Apuntes curso 4646 Diego

Compiladores:

En C: utilizamos la aplicación desde la línea de comandos: gcc.

En Python: utilizamos el intérprete de Python

Para poder utilizarlos en todo el entorno lo tenemos que añadir al path como *variable* de entorno.

Javascript: No necesita compilador se ejecuta directamente en un navegador que tenga intérprete de javascript (actualmente todos).

El programa utilizado para escribir el código es [**Visual Studio Code**](https://code.visualstudio.com/)

Es interesante mirar la documentación del plugin ya preinstalado en [VSCode](https://code.visualstudio.com/) llamado [EMMET](https://docs.emmet.io/) herramienta muy útil para codificar en distintos lenguajes de programación.

Creamos cuenta de Gmail para el curso. La llamo [diegosg4646@gmail.com](mailto:diegosg4646@gmail.com)

Control de versiones: (Utilizaremos Git (gratuito) *github*)

<https://git-scm.com>

*Otros gratuito muy utilizado es SUBVERSION, BITBUCKET*

Git es un sistema de gestión de control de versiones distribuido, es decir que varias personas participan en el proyecto teniendo acceso al código principal.

Github (es el servidor de control de versiones. La base de datos)

Tiene muchos proyectos accesibles a todo el mundo, desde la nasa, juegos como Doom y Quake y libros de todo tipo de lenguajes de programación: <https://github.com/EbookFoundation/free-programming-books/blob/master/free-programming-books.md>

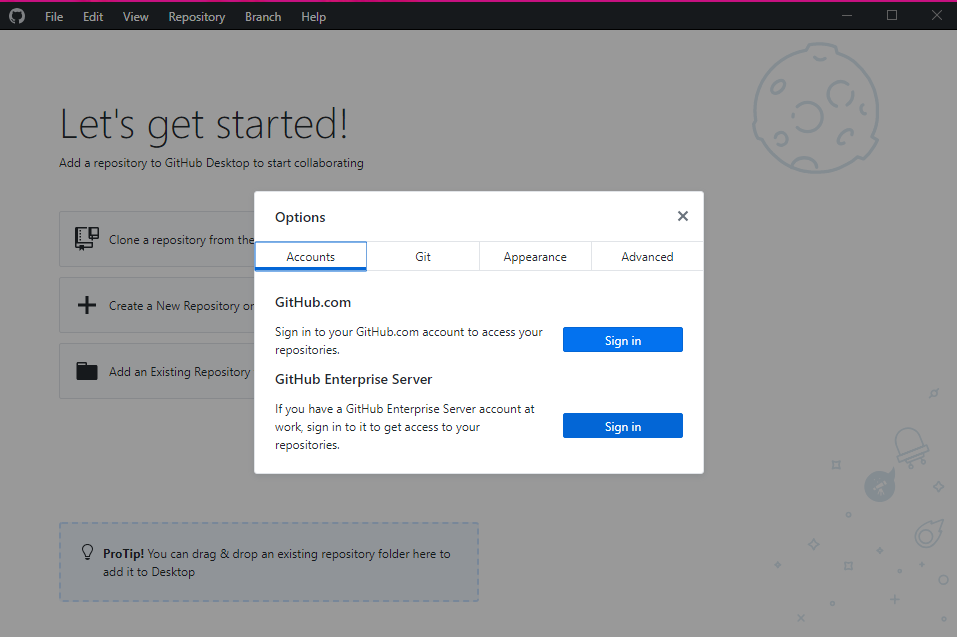
Para hacer uso de Git vamos a utilizar github-desktop.

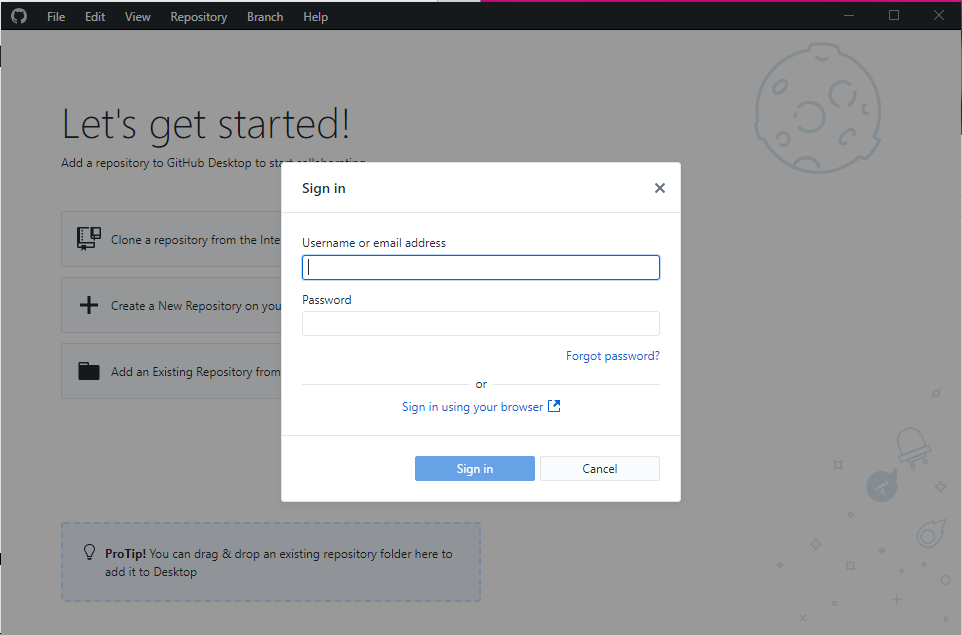
Para poder funcionar necesitamos una cuenta en Github (para alojar la base de datos), Git (usar desde la línea de comandos) instalado en nuestro pc y, opcionalmente github-desktop para facilitar el uso desde nuestro ordenador.

Creo una cuenta en github para el curso utilizando el correo de Gmail. Nombre *diegosg4646*

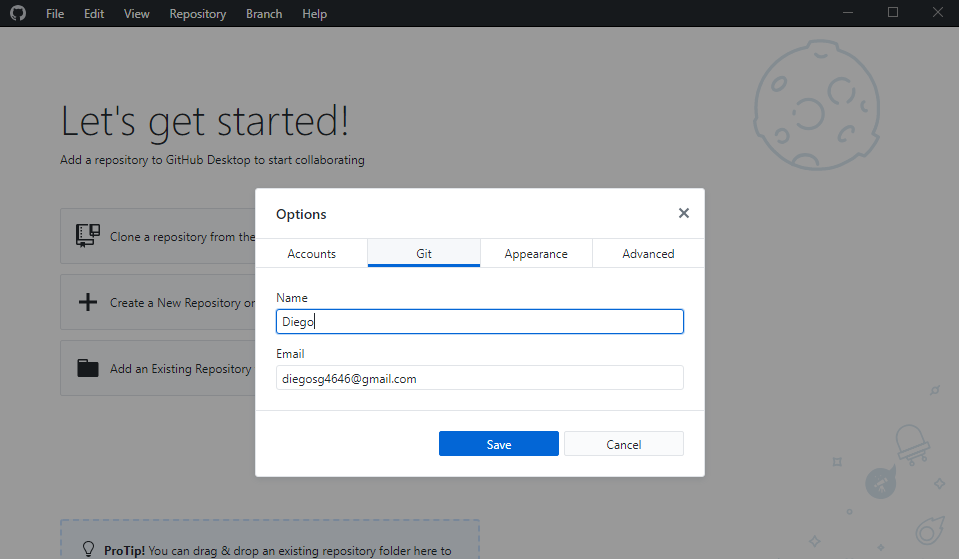
Github del profesor: [Fernando Paniagua](https://github.com/fpaniagua-ifct0609-2020)

Usaremos el github-desktop:



