# Data Science Portfolio

Saber programar en Python, tener conocimientos de machine learning, visualización, automatización, business intelligence… Todo esto es fundamental para convertirse en un científico de datos, pero no sirve de nada si no lo vendemos bien. Hay que demostrar que sabemos todo eso que hemos aprendido, y la mejor manera es mediante un portfolio.

El trabajo de Data Scientist exige una gran dedicación, compromiso y aprendizaje constante, que se tiene que ver reflejado más allá del curriculum.

Por supuesto, además del portfolio, necesitas apoyarte también en la red social para buscar trabajo más utilizada, que es [LinkedIn.](https://es.linkedin.com/) Antes de crear un portfolio, asegúrate de tener el LinkedIn actualizado.

Otras redes sociales utilizadas: Twitter, StackOverflow y Kaggle.

En [este artículo](https://towardsdatascience.com/how-to-build-a-data-science-portfolio-5f566517c79c) tienes muy buenos consejos sobre cómo crear tu perfil público como Data Scientist

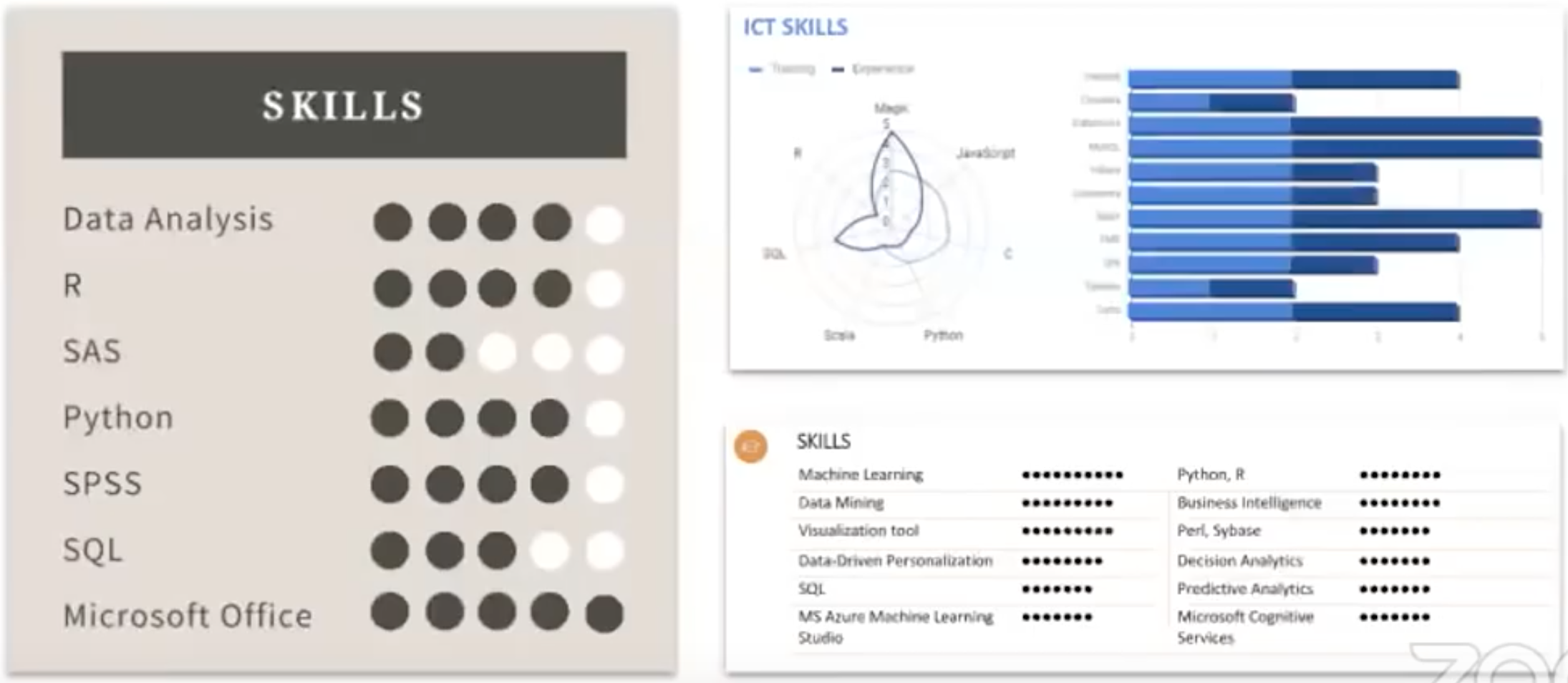
## ¿Qué incluir en el portfolio?

