**THIẾT KẾ KIẾN TRÚC HỆ THỐNG ĐƯỢC SỬ DỤNG**

1. Giới thiệu về kiến trúc hệ thống được sử dụng trong dự án

Trang web được xây dựng dựa trên mô hình Model-View-Controller (MVC) kết hợp mô hình 3 lớp.

1. Mô hình MVC

Mô hình MVC là một mẫu kiến trúc phân tách một ứng dụng thành ba thành phần logic chính Model, View, Controller. Mỗi thành phần kiến trúc được xây dựng để xử lý khía cạnh phát triển cụ thể của một ứng dụng. MVC tách lớp logic nghiệp vụ và lớp hiển thị ra riêng biệt.

* 1. Model trong MVC

Model chứa một cấu trúc dữ liệu có chức năng lưu trữ toàn bộ các thông tin dữ liệu của một ứng dụng. Trong mô hình MVC, Model đóng vai trò kết nối cho hai thành phần View và Controller. Model có chức năng vận chuyển thông tin từ nội hàm để hiển thị đến người dùng thông qua màn hình và xử lý các thông tin để người dùng dễ dàng tiếp cận nhất.

Model hoàn toàn độc lập với các thành phần còn lại trong MVC và nó chứa các tác vụ cần thiết nhất cho quá trình lập trình .

* 1. View trong MVC

View là thành phần liên quan đến giao diện mà người dùng trải nghiệm. Thông qua View, người dùng sẽ thực hiện các thao tác tìm kiếm, sử dụng thông tin trên trang web.

View thực hiện nhiệm vụ tạo tương tác với người dùng và hiển thị các kết quả từ tầng Controller. Đồng thời, View cũng thực hiện việc tiếp nhận các hoạt động, yêu cầu của người dùng để chuyển đến Controller xử lý.

* 1. Controller

Controller thực hiện chức năng kết nối tương tác giữa View và Model. Nó định nghĩa các lệnh và thực hiện xử lý các lệnh trong hệ thống. Controller đối chiếu hành động của người dùng từ View và tương tác với Model để chuyển tải thông tin cần thiết đến người dùng.

* 1. Luồng đi trong mô hình MVC
* Khi có một yêu cầu gửi từ phía khách hàng tới server, bộ phận Controller sẽ có nhiệm vụ nhận yêu cầu và xử lý yêu cầu đó. Nếu cần, nó sẽ gọi đến Model.
* Sau khi xử lý xong, toàn bộ kết quả sẽ được đẩy về phần View.
* View sẽ hiển thị lại thông tin đã xử lý cho người dùng.

1. Mô hình 3 lớp

Mô hình 3 lớp hay còn được gọi là mô hình Three Layer(3-Layer), mô hình này ra đời nhằm phân chia các thành phần trong hệ thống, các thành phần cùng chức năng sẽ được nhóm lại với nhau và phân chia công việc cho từng nhóm để dữ liệu không bị chồng chéo và chạy lộn xộn.

Mô hình này phát huy hiệu quả nhất khi bạn xây dựng một hệ thống lớn, việc quản lý code và xử lý dữ liệu lỗi dễ dàng hơn.

* 1. Thành phần
     1. Presentation Layer (GUI)

Lớp này có nhiệm vụ chính là giao tiếp với người dùng. Nó gồm các thành phần giao diện ( winform, webform, …) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).

* + 1. Business Logic Layer (BLL)

Layer này phân ra 2 thành nhiệm vụ:

Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.

Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

* + 1. Data Access Layer (DAL)

Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu ( tìm kiếm, thêm, xóa, sửa,…).

Lý do chọn mô hình MVC

* Đầu tiên, nhắc tới ưu điểm mô hình MVC thì đó là băng thông nhẹ vì không sử dụng viewstate nên khá tiết kiệm băng thông. Việc giảm băng thông giúp website hoạt động ổn định hơn.
* Kiểm tra đơn giản và dễ dàng, kiểm tra lỗi phần mềm trước khi bàn giao lại cho người dùng.
* Một lợi thế chính của MVC là nó tách biệt các phần Model, Controller và View với nhau.
* Sử dụng mô hình MVC chức năng Controller có vai trò quan trọng và tối ưu trên các nền tảng ngôn ngữ khác nhau
* Ta có thể dễ dàng duy trì ứng dụng vì chúng được tách biệt với nhau.
* Có thể chia nhiều developer làm việc cùng một lúc. Công việc của các developer sẽ không ảnh hưởng đến nhau.
* Hỗ trợ TTD (test-driven development). Chúng ta có thể tạo một ứng dụng với unit test và viết các won test case.
* Phiên bản mới nhất của MVC hỗ trợ trợ thiết kế responsive website mặc định và các mẫu cho mobile. Chúng ta có thể tạo công cụ View của riêng mình với cú pháp đơn giản hơn nhiều so với công cụ truyền thống.