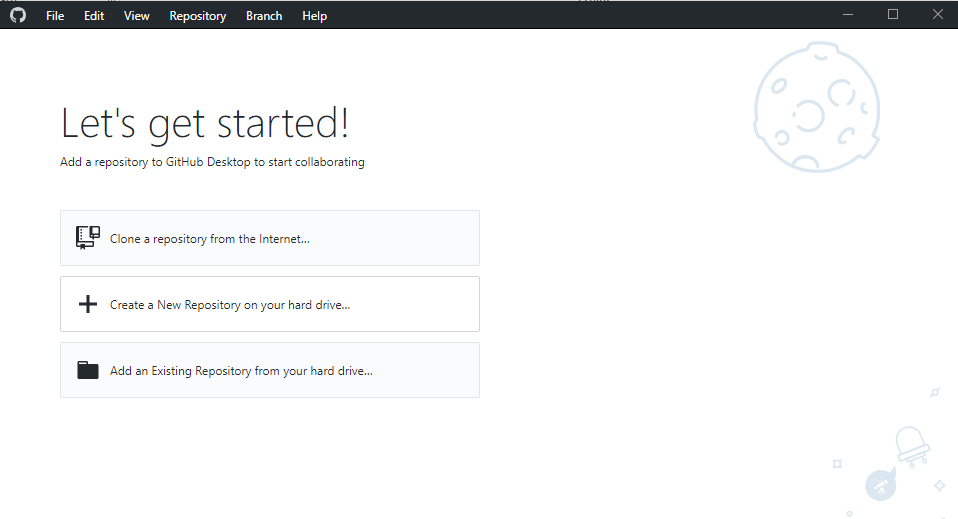


En *File* 🡪 *Options*, en la pestaña pulsaremos en *Singn in* para iniciar sesión con nuestra cuenta de *github*

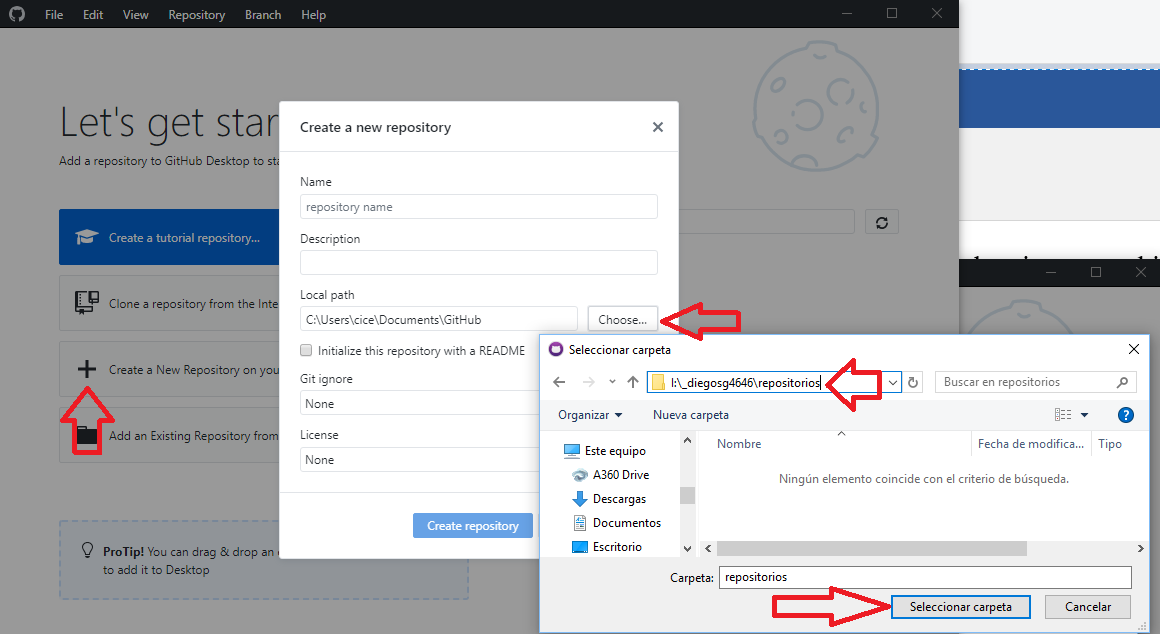


En File 🡪 Options Git. El nombre y correo da igual el que le pongas. Solo es una firma para indicar quien ha modificado el código.

Pantalla inicial.



Elegimos crear nuevo repositorio en el disco duro.



En *local path* es importante indicar que tenemos que seleccionar la carpeta raíz de la que “colgarán” los repositorios. En nuestro caso seleccionamos la ***carpeta*** ***repositorios***.

El nombre que pongamos en *Name* (*repository name*) será, además del nombre de nuestro repositorio, el nombre de la carpeta; que creará automáticamente.

***Git ignore***: Plantillas de un fichero que indica que tipo de ficheros no quiero subir al servidor (por ejemplo, los .exe en el caso del *lenguaje C*). Sólo subimos lo importante.

Si ponemos *NONE* incluye todo en el repositorio.

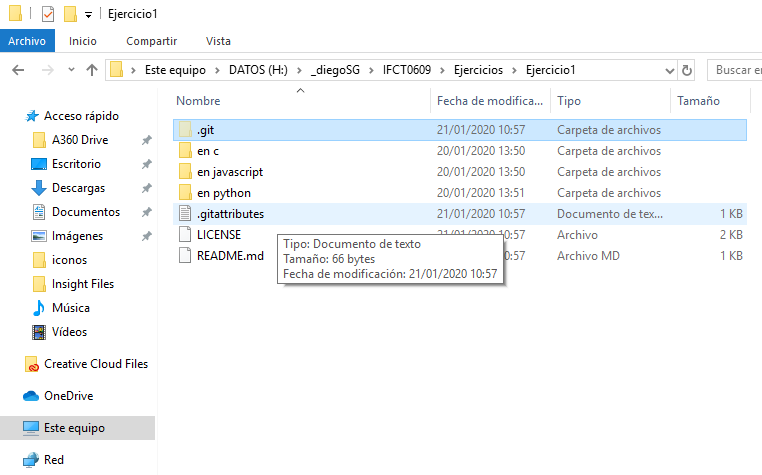
Por norma general el .*gitignore* lo realiza el jefe del proyecto en el que indica que archivos se guardan del proyecto y cual no. (Las plantillas personalizadas pueden incluir archivos **no** necesarios o que ignore archivos que **si** necesitemos versionar).

***License***: Elegir el tipo de licencia del proyecto subido.

Elegimos MIT como ejemplo. No es importante en éste momento.

Y pulsamos en crear: al hacer esto se crea una carpeta en local llamada como el nombre puesto en el campo ***Name*** y dentro genera automáticamente una ***carpeta llamada .git*** y un ***archivo .gitattributes.***

* ***Es importante indicar son archivos ocultos y que tenemos que tener habilitado en el sistema operativo el poder ver éste tipo de archivos.***





Una vez creado para que se *suba al servidor* (en nuestro caso *github*) tenemos que pulsar en ***Commit to master***. Esto crea el repositorio en nuestra cuenta github.

**DEFINICIONES INSTRUCCIONES Git (GitHub Desktop):**

**Repositorio**: copia de un proyecto

**Commit**: veces subidas o guardadas del proyecto.

**Hacer commit** (*commit to master*: es **confirmar los cambios** en la base de datos. (**En local**)

**Push**: subir los datos al servidor. Se hace después de hacer commit en local.

**Clone**: Cuando un proyecto ya está iniciado (creado en remoto) lo clono del servidor a mi equipo para trabajar con él.

