

ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**~~~~~~\*~~~~~~**

**BÁO CÁO CÁ NHÂN**

**KIẾN TRÚC PHÂN LỚP NGUYÊN KHỐI**

**CỦA DJANGO**

**Sinh viên thực hiện:**

Trần Đồng Gia Hân 3121411066

**Lớp: DCT121C3**

Giảng viên hướng dẫn: TS. Đỗ Như Tài

*TP. Hồ Chí Minh, tháng 03 năm 2025*

# **MỤC LỤC**

[**MỤC LỤC** 1](#_Toc194098470)

[**1. Giới thiệu về kiến trúc phân lớp nguyên khối** 3](#_Toc194098471)

[**1.1. Khái niệm** 3](#_Toc194098472)

[**1.2. Sơ đồ minh họa** 3](#_Toc194098473)

[**2. Triển khai kiến trúc MTV trong Django** 3](#_Toc194098474)

[**2.1. Cài đặt Django** 3](#_Toc194098475)

[**2.2. Xây dựng Model (M)** 3](#_Toc194098476)

[**2.3. Xây dựng View (V)** 3](#_Toc194098477)

[**2.4. Xây dựng Template (T)** 3](#_Toc194098478)

[**3. Ưu điểm và nhược điểm của kiến trúc phân lớp nguyên khối** 3](#_Toc194098479)

[**3.1. Ưu điểm** 3](#_Toc194098480)

[**3.2. Nhược điểm** 3](#_Toc194098481)

[**4. So sánh với kiến trúc Microservices** 3](#_Toc194098482)

[**5. Ứng dụng thực tế** 4](#_Toc194098483)

[**6. Kết luận** 4](#_Toc194098484)

## **1. Giới thiệu về kiến trúc phân lớp nguyên khối**

**1.1. Khái niệm**

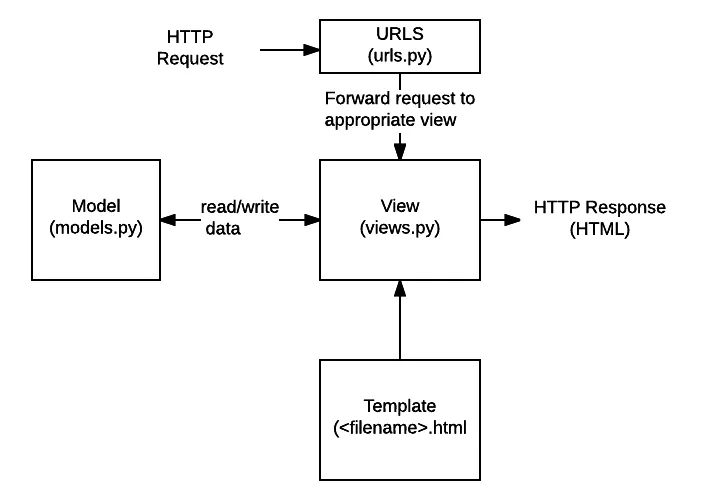
Kiến trúc phân lớp nguyên khối (Monolithic Layered Architecture) là mô hình thiết kế hệ thống phần mềm trong đó toàn bộ chức năng được tổ chức trong một đối tượng duy nhất.

Django sử dụng mô hình MTV (Model - Template - View), một dạng biến thể của kiến trúc phân lớp nguyên khối, trong đó:

* Model (M): Quản lý dữ liệu và giao tiếp với cơ sở dữ liệu.
* Template (T): Quản lý giao diện người dùng.
* View (V): Xử lý yêu cầu từ người dùng và kết nối Model và Template.

**1.2. Sơ đồ minh họa**

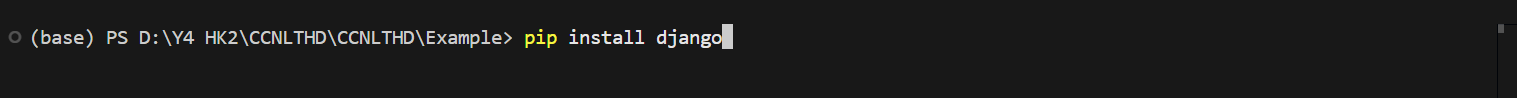
Dưới đây là sơ đồ minh họa các thành phần cơ bản của một ứng dụng Django:



**2. Triển khai kiến trúc MTV trong Django**

**2.1. Cài đặt Django**

Trước tiên, chúng ta cần cài đặt Django bằng câu lệnh pip install Django trong Terminal

