



# Django 2.0

Archivos estáticos

# Archivos estáticos

Los archivos estáticos son aquellos que nos permiten hacer uso de imágenes, hojas de estilo CSS(Cascade Style Sheet), códigos de Javascript, vídeos, documentos entre otros archivos que nos ayudarán a tener un sitio web más colorido y con mejor contenido.



# CSS(Cascade Style Sheet)

Las hojas de estilo en cascada nos ayudan a darle una mejor vista a nuestras páginas web. Interactúan con las etiquetas de HTML en forma de clases(porciones de código que tienen atributos de CSS) o directamente afectando a una etiqueta cambiando su apariencia(forma, color o posición).

# Frameworks de CSS



Aunque podemos aprender a crear estilos desde cero, no es el objetivo del curso, sin embargo existen herramientas como los frameworks de CSS que nos van a permitir colocar elementos ya creados de una forma más fácil ayudándonos así a la creación rápida de nuestras aplicaciones con Django. Los dos frameworks más famosos son:

- Materialize(Contiene el diseño de Material Design de Google)
- Bootstrap(Creado por Twitter)

# Materialize

Durante el curso utilizaremos este framework por ser más sencillo que Bootstrap y tener un mejor diseño y claridad del código que implementa, el código lo podremos descargar en:

- <https://materializecss.com/getting-started.html>

# Descomprimiendo materialize

## Download

Materialize comes in two different forms. You can select which version you want depending on your preference and expertise. To start using Materialize, all you have to do is download one of the options below.

### Materialize

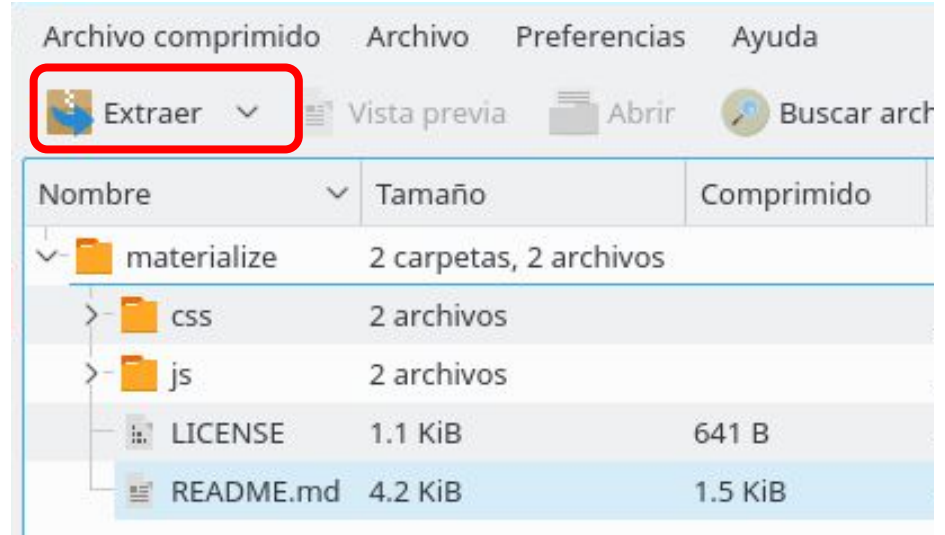
This is the standard version that comes with both the minified and unminified CSS and JavaScript files. This option requires little to no setup. Use this if you are unfamiliar with Sass.

MATERIALIZE 

### Sass

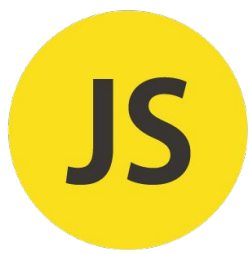
This version contains the source SCSS files. By choosing this version you have more control over which components to include. You will need a Sass compiler if you choose this option.