**Fetch**: sincronizar los datos con el servidor 🡪 y nos indica si hay algún cambio que tengamos que descargar con Pull.

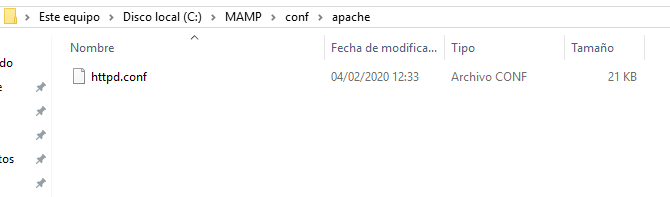
**Pull**: descargar los archivos actualizados del servidor.

* **Normalmente se hace *commit* y después *push* para sincronizar los datos tanto en local como en el servidor.**

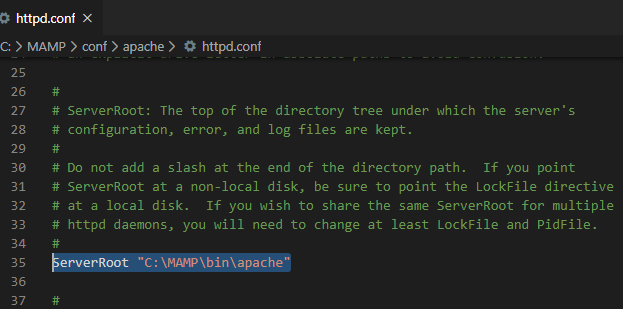
**CREAR UN WEB-SERVER EN LOCAL USANDO MAMP**

Cosas importantes a tener en cuenta:

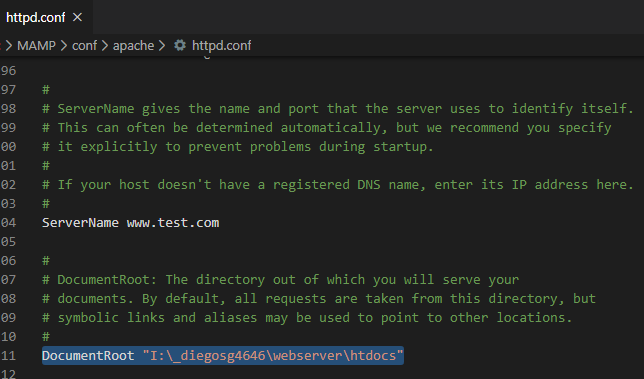
Dentro de la carpeta *mamp/conf/apache*



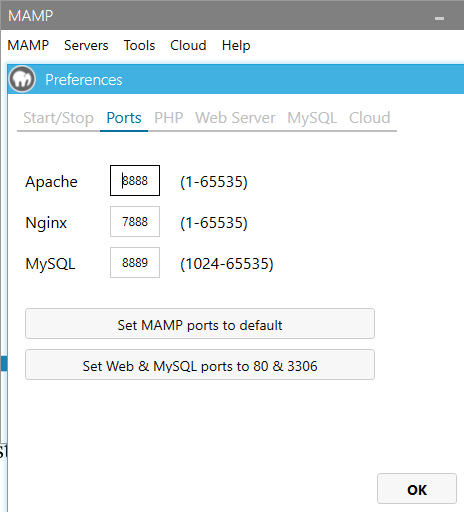
Tenemos el archivo de configuración de apache: *httpd.conf*; indicando en ServerRoot la ruta de acceso a la aplicación (donde se encuentra el ejecutable de apache: ***httpd.exe***).



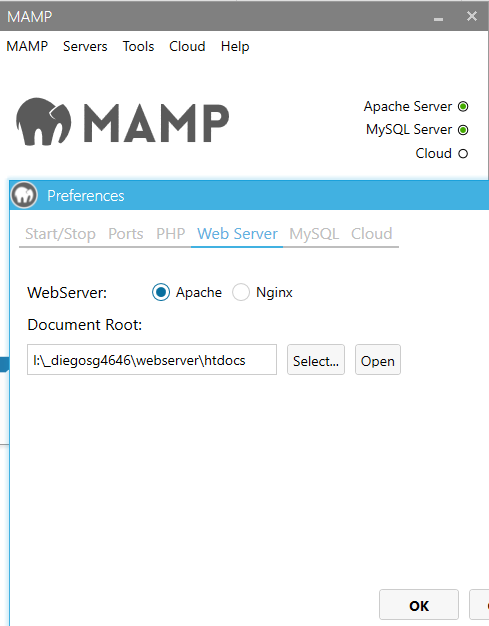
Y en ***DocumentRoot*** indicamos la ruta al directorio en el que añadiremos los “servicios (web…)” que publicaremos para que puedan ser accesibles desde fuera.



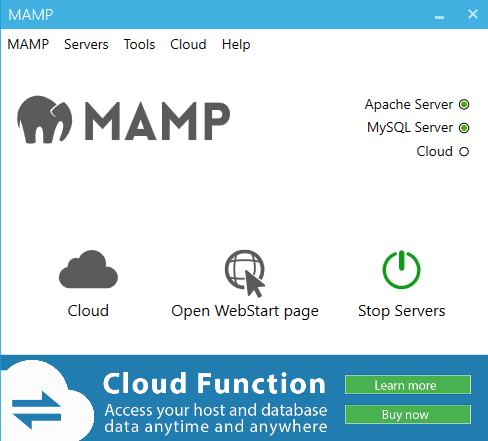
Esto se puede modificar también desde la propia aplicación MAMP 🡪 Preferences 🡪 Ports: aquí se configura el puerto de escucha de Apache (“Servidor que utilizaremos”). Por defecto suele estar el puerto 80.



Y en la pestaña MAMP 🡪 Preferences 🡪 Web Server: indicamos el Web Server que utilizamos y en Document Root 🡪 la ruta raíz donde pondremos la aplicación o servicio.



Y pulsamos en el botón de Iniciar.



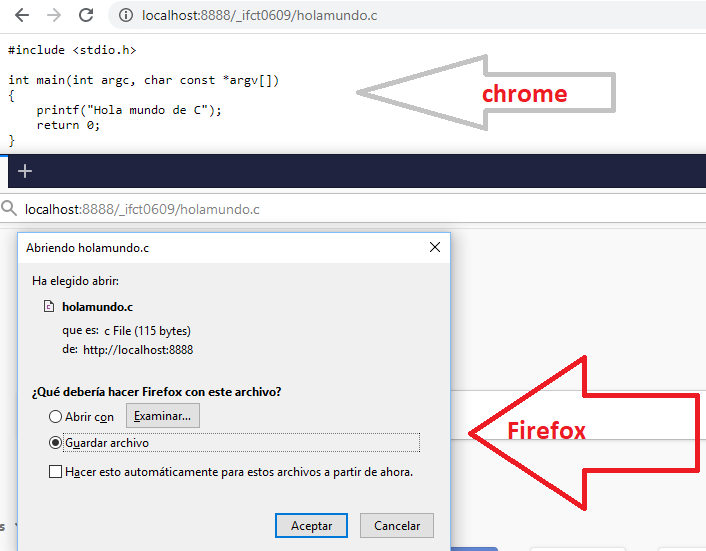
Ahora para acceder al servidor (al contenido que hemos alojado en la carpeta Document Root) accederemos desde la barra del navegador escribiendo lo siguiente:

* Localhost:puerto
* En nuestro caso sería localhost:8888 (que es la configuración que hemos puesto en el archivo *httpd.conf.*
* En lugar de localhost se puede poner la dirección local de localhost que modifica el navegador directamente: 127.0.0.1:*puerto*

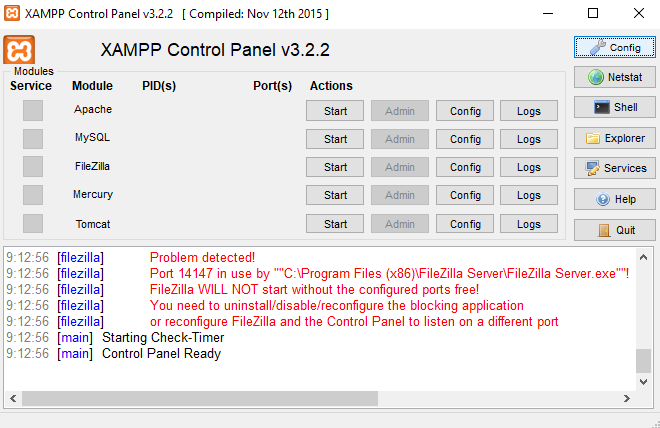
Si por ejemplo copiamos nuestro proyecto web directamente en la raíz de nuestra carpeta Document Root y ponemos en el navegador localhost:8888 cargaría nuestra página index.html por defecto sin poner nada mas.

