



Convertir un documento Word u ODT en una página web estática HTML compatible con MkDocs utilizando las IA Gemini 3 y ChatGPT 5.2

Serge Tahé, enero de 2026

[| French version](#) | [| English version](#) | [| German version](#) |

Este sitio web se ha creado con el conversor [Word u ODT -> HTML] desarrollado por las IA Gemini 3 y ChatGPT 5.2 en enero de 2026 y descrito en este documento.

Con el conversor Gemini / ChatGPT, se han creado varios sitios web a partir de documentos ODT de LibreOffice y documentos de Word:

Java

- [\[Introduction to the Java Language \(1998\)\];](#)
- [\[Introduction to Web Programming in Java with Servlets and JSP Pages \(2002\)\];](#)
- [\[Introduction to Struts 1.x through Examples \(2003\)\];](#)
- [\[The Basics of MVC Web Development in Java Through Examples \(2006\)\];](#)
- [\[Java 5 Persistence Through Practice \(2007\)\];](#)
- [\[Building a Java EE Web Service with the NetBeans 6.5 IDE and the GlassFish Java EE Server \(2009\)\];](#)
- [\[Introduction to Struts 2 Through Examples \(2012\)\];](#)
- [\[Introduction to Java Server Faces, PrimeFaces, and PrimeFaces Mobile with Examples \(2012\)\];](#)
- [\[Introduction to Java EE with the NetBeans IDE and the GlassFish Application Server \(2012\)\];](#)
- [\[A Client/Server Example - AngularJS 1.x / Spring 4 \(2014\)\];](#)

- [[Introduction to the Java Language and the Spring Ecosystem Through a Case Study \(2015\)](#)];
- [[Introduction to Spring MVC and Thymeleaf through examples \(2015\)](#)];
- [[Working with a relational database using the Spring ecosystem \(2015\)](#)];
- [[Introduction to RxJava. Application to Swing and Android environments \(2016\)](#)];

Android

- [[Introduction to Android Tablet Programming with Android Studio \(2016\)](#)];

.NET

- [[Introduction to VB.NET through Examples \(2004\)](#)];
- [[Web Development with ASP.NET 1.1 \(2004\)](#)];
- [[Spring IoC for .NET \(2005\)](#)];
- [[Creating a Three-Tier Web Application with Spring.NET and VB.NET \(2005\)](#)];
- [[Learning C# Version 3.0 with the .NET Framework 3.5 \(2008\)](#)];
- [[Building a Three-Tier Web Application with ASP.NET 2.0, C#, Spring.NET, and NHibernate \(2010\)](#)];
- [[Introduction to the NHibernate Framework for the .NET Platform \(2011\)](#)];
- [[Introduction to Entity Framework 5 Code First Through Examples \(2012\)](#)];
- [[Introduction to the ASP.NET MVC Framework through Examples \(2013\)](#)];

JavaScript

- [[A Client/Server Example - AngularJS 1.x / Spring 4 \(2014\)](#)];
- [[Introduction to the ECMAScript 6 language through examples \(2019\)](#)];
- [[Introduction to the VUE.JS framework through examples \(2019\)](#)];
- [[Introduction to the NUXT.JS framework through examples \(2019\)](#)];

PHP

- [[MVC Development Methodology for a PHP4 Web Application \(2004\)](#)];
- [[Introduction to the PHP5 language through examples \(2011\)](#)];
- [[Introduction to the PHP7 language through examples \(2019\)](#)];

Python

- [[Introduction to Python 2.7 through Examples \(2012\)](#)];
- [[Introduction to the Python Language and the Flask Web Framework Through Examples \(2020\)](#)];
- [[Generate a Python script using AI tools \(2025\)](#)];

VBScript

- [[Introduction to VBScript \(2002\)](#)];

SQL

- [[Introduction to the SQL language with the Firebird DBMS \(2006\)](#)];

Todos ellos son cursos antiguos de varios cientos de páginas. El tamaño del documento ODT o DOCX no importa al conversor Gemini / ChatGPT. Lo que le importa son las estructuras que se encuentran en él (véase [el capítulo: Ejemplos de este documento](#)).

1. Introducción

El PDF de este documento está disponible | [AQUÍ](#) |.

Los convertidores Gemini 3 / ChatGPT 5.2 [Word u ODT → HTML] están disponibles | [AQUÍ](#) |.

Este documento le ofrece dos convertidores:

- un conversor de LibreOffice ODT a HTML;
 - un conversor de Word DOCX a HTML;
-

En este artículo nos proponemos poner a disposición del lector un convertidor en Python de documentos Word u ODT a una página web estática en HTML. Este convertidor fue creado inicialmente por la IA [Gemini 3](#) y posteriormente por [ChatGPT 5.2](#). Estas dos IA necesitaron 356 iteraciones para producir el convertidor de este documento. Esto supuso varias semanas de depuración. Gemini 3 hizo todo el trabajo al principio. Primero se necesitaron varias decenas de iteraciones para obtener una primera versión más o menos correcta. Luego, periódicamente, añadía un nuevo problema planteado por nuevos documentos ODT. Y ahí, Gemini a menudo retrocedía. Es decir, lo que funcionaba en la etapa N ya no funcionaba en las etapas siguientes. Entonces procedí así: en cuanto Gemini producía un conversor que resolvía uno de mis problemas, lo convertía en una versión de referencia y se lo indicaba a Gemini. Guardaba esta referencia localmente. Luego, cuando veía que Gemini retrocedía demasiado, le pedía que volviera a la última versión de referencia conocida, facilitándosela. Así es como, poco a poco, construimos juntos este conversor: yo expresándole lo que quería, básicamente señalándole las anomalías que encontraba en el sitio HTML generado, y él generando el código solicitado.

He utilizado [Gemini 3](#) con una licencia pro de 22 euros al mes y [ChatGPT 5.2](#) de igual manera.

Gemini / ChatGPT generará dos scripts de Python:

- [convert] para convertir el documento ODT de [LibreOffice](#) o el documento DOCX de Word en un sitio [MkDocs](#);
- [build] para convertir el sitio MkDocs en un sitio HTML estático;

Nunca he mirado el código generado. Quería considerarlos como cajas negras. No hace falta ser desarrollador de Python para seguir este tutorial, ni siquiera desarrollador en general.

Al proponer mejoras del convertidor a la IA Gemini 3, esta se bloqueaba a veces. No conseguía realizar las mejoras solicitadas. En ese momento estábamos en la versión estable V316. Para poder avanzar, le proporcioné esta versión operativa a [ChatGPT 5.2](#) y le pedí las mejoras deseadas. ChatGPT modificó correctamente el código de Gemini para satisfacer mis nuevas peticiones. Por eso considero que fueron estas dos IA las que generaron el conversor.

Posteriormente, seguí utilizando esta técnica. Cuando una IA se atascaba en una funcionalidad, le daba a la otra IA la última versión estable conocida.

Las dos IA tienen métodos diferentes para entregar el código Python solicitado:

- Gemini proporciona el código del script generado en la página de consulta. A continuación, hay que copiar y pegar ese código;
- ChatGPT ofrece un enlace de descarga del script generado;

Más allá de las 1000 líneas de código generado, Gemini mostró graves deficiencias. Debido a las limitaciones técnicas propias de esta IA, esta no conseguía mostrar, en la página de consulta, la totalidad del código generado. Muy a menudo faltaban líneas de código. Debido a esta limitación, a partir de cierto momento ya no fue posible utilizar Gemini 3. Y fue, por tanto, ChatGPT 5.2 quien terminó de escribir el script.

También fue ChatGPT quien generó el conversor de Word a HTML. Le proporcioné el conversor de ODT a HTML que funcionaba y le pedí que lo adaptara para un documento de Word. Lo hizo en 18 iteraciones. Es una constante de estas dos IA: comprenden muy bien los scripts de Python que se les proporcionan y pueden aportarles mejoras e

es. Para mí, esta es la mejor forma de trabajar con ellas.

2. Los ejemplos de este documento

Me gustaría escribir un artículo breve. Las interacciones entre una IA y un usuario se presentan en el artículo [[Generar un script de Python con herramientas de IA](#)], al que a partir de ahora me referiré como [ref1]. Las interacciones con Gemini y ChatGPT solo se presentarán de forma marginal. De todos modos, era imposible presentar todas las iteraciones.

Ahora voy a presentar ejemplos de las particularidades de mis documentos ODT / DOCX que el conversor Gemini / ChatGPT gestiona correctamente. Es este mismo documento el que propondremos para la conversión a HTML mediante el script de Gemini / ChatGPT. Veremos qué hace con él.

2.1. Las listas

El conversor Gemini / ChatGPT sabe gestionar las listas con viñetas y las listas numeradas, incluso si están anidadas:

2.1.1. Listas con viñetas s

- Elemento 1;
- Elemento 2;
- Elemento 3;
 - Elemento 3.1;
 - Elemento 3.1.1
 - Elemento 3.1.2
 - Elemento 3.1.2.1
 - Elemento 3.1.2.2
 - Elemento 3.2;
- Elemento 4;

2.1.2. Listas numeradas

1. Elemento 1;
2. Elemento 2;
 - a. Elemento 2.1
 - i. Elemento 2.1.1
 1. Elemento 2.1.1.1
 2. Elemento 2.1.1.2
 - ii. Elemento 2.1.2
 - b. Elemento 2.2
3. Elemento 3;

2.1.3. Listas mixtas 1

- Elemento 1;
- Elemento 2;
- Elemento 3;
 - Elemento 3.1;
 1. Elemento 3.1.1
 2. Elemento 3.1.2

- Elemento 3.1.2.1
 - Elemento 3.1.2.2
 - Elemento 3.2;
- Elemento 4;

2.1.4. Listas mixtas 2

1. Elemento 1;
2. Elemento 2;
 - a. Elemento 2.1
 - i. Elemento 2.1.1
 - Elemento 2.1.1.1
 - Elemento 2.1.1.2
 - ii. Elemento 2.1.2
 - b. Elemento 2.2
3. Elemento 3;

2.1.5. Listas numeradas manualmente

Por numeración manual se entiende aquí que es el usuario quien establece el número de un párrafo numerado: [clic con el botón derecho del ratón sobre el párrafo numerado / párrafo / párrafo / reiniciar la numeración / comenzar con].

Empiezo una lista con un número distinto de 1.

6. Elemento 6
7. Elemento 7

Aquí interrumpo la lista para decir algo, pero luego quiero continuar con la numeración.

8. Elemento 8
9. elemento 9

A continuación, empiezo una nueva lista numerada:

11. elemento 11
12. elemento 12

2.2. Los bloques de código

Mis cursos contienen muchos bloques de código. A menudo se trata de códigos formateados (negrita, colores en las palabras clave) por los IDE (Eclipse, PyCharm, WebStorm, Netbeans). El conversor reproduce estos códigos formateados tal cual.

Cuando el código no está formateado (código procedente del Bloc de notas o Notepad o...), el conversor Gemini / ChatGPT lo reconoce (Java, C#, XML, HTML, ...) mediante palabras clave del lenguaje incluidas en un archivo de configuración. Cuando reconoce un lenguaje, inserta un marcador (fence) para MkDocs, de modo que este adapte el resaltado sintáctico del código al lenguaje utilizado en el bloque de código.

2.2.1. Bloques de código enriquecido (Eclipse, Visual Studio, etc.)

A continuación se muestran bloques de código enriquecidos por diferentes IDE:

Java

```
1. package istia.st.spring.core;
```

```

2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. clase pública Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // recuperación del contexto Spring
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // recuperamos los beans
16.         Persona p01 = ctx.getBean("persona_01", Persona.class);
17.         Persona p02 = ctx.getBean("persona_02", Persona.class);
18.         List<Persona> club = ctx.getBean("club", new ArrayList<Persona>().getClass());
19.         Piso piso01 = ctx.getBean(Piso.class);
20. ...

```

C#

```

11. using System;
12.
13. namespace Chap1 {
14.     class Impuestos {
15.         static void Main(string[] args) {
16.             // matrices de datos necesarias para el cálculo del impuesto
17.             decimal[] límites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
18. };
19.             decimal[] coeffR = { 0M, 0.068M, 0.191M, 0.283M, 0.374M, 0.426M,
20. 0.481M };
21.             decimal[] coeffN = { 0M, 291,09M, 1322,92M, 2668,39M, 4846,98M,
22. 6883,66M, 9505,54M };
23.
24.             // recuperamos el estado civil
25.             bool OK = false;
26.             string respuesta = null;
27.             while (!OK) {
28.                 Console.WriteLine("¿Está casado/a (S/N)? ");
29.                 respuesta = Console.ReadLine().Trim().ToLower();
30.                 if (respuesta != "s" && respuesta != "n")
31.                     Console.Error.WriteLine("Respuesta incorrecta. Vuelva
32. a intentarlo");
33.                 else OK = true;
34.             } //while
35.             bool marie = respuesta == "o";
36. ...

```

Python

```

21. # -----
22. def mostrar(cadena):
23.     # muestra cadena
24.     print("cadena=%s" % cadena)
25.
26.
27. # -----
28. def mostrar_tipo(variable):
29.     # muestra el tipo de variable
30.     print("tipo[%s]=%s" % (variable, tipo(variable)))
31.
32.
33. # -----
34. def f1(param):
35.     # suma 10 a param
36.     return param + 10

```

```

37.
38.
39. # -----
40. def f2():
41.     # devuelve una tupla de 3 valores
42.     return "uno", 0, 100
43.
44.
45. # ----- programa principal -----
46. ...

```

PHP

```

31. <?php
32.
33. // tipos estrictos para los parámetros de las funciones
34. declare(strict_types=1);
35.
36. // constantes globales
37. define("LÍMITE_QF_MEDIA PARTE", 1551);
38. define("LÍMITE_INGRESOS_SOLTERO PARA REDUCCIÓN", 21037);
39. define("LÍMITE_INGRESOS_PAREJA PARA REDUCCIÓN", 42074);
40. define("VALOR REDUCCIÓN MEDIA PARTE", 3797);
41. define("LÍMITE MÁXIMO DE DESCUENTO SOLTERO", 1196);
42. define("LÍMITE MÁXIMO DE DESCUENTO PARA PAREJAS", 1970);
43. define("LÍMITE IMPUESTO PAREJA PARA DESCUENTO", 2627);
44. define("LÍMITE MÁXIMO IMPUESTO SOLTERO PARA DESCUENTO", 1595);
45. define("DESCUENTO MÁXIMO DEL 10 POR CIENTO", 12502);
46. define("DESCUENTO DIEZ POR CIENTO MÍNIMO", 437);
47.
48. // definición de las constantes locales
49. $DATA = "taxpayersdata.txt";
50. $RESULTADOS = "resultados.txt";
51. $límites = array(9964, 27519, 73779, 156244, 0);
52. $coeffR = array(0, 0.14, 0.3, 0.41, 0.45);
53. $coeffN = array(0, 1394.96, 5798, 13913.69, 20163.45);
54.
55. // lectura de los datos
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "No se puede abrir el archivo de datos [$DATA] para lectura\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // esto es un comentario
43. // constante
44. const nombre = "dupont";
45. // una salida a pantalla
46. console.log("nombre: ", nombre);
47. // una matriz con elementos de diferentes tipos
48. const matriz = ["uno", "dos", 3, 4];
49. // su número de elementos
50. let n = array.length;
51. // un bucle
52. for (let i = 0; i < n; i++) {
53.     console.log("matriz[", i, "] = ", matriz[i]);
54. }
55. // inicialización de 2 variables con el contenido de un array
56. let [cadena1, cadena2] = ["cadena1", "cadena2"];
57. // concatenación de las 2 cadenas
58. const cadena3 = cadena1 + cadena2;
59. // visualización del resultado

```

```
60. console.log([cadena1, cadena2, cadena3]);
61. ...
```

VBScript

```
51. ' cálculo del impuesto de un contribuyente
52. ' el programa debe ejecutarse con tres parámetros: casado, hijos, salario
53. ' casado: carácter 0 si está casado, N si no lo está
54. ' hijos: número de hijos
55. ' salario: salario anual sin céntimos
56.
57. ' declaración obligatoria de las variables
58. Option Explicit
59. Dim error
60.
61. ' recuperamos los argumentos comprobando su validez
62. Dim marie, enfants, salario
63. error = getArguments(marie, enfants, salaire)
64. ' ¿Error?
65. If error(0)<>0 Then wscript.echo error(1) : wscript.quit error(0)
66.
67. ' se recuperan los datos necesarios para el cálculo del impuesto
68. Dim límites, coeffR, coeffN
69. error = getData(límites, coeffR, coeffN)
70. ' ¿Hay error?
71. Si error(0) ≠ 0 Entonces wscript.echo error(1) : wscript.quit 5
72.
73. ' se muestra el resultado
74. wscript.echo "impuesto=" & calcularImpuesto(esposa,hijos,salario,límites,coefR,coefN)
75.
76. ' salimos sin error
77. wscript.quit 0
```

XML

```
61. <?xml version="1.0" encoding="utf-8" ?>
62. <configuración>
63.
64.   <configSections>
65.     <sectionGroup name="spring">
66.       <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
67.       <section name="objects" type="Spring.Context.Support.DefaultSectionHandler, Spring.Core"
68.     />
69.   </sectionGroup>
70. </configSections>
71.
72. <spring>
73.   <context>
74.     <resource uri="config://spring/objects" />
75.   </context>
76.   <objects xmlns="http://www.springframework.net">
77.     <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
78.       <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
79.       <constructor-arg index="1"
80. value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
81.       <constructor-arg index="2" value="select limite, coeffr, coeffn from tranches"/>
82.     </object>
83.     <object name="metier" type="Metier.ImpotMetier, ImpotsV7-metier">
84.       <constructor-arg index="0" ref="dao"/>
85.     </object>
86.   </objects>
87. </spring>
88. </configuración>
```

2.2.2. Bloques de código sin formato (texto sin formato)

Estos son algunos ejemplos de código sin formato:

Resultados de la ejecución

Cabe señalar que el código no comienza en la línea n.º 1.

```
7. C:\Data\st-2020\dev\python\cours-2020\python3-flask-2020\venv\Scripts\python.exe
   C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/bases/bases_01.py
8. nombre=dupont
9. lista[0] = uno
10. lista[1] = dos
11. lista[2]=3
12. lista[3]=4
13. [cadena1,cadena2,cadena1cadena2]
14. cadena = cadena1
15. tipo[4]=<class 'int'>
16. tipo[cadena1] = <class 'str'>
17. tipo[['uno', 'dos', 3, 4]] = <class 'list'>
18. tipo[ha cambiado] = <class 'str'>
19. res1 = 14s
20. (res1, res2, res3) = [uno, 0, 100]
21. lista[0] = uno
22. lista[1] = 0
23. lista[2] = 100
24. lista[0] = 8
25. lista[1] = 5
26. suma = 13
27.
28. Proceso finalizado con código de salida 0
```

Este caso requirió decenas de iteraciones. El conversor nunca encontraba el número de la primera línea de código. Finalmente, le pregunté a Gemini cómo formatear el bloque de código para que lo reconociera. Esto es lo que me respondió:

Dans LibreOffice :

1. Placez votre curseur sur la **première ligne** de votre bloc de code numéroté (celle qui doit être le n°5).
2. Faites un **Clic-Droit > Paragraphe > Paragraphe...**
3. Allez dans l'onglet **Plan & numérotation** (ou *Outline & Numbering*).
4. Cochez la case **Recommencer la numérotation** (*Restart numbering at this paragraph*).
5. Cochez/Réglez **Commencer à** (*Start with*) sur **5**.
6. Cliquez sur **OK** et sauvegardez votre ODT.

Pourquoi ça marche ? Cette méthode oblige LibreOffice à inscrire un attribut explicite sur le paragraphe ou l'item de liste (`text:start-value="5"`), ce que le script détecte immédiatement. La méthode via l'icône de liste ("Personnaliser") modifie parfois seulement le *style* visuel sans changer la valeur logique sous-jacente dans le XML, surtout sur les documents importés de Word.

Seguí este procedimiento y funcionó. Yo utilizaba el método a través del icono de lista (Personalizar). Por lo tanto, es posible que haya que modificar ciertos elementos del documento ODT/DOCX para obtener lo que se desea.

El conversor sabe gestionar bloques de código sin numerar.

```

lista[0]=uno
lista[1]=0
lista[2]=100
lista[0]=8
lista[1]=5
suma = 13

```

Estos son los mismos ejemplos que en el apartado 2.2.1, pero sin enriquecer. En este caso, serán las palabras clave del archivo de configuración las que guiarán al convertidor hacia el lenguaje adecuado.

Java

```

1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. clase pública Demo01 {
10.
11. @SuppressWarnings({ "unchecked", "resource" })
12. public static void main(String[] args) {
13. // recuperación del contexto Spring
14. ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15. // recuperamos los beans
16. Persona p01 = ctx.getBean("persona_01", Persona.class);
17. Persona p02 = ctx.getBean("persona_02", Persona.class);
18. List<Persona> club = ctx.getBean("club", new ArrayList<Persona>().getClass());
19. Piso piso01 = ctx.getBean(Piso.class);
20. ...

```

C#

```

11. using System;
12.
13. namespace Chap1 {
14.     class Impuestos {
15.         static void Main(string[] args) {
16.             // Tablas de datos necesarias para el cálculo del impuesto
17.             decimal[] límites = { 4962M, 8382M, 14753M, 23888M, 38868M, 47932M, 0M
18. };
19.             decimal[] coeffR = { 0M, 0,068M, 0,191M, 0,283M, 0,374M, 0,426M,
20. 0,481M };
21.             decimal[] coeffN = { 0M, 291,09M, 1322,92M, 2668,39M, 4846,98M,
22. 6883,66M, 9505,54M };
23.
24.             // recuperamos el estado civil
25.             bool OK = false;
26.             string respuesta = null;
27.             while (!OK) {
28.                 Console.Write("¿Está casado/a (S/N)? ");
29.                 respuesta = Console.ReadLine().Trim().ToLower();
30.                 if (respuesta != "s" && respuesta != "n")
31.                     Console.Error.WriteLine("Respuesta incorrecta. Vuelva
32. a intentarlo");
33.                 else OK = true;
34.             } //while
35.             bool marie = respuesta == "o";
36. ...

```

Python

```

21. # -----

```

```

22. def mostrar(cadena):
23.     # muestra cadena
24.     print("cadena=%s" % cadena)
25.
26.
27. # -----
28. def mostrar_tipo(variable):
29.     # muestra el tipo de variable
30.     print("tipo[%s]=%s" % (variable, tipo(variable)))
31.
32.
33. # -----
34. def f1(param):
35.     # suma 10 a param
36.     return param + 10
37.
38.
39. # -----
40. def f2():
41.     # devuelve una tupla de 3 valores
42.     return "uno", 0, 100
43.
44.
45. # ----- programa principal -----
46. ...

```

PHP

```

31. <?php
32.
33. // tipos estrictos para los parámetros de las funciones
34. declare(strict_types=1);
35.
36. // constantes globales
37. define("LÍMITE_QF_MEDIA PARTE", 1551);
38. define("LÍMITE_INGRESOS_SOLTERO PARA REDUCCIÓN", 21037);
39. define("LÍMITE_INGRESOS PAREJA PARA REDUCCIÓN", 42074);
40. define("VALOR REDUCCIÓN MEDIA PARTE", 3797);
41. define("LÍMITE MÁXIMO DE DESCUENTO SOLTERO", 1196);
42. define("LÍMITE MÁXIMO DE DESCUENTO PARA PAREJAS", 1970);
43. define("LÍMITE IMPUESTO PAREJA PARA DESCUENTO", 2627);
44. define("LÍMITE MÁXIMO IMPUESTO SOLTERO PARA DESCUENTO", 1595);
45. define("DESCUENTO MÁXIMO DEL 10 POR CIENTO", 12502);
46. define("DESCUENTO DIEZ POR CIENTO MÍNIMO", 437);
47.
48. // definición de las constantes locales
49. $DATA = "taxpayersdata.txt";
50. $RESULTADOS = "resultados.txt";
51. $límites = array(9964, 27519, 73779, 156244, 0);
52. $coeffR = array(0, 0.14, 0.3, 0.41, 0.45);
53. $coeffN = array(0, 1394.96, 5798, 13913.69, 20163.45);
54.
55. // lectura de los datos
56. $data = fopen($DATA, "r");
57. if (!$data) {
58.     print "No se puede abrir el archivo de datos [$DATA] para lectura\n";
59.     exit;
60. }
61.
62. ...

```

ECMAScript

```

41. 'use strict';
42. // esto es un comentario
43. // constante
44. const nombre = "dupont";

```

```

45. // una salida a pantalla
46. console.log("nombre: ", nombre);
47. // una matriz con elementos de diferentes tipos
48. const matriz = ["uno", "dos", 3, 4];
49. // su número de elementos
50. let n = array.length;
51. // un bucle
52. for (let i = 0; i < n; i++) {
53.   console.log("matriz[" + i, "] = ", matriz[i]);
54. }
55. // inicialización de 2 variables con el contenido de un array
56. let [cadena1, cadena2] = ["cadena1", "cadena2"];
57. // concatenación de las 2 cadenas
58. const cadena3 = cadena1 + cadena2;
59. // visualización del resultado
60. console.log([cadena1, cadena2, cadena3]);
61. ...

