INTRODUCCIÓN

LA WEB

La web son páginas (En formato HTML, Imagen, JSON, XML ,etc) que se interconectan entre ellas por enlaces (urls).

El navegador (cliente) solicita por TCP/IP el recurso a obtener. El formato de como solicita ese recurso es por el protocolo HTTP.

Y el servidor de TCP/IP responde usando el protocolo HTTP devolviendo los datos. A ese software lo llamaremos servidor Web. El servidor es "personalizable" permitiendo que se ejecute código específico para nuestra aplicación. Ese código específico es el código de servidor que se suele escribir en Java, PHP, NodeJS, etc.

El servidor no solo puede comunicarse con servicios en local, sino también puede acceder a servicios externos.

Chrome DevTools

Los navegadores suelen llevar herramientas para depurar las páginas web. En Chrome , si pulsamos F12 y pinchamos en la opción de "Network" (Menú superior), podemos ver todas las peticiones que hace el navegador al cargar una página.

Desplegar

Desplegar es instalar a aplicación web que hemos desarrollado (HTML,CSS,JS,Código Servidor, etc) en un servidor Web.

Desafíos y tareas para desplegar

1. Generación de la aplicación a instalar
1. Obtener el código fuente
2. Compilar el código fuente
3. Probar la aplicación
4. Analizar la calidad del código

2. Múltiples programas a instalar
1. Sistema Operativo
2. Servidor Web
3. Base de datos
4. La aplicación
5. Servidor de envío de correos
6. Etc.

3. Hosts
1. La base de datos puede estar en un Host y el servidor web en otro Host, etc.
2. Los Hosts pueden ser máquinas virtuales o máquinas físicas, estando cada una de ellas en proveedores distintos (Amazon AWS, Microsoft Azure, Google Cloud, etc).

4. Administración
1. Creación de copias de seguridad
2. Logs
3. Seguridad: Recuperación en caso de pérdida de datos por algún fallo

5. Rendimiento: Permitir que la aplicación siga funcionando aunque haya un pico de peticiones
1. Añadir nuevas máquinas si hay un pico de peticiones y quitarlas cuando ya no las hay tantas peticiones
2. Balanceo de carga entre todos los servidores de la aplicación
3. Monitorización del rendimiento
4. Alertas de bajo rendimiento

6. Fiabilidad: Permitir que la aplicación siga funcionando aunque falle algún Host.

7. Microservicios:
1. Una única aplicación se divide en pequeñas micro-aplicaciones, llamadas microservicios, con lo que se multiplican todas las complicaciones anteriores por el número de microservicios que tengamos.

Herramientas relacionadas con el despliegue

Herramientas que vamos a usar en clase:

- Script de Bash
- Git
- Docker
- Git Actions
- NodeJS

Otras herramientas y Servicios:

- Kubernetes
- Vagrant
- AWS
- Ansible
- Jenkins
- etc. Hay muchísimas herramientas distintas y servicios que ayudan al despliegue de aplicaciones.

Conceptos

- VPS (Virtual Private Server): Una máquina virtual que alquilas a una empresa.
- Empresa de Hosting: Empresa que alquila los Host sean tanto un VPSs como una máquina real
- Balanceador de carga: Software que le llega una petición y la redirige a otro Host de entre varios para no sobrecargar ningún Host o evitar enviarlo a un host que no funciona
- Escalabilidad: Diseñar la aplicación de forma que se alquilen o se desalquilen Hosts (VPS o máquinas reales) en función de la carga del sistema
- Tolerancia a fallos: Diseñar la aplicación de forma que aunque un host falle, la aplicación siga funcionando

IAAS - PAAS - SAAS

- IAAS (Infraestructura como servicio): Si la empresa de Hosting solo se ofrece el Host y nosotros nos tenemos que instalar todo el software, incluyendo el sistema operativo y administrarlo todo.
- PAAS (Plataforma como servicio): Si la empresa de Hosting nos ofrece el Host pero también software genérico ya instalado como el Sistema Operativo, Servidor Web, Servidor de Correo, Balanceador de Carga , etc. En este caso nosotros solo debemos instalar el código específico de nuestra aplicación. En este caso aunque nos ofrecen un host ya que en algún sitio debe estar la app, realmente nos están ofreciendo el servidor web donde instalar nuestra app. En el caso de PAAS, no tenemos que administrar nosotros ni el Sistema Operativo ni el servidor.
- ***SAAS (Software como Servicio):*** Como desarrolladores nunca usamos un SASS ya que la empresa de hosting ya ofrece hasta la aplicación instalada. Un ejemplo sería Google con "Google Docs" , Microsoft con su "MS Office 365", Dropbox , etc. que ya ofrecen todo para el usuario final.