Si queremos acceder al resto de carpetas se puede poner /carpeta para poder navegar entre las distintas carpetas de nuestro proyecto.

Cuando pedimos un recurso al Web server a través del navegador si lo tiene éste nos lo devuelve… luego dependiendo del tipo de archivo que sea el navegador lo muestra o da opciones de descargar o mostrar… pero el Web server si tiene el recurso solicitado lo entrega.

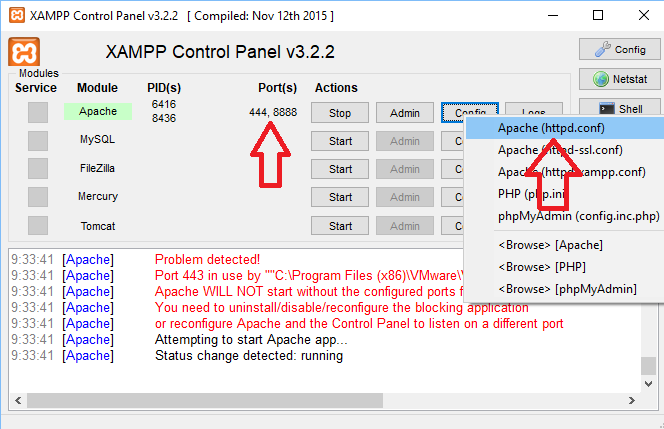


Otro sistema parecido y más usado es XAMPP.



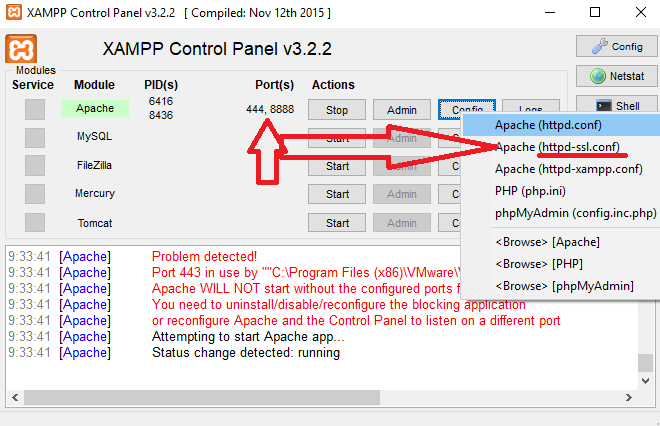
En el archivo de configuración ***httpd.conf*** indicamos el puerto 8888 como prueba, por defecto es el puerto 80 🡪 comunicación http.

Aquí también podemos modificar la ruta del directorio de trabajo y la ruta donde está el ejecutable de apache. También la ruta inicial en caso de que no indiquemos recurso… por defecto index.php. Buscando en el archivo de configuración ***index.php*** podemos encontrar donde modificar ésta preferencia.



En el archivo de configuración httpd-ssl.conf aquí modificamos el puerto de escucha para las conexiones seguras (cifradas ssl) https.

En nuestro caso utilizamos el puerto 444. Por defecto es 443



Prueba de envío de formulario por GET

 <!-- http GET PETICION DE DATOS A TRAVES DEL CONTENEDOR DESDE LA PROPIA URL-->

    <form action="" method="get">

      <input type="email" placeholder="E-mail" name="email" id="email"><br>

      <input type="password" placeholder="Contraseña" name="pwd" id="pwd"><br>

      <input type="submit" value="Validar"><br>

    </form>

Prueba de envío de formulario por POST

<h1>SUPER CURSO INGLÉS</h1>

    <form action="./php/traductor.php" method="POST">

        <!--<input type="text" name="palabraCastellano" placeholder="Palabra español">-->

        <select name="palabraCastellano">

            <option value="Perro">Perro</option>

            <option value="Gato">Gato</option>

            <option value="Patata">Patata</option>

        </select>

        <input type="text" name="palabraIngles" placeholder="Palabra inglés">

        <input type="submit" value="Verificar">

    </form>

FTP:

Protocolo de transferencia de archivos. Casi todos los sistemas tienen el cliente instalado. Se accede escribiendo ftp en la consola.

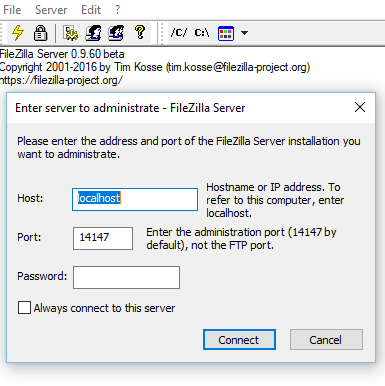
Hay una aplicación para poder usarlo de manera visual llamada ***Filezilla***. (Hay **servidor** y **cliente**)

192.168.3.13 IP PC del profesor

**Filezilla Server interface. (Servidor)**

Lo utilizamos para crear un servidor ftp en local para pruebas.

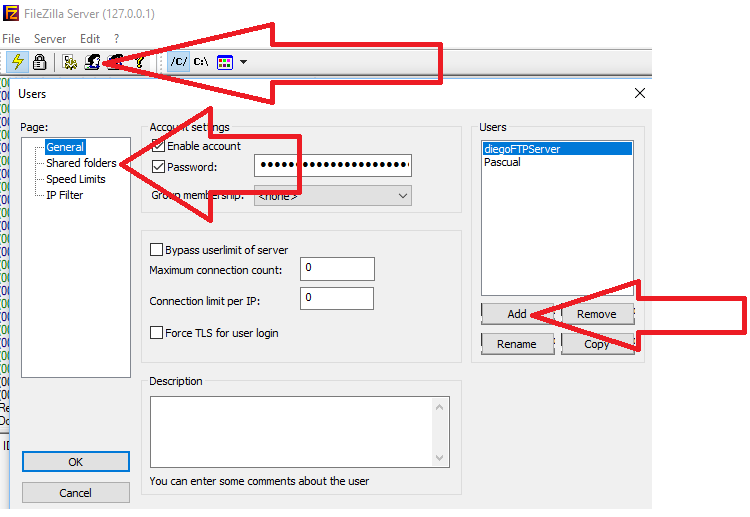
Al ejecutar la aplicación sale la siguiente pantalla. Por defecto no ponemos nada y damos en aceptar.



Pulsamos en el botón  *Users* para poder añadir usuarios para poder acceder a nuestro servidor ftp.

Después de añadir el usuario le deberíamos poner una contraseña de acceso y en Shared folders añadiríamos la ruta o ruta a los directorios de trabajo del cliente para que pueda utilizarlo. Lo normal es crear un directorio de trabajo para cada persona que vaya a trabajar en el proyecto.

