HƯỚNG DẪN DỰ ÁN: QUẢN LÝ NGƯỜI DÙNG VỚI MONGODB VÀ TKINTER

Tác giả: Đặng Kim Thi

Mô tả: Ứng dụng CRUD (Create, Read, Update, Delete) quản lý thông tin người dùng sử dụng MongoDB làm cơ sở dữ liệu và Tkinter làm giao diện đồ họa.

PHÂN TÍCH CHI TIẾT MÃ NGUỒN (TOP-DOWN)

Dưới đây là phân tích chi tiết mã nguồn theo cách tiếp cận **top-down** (từ tổng quan đến chi tiết), giải thích từng hàm, từng dòng lệnh và ý nghĩa của từng biến.

Tổng quan chương trình

- Chương trình bắt đầu từ hàm `main()` trong khối `if __name__ == "__main__":`, tạo một instance của lớp `MongoCRUD` và chạy giao diện bằng phương thức `run()`.
- Lớp `MongoCRUD` chứa toàn bộ logic của ứng dụng, bao gồm kết nối MongoDB, xây dựng giao diện Tkinter, và các chức năng CRUD.

Phân tích chi tiết từng phần

1. Khai báo thư viện

```
from pymongo import MongoClient
import tkinter as tk
from tkinter import messagebox, ttk
import sys
```

- **Ý nghĩa từng dòng:**
  - `from pymongo import MongoClient`: Nhập lớp `MongoClient` từ thư viện `pymongo` để kết nối với MongoDB.
  - `import tkinter as tk`: Nhập thư viện `tkinter` (giao diện đồ họa) và gán bí danh `tk`.
  - `from tkinter import messagebox, ttk`: Nhập hai module con từ `tkinter`:
    - `messagebox`: Hiển thị hộp thoại thông báo (lỗi, cảnh báo, thành công).
    - `ttk`: Cung cấp các widget nâng cao như `Treeview` để hiển thị bảng.
  - `import sys`: Nhập thư viện `sys` để sử dụng `sys.exit()` khi thoát chương trình.

2. Lớp `MongoCRUD`

```
class MongoCRUD:
```

- **Ý nghĩa**: Định nghĩa lớp `MongoCRUD` chứa toàn bộ logic của ứng dụng.

Hàm `__init__` (Khởi tạo)

```
def __init__(self):
    # Kết nối đến MongoDB
    self.client = MongoClient('mongodb://localhost:27017/')
    self.db = self.client['python_test']
    self.collection = self.db['users']
```

- **Ý nghĩa**: Hàm khởi tạo đối tượng `MongoCRUD`.
- **Chi tiết từng dòng:**
  - `self.client = MongoClient('mongodb://localhost:27017/')`:
    - Tạo kết nối đến MongoDB server chạy trên `localhost` (máy cục bộ), port `27017` (port mặc định của MongoDB).
    - Biến `self.client`: Đối tượng kết nối MongoDB, dùng để truy cập cơ sở dữ liệu.
  - `self.db = self.client['python_test']`:
    - Chọn cơ sở dữ liệu có tên `python_test`. Nếu chưa tồn tại, MongoDB sẽ tự tạo khi dữ liệu được thêm vào.
    - Biến `self.db`: Đại diện cho cơ sở dữ liệu `python_test`.
  - `self.collection = self.db['users']`:
    - Chọn collection (tương tự bảng trong SQL) có tên `users` trong cơ sở dữ liệu `python_test`.
    - Biến `self.collection`: Đại diện cho collection `users`, nơi lưu trữ dữ liệu người dùng.

```
    # Tạo giao diện Tkinter
    self.window = tk.Tk()
    self.window.title("Quản Lý Người Dùng - Đặng Kim Thi")
    self.window.geometry("700x400")
```