Todo lo que tenga que ver con **proyectos personales. NO incluir ejercicios de clase**. Las opciones serían las siguientes:

1. Trabajos personales del bootcamp
2. Hackathones/Datathones
3. Analíticas personales. Por ejemplo, descargarse los datos de COVID de alguna página del gobierno y realizar una analítica sobre su evolución o impacto por comunidad autónoma.
4. Competiciones de Kaggle.
5. El CV. Si vas a crear una página con tus proyectos, acuérdate de incluir un apartado con tus habilidades, formación y experiencia laboral.

Intenta mantener el portfolio actualizado con nuevos proyectos o artículos. Si tienes el portfolio en una página web, no tiene por qué ser sólo de tus proyectos, podrías tener un blog donde cuentes tus experiencias, habilidades o descubrimientos como Data Scientist.

¡Se original con el CV! Crea gráficas para tus skills/conocimientos. Luego estas gráficas las podrás embeber en la web como una imágen o un HTML de Plotly.



## Opciones de portfolio

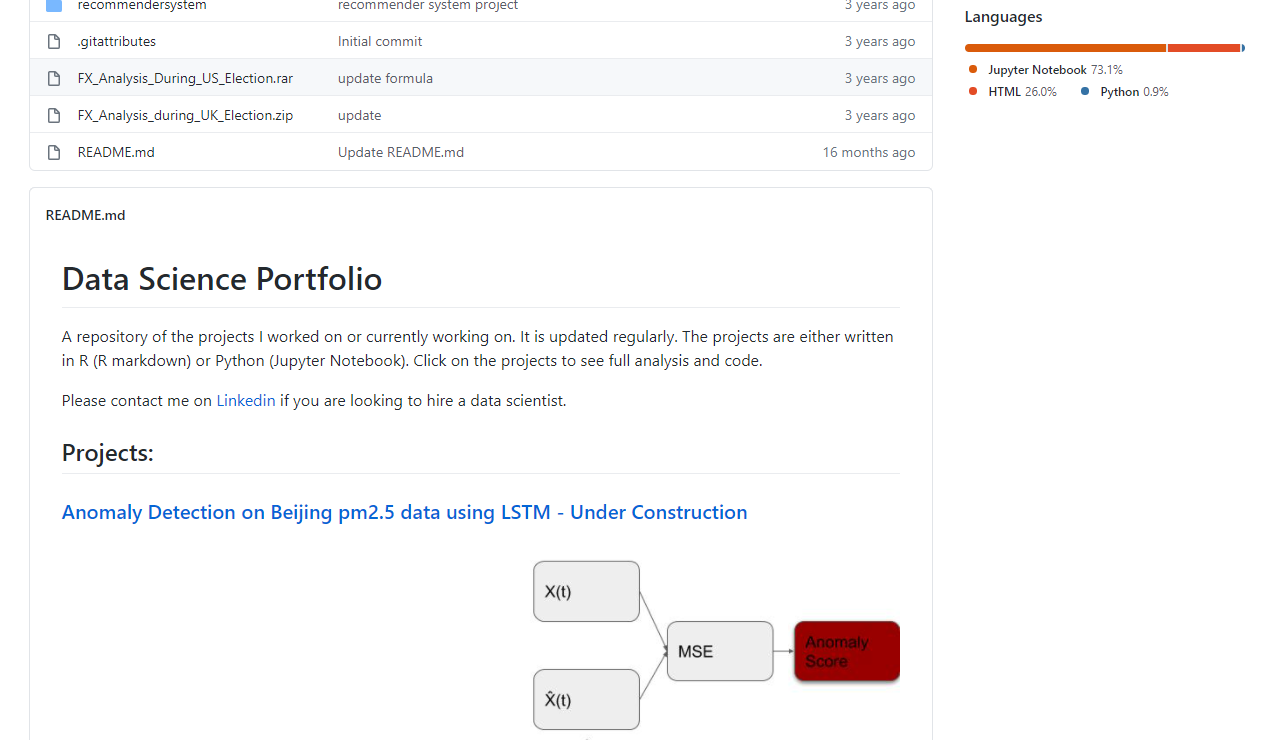
Como todo, tendremos que poner en una balanza tiempo vs portfolio trabajado. En este apartado tienes opciones para crear el portfolio. Desde dedicarle una tarde, a una semana de trabajo, únicamente para crear el portfolio/blog. Después habrá que documentar los proyectos. Tenemos:

1. **GitHub:** directamente en GitHub, crear un repositorio donde tengas los proyectos por carpetas y documentados en Markdown. Rápido
2. **GitHub Pages**: con cuatro clicks (literalmente) puedes crear una página web estática en el propio GitHub. Sencilla, pero que para un blog/portfolio nos sirve. La tecnología que lleva por debajo es Jekyll + Ruby. Desde cero lleva tiempo.
3. **Jekyll Now**: Es una manera sencilla de crear GitHub Pages con Jekyll. Básicamente es una plantilla con todo el front ya creado. Sólo hay que hacer un fork desde GitHub, y empezar a personalizarlo en el propio GitHub o en local. Requiere de aprendizaje pero el resultado es muy bueno para el tiempo dedicado.
4. **Glitch**: página donde publicar tus proyectos o tu CV. Es bastante rápido crearte un perfil mediante una plantilla. En [este artículo](https://towardsdatascience.com/how-to-create-an-elegant-website-for-your-data-science-portfolio-in-10-minutes-577f77d1f693) tienes una buena guía. En esta guía no se va a profundizar en esta opción ya que es bastante sencilla la implementación.
5. **Otras opciones**: siempre puedes crear tu portfolio/blog desde los clásicos [WordPress](https://es.wordpress.com/) o [Wix](https://es.wix.com/), desde los que montar páginas profesionales. Requiere de tiempo si queremos presentar algo que merezca la pena.

**En este documento se profundizará en GitHub Pages y Jekyll para la realización del portfolio.**

Quizá te resulte más atractivo un Wordpress para alojar tu portfolio. Aun así te recomiendo que sigas esta guía en la que aprenderás cómo crear páginas webs fácilmente con GitHub. No solo te resultará útil para manejarte más con GitHub, sino que te servirá también de ayuda para otros propósitos. Crear páginas webs de manera ágil siempre es útil :)

### GitHub



Esta es la opción **más sencilla**, donde tendrás un repositorio donde se encuentren tus proyectos de manera ordenada. Necesitas:

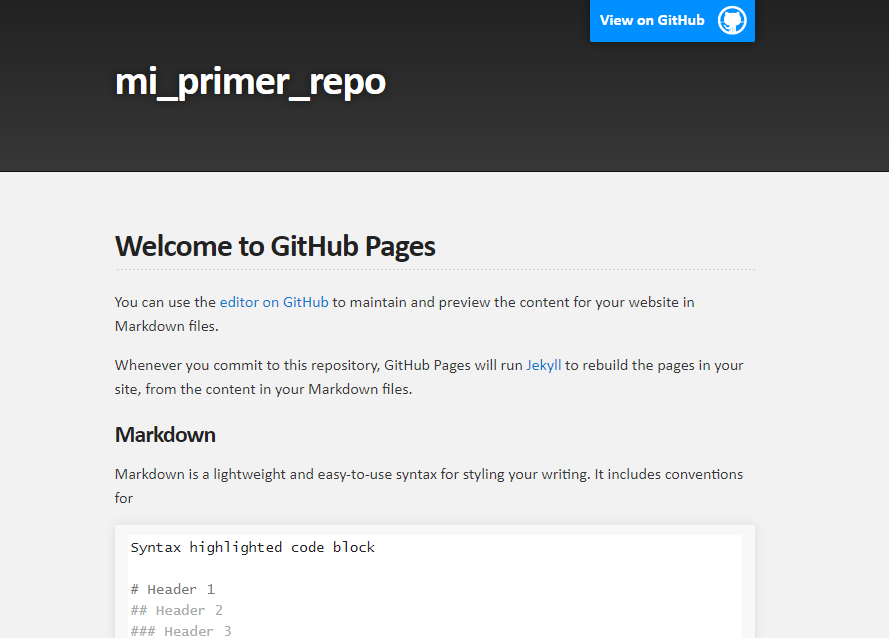
1. [Conocimientos de markdown](https://www.markdownguide.org/basic-syntax/)
2. [Cuenta en GitHub](https://github.com/)

Sigue los siguientes pasos:

1. Crea un nuevo repositorio en GitHub
2. Clonalo a local, para trabajar desde local y subir los cambios. Tienes también la opción de trabajar solo desde GitHub, crear los archivos, modificarlos… pero **se recomienda hacer todo eso en local**, e ir subiendo los cambios.
3. Crea una carpeta por proyecto dentro del repositorio.
4. Guarda las imágenes en una carpeta aparte.
5. Crea una portada donde de un golpe de vista se puedan ver tus proyectos. Alguna imagen de tu analítica, con resumen del proyecto. Pon enlaces en todos los proyectos para ir al detalle (carpeta creadas por proyecto.).
6. [Este sería un buen ejemplo](https://github.com/alexhuang1117/Data-Science-Portfolio)

Recuerda que puedes combinar Markdown con HTML. Con Markdown tienes lo básico (títulos, imágenes, negrita, cursiva, citas...), pero si quieres algo más avanzado, tendrás que desarrollarlo en HTML.