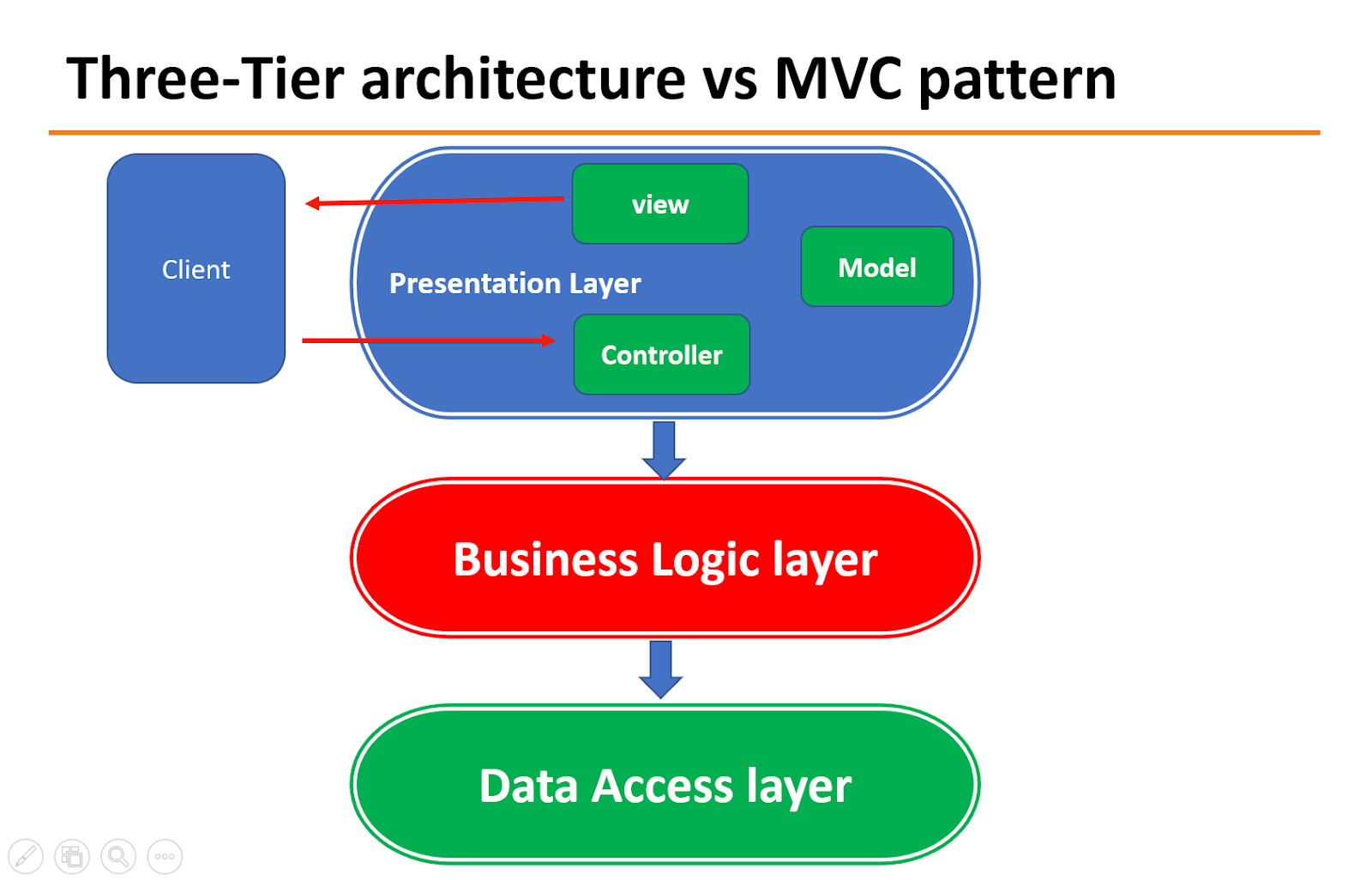
Tuy nhiên, MVC chỉ mô tả luồng đi của dữ liệu, nó không nói rõ như code đặt ở đâu (ở Model, View hay Controller), rồi lưu trữ Model vào database kiểu gì,... Do đó, đối với ứng dụng hoàn chỉnh cần kết hợp cả mô hình MVC và 3 lớp lại với nhau.

1. Công nghệ sử dụng

* Công nghệ được sử dụng trong project chính là Spring Boot.
* Spring Boot là gì?