```

VBScript

```

51. ' cálculo de los impuestos de un contribuyente
52. ' el programa debe ejecutarse con tres parámetros: casado, hijos, salario
53. ' casado: carácter 0 si está casado, N si no lo está
54. ' hijos: número de hijos
55. ' salario: salario anual sin céntimos
56.
57. ' declaración obligatoria de las variables
58. Option Explicit
59. Dim error
60.
61. ' recuperamos los argumentos comprobando su validez
62. Dim marie, enfants, salaire
63. error = getArguments(marie, enfants, salaire)
64. ' ¿Error?
65. If error(0) > 0 Then wscript.echo error(1) : wscript.quit error(0)
66.
67. ' se recuperan los datos necesarios para el cálculo del impuesto
68. Dim límites, coeffR, coeffN
69. error = getData(límites, coeffR, coeffN)
70. ' ¿Hay error?
71. Si error(0) ≠ 0 Entonces wscript.echo error(1) : wscript.quit 5
72.
73. ' se muestra el resultado
74. wscript.echo "impuesto=" & calcularImpuesto(esposa,hijos,salario,límites,coefR,coefN)
75.
76. ' salimos sin error
77. wscript.quit 0

```

XML

```

61. <?xml version="1.0" encoding="utf-8" ?>
62. <configuración>
63.
64.   <configSections>
65.     <sectionGroup name="spring">
66.       <section name="context" type="Spring.Context.Support.ContextHandler, Spring.Core" />
67.       <section name="objects" type="Spring.Context.Support.DefaultSectionHandler, Spring.Core"
68.     />
69.   </grupoDeSecciones>
70. </configSections>
71.
72. <spring>
73.   <context>
74.     <resource uri="config://spring/objects" />
75.   </context>
76.   <objects xmlns="http://www.springframework.net">
77.     <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">

```

```

77.     <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
78.     <constructor-arg index="1"
value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
79.     <constructor-arg index="2" value="select limite, coeffr, coeffn from tranches"/>
80.     </object>
81.     <objeto nombre="metier" tipo="Metier.ImpotMetier, ImpotsV7-metier">
82.     <argumento-constructor index="0" ref="dao"/>
83.     </object>
84. </objects>
85. </spring>
86. </configuración>

```

HTML

```

71. <!DOCTYPE HTML>
72. <HTML>
73.   <head>
74.     <title>Laragon</title>
75.
76.     <link href="https://fonts.googleapis.com/css?family=Karla:400 " rel="stylesheet"
type="text/css">
77.
78.     <style>
79.       HTML, body {
80.         height: 100%;
81.       }
82.
83.       body {
84.         margin: 0;
85.         padding: 0;
86.         ancho: 100%;
87.         display: table;
88.         peso de fuente: 100;
89.         font-family: 'Karla';
90.       }
91.
92.       .container {
93.         alineación del texto: centrada;
94.         display: table-cell;
95.         vertical-align: middle;
96.       }
97.
98.       .content {
99.         alineación del texto: centrada;
100.        display: inline-block;
101.      }
102.
103.      .title {
104.        tamaño de fuente: 96px;
105.      }
106.
107.      .opt {
108.        margin-top: 30px;
109.      }
110.
111.      .opt a {
112.        text-decoration: none;
113.        tamaño de fuente: 150%;
114.      }
115.
116.      a:hover {
117.        color: red;
118.      }
119.    </style>
120.  </head>
121.  <body>
122.    <div class="container">
123.      <div class="content">
124.        <div class="title" title="Laragon">Laragon</div>

```

```

125.
126.         <div class="info"><br />
127.             Apache/2.4.35 (Win64) OpenSSL/1.1.0i PHP/7.2.11<br />
128.             Versión de PHP: 7.2.11 <span><a title="phpinfo()" href="/?
q=info">info</a></span><br />
129.             Raíz del documento: C:/misprogramas/laragon-lite/www<br />
130.
131.         </div>

```

2.3. Los enlaces

El conversor Gemini / ChatGPT conserva los enlaces externos del documento ODT / DOCX. Por ejemplo, Gemini 3 o [\[Generar un script de Python con herramientas de IA\]](#).

Sabe gestionar un enlace a un capítulo [Enlace a un capítulo](#)

Una referencia a un capítulo: 2.1.1 .

Una referencia a una marca de texto anterior: Gemini 3

Una referencia a un marcador de texto posterior: GitHub

2.4. El enriquecimiento del texto

El conversor sabe gestionar la negrita, la cursiva, el subrayado y el resaltado. Respeta el color del resaltado.

Un texto con palabras en **negrita**, *cursiva*, subrayadas o **resaltadas** o **resaltadas** o **resaltadas**.

Lo mismo ocurre con los enlaces: [\[Generar un script de Python con herramientas de IA\]](#).

El **conversor** también **gestiona** el **color** de los **caracteres**.

También gestiona los bordes superior e inferior de los párrafos.

- También gestiona los bordes superior e inferior de los párrafos.
 - También gestiona los bordes superior e inferior de los párrafos.
 - También gestiona los bordes superior e inferior de los párrafos.
-

1. También gestiona los bordes superior e inferior de los párrafos.
 2. También gestiona los bordes superior e inferior de los párrafos.
 3. También gestiona los bordes superior e inferior de los párrafos.
-

10. Un texto con palabras en **negrita**, *cursiva*, subrayadas o **resaltadas**.
11. Lo mismo ocurre con los enlaces: [\[Generar un script de Python con herramientas de IA\]](#).
12. El **conversor** también **gestiona** el **color** de los **caracteres**.

- Un texto con palabras en **negrita**, *cursiva*, subrayadas o **resaltadas**.
- Lo mismo ocurre con los enlaces: [\[Generar un script de Python con herramientas de IA\]](#).
- El **conversor** también **gestiona** el **color** de los **caracteres**.

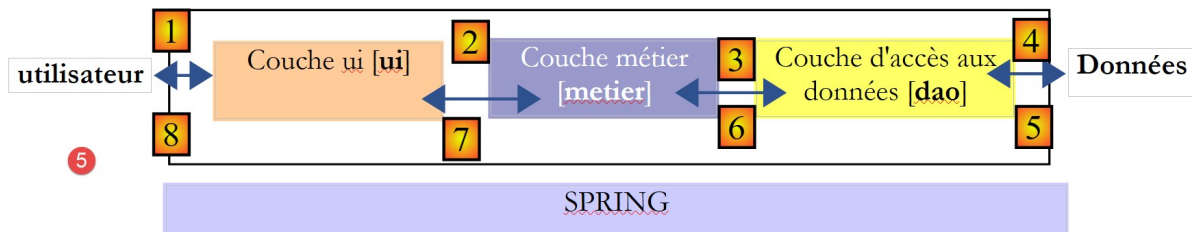
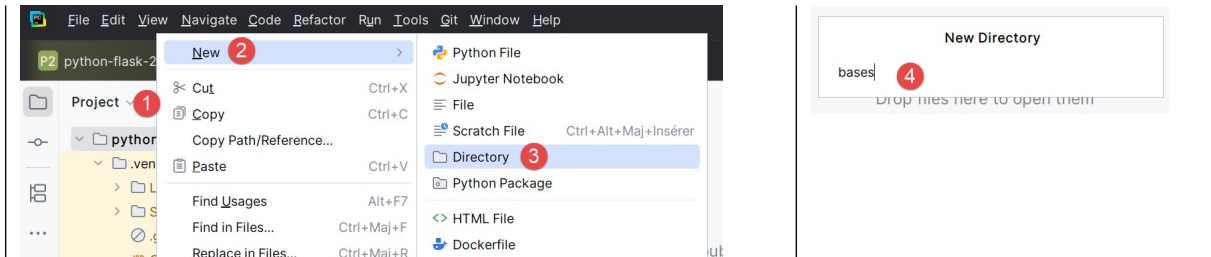
También gestiona el fondo del párrafo

- También gestiona el fondo del párrafo
- 1. También gestiona el fondo del párrafo

2.5. Un **título** también se puede *enriquecer*.

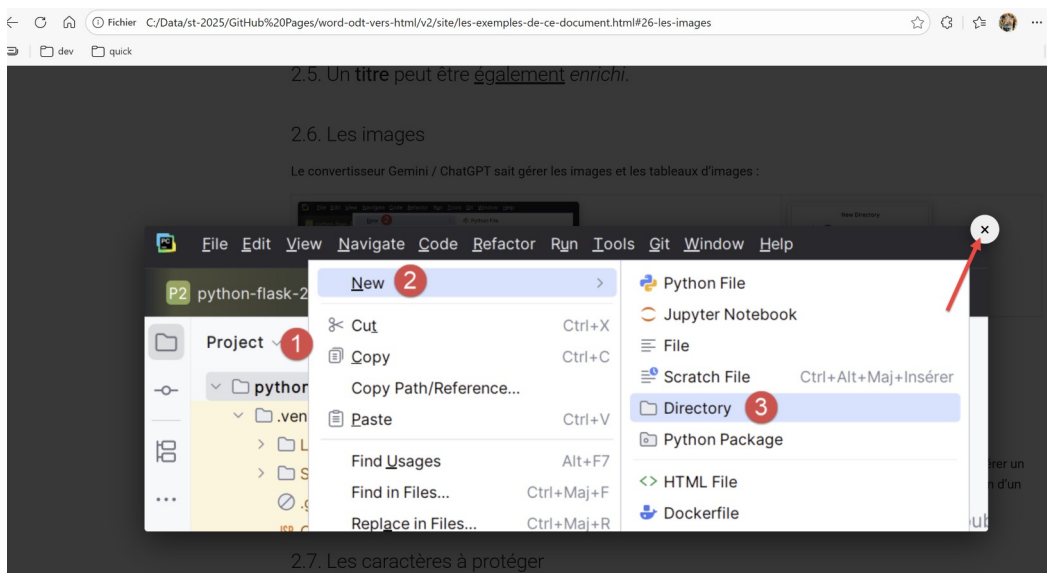
2.6. Las imágenes

El conversor Gemini / ChatGPT sabe gestionar imágenes y tablas de imágenes:



En los documentos ODT es frecuente encontrar dibujos. A pesar de decenas de intentos, Gemini no ha logrado generar un script que genere la imagen (como una captura de pantalla) del dibujo. Así, arriba, la imagen 5 es la captura de pantalla de un dibujo de un documento ODT.

Todas las imágenes son clicables para ampliarlas. Si arriba hacemos clic en la imagen [1-3], obtenemos la siguiente ampliación:








2.7. Los caracteres que hay que proteger

Un sitio web de MkDocs¹ tiene páginas cuyo contenido no es HTML, sino Markdown. Si el documento ODT/DOCX contiene caracteres que existen en Markdown, es posible que MkDocs los interprete y, por lo tanto, no se muestren como se esperaba. He aquí dos ejemplos:

El asterisco * tiene un significado en MarkDown. Por lo tanto, la siguiente línea puede interpretarse incorrectamente:

El impuesto I es entonces igual a **0,15*R – 2072,5*nbParts**.

Otro ejemplo es cuando quieres insertar un bloque de código Markdown en tu documento como este:

```
1. # Convertidor de Word/ODT a sitio HTML (MkDocs)
2.
3.  [Ver el sitio de demostración generado](https://stahe.github.io/word-odt-vers-html-janv-2026/)
4.
5. ---
6.
7. ##  Descripción
8.
9. El objetivo de este proyecto es poner a disposición del lector un conversor en Python de documentos Word u ODT a un sitio web HTML estático.
10.
11. Cuando el documento ODT/DOCX es adecuado, el conversor genera un sitio HTML mediante MkDocs que tiene el aspecto profesional de los sitios generados por Pandoc.
12.
13. ##  Contexto de creación
14.
15. Este conversor ha sido creado íntegramente por la IA Gemini 3 (con una suscripción pro). Es el resultado de sucesivas iteraciones para gestionar con precisión la estructura de los documentos ODT (OpenDocument Text).
16.
17. ##  Funcionalidades
18.
19. El script `convert.py` realiza las siguientes acciones:
20.
21. * Conversión de ODT a Markdown: Analiza el archivo `.odt` (XML) para extraer su estructura.
22. * Gestión de títulos: Genera automáticamente la tabla de contenidos (TOC) y la navegación lateral.
23. * Bloques de código: Detección automática de lenguajes, coloración sintáctica y gestión precisa de la numeración de líneas (atributos `start-value`).
24. * Listas: Compatibilidad con listas con viñetas y numeradas con la sangría correcta.
25. * Formato: Compatibilidad con negrita, cursiva, subrayado y resaltado (respetando los colores originales).
26. * Imágenes: Extracción e integración automática de las imágenes contenidas en el documento.
27. * Configuración: Personalización mediante un archivo `config.py` (pie de página, Google Analytics, etc.).
28.
29. ##  Instalación
30.
31. ### Requisitos previos
32.
33. * Python 3.x
34. * Las siguientes bibliotecas:
35.
36. ```bash
37. pip install odfpy unidecode mkdocs mkdocs-material
38.
39. ```
```

¹[MkDocs](https://www.mkdocs.org/)

2.8. Las tablas

Una tabla puede contener diferentes contenidos:

1

2



```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import
   org.springframework.context.ApplicationContext;
7. import
   org.springframework.context.support.ClassPathXmlApp
   licationContext;
8.
9. clase pública Demo01 {
10.
11.
```

3

4



```
1. # -----
2. def mostrar(cadena):
3.     # muestra cadena
4.     print("cadena=%s" % cadena)
5.
6.
7. # -----
8. def mostrar_tipo(variable):
9.     # muestra el tipo de variable
10.    print("tipo[%s]=%s" % (variable,
11.                           tipo(variable)))
12.
```

Una tabla que contiene enlaces:

El conversor Gemini / ChatGPT sabe conservar los enlaces externos del documento ODT / DOCX. ejemplo [Gemini 3](#) o [\[Generar un script de Python con herramientas de IA\]](#).

Una referencia a un capítulo:2.1.1 .

Una referencia a un marcador de texto anterior: Gemini 3

Una referencia a un marcador de texto posterior: GitHub

Sabe gestionar un enlace a un capítulo [Enlace a un capítulo](#)

Una tabla cuyas celdas contienen otra tabla:

`html:form`

sirve tanto para generar la etiqueta HTML `<form>` como para proporcionar información al controlador que tendrá que procesar este formulario:

<code>action</code>	url a la que se enviarán los valores del formulario
<code>name</code>	nombre del formulario HTML. También es el nombre del bean que almacenará los valores del mismo
<code>type</code>	nombre de la clase que deberá instanciarse para obtener el bean de almacenamiento del formulario

Cabe señalar que no se especifica el método de envío de los parámetros del formulario (GET/POST) al controlador. Se podría hacer con el atributo `method`. En ausencia de este, se utiliza el método **POST** por defecto.

`html:text` sirve para generar la etiqueta `<input type="text" value="...">`:

`property` nombre del campo del bean del formulario que se asociará al campo de entrada. Al enviar el formulario al servidor (cliente -> servidor), el campo del bean tomará el valor del campo de entrada. Al mostrar el formulario (servidor -> cliente), el valor contenido en el campo del bean se muestra en el campo de entrada de .

También gestiona el fondo del párrafo en una tabla

- También gestiona el fondo del párrafo en una tabla
- 1. También gestiona el fondo del párrafo en una tabla

2.9. Las notas al pie

El convertidor Gemini² / ChatGPT gestiona las notas al pie de página. Aquí hay otra nota al pie de página³.

3. Lo que hay en Internet

Antes de empezar, podemos preguntarle a la IA qué hay en el mercado para convertir un documento Word u ODT en una página web HTML estática y que sea gratuito.

Hagámosle la pregunta a Gemini:

Existe-t-il sur internet un logiciel gratuit capable de convertir un document Word en site statique HTML ?

Esta es su respuesta:

Sí, existen varias soluciones gratuitas, pero la elección depende de tu definición de «sitio web estático» (una simple página HTML frente a un sitio web completo con navegación) y de tus habilidades técnicas.

Estas son las mejores opciones clasificadas por método:

- **1. La solución «todo en uno» (para crear un sitio web navegable)**

Si quieres convertir un documento largo de Word (con capítulos y subcapítulos) en un sitio web completo con un menú de navegación a la izquierda, esta es la mejor opción.

- **Software: HelpNDoc** (versión gratuita para uso personal)
- **Cómo funciona:** es un software de creación de documentación. Importas tu archivo de Word, analiza la estructura (Títulos 1, Títulos 2...) y genera automáticamente un sitio HTML5 estático y «responsive» (adaptado a dispositivos móviles).
- **Ventajas:** Crea un sitio web completo con menú, barra de búsqueda e índice sin escribir una sola línea de código.
- **Inconveniente:** La versión gratuita añade un pequeño banner «Generado por HelpNDoc» al pie de página.

²[Google Gemini](#)

³La nota al pie

- **2. La solución «Limpiador en línea» (para convertir página por página)**

Si tienes un documento corto y solo quieres obtener el código HTML para insertarlo en una página ya existente, los convertidores en línea son ideales para limpiar el código «sucio» generado por Word.

- **Herramienta: WordToHTML.net** (o Word2CleanHTML)

- **Cómo funciona:**

1. Copia tu texto en Word.
2. Lo pegas en el editor visual de la izquierda de la página web.
3. El sitio web genera un código HTML limpio (sin las etiquetas innecesarias de Microsoft) en la ventana de la derecha.

- **Ventajas:** Muy sencillo, visual, elimina lo «superfluo» del código de Microsoft.

- **Inconveniente:** ideal para una sola página, no para generar la arquitectura de un sitio web completo.

- **3. La solución «Pro / Desarrollador» (la mejor calidad)**

Es el método recomendado por los expertos para obtener un código ultra limpio y duradero.

- **Software: Pandoc** (línea de comandos)

- **Cómo funciona:** Es un pequeño programa sin interfaz gráfica (que se utiliza a través del terminal o la línea de comandos). Convierte los formatos con gran precisión.

- **Comando típico:** `pandoc mi_documento.docx -t html -o mi_sitio.html`

- **Ventajas:** Código HTML5 perfecto, gratuito, de código abierto, gestiona muy bien las imágenes y las fórmulas matemáticas (LaTeX).

- **Inconvenientes:** Requiere estar familiarizado con la línea de comandos (pantalla negra).

- **4. La solución «improvisada» (Microsoft Word)**

Puedes hacerlo directamente desde Word, pero ten cuidado con el resultado.

- **Método:** Archivo > Guardar como > Selecciona «Página web filtrada (*.htm; *.html)».

- **Importante:** No elijas «Página web de un solo archivo (.mht)», elige bien «Filtrada». La opción «Filtrada» reduce (un poco) el código superfluo que Word suele añadir.

- **Contras:** El código suele seguir siendo pesado y difícil de modificar posteriormente.

- **Tabla comparativa rápida**

Solución	Tipo de resultado	Calidad del código	Facilidad	Ideal para...
HelpNDoc	Sitio web completo (con menú)	Buena	Regular	Manuales, tesis, documentación
WordToHTML	Página única	Excelente	Muy fácil	Artículos de blog, páginas sencillas
Pandoc	Página única (sin formato)	Perfecto	Difícil	Desarrolladores, archivado propio

Solución	Tipo de resultado	Calidad del código	Facilidad	Ideal para...
Word (nativo)	Página única	Mala	Fácil	Solución rápida

- **Mi consejo**

Si tu objetivo es publicar este documento en línea de forma profesional, te recomiendo **HelpNDoc** si eres principiante y quieres una estructura de sitio web inmediata.

Si simplemente quieres el contenido limpio para pegarlo en otro sitio (WordPress, etc.), utiliza **WordToHTML.net**.

¿Quieres que te explique paso a paso el procedimiento para una de estas herramientas (por ejemplo, WordToHTML)?

He utilizado [Pandoc] en alguna ocasión, pero me pareció difícil de usar. Recuerdo que había que «preparar» el documento de Word para que Pandoc lo gestionara correctamente. Sin embargo, el resultado era excelente. **HelpNDoc** parece un software atractivo, ya que genera un sitio web con menú. Lo probé con uno de mis documentos de Word y el resultado no fue satisfactorio.

Ahora veamos si existe algún software de conversión para documentos ODT:

Existe-t-il sur internet un logiciel gratuit capable de transformer un document ODT en site HTML statique ?

La respuesta de Gemini es la siguiente:

Sí, por supuesto. Al igual que con Word, el formato **ODT** (utilizado por LibreOffice y OpenOffice) cuenta con sus propias soluciones.

Estas son las mejores opciones gratuitas específicas para archivos ODT, clasificadas por método:

- **1. La solución «Extensión» (la mejor calidad integrada)**

Si utilizas LibreOffice u OpenOffice, la exportación HTML básica es (al igual que en Word) bastante mediocre. Sin embargo, existe una extensión gratuita estupenda diseñada específicamente para ello.

- **Herramienta: Writer2xhtml** (extensión para LibreOffice)

- **Cómo funciona:**

1. Descargas e instalas la extensión en LibreOffice.
2. Ve a Archivo > Exportar y aparecerá una nueva opción «XHTML 1.1 + MathML» (o similar).

- **Ventajas:** El código es mucho más limpio que el de la exportación estándar, y gestiona muy bien las fórmulas matemáticas y el formato complejo.

- **Contras:** Es necesario instalar una extensión (una única intervención técnica).

- **2. La solución «Pro / Línea de comandos» (universal)**

Es la misma herramienta que para Word, ya que también funciona muy bien con el formato ODT.

- **Software: Pandoc**

- **Cómo funciona:** A través del terminal de comandos.

- **Comando:** `pandoc mi_archivo.odt -t html -o mi_sitio.html`

- **Ventajas:** Es sin duda el conversor más potente del mercado para el formato ODT a HTML. «Entiende» la estructura del documento ODT mejor que nadie.
- **3. La solución «Sitio completo con navegación» (a través de HelpNDoc)**

HelpNDoc (mencionado anteriormente) sigue siendo la mejor opción gratuita para crear un sitio web auténtico con menú a la izquierda.

Sin embargo, HelpNDoc no siempre importa de forma nativa el formato .odt tan bien como el .docx.

- **El truco:**
 1. Abre tu archivo en LibreOffice.
 2. Seleccione «Guardar como...» y elija «**Word 2007-365 (.docx)**».
 3. Importa este archivo .docx a HelpNDoc para generar tu sitio web.
- **¿Por qué?** Esto garantiza que la jerarquía de tus capítulos (Títulos 1, Títulos 2) se reconozca perfectamente para crear el menú del sitio.
- **4. La solución en línea (limpieza rápida)**

Algunas herramientas en línea aceptan directamente el formato ODT para generar HTML limpio.

- **Herramienta: WordToHTML.net** (también admite ODT)
- **Cómo funciona:** Puedes subir tu archivo ODT directamente o pegar tu texto.
- **Ventajas:** Muy práctico para convertir un único artículo sin necesidad de instalar nada.
- **Resumen de mi consejo para ODT:**
 1. **Para un sitio web completo (documentación/tesis):** Convierte tu ODT a **DOCX** mediante LibreOffice y, a continuación, utiliza **HelpNDoc** (es el método más fiable para obtener un menú de navegación).
 2. **Para una página sencilla y limpia:** utiliza la extensión **Writer2xhtml** si lo haces a menudo, o **Pandoc** si quieres automatizar la tarea.

¿Quieres que te explique cómo instalar la extensión **Writer2xhtml** en LibreOffice?

He probado la extensión de LibreOffice [Writer2xhtml] en uno de mis documentos. El resultado es legible, pero no lo suficientemente profesional. No hay índice a la izquierda. Los bloques de código pierden el resaltado sintáctico.

Al final, creo que el conversor generado por Gemini es una alternativa a lo que se propone arriba. El resultado es comparable al producido por Pandoc si el documento ODT es adecuado para el conversor.

4. La indicación inicial para Gemini 3

Se supone aquí que el lector tiene un conocimiento básico de la IA [Gemini](#). Si no es así, se le recomienda leer el documento [\[ref1\]](#). La indicación inicial dada a Gemini se proporcionó en forma del siguiente archivo de texto [Prompt User.txt]:

1. Solicitudes del usuario
- 2.
3. - Crea un script de Python [convert] para crear un sitio MkDocs con el tema «material» a partir de un documento ODT. El nombre del documento ODT será el único argumento de convert.
- 4.
5. - A continuación, crea un script de Python [build] que utilizará el archivo [mkdocs.yml] creado por convert para crear un sitio HTML estático
6. - Al final, el script [build] mostrará la página de inicio del sitio en un navegador.