### 2. GitHub Pages



[GitHub tiene un servicio gratuito de hosting](https://pages.github.com/) que te permite tener alojado tu portfolio/blog dentro de uno de tus repositorios, como si de una nueva branch se tratase. ¿Qué nos aporta esto?

1. Es gratis
2. Lo tenemos todo integrado en un mismo repo
3. Es sencillo de crear y desarrollar
4. Es una página web aparte del repositorio.
5. Las webs son responsive
6. Hay plantillas que nos ahorran trabajo

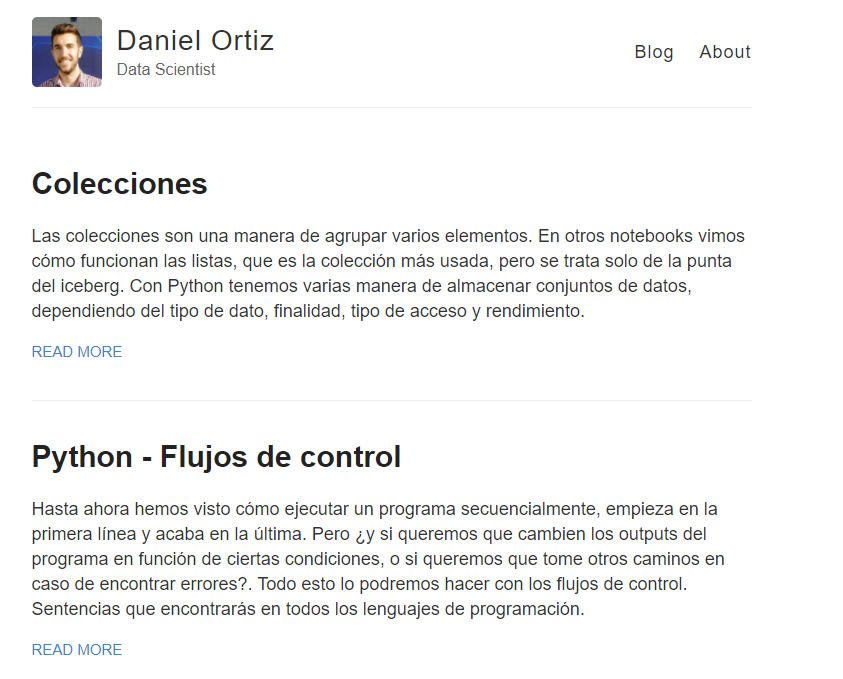
GitHub Pages utiliza Jekyll, que es un framework que permite desarrollar páginas web estáticas. Éstas son las típicas páginas web informativas, como un blog o un portfolio. No llevan registros de usuario, ni cosas complejas como pagos a través de la web. Son páginas que con pocos conocimientos web se pueden desarrollar en unas pocas horas, y a pesar de ello tienen unos diseños muy conseguidos, interfaz de usuario sencilla de usar, fácil de mantener, se pueden añadir anuncios, enlaces, gráficas interactivas, Google Analytics...

NOTA: este tipo de servicios gratuitos vienen siempre con el nombre de la empresa en la URL. No obstante, GitHub te da la opción de poner tu propio dominio.

Ahora bien, **¿cómo creamos una de estas?**

1. Vamos a un repo propio de GitHub, o creamos uno nuevo
2. Apartado Settings del repo -> Opciones -> scroll hasta el apartado GitHub Pages
3. Theme Chooser: elige tema para la web, por ejemplo Slate theme
4. Ya estaría, en el apartado de opciones tendrás ahora una URL con la que acceder a la nueva página web.
5. ¿Cómo modifico la página? Hay una nueva branch en el repo llamada “gh-pages”, desde donde se monta toda la página web. Puedes, o bien modificar esa branch desde la web, o bajarte el repo a local, modificarlo y subir los cambios. Cada vez que hay un nuevo commit tarda un poco en actualizar la página (puede llegar a minutos).
6. Si quieres ponerte desde 0 y aprender bien cómo funciona GitHub pages, este es tu apartado, ahora bien, hay plantillas ya hechas que nos van a ahorrar bastante trabajo (siguientes apartados).

### 3. Jekyll Now



Plantilla que utiliza Jekyll para montar una página web estática de manera sencilla. Jekyll es el motor que hay por debajo de GitHub Pages.

Sigue [esta guía](https://github.com/barryclark/jekyll-now) para crear una página con la plantilla Jekyll Now. Básicamente son los siguientes pasos:

1. **Fork repository**: Es como crear una copia del repositorio del enlace que te he dejado encima. Después, desde “Settings” renombra el repo a nombrequequieras.github.io. De esta manera GitHub lo interpreta como GitHub Pages sin necesidad de añadirle un template como en el caso anterior.

