**[1] Em biết gì về web?**

**- Hạ tầng web:** Bao gồm:

Web server, trình duyệt, giao thức HTTP/HTTPS, địa chỉ trang web, kết nối Internet

Web app

Web Server apache,nginx

HĐH (OS) tcp,ip

HardWare

**- Phân biệt webpage, website, web app?**

Webpage: HTML CSS JS

Website: có nhiều trang web được tổ chức theo logic

Web app: được tổ chức logic hơn, phức tạp hơn, dữ liệu phức tạp hơn.

**- Để làm web app?**

**(5 công cụ)**

**+ Ngôn ngữ:** Python, JavaScript, Java, C#, PHP

**+ Công cụ viết mã:** VSCode, Sublime Text, PyCharm

**+ Công cụ CSDL**: MySQL, PostgreSQL, Microsoft SQL Server, MongoDB, SQLite

**+ Quản lý mã nguồn:** Git, Github, Gitlab, BitBucket

**+ Web server:** Nginx, Apache, Node.js

Web server:

Phần cứng:

Phần mềm: NodeJS thi

Trình duyệt mã NodeJS: V8 (của gg), nhiều thư viện.

B1 Tải nodeJs trình duyệt mã V8

B2 Cài NPM (Node Package Manage) là phần mềm quản lý gói của Node có sẵn trong node và có 1 kho chứa Registry chứa các thư viện

---------------

**[2] NPM: (thành phần, local, global, dependencies, devDependencies)**

**(1) developer -> devDependencies**

**(2) production -> Dependencies**

* Node.js: môi trường thực thi JavaScript đa nền tảng, mã nguồn mở, và miễn phí, dùng để tạo web server, ứng dụng web và các công cụ dòng lệnh
* NPM: Công cụ quản lý các thư viện (package) trong ứng dụng Node.js..
* Thành phần của NPM:
* npm CLI: Công cụ dòng lệnh
* npm Registry: CSDL trực tuyến chứa các gói JavaScript mã nguồn mở
* package.json: Tệp cấu hình quan trọng nằm ở thư mục gốc của dự án

***Cài đặt Local và Global:***

* Local: Sử dụng trong dự án cài đặt đó
* Global: Sử dụng trong bất kì dự án nào trên hệ thống

Local vs Global được phân biệt bằng đường biên trong đường biên là local ngoài là global VD Đường biên là TeoShop thì Local trong TeoShop và Global là ngoài TeoShop.

**Câu hỏi:**

1. Một gói nằm trong Dependencies có dùng được trong quá trình phát triển (development) không? Giải thích và cho ví dụ ?

Dùng Dependencies sẽ dùng được DevDependencies. Ngược lại thì không

1. Tại sao họ lại chia thành Dependencies và DevDependencies ?

Package.json:

{

"name": "teoshop", // Tên gói

"version": "1.0.0", //Phiên bản hiện tại

"main": "index.js", //File JavaScript dùng để chạy ứng dụng

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

}, //Lệnh chạy bằng npm hoặc yarn

"keywords": [], //Mảng từ khoá liên quan đến gói

"author": "", //Tác giả của gói

"license": "ISC", //Giấy phép

"description": "", //Mô tả

"dependencies": { //Các gói phụ thuộc để dự án chạy được: Số phiên bản

"express": "^4.21.2",

"express-handlebars": "^8.0.1",

"pg": "^8.15.6",

"pg-hstore": "^2.3.4",

"sequelize": "^6.37.7"

}

}

1. **Một gói nằm trong dependencise có dùng được trong quá trình phát triển (development) không? giải thích và cho ví dụ?**

Có, vì dependencies liệt kê các gói mà ứng dụng *cần* để chạy. Điều này bao gồm cả những thư viện sử dụng để xây dựng các tính năng, tương tác với cơ sở dữ liệu, v.v. Những thư viện này cần phải có mặt trong cả môi trường phát triển để có thể viết code, chạy thử nghiệm và gỡ lỗi, cũng như trong môi trường production để ứng dụng hoạt động.

Ví dụ: ở file trên:

Express: Xây dựng server, xử lí các request, response

Express-handlebars: Dùng làm template engine hiển thị giao diện

Pg và pg-hstore: Kết nối và tương tác csdl PostgreSQL

Sequelize: ORM dùng để tương tác với CSDL

1. **Tại sao họ lại chia thành dependencies và devDependencies?**

Giúp quản lí các gói một cách hiệu quả hơn:

* dependencies chứa các gói **cần thiết để ứng dụng chạy** trong môi trường production
* devDependencies chứa các gói **chỉ cần thiết trong quá trình phát triển**, kiểm thử và xây dựng ứng dụng

**3. Một gói được cài đặt theo kiểu global có nằm trong package.json hay không? giải thích và cho ví dụ?**

Không, vì gói đó được cài đặt vào một thư mục hệ thống khác, chứ không nằm trong node\_modules. File package.json theo dõi các dependency *của riêng dự án đó*, ghi lại những gói cần thiết để build, chạy và test *dự án hiện tại*. Các gói global nằm ngoài phạm vi quản lý của từng package.json riêng lẻ.