- 7.
8. - El sitio tendrá dos columnas independientes. La de la izquierda contendrá la tabla de contenidos, la de la derecha el contenido del documento ODT.
- 9.
10. - El índice incluirá todos los títulos del documento ODT (Título 1, Título 2, Título 3, etc.), su numeración y su jerarquía
- 11.
12. - Las imágenes se encuentran, en su mayoría, en tablas y han sido redimensionadas.
13. Explorarás las tablas en busca de estas imágenes y respetarás su redimensionamiento.
- 14.
15. - Hay bloques de código en el documento ODT. Se pueden identificar por sus estilos [Código fuente numerado resultados, Código fuente].
16. Los demás atributos del párrafo de código deben ignorarse. Por ejemplo, si el párrafo de código es también una lista numerada, debes ignorar este atributo.
17. Todos estos bloques de código deben tratarse como código en el sitio MkDocs. Las líneas de código deben estar numeradas en el resultado HTML.
- 18.
19. - Para todos los bloques de código, utilizarás el resaltado sintáctico de Python.
- 20.
21. - No debes modificar las líneas de código. En particular, no debes eliminar las sangrías ni los espacios que preceden al primer carácter de la línea de código.
- 22.
23. - Debes localizar todas las listas con viñetas. ODT tiene varias formas de crearlas. Debes explorarlas todas. Las listas con viñetas
24. a veces anidadas unas dentro de otras. Debes respetar esta anidación.
- 25.
26. - El título del sitio será «Generar un script de Python con herramientas de IA»
- 27.
28. - Debes incluir cada imagen extraída del documento ODT, así como cada capítulo MkDocs creado
- 29.
30. - Debes respetar los enlaces que encuentres en el documento ODT. Deben seguir siendo enlaces en el documento HTML generado.
- 31.
32. - Debes respetar los textos en negrita, subrayados o en cursiva. Deben reproducirse tal cual en el documento HTML
- 33.

- Línea 19: la prueba inicial se realizó en un curso publicado anteriormente que contenía código Python;
- este comando permitió iniciar el chat. Tras varias decenas de iteraciones, quedó obsoleto;

5. Crear un entorno de trabajo Python

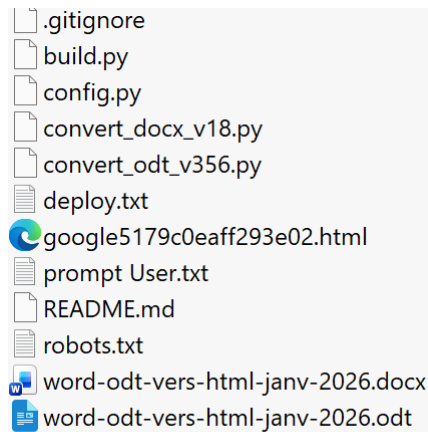
Gemini y ChatGPT generarán dos scripts de Python. Por lo tanto, necesitamos un entorno de Python:

- La instalación de Python se describe en [el apartado 3.1](#) de [\[ref1\]](#);
- La instalación de PyCharm se describe en [el apartado 3.1](#) de [\[ref1\]](#). [\[ref1\]](#) presenta una configuración detallada de PyCharm. En este caso, PyCharm solo se utilizará para ejecutar los scripts de Python generados por Gemini y ChatGPT. Por lo tanto, no es necesaria ninguna configuración de PyCharm. PyCharm también se utilizará para presentar los archivos necesarios para el conversor;

6. La carpeta de trabajo del conversor

Puede descargar la carpeta de trabajo [| AQUÍ |](#).

La carpeta de trabajo es la siguiente:






.gitignore
build.py
config.py
convert_docx_v18.py
convert_odt_v356.py
deploy.txt
google5179c0eaff293e02.html
prompt User.txt
README.md
robots.txt
word-odt-vers-html-janv-2026.docx
word-odt-vers-html-janv-2026.odt

- Por ahora, vamos a ignorar los siguientes archivos [.gitignore, deploy.txt, google*.html, README.md, robots.txt, deploy.txt]. Son archivos que nos servirán para la implementación en GitHub del sitio HTML generado localmente;
- [prompt User.txt] es el prompt con el que comencé las iteraciones con Gemini;
- El archivo [word-odt-vers-html-janv-2026.odt] es el documento ODT de este artículo. Se convierte a HTML mediante el script [convert_odt_v356.py];
- El archivo [word-odt-vers-html-janv-2026.docx] es el documento DOCX de este mismo artículo. Se convierte a HTML mediante el script [convert_docx_v18.py];
- Los scripts [convert*.py] y [build.py] son los dos scripts de Python generados por la IA;
 - [convert*] convierte un documento ODT o DOCX en un sitio [MkDocs]. Un sitio MkDocs es un sitio estático cuyas páginas están escritas según la sintaxis [Markdown]. [MkDocs] proporciona un servidor capaz de mostrar los sitios MkDocs;
 - [build] convierte el sitio MkDocs en un sitio HTML estático estándar. Al final de esta conversión, abre un navegador para mostrar la raíz del sitio;

En ningún momento veremos el código Python generado. Consideraremos los dos scripts como dos cajas negras. En ningún momento he modificado el código a mano. Siempre le he pedido a Gemini / ChatGPT que corrija él mismo su script. Por eso no es necesario saber Python para utilizar el conversor.

- [config.py] es un archivo de configuración para los scripts [convert*]. Al principio, [convert*] no tenía ningún archivo de configuración. Luego, al cambiar los documentos que quería convertir, me di cuenta poco a poco de que había elementos que le pedía a la IA que generara (por ejemplo, el nombre del sitio, que cambia para cada documento) que estarían mejor ubicados en un archivo de configuración que el usuario pudiera modificar por sí mismo. También lo construí mediante iteraciones con la IA;

7. Los archivos de configuración de los convertidores

- 1  config.py
- 2  config_code.py
- 3  config_colors.py
- 4  config_presentation.py
- 5  config_styles.py
- 6  config_texts.py
- 7  config_urls.py

El archivo de configuración [config.py] recopila la información que puede variar de un documento a otro. Utiliza otros seis archivos de configuración. En estos últimos se han reunido los parámetros más susceptibles de cambiar al pasar de un documento a otro para la conversión.

El archivo [config.py] importa el contenido de los otros 6 archivos de configuración:

```
1. from config_urls import URLS
2. from config_texts import TEXTS
3. from config_code import CODE
4. from config_styles import STYLES
5. from config_colors import COLORS
6. from config_presentation import PRESENTATION
```

config_urls

Este archivo recopila las tres URL de la configuración [config.py]. Estas cambian con cada documento.

```
1. URLS = {
2.     "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
3.     "repo_url": "https://stahe.github.io/word-odt-vers-html-janv-2026",
4.     "author_site": "https://tahe.developpepez.com",
5. }
```

- línea 2: la URL del sitio web donde vas a implementar el sitio HTML generado por el conversor;
- línea 3: la URL de tu repositorio Git (véase el apartado12);
- línea 4: el sitio web del creador del sitio web. Esta URL se coloca en la parte inferior de las páginas del sitio web. Puede estar vacía.

config_texts

Este archivo recopila los textos de [config.py] que pueden cambiar al cambiar el idioma del sitio HTML generado.

```
1. TEXTS = {
2.     "home_label": "Inicio",
3.     "site_name": "Convertir un documento Word u ODT a un sitio HTML estático compatible con MkDocs con las IA Gemini 3 y ChatGPT 5.2",
4.     "site_description": "Convertir un documento Word u ODT a un sitio HTML estático compatible con
```

```

MkDocs y las IA Gemini 3 y ChatGPT 5.2",
5. "toggle_to_dark": "Cambiar al modo oscuro",
6. "toggle_to_light": "Cambiar a modo claro",
7. "footer_license_sentence": (
8. "Este curso-tutorial escrito por <strong>Serge Tahé</strong> se pone a disposición del público
según los términos de la\n"
9. "    <em>Licencia Creative Commons Reconocimiento - Sin uso comercial -\n"
10. "        Compartir en las mismas condiciones 3.0 no adaptada</em>.\n"
11. ),
12. "copy_label": "Copiar",
13. "copy_copied_label": "Copiado",
14. }

```

- línea 2: el nombre del primer enlace en la tabla de contenidos. Su contenido recoge todo lo que hay en el documento antes del primer título de nivel 1 (Título 1). Es el inicio del documento;
- línea 3: el título del documento. Se muestra en la barra superior del sitio web generado;
- línea 4: también el título del documento, esta vez destinado a los motores de búsqueda;
- líneas 5-6: las burbujas que aparecen al pasar el ratón por el icono de la barra superior del sitio para cambiar del modo claro al oscuro o viceversa;
- líneas 7-11: la licencia del sitio. Se muestra en el pie de página del sitio generado;
- líneas 12-13: el código que se muestra en el sitio, acompañado de un botón que permite copiar el código al portapapeles. La línea 12 es el texto del botón antes de copiar, la línea 13 el texto una vez que se ha copiado el código;

Así es como se ve este archivo para una versión en inglés del documento:

```

1. TEXTS = {
2.   "home_label": "Welcome",
3.   "site_name": "Convert a Word or ODT document into a static HTML website using Gemini 3 and
ChatGPT 5.2",
4.   "site_description": "Convertir un documento de Word u ODT en un sitio web HTML estático
utilizando Gemini 3 y ChatGPT 5.2 AI",
5.   "toggle_to_dark": "Cambiar al modo oscuro",
6.   "toggle_to_light": "Cambiar al modo claro",
7.   "footer_license_sentence": (
8.     'Este tutorial, escrito por <strong>Serge Tahé</strong>, se pone a disposición del
público bajo los términos de la\n'
9.     '<em>Creative Commons Attribution - Non-Commercial -\n'
10.    'ShareAlike 3.0 Unported</em>.'
11.   ),
12.   "copy_label": "Copiar",
13.   "copy_copied_label": "Copiado",
14. }

```

config_styles

Este archivo sirve para asignar el nombre del estilo del título del documento. En un documento DOCX, suele ser «Título», pero en un documento ODT varía según el documento.

```

1. STYLES = {
2.   "style_names": [
3.     "p1"
4.   ]
5. }

```

- línea 2: [style_names] es una matriz. En ella se pueden introducir los nombres de los estilos que puede tener el título del documento;
- línea 3: el documento [word-odt-vers-html-janv-2026.odt] tiene un título con el estilo P1;

config_code

Los bloques de código en el documento DOCX u ODT pueden encontrarse en dos formas:

- código enriquecido (colores, negrita, cursiva, etc.). En este caso, se copia tal cual en el sitio HTML;
- código sin formato (texto plano). En este caso, se dan indicaciones a los convertidores para que identifiquen por sí mismos el lenguaje utilizado en el bloque de código. Para ello, se les proporcionan palabras clave:

```

1. CODE = {
2.   # Reglas de detección automática de lenguaje basadas en el contenido
3.   "detection_rules": {
4.     "csharp": [
5.       "using", "Console.WriteLine", "public static void Main", "WebMethod",
6.       "TryParse", "EventArgs", "String.Format", "System.Web.Services"
7.     ],
8.     "java": [
9.       "System.out.println", "public static void main(String", "package",
10.      "JUnitTest", "Class.forName", "PreparedStatement", "private static void",
11.      "private void", "getAgendaMedecinJour", "@PostConstruct", "@ResponseBody",
12.      "@RequestMapping", "getMedecin", "@Entity", "@Autowired", "@Bean",
13.      "Serializable", "getClient", "getCreneau", "getRv", "PostAjouterRv",
14.      "PostSupprimerRv", "@EnableJpaRepositories", "@Component", "getAgendaMedecinJour",
15.      "getResponse", "getMessagesForException", "getBase64", "ajouterRv",
16.      "Respuesta", "getPartialViewAgenda", "setModelforAgenda", "ActionContext",
17.      "getActionContext", "PostLang", "PostUser", "PostGetAgenda", "@NotNull",
18.      "@EnableAutoConfiguration", "HttpSecurity"
19.    ],
20.     "html": [
21.       "<html>", "</div>", "<body>", "<script>", "href=", "<span>", "<p>",
22.       "<h2>", "<form", "<table", "<input"
23.     ],
24.     "sql": [
25.       "SELECT", "INSERT INTO", "UPDATE", "DELETE FROM", "WHERE",
26.       "CREATE TABLE", "ALTER TABLE"
27.     ],
28.     "python": [
29.       "def", "import", "print(", "from"
30.     ],
31.     "xml": [
32.       "<?xml", "<project", "<version>", "<configuration>", "<build>",
33.       "<dependency>", "<properties>", "<configuration>", "<start-class>"
34.     ],
35.     "javascript": [
36.       "use strict", "console.log", "let", "constructor", "async", "export"
37.     ],
38.     "php": [
39.       "<?php", "declare", "require"
40.     ],
41.     "vbscript": [
42.       "Option", "Dim", "Explicit"
43.     ],
44.     "markdown": [
45.       "# ", "## ", "### ", "***", "___", "![", "](" # Marcadores típicos de Markdown
46.     ],
47.   },
48. }

```

Una vez más, estas reglas de detección solo se aplican a los bloques de código en «texto sin formato» y no a los bloques de código enriquecido. Si descubres en tu sitio web que algunos bloques de código no se han resaltado, aquí es donde debes acudir.

Para cada lenguaje se indican las cadenas de caracteres que permiten identificarlo. Algunas palabras clave pueden aparecer en varios lenguajes. El conversor selecciona el lenguaje para el que ha encontrado más palabras clave. El principio es sencillo: revisa tu código, selecciona palabras clave características y las introduces en estas líneas para el lenguaje adecuado. Ten en cuenta que no puedes poner cualquier cosa como nombre de lenguaje: debes utilizar los nombres reconocidos por MkDocs;

config presentation

Este archivo define el aspecto de los diferentes elementos del sitio web generado. No es algo que se suele cambiar a menudo:

```
1. PRESENTATION = {
2.   # -----
3.   # Título del documento
4.   # -----
5.   "document_title": {
6.     "tamaño_fuente": "28px",
7.     "peso_de_la_fuente": "negrita",
8.     "margen_inferior": "1em",
9.     "line_height": "1.2"
10.  },
11.
12.  # -----
13.  # Bloques de código enriquecidos
14.  # -----
15.  "code": {
16.    "rich_line_height": "12px",
17.    "rich_font_family": "Consolas, 'Courier New', monospace",
18.    "rich_font_size": "15px"
19.  },
20.
21.  # -----
22.  # Botón Copiar
23.  # -----
24.  "copy_button": {
25.    "container": "position: relative;",
26.    "btn": (
27.      "position: absolute; top: .5rem; right: .5rem; "
28.      "display: inline-flex; align-items: center; justify-content: center; "
29.      "gap: 0.35rem; "
30.      "padding: 0.25rem 0.6rem; "
31.      "font-size: .72rem; font-weight: 600; letter-spacing: .01em; "
32.      "line-height: 1.2; "
33.      "border-radius: 999px; "
34.      "sombra de cuadro: 0 1px 2px rgba(0,0,0,.10); "
35.      "backdrop-filter: saturate(180%) blur(6px); "
36.      "cursor: pointer; user-select: none; "
37.      "transition: transform .08s ease, box-shadow .12s ease, background .12s ease; "
38.      "z-index: 5;"
39.    ),
40.    "btn_hover": (
41.      "sombra de cuadro: 0 3px 10px rgba(0,0,0,.18); "
42.      "transform: translateY(-1px);"
43.    ),
44.    "btn_copied": "opacity: .85;"
45.  },
46.
47.  # -----
48.  # Imágenes
49.  # -----
50.  "imágenes": {
51.    "sombra": {
52.      "enabled": True,
53.      "border_radius": "8px"
54.    }
55.  },
56.
57.  # -----
58.  # Marco / borde del encabezado
59.  # -----
60.  "frame": {
61.    "ancho_borde_superior_encabezado": "4px"
62.  }
63. }
```

Si lo deseas, puedes modificar el aspecto:

- líneas 5-10: del título del documento;
- líneas 15-19: de las líneas de código enriquecido;
- líneas 24-45: del botón [Copiar], que permite guardar el código en el portapapeles;
- líneas 50-55: de las sombras de las imágenes;
- líneas 60-62: del marco que bordea la barra superior de las páginas del sitio;

config_colors

En este archivo se controlan los colores del sitio web, principalmente el color de fondo de la barra superior de las páginas:

```

1.  COLORS = {
2.      "theme": {
3.          "paleta": {
4.              "light": {
5.                  "media": "(prefers-color-scheme: light)",
6.                  "esquema": "predeterminado",
7.                  "primario": "verde azulado",
8.                  "accent": "purple"
9.              },
10.             "dark": {
11.                 "media": "(prefers-color-scheme: dark)",
12.                 "esquema": "pizarra",
13.                 "primario": "verde azulado",
14.                 "accent": "purple"
15.             }
16.         }
17.     },
18.     "frame": {
19.         # "header_bg_light": "#1E88E5", (azul)
20.         "header_bg_light": "",
21.         "header_bg_dark": "",
22.         "header_top_border_light": "",
23.         "header_top_border_dark": ""
24.     },
25.     "document_title": {
26.         "color": "#2c3e50"
27.     },
28.     "copy_button": {
29.         "border": "rgba(0,150,136,.45)",
30.         "background": "rgba(0,150,136,.12)",
31.         "text": "rgb(0,150,136)",
32.         "background_hover": "rgba(0,150,136,.20)"
33.     },
34.     "images": {
35.         "sombra": {
36.             "zoomable": "0 8px 24px rgba(0,0,0,.28), 0 18px 60px rgba(0,0,0,.20)",
37.             "zoomable_hover": "0 12px 30px rgba(0,0,0,.32), 0 26px 80px rgba(0,0,0,.22)",
38.             "lightbox": "0 24px 80px rgba(0,0,0,.65)"
39.         }
40.     }
41. }

```

Aquí hay que tener algunos conocimientos de CSS, cosa que no es mi caso. Solo he utilizado los colores de las líneas 7 y 13, que definen el color de la barra superior.

config_files_to_copy

Este archivo enumera los archivos que deben copiarse en la raíz del sitio web generado:

```

1.  FILES_TO_COPY =[
2.      "google5179c0eaff293e02.html",

```

```

3.     "robots.txt",
4.     "word-odt-a-html-ene-2026.pdf",
5.     "word-odt-vers-html-janv-2026.zip"
6. ]

```

- líneas 2-3: son necesarias para el seguimiento de Google del sitio generado (véase el apartado 13);
- líneas 4-5: los archivos que desea copiar en la raíz de su sitio web para ponerlos a disposición de sus visitantes;

config

El archivo [config.py] es el archivo de configuración utilizado por los dos convertidores. Reúne toda la información que necesitan. Es un archivo que, en principio, nunca debería modificarse. Es complejo y, por eso, se ha decidido extraer los parámetros de configuración susceptibles de cambiar y colocarlos en archivos externos.

```

1. from config_urls import URLS
2. from config_texts import TEXTS
3. from config_code import CODE
4. from config_styles import STYLES
5. from config_colors import COLORS
6. from config_presentation import PRESENTATION
7. from config_files_to_copy import FILES_TO_COPY
8.
9. config = {
10.     "toc": {
11.         "home_label": TEXTS["home_label"]
12.     },
13.
14.     "mkdocs": {
15.         "site_name": TEXTS["site_name"],
16.         "site_url": URLS["site_url"],
17.         "site_description": TEXTS["site_description"],
18.         "site_author": "Serge Tahé",
19.         "repo_url": URLS["repo_url"],
20.         "repo_name": "GitHub",
21.         "use_directory_urls": False,
22.
23.         "theme": {
24.             "name": "material",
25.             "custom_dir": "overrides",
26.             "features": [
27.                 "navigation.sections",
28.                 "navegación.indices",
29.                 "navigation.expand",
30.                 "toc.integrate",
31.                 "navegación.top"
32.             ],
33.             "paleta": [
34.                 {
35.                     "media": COLORS["theme"]["palette"]["light"]["media"],
36.                     "esquema": COLORS["tema"]["paleta"]["claro"]["esquema"],
37.                     "primary": COLORS["theme"]["palette"]["light"]["primary"],
38.                     "accent": COLORS["theme"]["palette"]["light"]["accent"],
39.                     "toggle": {
40.                         "icon": "material/brightness-7",
41.                         "name": TEXTS["toggle_to_dark"]
42.                     }
43.                 },
44.                 {
45.                     "media": COLORS["theme"]["palette"]["dark"]["media"],
46.                     "scheme": COLORS["theme"]["palette"]["dark"]["scheme"],
47.                     "primary": COLORS["theme"]["palette"]["dark"]["primary"],
48.                     "accent": COLORS["theme"]["palette"]["dark"]["accent"],
49.                     "toggle": {
50.                         "icon": "material/brightness-4",
51.                         "name": TEXTS["toggle_to_light"]
52.                     }
53.                 }
54.             ]
55.         }
56.     }
57. }

```

```

53.         }
54.     ]
55. },
56.
57.     "markdown_extensions": [
58.         "admonition",
59.         "attr_list",
60.         "pymdownx.superfences",
61.         "pymdownx.mark",
62.         {
63.             "pymdownx.highlight": {
64.                 "anchor_linenums": True,
65.                 "linenums": None
66.             }
67.         },
68.         "md_in_html",
69.         "footnotes"
70.     ],
71.
72.     "extra_javascript": [
73.         "javascripts/focus.js"
74.     ],
75.     "extra_css": [
76.         "stylesheets/focus.css"
77.     ]
78. },
79.
80.     "footer": (
81.         "{% block footer %}\n"
82.         " <div class=\"md-footer-meta md-typeset\">\n"
83.         "   <div class=\"md-footer-meta__inner\">\n\n"
84.         "     <div>\n"
85.         f"       <a href=\"{URLS['author_site']}\" target=\"_blank\">\n"
86.         f"         {URLS['author_site']}\n"
87.         "     </a>\n"
88.         "     <br>\n"
89.         f"     {TEXTS['footer_license_sentence']}\n"
90.         "   </div>\n\n"
91.         " </div>\n"
92.         " </div>\n"
93.         "{% endblock %}"
94.     ),
95.
96.     "extra": {
97.         "analytics": {
98.             "provider": "google",
99.             "property": "G-XXXXXXX"
100.        }
101.    },
102.
103.     "document_title": {
104.         "style_names": STYLES["style_names"],
105.         "css": (
106.             f"font-size: {PRESENTATION['document_title']['font_size']}; "
107.             f"font-weight: {PRESENTATION['document_title']['font_weight']}; "
108.             f"margin-bottom: {PRESENTATION['document_title']['margin_bottom']}; "
109.             f"line-height: {PRESENTATION['document_title']['line_height']}; "
110.             f"color: {COLORS['document_title']['color']};"
111.         )
112.     },
113.
114.     "code": {
115.         "case_insensitive_languages": ["sql", "html", "vbnet", "vbscript"],
116.         "palabras_clave_de_estilo": ["código"],
117.         "default_language": "text",
118.         "rich_line_height": PRESENTATION["code"]["rich_line_height"],
119.         "rich_font_family": PRESENTATION["code"]["rich_font_family"],
120.         "rich_font_size": PRESENTACIÓN["code"]["rich_font_size"],
121.         "detection_rules": CODE["detection_rules"],
122.         "copy_button": True,

```

```

123.     "copy_label": TEXTOS["copy_label"],
124.     "copy_copied_label": TEXTS["copy_copied_label"],
125.     "copy_only_recognized_language": True,
126.     "copy_min_lines": 4,
127.     "copy_allow_pygments_heuristic": True,
128.     "copy_style": {
129.         "container": PRESENTATION["copy_button"]["container"],
130.         "btn": (
131.             PRESENTATION["copy_button"]["btn"]
132.             + f"border:1px solid {COLORS['copy_button']['border']}); "
133.             + f"background:{COLORS['copy_button']['background']}); "
134.             + f"color:{COLORS['copy_button']['text']}); "
135.         ),
136.         "btn_hover": (
137.             f"background:{COLORS['copy_button']['background_hover']}); "
138.             + PRESENTATION["copy_button"]["btn_hover"]
139.         ),
140.         "btn_copied": PRESENTATION["copy_button"]["btn_copied"]
141.     }
142. },
143.
144.     "imágenes": {
145.         "shadow": {
146.             "enabled": PRESENTACIÓN["imágenes"]["sombra"]["enabled"],
147.             "border_radius": PRESENTACIÓN["images"]["shadow"]["border_radius"],
148.             "zoomable": COLORS["images"]["shadow"]["zoomable"],
149.             "zoomable_hover": COLORS["images"]["shadow"]["zoomable_hover"],
150.             "lightbox": COLORS["images"]["shadow"]["lightbox"],
151.         }
152.     },
153.
154.     "frame": {
155.         "header_bg_light": COLORS["frame"]["header_bg_light"],
156.         "header_bg_dark": COLORS["frame"]["header_bg_dark"],
157.         "header_top_border_light": COLORS["frame"]["header_top_border_light"],
158.         "header_top_border_dark": COLORS["frame"]["header_top_border_dark"],
159.         "header_top_border_width": PRESENTACIÓN["frame"]["header_top_border_width"]
160.     },
161.
162.     "files_to_copy": FILES_TO_COPY,
163.
164.     "debug": True
165. }