- **Ý nghĩa**: Khởi tạo cửa sổ giao diện chính bằng Tkinter.
- **Chi tiết từng dòng:**
  - `self.window = tk.Tk()`: Tạo một cửa sổ giao diện chính, gán vào biến `self.window`.
  - `self.window.title("Quản Lý Người Dùng - Đặng Kim Thi")`: Đặt tiêu đề cho cửa sổ, bao gồm tên tác giả.
  - `self.window.geometry("700x400")`: Đặt kích thước cửa sổ là 700px (rộng) x 400px (cao).

```
    # Frame cho các trường nhập liệu
    input_frame = tk.LabelFrame(self.window, text="Thông tin người dùng",
padx=10, pady=10)
    input_frame.pack(padx=10, pady=5, fill="x")
```

- **Ý nghĩa**: Tạo một khung (`Frame`) để chứa các trường nhập liệu.
- **Chi tiết từng dòng:**

- `input_frame = tk.LabelFrame(self.window, text="Thông tin người dùng", padx=10, pady=10)`:
  - Tạo một `LabelFrame` (khung có tiêu đề) bên trong `self.window`.
  - `text="Thông tin người dùng"`: Tiêu đề của khung.
  - `padx=10, pady=10`: Khoảng cách bên trong khung (10px theo chiều ngang và dọc).
  - Biến `input_frame`: Đối tượng khung chứa các trường nhập liệu.
- `input_frame.pack(padx=10, pady=5, fill="x")`:
  - Đặt khung vào cửa sổ chính.
  - `padx=10, pady=5`: Khoảng cách bên ngoài (10px ngang, 5px dọc).
  - `fill="x"`: Khung sẽ giãn theo chiều ngang để lấp đầy cửa sổ.

```
    # Các trường nhập liệu
    tk.Label(input_frame, text="Tên:").grid(row=0, column=0, padx=5,
pady=5)
    self.name_entry = tk.Entry(input_frame)
    self.name_entry.grid(row=0, column=1, padx=5, pady=5)

    tk.Label(input_frame, text="Tuổi:").grid(row=1, column=0, padx=5,
pady=5)
    self.age_entry = tk.Entry(input_frame)
    self.age_entry.grid(row=1, column=1, padx=5, pady=5)

    tk.Label(input_frame, text="Email:").grid(row=2, column=0, padx=5,
pady=5)
    self.email_entry = tk.Entry(input_frame)
    self.email_entry.grid(row=2, column=1, padx=5, pady=5)
```

- **Ý nghĩa**: Tạo các nhãn (`Label`) và trường nhập liệu (`Entry`) cho "Tên", "Tuổi", "Email".
- **Chi tiết từng dòng:**
  - `tk.Label(input_frame, text="Tên:")`: Tạo nhãn "Tên:" bên trong `input_frame`.
  - `.grid(row=0, column=0, padx=5, pady=5)`: Đặt nhãn ở hàng 0, cột 0, với khoảng cách 5px.
  - `self.name_entry = tk.Entry(input_frame)`: Tạo trường nhập liệu cho tên, gán vào `self.name_entry`.
  - `.grid(row=0, column=1, padx=5, pady=5)`: Đặt trường nhập liệu ở hàng 0, cột 1.
  - Tương tự cho "Tuổi" (hàng 1) và "Email" (hàng 2).
  - **Biến**:
    - `self.name_entry`: Lưu trữ dữ liệu tên người dùng nhập vào.
    - `self.age_entry`: Lưu trữ tuổi.
    - `self.email_entry`: Lưu trữ email.

```
    # Frame cho các nút chức năng
    button_frame = tk.Frame(self.window)
    button_frame.pack(pady=10)
```

- **Ý nghĩa**: Tạo khung chứa các nút chức năng.

- **Chi tiết từng dòng:**
  - `button_frame = tk.Frame(self.window)`: Tạo một `Frame` đơn giản trong `self.window`.
  - `button_frame.pack(pady=10)`: Đặt khung với khoảng cách 10px phía trên/dưới.

```
    # Các nút chức năng
    tk.Button(button_frame, text="Thêm",
command=self.create).pack(side=tk.LEFT, padx=5)
    tk.Button(button_frame, text="Xem DS",
command=self.read).pack(side=tk.LEFT, padx=5)
    tk.Button(button_frame, text="Cập nhật",
command=self.update).pack(side=tk.LEFT, padx=5)
    tk.Button(button_frame, text="Xóa",
command=self.delete).pack(side=tk.LEFT, padx=5)
    tk.Button(button_frame, text="Thoát",
command=self.quit).pack(side=tk.LEFT, padx=5)
```

