DESPLIEGUE DE APLICACIONES WEB

Apuntes 1ª Evaluación

Descripción breve

Apuntes de la asignatura Despliegue de Aplicaciones Web del 2º curso del grado superior DAW

Raquel Rodríguez
raquelsweeta@gmail.com

Contenido

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 (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 Ofice 365", DropBox , etc. que ya ofrecen todo para el usuario final.

IAAS vs PAAS

El IAAS es más versátil ya que solo nos ofrecen el ordenador y nosotros nos montamos todo como queremos. El problema es que es más 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 más sencillo. El problema es que ya no hay tanta versatilidad, ya que debemos ceñirnos al entorno que nos ofrece la empresa

TIPOS DE SERVIDORES WEB

Hay dos grandes tipos de servidores web:

- **Servidores Web externos:** Son programas completos que hacen de servidor Web. Una vez instalados/ejecutados se añade el código específico de la aplicación en la carpeta del servidor que indique la documentación
- **Servidores web integrados (o librerías de Servidores web):** Se hace una aplicación (por ejemplo, en Java) y se añade como una librería (un JAR), el código del Servidor Web.

EL PROTOCOLO HTTP

El protocolo HTTP se usa para enviar y recibir datos en la Web. Fue diseñado para transmitir HTML (HyperText Markup Language) pero hoy en día se utiliza para transmitir todo tipo de documentos (imágenes, audio, video, PDF, etc.) y para crear aplicaciones Web.

Características

- **Sencillo**: Es en modo texto y fácil de usar directamente por una persona.
- **Extensible**: Se pueden enviar más metadatos que los que están por defecto. Ej. N.º de página
- **Sin estado**: Cada petición es independiente. Eso es un problema en sitios como por ejemplo un carrito de la compra.

Ventajas

- **Cache**: Mejora la velocidad al controlar la cache de las páginas.
- **Autenticación**: Permite identificar a un usuario.
- **Proxys**: Permite de forma transparente usar proxies.
- **Sesiones**: Gracias a las cookies podemos mantener el estado entre peticiones.
- **Formatos**: Permite indicar el formato de lo que se envía, de lo que se pide y de lo que se retorna.

Formato

Un mensaje HTTP (no importa si es de petición o respuesta) se compone de 3 partes:

- La primera línea (que es diferente para la petición y la respuesta).
- Los encabezados.
- El cuerpo (opcional)

El verbo HTTP

La primera línea de un mensaje de petición empieza con un **verbo** (también se le conoce como **método**). Los **verbos** definen la acción que se quiere realizar sobre el recurso. Los verbos más comunes son:

- **GET**: se utiliza para solicitar un recurso.
- **POST**: se utiliza para publicar un recurso.
- **PUT**: se utiliza para reemplazar un recurso.
- **DELETE**: se utiliza para eliminar un recurso.

El código de respuesta

La primera línea de un mensaje de respuesta tiene un código de 3 dígitos que le indica al cliente cómo interpretar la respuesta.

Los códigos de respuesta se dividen en cinco categorías dependiendo del dígito con el que inician:

- 1XX: Información
- 2XX: Éxito
- 3XX: Redirección
- 4XX: Error en el cliente
- 5XX: Error en el servidor

Los encabezados

Los encabezados brindan información adicional sobre la petición o la respuesta. Los encabezados tienen la siguiente sintaxis:

```
[nombre del encabezado]: [valor del encabezado]
```

Encabezados comunes incluyen:

- `Content-Type`: el tipo de contenido que se está enviando en el cuerpo de un mensaje de petición, por ejemplo, `text/html`.
- `Accept`: el tipo de contenido que el cliente está esperando.
- `User-Agent`: el tipo de navegador que está haciendo la petición.

SERVIDORES INTEGRADOS

Puerto **Apache**: 8081

Puerto **Nginx**: 8083

Configurar el proxy

npm config set proxy http://172.16.0.9:8080
npm config set https-proxy http://172.16.0.9:8080

Crear un proyecto

- Inicializar un proyecto en Node. Se crea el fichero package.json
  - **npm init**
- Instalar la librería de JavaScript llamada "jQuery"
  - **npm install jquery**
- Instalar un paquete de forma global. Se guarda en "/usr/bin"
  - **npm install typescript -g**
- Instalar todo de nuevo si no está la carpeta node_modules
  - **npm install**

Programar en NodeJS

Creamos un fichero llamado "index.js" con el contenido siguiente:

Console.log('Hola Mundo');