```

- líneas 1-7: se importan todos los archivos de configuración;
- línea 9: el archivo de configuración es un script de Python. Define una única variable [config]. Esta es un diccionario que contendrá todos los valores de la configuración;
- línea 11: el texto del enlace «Inicio»;
- línea 14: el diccionario [mkdocs] configurará el archivo [mkdocs.yml] generado por el conversor Gemini / ChatGPT. Este archivo se utiliza posteriormente en el script [build] para construir un sitio HTML a partir del sitio MkDocs generado por el conversor;
- líneas 15-20: configuran el sitio GitHub que alojará el sitio HTML creado a partir del documento ODT / DOCX (véase el apartado12)
- línea 21: esta línea es importante. Si no está presente, en lugar de mostrar una página HTML, el navegador abrirá la carpeta de dicha página;
- línea 24: MkDocs ofrece varios temas para un sitio MkDocs / HTML. Aquí hemos elegido el tema «material», línea 24;
- líneas 27-31: configuración de la navegación en el sitio. Esta se realizará mediante una tabla de contenidos situada en la columna izquierda de la página mostrada (línea 30);
- líneas 33-54: definen dos paletas de colores: el modo claro (líneas 35-42) o el modo oscuro (líneas 45-52). Un icono en la barra superior de las páginas mostradas permite cambiar de uno a otro;
- líneas 57-70: extensiones del lenguaje Markdown utilizado por MkDocs. Estas extensiones han sido generadas por Gemini a raíz de algunas de mis peticiones;

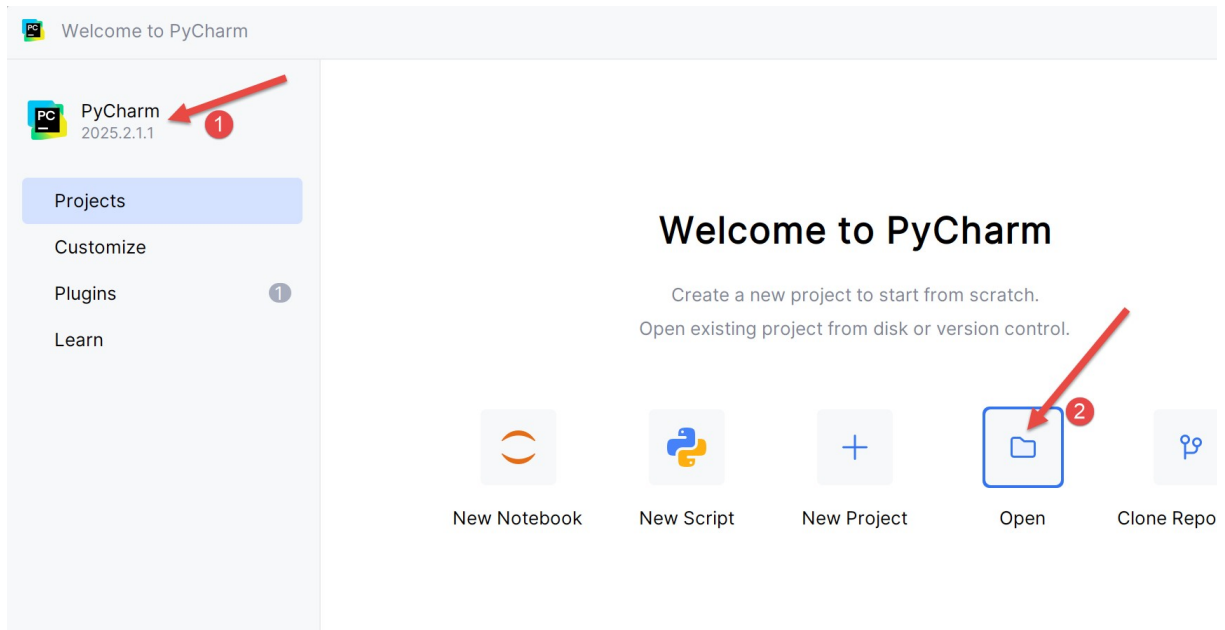
- línea 73: el script [focus.js] es un script de JavaScript generado por Gemini. Está asociado al botón de la barra superior del sitio para mostrar u ocultar la tabla de contenidos;
- línea 76: el CSS utilizado por este botón;
- líneas 80-95: la definición del pie de página del sitio HTML. Ha sido generado por Gemini a partir de un texto de ejemplo que le proporcioné;
- líneas 97-100: definición del marcador de Google Analytics (GA) que permitirá realizar un seguimiento de las visitas al sitio;
- línea 99: introduzca aquí su código de GA;
- líneas 103-112: el estilo del título del documento, el que aparece en el documento ODT/DOCX antes del primer título de nivel 1. El estilo indicado en la línea 104 cambia con cada documento ODT. En cambio, puede ser constante («Título») para los documentos DOCX. Por defecto, el conversor registra los estilos de todos los párrafos del documento ODT/DOCX que preceden al primer título de nivel 1. Por lo tanto, hay que realizar una primera ejecución, localizar el estilo del párrafo que se quiere convertir en título y, a continuación, introducirlo en el archivo [config_styles];;
- líneas 105-111: el estilo CSS que desea aplicar al título del documento. Este título se encuentra en la página «Inicio», la primera página que se muestra al abrir el sitio;
- línea 115: aquí se introducen los idiomas que no distinguen entre mayúsculas y minúsculas;
- línea 116: los bloques de código se identifican por sus estilos en el documento ODT / DOCX. Puede haber varios. En la línea 116, se incluyen todas las palabras clave que permiten detectar un estilo de código. En mis documentos, todos mis estilos de código tienen la palabra «code» en su nombre. Y ningún otro estilo tiene esa palabra en su nombre. Por lo tanto, me basta con poner una sola palabra clave. Si hay varias, se separan con comas en la línea 116;
- línea 117: establece el lenguaje por defecto. En el caso de un bloque de código «texto sin formato», si no se detecta ningún lenguaje, el lenguaje por defecto será «texto». Para MkDocs, esto significa un bloque de código sin coloración sintáctica. Es, por ejemplo, el caso de los resultados de ejecución;
- línea 121: esta configuración se utiliza para los códigos en «texto sin formato» que inicialmente no tienen coloración sintáctica. Estas líneas tienen como objetivo proporcionarles una según el lenguaje utilizado en el bloque de código. **Cabe señalar que si todos sus códigos están formateados porque proceden de un IDE como Eclipse, Visual Studio, etc., y no tiene ningún bloque de código de «texto sin formato» asociado a un lenguaje, entonces no tiene que introducir nada en esta parte de la configuración. Puede dejar vacías las tablas asociadas a los lenguajes.** Todo esto se lleva a cabo en el archivo de configuración del código [config_code];
- líneas 118-120: estas líneas se refieren a los bloques de código enriquecidos (negrita, cursiva, subrayado, resaltado, color de los caracteres). Estos bloques no se someten al mismo tratamiento que los bloques de código «texto sin formato». Se representan tal cual en el HTML;
 - línea 118: establece la altura de las líneas de código;
 - línea 119: establece la fuente CSS del bloque de código;
 - línea 120: establece el tamaño de los caracteres;
- líneas 144-152: establecen el aspecto de las imágenes;
- líneas 154-160: establecen el aspecto del marco en el que se muestra la página web;
- línea 162: la lista de archivos que se deben copiar en el directorio raíz del sitio HTML;
- línea 164: [debug] en True permite los registros de los estilos de los párrafos que preceden al primer título de nivel 1 del documento ODT / DOCX. Es en estos registros donde se encontrará el estilo exacto del párrafo que servirá de título en la página de inicio del sitio;

Al final, al pasar de un documento ODT/DOCX a otro, no se modificará nada en el archivo [config.py]. Solo se modificarán los demás archivos de configuración.

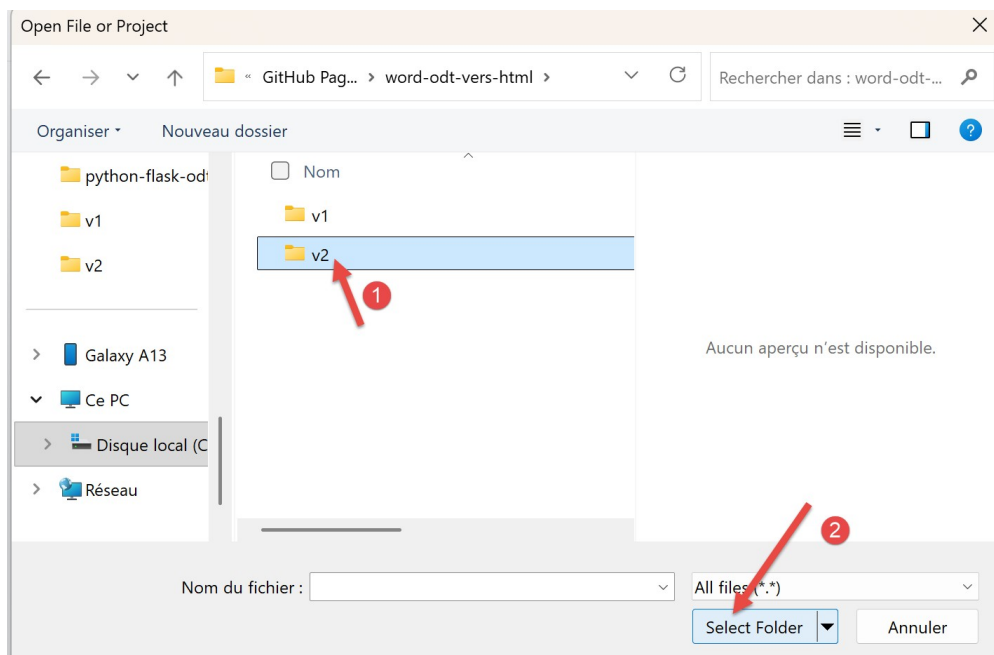
8. Uso del conversor ODT → HTML

Recordamos que la carpeta de trabajo se puede encontrar |[AQUÍ](#)|.

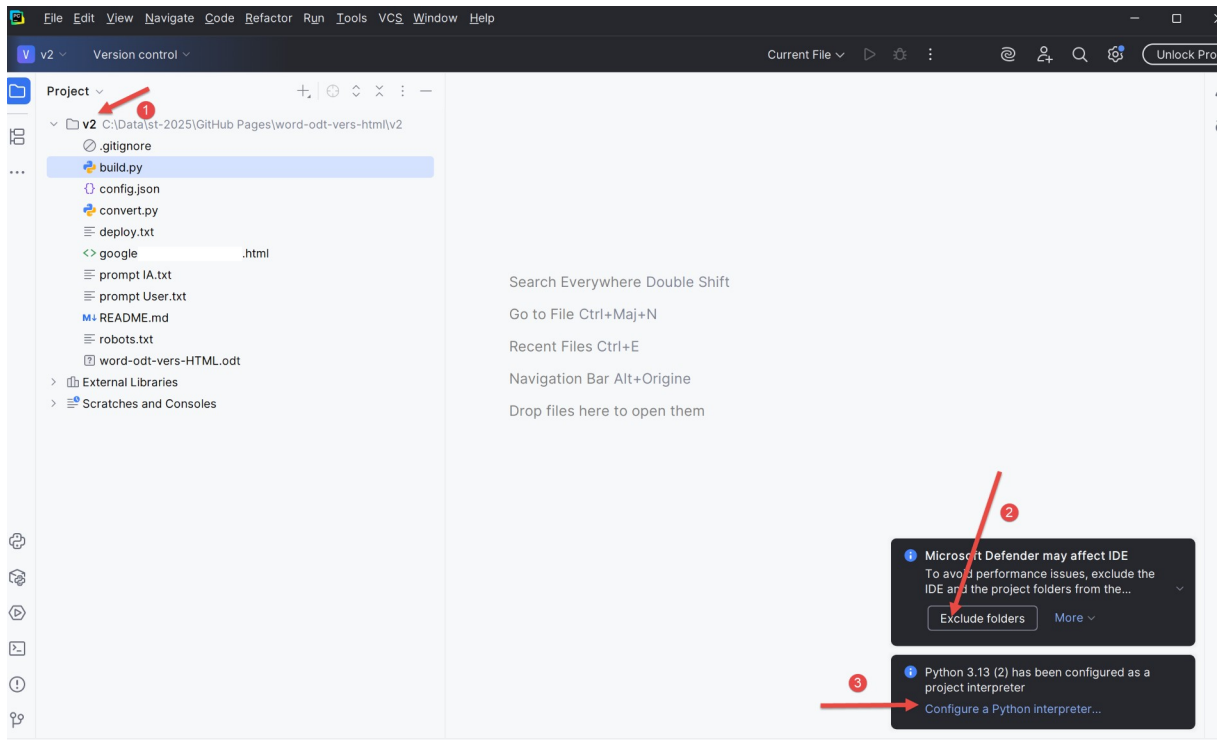
Iniciemos PyCharm:



- En [2], abra un nuevo proyecto;

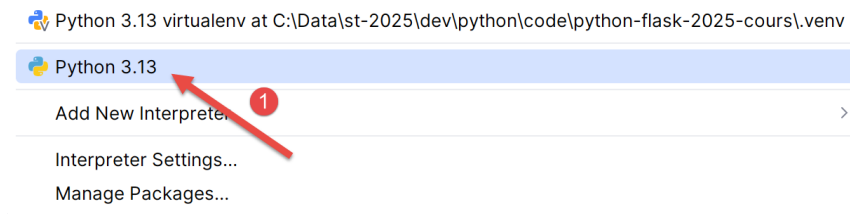


- En [1], seleccione la carpeta donde ha guardado la carpeta de trabajo;
- En [2], confirme esta selección;



- En [1], su proyecto Python;
- En [2], confirme para que el sistema de seguridad de Windows no le moleste;
- En [3], elija un intérprete de Python;

Python Interpreter



- En [1], elige el intérprete de Python que instalaste en el apartado 5 ;

El conversor Gemini / ChatGPT utiliza dependencias de Python que debes instalar. Vamos a preguntarle a Gemini cuáles son:

rappelle moi toutes les dépendances Python nécessaires à ton convertisseur [convert]

La respuesta de Gemini es la siguiente:

<https://tahe.developpez.com>

Comande d'installation rapide

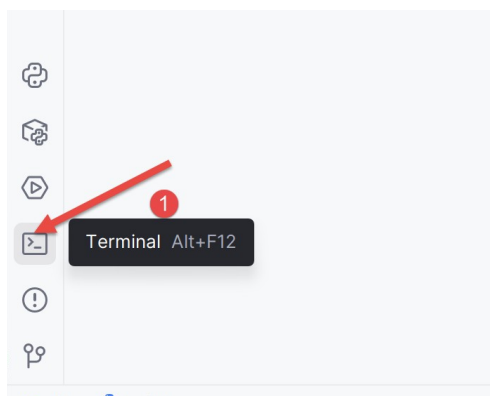
Vous pouvez installer tout le nécessaire en une seule commande dans votre terminal :

Bash



```
pip install odfpy unicode mkdocs mkdocs-material
```

Vamos a instalar estas dependencias:



- En [1], abra un terminal;



- En [2], escribe el comando proporcionado por Gemini;

La respuesta es la siguiente:

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> pip install odfpy unicode mkdocs mkdocs-material
2. Se utiliza la instalación del usuario por defecto, ya que la carpeta site-packages normal no es escribible
3. Requisito ya cumplido: odfpy en c:\users\serge\appdata\roaming\python\python313\site-packages (1.4.1)
4. Requisito ya cumplido: unicode en c:\users\serge\appdata\roaming\python\python313\site-packages (1.4.0)
5. Requisito ya cumplido: mkdocs en c:\users\serge\appdata\roaming\python\python313\site-packages (1.6.1)

6. Requisito ya cumplido: mkdocs-material en c:
\users\serge\appdata\roaming\python\python313\site-packages (9.7.0)
7. Requisito ya cumplido: defusedxml en c:\users\serge\appdata\roaming\python\python313\site-packages (de odfpy) (0.7.1)
8. Requisito ya cumplido: click>=7.0 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (8.3.1)
9. Requisito ya cumplido: colorama>=0.4 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (0.4.6)
10. Requisito ya cumplido: ghp-import>=1.0 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (2.1.0)
11. Requisito ya cumplido: jinja2>=2.11.1 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (3.1.6)
12. Requisito ya cumplido: markdown>=3.3.6 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (3.10)
13. Requisito ya cumplido: markupsafe>=2.0.1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (3.0.2)
14. Requisito ya cumplido: mergedeep>=1.3.4 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (1.3.4)
15. Requisito ya cumplido: mkdocs-get-deps>=0.2.0 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (0.2.0)
16. Requisito ya cumplido: packaging>=20.5 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (25.0)
17. Requisito ya cumplido: pathspec>=0.11.1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (0.12.1)
18. Requisito ya cumplido: pyyaml-env-tag>=0.1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (1.1)
19. Requisito ya cumplido: pyyaml>=5.1 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (6.0.2)
20. Requisito ya cumplido: watchdog>=2.0 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs) (6.0.0)
21. Requisito ya cumplido: babel>=2.10 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (2.17.0)
22. Requisito ya cumplido: backrefs>=5.7.post1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (6.1)
23. Requisito ya cumplido: mkdocs-material-extensions>=1.3 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (1.3.1)
24. Requisito ya cumplido: paginate>=0.5 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (0.5.7)
25. Requisito ya cumplido: pygments>=2.16 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (2.19.2)
26. Requisito ya cumplido: pymdown-extensions>=10.2 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (10.17.2)
27. Requisito ya cumplido: requests>=2.26 en c:\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-material) (2.32.5)
28. Requisito ya cumplido: python-dateutil>=2.8.1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de ghp-import>=1.0->mkdocs) (2.9.0.post0)
29. Requisito ya cumplido: platformdirs>=2.2.0 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de mkdocs-get-deps>=0.2.0->mkdocs) (4.5.0)
30. Requisito ya cumplido: six>=1.5 en c:\users\serge\appdata\roaming\python\python313\site-packages (de python-dateutil>=2.8.1->ghp-import>=1.0->mkdocs) (1.17.0)
31. Requisito ya cumplido: charset_normalizer<4,>=2 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de requests>=2.26->mkdocs-material) (3.4.3)
32. Requisito ya cumplido: idna<4,>=2.5 en c:\users\serge\appdata\roaming\python\python313\site-packages (de requests>=2.26->mkdocs-material) (3.10)
33. Requisito ya cumplido: urllib3<3,>=1.21.1 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de requests>=2.26->mkdocs-material) (2.5.0)
34. Requisito ya cumplido: certifi>=2017.4.17 en c:
\users\serge\appdata\roaming\python\python313\site-packages (de requests>=2.26->mkdocs-material) (2025.8.3)
35. PD: C:\Data\st-2025\GitHub\Pages\word-odt-vers-html\v2>

En mi PC, ya estaba todo instalado. Si no es tu caso, se instalarán todas las dependencias requeridas.

Ahora podemos utilizar el conversor. En el terminal abierto, escribe el siguiente comando:

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_odt_v356.py .\word-
odt-vers-html-janv-2026.odt .\config.py
2. --- Convertidor de ODT a MkDocs V356 ---
3. Procesando .\word-odt-vers-html-janv-2026.odt...
4. Copiado: google5179c0eaff293e02.html
5. Copiado: robots.txt
6. Copiado: word-odt-a-html-ene-2026.pdf
7. Copiado: word-odt-a-html-ene-2026.zip
8. Escaneando referencias...
9. Generación de Markdown...
10. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
11. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
12. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
13. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
14. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
15. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
16. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
17. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
18. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar' Texto='...'
19. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar (WW)' Texto='...'
20. [DEBUG PRE-H1] Etiqueta=text:p Estilo='Estándar (WW)' Texto='...'
21. [DEBUG PRE-H1] Tag=text:p Style='P1' Text='Convertir un documento Word u ODT a un sitio
est...'
22. >>> TÍTULO DEL DOCUMENTO ENCONTRADO: Convertir un documento Word u ODT a un sitio HTML
estático compatible con MkDocs con las IA Gemini 3 y ChatGPT 5.2
23. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P2' Texto='...'
24. [DEBUG PRE-H1] Tag=text:p Style='P2' Text='...'
25. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P2' Texto='...'
26. [DEBUG PRE-H1] Tag=text:p Style='P3' Text='Serge Tahé, enero de 2026...'
27. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P2' Texto='...'
28. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P2' Texto='...'
29. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P2' Texto='...'
30. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P4' Texto='...'
31. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P4' Texto='...'
32. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P4' Texto='...'
33. [DEBUG PRE-H1] Etiqueta=text:p Estilo='P4' Texto='Este sitio web ha sido creado con el
conversor [Word o ...]'
34. [DEBUG PRE-H1] Tag=text:h Style='P5' Text='Introducción...'
35. >>> CAPÍTULO 1: 1 Introducción
36. >>> CAPÍTULO 1: 2 Ejemplos de este documento
37. >>> CAPÍTULO 2: 2.1 Las listas
38. >>> CAPÍTULO 3: 2.1.1 Listas con viñetas
39. >>> CAPÍTULO 3: 2.1.2 Listas numeradas
40. >>> CAPÍTULO 3: 2.1.3 Listas mixtas 1
41. >>> CAPÍTULO 3: 2.1.4 Listas mixtas 2
42. >>> CAPÍTULO 2: 2.2 Los bloques de código
43. >>> CAPÍTULO 3: 2.2.1 Bloques de código enriquecido (Eclipse, Visual Studio, ...)
44. >>> CAPÍTULO 3: 2.2.2 Bloques de código sin formato (texto sin formato)
45. >>> CAPÍTULO 2: 2.3 Los enlaces
46. >>> CAPÍTULO 2: 2.4 El enriquecimiento de texto
47. >>> CAPÍTULO 2: 2.5 Un título también se puede formatear.
48. >>> CAPÍTULO 2: 2.6 Las imágenes
49. >>> CAPÍTULO 2: 2.7 Los caracteres que hay que proteger
50. >>> CAPÍTULO 2: 2.8 Las tablas
51. >>> CAPÍTULO 2: 2.9 Las notas al pie
52. >>> CAPÍTULO 1: 3 Lo que hay en Internet
53. >>> CAPÍTULO 1: 4 La indicación inicial en Gemini 3
54. >>> CAPÍTULO 1: 5 Crear un entorno de trabajo en Python
55. >>> CAPÍTULO 1: 6 La carpeta de trabajo del conversor
56. >>> CAPÍTULO 1: 7 El archivo de configuración del conversor
57. >>> CAPÍTULO 1: 8 Uso del conversor ODT → HTML
58. >>> CAPÍTULO 1: 9 Uso del conversor DOCX → HTML
59. >>> CAPÍTULO 1: 10 Creación del sitio HTML estático
60. >>> CAPÍTULO 1: 11 Revisión del sitio HTML generado
61. >>> CAPÍTULO 2: 11.1 La barra superior del sitio
62. >>> CAPÍTULO 2: 11.2 El pie de página del sitio

```