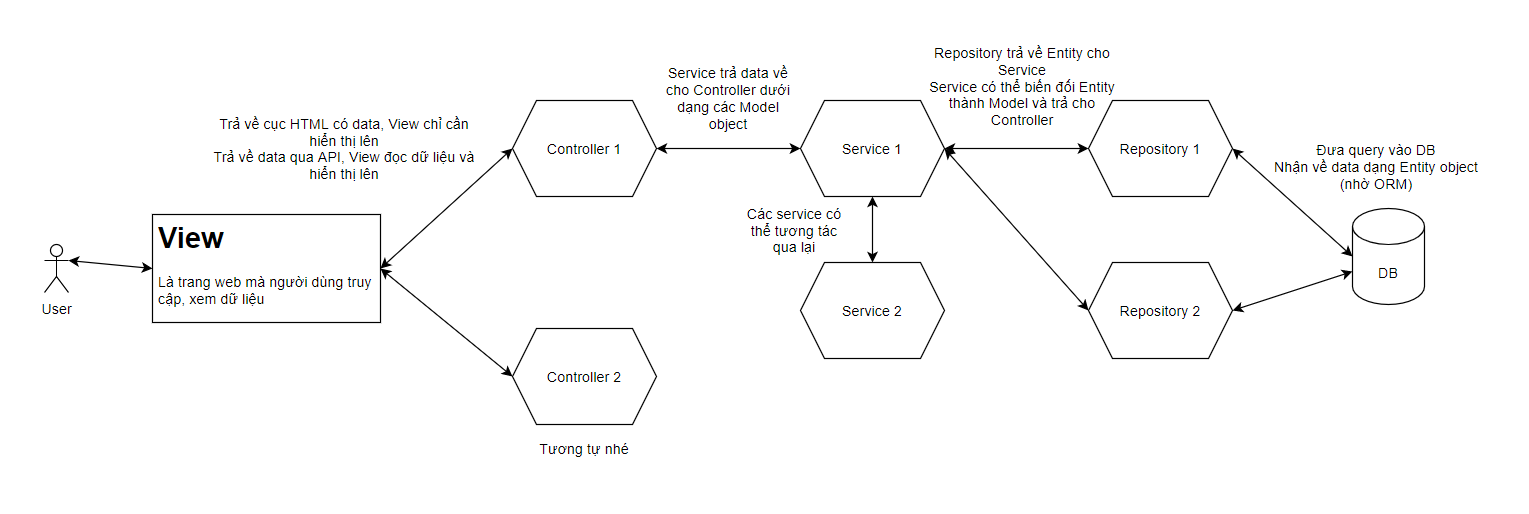
Spring Boot là một framework vi mô mã nguồn mở được duy trì bởi Pivotal. Framework này cung cấp cho các nhà phát triển Java một nền tảng để bắt đầu với một ứng dụng Spring cấu hình tự động. Với nó, các nhà phát triển có thể bắt đầu nhanh chóng mà không mất thời gian chuẩn bị và cấu hình ứng dụng Spring của họ.

Cấu trúc source code của Spring Boot được dựa trên hai mô hình là mô hình MVC và mô hình 3 lớp.



Kết hợp hai mô hình lại, chúng ta có được ứng dụng Spring Boot hoàn chỉnh, gồm các thành phần sau:

* Controller: trả về View (có chứa data sẵn, dạng trang HTML), hoặc Model thể hiện dưới dạng API cho View (View viết riêng bằng React, Vue, hoặc Angular).
* Service: chứa các code tính toán, xử lý. Khi Controller yêu cầu, thì Service tương ứng sẽ tiếp nhận và cho ra dữ liệu trả cho Controller (trả về Model). Controller sẽ gửi về View như trên.
* Repository: Service còn có thể tương tác với service khác, hoặc dùng Repository để gọi DB. Repository là thằng trực tiếp tương tác, đọc ghi dữ liệu trong DB và trả cho service.
* Model chỉ đơn giản là các đối tượng được Service tính toán xong trả về cho Controller.
* View thì có 2 loại, một là dạng truyền thống là trả về 1 cục HTML có data rồi. Lúc này Controller sẽ pass dữ liệu vào View và return về (Spring MVC có JSP hoặc template engine như Thymeleaf làm điều đó).
* View dạng 2 là dạng View tách riêng (không thuộc về project Spring boot). Thường có trong các hệ thống dùng API. View sẽ được viết riêng bằng React, Angular,... Controller sẽ đưa dữ liệu Model thông qua API cho View, và cũng nhận lại các yêu cầu qua API luôn.



(Sơ đồ luồng đi)

* Lý do chọn Spring Boot:
* Spring Boot sẽ tự động cấu hình, gần như mọi thứ đã trở nên dễ dàng và giúp người lập trình tập trung vào việc code.
* Spring Boot giúp quản lý từng Microservice một cách dễ dàng.
* Annotation Based Configuration là một tính năng đắc lực hỗ trợ tạo lập bean thay vì phải XML.
* Servlet sẽ được nhúng sẵn trong Spring Boot, nên có thể bật và chạy Server Tomcat dễ dàng hơn.
* Spring Boot rất linh hoạt để có thể chạy cấu hình Java Beans, XML và kể cả là Database Transaction (giao dịch hoặc trao đổi).
* Gia tăng được năng suất trong quá trình lập trình.
* Giảm ở mức tối thiểu thời gian lập trình.
* Giúp người dùng mặc dù không có nhiều kiến thức lập trình vẫn có thể xây dựng được một ứng dụng.