Ya lo tendrías montado. Ahora accede a la página nombrequequieras.github.io donde encontrarás tu nueva web. Vamos a hacer algunas modificaciones

1. **Bajarlo a local:** en este momento tienes dos opciones de trabajo. O bien, desde el propio GitHub, vas creando y modificando archivos (nuevos commits), o te lo bajas a local, modificas, commit….y luego lo subes. Lo que te sea más cómodo. Para 4 cosas básicas, desde el propio GitHub puedes llevarlo a cabo, pero si vamos a dedicarle tiempo a la página es recomendable trabajar en local.
2. **Prose**: otra opción para trabajar es mediante [Prose,](http://prose.io/) que es una interfaz algo más simple de trabajar que GitHub.
3. **Añade un post**: en el repositorio deberías tener una carpeta llamada “\_posts”, aquí se almacenan todos los artículos/proyectos publicados. El archivo “index.html” es el encargado de renderizarlos en la pantalla principal. Esto está genial porque si ya los tienes creados en Markdown para esta web, te servirá en otras simplemente copiando y pegando los archivos markdown. Dos cosas importantes:
   1. Manten el formato “AAAA-M-D-Nombredelpost.md”. Lo necesita Jekyll para renderizar la página.
   2. Mantén el [**Front Matter**](https://jekyllrb.com/docs/front-matter/). Es necesario para renderizar los posts en la web, casi sin que toquemos nada. Por ejemplo, el front matter que viene en el post de ejemplo viene a decir:
      1. **Layout**: post. Significa que utilice el HTML definido como “post”, que encontrarás en la carpeta “\_layouts”. En la plantilla ya vienen definidos layouts, por lo que nosotros sólo tenemos que nombrarlos.
      2. **Title**: You're up and running!. Es el título del post cuando está en versión reducida en la pantalla principal de la página, y cuando vemos el detalle mantiene el mismo título.

Hay más opciones configurables que veremos más adelante.

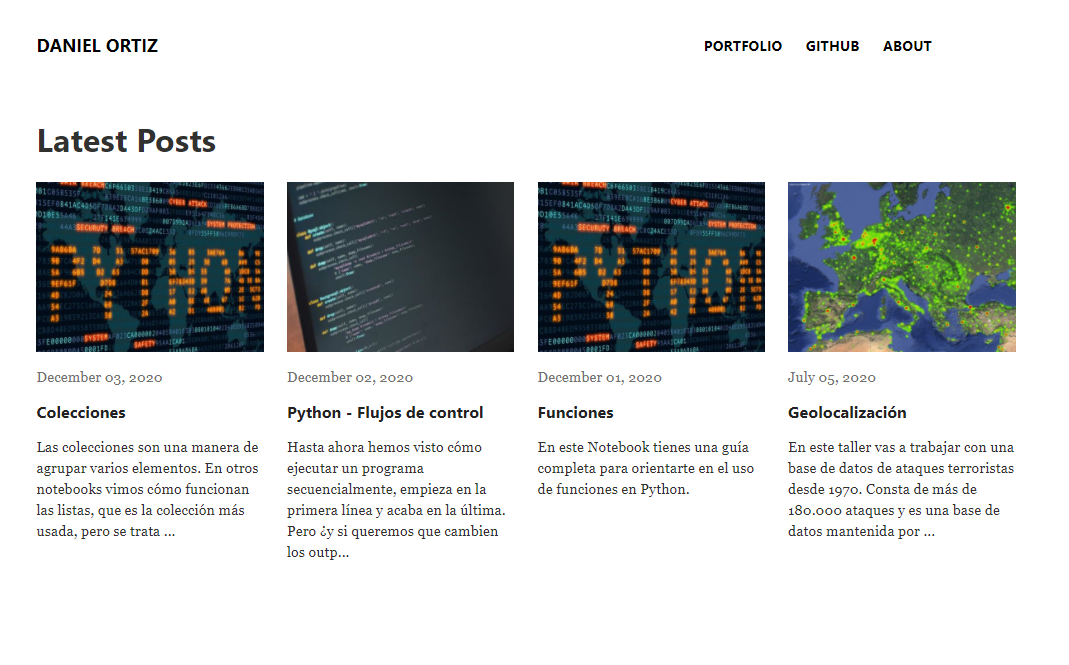
1. **Modifica “\_config.yaml”**: archivo principal de configuración con tu nombre, contacto, nombre del blog, entre otras opciones... Añade enlaces a tu GitHub, LinkedIn y Twitter, simplemente poniendo tus nombres de usuario.
2. **Añade una imagen en un post**: Prueba a introducir una imagen en tu post nuevo. Para ello tendrás que guardarla en la carpeta “images”, y vincularla después desde el post:

![\_config.yml]({{ site.baseurl }}/images/config.png)

1. **About**: fíjate que hay una pestaña de about. En el repo tienes un “about.md”. Prueba a hacer algunos cambios. Aquí debería ir tu carta de presentación.