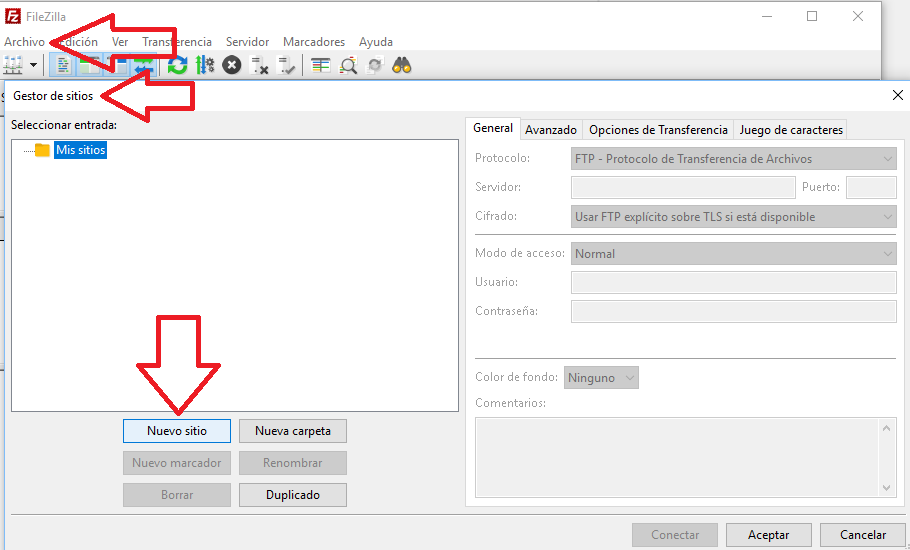
Luego indicando al cliente la ruta al servidor ftp, sus credenciales (usuario y contraseña) podrá acceder a su carpeta de trabajo asignada en el servidor ftp; en la cual dependiendo de los permisos sobre la/las carpeta/s asignadas al usuario podrá crear, modificar, añadir, etc…



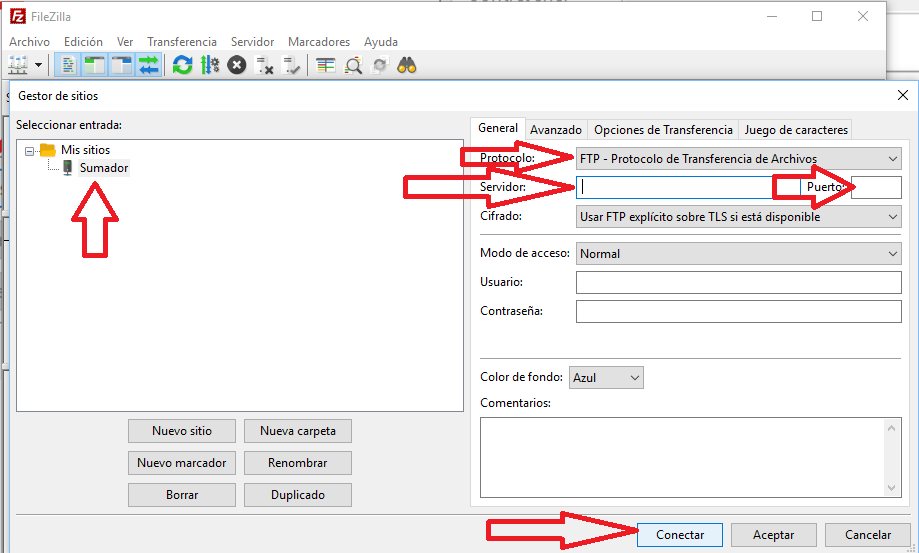
**Para el cliente**:

Usamos la aplicación de ***Filezilla Client***

Archivo 🡪 Gestor de Sitios 🡪 Nuevo Sitio



Le damos un nombre al servidor al que vamos a acceder y en Servidor ponemos la dirección IP del servidor y el puerto al que nos diga el servidor



TCP

IP: Dirección del elemento de RED

DNS:

TCP/IP

UDP

HTTP

FTP

HTTPS

POP3

IMAP

SMTP

HTML 5

Estándar gratuito creado para intercambiar documentos entre humanos.

Etiquetas principales.

Etiquetas contenedoras.

DIV 🡪 Se utiliza para hacer secciones. Es un contenedor que se comporta con un salto de línea al final por defecto.

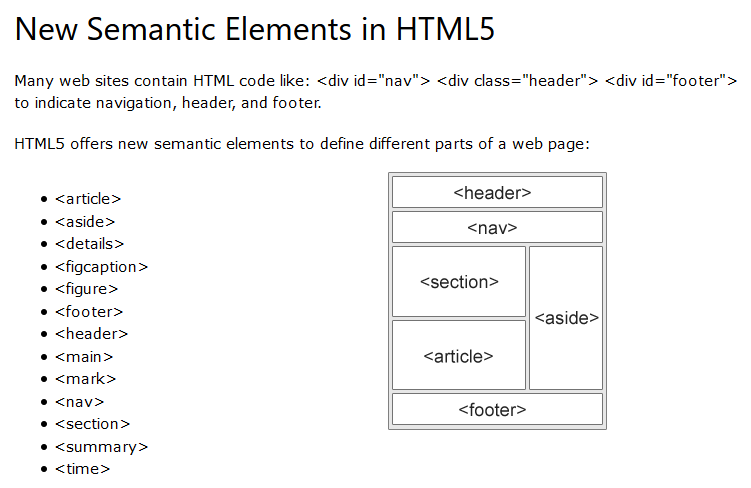
SPAN 🡪 Contenedor sin salto de línea

BR 🡪 Salto de línea. Esta etiqueta no tiene cierre.

P 🡪 Párrafo

H1…H6 🡪 Encabezados. Elementos estructurales.

**Añadidas en HTML5**: (***Etiquetado semántico*** para organizar las páginas web. Estéticamente funcionan igual que la etiqueta <div>)



<article> 🡪 Estéticamente es igual que un <div> 🡪 Bloque de información coherente

<header> 🡪 Bloque de información coherente. Los buscadores le dan más importancia que si utilizaramos <div> aunque el comportamiento sea igual.

<nav> 🡪 Suele incluir el Menú de navegación en la parte superior

<footer> 🡪 Pie de página.

<aside> 🡪 Barra de información lateral

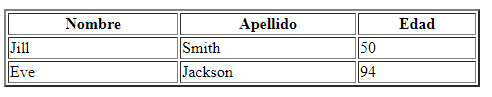
<section> 🡪 Delimita una sección. Se utiliza para

<TABLE> Tablas: No se debe usar para maquetar páginas web

La tabla también es un elemento contendor que puede contener otros elementos en su interior… no solo los elementos propios de la tabla.

<https://freshdesignweb.com/free-css-tables/>

<table style="width:100%" border=”2”>  
  <tr>  
    <th>Nombre</th>  
    <th>Apellido</th>   
    <th>Edad</th>  
  </tr>  
  <tr>  
    <td>Jill</td>  
    <td>Smith</td>   
    <td>50</td>  
  </tr>  
  <tr>  
    <td>Eve</td>  
    <td>Jackson</td>   
    <td>94</td>  
  </tr>  
</table>



&nbsp; 🡪 con el ‘;’ incluido es un espacio en blanco

FORMULARIOS (se utiliza para enviar datos al servidor junto con botón tipo submit)

TIPOS INPUT (Pueden o no estar dentro de un formulario).

<input type=”text”> 🡪 Genérico

 <input type="text" name="nombre" id="nombre" placeholder="Introduce tu nombre" size="100" maxlength="10">

El atributo ***name*** es obligatorio para que se envíe el contenido del input al servidor al hacer submit en el formulario

El atributo ***id*** se utiliza para identificarlo en css o javascript

El atributo ***placeholder*** es lo que aparece dentro del input como información si no hay nada escrito

El atributo ***size***: indica el tamaño del componente input

El atributo ***maxlength*** indica la máxima longitud del contenido.