```

63. >>> CAPÍTULO 2: 11.3 La página de inicio
64. >>> CAPÍTULO 2: 11.4 Las listas con viñetas
65. >>> CAPÍTULO 2: 11.5 Las listas numeradas
66. >>> CAPÍTULO 3: 11.5.1 Listas mixtas 1
67. >>> CAPÍTULO 3: 11.5.2 Listas mixtas 2
68. >>> CAPÍTULO 2: 11.6 Los bloques de código enriquecidos
69. >>> CAPÍTULO 3: 11.6.1 Ejemplo 1
70. >>> CAPÍTULO 3: 11.6.2 Ejemplo 2
71. >>> CAPÍTULO 3: 11.6.3 Ejemplo 3
72. >>> CAPÍTULO 2: 11.7 Los bloques de código sin formato (texto sin formato)
73. >>> CAPÍTULO 3: 11.7.1 Ejemplo 1
74. >>> CAPÍTULO 3: 11.7.2 Ejemplo 2
75. >>> CAPÍTULO 3: 11.7.3 Ejemplo 3
76. >>> CAPÍTULO 2: 11.8 Otros bloques de código
77. >>> CAPÍTULO 2: 11.9 Los enlaces
78. >>> CAPÍTULO 2: 11.10 El enriquecimiento del texto
79. >>> CAPÍTULO 2: 11.11 Las imágenes
80. >>> CAPÍTULO 2: 11.12 Los caracteres especiales
81. >>> CAPÍTULO 2: 11.13 Las tablas
82. >>> CAPÍTULO 2: 11.14 Notas al pie
83. >>> CAPÍTULO 2: 11.15 Anomalías conocidas
84. >>> CAPÍTULO 2: 11.16 Otros casos
85. >>> CAPÍTULO 1: 12 Alojamiento del sitio HTML en GitHub
86. >>> CAPÍTULO 1: 13 Realizar el seguimiento del sitio con Google Analytics y Google Search
    Console
87. >>> CAPÍTULO 1: 14 Conclusión
88. Terminado.

```

- Línea 1: el comando que convierte el documento ODT en un sitio MkDocs [`python convert_odt_v356.py .\word-odt-vers-html-janv-2026.odt .\config.py`]. Adapta el número de versión (aquí 356) a la versión que hayas descargado. El primer parámetro del convertidor es el documento ODT que se va a convertir, el segundo es el archivo de configuración del convertidor;
- líneas 4-7: los archivos que el conversor copia en la raíz del sitio MkDocs que crea;
- líneas 10-34: depuración de los estilos de los párrafos que preceden al primer título de nivel 1. Estos párrafos constituirán la página de inicio. Uno de los párrafos hace las veces de título de la página de inicio y, por tanto, del sitio. Es el párrafo de la línea 21. Anotamos su estilo P1. Debemos incluir este estilo en el archivo de configuración [`config_styles`]:

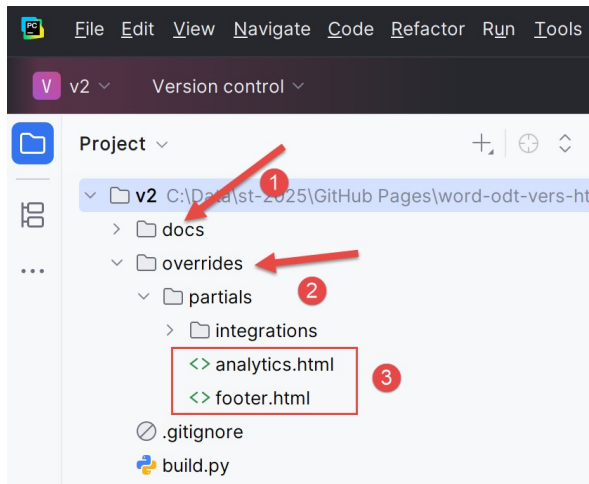
```

1.     STYLES = {
2.         "style_names": [
3.             "P1"
4.         ]
5.     }

```

- líneas 35-88: le pedí a la IA que registrara todos los capítulos que encontrara;

Esta ejecución ha modificado tu carpeta de trabajo:



- En [1], [docs] es el sitio MkDocs que ha creado el conversor Gemini / ChatGPT. Quizás te interese visitarlo;
- en [2], se ha creado una carpeta [overrides]. La utilizará el generador [build] del sitio HTML;
- en [3]: [analytics.html] se utilizará para el seguimiento del sitio mediante Google Analytics. [footer.html] es el pie de página que ha definido en el archivo [config.py];

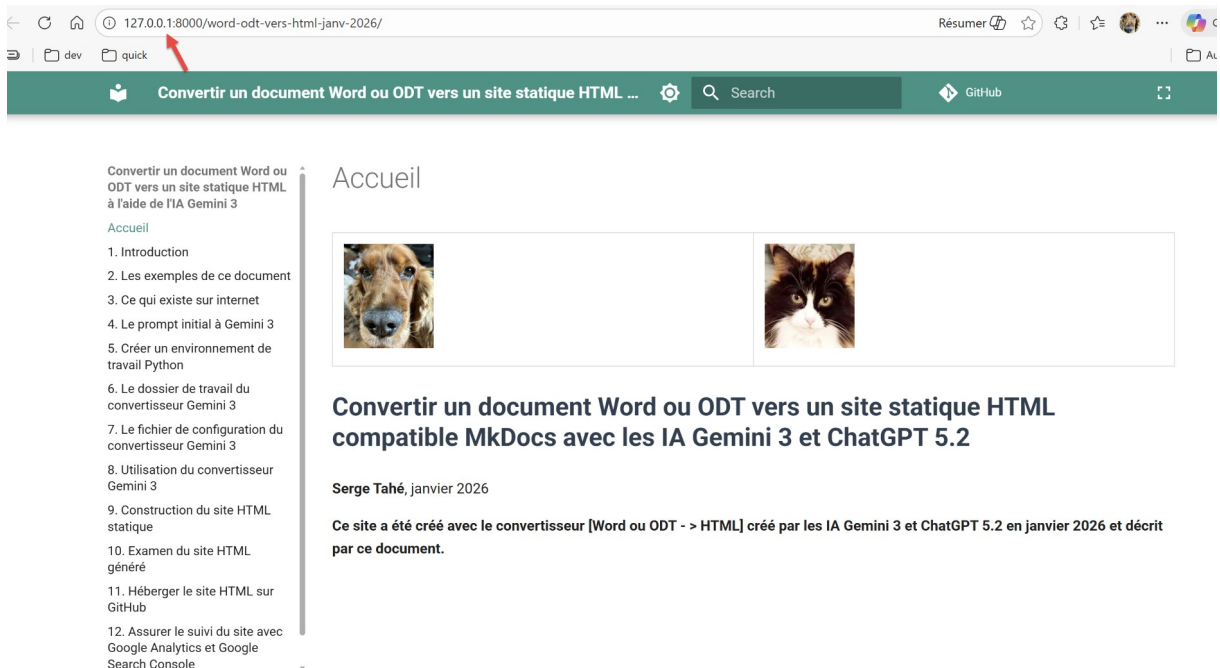
Ya podríamos utilizar el sitio MkDocs. El comando `[python -m mkdocs serve]` permite visualizarlo. Puedes probarlo:

```

1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2. INFO - Compilando la documentación...
3. INFO - Limpiando el directorio del sitio
4. INFO - El archivo de documentación «les-exemples.md» contiene un enlace «#_Les_exemples», pero no existe tal ancla en esta página.
5. INFO - Documentación generada en 0,41 segundos
6. INFO - [15:46:06] Servido en http://127.0.0.1:8000/word-odt-vers-html-janv-2026/

```

Haga clic con Ctrl en el enlace de la línea 6. Debería aparecer el sitio web de MkDocs:



Este es el resultado de un largo trabajo. Para detener el servidor de MkDocs, simplemente pulse Ctrl-C en el terminal que lo inició.

9. Uso del conversor DOCX → HTML

El comando para convertir un documento DOCX de Word es muy similar al de la conversión de un documento ODT de LibreOffice. Vamos a modificar el estilo del título del documento en [config.py]:

```
1.     STYLES = {
2.         "style_names": [
3.             "Título"
4.         ]
5.     }
```

- línea 3: escribe «Título». Este es el estilo del documento DOCX que vas a convertir. Lo veremos en las líneas de depuración del conversor.

Siempre en el terminal de PyCharm, escribe el siguiente comando:

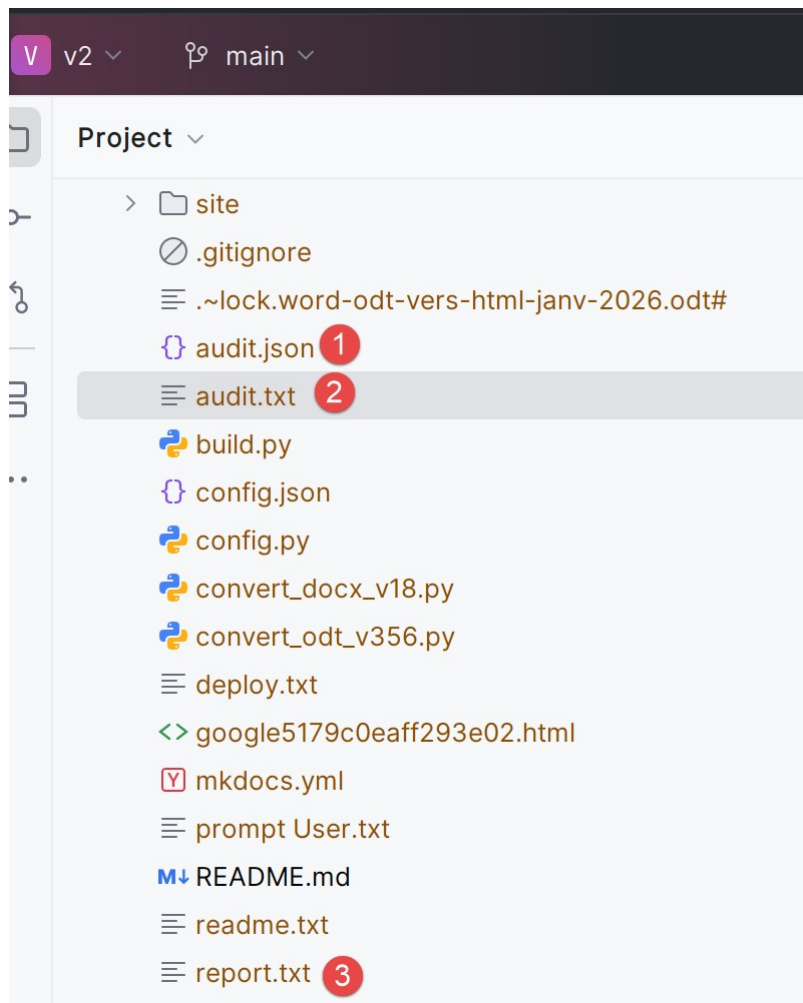
```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\convert_docx_v18.py .\word-
odt-vers-html-janv-2026.docx .\config.py
2. C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\convert_docx_v18.py:976: SyntaxWarning:
secuencia de escape no válida '\h'
3.   - Marcador REF \h
4. --- DOCX to MkDocs Converter V16 ---
5. Copiado: google5179c0eaff293e02.html
6. Copiado: robots.txt
7. Copiado: word-odt-a-html-ene-2026.pdf
8. Copiado: word-odt-a-html-ene-2026.zip
9. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
10. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
11. [DEBUG PRE-H1] estilo=Estándar encabezado=Ninguno rebase=0 numId=Ninguno ilvl=Ninguno
lista=Ninguno:0/Ninguno txt='...'
12. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
13. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
14. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
15. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
16. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
17. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
18. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
19. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
20. [DEBUG PRE-H1] style=Título heading=Ninguno rebase=0 numId=Ninguno ilvl=Ninguno
list=Ninguno:0/Ninguno txt='<span>Convertir un documento Word u ODT a un sitio HTML estático
compatible con Mk...'
21. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
22. [DEBUG PRE-H1] style=Estándar heading=Ninguno rebase=0 numId=Ninguno ilvl=Ninguno
list=Ninguno:0/Ninguno txt='...'
23. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
24. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Serge Tahé</b><span>, enero de 2026</span>...'
```

```

25. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
26. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
27. [DEBUG PRE-H1] style=Standard heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
28. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
29. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
30. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='...'
31. [DEBUG PRE-H1] style=StandardWW heading=None rebase=0 numId=None ilvl=None list=None:0/None
txt='<b>Este sitio ha sido creado con el conversor &#91;Word u ODT - &gt; HTML&#93; cr...'
32. [DEBUG PRE-H1] style=Título1 heading=1 rebase=0 numId=1 ilvl=0 list=numPr:1/ordered
txt='<span>Introducción</span>...'
33. Finalizado. (audit.json, audit.txt, report.txt generados)

```

- línea 1: el comando es el siguiente: `[python .\convert_docx_v18.py .\word-odt-vers-html-janv-2026.docx .\config.py]` (Adapta el número de versión (aquí 18) a la versión que hayas descargado):
 - el primer parámetro `[.\convert_docx_v18.py]` es el conversor de DOCX a HTML
 - el segundo parámetro `[.\word-odt-vers-html-janv-2026.docx]` es el nombre del documento DOCX que se va a convertir;
 - el tercer parámetro `[.\config.py]` es el archivo de configuración;
- línea 33: el conversor indica que se han generado tres archivos:



El archivo [audit.txt] es el siguiente:

```
1. Versión: V16
2. Párrafos: 2029
3. Tablas: 97
4. Imágenes (blips): 2
5. Encabezados detectados (sin procesar): 53
6. Nivel mínimo de encabezado detectado (sin procesar): 1
7. Desplazamiento de rebase aplicado: 0
8.
9. Estilos de párrafo principales:
10. - SourceCodenumrot: 1054
11. - StandardWW: 594
12. - Estándar: 146
13. - Parágrafo de la lista: 113
14. - SourceCodenumrotrrsultados: 33
15. - código nuevo: 28
16. - Título 2: 25
17. - Título 1: 14
18. - Título 3: 14
19. - EstándarWWW: 6
20. - Texto sin formato: 1
21. - Título: 1
22.
23. Lista de párrafos:
24. - con numeración: 1329
25. - por estilo de reserva: 49
26. - no reconocidos: 0
```

- línea 2: el número de párrafos del documento de Word;
- línea 3: el número de tablas;
- líneas 9-21: los estilos encontrados en el documento;
 - líneas 10, 14, 15: el estilo de los bloques de código. Probablemente habría bastado con un solo estilo;
 - líneas 11-12, 19: el estilo de los párrafos estándar. Probablemente habría bastado con un solo estilo;
 - líneas 16-18, 21: los estilos de los títulos del documento. En la línea 21, vemos que solo un párrafo tiene el estilo «Título». Se trata del título del documento que precede al primer «Título 1»;

Esta revisión del documento de Word es una buena forma de evaluar la calidad del documento. Aquí veo que he utilizado demasiados estilos diferentes para lo mismo en mi documento de Word.

El archivo [audit.json] es idéntico al archivo [audit.txt], pero en formato JSON:

```
1. {
2.   "version": "V16",
3.   "file": "word-odt-vers-html-janv-2026.docx",
4.   "counts": {
5.     "paragraphs": 2029,
6.     "tables": 97,
7.     "image_blips": 2,
8.     "headings_raw": 53
9.   },
10.  "listas": {
11.    "with_numpr": 1329,
12.    "by_style": 49,
13.    "unrecognized": 0
14.  },
15.  "heading": {
16.    "min_level_raw": 1,
17.    "rebase_offset": 0
```

```

18.  },
19.  "top_styles": [
20.    [
21.      "SourceCodenumrot",
22.      1054
23.    ],
24.    [
25.      "StandardWW",
26.      594
27.    ],
28.    [
29.      «Standard»,
30.      146
31.    ],
32.    [
33.      «Párrafo de la lista»,
34.      113
35.    ],
36.    [
37.      "Número de resultados del código fuente",
38.      33
39.    ],
40.    [
41.      "codenouveau",
42.      28
43.    ],
44.    [
45.      "Título2",
46.      25
47.    ],
48.    [
49.      "Título1",
50.      14
51.    ],
52.    [
53.      "Título 3",
54.      14
55.    ],
56.    [
57.      "EstándarWWW",
58.      6
59.    ],
60.    [
61.      "Texto sin formato",
62.      1
63.    ],
64.    [
65.      "Título",
66.      1
67.    ]
68.  ]
69. }

```

El archivo [report.txt] es este:

1. [RESUMEN] Listas detectadas mediante el método de reserva «por estilo» (agregado)
2. - Paragraphedeliste -> level=1 type=unordered: 49
- 3.
4. [RESUMEN] Bloques de Word ignorados (agregado)

```
5. - <w:sectPr>: 1
```

No lo he entendido...

Es posible solicitar solo la auditoría del documento de Word para evaluar su calidad con el parámetro [--audit]:

```
python .\convert_docx_v18.py .\word-odt-vers-html-janv-2026.docx .\config.py --audit
```

En este caso, solo se realiza la auditoría del documento. No se genera el sitio MkDocs.

Como se ha mostrado anteriormente, puede visualizar el sitio MkDocs generado por el conversor:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs serve
2. INFO - Compilando la documentación...
3. INFO - Limpiando el directorio del sitio
4. INFO - Documentación generada en 0,59 segundos
5. INFO - [06:05:48] Servido en http://127.0.0.1:8000/word-odt-vers-html-janv-2026/
```

Ctrl-Haga clic en la URL de la línea 5:



10. Creación del sitio HTML estático

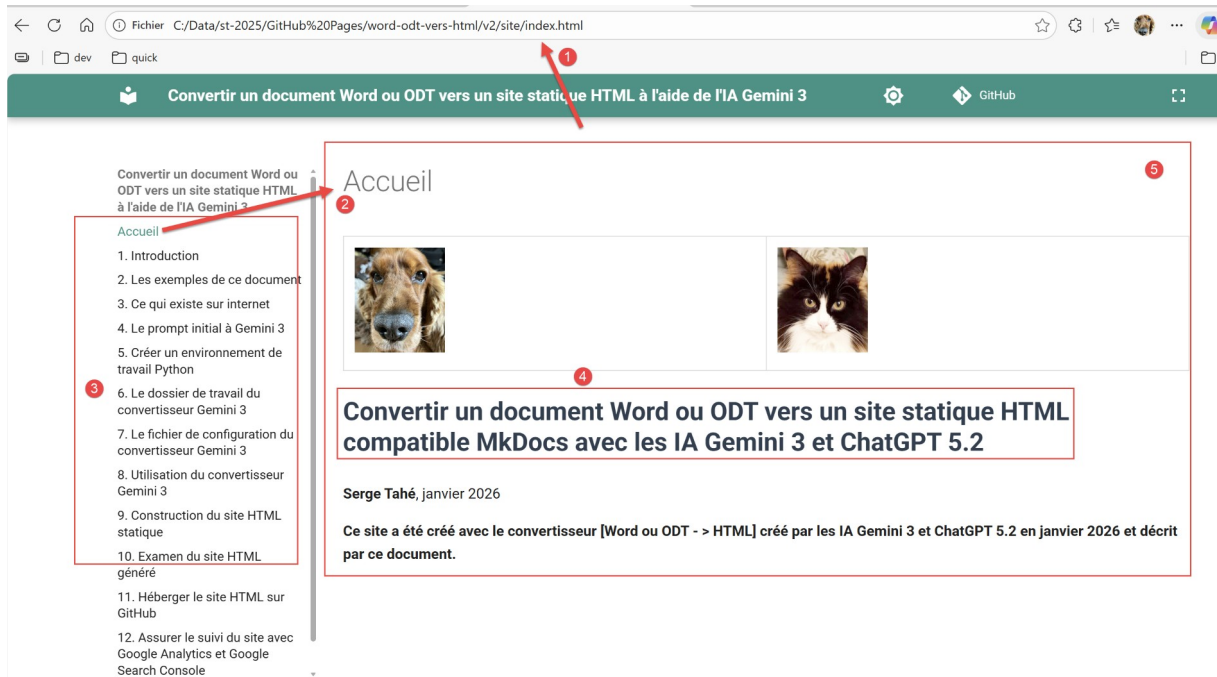
A partir de ahora ya no hay diferencia entre documentos ODT y DOCX. Trabajamos en el sitio MkDocs creado por cualquiera de los dos convertidores.

Ahora construyamos el sitio HTML. Esto se hace con el script [build]. Siguiendo en el terminal, escribe el siguiente comando:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python .\build.py
2. Inicio de la construcción del sitio MkDocs...
3. ¡Creación completada! El sitio se encuentra en: C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site
4. Apertura de C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site\index.html en el navegador...
```

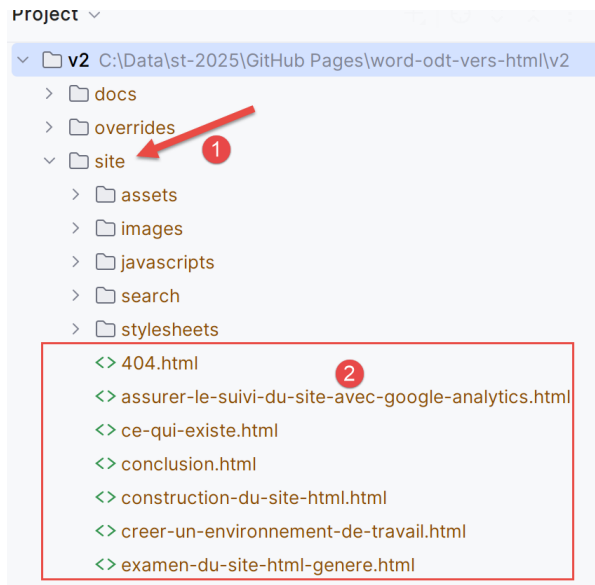
- línea 1: el comando [python build] crea el sitio HTML a partir del sitio MkDocs;
- línea 4: el sitio se muestra en un navegador;

El resultado es el mismo que para el sitio MkDocs:



- En [1], vemos que se muestra una página HTML;
- En [2], esta página corresponde a la página de inicio del sitio;
- En [3], el índice del documento HTML;
- En [4], el título del documento se ha formateado según su configuración en [config.py];
- En [5], la primera página mostrada es [Inicio]. El contenido de esta página es el contenido del documento ODT / DOCX que **precede al primer capítulo de nivel Título 1;**

La ejecución de [build] ha modificado su carpeta de trabajo:



- En [1], se ha creado una nueva carpeta [site]. Contiene su sitio HTML estático;
- En [2], las páginas de su sitio;

11. Revisión del sitio HTML generado

Ahora vamos a examinar el resultado HTML de este documento ODT / DOCX. Ya hemos visto que el conversor respeta la tabla de contenidos.

11.1. La barra superior del sitio

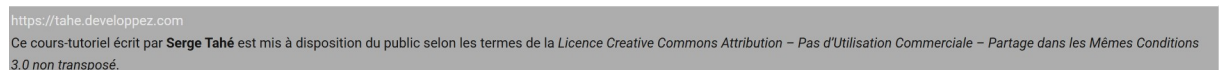
Echemos un vistazo a la barra superior del sitio:



- En [1], el nombre del sitio definido en [config.py];
- En [2], el icono que permite cambiar al modo oscuro o claro;
- En [3], el icono que es un enlace al repositorio de GitHub donde se exportará el sitio HTML. También definido en [config.py];
- En [4], el icono que permite ocultar o mostrar el índice;

11.2. El pie de página del sitio

Veamos ahora el pie de página:



Este es el pie de página definido en el archivo [config.py].

11.3. La página de inicio

La página de título del documento ODT/DOCX era la siguiente:



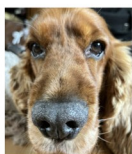
Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec les IA Gemini 3 et ChatGPT 5.2

Serge Tahé, janvier 2026

Ce site a été créé avec le convertisseur [Word ou ODT - > HTML] créé par les IA Gemini 3 et ChatGPT 5.2 en janvier 2026 et décrit par ce document.

Esta página de título del documento ODT/DOCX se convierte en la página de inicio del sitio HTML:

Accueil



Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec les IA Gemini 3 et ChatGPT 5.2 1

Serge Tahé, janvier 2026

Ce site a été créé avec le convertisseur [Word ou ODT - > HTML] créé par les IA Gemini 3 et ChatGPT 5.2 en janvier 2026 et décrit par ce document.

El conversor Gemini 3 / ChatGPT incluye en la página de inicio todo lo anterior, es decir, en el documento ODT / DOCX, el primer título de nivel 1, con el estilo «Título 1». Si incluye imágenes como las anteriores, las mostrará. Por lo tanto, puede imaginar que le da una portada a su sitio web como si fuera un libro real. En [1], se trata del título principal del documento. Su visualización se controla mediante las siguientes líneas del archivo de configuración [config.py]:

```
1. # -----
2. # Detección del título del documento
3. # -----
4. "document_title": {
5.     # Estilos ODT que se deben considerar como el título principal del documento (H1
6.     global)
7.     "style_names": [
8.         "P1"
9.     ],
10.    # CSS aplicado a este título en el Markdown generado
11.    "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2;
12.    color: #2c3e50;"
13. },
```

- líneas [6-8]: la lista de estilos disponibles para el título de tu documento. Cuando miro este documento, el estilo de LibreOffice de mi título es «Título principal». Pero el conversor Gemini no lo encontraba. Generó registros de los estilos que encontraba y mostró [P1]. Este es un gran problema con LibreOffice: los nombres que se muestran de los estilos no se corresponden con los nombres internos que utiliza el programa. Solo están ahí para adaptarse al idioma del usuario;
- línea 10: una vez detectado el título principal, puedes configurar su apariencia. Yo quería una fuente de tamaño 28 (font-size: 28px;) y negrita (font-weight: bold);

Con las imágenes y el estilo del título, puedes crear una página de inicio atractiva.

Es posible que el título principal de tu documento no tenga ninguno de los estilos definidos en las líneas [6-8]. Para encontrar el estilo de tu título principal, utiliza la siguiente línea del archivo [config.py]

```
1. "debug": True
```

Con el valor [true], el estilo de los párrafos que preceden al primer título de nivel 1, es decir, los párrafos de la página de inicio, se mostrará al ejecutar el conversor Gemini / ChatGPT. Así, para un documento distinto a este, obtuve los siguientes registros:

```
1. [DEBUG PRE-H1] Style='Standard (WW)' (Clean='standard (ww)') | Texto='...'
2. [DEBUG PRE-H1] Estilo='P1' (Limpio='p1') | Texto='...'
3. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texto='...'
4. [DEBUG PRE-H1] Style='P1' (Clean='p1') | Texto='...'
5. [DEBUG PRE-H1] Estilo='P3' (Limpieza='p3') | Texto='...'
6. [DEBUG PRE-H1] Estilo='P1' (Limpio='p1') | Texto='...'
7. [DEBUG PRE-H1] Estilo='P1' (Limpio='p1') | Texto='...'
8. [DEBUG PRE-H1] Estilo='P1' (Limpio='p1') | Texto='...'
9. [DEBUG PRE-H1] Estilo='P1' (Limpio='p1') | Texto='...'
10. [DEBUG PRE-H1] Estilo='P4' (Limpio='p4') | Texto='Introducción al lenguaje PHP7 a través de
11. ejemplos...'
12. >>> TÍTULO DEL DOCUMENTO DETECTADO: Introducción al lenguaje PHP7 a través de ejemplos
13. [DEBUG PRE-H1] Estilo='Estándar (WW)' (Limpio='estándar (ww)') | Texto='...'
14. [DEBUG PRE-H1] Estilo='Estándar (WW)' (Limpio='estándar (ww)') | Texto='...'
15. [DEBUG PRE-H1] Estilo='Estándar (WW)' (Limpio='estándar (ww)') | Texto='...'
16. [DEBUG PRE-H1] Estilo='Estándar (WW)' (Limpieza='estándar (ww)') | Texto='...'
17. [DEBUG PRE-H1] Estilo='Estándar (WW)' (Limpieza='estándar (ww)') | Texto='...'
18. [DEBUG PRE-H1] Estilo='P5' (Limpieza='p5') | Texto='Serge Tahé, julio de 2019...'
19. [DEBUG PRE-H1] Estilo='P5' (Limpio='p5') | Texto='...'
20. [DEBUG PRE-H1] Estilo='P6' (Limpio='p6') | Texto='...'
21. [DEBUG PRE-H1] Estilo='Encabezado 1' (Clean='encabezado 1') | Texto='Introducción al lenguaje
22. PHP 7...'
```

- línea 10, el título del documento tiene el estilo «P4»;

En el archivo [config.py], he añadido las siguientes líneas:

```
1. "document_title": {
2.   "style_names": [
3.     "P4"
4.   ],
5.   "css": "font-size: 28px; font-weight: bold; margin-bottom: 1em; line-height: 1.2; color:
6.     #2c3e50;"
7. },
```

- línea 3, el estilo que buscaba;

Por este motivo, el depurador muestra las líneas:

```
1. [DEBUG PRE-H1] Style='P4' (Clean='p4') | Texto='Introducción al lenguaje PHP7 a través de
ejemplos...'
2. >>> TÍTULO DEL DOCUMENTO DETECTADO: Introducción al lenguaje PHP7 a través de ejemplos
```

Ha encontrado el estilo «P4» y muestra que se ha encontrado el título del documento. Una vez encontrado, puedes establecer la clave [debug] en [false] en [config.py]:

```
1. "debug": False
```

Veamos ahora la conversión del capítulo [Ejemplos], que agrupa los ejemplos que el conversor Gemini / ChatGPT sabe gestionar:

11.4. Listas con viñetas

Documento ODT / DOCX:

Listes à puces

- Élément 1 ;
- Élément 2 ;
- Élément 3 ;
 - Élément 3.1 ;
 - Élément 3.1.1
 - Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Arriba, el texto [Listas con viñetas] aparece resaltado porque es una referencia asociada a un enlace.

Documento HTML:

Listes à puces

- Élément 1 ;
- Élément 2 :
- Élément 3 ;
 - Élément 3.1 ;
 - Elément 3.1.1
 - Elément 3.1.2
 - Elément 3.1.2.1
 - Elément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Se observa que, mientras que el documento ODT / DOCX utiliza diferentes viñetas, el documento HTML solo utiliza un tipo de viñeta.

11.5. Listas numeradas

Documento ODT / DOCX:

Listes numérotées

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 1. Élément 2.1.1.1
 2. Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

Documento HTML:

2.1.2. Listes numérotées

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 1. Élément 2.1.1.1
 2. Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

11.5.1. Listas mixtas 1

Documento ODT / DOCX

2.1.3. Listes mixtes 1

- Élément 1 ;
- Élément 2 ;
- Élément 3 ;
 - Élément 3.1 ;
 1. Élément 3.1.1
 2. Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

Documento HTML

2.1.3. Listes mixtes 1

- Élément 1 ;
- Élément 2 :
- Élément 3 ;
 - Élément 3.1 ;
 1. Élément 3.1.1
 2. Élément 3.1.2
 - Élément 3.1.2.1
 - Élément 3.1.2.2
 - Élément 3.2 ;
- Élément 4 ;

En este caso también, a veces hay diferencias entre los tipos de viñetas utilizados.

11.5.2. Listas mixtas 2

Documento ODT / DOCX

2.1.4. Listes mixtes 2

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 - Élément 2.1.1.1
 - Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

Documento HTML

2.1.4. Listes mixtes 2

1. Élément 1 ;
2. Élément 2 ;
 - a. Élément 2.1
 - i. Élément 2.1.1
 - Élément 2.1.1.1
 - Élément 2.1.1.2
 - ii. Élément 2.1.2
 - b. Élément 2.2
3. Élément 3 ;

11.6. Los bloques de código enriquecido

Los bloques de código enriquecido se representan de forma idéntica en HTML (excepto el color de fondo). A continuación se muestran tres ejemplos:

11.6.1. Ejemplo 1

Documento ODT / DOCX

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplica
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // récupération du contexte Spring
14.         ApplicationContext ctx = new ClassPathXmlAppli
15.         // on récupère les beans
16.         Personne p01 = ctx.getBean("personne_01", Perso
17.         Personne p02 = ctx.getBean("personne_02", Perso
18.         List<Personne> club = ctx.getBean("club", new
19.         ArrayList<Personne>().getClass());
20.         Appartement appart01 = ctx.getBean(Appartement
```

Representación HTML

Java

```
1 package istia.st.spring.core;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.context.ApplicationContext;
7 import org.springframework.context.support.ClassPathXmlApplicat
8
9 public class Demo01 {
10
11     @SuppressWarnings({ "unchecked", "resource" })
12     public static void main(String[] args) {
13         // récupération du contexte Spring
14         ApplicationContext ctx = new ClassPathXmlApplicationCor
15         // on récupère les beans
16         Personne p01 = ctx.getBean("personne_01", Personne.class);
17         Personne p02 = ctx.getBean("personne_02", Personne.class);
18         List<Personne> club = ctx.getBean("club", new ArrayList<>());
19         Appartement appart01 = ctx.getBean(Appartement.class);
```

11.6.2. Ejemplo 2

Documento ODT / DOCX:

Python

```
1. # -----
2. def affiche(chaine):
3.     # affiche chaine
4.     print("chaine=%s" % chaine)
5.
6.
7. # -----
8. def affiche_type(variable):
9.     # affiche le type de variable
10.    print("type[%s]=%s" % (variable, type(variable)))
11.
12.
13. # -----
14. def f1(param):
15.    # ajoute 10 à param
16.    return param + 10
17.
18.
19. # -----
20. def f2():
21.    # rend un tuple de 3 valeurs
22.    return "un", 0, 100
23.
24.
25. # -----
26. ...
```

Resultado HTML

Python

```
1 # -----
2 def affiche(chaine):
3     # affiche chaine
4     print("chaine=%s" % chaine)
5
6
7 # -----
8 def affiche_type(variable):
9     # affiche le type de variable
10    print("type[%s]=%s" % (variable, type(variable)))
11
12
13 # -----
14 def f1(param):
15     # ajoute 10 à param
16     return param + 10
17
18
19 # -----
20 def f2():
21     # rend un tuple de 3 valeurs
22     return "un", 0, 100
```

11.6.3. Exemple 3

Document ODT / DOCX

ECMAScript

```
1. 'use strict';
2. // ceci est un commentaire
3. // constante
4. const nom = "dupont";
5. // un affichage écran
6. console.log("nom : ", nom);
7. // un tableau avec des éléments de type différents
8. const tableau = ["un", "deux", 3, 4];
9. // son nombre d'éléments
10. let n = tableau.length;
11. // une boucle
12. for (let i = 0; i < n; i++) {
13.     console.log("tableau[" + i + "] = ", tableau[i]);
14. }
15. // initialisation de 2 variables avec le contenu
16. let [chaine1, chaine2] = ["chaine1", "chaine2"];
17. // concaténation des 2 chaînes
18. const chaine3 = chaine1 + chaine2;
19. // affichage résultat
20. console.log([chaine1, chaine2, chaine3]);
21. ...
```

Resultado HTML

ECMAScript

```
1  'use strict';
2  // ceci est un commentaire
3  // constante
4  const nom = "dupont";
5  // un affichage écran
6  console.log("nom : ", nom);
7  // un tableau avec des éléments de type différent
8  const tableau = ["un", "deux", 3, 4];
9  // son nombre d'éléments
10 let n = tableau.length;
11 // une boucle
12 for (let i = 0; i < n; i++) {
13   console.log("tableau[" + i + "] = ", tableau[i]);
14 }
15 // initialisation de 2 variables avec le contenu d
16 let [chaine1, chaine2] = ["chaine1", "chaine2"];
17 // concaténation des 2 chaînes
18 const chaine3 = chaine1 + chaine2;
19 // affichage résultat
20 console.log([chaine1, chaine2, chaine3]);
21 ...
```

11.7. Los bloques de código sin formato (texto sin formato)

Los bloques de código sin formato que se encuentran en el documento ODT / DOCX son resaltados sintácticamente por MkDocs según el lenguaje detectado en el bloque de código. Para ayudar al conversor a identificar el lenguaje correcto, se han incluido «cadenas clave» en el archivo [config.py] para cada lenguaje. El conversor cuenta las «cadenas clave» encontradas. A continuación, asocia el bloque de código al lenguaje que tenga más «cadenas clave» encontradas.

Veamos algunos ejemplos.

11.7.1. Ejemplo 1

Documento ODT / DOCX (Java)

Java

```
1. package istia.st.spring.core;
2.
3. import java.util.ArrayList;
4. import java.util.List;
5.
6. import org.springframework.context.ApplicationContext;
7. import org.springframework.context.support.ClassPathXmlApplicationContext;
8.
9. public class Demo01 {
10.
11.     @SuppressWarnings({ "unchecked", "resource" })
12.     public static void main(String[] args) {
13.         // récupération du contexte Spring
14.         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15.         // on récupère les beans
16.         Personne p01 = ctx.getBean("personne_01", Personne.class);
17.         Personne p02 = ctx.getBean("personne_02", Personne.class);
18.         List<Personne> club = ctx.getBean("club", new ArrayList<Personne>().getClass());
19.         Appartement appart01 = ctx.getBean(Appartement.class);
20.         ...
    }
```

Resultado HTML

Java

```
1 package istia.st.spring.core;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import org.springframework.context.ApplicationContext;
7 import org.springframework.context.support.ClassPathXmlApplicationContext;
8
9 public class Demo01 {
10
11     @SuppressWarnings({ "unchecked", "resource" })
12     public static void main(String[] args) {
13         // récupération du contexte Spring
14         ApplicationContext ctx = new ClassPathXmlApplicationContext("config-01.xml");
15         // on récupère les beans
16         Personne p01 = ctx.getBean("personne_01", Personne.class);
17         Personne p02 = ctx.getBean("personne_02", Personne.class);
18         List<Personne> club = ctx.getBean("club", new ArrayList<Personne>().getClass());
19         Appartement appart01 = ctx.getBean(Appartement.class);
20         ...
    }
```

En el documento HTML se observa que el código Java se ha resaltado sintácticamente.

11.7.2. Ejemplo 2

Documento ODT / DOCX (XML)

```

1.     <?xml version="1.0" encoding="utf-8" ?>
2.     <configuration>
3.
4.         <configSections>
5.             <sectionGroup name="spring">
6.                 <section name="context" type="Spring.Context.Support.ContextSupport" />
7.                 <section name="objects" type="Spring.Context.Support.DefaultSpring.Core" />
8.             </sectionGroup>
9.         </configSections>
10.
11.        <spring>
12.            <context>
13.                <resource uri="config://spring/objects" />
14.            </context>
15.            <objects xmlns="http://www.springframework.net">
16.                <object name="dao" type="Dao.DataBaseImpot, ImpotsV7-dao">
17.                    <constructor-arg index="0" value="MySQL.Data.MySqlClient"/>
18.                    <constructor-arg index="1"
19. value="Server=localhost;Database=bdimpots;Uid=admimpots;Pwd=mdpimpots;"/>
20.                    <constructor-arg index="2" value="select limite, coeffr, c
21.                </object>
22.                <object name="metier" type="Metier.ImpotMetier, ImpotsV7-met
23.                    <constructor-arg index="0" ref="dao"/>
24.                </object>
25.            </objects>
26.        </spring>
27.    </configuration>

```

Resultado HTML

XML

```

1     <?xml version="1.0" encoding="utf-8" ?>
2     <configuration>
3
4     <configSections>
5         <sectionGroup name="spring">
6             <section name="context" type="Spring.Context.Support
7             <section name="objects" type="Spring.Context.Support
8         </sectionGroup>
9     </configSections>
10
11    <spring>
12        <context>
13            <resource uri="config://spring/objects" />
14        </context>
15        <objects xmlns="http://www.springframework.net">
16            <object name="dao" type="Dao.DataBaseImpot, ImpotsV
17                <constructor-arg index="0" value="MySQL.Data.MySql
18                <constructor-arg index="1" value="Server=localhost
19                <constructor-arg index="2" value="select limite, c
20            </object>
21            <object name="metier" type="Metier.ImpotMetier, Impo
22                <constructor-arg index="0" ref="dao"/>
23            </object>
24        </objects>
25    </spring>
26 </configuration>

```

11.7.3. Ejemplo 3

Documento ODT / DOCX (HTML)

HTML

```
1. <!DOCTYPE HTML>
2. <HTML>
3.   <head>
4.     <title>Laragon</title>
5.
6.     <link href="https://fonts.googleapis.com/css?
   type="text/css">
7.
8.     <style>
9.       HTML, body {
10.        height: 100%;
11.      }
12.
13.      body {
14.        margin: 0;
15.        padding: 0;
16.        width: 100%;
17.        display: table;
18.        font-weight: 100;
19.        font-family: 'Karla';
20.      }
21.
22.      .container {
23.        text-align: center;
24.        display: table-cell;
25.        vertical-align: middle;
26.      }
27.
28.      .content {
29.        text-align: center;
30.        display: inline-block;
31.      }
```

Resultado HTML

HTML

```
1  <!DOCTYPE HTML>
2  <HTML>
3    <head>
4      <title>Laragon</title>
5
6      <link href="<a href="view-source:https://font
7
8      <style>
9        HTML, body {
10         height: 100%;
11       }
12
13       body {
14         margin: 0;
15         padding: 0;
16         width: 100%;
17         display: table;
18         font-weight: 100;
19         font-family: 'Karla';
20       }
21
22       .container {
23         text-align: center;
24         display: table-cell;
25         vertical-align: middle;
26       }
27
```

11.8. Otros bloques de código

Documento ODT / DOCX

Un resultado de ejecución con una primera línea que no empieza por 1:

Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7. C:\Data\st-2020\dev\python\cours-2020\python3-flask-2020\ve
C:/Data/st-2020/dev/python/cours-2020/python3-flask-2020/ba
8. nom=dupont
9. liste[0]=un
10. liste[1]=deux
11. liste[2]=3
12. liste[3]=4
13. [chaine1,chaine2,chaine1chaine2]
14. chaine=chaine1
15. type[4]=<class 'int'>
16. type[chaine1]=<class 'str'>
17. type[['un', 'deux', 3, 4]]=<class 'list'>
18. type[a changé]=<class 'str'>
```

Resultado HTML

Résultats d'exécution

On notera que le code ne commence pas avec la ligne n° 1.

```
7 C:\Data\st-2020\dev\python\cours-2020\python3-flask-2
8 nom=dupont
9 liste[0]=un
10 liste[1]=deux
11 liste[2]=3
12 liste[3]=4
13 [chaine1,chaine2,chaine1chaine2]
14 chaine=chaine1
15 type[4]=<class 'int'>
16 type[chaine1]=<class 'str'>
17 type[['un', 'deux', 3, 4]]=<class 'list'>
18 type[a changé]=<class 'str'>
```

Un bloque de código sin numerar en ODT, el resto en HTML:

Documento ODT / DOCX

Le convertisseur sait gérer les blocs de code non numérotés.

```
liste[0]=un
liste[1]=0
liste[2]=100
liste[0]=8
liste[1]=5
somme=13
```

Resultado HTML

Le convertisseur sait gérer les blocs de code non numérotés.

```
liste[0]=un
liste[1]=0
liste[2]=100
liste[0]=8
liste[1]=5
somme=13
```

11.9. Los enlaces

Documento ODT / DOCX

2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [\[Générer un script Python avec des outils d'IA\]](#).

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Un renvoi vers un chapitre : [2.1](#).

Un renvoi vers un repère de texte qui précède : [Gemini 3](#)

Un renvoi vers un repère de texte qui suit : [GitHub](#)

Resultado HTML

2.3. Les liens

Le convertisseur Gemini sait conserver les liens externes du document ODT. Par exemple [Gemini 3](#) ou [\[Générer un script Python avec des outils d'IA\]](#).

Il sait gérer un lien vers un chapitre [Lien vers un chapitre](#)

Un renvoi vers un chapitre : [2.1](#).

Un renvoi vers un repère de texte qui précède : [Gemini 3](#)

Un renvoi vers un repère de texte qui suit : [GitHub](#)

11.10. Formato de texto

Documento ODT / DOCX

Un texte avec des mots en **gras**, en *italiques*, soulignés ou **surlignés** ou **surlignés** ou **surlignés**.

C'est vrai également pour les liens : [\[Générer un script Python avec des outils d'IA\]](#).

Le **convertisseur** **gère** également la **couleur** des **caractères**.

Il gère également les bordures supérieure et inférieure des paragraphes.

2.5. Un **titre** peut être également enrichi.

Representación HTML

2.4. L'enrichissement de texte

Le convertisseur sait gérer le gras, l'italique, le souligné et le surlignage. Il respecte la couleur du surlignage.

Un texte avec des mots en **gras**, en *italiques*, soulignés ou **surlignés** ou **surlignés** ou **surlignés**.

C'est vrai également pour les liens : [Générer un script Python avec des outils d'IA](#).

Le **convertisseur** gère également la **couleur** des **caractères**.

Il gère également les bordures supérieure et inférieure des paragraphes.

2.5. Un titre peut être également enrichi.

11.11. Las imágenes

2.5. Les images

Le convertisseur Gemini sait gérer les images et les tableaux d'images :



Cabe señalar que el conversor Gemini / ChatGPT respeta los cambios de tamaño de las imágenes realizados en el documento ODT / DOCX.

11.12. Los caracteres protegidos

Documento ODT / DOCX

L'astérisque * a une signification Markdown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à $0.15 * R - 2072.5 * nbParts$.

Resultado HTML

L'astérisque * a une signification Markdown. La ligne suivante peut être alors mal interprétée :

L'impôt I est alors égal à $0.15 \cdot R - 2072.5 \cdot \text{nbParts}$.

Documento ODT / DOCX (Markdown)

```
5. ---
6.
7. ## 📄 Description
8.
9. On se propose dans ce projet de mettre à disposition du
documents Word ou ODT vers un site statique HTML.
10.
11. Lorsque le document ODT convient, le convertisseur prod
l'aspect professionnel des sites produits par Pandoc.
12.
13. ## 📄 Contexte de création
14.
15. Ce convertisseur a été entièrement construit par l'IA *
est le résultat d'itérations successives pour gérer finement la
(OpenDocument Text).
16.
17. ## ✨ Fonctionnalités
18.
19. Le script `convert.py` effectue les actions suivantes :
20.
21. * **Conversion ODT vers Markdown** : Analyse le fichier
structure.
22. * **Gestion des Titres** : Génère automatiquement la Ta
latérale.
23. * **Blocs de Code** : Détection automatique des langage
précise de la numérotation des lignes** (attributs `start-value
24. * **Listes** : Support des listes à puces et numérotées
25. * **Mise en forme** : Support du *gras*, *italique*, *s
respect des couleurs d'origine).
```

Resultado HTML

Un autre exemple est lorsque vous voulez insérer un bloc de code Markdown dans votre document