SOURCE 



**Descargar y extraer en alguna carpeta, puede ser el Escritorio o Documentos.**

# Javascript



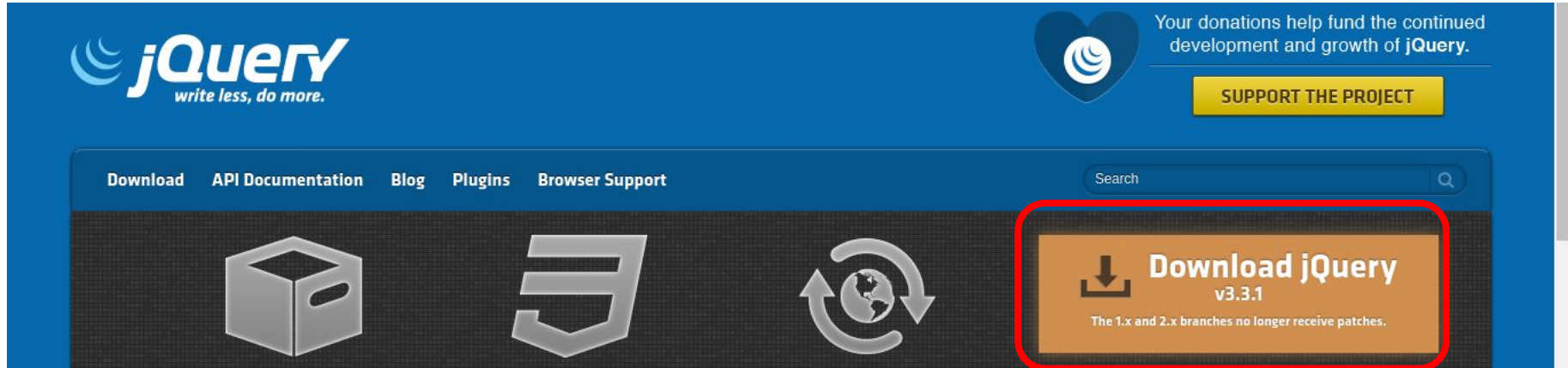
Javascript es un lenguaje de programación dedicado al desarrollo web, muy útil a la hora de manipular información, aunque es muy poderoso y podemos integrarlo muy bien con Django, primero debemos conocer más de Django a fondo antes de dar el siguiente paso. Sin embargo haremos uso de una biblioteca muy famosa hecha con Javascript llamada JQuery.



# jQuery

Para descargar jQuery, visitamos su sitio oficial:

- <https://jquery.com/>



Click a Download jQuery



# El archivo que hay que descargar es el siguiente:

## Downloading jQuery

Compressed and uncompressed copies of jQuery files are available. The uncompressed file is best used during development or debugging; the compressed file saves bandwidth and improves performance in production. You can also download a [sourcemap file](#) for use when debugging with a compressed file. The map file is *not* required for users to run jQuery, it just improves the developer's debugger experience. As of jQuery 1.11.0/2.1.0 the `/** sourceMappingURL` comment is [not included](#) in the compressed file.

To locally download these files, right-click the link and select "Save as..." from the menu.

## jQuery

For help when upgrading jQuery, please see the [upgrade guide](#) most relevant to your version. We also recommend using the [jQuery Migrate plugin](#).

[Download the compressed, production jQuery 3.3.1](#)

[Download the uncompressed, development jQuery 3.3.1](#)

[Download the map file for jQuery 3.3.1](#)

**Descargar** y guardar en la carpeta "js" que está dentro de la carpeta de materialize.

# Creando la carpeta **static**

Para poder hacer uso de los archivos estáticos en nuestra aplicación debemos crear una carpeta llamada "static" dentro de nuestra aplicación y ahí dentro colocar las carpetas de "css" y "js" que contienen nuestros archivos de materialize y jQuery, la estructura se ve de la siguiente forma:

## FOLDERS

misitio

blog

migrations

static

css

~~materialize.css~~

materialize.min.css

js

jquery-3.3.1.min.js

~~materialize.js~~

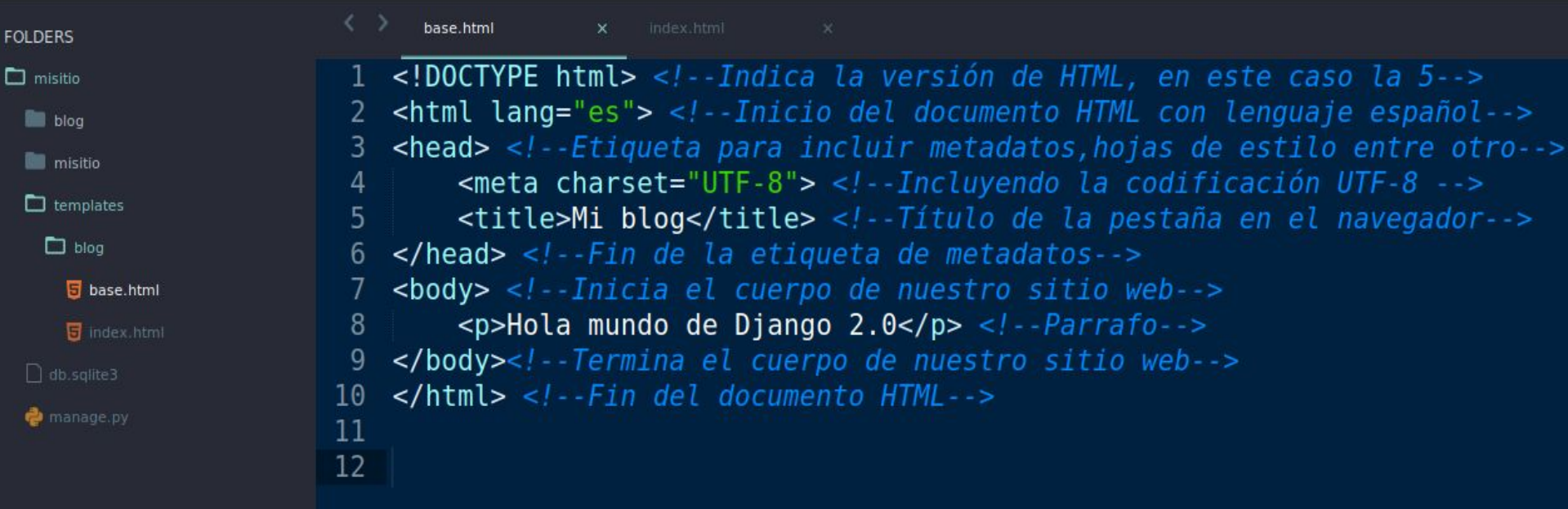
materialize.min.js

Borraremos de las carpetas los archivos: *materialize.css* y *materialize.js*, ya que esos son archivos completos y no minificados(que tienen un menor peso y el código está escrito en una sola línea).

# El template **base**



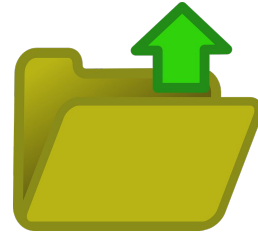
Ya que tenemos los archivos en su lugar correspondiente, procederemos a crear un nuevo template que nos servirá de base al momento de manejar nuestra aplicación. Lo nombraremos "**base.html**" y ahí cortaremos el código del archivo "**index.html**" y lo pondremos en "**base.html**".



```
1 <!DOCTYPE html> <!--Indica la versión de HTML, en este caso la 5-->
2 <html lang="es"> <!--Inicio del documento HTML con lenguaje español-->
3 <head> <!--Etiqueta para incluir metadatos,hojas de estilo entre otro-->
4     <meta charset="UTF-8"> <!--Incluyendo la codificación UTF-8 -->
5     <title>Mi blog</title> <!--Título de la pestaña en el navegador-->
6 </head> <!--Fin de la etiqueta de metadatos-->
7 <body> <!--Inicia el cuerpo de nuestro sitio web-->
8     <p>Hola mundo de Django 2.0</p> <!--Parrafo-->
9 </body><!--Termina el cuerpo de nuestro sitio web-->
10 </html> <!--Fin del documento HTML-->
11
12
```

Podemos quitar los comentarios del archivo si nos resultan obstáculo para leer nuestro código. En las siguientes diapositivas los eliminaré para tener un código más legible.

