinter.

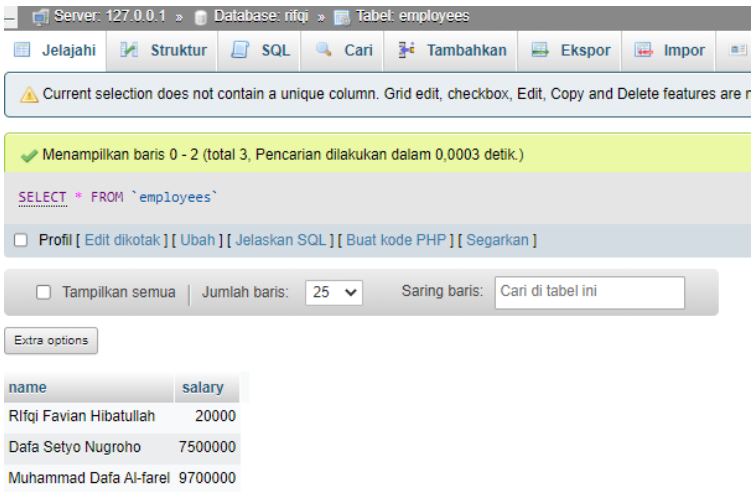
```
        self.name_field.insert(0, employee.name)
        self.salary_field.insert(0, employee.salary)
```

Secara keseluruhan, method read_employee digunakan untuk menampilkan data karyawan yang ada pada index yang diinputkan oleh user ke dalam field yang sesuai, sehingga user dapat melihat data karyawan yang diinginkan.

3. Logika Fitur Update



Hasil pada database :



name	salary
Rifqi Favian Hibatullah	20000
Dafa Setyo Nugroho	7500000
Muhammad Dafa Al-farel	9700000

Ketika kita menekan tombol update, maka method update_employee akan dijalankan, Method update_employee digunakan untuk memperbarui data karyawan yang ada di dalam aplikasi dan di database MySQL. Logika dari method ini sebagai berikut:

1. Mengambil input index dari field yang diisi oleh user menggunakan method get() dari object Entry pada Tkinter.

```
        index = self.listbox.curselection()[0]
```

2. Mengambil objek Employee dari list employees pada index yang diinputkan menggunakan method get_employee() dari objek EmployeeList.

```
        employee = self.employee_list.get_employee(index)
```

3. Mengambil input nama dan gaji baru dari field yang diisi oleh user menggunakan method get() dari object Entry pada Tkinter.

```
        name = self.name_field.get()
        salary = self.salary_field.get()
```

4. Membuat objek Employee baru dengan nama dan gaji yang diinputkan.

```
        updated_employee = Employee(name, salary)
```

5. Menghapus objek Employee lama dari list employees menggunakan method delete_employee() dari objek EmployeeList

```
    def delete_employee(self):
        index = self.listbox.curselection()[0]
        employee = self.employee_list.get_employee(index)
        self.employee_list.delete_employee(employee)
        self.refresh_listbox()
```

6. Menambahkan objek Employee baru ke dalam list employees.

```
        self.employee_list.update_employee(employee, updated_employee)
        self.refresh_listbox()
```

```
    def refresh_listbox(self):
          self.listbox.delete(0, tk.END)
          for employee in self.employee_list.employees:
                self.listbox.insert(tk.END, f"{employee.name}:
{employee.salary}")
```

7. Memperbarui record di tabel employees dan di database MySQL menggunakan query UPDATE dan method execute() dari objek cursor.

```
    def update_employee(self, employee, updated_employee):
        self.delete_employee(employee)
```