```


1 # Convertisseur Word/ODT vers Site HTML (MkDocs)
2
3 🔗 **[Voir le site de démonstration généré](https://stahe.github.io/word-odt-v
4
5 ---
6
7 ## 📖 Description
8
9 On se propose dans ce projet de mettre à disposition du lecteur un convertisseur
10
11 Lorsque le document ODT convient, le convertisseur produit un site HTML via **
12
13 ## 🏠 Contexte de création
14
15 Ce convertisseur a été entièrement construit par l'IA **Gemini 3** (avec un abo
16
17 ## ✨ Fonctionnalités
18
19 Le script `convert.py` effectue les actions suivantes :
20

```

- Se ha conservado el código Markdown;


11.13. Las tablas

Documento ODT / DOCX

1	2		<pre> 1. package istia.st.spring.core; 2. 3. import java.util.ArrayList; 4. import java.util.List; 5. 6. import org.springframework.context.ApplicationContext; 7. import org.springframework.context.support.ClassPathXmlApplication Context; 8. 9. public class Demo01 { 10. 11. </pre>
3	4		<pre> 1. # ----- 2. def affiche(chaine): 3. # affiche chaîne 4. print("chaîne=%s" % chaine) 5. 6. 7. # ----- 8. def affiche_type(variable): 9. # affiche le type de variable 10. print("type[%s]=%s" % (variable, 11. type(variable))) 12. </pre>

Resultado HTML

Un tableau peut contenir différents contenus :

1	2		<pre>1 package istia.st.spring.core; 2 3 import java.util.ArrayList; 4 import java.util.List; 5 6 import org.springframework.context.ApplicationContext; 7 import org.springframework.context.support.ClassPathXmlApplicationConte 8 9 public class Demo01 { 10 11</pre>
3	4		<pre>1 # ----- 2 def affiche(chaine): 3 # affiche chaine 4 print("chaine=%s" % chaine) 5</pre>

11.14. Notas al pie

Le convertisseur Gemini² gère les notes de bas de page. Voici une autre note³ de bas de page.

-
1. MkDocs
 2. Google Gemini
 3. La note de bas de page

11.15. Anomalías conocidas

Se han detectado algunas anomalías, pero pueden corregirse modificando el ODT / DOCX:

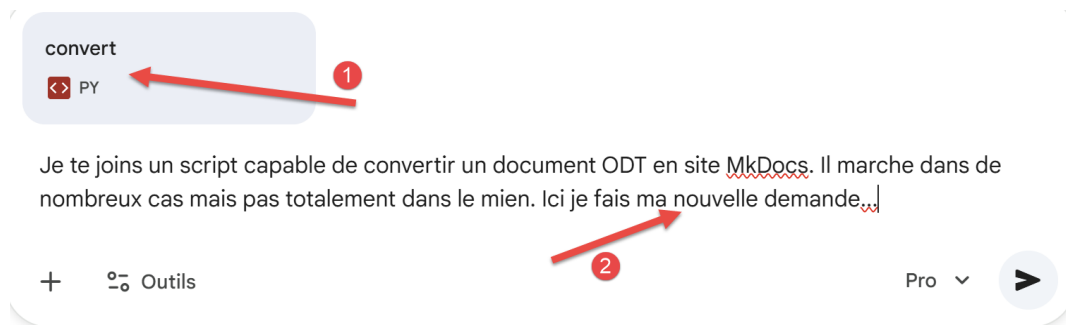
- **los bloques de código deben ir seguidos de una línea en blanco;** de lo contrario, el bloque de código puede mostrarse incorrectamente. El caso detectado es un código seguido inmediatamente de un título sin estar separado de este por una línea en blanco;
- las listas con viñetas no pueden tener borde inferior. **Para que lo tengan, hay que añadir una línea en blanco detrás del último elemento de la lista;**

- **las listas con viñetas deben estar jerarquizadas.** Así, una lista de nivel 2 debe incluirse siempre dentro de una lista de nivel 1; de lo contrario, la lista de nivel 2 se muestra como código;

A medida que se actualicen las versiones del conversor, algunas de estas anomalías desaparecerán. Las tres anteriores se pueden evitar corrigiendo el documento de origen.

11.16. Otros casos

Si su documento utiliza otras particularidades distintas a las mencionadas anteriormente, es muy probable que el conversor Gemini / ChatGPT no las tenga en cuenta. ¿Qué hacer entonces? Puede enviar su nueva solicitud a una de las IA proporcionándole el conversor actual:



- En [1], adjunto el convertidor de este documento;
- En [2], realizo mi nueva solicitud;

Probablemente tendrás que pasar por muchas iteraciones. Cuando una versión sea estable, anota su número para poder volver a dárselo a la IA en caso de regresión. También es recomendable hacer una copia de cada versión estable. Una desventaja importante de ambas IA es que sufren regresiones con bastante facilidad. Basta con pedirle una nueva funcionalidad para que la IA rompa el código que antes funcionaba. De ahí la importancia de anotar el número de versión de las versiones estables para poder volver a ellas. En enero de 2026, me pareció que ChatGPT 5.2 tenía menos tendencia a sufrir regresiones que Gemini 3.

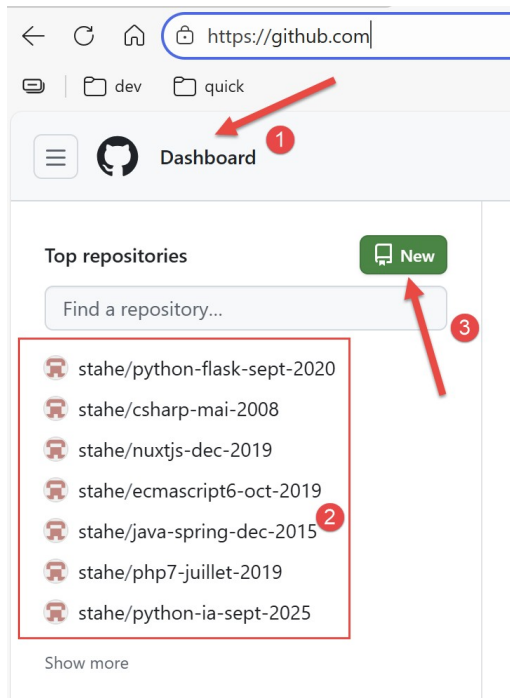
12. Alojamiento del sitio HTML en GitHub

Fue el propio Gemini quien me propuso alojar en GitHub el sitio HTML generado por sus dos scripts:⁴. No sabía que eso fuera posible. GitHub es un sitio que aloja proyectos de desarrollo. Publicar allí cursos de programación parece lo más lógico.

Primero debes tener una cuenta [de GitHub](#). Si es necesario, créala.

Inicie sesión en su cuenta de GitHub:

⁴Nota al pie para GitHub



- En [2], tus repositorios existentes, si los tienes;
- En [3], crea un nuevo repositorio de GitHub;

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name * 1

✔ word-odt-vers-html-janv-2026 is available.

Great repository names are short and memorable. How about [sturdy-telegram](#)?

Description

2

84 / 350 characters

2 Configuration

Choose visibility * Public

Choose who can see and commit to this repository

Add README Off Off

READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore No .gitignore

.gitignore tells git which files not to track. [About ignoring files](#)

Add license No license

Licenses explain how others can use your code. [About licenses](#)

3

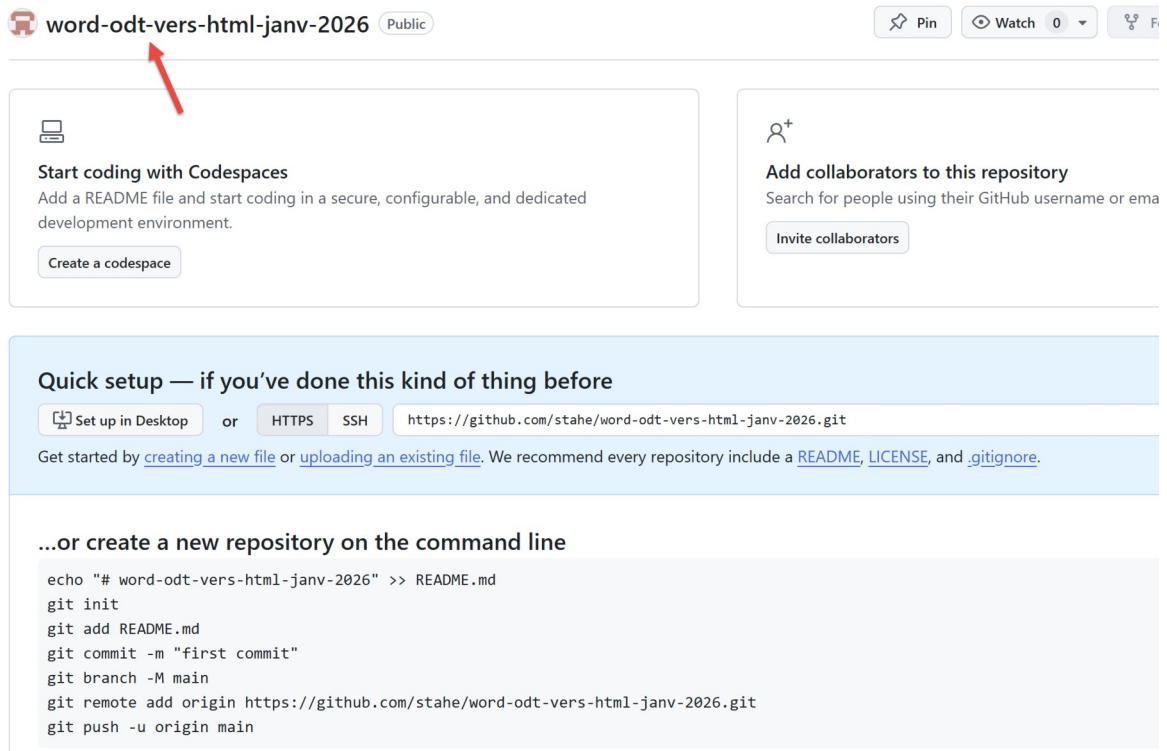
- En [1], utiliza el nombre que has puesto en [config.py]:

```
1. "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
```

- En [2], también puedes poner lo mismo que en [config.py]:

```
1. "site_description": "Convertir un documento Word u ODT en un sitio HTML estático utilizando las IA Gemini 3 y ChatGPT 5.2",
```

- En [3], confirma la creación de tu repositorio GitHub;

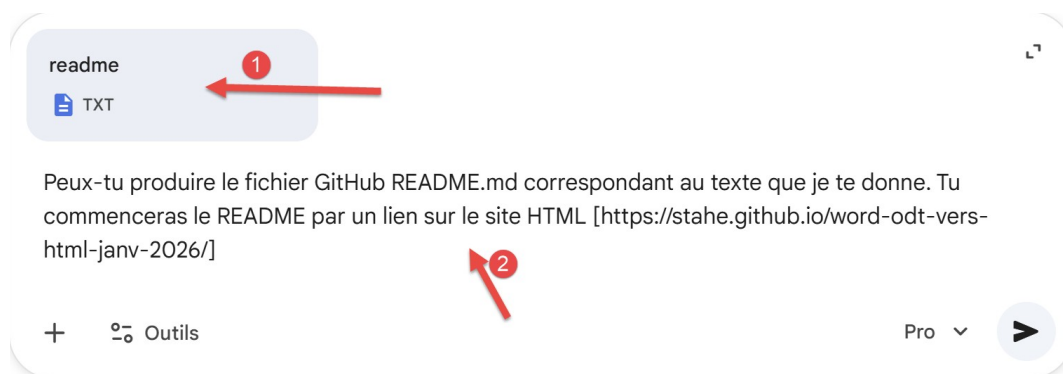


Se recomienda crear un archivo [README.md] para cada repositorio de GitHub, que se mostrará en lugar de la pantalla anterior. Debe estar escrito en Markdown, algo que no es sencillo. Por lo tanto, vamos a escribir un archivo de texto con el contenido que nos gustaría ver en el README. Luego le pediremos a la IA que nos proporcione el README.md correspondiente.

El archivo de texto [readme.txt] será el siguiente:

1. En este artículo nos proponemos poner a disposición del lector un conversor en Python de documentos Word u ODT a un sitio web estático en HTML.
- 2.
3. Este conversor ha sido creado por la IA Gemini 3.
- 4.
5. Cuando el documento ODT es adecuado, el conversor genera un sitio HTML a través de MkDocs que tiene el aspecto profesional de los sitios generados por Pandoc.
- 6.
7. He utilizado Gemini 3 con una suscripción pro.







La línea de comandos de Gemini es la siguiente:



- En [1], el archivo adjunto [readme.txt];
- En [2], la indicación;

Copia la respuesta que te da Gemini en el archivo [README.md] de tu carpeta de trabajo. Esta es la respuesta que me dio Gemini:

```

1. # Convertidor de Word/ODT a sitio HTML (MkDocs)
2.
3.  **[Ver el sitio de demostración generado](https://stahe.github.io/word-odt-vers-html-janv-2026/)**
4.
5. ---
6.
7. ##  Descripción
8.
9. El objetivo de este proyecto es poner a disposición del lector un conversor en Python de documentos Word u ODT a un sitio web HTML estático.
10.
11. Cuando el documento ODT es adecuado, el conversor genera un sitio HTML mediante **MkDocs** que tiene el aspecto profesional de los sitios generados por Pandoc.
12.
13. ##  Contexto de creación
14.
15. Este conversor ha sido creado íntegramente por la IA **Gemini 3** (con una suscripción pro). Es el resultado de sucesivas iteraciones para gestionar con precisión la estructura de los documentos ODT (OpenDocument Text).
16.
17. ##  Funcionalidades
18.
19. El script `convert.py` realiza las siguientes acciones:
20.
21. * **Conversión de ODT a Markdown**: Analiza el archivo `.odt` (XML) para extraer su estructura.
22. * **Gestión de títulos**: Genera automáticamente la tabla de contenidos (TOC) y la navegación lateral.
23. * **Bloques de código**: Detección automática de lenguajes, coloración sintáctica y **gestión precisa de la numeración de líneas** (atributos `start-value`).
24. * **Listas**: Compatibilidad con listas con viñetas y numeradas con la sangría correcta.
25. * **Formato**: Compatibilidad con *negrita*, *cursiva*, *subrayado* y *resaltado* (respetando los colores originales).
26. * **Imágenes**: Extracción e integración automática de las imágenes contenidas en el documento.
27. * **Configuración**: Personalización mediante un archivo `config.py` (pie de página, Google Analytics, etc.).
28.
29. ##  Instalación
30.
31. ### Requisitos previos
32.
33. * Python 3.x
34. * Las siguientes bibliotecas:
35.
36. ```bash
37. pip install odfpv unicode mkdocs mkdocs-material
38. ```
39.
40.
41. ### Estructura del proyecto
42.
43. Asegúrate de tener los siguientes archivos:
44.
45. * `convert.py`: El script de conversión.
46. * `config.py`: Tu archivo de configuración.
47. * `tu-documento.odt`: El documento de origen.
48.
49. ##  Uso
50.

```

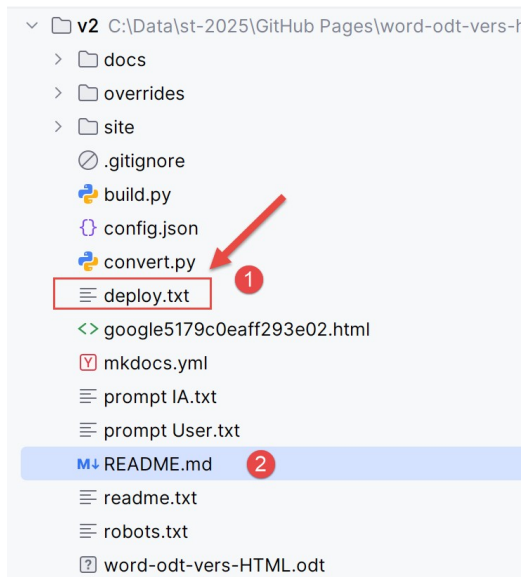
```

51. 1. Conversion
52. Ejecuta el script indicando el archivo ODT de origen y el archivo de configuración:
53. ```bash
54. python convert.py su-documento.odt config.py
55.
56. ```
57.
58.
59. *Esto generará una carpeta `docs/` que contendrá los archivos Markdown y un archivo
60. `mkdocs.yml`.*
61. 2. Vista previa
62. Para ver el sitio en local:
63. ```bash
64. mkdocs serve
65.
66. ```
67.
68. 3. Generación
69. Para compilar el sitio estático (carpeta `site/`):
70. ```bash
71. mkdocs build
72.
73. ```
74.
75.
76.
77. ## ⚙️ Configuración (`config.py`)
78.
79. El archivo `config.py` permite controlar el aspecto del sitio:
80.
81. * mkdocs: Parámetros generales del sitio (título, descripción, tema Material).
82. * pie de página: Código HTML completo para personalizar el pie de página.
83. * código: Reglas de detección de lenguajes para el resaltado sintáctico.
84. * extra: Configuración de Google Analytics (GA4).
85.
86. ## 📄 Licencia
87.
88. Este tutorial escrito por Serge Tahé se pone a disposición del público según los términos
89. de la:
90. *Licencia Creative Commons Reconocimiento - Sin uso comercial - Compartir bajo las mismas
91. condiciones 3.0 no adaptada.*

```

Es un README muy completo el que me ha proporcionado. Esto se debe a que Gemini conoce muy bien este proyecto en el que llevamos trabajando semanas. Seguramente tendrás un [README.md] menos detallado.

Ahora volvamos a nuestra carpeta de trabajo:



- En [2], el archivo README que acabáis de modificar;
- En [1], el archivo [deploy.txt] explica cómo exportar tu sitio HTML a tu repositorio de GitHub;

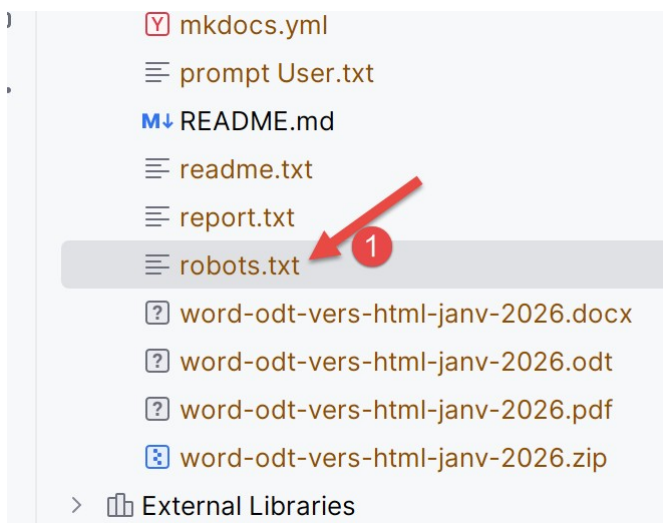
El contenido del archivo [deploy.txt] es el siguiente:

```
1. git init
2. git branch -M main
3. # ¡Comprueba primero .gitignore!
4. git add .
5. # primera confirmación
6. git commit -m "Commit inicial: Código fuente MkDocs"
7. git remote add origin https://github.com/stahe/word-odt-vers-html-janv-2026.git
8. git push -u origin main
9. python -m mkdocs gh-deploy
```

Esta es la secuencia de comandos que exportará tu sitio HTML a tu repositorio de GitHub. Tendrás que modificar la línea 7 con la URL de tu propio repositorio de GitHub, que se encuentra en el archivo [config.py]:

```
1. "repo_url": "https://github.com/stahe/word-odt-vers-html-janv-2026",
```

También debes comprobar otra URL que aparece en el archivo [robots.txt]:



El contenido del archivo [robots.txt] es el siguiente:

```
1. User-agent: *
2. Allow: /
3. Sitemap: https://stahe.github.io/word-odt-vers-html-janv-2026/sitemap.xml
```

En la línea 3, introduce la URL de tu sitio web, la misma que en el archivo [config.py]:

```
1. # URL de publicación del sitio (p. ej.: GitHub Pages)
2. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

El archivo [robots.txt] no se utiliza durante la compilación local del sitio, pero sí se utilizará cuando esté alojado en GitHub.

La secuencia de comandos de [deploy.txt] utiliza un software llamado Git. Debes instalarlo [[Git - Install for Windows](#)].

Una vez hecho esto, comprueba el archivo llamado [.gitignore] en tu carpeta de trabajo. Este le indica a Git qué archivos debe ignorar. Mi archivo [.gitignore] es el siguiente:

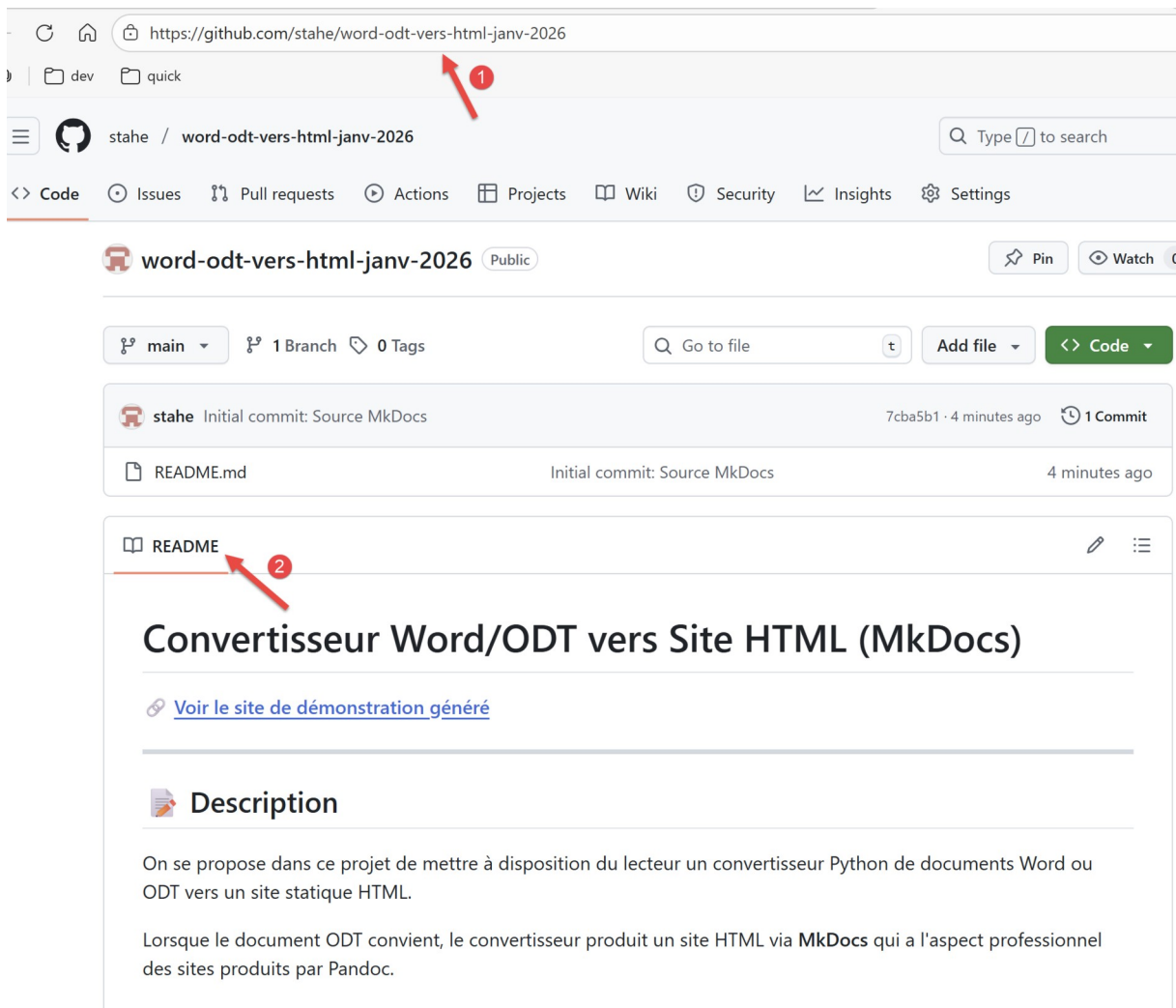
```
1. # 1. Se ignoran TODOS los archivos y carpetas del proyecto
2. *
3.
4. # 3. Se hace una excepción (!) para el único archivo que quieres conservar
5. !README.md
```

Es muy sencillo. Se ignoran todos los archivos (línea 2) excepto el archivo [README.md] (línea 5). GitHub está pensado para alojar proyectos de desarrollo. Por lo general, se exporta todo el proyecto de desarrollo a GitHub. Nosotros solo queremos exportar un sitio HTML, no un proyecto de desarrollo. El único archivo que queremos exportar a nuestro repositorio de GitHub es el archivo [README.md], que explica a los visitantes qué contiene nuestro sitio HTML.

Ahora, en tu terminal, escribe la siguiente secuencia de comandos en el orden indicado por [deploy.txt] hasta llegar al comando 8. No ejecutes el comando 9 por el momento.

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git init
2. Repositorio Git vacío inicializado en C:/Data/st-2025/GitHub Pages/word-odt-vers-html/v2/.git/
3. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git branch -M main
4. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git add .
5. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git commit -m "Commit inicial: Código fuente MkDocs"
6. [main (root-commit) 7cba5b1] Commit inicial: Fuente MkDocs
7. 1 archivo modificado, 89 inserciones(+)
8. crear modo 100644 README.md
9. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git remote add origin
https://github.com/stahe/word-odt-vers-html-janv-2026.git
10. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> git push -u origin main
11. Enumerando objetos: 3, hecho.
12. Contando objetos: 100 % (3/3), hecho.
13. Compresión delta utilizando hasta 8 subprocesos
14. Comprimiendo objetos: 100 % (2/2), hecho.
15. Escribiendo objetos: 100 % (3/3), 1,70 KiB | 1,70 MiB/s, hecho.
16. Total 3 (delta 0), reutilizados 0 (delta 0), paquetes reutilizados 0 (de 0)
17. A https://github.com/stahe/word-odt-vers-html-janv-2026.git
18. * [nueva rama] main -> main
19. Rama «main» configurada para seguir «origin/main».
```

Haz clic con el botón derecho del ratón sobre la URL de la línea 17. Esto te llevará a tu repositorio de GitHub:



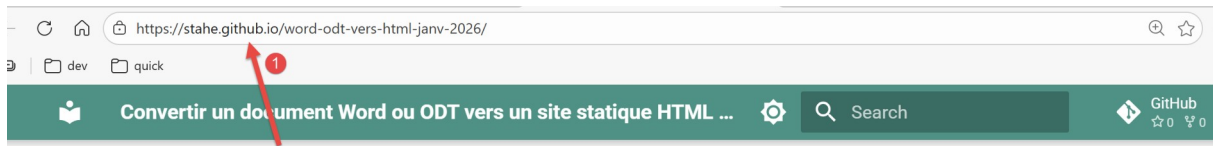
- En [1], la URL de tu repositorio de GitHub;
- En [2], el nuevo README.md;

Ahora pasemos al comando 9 del archivo [deploy.txt]. Es el que exporta el sitio HTML a GitHub:

```
1. PS C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2> python -m mkdocs gh-deploy
2. INFO     - Limpieza del directorio del sitio
3. INFO     - Compilación de la documentación en el directorio: C:\Data\st-2025\GitHub
  Pages\word-odt-vers-html\v2\site
4. INFO     - El archivo de documentación «les-exemples.md» contiene un enlace «#_Les_exemples»,
  pero no existe tal ancla en esta página.
5. INFO     - Documentación generada en 1,79 segundos
6. ADVERTENCIA - Se ha omitido la comprobación de la versión: no se ha especificado ninguna
  versión en la implementación anterior.
7. INFO     - Copiando «C:\Data\st-2025\GitHub Pages\word-odt-vers-html\v2\site» a la rama «gh-
  pages» y subiéndolo a GitHub.
8. Enumerando objetos: 91, hecho.
9. Contando objetos: 100 % (91/91), hecho.
10. Compresión delta utilizando hasta 8 subprocesos
11. Comprimiendo objetos: 100 % (85/85), hecho.
12. Escribiendo objetos: 100 % (91/91), 1,64 MiB | 2,01 MiB/s, hecho.
13. Total 91 (delta 9), reutilizados 0 (delta 0), paquetes reutilizados 0 (de 0)
14. remoto: Resolución de deltas: 100 % (9/9), completado.
15. remoto:
16. remoto: Crea una solicitud de extracción para «gh-pages» en GitHub visitando:
```