# Cargando los archivos estáticos



Para ligar(establecer la comunicación entre HTML y CSS) necesitamos cargar los archivos estáticos mediante la etiqueta de Django **{% load staticfiles %}** la cual colocaremos después de la etiqueta de versión de HTML de nuestro archivo base como se muestra a continuación:

```
< > base.html x index.html x
1 <!DOCTYPE html> <!--Indica la versión de HTML, en este caso la 5-->
2 {% load staticfiles %}
3 <html lang="es"> <!--Inicio del documento HTML con lenguaje español-->
```

# Ligando los archivos estáticos



Para ligar los archivos estáticos, necesitaremos hacer uso de las etiquetas comunes de HTML `<link>` y `<script></script>` pero añadiendo en el atributo ***href*** y ***src*** respectivamente la etiqueta especial de django:

- `{% static 'ruta_del_archivo_estático' %}`

```
base.html x
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" href="{% static 'css/materialize.css' %}">
  <title>M1 blog</title>
</head>
<body>
  <p>Hola mundo de Django 2.0</p>
</body>
<script src="{% static 'js/jquery-3.3.1.min.js' %}"></script>
<script src="{% stati 'js/materialize.min.js' %}"></script>
</html>
```

El archivo *materialize.min.js* depende directamente de *jQuery* por lo que es indispensable colocarlos de forma descendente.



# Un vistazo a la documentación de Materialize

Ya teniendo los archivos ligados a nuestro código de HTML, podemos ir a ver qué elementos nos ofrece Materialize, vamos a buscar una barra de navegación(Mobile Collapse Button la mejor para nuestro proyecto) y la incluiremos en nuestro `<body></body>`





1.0.0-rc.2 ▼

Navbar



[Navbar.html](#)

Getting Started

CSS

Components

Badges

Buttons

Breadcrumbs

Cards

Collections

# Mobile Collapse Button

Logo


Sass

Components

Javascript

Mobile

**BUTTONS**

When your nav bar is resized, you will see that the links on the right turn into a hamburger icon . Take a look at the example below to get this functionality. Add the entire `sidenav-trigger` line to your `nav`.

language-markup

```
<nav>
  <div class="nav-wrapper">
    <a href="#" class="brand-logo">Logo</a>
    <a href="#" data-target="mobile-demo" class="sidenav-trigger"><i
  <ul class="right hide-on-med-and-down">
    <li><a href="sass.html">Sass</a></li>
    <li><a href="badges.html">Components</a></li>
    <li><a href="collapsible.html">Javascript</a></li>
    <li><a href="mobile.html">Mobile</a></li>
  </ul>
</div>
</nav>

<ul class="sidenav" id="mobile-demo">
```

```
<body>
  <!--Inicio de barra de navegación-->
  <nav>
    <div class="nav-wrapper">
      <a href="#" class="brand-logo">Logo</a>
      <a href="#" data-target="mobile-demo" class="sidenav-trigger"><i
class="material-icons">menu</i></a>
      <ul class="right hide-on-med-and-down">
        <li><a href="sass.html">Sass</a></li>
        <li><a href="badges.html">Components</a></li>
        <li><a href="collapsible.html">Javascript</a></li>
        <li><a href="mobile.html">Mobile</a></li>
      </ul>
    </div>
  </nav>
  <ul class="sidenav" id="mobile-demo">
    <li><a href="sass.html">Sass</a></li>
    <li><a href="badges.html">Components</a></li>
    <li><a href="collapsible.html">Javascript</a></li>
    <li><a href="mobile.html">Mobile</a></li>
  </ul>
  <!--Fin de barra de navegación-->
</body>
```

# Etiqueta **block** y **extends**

El archivo base nos ayudará a ahorrar código de forma que cargará el contenido de él, dentro de otros archivos HTML, para esto debemos agregar la etiqueta:

- `{% extends 'ruta_del_archivo_base' %}`

Para mostrar el contenido de otros documentos de HTML en el base debemos hacer uso de la etiqueta:

- `{% block content %} {% endblock %}`

en ambos archivos.

base.html

index.html

index.html

```
{% extends 'blog/base.html' %}
```

base.html

```
{% block content %}
```

```
{% endblock %}
```

base.html

index.html

```
<li><a href="collapsible.html">Javascript</a></li>
<li><a href="mobile.html">Mobile</a></li>
</ul>
<!--Fin de barra de navegación-->
```

```
{% block content %}
```

```
{% endblock %}
```

```
</body>
```

# Agregando contenido

En este momento, cualquier etiqueta o información que coloquemos dentro de la etiqueta `{% block content %}` `{% endblock %}` se mostrará en nuestra página.

A screenshot of a code editor with a dark theme. At the top, there are two tabs: 'base.html' and 'index.html', with 'index.html' being the active tab. The code in the editor is as follows:

```
{% extends 'blog/base.html' %}

{% block content %}

<h3>Probando la etiqueta block. </h3>

{% endblock %}
```

# ¡Levantemos el servidor!



Hasta este momento debemos poder ver la implementación del framework de CSS y nuestras etiquetas de Django, si no es así, regresa a repasar las páginas anteriores o envía un correo con tus dudas a:

[jorgechavez.proteco@gmail.com](mailto:jorgechavez.proteco@gmail.com)

# Ejercicio

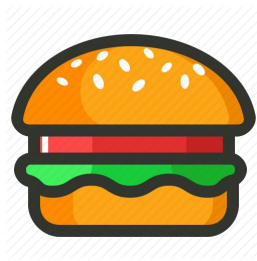


Muy bien, ahora te toca a ti aventurarte en las etiquetas de Django y un poco de CSS, tu ejercicio será:

- Agregar una etiqueta de bloque de Django para que cambie el título de cada página a la que carguemos el archivo base.
- Agregar un footer(lo encontrarás en la documentación de materialize) a tu página base.



# ¿Hamburguesas?



Sí, el icono con el cual se despliega el menú lateral se llama hamburger icon, vamos a ver cómo hacer que se me muestre añadiendo el siguiente link debajo de nuestra etiqueta en la que ligamos nuestro CSS:

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

```
<link rel="stylesheet" href="{% static 'css/materialize.css' %}">  
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"  
rel="stylesheet">
```

# Se ve, pero doy click y no funciona.

Sí, no te equivocaste, la hamburguesa no funciona ya que nos falta agregar un poco de código de jQuery para darle vida.

Es hora de crear un nuevo archivo de Javascript, y ligarlo(tú puedes), le llamaremos “app.js” y colocaremos el código de la siguiente página dentro:

# Inicialización para el efecto de la hamburguesa

```
$(document).ready(function(){  
    $('.sidenav').sidenav();  
});
```

# Tu aplicación ahora se ve cool



¡Felicidades! has agregado archivos estáticos a tu proyecto de Django, estás listo para dar un paso hacia los modelos de Django pero antes haremos una pausa para ver una herramienta que nos ayudará a mantener nuestro código al día y posteriormente hacer un **deploy**.

# Git, gitHub... Controlando mis versiones.



**GitHub**

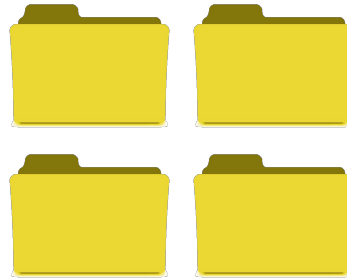
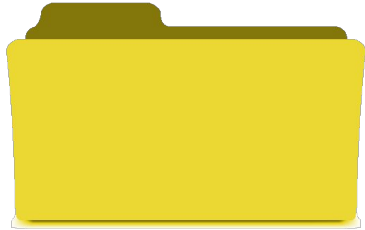


La herramienta de la que hablé anteriormente se llama Git, y su versión en “línea” se llama gitHub, veamos de qué se trata, cómo configurarla y utilizarla.