* package.json quản lý các dependency cục bộ của một dự án, còn các gói cài đặt global thì được quản lý ở cấp hệ thống và không được theo dõi trong package.json của từng dự án.

VD: npm: Quản lí các gói được cài đặt global dùng lệnh npm install, không có trong package.json

**[3] Lập trình với Express**

'use strict'

const express = require('express')

const app = express();

const port = process.env.PORT || 9000

const expressHandlebars = require('express-handlebars');

app.use(express.static(\_\_dirname + '/public'));

// cấu hình để sử dụng express handlebars, định nghĩa engine

app.engine('hbs', expressHandlebars.engine({

// Thư mục chứa các layout

layoutsDir: \_\_dirname + '/views/layouts',

// Thư mục chứa các template con

partialsDir: \_\_dirname + '/views/partials',

// Đuôi của tập tin layout

extname: 'hbs',

// Tập tin layout chính

defaultLayout: 'main'

}));

// Khai báo để sử dụng engine đã định nghĩa

app.set('view engine', 'hbs');

// routes và render trang index

app.get('/createTables', (req, res) => {

let models = require('./models');

models.sequelize.sync().then(()=> {

res.send('tables created');

});

});

app.get('/', (req, res) => {

res.render('index'); // se lay index.hbs do vao {{{ body }}} trong main.hbs

});

// Hiển thị các trang con

app.get('/:page', (req, res) =>{

res.render(req.params.page);

});

// Khởi động web server

app.listen(port, () => {

console.log(`server dang chay tren cong ${port}`);

});

[4] Callback, hàm trong js

Hàm ẩn danh: Hàm không có tên khi định nghĩa, được gán vào biến, truyền làm tham số, hoặc thực thi ngay lập tức

Hàm mũi tên: Cú pháp ngắn gọn, không có ngữ cảnh riêng, không thể làm constructor, không có biến arguments (…) => {…}

Hàm callback: Hàm được truyền vào hàm khác như tham số, thực thi sau khi hàm cha hoàn thành tác vụ

Lập trình đồng bộ: Các lệnh được thực thi tuần tự từ trên xuống dưới, có thể chặn luồng khi tác vụ mất nhiều thời gian

Lập trình bất đồng bộ: Cho phép thực hiện nhiều tác vụ mà không cần chờ đợi

Các kĩ thuật cho LTBĐB:

* Callback: Hàm được thực thi khi tác vụ hoàn tất
* Promise: Kết quả cuối cùng của 1 tác vụ bất đồng bộ
* Async/await:Viết mã bất đồng bộ giống như đồng bộ, giúp dễ đọc. Trong đó, async đặt trước function, await đặt trước lời gọi hàm

[5]Lập trình Client - server

Ứng dụng cục bộ: Ứng dụng chạy trực tiếp trên thiết bị, không cần internet

Ứng dụng client-server: Ứng dụng mà trong đó client hiển thị, server xử lí logic và dữ liệu, cần kết nối mạng giữa chúng

Ứng dụng web là ứng dụng client-server, trong đó server là web server và client là trình duyệt

Vai trò:

* + **Front-end**: Thiết kế giao diện
  + **Back-end**: Xử lý logic và dữ liệu
  + **Full-stack**: Làm được cả hai

protocol://domain:port/path?query#fragment

Giải thích các thành phần của một URL:

- protocol (giao thức): xác định phương thức giao tiếp mà trình duyệt sử dụng để giao tiếp với server. Các giao thức phổ biến bao gồm http (Hypertext Transfer Protocol) và https (Hypertext Transfer Protocol Secure).

- domain (tên miền): tên của máy server. Ví dụ: google.com, wikipedia.org, localhost.

- port (cổng) (tùy chọn): số cổng mà server sử dụng để lắng nghe yêu cầu (request). Cổng mặc định cho http là 80 và cho https là 443.

- path (đường dẫn): đường dẫn đến một vị trí (thư mục, tập tin, API) trên server. Ví dụ: /images/logo.png, /articles/how-to-use-url, /users.