El atributo required (required=”true”) indica que es obligatorio.

<input type="number" name="edad" id="edadId" placeholder="Introduce tu edad" min="10" max="20">

La etiqueta input de tipo ***number***:

El atributo ***min*** indica el mínimo y el atributo ***max*** indica el número máximo.

<input type="submit" value="Verificar">

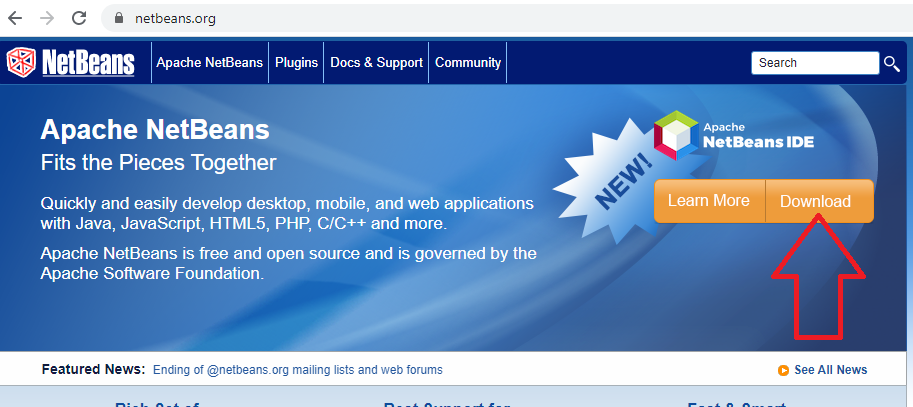
El tipo *submit* se utiliza para enviar los datos del formulario que tienen propiedad *name*.

CREAMOS UNA APLICACIÓN EN JAVA

Usaremos en el primer ejemplo el IDE (Entorno de Desarrollo Integrado) 🡪 NETBEANS

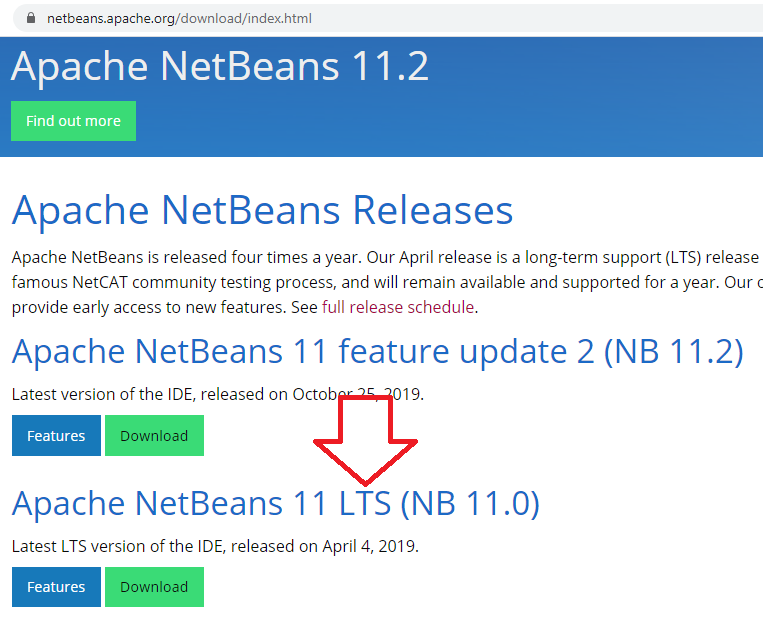
Es el editor oficial de JAVA

<https://netbeans.org/>



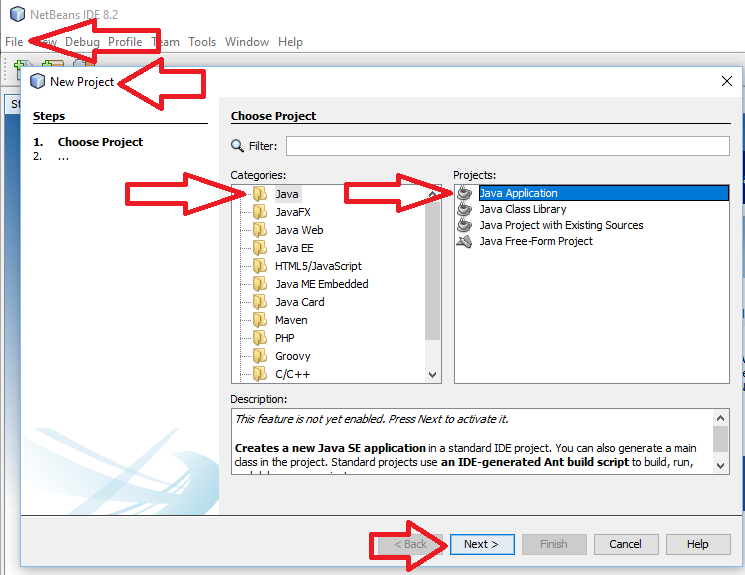
Pulsamos en Download

Y elegimos la versión LTS que es la estable para instalar.

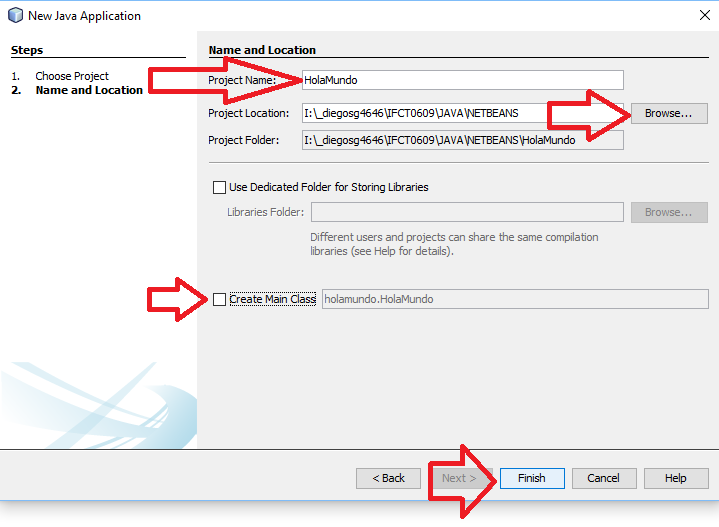


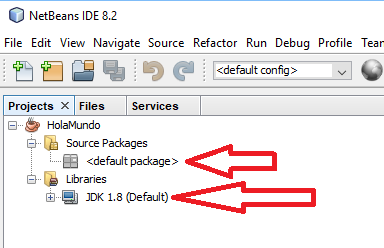
La versión en la que indica LTS (Long Time Support) 🡪 Asegura que tiene soporte durante bastante tiempo. Es la versión recomendada a utilizar.

Empezaremos utilizando la versión de NETBEARNS IDE 8.2



File 🡪 New Project 🡪 Java 🡪 Java Application. Y pulsamos Next



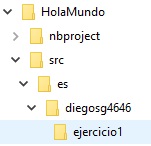


Una buena manera (convención) de crear los nombres de los package es utilizar nuestra dirección de dominio escrita a la inversa y en lugar de www poner unas siglas identificativas de nuestro proyecto.