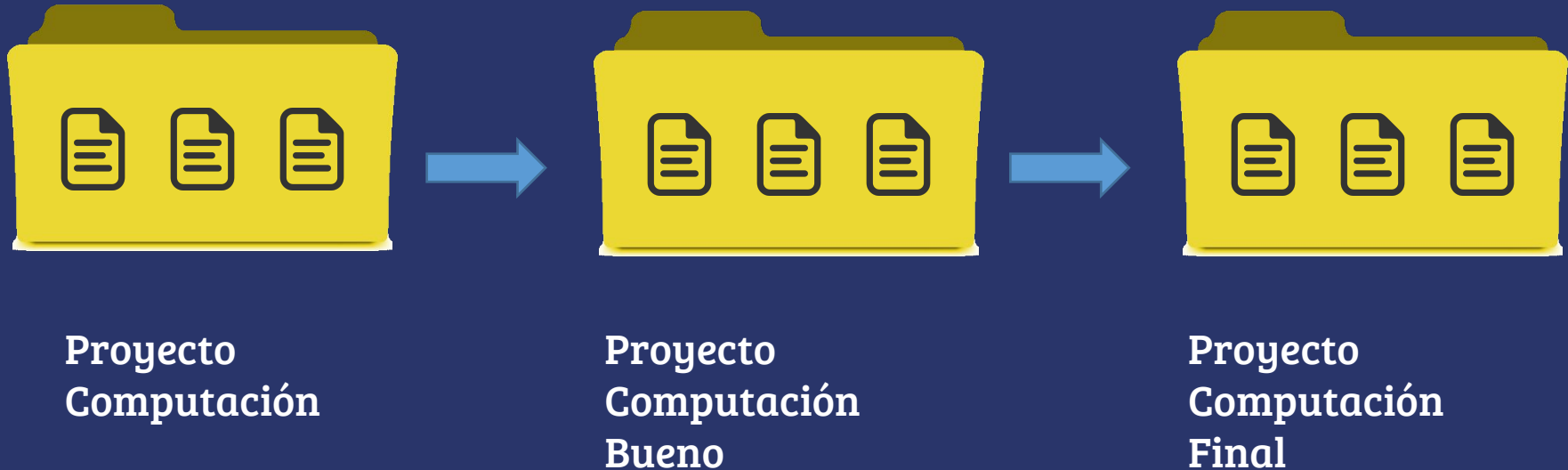
# Controlador de versiones

## git

**Un sistema de control de versiones nos registra los cambios realizados en un archivo o un conjunto de archivos.**



# Imaginemos que tenemos un proyecto





# Entonces eso es **git**

***git* lo que hace es monitorear todo en nuestro proyecto (agregar cambiar o eliminar algo).**



**Y nos permite viajar en el tiempo entre las versiones**



**Así como tener una mejor organización**

## **Working Directory**

**Área de trabajo en nuestro  
proyecto**

## **Repositorio**

**Registro de nuestro  
proyecto**

# Entonces el flujo de trabajo es



# Instalación de **git**

**<https://git-scm.com/downloads>**

# Primeros pasos

Lo primero que debemos hacer es

```
git config --global user.name "nombre"
```

```
git config --global user.email "email"
```

```
git config --global --list
```

# Agregando mis archivos listos

Para agregar mis archivos necesito el siguiente comando:

```
git add .      o en su defecto  
git add nombre_archivo_listo
```

En este momento, nuestros archivos ya están listos para registrar los cambios, este paso lo haremos con el comando:

```
git commit -m "Aquí va algún texto  
que identifique a la versión"
```



Ahora estamos listos para subir  
nuestro proyecto a gitHub, para ello  
tenemos un bonito tutorial:

<https://www.youtube.com/watch?v=lCj0xpkLXE4>

¡Nos vemos la próxima amigos!