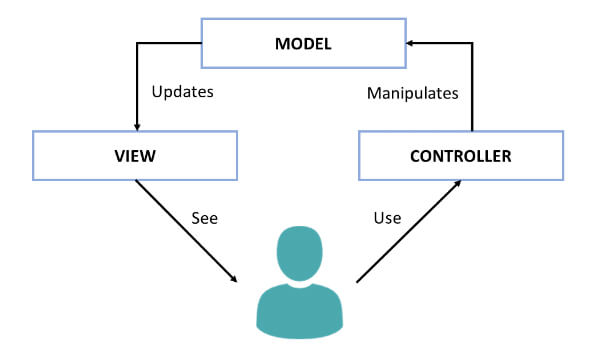
Họ và tên : Dương Đức Trung Nguyên MSV: 20224067

**Mô hình MVC**

   **MVC (Model – View – Controller)** là một mô hình thiết kế hay kiến trúc được sử dụng phổ biến trong kỹ thuật phần mềm. Nó có nhiệm vụ phân bổ các source code thành 3 phần chính. Mỗi thành phần có một nhiệm vụ riêng biệt và xử lý độc lập với các thành phần khác. Mô hình MVC giúp các nhà thiết kế, lập trình web xử lý các yêu cầu kỹ thuật và hoàn thiện sản phẩm dễ dàng và nhanh chóng hơn. Hiện nay có nhiều mã nguồn mở và ngôn ngữ lập trình web phổ biến khác như Magento, Ruby On Rails, Drupal, Codeingteger… cũng đang được tối ưu và thiết lập theo đúng với mô hình MVC.

**Model (M - Model):** Đây là thành phần chịu trách nhiệm cho xử lý dữ liệu và logic kinh doanh của ứng dụng. Model là nơi lưu trữ thông tin, thực hiện các thao tác cập nhật và truy vấn dữ liệu, mà không quan tâm đến cách dữ liệu được hiển thị hoặc tương tác với người dùng.

**View (V - View):** View là thành phần hiển thị giao diện người dùng và đảm nhận trách nhiệm hiển thị thông tin từ Model. View không có logic kinh doanh và chỉ chịu trách nhiệm về việc hiển thị dữ liệu một cách đẹp mắt và dễ hiểu cho người dùng.

**Controller (C - Controller):** Controller là thành phần điều phối và xử lý sự kiện từ người dùng. Nó nhận lệnh từ người dùng thông qua View, sau đó cập nhật Model dựa trên những thay đổi này và điều hướng hiển thị trở lại View. Controller giữ vai trò quan trọng trong việc duy trì sự đồng bộ giữa Model và View.

Mô hình MVC giúp tách biệt logic ứng dụng thành các phần nhỏ, dễ quản lý và bảo trì. Nó cũng thúc đẩy tái sử dụng mã nguồn, khiến cho quá trình phát triển và bảo trì ứng dụng trở nên hiệu quả hơn.

### **Tìm hiểu các thành phần trong mô hình MVC**

Mô hình MVC bao gồm ba thành phần chính: Model, View, và Controller. Dưới đây là mô tả chi tiết về mỗi thành phần:

**Model:**

**Mô tả:** Model đại diện cho dữ liệu và logic kinh doanh của ứng dụng. Nó là nơi lưu trữ thông tin, thực hiện các thao tác cập nhật và truy vấn dữ liệu. Model không quan tâm đến cách dữ liệu được hiển thị hoặc tương tác với người dùng.

**Trách nhiệm:**

Lưu trữ và quản lý dữ liệu.

Thực hiện logic kinh doanh và xử lý sự kiện.

Cung cấp API cho các thành phần khác để truy xuất và cập nhật dữ liệu.

**View:**

**Mô tả:** View là thành phần chịu trách nhiệm hiển thị thông tin từ Model cho người dùng. Nó không có logic kinh doanh và chỉ tập trung vào việc hiển thị dữ liệu một cách đẹp mắt và dễ hiểu.

**Trách nhiệm:**

Hiển thị dữ liệu cho người dùng.

Tương tác với người dùng thông qua giao diện người dùng.

Thông báo về các sự kiện tương tác đến Controller.

**Controller:**

**Mô tả:** Controller là thành phần điều phối và xử lý sự kiện từ người dùng. Nó nhận lệnh từ người dùng thông qua View, sau đó cập nhật Model dựa trên những thay đổi này và điều hướng hiển thị trở lại View.

**Trách nhiệm:**

Nhận các sự kiện từ View.

Gửi các yêu cầu cập nhật hoặc truy vấn dữ liệu đến Model.

Cập nhật View dựa trên sự thay đổi trong Model.