Por ejemplo:

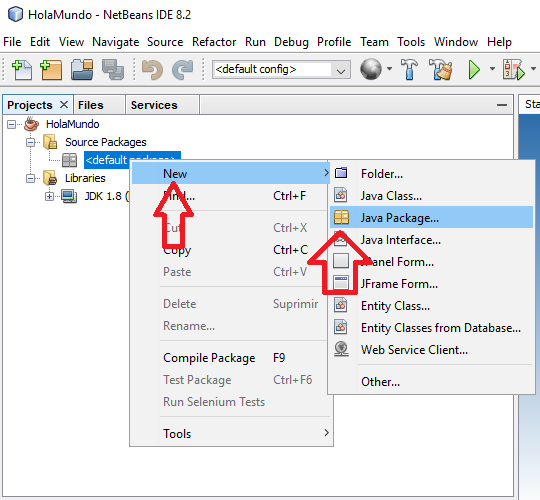
[www.diegosg.com](http://www.diegosg.com) 🡪 com.diegosg.ejercicio1 (***Utilizar minúsculas***)

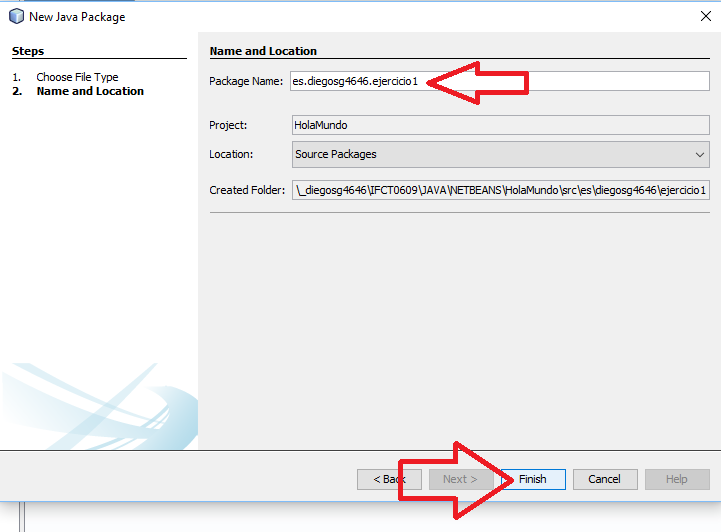
Físicamente cada “.” crea una carpeta (una carpeta com 🡪 diegosg 🡪 ejercicio1 ) en estructura de directorios dentro de la carpeta src (com/diegosg/ejercicio1)



Creamos un package utilizando la nomenclatura anterior:

Click botón derecho 🡪 New 🡪 Java Package…





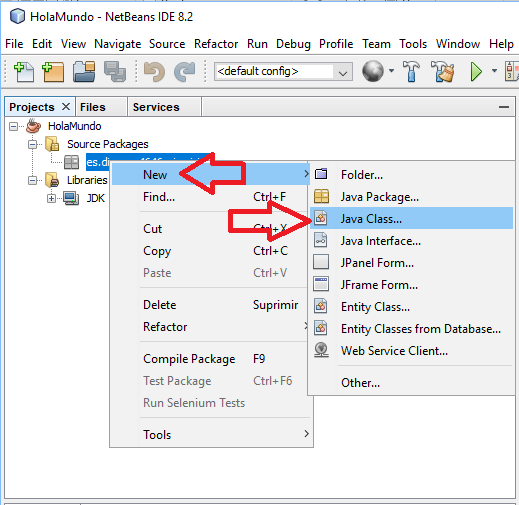
Creamos una clase.

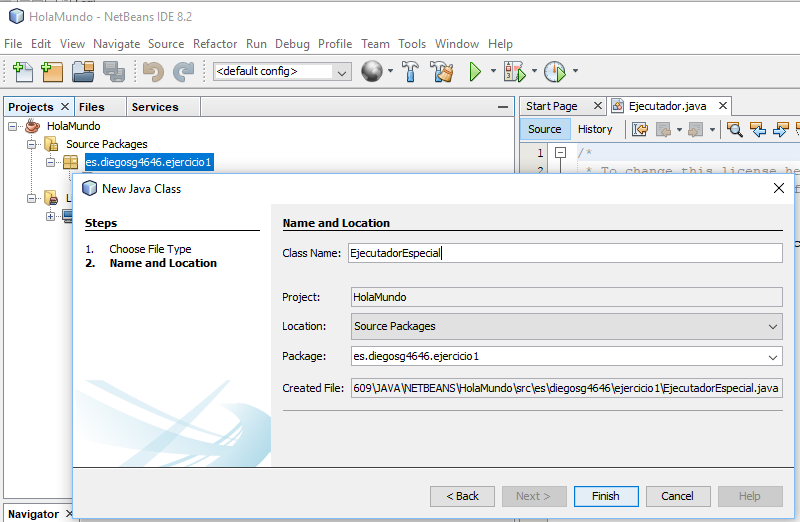
La convención de nomenclatura para las clases en Java es:

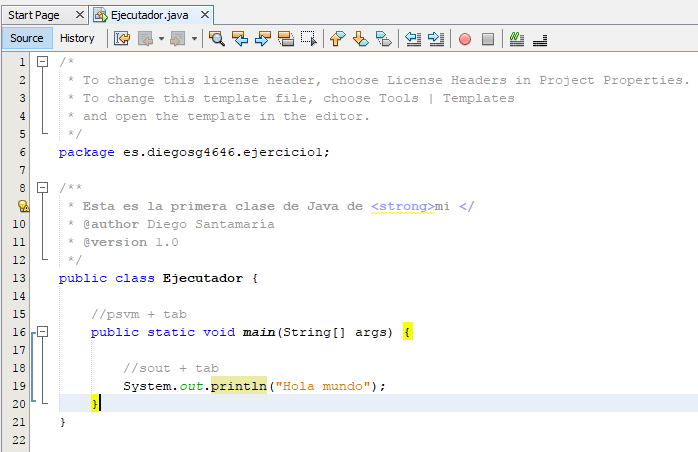
Primera letra mayúscula y resto en Camel Case 🡪 *FacturasClientes*

Creamos la clase pulsando botón derecho sobre el package donde queramos trabajar.

Botón derecho 🡪 New 🡪 Class (Si no hubiéramos creado el package anteriormente al crear la clase nos “obliga” a crear el package







OTROS IDE UTILIZADOS PARA PROGRAMAR EN JAVA:

JETBRAINS

<https://www.jetbrains.com/es-es/idea/>

ECLIPSE

<https://www.eclipse.org/>

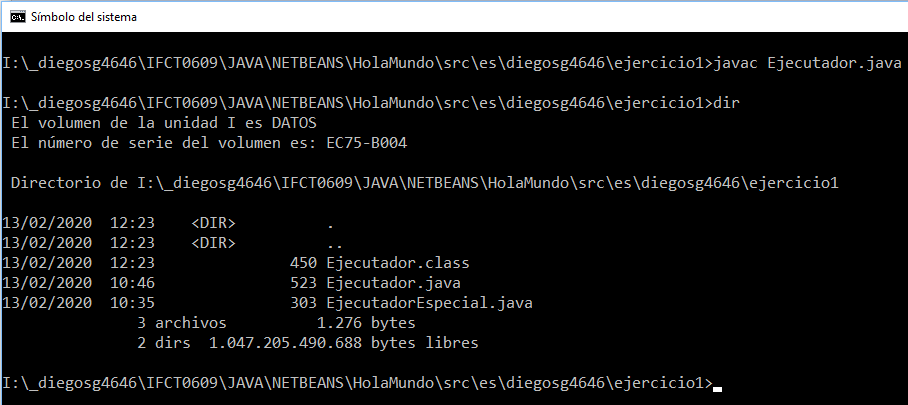
SCRIPT DESDE LA LINEA DE COMANDOS.