- **Ý nghĩa**: Tạo các nút cho các chức năng CRUD và thoát.
- **Chi tiết từng dòng:**
  - `tk.Button(button_frame, text="Thêm", command=self.create)`: Tạo nút "Thêm" gọi hàm `self.create`.
  - `.pack(side=tk.LEFT, padx=5)`: Đặt nút sang trái, khoảng cách 5px với nút kế bên.
  - Tương tự cho các nút "Xem DS" (`self.read`), "Cập nhật" (`self.update`), "Xóa" (`self.delete`), "Thoát" (`self.quit`).

```
    # Treeview để hiển thị dữ liệu dạng bảng
    self.tree = ttk.Treeview(self.window, columns=("ID", "Tên", "Tuổi",
"Email"), show="headings")
    self.tree.heading("ID", text="ID")
    self.tree.heading("Tên", text="Tên")
    self.tree.heading("Tuổi", text="Tuổi")
    self.tree.heading("Email", text="Email")

    self.tree.column("ID", width=180)
    self.tree.column("Tên", width=150)
    self.tree.column("Tuổi", width=50, anchor="center")
    self.tree.column("Email", width=200)

    self.tree.pack(padx=10, pady=10, fill="both", expand=True)
```

- **Ý nghĩa**: Tạo bảng (`Treeview`) để hiển thị danh sách người dùng.
- **Chi tiết từng dòng:**
  - `self.tree = ttk.Treeview(...)`: Tạo bảng với 4 cột: "ID", "Tên", "Tuổi", "Email".
    - `show="headings"`: Chỉ hiển thị tiêu đề cột, không hiển thị cột mặc định.
    - Biến `self.tree`: Đối tượng bảng để hiển thị dữ liệu.
  - `self.tree.heading("ID", text="ID")`: Đặt tiêu đề "ID" cho cột đầu tiên (tương tự cho các cột khác).

- `self.tree.column("ID", width=180)`: Đặt chiều rộng cột "ID" là 180px (tương tự cho các cột khác).
- `self.tree.column("Tuổi", width=50, anchor="center")`: Cột "Tuổi" rộng 50px, căn giữa.
- `self.tree.pack(padx=10, pady=10, fill="both", expand=True)`: Đặt bảng vào cửa sổ, giãn đầy không gian.

Hàm `clear_entries`

```
def clear_entries(self):
    self.name_entry.delete(0, tk.END)
    self.age_entry.delete(0, tk.END)
    self.email_entry.delete(0, tk.END)
```

- **Ý nghĩa**: Xóa dữ liệu trong các trường nhập liệu.
- **Chi tiết từng dòng:**
  - `self.name_entry.delete(0, tk.END)`: Xóa từ vị trí 0 đến cuối trong trường "Tên".
  - Tương tự cho `self.age_entry` (Tuổi) và `self.email_entry` (Email).

Hàm `create`

```
def create(self):
    try:
        name = self.name_entry.get()
        age = int(self.age_entry.get())
        email = self.email_entry.get()

        if not all([name, age, email]):
            messagebox.showerror("Lỗi", "Vui lòng điền đầy đủ thông tin!")
            return

        user = {"name": name, "age": age, "email": email}
        result = self.collection.insert_one(user)

        self.clear_entries()
        self.read()
        messagebox.showinfo("Thành công", f"Đã thêm người dùng với ID:
{result.inserted_id}")
    except ValueError:
        messagebox.showerror("Lỗi", "Tuổi phải là số nguyên!")
    except Exception as e:
        messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")
```

- **Ý nghĩa**: Thêm một người dùng mới vào MongoDB.
- **Chi tiết từng dòng:**
  - `try:`: Bắt đầu khối xử lý lỗi.
  - `name = self.name_entry.get()`: Lấy dữ liệu từ trường "Tên".