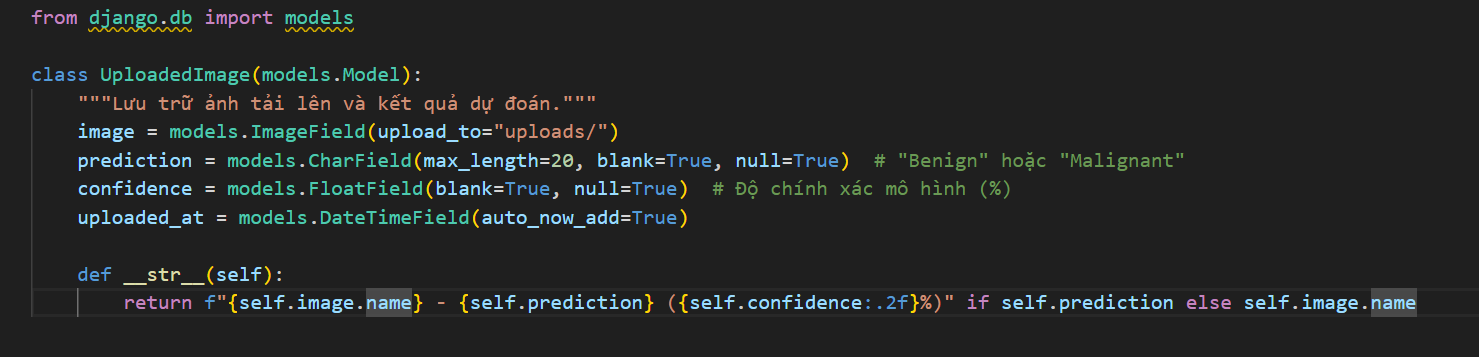


**2.2. Xây dựng Model (M)**

Vai trò:

* Quản lý dữ liệu của ứng dụng.
* Giao tiếp với cơ sở dữ liệu (SQLite, PostgreSQL, MySQL, v.v.).
* Định nghĩa bảng và các trường dữ liệu.

Tạo Model trong detect/models.py:

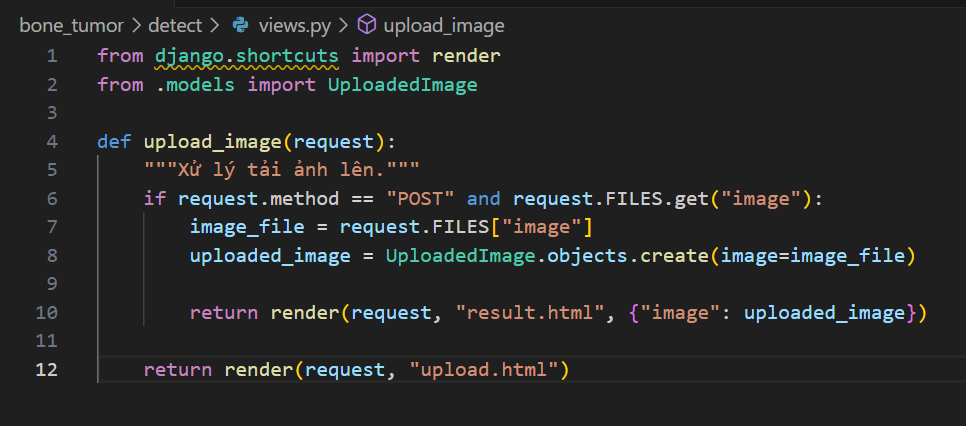


**2.3. Xây dựng View (V)**

Vai trò:

* Xử lý yêu cầu từ người dùng (nhận file, chạy mô hình AI, lưu kết quả).
* Lấy dữ liệu từ Model.
* Trả dữ liệu về Template để hiển thị.

Tạo View trong detect/views.py

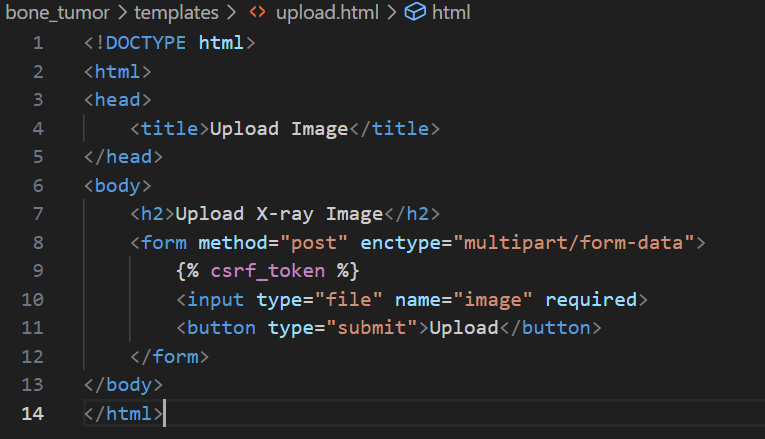


**2.4. Xây dựng Template (T)**

Vai trò:

* Hiển thị dữ liệu từ View.
* Dùng ngôn ngữ HTML + Django Template Language (DTL) để hiển thị dữ liệu động.