El compilador desde la línea de comandos (CMD, POWERSHELL, …) para java es ***javac***

Nos colocamos en la carpeta donde tenemos el archivo .java y escribimos el comando:

Javac Ejecutador.java

Asi nos crea el archivo .class



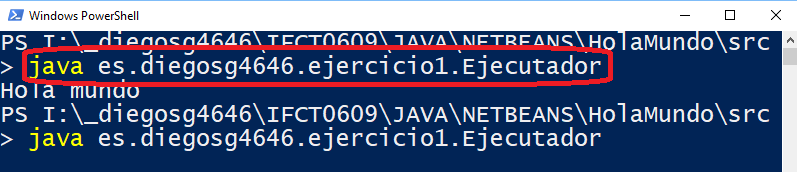
Así se genera el archivo Ejecutador.class

Para ejecutarlo la instrucción es ***java***

Como para ejecutar el archivo .class hay que indicar el nombre de la clase completo desde dentro del package y sin poner la extensión .class al final:

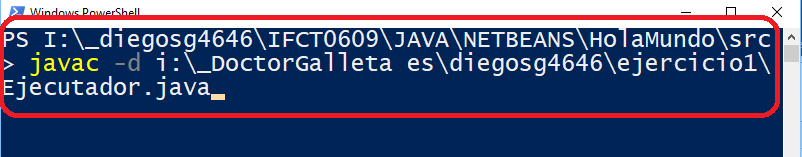
En nuestro ejemplo 🡪 ***es.diegosg4646.ejercicio1.Ejecutador***

Por esto tenemos que colocarnos en la carpeta src en la consola



Para que al compilar usando ***javac <nombre de archivo.java>*** el ejecutable (.class) lo cree en otra ubicación:

javac - d i:\\_DoctorGalleta es\diegosg4646\ejercicio1\Ejecutador.java

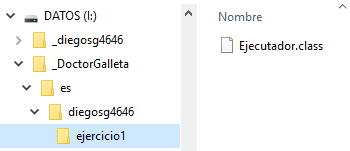


Así nos crea la clase en el directorio indicado i:\\_DoctorGalleta

Al ejecutar el archivo .java que está en la dirección 🡪 es\diegosg4646\ejercicio1\Ejecutador.java

Desde la ruta donde estamos.

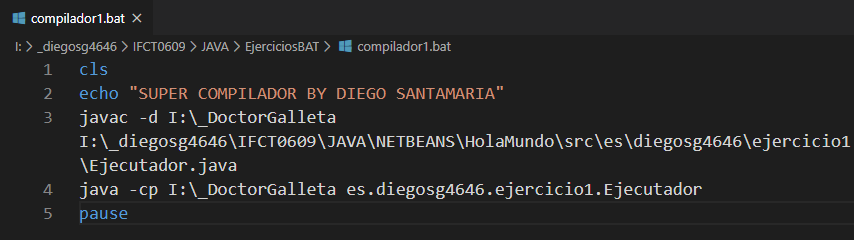
Nos crea la siguiente estructura:



Y con la instrucción: ***java –cp I:\\_DoctorGalleta es.diegosg4646.ejercicio1.Ejecutador***

ejecutamos el archivo .class generado

Para automatizar los dos procesos 🡪 Nos creamos un archivo .bat para 🡪 compilar y ejecutar



Utilizamos rutas absolutas para que no importe donde está el archivo .bat para que funcione.

Éste archivo .bat no sería muy reutilizable puesto que sólo sirve para:

1.- compilar el archivo Ejecutador.java ubicado en nuestro directorio y crear el ejecutable en el directorio indicado (I:\\_DoctorGalleta).

2.- ejecutar el archivo generado en el paso anterior.

SCRIPT EN LINUX.

El lenguaje se llama ***bash*** de Linux (es equivalente de .bat del cmd en windows)

La extensión es .sh

Desde Linux se crea un archivo con la extensión .sh

Al principio del archivo ponemos la siguiente instrucción:

**#!/bin/bash**

Para ejecutar el archivo tenemos que tener en cuenta los permisos del archivo (tiene que tener permiso de ejecución para funcionar)

De la siguiente manera:

**./<nombreDelArchivo.extension>**

\*\* Email profesor: fernando.paniagua@ciceonline.com

**JQUERY (JAVASCRIPT) Y BOOTSTRAP**

jQuery es una librería que utiliza javascript. Sirve para facilitar el uso de javascript.

<https://jquery.com/>

Alternativas a jQuery: <https://umbrellajs.com/>

Bootstrap 🡪 Conjunto de plantillas para diseñar aplicaciones y páginas web

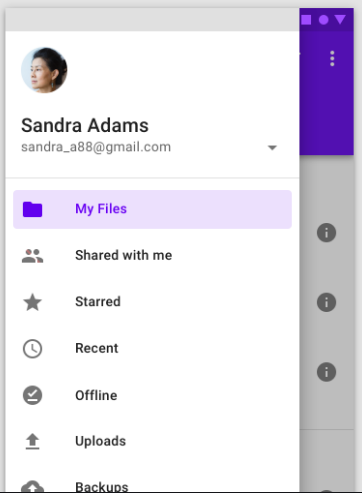
<https://getbootstrap.com/>

Otro ejemplo parecido a Bootstrap: <http://getskeleton.com/>

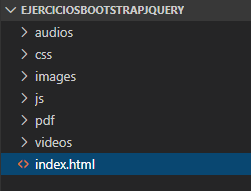
Bootstrap:

Ejercicio.

Vamos a crear una aplicación con menú del estilo siguiente utilizando Bootstrap y jQuery



Creamos un proyecto web

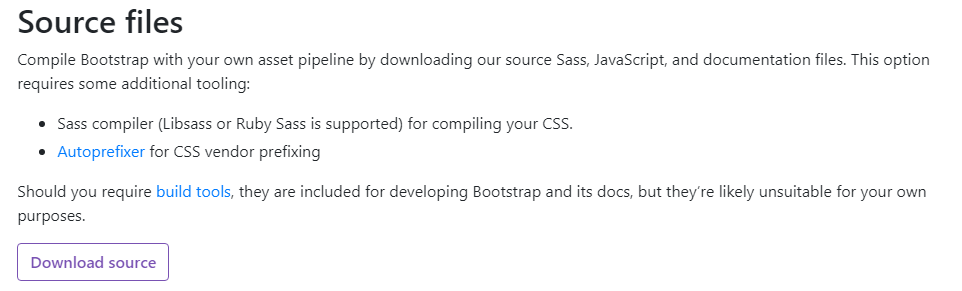


Descargamos Bootstrap

Versión compilada (compactada) -dist:

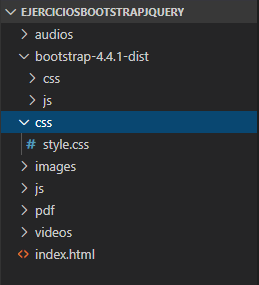


Versión más legible:

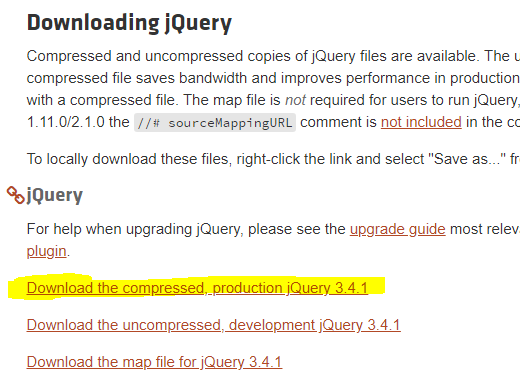


Usaremos la versión compactada –dist

Y la agregamos a nuestro proyecto:



Descargamos la versión comprimida de jQuery



Y pulsamos click derecho guardar como para guardarlo en el directorio del proyecto.

También se puede elegir enlazar con la librería online en lugar de descargarlo en nuestro proyecto.

Por ejemplo en lugar de escribir ***var a = document.getelementbyid(“menu”);***

***En jQuery:***

***var a = $(“#menu”);***