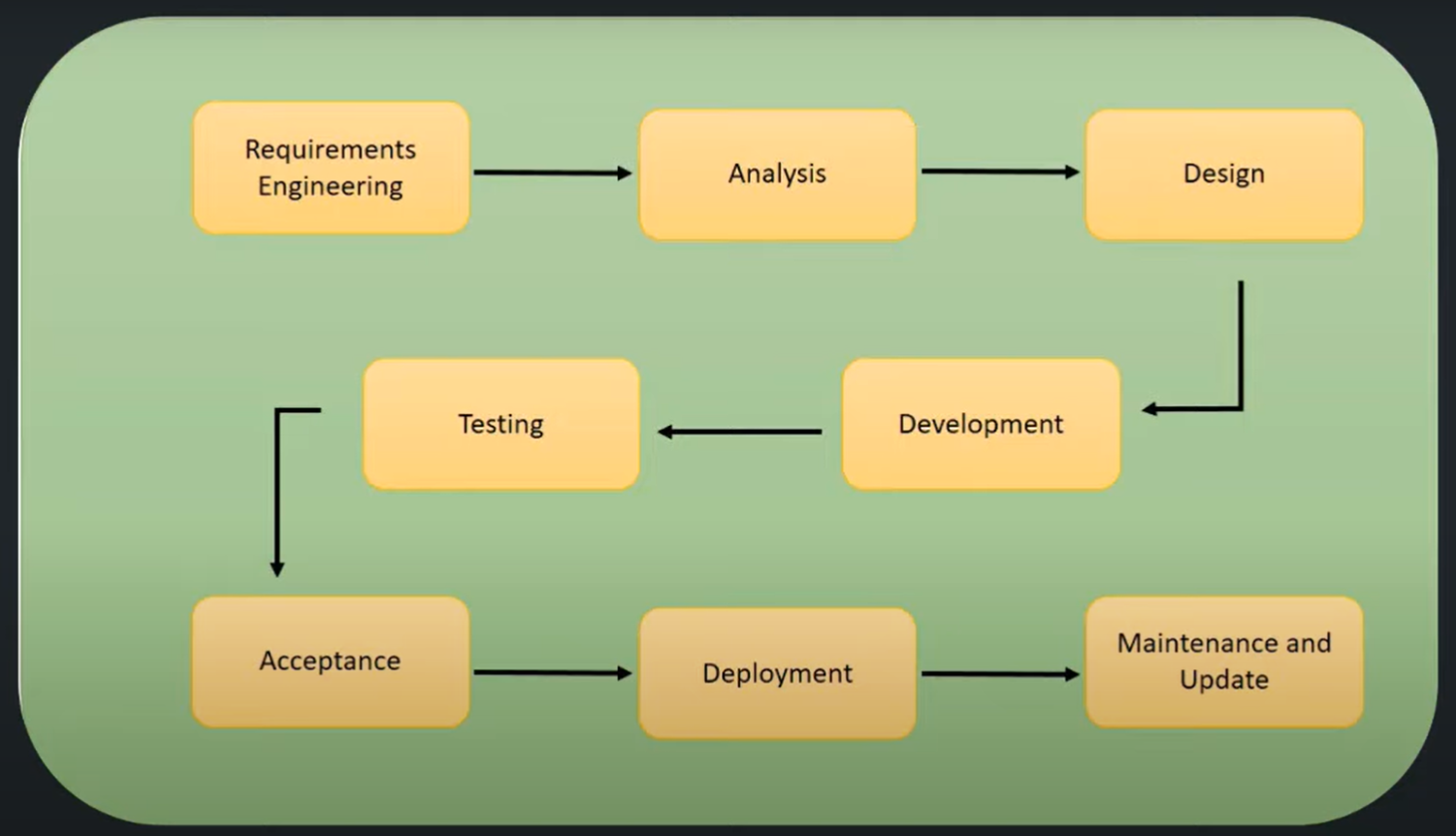
**Day 1**

1. Architecture Concepts

[App Architecture - Understanding Frontend, Backend and Web Servers (youtube.com)](https://www.youtube.com/watch?v=d1Gd-MGaleE&list=PLUU3EzfPr915ebZONvUVHKm8Bls6D7EgA)