- `age = int(self.age_entry.get())`: Lấy dữ liệu từ trường "Tuổi" và chuyển thành số nguyên.
- `email = self.email_entry.get()`: Lấy dữ liệu từ trường "Email".
- `if not all([name, age, email])`: Kiểm tra xem có trường nào trống không.
  - `messagebox.showerror(...)`: Hiển thị lỗi nếu thiếu thông tin.
  - `return`: Thoát hàm.
- `user = {"name": name, "age": age, "email": email}`: Tạo dictionary chứa thông tin người dùng.
- `result = self.collection.insert_one(user)`: Thêm `user` vào collection `users`.
  - Biến `result`: Kết quả của lệnh thêm, chứa `inserted_id` (ID của bản ghi mới).
- `self.clear_entries()`: Xóa các trường nhập liệu.
- `self.read()`: Cập nhật bảng dữ liệu.
- `messagebox.showinfo(...)`: Thông báo thành công với ID mới.
- `except ValueError:`: Xử lý lỗi nếu tuổi không phải số nguyên.
- `except Exception as e:`: Xử lý các lỗi khác.

Hàm `read`

```
def read(self):
    self.tree.delete(*self.tree.get_children())
    users = self.collection.find()
    for user in users:
        self.tree.insert("", "end", values=(str(user["_id"]),
user["name"], user["age"], user["email"]))
```

- **Ý nghĩa**: Hiển thị danh sách người dùng trong bảng.
- **Chi tiết từng dòng:**
  - `self.tree.delete(*self.tree.get_children())`: Xóa toàn bộ dữ liệu hiện tại trong bảng.
    - `self.tree.get_children()`: Lấy danh sách các dòng trong bảng.
  - `users = self.collection.find()`: Truy vấn tất cả bản ghi trong collection `users`.
    - Biến `users`: Cursor chứa danh sách người dùng.
  - `for user in users:`: Duyệt qua từng người dùng.
  - `self.tree.insert(...)`: Thêm một dòng mới vào bảng.
    - `""`: Thêm vào gốc (root).
    - `"end"`: Thêm vào cuối danh sách.
    - `values=(str(user["_id"]), ...)`: Giá trị các cột: ID (chuỗi), Tên, Tuổi, Email.

Hàm `update`

```
def update(self):
    try:
        email = self.email_entry.get()
        if not email:
            messagebox.showerror("Lỗi", "Vui lòng nhập email để cập
```

```
nhật!")
            return

        update_data = {}
        if self.name_entry.get():
            update_data["name"] = self.name_entry.get()
        if self.age_entry.get():
            update_data["age"] = int(self.age_entry.get())

        if update_data:
            result = self.collection.update_one(
                {"email": email},
                {"$set": update_data}
            )
            self.clear_entries()
            self.read()
            if result.modified_count > 0:
                messagebox.showinfo("Thành công", "Cập nhật thành công!")
            else:
                messagebox.showwarning("Cảnh báo", "Không tìm thấy người
dùng hoặc không có thay đổi!")
        else:
            messagebox.showwarning("Cảnh báo", "Vui lòng nhập ít nhất một
thông tin để cập nhật!")
    except ValueError:
        messagebox.showerror("Lỗi", "Tuổi phải là số nguyên!")
    except Exception as e:
        messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")
```

- **Ý nghĩa**: Cập nhật thông tin người dùng dựa trên email.
- **Chi tiết từng dòng:**
  - `email = self.email_entry.get()`: Lấy email để tìm người dùng.
  - `if not email:`: Kiểm tra email có trống không.
  - `update_data = {}`: Dictionary chứa dữ liệu cần cập nhật.
  - `if self.name_entry.get():`: Nếu có tên mới, thêm vào `update_data`.
  - `if self.age_entry.get():`: Nếu có tuổi mới, thêm vào sau khi chuyển thành số nguyên.
  - `if update_data:`: Kiểm tra xem có dữ liệu để cập nhật không.
    - `result = self.collection.update_one(...)`: Cập nhật một bản ghi.
      - `{"email": email}`: Điều kiện tìm kiếm.
      - `{"$set": update_data}`: Cập nhật các trường trong `update_data`.
    - `self.clear_entries()`: Xóa trường nhập liệu.
    - `self.read()`: Cập nhật bảng.
    - `if result.modified_count > 0:`: Kiểm tra xem có bản ghi nào được cập nhật không.