- query (truy vấn) (tùy chọn): chuỗi các tham số được sử dụng để truyền dữ liệu đến server. Ví dụ: ?name=shoes&size=40.

- fragment (phân mảnh) (tùy chọn): một phần cụ thể của tài nguyên, thường là một phần của trang HTML. Ví dụ: #section-2.

Như vậy, URL http://localhost:9000/?name=shoes&size=40

Có nghĩa là:

- Client sử dụng giao thức http để gửi request tới web server

- Web server có địa chỉ là localhost

- Gửi request vào cổng 9000 của web server

- Gửi request, có kèm theo 2 tham số là name=shoes và size=40

---- >>>> nâng cao >>>>>

[6]Lập trình hướng sự kiện

Khái niệm: Là mô hình lập trình mà luồng thực thi của chương trình được quyết địng bởi các sự kiện

Thành phần chính:

* Event: Sự kiện (nhấn chuột, gõ phím,…)
* Event Emitter: Bộ quản lí phát ra sự kiện gửi đến nơi cần nghe
* Event Handler: Hàm xử lí sự kiện, được gọi khi sự kiện xảy ra
* Event Loop: Vòng lặp sự kiện kiểm tra hàng đợi và xử lí callback

Mô hình sự kiện:

Event Emitter phát ra sự kiện, đưa vào hàng đợi (Events), Event Loop kiểm tra, lấy sự kiện chuyển cho Event Handler xử lí, Event Loop tiếp tục vòng lặp, chờ sự kiện tiếp theo

[7]CSDL

Dữ liệu thô: Chuỗi các kí hiệu được lưu dưới dạng bit 0 và 1 trong máy, không có ngữ nghĩa, cần được xử lí

Tháp xử lí thông tin: Data -> Information -> Knowledge -> Wisdom

Cơ sở dữ liệu: Tập các dữ liệu được tổ chức, lưu trữ và quản lí có hệ thống trong máy, giúp truy cập dễ dàng, giảm trùng lặp, bảo mật cai và hỗ trợ đa người dùng

Thành phần CSDL: Gồm dữ liệu, mô hình quan hệ hoặc phi quan hệ, hệ quản trị CSDL

Hệ quản trị CSDL: Phần mềm quản lí CSDL

RDBMS: DBMS quản lí CSDL quan hệ

Thành phần của SQL Server:

* Database Engine: Thành phần cốt lõi
* Sqlcmd: Công cụ dòng lệnh
* SSMS: Giao diện đồ hoạ

Lập trình với CSDL:

Code First: Định nghĩa mô hình dữ liệu trước trong mã nguồn

Database First: Thiết kế CSDL trước

Code First linh hoạt, phát triển nhanh, nhưng khó tích hợp với CSDL có sẵn, phú hợp dự án chưa có CSDL, phát triển nhanh, tập trung logic ứng dụng

Database First phù hợp với hệ thống sẵn có, tối ưu hoá hiệu suất nhưng ít linh hoạt

[8]Git

Khái niệm: Git là hệ thống quản lí phiên bản phân tán sử dụng rộng rãi trong phát triển phần mềm, cho phép theo dõi và quản lí các thay đổi trong mã nguồn một cách hiệu quả

Tính năng:

* Theo dõi lịch sử thay đổi
* Hỗ trợ đa người dùng
* Phân nhánh và hợp nhất
* Lưu trữ dưới dạng bản ghi, đảm bảo bảo mật
* Hỗ trợ làm việc không cần kết nối mạng và đồng bộ hóa sau đó

Một số lệnh cmd:

* Kiểm tra: git –version hoặc git –v
* Nhúng Git: git init
* Đinh danh người dùng: git config –[system/global] user.name=”…”
* Xem cấu hình: git config –list
* Xoá: git config –[system/global] –unset user.name

Khu vực làm việc:

* Thư mục làm việc: Nơi chỉnh sửa trực tiếp các tập tin trên máy tính, lưu các thay đổi được Git theo dõi hoặc chưa.

Cách khởi tạo:

* Tạo mới: git init
* Sao chép: git clone
* git add: Đưa các thay đổi từ Thư mục làm việc vào Khu tạm
* git status: Hiển thị trạng thái hiện tại của Kho lưu trữ và Thư mục làm việc
* Khu tạm: Nơi chọn lọc và tổ chức các thay đổi đưa vào commit tiếp theo.
* gitignore: Chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua, giúp tránh commit rác, giữ lịch sử sạch, giảm kích thước kho chứa, tránh xung đột
* Kho chứa: Nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của dự án. Thư mục .git chính là kho chứa này.
* git commit: Lưu các thay đổi từ Khu tạm vào Kho chứa