[(79) App Architecture - Understanding Frontend, Backend and Web Servers - YouTube](https://www.youtube.com/watch?v=d1Gd-MGaleE&list=PLUU3EzfPr915ebZONvUVHKm8Bls6D7EgA)

* Quy trình phát triển phần mềm (Software Development Life Cycle – SDLC): là chuỗi các hành động được thực hiện theo một thứ tự nhất định để xây dựng và cung cấp một sản phẩm có thể đáp ứng được yêu cầu kỹ thuật và phục vụ cho nhu cầu của khách hàng.



* Quy trình chung của SDLC gồm 8 giai đoạn hoặc 6 giai đoạn(gộp 1+2; 5+6):
* Requirements engineering: là những yêu cầu của phía bên khách hàng về phần mềm
* Analysis: thu thập và xác định rõ các yêu cầu của người dùng và các bên liên quan. Nghiên cứu thị trường giúp xác định các chức năng cần cung cấp cho người dung và khả năng tồn tại của phần mềm trên thị trường. Kết quả sẽ được tổng hợp thành **tài liệu đặc tả yêu cầu phần mềm**.
* Design: Dựa trên yêu cầu và thông số kỹ thuật, tạo kiến trúc tổng thể cần thiết để tạo phần mềm
* Development: Tạo mã nguồn phần mềm dưa trên thiết kế đã xác định
* Testing: Kiểm tra phần mềm để đảo bảo tính đúng đắn và hiệu quả
* Acceptance: Các thử nghiệm đã được thông qua, hệ thống hoạt động tốt sẵn sàng deloy
* Deploy: Đưa phần mềm vào hoạt động
* Maintenance: duy trì và cải tiến phần mềm sau khi triển khai
* Có 4 mô hình phát triển phần mềm thường được sử dụng: Waterfall model (thác nước), V model, Spiral model (xoắn ốc), Agile and Scrum.
* Web Architecture:

A diagram of a computer system

Description automatically generated

* Laravel sử dụng kiến trúc MVC (Model – View – Controller)

A diagram of a system

Description automatically generated

A diagram of a computer system

Description automatically generated

+ Model : tương tác trực tiếp với dữ liệu. Model giúp truy xuất dữ liệu từ cơ sở dữ liệu và sau đó thực hiện một số thao tác mà ứng dụng phải thực hiện, sau đó lưu trữ dữ liệu đó lại cơ sở dữ liệu. Nói tóm lại, model là nơi chịu trách nhiệm quản lí dữ liệu được truyền từ cơ sở dữ liệu và giao diện người dùng (view). Tức là phần Database

+ View: Là giao diện người dùng, chứa mọi thứ mà người dùng có thể mình thấy trên máy tính. Tức là phần front-end.

+ Controller: Là nơi xử lí các yêu cầu người dùng thông qua view. Là cầu nối của Model và View. Controller giúp tương tác với component model để lấy dữ liệu từ database, rồi chuyển dữ liệu đó về view component để có được output mong muốn và được hiển thị trên màn hình View. Tương tự, khi người dùng thực hiện các thao tác trên màn hình view, sẽ tạo một số dữ liệu, Controller sẽ tìm nạp dữ liệu đó, sau đó sẽ thực hiện một số thao tác hoặc chèn dữ liệu đó vào cở sở dữ liệu Model

Luồng xử lí trong MVC:

+ Khi một request từ client gửi đến server. Thì bị Controller chặn lại để xem đó là URL request hay sự kiện.

+ Sau đó, Controller xử lí input của user rồi giao tiếp với Model.

+ Model chuẩn bị data và gửi lại cho Controller.

+ Cuối cùng, khi xử lí xong request thì Controller gửi lại dữ liệu trở về View và hiển thị cho người dùng trên trình duyệt.

Ưu, nhược điểm MVC??

1. Request lifecycle

* Request là gì?