Hàm `delete`

```
def delete(self):
    try:
```

```
        email = self.email_entry.get()
        if not email:
            messagebox.showerror("Lỗi", "Vui lòng nhập email để xóa!")
            return

        result = self.collection.delete_one({"email": email})
        self.clear_entries()
        self.read()
        if result.deleted_count > 0:
            messagebox.showinfo("Thành công", "Xóa thành công!")
        else:
            messagebox.showwarning("Cảnh báo", "Không tìm thấy người dùng
với email này!")
    except Exception as e:
        messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")
```

- **Ý nghĩa**: Xóa người dùng dựa trên email.
- **Chi tiết từng dòng:**
  - `result = self.collection.delete_one({"email": email})`: Xóa bản ghi có email tương ứng.
    - Biến `result`: Kết quả xóa, chứa `deleted_count` (số bản ghi bị xóa).

Hàm `quit`

```
def quit(self):
    self.client.close()
    self.window.quit()
    sys.exit(0)
```

- **Ý nghĩa**: Thoát chương trình.
- **Chi tiết từng dòng:**
  - `self.client.close()`: Đóng kết nối MongoDB.
  - `self.window.quit()`: Đóng cửa sổ Tkinter.
  - `sys.exit(0)`: Thoát chương trình với mã trạng thái 0 (thành công).

Hàm `run`

```
def run(self):
    self.window.mainloop()
```

- **Ý nghĩa**: Chạy vòng lặp sự kiện Tkinter để hiển thị giao diện.
- **Chi tiết**: `self.window.mainloop()` giữ cửa sổ mở và xử lý các sự kiện (nhấn nút, nhập liệu, v.v.).

3. Khối thực thi chính

```
if __name__ == "__main__":
    app = MongoCRUD()
    app.run()
```

- **Ý nghĩa**: Điểm bắt đầu chương trình.
- **Chi tiết từng dòng:**
  - `if __name__ == "__main__":`: Đảm bảo đoạn code chỉ chạy khi file được thực thi trực tiếp.
  - `app = MongoCRUD()`: Tạo instance của lớp `MongoCRUD`.
  - `app.run()`: Chạy giao diện.

---

1. MÔ TẢ DỰ ÁN

Dự án này là một ứng dụng quản lý người dùng với các chức năng cơ bản:

- **Thêm (Create):** Thêm thông tin người dùng mới vào cơ sở dữ liệu.
- **Xem danh sách (Read):** Hiển thị danh sách người dùng dưới dạng bảng.
- **Cập nhật (Update):** Chỉnh sửa thông tin người dùng dựa trên email.
- **Xóa (Delete):** Xóa người dùng dựa trên email.

Ứng dụng sử dụng:

- **MongoDB:** Cơ sở dữ liệu NoSQL để lưu trữ thông tin người dùng.
- **Python:** Ngôn ngữ lập trình chính.
- **Tkinter:** Thư viện giao diện đồ họa để xây dựng giao diện người dùng.
- **Pymongo:** Thư viện kết nối Python với MongoDB.

---

2. YÊU CẦU HỆ THỐNG

Phần mềm cần thiết:

1. **Python**: Phiên bản 3.6 trở lên.
2. **MongoDB**: Cài đặt và chạy trên localhost (port mặc định: 27017).
3. **Thư viện Python**:
   - `pymongo`: Để kết nối với MongoDB.
   - `tkinter`: Đi kèm với Python (không cần cài thêm).
   - `ttk`: Một phần của Tkinter để tạo bảng (Treeview).

Cài đặt thư viện:

Mở terminal/command prompt và chạy lệnh sau:

```
pip install pymongo
```

Cài đặt MongoDB:

- Tải MongoDB từ trang chính thức: https://www.mongodb.com/try/download/community
- Cài đặt và chạy MongoDB server bằng lệnh:

```
mongod
```

- Đảm bảo MongoDB đang chạy trên `localhost:27017`.