**Para una guía más completa de Jekyll te recomiendo que sigas** [**este artículo**](https://www.smashingmagazine.com/2014/08/build-blog-jekyll-github-pages/)**.**

### 4. Plantilla Jekyll



Al final de la guía de [Jekyll Now](https://github.com/barryclark/jekyll-now), en el apartado “Other forkable themes”, tienes otras plantillas, que están muy bien. En concreto, para esta guía se va a usar la de [Skinny-bones](https://github.com/mmistakes/jekyll-theme-skinny-bones). Vamos a ir poco a poco. En vez de hacer un fork de este repo, vamos a [bajarlo semivacío](https://github.com/mmistakes/jekyll-theme-skinny-bones/archive/master.zip), para ir probando cosas. Básicamente seguiremos [esta guía](https://mmistakes.github.io/jekyll-theme-skinny-bones/getting-started/#running-jekyll) e haremos una web parecida a la de la imagen superior.

1. **Instalación**: se recomienda trabajar en local, y luego subir a GitHub. Para ello, habrá que instalar Ruby y Jekyll. En [esta página](https://jekyllrb.com/docs/) tienes las instrucciones. [Para Windows](https://jekyllrb.com/docs/installation/windows/).
2. **Running Jekyll**: prueba a bajarte el zip de la guía y ejecutar la página en local mediante el comando “bundle exec jekyll serve”. Accede a la página a través de <http://127.0.0.1:4000>

Debería aparecer la plantilla semivacía:



1. **config.yml**: Vamos a empezar a hacer algunos cambios. Modifica del “config\_yml”:
   1. El título y la descripción
   2. **Imagen de logo y de teaser.** La de logo aparecerá en el menú hamburguesa de la derecha. Y la de teaser es la imagen por defecto de los artículos.
   3. **Nombre y mail** del apartado owner.
2. **Menu navigation (carpeta data -> “navigation.yml”)**:
   * 1. Añade un **enlace a tu GitHub**:

- title: GitHub

url: https://github.com/daorti

* + 1. Añade una **pestaña de about** para acceder a tu carta de presentación:

- title: About

url: /about/

Ahora tendrás un botón nuevo en el menú de navegación de arriba, que de momento no hace nada.

* 1. **About**: ahora hay que implementar lo que va dentro del about. Crea una carpeta nueva en el repo (about), y añde dentro un archivo markdown (index.md). Añade el Front Matter:

---

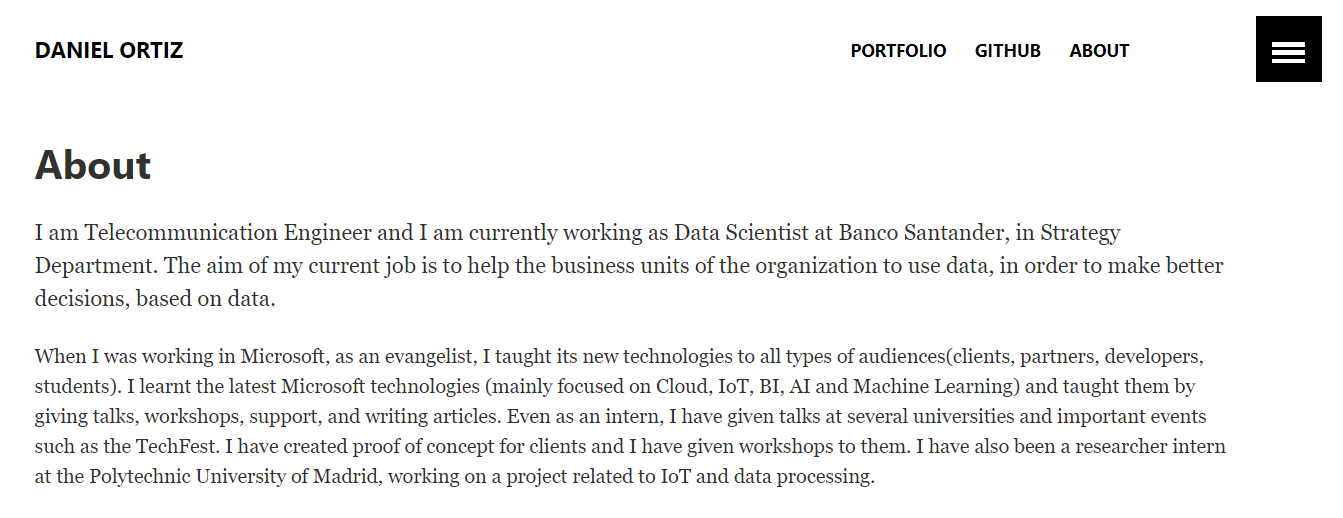
layout: archive

title: About

---

¿Qué estamos haciendo con esto? Archive es un layout de HTML ya implementado en la plantilla. No es más que el esqueleto. Ahora cada vez que pinches en “About” en el menú de arriba, se abrirá esta página.

Lo único que te queda, además del Front Matter, es añadir lo que consideres en esta pestaña, es decir, dentro del archivo “about/index.md”.



1. **Posts**: crea una carpeta de posts, y recicla los creados en el ejercicio anterior. Los mismos archivos Markdown nos sirven ahora también. Hay que hacer algunos cambios. Ahora los posts usarán el layout “article”, que es un HTML que encontrarás en la carpeta “\_layouts”. Simplemente tienes que cambiar “layour” del Front Matter. Añade también la categoría “articles”. Debería quedar el Front Maker así:

---

layout: article

title: Titulo del artículo

categories: articles

---

Ahora el encargado de renderizar todos los posts será el “index.md” de la raíz del proyecto. Cada nuevo post irá a la portada.

¿Por qué hemos añadido la categoría? Porque algunos de los posts que añadiremos serán proyectos para el portfolio, y esta es la manera de distinguirlos.

1. **Portfolio**: lo último que haremos será añadir un apartado de portfolio. Este apartado estará en el menú de arriba de navegación. Para ello tendrás que incluir una nueva entrada en “data/navigation.yml”:

- title: Portfolio

url: /portfolio/

Al igual que en “about”, crea una carpeta en la raíz, llamada “portfolio”, y añade un nuevo archivo markdown: “index.md”. Incluye lo siguiente dentro del index:

<div class="tiles">

{% for post in site.categories.portfolio %}

{% include post-grid.html %}

{% endfor %}

</div>

¿Qué hace este código? Crea un grid de posts, al igual que en la portada, lo único que ahora seleccionará como categoría aquellos que sean “portfolio”. Por tanto, ahora mismo debería estar vacío porque ninguno es portfolio.

Prueba a crear un nuevo post cuya categoría sea portfolio:

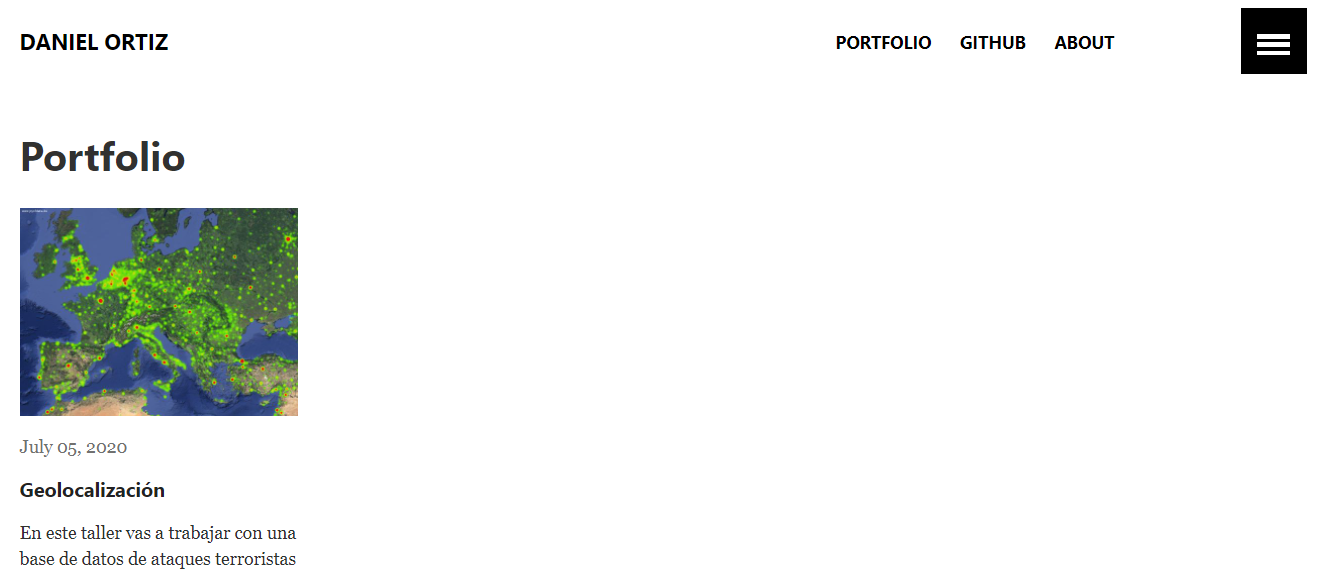
---

layout: article

title: Titulo del artículo

categories: portfolio

---



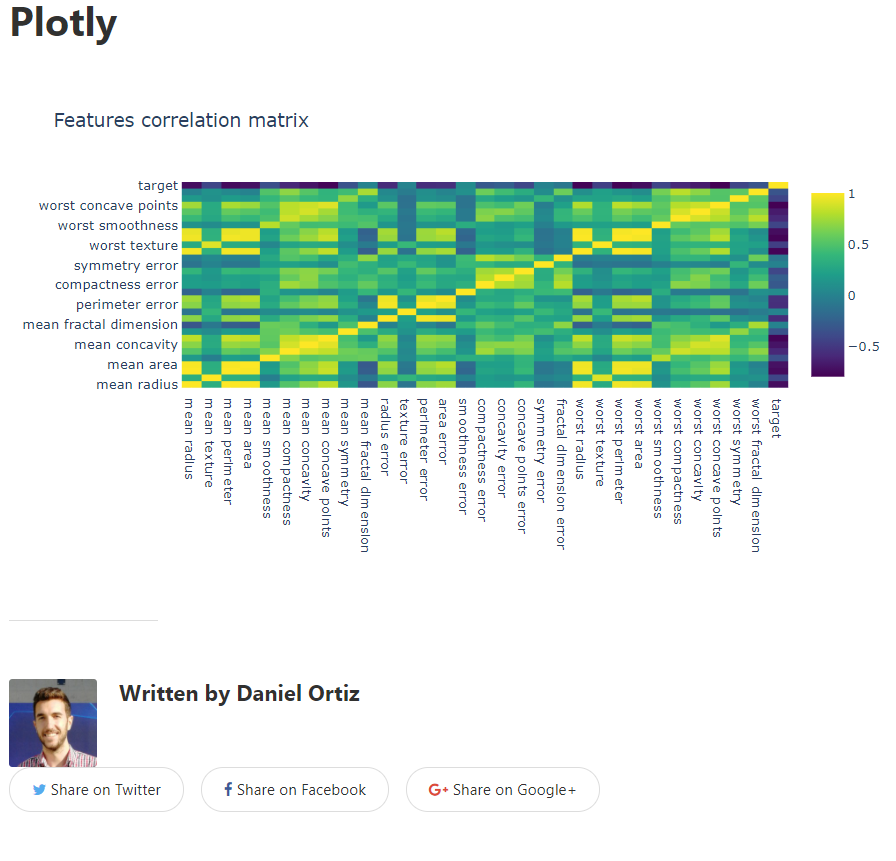
Ya tendríamos una primera versión de la página web, donde en la portada estarán todos los posts, y luego en un botón aparte, todo lo que tenga que ver con proyectos.

1. **Subir a GitHub**: Por hacerlo sencillo, crea un repo vacío en GitHub, haz un clone en local, y añade todo lo que has hecho dentro del repo. Haz un push.

Si en settings/GitHub Pages no te aparece ninguna URL, modifica el nombre del repo a “nombrequequieras.github.io”

Ahora que tienes el nombre de la web, utiliza ese para ponerlo en el apartado “url” del “\_config.xml”.

Finalmente, **recuerda que Markdown es perfectamente compatible con HTML**. ¿Esto en que nos ayuda? Por un lado, todo lo que sepas de HTML lo podrás aplicar aquí, y además toda librería de visualización de Python con gráficos interactivos también te servirá para incrustar gráficos interactivos en tus artículos. Prueba a exportar un gráfico de **Plotly** a HTML, abrelo en texto plano, copia todo lo que veas dentro del primer <div> (las etiquetas de div incluidas. Sin espacios antes del div), y pégalo en tu archivo Markdown.



### 5. Plantilla pro de Skinny-Bones

Ya tenemos los conocimientos básicos para montar una página web decente con GitHub Pages. Para este apartado realiza un fork de [este repo](https://github.com/mmistakes/jekyll-theme-skinny-bones). Y utiliza el branch “gh-pages”. Clona el repo a local y empieza a realizar modificaciones. Recuerda que para que se renderice bien la página tendrás que cambiar la url en el archivo de “\_config.yml”, poniendo la que te haya dado el apartado de Settings del repositorio creado mediante el fork.