[Kiến trúc hệ thống trên Laravel – phần 6 (viblo.asia)](https://viblo.asia/p/kien-truc-he-thong-tren-laravel-phan-6-L4x5xQXOKBM)

[Kiến trúc hệ thống trên Laravel – phần 9 (viblo.asia)](https://viblo.asia/p/kien-truc-he-thong-tren-laravel-phan-9-RQqKLM4bZ7z)

[Khoá học Laravel Framework 8.x - Bài 2: Cấu trúc thư mục - Vòng đời Request trong Laravel (youtube.com)](https://www.youtube.com/watch?v=Cm0YBC8YNtw)

Vòng đời của request:

Khi client gửi một request đến server. Đầu tiên, request sẽ gửi đến publish/index.php, sau đó chuyển đến app/http/Kernel.php (đây được gọi là trung tâm request, nơi chứa tất cả các request từ client trước khi request được sử lí). Sau đó, request được chuyển đến Service provider ( đây chính là trung tâm khởi đông để chạy các service). Trước khi request được điều hướng đến View hay Controller cụ thể nào đó, nó phải thông qua Router. Sau đó , request sẽ qua Middleware, đây là nơi kiểm tra xem request có thỏa mãn điều kiện hay không, nếu có sẽ cho đi tiếp đến Controller hoặc không thì sẽ dừng lại hoặc chuyển hướng sang một thứ khác. Request khi đến COntroller sẽ có 2 hướng: 1 là sẽ đến view rồi trả kết quả cho client; 2 là sẽ trả trực tiếp kết quả cho client.

A diagram of a lifecycle

Description automatically generated

* Container là gì?

[(2) Cộng đồng Lập Trình Web • Front-end & Back-end Việt Nam | Một số lý thuyết và khái niệm để các bạn có thể sử dụng \*\*Docker \*\*<3 | Facebook](https://www.facebook.com/groups/laptrinhwebvietnam/posts/3495514094095369/)

Container được hiểu như một dạng thùng chứa, bao gồm tất cả các thành phần cần thiết như: source code, runtime, hệ điều hành,..để có thể chạy được một ứng dụng ở mọi môi trường. Container là một phần quan trọng của lập trình web, giúp tạo ra môi trường đồng nhất và dễ di chuyển giữa các môi trường.

Docker là một nền tảng containerization cho phép đóng gói, phân phối, chạy ứng dụng trong các container cô lập và di động. Nói dễ hiểu, docker là một nền tảng open-source giúp các devs có thể build, deploy, run và update ứng dụng cuat mình bên trong các containers. Nhờ docker, tất cả các ứng dụng có thể được đóng gói vào các container, giúp tiện lợi cho việc deploy, chạy trên các môi trường khác nhau.

Ưu điểm của Container

* Nhẹ và hiệu quả: Containers tiết kiệm tài nguyên hơn so với máy ảo (VMs) vì chúng chia sẻ kernel hệ điều hành.
* Di động và dễ triển khai: Container có thể di chuyển giữa các môi trường mà không cần thay đổi mã nguồn

1. Service Container

Service Container là một phần quan trọng của hệ thống, giúp quản lí các phụ thuộc ̣(dependencies) của các class và thực hiện dependency injection. Dùng để quản lí việc tạo và phân phối các service trong ứng dụng.

Dependency injection: cho phép các dependences được tiêm vài class thông qua constructor hoặc qua các phương thức setter.

Lợi ích của Service Container:

* Tự động giải quyết phụ thuộc: nếu một class không có phụ thuộc hoặc chỉ phụ thuộc vào class cụ thể ̣(không phải interface), SC tự động giải quyết phụ thuộc đó.
* Tiện lợi và không cần cấu hình: các class như Controller, even listener, middleware và nhiều class khác trong Laravel tự động nhận phụ thuộc từ Service Container

1. Service Providers

Service providers(nhà cung cấp dịch vụ) là một khái niệm quan trọng khi sử dụng dịch vụ các framework như là laravel. Giúp khởi tạo và đăng kí các service trong ứng dụng. Được sử dụng để quản lí và cấu hình các dịch vụ, gấn kết các class, thực hiện các tác vụ khởi tạo.