---

3. CẤU TRÚC MÃ NGUỒN

Mã nguồn đầy đủ:

```
# Quản lý người dùng với MongoDB và Tkinter
# Tác giả: Đặng Kim Thi

from pymongo import MongoClient
import tkinter as tk
from tkinter import messagebox, ttk
import sys

class MongoCRUD:
    def __init__(self):
        # Kết nối đến MongoDB
        self.client = MongoClient('mongodb://localhost:27017/')
        self.db = self.client['python_test']
        self.collection = self.db['users']

        # Tạo giao diện Tkinter
        self.window = tk.Tk()
        self.window.title("Quản Lý Người Dùng - Đặng Kim Thi")
        self.window.geometry("700x400")

        # Frame cho các trường nhập liệu
        input_frame = tk.LabelFrame(self.window, text="Thông tin người
dùng", padx=10, pady=10)
        input_frame.pack(padx=10, pady=5, fill="x")

        # Các trường nhập liệu
        tk.Label(input_frame, text="Tên:").grid(row=0, column=0, padx=5,
pady=5)
        self.name_entry = tk.Entry(input_frame)
        self.name_entry.grid(row=0, column=1, padx=5, pady=5)

        tk.Label(input_frame, text="Tuổi:").grid(row=1, column=0, padx=5,
pady=5)
        self.age_entry = tk.Entry(input_frame)
        self.age_entry.grid(row=1, column=1, padx=5, pady=5)

        tk.Label(input_frame, text="Email:").grid(row=2, column=0, padx=5,
pady=5)
```

```
        self.email_entry = tk.Entry(input_frame)
        self.email_entry.grid(row=2, column=1, padx=5, pady=5)

        # Frame cho các nút chức năng
        button_frame = tk.Frame(self.window)
        button_frame.pack(pady=10)

        # Các nút chức năng
        tk.Button(button_frame, text="Thêm",
command=self.create).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Xem DS",
command=self.read).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Cập nhật",
command=self.update).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Xóa",
command=self.delete).pack(side=tk.LEFT, padx=5)
        tk.Button(button_frame, text="Thoát",
command=self.quit).pack(side=tk.LEFT, padx=5)

        # Treeview để hiển thị dữ liệu dạng bảng
        self.tree = ttk.Treeview(self.window, columns=("ID", "Tên",
"Tuổi", "Email"), show="headings")
        self.tree.heading("ID", text="ID")
        self.tree.heading("Tên", text="Tên")
        self.tree.heading("Tuổi", text="Tuổi")
        self.tree.heading("Email", text="Email")

        self.tree.column("ID", width=180)
        self.tree.column("Tên", width=150)
        self.tree.column("Tuổi", width=50, anchor="center")
        self.tree.column("Email", width=200)

        self.tree.pack(padx=10, pady=10, fill="both", expand=True)

    def clear_entries(self):
        self.name_entry.delete(0, tk.END)
        self.age_entry.delete(0, tk.END)
        self.email_entry.delete(0, tk.END)

    def create(self):
        try:
            name = self.name_entry.get()
            age = int(self.age_entry.get())
            email = self.email_entry.get()

            if not all([name, age, email]):
                messagebox.showerror("Lỗi", "Vui lòng điền đầy đủ thông
tin!")
                return

            user = {"name": name, "age": age, "email": email}
            result = self.collection.insert_one(user)

            self.clear_entries()
```

```
            self.read()
            messagebox.showinfo("Thành công", f"Đã thêm người dùng với ID:
{result.inserted_id}")
        except ValueError:
            messagebox.showerror("Lỗi", "Tuổi phải là số nguyên!")
        except Exception as e:
            messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")

    def read(self):
        self.tree.delete(*self.tree.get_children())
        users = self.collection.find()
        for user in users:
            self.tree.insert("", "end", values=(str(user["_id"]),
user["name"], user["age"], user["email"]))

    def update(self):
        try:
            email = self.email_entry.get()
            if not email:
                messagebox.showerror("Lỗi", "Vui lòng nhập email để cập
nhật!")
                return

            update_data = {}
            if self.name_entry.get():
                update_data["name"] = self.name_entry.get()
            if self.age_entry.get():
                update_data["age"] = int(self.age_entry.get())

            if update_data:
                result = self.collection.update_one(
                    {"email": email},
                    {"$set": update_data}
                )
                self.clear_entries()
                self.read()
                if result.modified_count > 0:
                    messagebox.showinfo("Thành công", "Cập nhật thành
công!")
                else:
                    messagebox.showwarning("Cảnh báo", "Không tìm thấy
người dùng hoặc không có thay đổi!")
            else:
                messagebox.showwarning("Cảnh báo", "Vui lòng nhập ít nhất
một thông tin để cập nhật!")
        except ValueError:
            messagebox.showerror("Lỗi", "Tuổi phải là số nguyên!")
        except Exception as e:
            messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")

    def delete(self):
        try:
            email = self.email_entry.get()
            if not email:
```

```
                messagebox.showerror("Lỗi", "Vui lòng nhập email để xóa!")
                return

            result = self.collection.delete_one({"email": email})
            self.clear_entries()
            self.read()
            if result.deleted_count > 0:
                messagebox.showinfo("Thành công", "Xóa thành công!")
            else:
                messagebox.showwarning("Cảnh báo", "Không tìm thấy người
dùng với email này!")
        except Exception as e:
            messagebox.showerror("Lỗi", f"Có lỗi xảy ra: {e}")

    def quit(self):
        self.client.close()
        self.window.quit()
        sys.exit(0)

    def run(self):
        self.window.mainloop()

if __name__ == "__main__":
    app = MongoCRUD()
    app.run()
```