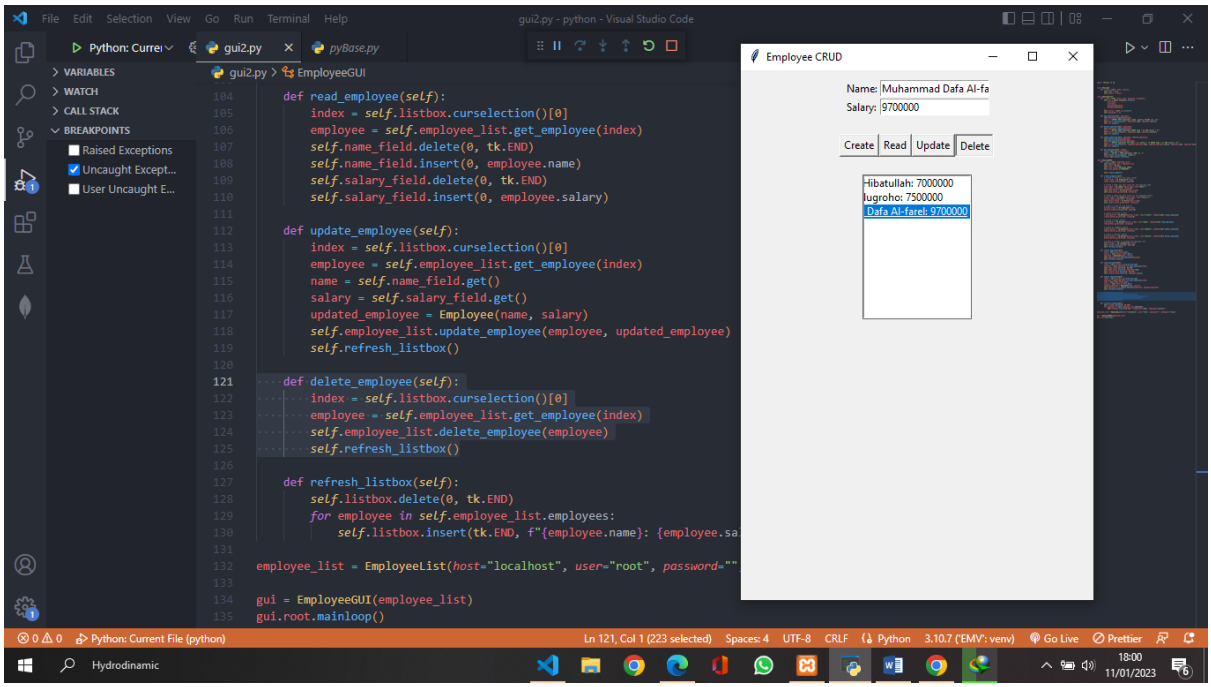
```
        self.add_employee(updated_employee)
        query = "UPDATE employees SET name = %s, salary = %s WHERE name =
%s AND salary = %s"
        self.cursor.execute(query, (updated_employee.name,
updated_employee.salary, employee.name, employee.salary))
```

8. Melakukan commit untuk mengirim perubahan ke dalam database menggunakan method commit() dari objek MySQL Connector.

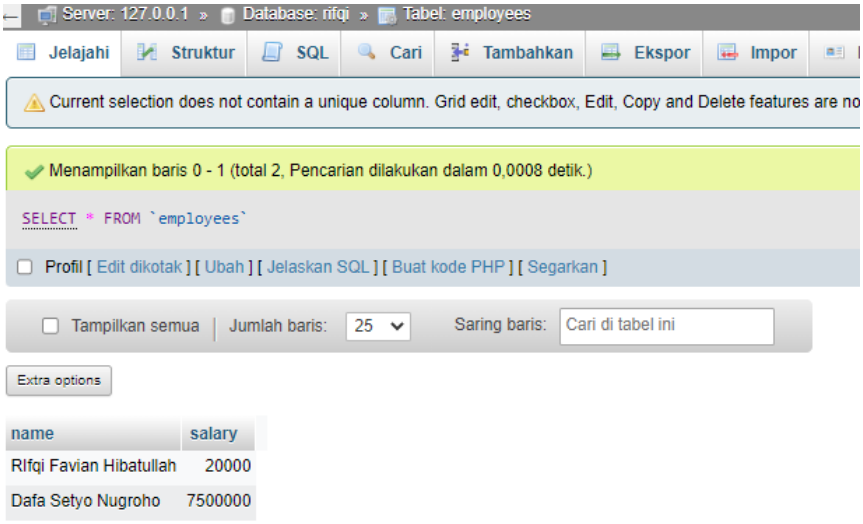
```
        self.cnx.commit()
```

Secara keseluruhan, method update_employee digunakan untuk memperbarui data karyawan yang ada pada index yang diinputkan oleh user dengan data yang diinputkan ke dalam field yang sesuai. Perubahan data akan disimpan ke dalam list employee dan database MySQL agar dapat diakses dan digunakan pada waktu yang akan datang.

3.Logika Fitur Delete



Hasil pada database



Method delete_employee digunakan untuk menghapus data karyawan yang ada di dalam aplikasi dan di database MySQL. Logika dari method ini sebagai berikut:

1. Mengambil input index dari field yang diisi oleh user menggunakan method get() dari object Entry pada Tkinter.

```
        index = self.listbox.curselection()[0]
```

2. Mengambil objek Employee dari list employees pada index yang diinputkan menggunakan method get_employee() dari objek EmployeeList.

```
        employee = self.employee_list.get_employee(index)
```

3. Menghapus objek Employee dari list employees menggunakan method delete_employee() dari objek EmployeeList.

```
        self.employee_list.delete_employee(employee)
```

4. Menghapus record di tabel employees di database MySQL menggunakan query DELETE dan method execute() dari objek cursor.

```
    def delete_employee(self, employee):
        self.employees.remove(employee)
        query = "DELETE FROM employees WHERE name = %s AND salary = %s"
        self.cursor.execute(query, (employee.name, employee.salary))
```

5. Melakukan commit untuk mengirim perubahan ke dalam database menggunakan method commit() dari objek MySQL Connector.

```
        self.cnx.commit()
```

Secara keseluruhan, method delete_employee digunakan untuk menghapus data karyawan yang ada pada index yang diinputkan oleh user, perubahan data yang dihapus akan disimpan di list employee dan dihapus dari database MySQL.

1.3 Link Video : https://drive.google.com/drive/folders/15h9k6Y__G1NxlEt2-ApLahEuU2fxiesW