### **Phân tích luồng xử lý trong MVC**

Luồng xử lý trong mô hình MVC mô tả cách các thành phần chính (Model, View, và Controller) tương tác với nhau để xây dựng và duy trì ứng dụng. Dưới đây là một phân tích chi tiết về luồng xử lý trong MVC:

**Người dùng tương tác:**

Người dùng tương tác với giao diện người dùng (View) thông qua các sự kiện như nhấn nút, nhập liệu, hoặc thao tác khác.

**Controller nhận sự kiện:**

Controller (C) nhận và xử lý sự kiện từ giao diện người dùng (View).

Nhiệm vụ của Controller là điều phối các sự kiện và xác định cách ứng dụng phản ứng với hành động của người dùng.

**Controller gửi yêu cầu đến Model:**

Sau khi xử lý sự kiện, Controller gửi yêu cầu đến Model để thực hiện các thao tác cập nhật hoặc truy vấn dữ liệu.

Model xử lý yêu cầu và thực hiện các hoạt động liên quan đến dữ liệu.

**Model cập nhật dữ liệu:**

Model cập nhật dữ liệu dựa trên yêu cầu từ Controller.

Nếu có thay đổi trong dữ liệu, Model thông báo về sự thay đổi này cho các thành phần khác thông qua các cơ chế thông báo hoặc sự kiện.

**Model thông báo về sự thay đổi:**

Model thông báo về sự thay đổi trong dữ liệu đến Controller và View thông qua cơ chế quảng bá (broadcast) hoặc callback.

**Controller cập nhật View:**

Controller nhận thông báo từ Model và quyết định cách cập nhật giao diện người dùng (View) dựa trên sự thay đổi này.

Controller gửi các thông tin cần thiết đến View để cập nhật hiển thị.

**View hiển thị dữ liệu mới:**

View nhận thông tin từ Controller và cập nhật giao diện người dùng để hiển thị dữ liệu mới.

View không thực hiện logic kinh doanh, chỉ làm nhiệm vụ hiển thị dữ liệu.

Quá trình này lặp lại mỗi khi có sự kiện từ người dùng. Sự tách biệt rõ ràng giữa các thành phần giúp dễ dàng bảo trì, tái sử dụng mã nguồn, và cải thiện khả năng mở rộng của ứng dụng. Đồng thời, nó cung cấp một cấu trúc tổ chức mà mỗi thành phần chỉ chịu trách nhiệm cho một khía cạnh cụ thể của ứng dụng.

### **Ưu và nhược điểm của MVC**

Mô hình MVC (Model-View-Controller) mang lại nhiều ưu điểm trong phát triển phần mềm, nhưng cũng có nhược điểm cần xem xét. Dưới đây là một phân tích về ưu và nhược điểm của MVC:

**Ưu điểm của MVC:**

**Tách biệt logic:**

MVC giúp tách biệt logic ứng dụng thành các phần riêng biệt (Model, View, và Controller), làm cho mã nguồn dễ hiểu và dễ bảo trì.

**Tái sử dụng mã nguồn:**

Các thành phần trong MVC có thể được tái sử dụng một cách linh hoạt. Model và View có thể được sử dụng lại trong nhiều phần của ứng dụng hoặc trong các ứng dụng khác nhau.

**Phân chia công việc:**

Mô hình này giúp phân chia công việc giữa các thành viên trong nhóm phát triển, với người làm Model tập trung vào logic xử lý dữ liệu, người làm View chịu trách nhiệm về giao diện người dùng, và người làm Controller quản lý sự tương tác.

**Dễ mở rộng:**

Do sự tách biệt giữa các thành phần, việc mở rộng và thay đổi ứng dụng trở nên dễ dàng hơn. Một thành phần có thể được thay thế hoặc mở rộng mà không làm ảnh hưởng đến các thành phần khác.

**Kiểm thử dễ dàng:**

Sự tách biệt giữa Model, View và Controller giúp kiểm thử tốt hơn, vì bạn có thể kiểm thử mỗi thành phần một cách độc lập, giảm rủi ro gây ảnh hưởng lẫn nhau.

**Nhược điểm của MVC:**

**Phức tạp cho ứng dụng nhỏ:**

Đối với các ứng dụng nhỏ hoặc đơn giản, việc triển khai mô hình MVC có thể quá phức tạp và làm tăng độ phức tạp của mã nguồn.

**Tăng độ phức tạp của mã nguồn:**

Trong một số trường hợp, việc chia nhỏ ứng dụng thành nhiều thành phần có thể làm tăng độ phức tạp của mã nguồn và làm cho mã trở nên khó hiểu.

**Khó khăn trong việc đảm bảo sự đồng bộ:**

Đôi khi có thể khó khăn để đảm bảo sự đồng bộ giữa Model và View, đặc biệt là trong các ứng dụng lớn và phức tạp.