---

4. HƯỚNG DẪN SỬ DỤNG

Cách chạy chương trình:

1. Lưu mã nguồn vào file (ví dụ: `mongo_crud.py`).
2. Đảm bảo MongoDB đang chạy trên `localhost:27017`.
3. Mở terminal, di chuyển đến thư mục chứa file và chạy:

```
python mongo_crud.py
```

Giao diện:

- **Trường nhập liệu**: Điền thông tin "Tên", "Tuổi", "Email".
- **Các nút chức năng**:
  - **Thêm**: Thêm người dùng mới vào cơ sở dữ liệu.
  - **Xem DS**: Hiển thị danh sách người dùng trong bảng.
  - **Cập nhật**: Cập nhật thông tin dựa trên email.
  - **Xóa**: Xóa người dùng dựa trên email.
  - **Thoát**: Đóng ứng dụng.
- **Bảng Treeview**: Hiển thị danh sách người dùng với các cột: ID, Tên, Tuổi, Email.

Hướng dẫn thao tác:

1. **Thêm người dùng**:
    - Nhập đầy đủ "Tên", "Tuổi", "Email" vào các trường.
    - Nhấn nút "Thêm".
2. **Xem danh sách**:
    - Nhấn nút "Xem DS" để tải danh sách người dùng.
3. **Cập nhật**:
    - Nhập email của người dùng cần cập nhật.
    - Điền thông tin mới (Tên hoặc Tuổi hoặc cả hai).
    - Nhấn nút "Cập nhật".
4. **Xóa**:
    - Nhập email của người dùng cần xóa.
    - Nhấn nút "Xóa".

---

5. XỬ LÝ LỖI

- **Tuổi không phải số nguyên**: Hiển thị thông báo lỗi.
- **Trường trống**: Yêu cầu điền đầy đủ thông tin khi thêm, hoặc email khi cập nhật/xóa.
- **Không tìm thấy người dùng**: Hiển thị cảnh báo khi email không tồn tại trong cơ sở dữ liệu.
- **Lỗi kết nối MongoDB**: Thông báo nếu MongoDB không chạy hoặc cấu hình sai.

---

6. KẾT LUẬN

Ứng dụng này cung cấp một giao diện đơn giản và trực quan để quản lý dữ liệu người dùng trong MongoDB. Dự án có thể được mở rộng bằng cách:

- Thêm tính năng tìm kiếm.
- Hỗ trợ nhiều trường dữ liệu hơn (địa chỉ, số điện thoại, v.v.).
- Tích hợp xác thực người dùng.

Tác giả: Đặng Kim Thi