Tạo file templates/upload.html:



**3. Ưu điểm và nhược điểm của kiến trúc phân lớp nguyên khối**

**3.1. Ưu điểm**

* Dễ phát triển ban đầu: Vì tất cả thành phần nằm trong cùng một ứng dụng nên dễ dàng xây dựng và triển khai.
* Dễ dàng bảo trì: Khi ứng dụng còn nhỏ, việc sửa lỗi hoặc thêm tính năng có thể thực hiện nhanh chóng.
* Hiệu suất tốt trong hệ thống nhỏ: Do không có nhiều dịch vụ phân tán, việc trao đổi dữ liệu diễn ra nhanh chóng.
* Đơn giản hóa việc triển khai: Chỉ cần deploy một ứng dụng duy nhất mà không phải quản lý nhiều service khác nhau.

**3.2. Nhược điểm**

* Khó mở rộng: Khi hệ thống lớn, việc thêm tính năng mới có thể làm ứng dụng trở nên cồng kềnh và khó bảo trì.
* Khả năng chịu tải kém: Khi lượng người dùng tăng cao, kiến trúc nguyên khối khó mở rộng theo chiều ngang.
* Khó áp dụng công nghệ mới: Vì tất cả các thành phần liên kết chặt chẽ, việc thay đổi công nghệ trở nên khó khăn.
* Rủi ro khi deploy: Nếu một lỗi xuất hiện, nó có thể làm ảnh hưởng toàn bộ ứng dụng thay vì chỉ một phần nhỏ.

**4. So sánh với kiến trúc Microservices**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Monolithic Architecture** | **Microservices Architecture** |
| Cấu trúc | Tất cả chức năng nằm trong một khối duy nhất | Chia thành nhiều dịch vụ nhỏ, mỗi dịch vụ đảm nhận một nhiệm vụ riêng |
| Khả năng mở rộng | Hạn chế khi mở rộng hệ thống lớn | Dễ mở rộng theo từng dịch vụ riêng lẻ |
| Triển khai | Đơn giản, dễ triển khai | Phức tạp hơn do cần quản lý nhiều dịch vụ |
| Độ phức tạp | Ít phức tạp, dễ phát triển ban đầu | Yêu cầu nhiều công nghệ, khó quản lý |
| Khả năng bảo trì | Khó bảo trì khi hệ thống lớn | Dễ dàng thay đổi hoặc nâng cấp từng phần |

**5. Ứng dụng thực tế**

Kiến trúc phân lớp nguyên khối thường được sử dụng trong:

* Ứng dụng quản lý doanh nghiệp nhỏ: Các hệ thống ERP đơn giản, quản lý khách hàng, quản lý kho hàng.
* Blog và website cá nhân: Những trang web nhỏ không yêu cầu khả năng mở rộng lớn.
* Ứng dụng nội bộ doanh nghiệp: Những hệ thống chỉ phục vụ một tổ chức và không cần mở rộng quy mô lớn.

**6. Kết luận**

Kiến trúc phân lớp nguyên khối trong Django rất phù hợp cho các dự án nhỏ và vừa, đặc biệt là trong giai đoạn đầu phát triển. Tuy nhiên, khi hệ thống mở rộng, ta cần cân nhắc chuyển sang các mô hình linh hoạt hơn như Microservices để tăng khả năng mở rộng và bảo trì. Django với mô hình MTV giúp tổ chức mã nguồn tốt, dễ dàng quản lý và phát triển, tuy nhiên cũng có những hạn chế cần cân nhắc khi hệ thống phát triển lớn hơn.

Ngoài ra, nếu dự án có nhu cầu mở rộng hoặc phục vụ số lượng người dùng lớn, có thể kết hợp Django với các công nghệ khác như **Docker, Kubernetes** để triển khai theo hướng **Microservices**, giúp tối ưu hiệu suất và khả năng mở rộng.