Para ejecutarlo hay que lanzar la orden:

node index.js

COPIAR - PEGAR EN DIST - COMPILAR - ELIMINAR

Para comandos en el terminal siempre hacerlo desde raíz

SCRIPTS DE NODE

Los scripts se encuentran en la carpeta de package.json

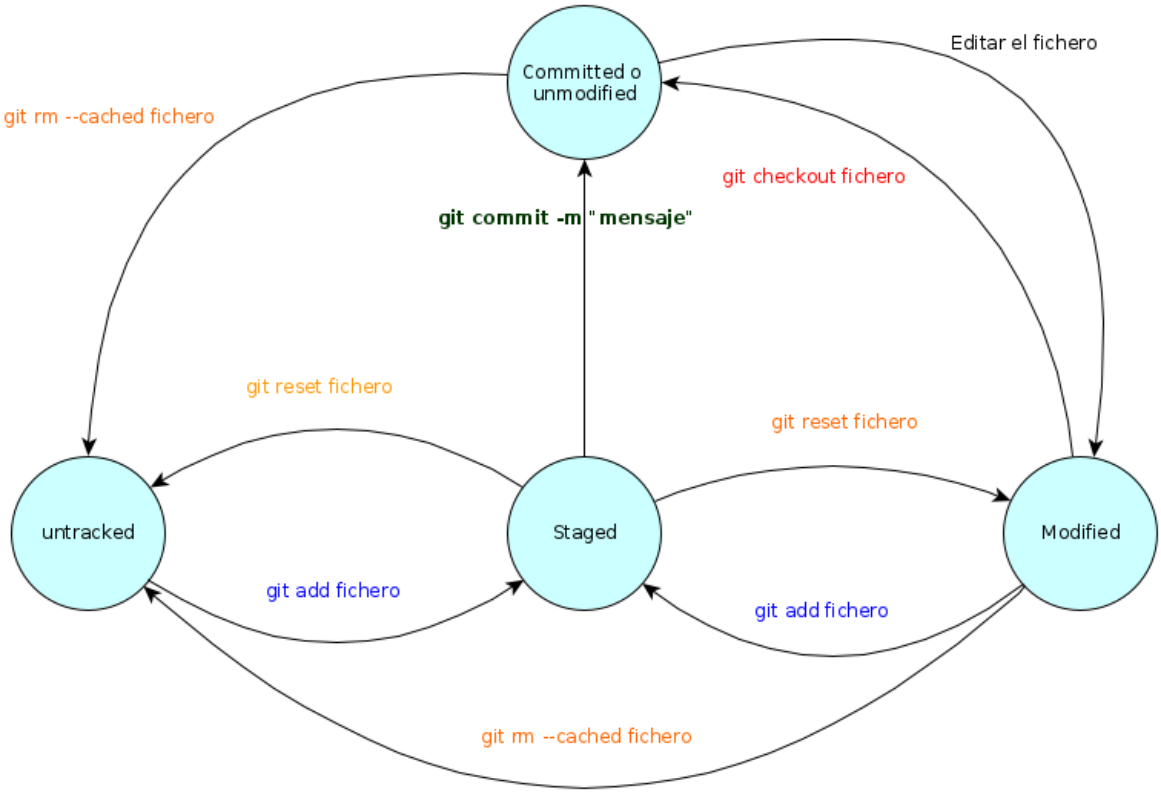
- Ejecutar script para compilar SASS
  - npm run compile-scss
- Ejecutar script para compilar TS
  - npm run compile-ts
- Ejecutar script para compilar los dos archivos a la vez
  - npm run compile
- Ejecutar script para que cuando haya cambios se actualicen automáticamente
  - npm run onchange

GIT

ESTADOS DE UN FICHERO

- **untracked** o archivos sin seguimiento: Son ficheros que existe en el área de trabajo, pero no existen para git

- **staged** o cambios a ser confirmados: Son ficheros modificados que se añadirán al siguiente commit.
- **committed**: Son ficheros que se guardaron en el último commit y que no han sido modificados desde dicho commit.
- **modified** o cambios no rastreados para el commit: Son ficheros que se han modificado desde el último commit pero que aún no se han añadido para el próximo commit.



COMANDOS DE GIT

- Comando para descargarte el código de un repositorio remoto
  - git clone <link-con-nombre-del-repositorio>
- Comando branch para crear, listar y eliminar ramas
  - git branch <nombre-de-la-rama>
- Comando para cambiarse de una rama a otra
  - git switch <nombre-de-la-rama> / git checkout <nombre-de-la-rama>
- Comando para ver la información de nuestro archivo actual
  - git status
- Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit. Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.
  - git add <archivo>
- Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.
  - git commit -m "mensaje de confirmación
- Git push envía tus commits a GitHub.
  - git push <nombre-remoto> <nombre-de-tu-rama>
- Comando para deshacer los cambios que hemos hecho
  - git revert
- Comando para copiar un repositorio
  - git clone /path/to/repository (si está en local)
  - git clone nombredeusuario@host:/path/to/repository (si está en un servidor remoto)
- Comando para descargarse otro repositorio
  - Git fetch