IAAS vs PAAS

El IAAS es mas versátil ya que solo nos ofrecen el ordenador y nosotros nos montamos todo como queremos. El problema es que es mas complicado todo de hacer y tenemos que administrarlo todo: Sistema operativo, servidor web y aplicación.

Por otro lado en el PAAS, solo nos tenemos que preocupar de nuestra aplicación lo que hace que sea mas sencillo. El problema es que ya no hay tanta versatilidad, ya que debemos ceñirnos al entorno que nos ofrece la empresa.

NodeJS

NodeJS (o simplemente node) es un lenguaje de programación basado en JavaScript. Al ser un lenguaje interpretado, su forma de trabajar es mas similar a BASH que a Java. Además, incluye un gestor de paquetes: `npm` *(Node package manager)*

Proyectos con node

- Inicializar un proyecto en Node. Se crea el fichero package.json que es fundamental en node ya que contiene toda la información del proyecto

```shell
npm init
```

- Instalar un paquete nuestro proyecto. Se guarda en la carpeta node_modules:
  - Instalar la librería de JavaScript llamada "jQuery"

```shell
npm install jquery
```

  - Instalar un paquete de forma global. Se guarda en "/usr/bin"

```shell
npm install typescript -g
```

  - Instalar todo de nuevo si no está la carpeta node_modules

```shell
npm install
```

Programar en NodeJS

Ahora vamos a ver como ejecutar código node.

El programa mas sencillo es hacer el "Hola Mundo". Para ello creamos un fichero llamado "index.js" con el contenido siguiente:

```
#!/usr/bin/env node

console.log(``'Hola Mundo'``);
```

Para ejecutarlo hay que lanzar la orden:

```
node index.js
```
SHELL

Y mostrará por consola el mensaje "Hola mundo"

Como en NodeJS se usa JavaScript, podemos usar todo lo que sabemos de JavaScript en un programa de node.

Unicode y UTF-8

Unicode

Unicode es un estándar de codificación que tiene como objetivo permitir la representación de todos los caracteres utilizados en los sistemas de escritura del mundo, además de símbolos y emojis. A diferencia del ASCII (7 bits) y ANSI (8 bits), Unicode asigna a cada carácter un número único, denominado "punto de código" y los puntos de código son desde U+0000 hasta U+10FFFF (2.097.152 de caracteres).

Además de unicode están:

- ASCII: ASCII es un código de 7 bits que permite representar 128 caracteres diferentes, incluyendo las letras mayúsculas y minúsculas del alfabeto inglés, los números, algunos símbolos de puntuación y caracteres de control (como el salto de línea o el tabulador). Fue desarrollado en los años 60 y está diseñado principalmente para manejar textos en inglés, ya que no incluye acentos ni caracteres especiales de otros idiomas.
- ANSI: Amplían el estándar ASCII a 8 bits. Sin embargo hay distintas codificaciones ANSI en los nuevos 128 caracteres para distintos idiomas como ruso o griego. Es decir es como que hay distintos estándares ANSI llamados ISO-8859-x. Por ejemplo hay un estándar ANSI para el griego (ISO 8859-7), otro para el cirílico (ISO 8859-7) , etc. Y son distintos en los 128 caracteres últimos. Cada uno de estos estándares se llama Pagina de código.
- Windows: Windows tiene sus propio estándar que es muy similar al ANSI

UTF-8

UTF-8 (Unicode Transformation Format - 8 bits) es un esquema de codificación variable que utiliza entre 1 y 4 bytes para representar cada carácter Unicode. Es altamente eficiente para textos en idiomas que utilizan caracteres latinos comunes, ya que estos se representan en un solo byte. UTF-8 se ha convertido en la codificación más utilizada en la web debido a su compatibilidad con sistemas más antiguos que utilizan ASCII y su capacidad para codificar cualquier carácter Unicode. Además, es muy eficiente en términos de almacenamiento para la mayoría de los lenguajes occidentales.