Chức năng của Service Providers:

* Đăng kí service: SP cho phép đăng kí các dịch vụ vào container của Laravel
* Cấu hình ứng dung: cấu hình các dịch vụ, middleware và các tùy chọn khác thông qua SP
* Tạo các singleton: SP giúp tạo ra các singleton, đảm bảo chỉ có một phiên bản duy nhất của class được tạo ra.

Trong Laravel đã tích hợp sẵn một số Service Providers, nhưng mình có thể tạo riêng cho ứng dụng. Cú pháp để tạo một Service Providers, sử dụng terminal:

**php artisan make:provider MyServiceProvider**

1. Facades

[Facades - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/facades)

Facades giúp tạo ra một giao diện đơn giản để truy cập các đối tượng từ container dịch vụ của ứng dụng. Facades cung cấp một giao diện tĩnh cho các class có sẵn trong container dịch vụ của Laravel, giúp viết mã ngắn gọn và dễ nhớ mà không cần nhớ tên dài của các class.

Lợi ích của facades:

* Cung cấp cú pháp ngắn gọn và dễ nhớ
* Dễ kiểm thử vì không cần inject hoặc cấu hình các class thủ công
* Tuy nhiên, cần chú ý không để class trở nên quá lớn khi sử dụng quá nhiều facades.

Nói chung, facades giúp viết mã ngắn gọn và dễ nhớ khi truy cập các tính năng của framework mà không cần phải nhớ đến tên dài của các class

A computer screen shot of a program code

Description automatically generated

Cú pháp của facades**: Illuminate\Support\Facades\Tên\_class.**

1. Application structure: Cấu trúc thư mục của Laravel 11

[Cấu trúc project Laravel | Laravel | Hướng dẫn học | Học web chuẩn (hocwebchuan.com)](https://hocwebchuan.com/tutorial/laravel/laravel_project_structure.php)

[Dive into the Streamlined Directory Structure in Laravel 11 - Laravel News (laravel-news.com)](https://laravel-news.com/laravel-11-directory-structure)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | * app: chứa tất cả cốt lõi trong ứng dụng. Hầu hết các class trong project được tạo tại đây. * http: * http/Controllers: Chứa Controller của project * http/Middleware: chứa các tập tin lọc và ngăn chặn các requests. * Models: chính là phần Model trong MVC * Providers: chứa các file thực hiện việc khai báo service và bind vào trong Service Container. * Console(Laravel\_10): chứa các tập tin định nghĩa các câu lệnh trên artisan * Exceptions(Laravel\_10): chứa các tập tin quản lí, điều hướng lỗi. * bootstrap: chứa những file khởi động của framework và những file cấu hình auto loading, route, file cache. * config: chứa tất cả những file cấu hình. * database: chứa 2 folder: migration (tạo và thao tác với database) và seeds (Tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này. * Factories: chứa các file định nghĩa các cộ bảng dữ liệu để tạo ra các dữ liệu mẫu * Migrations: chứa các file tạo và chỉnh sửa dữ liệu * Seeds: chứa các file tạo dữ liệu thêm vào CSDL * publish: chứa file index.php, là cổng cho tất cả các request vào project, bên trong thư mục còn chứa file JavaScript và CSS. * resources: chứa những file view và raw, các file biên soạn như LESS, SASS, JavaScript. Ngoài ra còn chứa tất cả file lang trong project. Folder views chính là View trong MVC * routes: chứa tất cả các điều khiển route (đường dẫn) trong project. * Api.php: điều khiển các route của ứng dụng, như route của ứng dụng User (đăng nhập, đăng kí) * Web.php: điều khiển các route của view, như route của trang sản phẩm,.. * Console.php: * Channels.php: * storage: chứa các file biên soạn blade template, gồm file based sessions, fide caches và những file sinh ra từ project. * tests: chứa những file test như PHPUnit test. * vendor: chứa các thư viện của Composer và các file core. * .env: chứa các config chính của Laravel. File cấu hình, thiết lập môi trường. * .gitattributes, .gitinore: file dành cho xử lí git. * artisan: file thực hiện lệnh của Laravel. * composer.json, composer.lock, composer-setup.php: file của composer * package.json: chứa các package cần dùng cho project. * phpunit.xml: của phpunit dùng để testing project * vite.config.js: |