BORRADO DE COMMITS

- Comando para volver a un commit anterior, modificando el área de trabajo y vaciando el staged área.
  - Git reset –hard <código-del-commit>
- Comando para volver a un commit anterior, sin modificar el área de trabajo ni la staged área.
  - Git reset –soft <código-del-commit>
- Comando para volver a un commit anterior, sin modificar el área de trabajo, pero vaciando el staged área.
  - Git reset –mixed <código-del-commit>

RESET VS CHECKOUT

- Git checkout sirve para 3 cosas diferentes:
  - Descartar los cambios de un fichero
  - Ir hasta X commit
  - Moverse a una rama

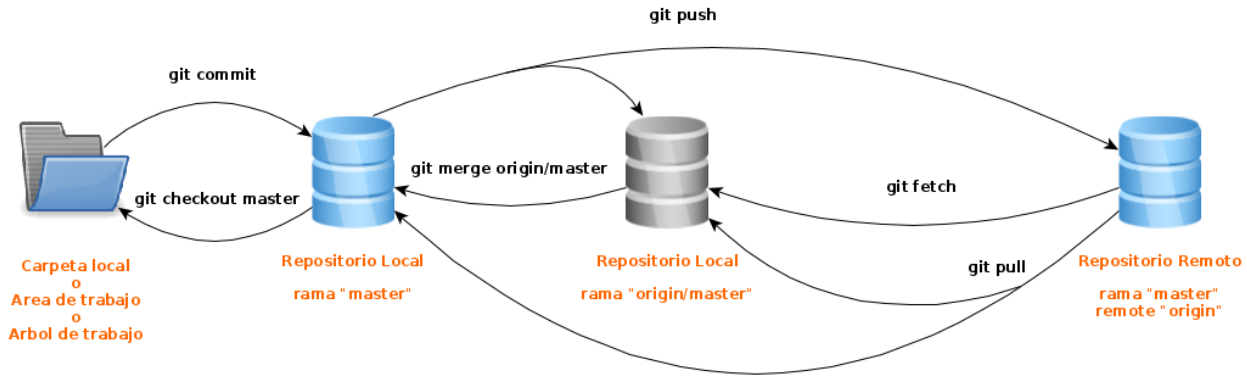
- Git reset sirve para 2 cosas diferentes:
  - Quitar el fichero de la staged área
  - Ir hasta X commit, borrando los siguientes a el

Sincronización de Git

Una misma rama en git puede estar en varios sitios a la vez. El repositorio es el historio donde están todos los *commits* que se han hecho.

Vamos a explicar cada uno de ellos suponiendo que estamos en la rama `master`.

- Carpeta local o área de trabajo o árbol de trabajo: Es donde están los ficheros con lo que trabajamos nosotros directamente mientras programamos.
- Repositorio local: Es el repositorio que tenemos en nuestro ordenador.
  - Rama **master**: Es donde se guardan al hacer commit
  - Rama **origin/master**: Es una copia que hay del repositorio remoto. El nombre de la rama se llama `origin/master`. Hay que fijarse que el nombre va todo junto y no con un espacio en medio. Ya que es el nombre de una rama local solo que es una copia de un remoto llamado `origin` de su rama `master`.
- Repositorio remoto: Es un repositorio que está en otra máquina.
  - Rama **master**: Es donde queremos copiar los *commits* de nuestra rama `master`. Al hacer referencia a una rama de un repositorio remoto, se usará la expresión `origin master` con un espacio en medio. Ya que por separado está el nombre del **remote** llamado *origin* y de la rama llamada `master`



- Quitar el proxy del instituto
    - Git config –global unset hhtp.proxy

Mensajes de commit

Los mensajes de commit deben seguir el siguiente formato:

```
type(#issue):titulos

Explicación (opcional)
```

- type
    - **feat**: Si se añade una nueva funcionalidad (feature)
    - **fix**: Si se arregla (fix) un error
    - **docs**: Si solo cambia cosas de la documentación
    - **style**: Si solo se cambia el estilo del código como tabuladores, puntos y comas, formateo, etc.
    - **refactor**: Si se cambia el código para mejorar su calidad, pero sin modificar la funcionalidad. Eso se llamada refactorizar.
    - **test**: Si se cambian cosas relacionadas con test automáticos.
    - **chore**: Si se cambian cosas relacionadas con el despliegue.
- #issue: Es el Nº de la incidencia a la que hace referencia.

Ejemplo:

- Se arregla el fallo 45 que se llama "Falla si la fecha está vacía"

```
fix(#45):Falla si la fecha está vacía
```