```
17. remoto:      https://github.com/stahe/word-odt-vers-html-janv-2026/pull/new/gh-pages
18. remoto:
19. A https://github.com/stahe/word-odt-vers-html-janv-2026.git
20. * [nueva rama]    gh-pages -> gh-pages
21. INFO    - Tu documentación estará disponible en breve en: https://stahe.github.io/word-odt-vers-html-janv-2026/
```

Haz clic con el botón derecho del ratón en la URL de la línea 21. Esto te llevará a tu nuevo sitio HTML en GitHub:



Convertir un document Word ou ODT vers un site statique HTML à l'aide de l'IA Gemini

Accueil

1. Introduction
2. Les exemples
3. Ce qui existe
4. Le prompt initial
5. Créer un environnement de travail
6. Le dossier de travail
7. Le fichier de configuration [config.json]
8. Utilisation du convertisseur
9. Construction du site HTML
10. Examen du site HTML généré
11. Héberger le site HTML sur GitHub

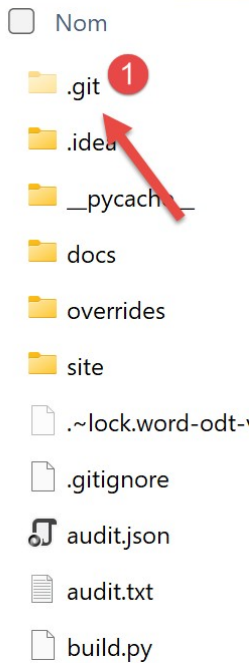
Accueil

Convertir un document Word ou ODT vers un site statique HTML compatible MkDocs avec l'IA Gemini 3

Serge Tahé, janvier 2026

- En [1], vemos que estás mostrando un sitio web en GitHub;

Es muy fácil equivocarse al ejecutar los comandos de [deploy.txt]. Entonces es difícil dar marcha atrás, ya que Git guarda un registro de lo que se ha hecho (mal). Para empezar de cero, visualice la carpeta de trabajo:



- En [1], elimine la carpeta [.git];

Vuelve luego a PyCharm y vuelve a ejecutar la serie de comandos de [deploy.txt].

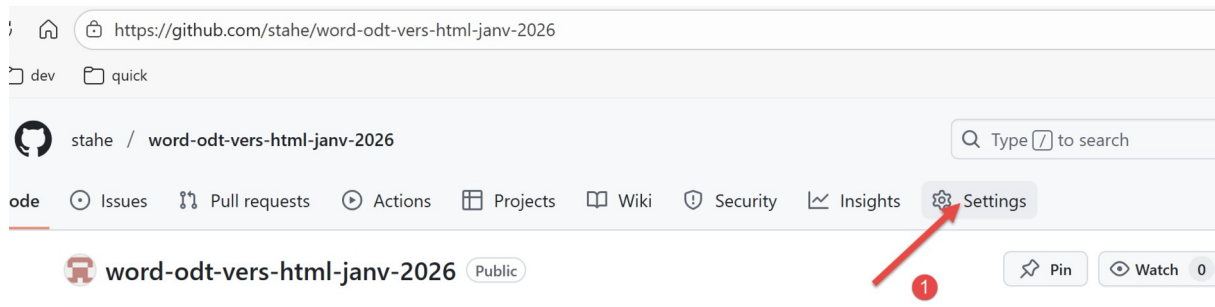
¿Qué hacer si modificas tu documento ODT / DOCX? Haz lo siguiente:

1. vuelve a convertir tu documento ODT/DOCX con [convert];
2. crea el sitio HTML con [build];
3. Exporte el sitio HTML a GitHub con el comando [python -m mkdocs gh-deploy]. Este comando es suficiente siempre que no modifique el archivo [README.md]. Si modifica el archivo [README.md], tendrá que ejecutar algunos comandos más:

```
1. git add README.md
2. # commit
3. git commit -m "modif README"
4. git push -u origin main
5. python -m mkdocs gh-deploy
```

Si solo has modificado el README, solo son necesarios los comandos 1, 3 y 4. El comando 5 es innecesario si ya has desplegado el sitio HTML y no ha cambiado desde entonces.

¿Qué hacer si quieres empezar de cero porque las cosas se han descontrolado? Puedes eliminar tu repositorio de GitHub y volver a realizar todas las operaciones del capítulo12 . La opción para eliminar un repositorio de GitHub está bien escondida:



- En [1], ve a la configuración del repositorio;

Danger Zone

Change repository visibility
This repository is currently public. [Change visibility](#)

Disable branch protection rules
Disable branch protection rules enforcement and APIs [Disable branch protection rules](#)

Transfer ownership
Transfer this repository to another user or to an organization where you have the ability to create repositories. [Transfer](#)

Archive this repository
Mark this repository as archived and read-only. [Archive this repository](#)

Delete this repository
Once you delete a repository, there is no going back. Please be certain. [Delete this repository](#)

Ve hasta la parte inferior de la página de [configuración]. Allí encontrarás el botón para eliminar tu repositorio [2].

13. Realizar un seguimiento del sitio con Google Analytics y Google Search Console

Si alojas tu sitio web en GitHub, probablemente te interese saber el número de visitantes del sitio. Para ello, necesitas una etiqueta de Google Analytics (GA). Le preguntamos a [ChatGPT](#) cómo crear una etiqueta GA:

comment obtient-t-on un marqueur Google Analytics pour un site web ?

Pour un site web, le **marqueur Google Analytics** correspond aujourd'hui à l'**identifiant de mesure GA4** (format `G-XXXXXXXX`). Voici la procédure complète et à jour pour l'obtenir.

1. Créer ou utiliser un compte Google Analytics


1. Accédez à Google Analytics et connectez-vous avec votre compte Google.
2. Si aucun compte n'existe encore :
 - Cliquez sur **Créer un compte**
 - Donnez un nom au compte (ex. : *Entreprise X*)
3. Si un compte existe déjà, sélectionnez-le.

2. Créer une propriété GA4

1. Dans **Administration** (icône engrenage, en bas à gauche),
2. Colonne **Propriété** → **Créer une propriété**,
3. Renseignez :
 - Nom de la propriété
 - Fuseau horaire
 - Devise
4. Validez.

Depuis plusieurs années, **GA4 est le seul type de propriété disponible**. Les anciennes propriétés Universal Analytics ne sont plus utilisables.

3. Créer un flux de données Web

1. Toujours dans **Administration**,
2. Colonne **Propriété** → **Flux de données**,
3. Cliquez sur **Ajouter un flux** → **Web**,
4. Saisissez :
 - L'URL du site (ex. `https://www.monsite.fr`)
 - Un nom de flux
5. Validez. 

- En [1], introduisez la URL de votre site web en GitHub;

4. Récupérer le marqueur (ID de mesure)

Après la création du flux web :

- Un **ID de mesure** est affiché, au format :

```
css Copier le code  
  
G-XXXXXXXXXX 2
```

👉 C'est ce marqueur **Google Analytics**

Il identifie de manière unique votre site dans GA4.


- En [2], anota el código de seguimiento de tu sitio;

Introduce este código de seguimiento en el archivo de configuración [config.py]:

```
1. "extra": {  
2.     "analytics": {  
3.         "provider": "google",  
4.         "property": "G-XXXXXXXXXX"  
5.     }  
6. },
```

En la línea 4, escribe tu código de seguimiento. MkDocs lo insertará automáticamente en todas tus páginas HTML cuando genere el sitio web. Por lo tanto, no es necesario que sigas las demás instrucciones de ChatGPT que muestran cómo instalar el código de seguimiento de GA en tus páginas HTML.

Vamos a pedirle a ChatGPT instrucciones precisas para comprobar que el marcador funciona:

 donne moi des instructions plus précises pour vérifier que le marqueur fonctionne. Donne moi l'URL à utiliser.

Voici une procédure **opérationnelle et vérifiable pas à pas** pour contrôler que votre marqueur GA4 fonctionne, avec l'**URL exacte à utiliser**.

Méthode recommandée : vérification via le rapport *Temps réel*

1. Accéder à Google Analytics

Ouvrez directement l'interface de **Google Analytics** à l'adresse suivante :

[https://analytics.google.com/ ↗](https://analytics.google.com/)

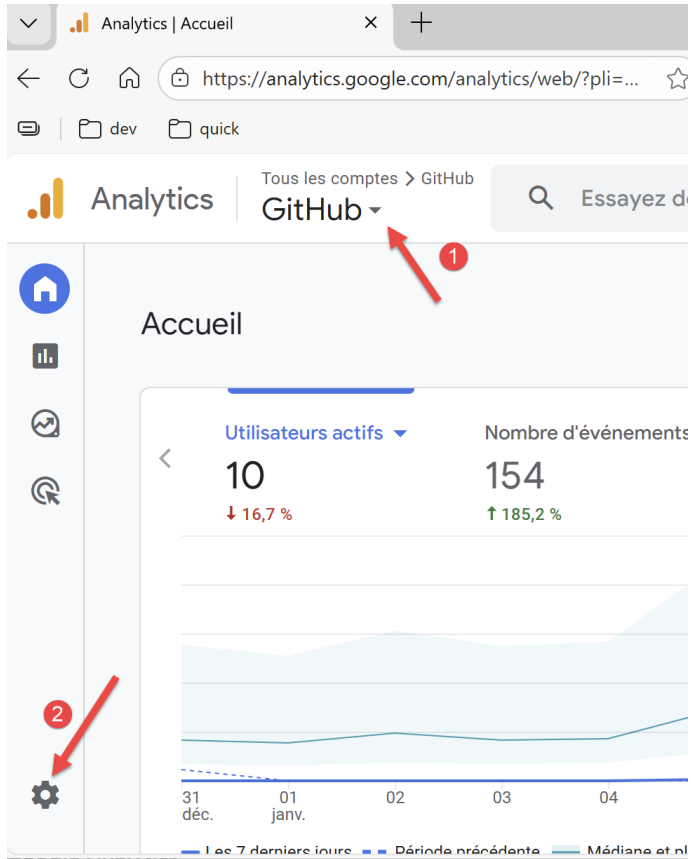
Connectez-vous avec le compte Google qui possède la propriété GA4.

2. Sélectionner la bonne propriété

En haut à gauche de l'interface :

1. Vérifiez que la **propriété GA4** correspondant à votre site est bien sélectionnée.
2. Si nécessaire, changez-la via le sélecteur de compte/propriété.

El paso anterior no es lo suficientemente claro. Aquí tienes algunas capturas de pantalla para ayudarte:



- En [1], sélectionna la cuenta que deseas seguir (en caso de que tengas varias);
- En [2], selecciona el botón de administración;

Analytics Tous les comptes > GitHub Essayez de rechercher "comment créer un entonnoir de conversion"

Administrateur
Ces paramètres s'appliquent à tous les utilisateurs de ce compte et de cette propriété. Pour voir ceux qui ne s'appliquent qu'à vous, accédez à [Mes préférences](#) dans le panneau de navigation de gauche.

PARAMÈTRES DU COMPTE

Compte
Ces paramètres ont une incidence sur votre compte Analytics [Qu'est-ce qu'un compte ?](#)

📄	Détails du compte	🔗	📊	Historique des quotas de l'API Account data	🔗
👤	Gestion des accès au compte	🔗	🗑️	Corbeille	🔗
🕒	Historique des modifications du compte	🔗			

PARAMÈTRES DE LA PROPRIÉTÉ

Propriété
Ces paramètres ont une incidence sur votre propriété [Qu'est-ce qu'une propriété ?](#)

📄	Détails de la propriété	🔗			
👤	Gestion des accès à la propriété	🔗			

Collecte et modification des données
Ces paramètres déterminent la manière dont les données sont collectées et modifiées

📄	Flux de données	3	🔗
📄	Collecte des données		🔗

- En [3], sélectionna [Flujo de datos];

Flux de données

Toutes les vues iOS Android Web Ajouter un flux

🌐	GitHub https://stahe.github.io	4	13112486249	La propriété a reçu du trafic au cours des dernières 48 heures.	>
---	-----------------------------------	---	-------------	---	---

- En [4], haz clic en el flujo de datos;

Détails du flux Web

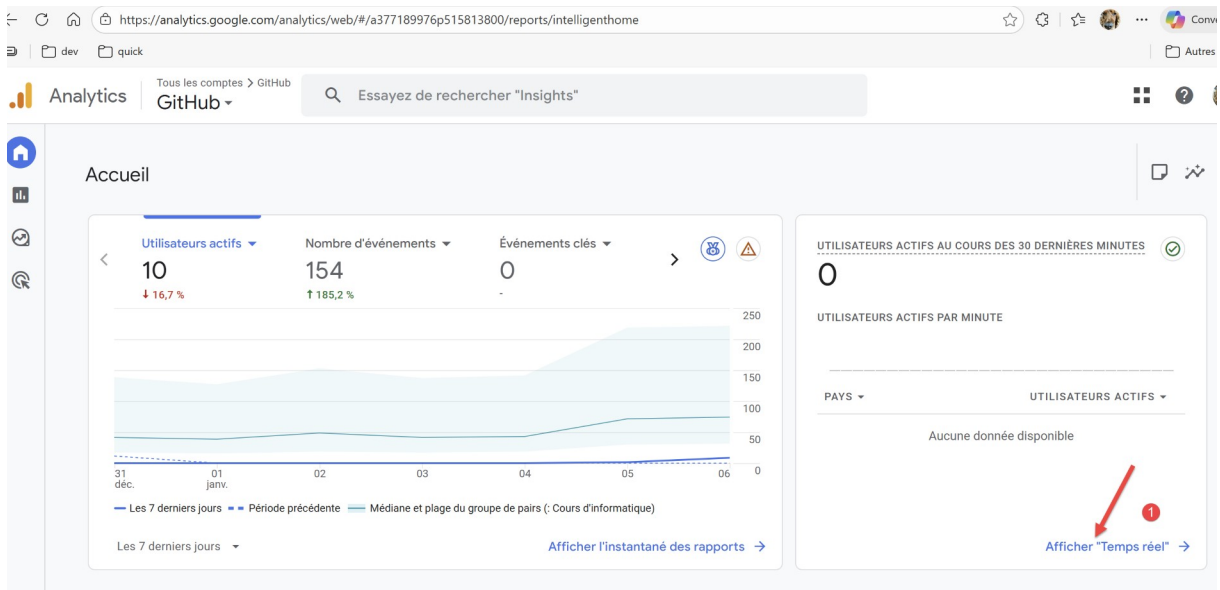
✅ La collecte de données a été active au cours des 48 dernières heures.

Détails du flux ✎

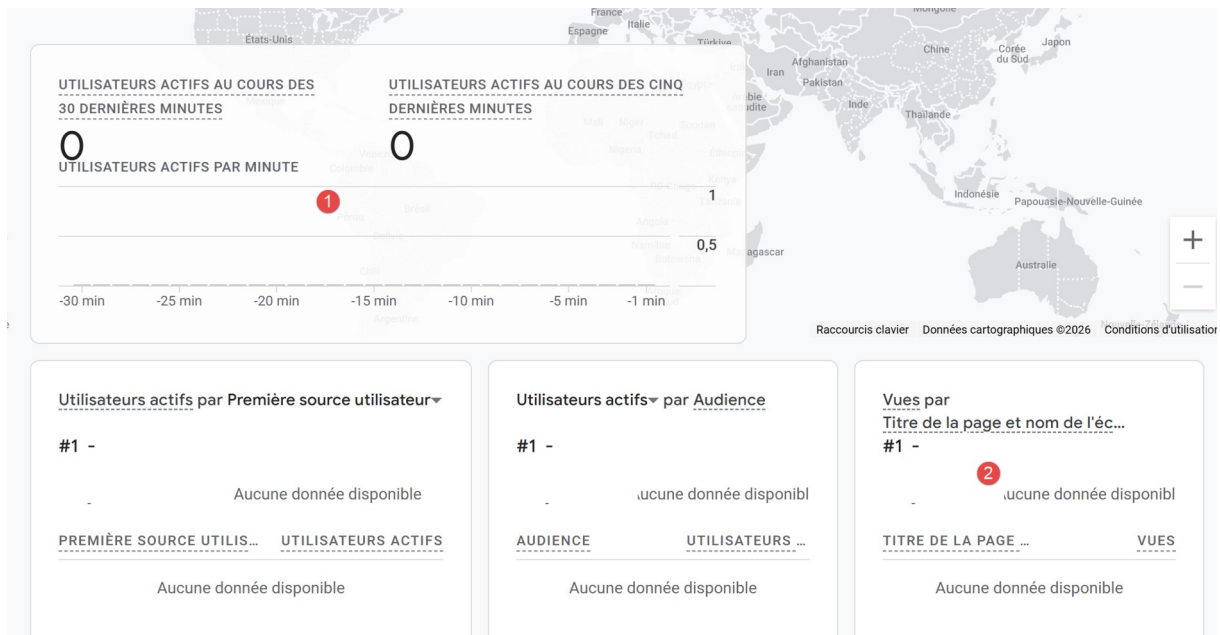
NOM DU FLUX	URL DE FLUX	ID DE FLUX	ID DE MESURE
GitHub	https://stahe.github.io	13112486249	G-EBZC: 5

- En [5], encontrarás el marcador GA4 que has creado;
- En [6], el ID del flujo. Es un concepto diferente del ID de medición;

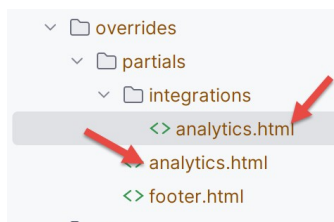
Volvamos a la página de inicio de GA:



- En [1], sollicite la visualización en tiempo real de las visitas;



- En [1], verá a sus visitantes;
- En [2], la página que se ha visitado. Le pedí al conversor Gemini que asociara el nombre del sitio web, en lugar del nombre de la página, al marcador GA4. El archivo [analytics.html] es el que se encarga de este cambio de nombre asociado a GA4:



Ahora podemos continuar con las explicaciones de ChatGPT:

4. Générer une visite test propre

Pour éviter toute ambiguïté :

1. Ouvrez une **fenêtre de navigation privée** (Ctrl+Maj+N ou Ctrl+Maj+P).
2. Saisissez l'URL exacte de votre site (ex. `https://www.monsite.fr`).
3. Naviguez sur une ou deux pages.
4. Attendez 5 à 30 secondes.

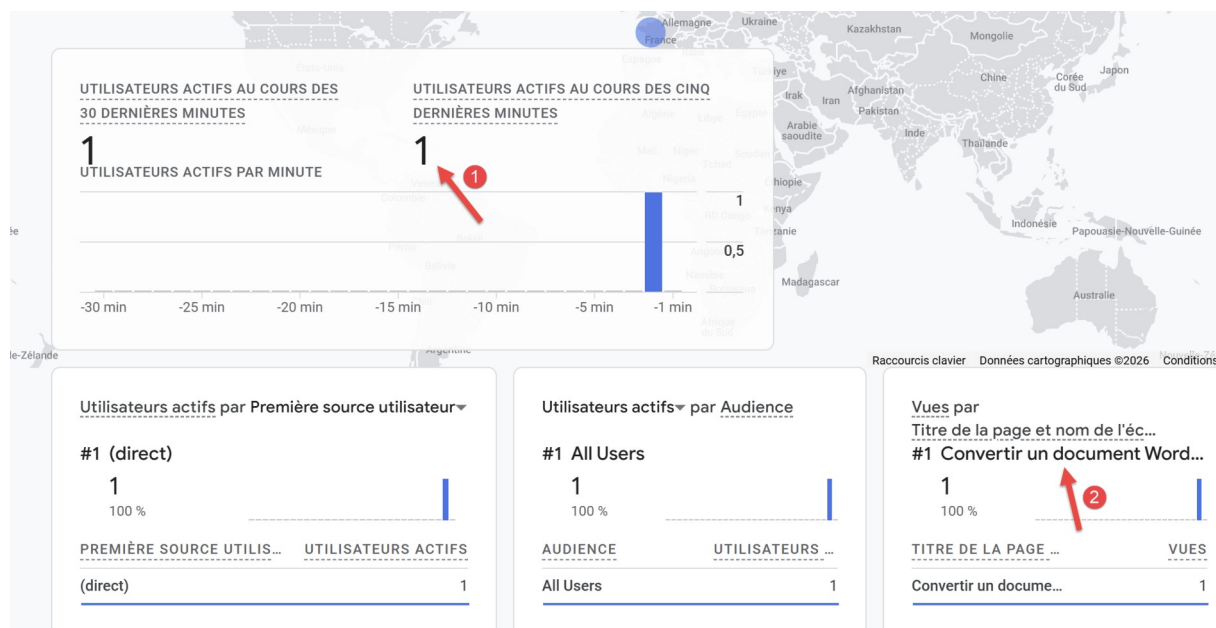
5. Résultat attendu

Dans le rapport *Temps réel*, vous devez voir au minimum :

- 1 utilisateur actif
- Une page consultée
- Une source de trafic (souvent *Direct*)

Si ces éléments apparaissent → le marqueur fonctionne correctement.

He hecho lo que se pide en [4] sin pasar por una ventana privada. Y he obtenido esto:



El marcador GA4 de Google Analytics funciona correctamente.

Ahora vamos a presentar otra herramienta de seguimiento de tu sitio web [Google Search Console]. Preguntemos a ChatGPT para qué sirve esta herramienta:

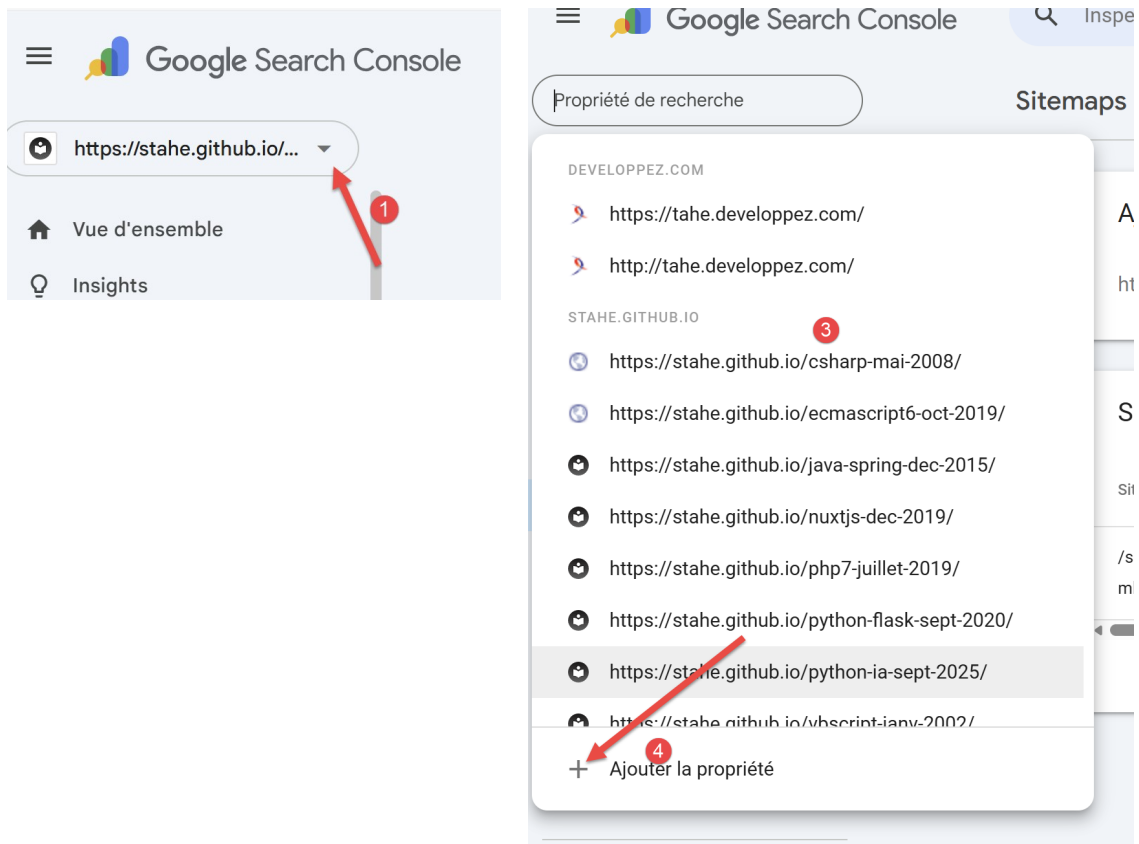
Conclusion

Google Search Console sert à :

- comprendre **comment Google perçoit votre site**,
- améliorer votre **référencement naturel (SEO)**,
- détecter rapidement les **problèmes bloquants**,
- piloter votre visibilité dans Google Search.

Casi nunca utilizo esta herramienta. Solo lo hago para animar al motor de búsqueda de Google a explorarla e indexarla, de modo que los usuarios puedan encontrarla.

La URL de la herramienta [<https://search.google.com/search-console>]:



- En [1], la lista de sitios web que se han declarado a Google;
- En [3], la lista de tus sitios web;
- En [4], añade tu nuevo sitio web;

Sélectionnez le type de propriété

Domaine

- Toutes les URL de tous les sous-domaines (m. www. ...)
- Toutes les URL sur https et http
- Validation DNS obligatoire

example.com
Saisissez un domaine ou sous-domaine

CONTINUER

ou

Préfixe de l'URL

- Seules les URL de l'adresse saisie
- Seules les URL du protocole spécifié
- Différentes méthodes de validation acceptées

sub.io/word-odt-vers-html-janv-2026/
Saisissez une URL

CONTINUER

- En [5], écrivez la URL de votre site web. Se trouve dans le fichier [config.py]:

```
1. "site_url": "https://stahe.github.io/word-odt-vers-html-janv-2026/",
```

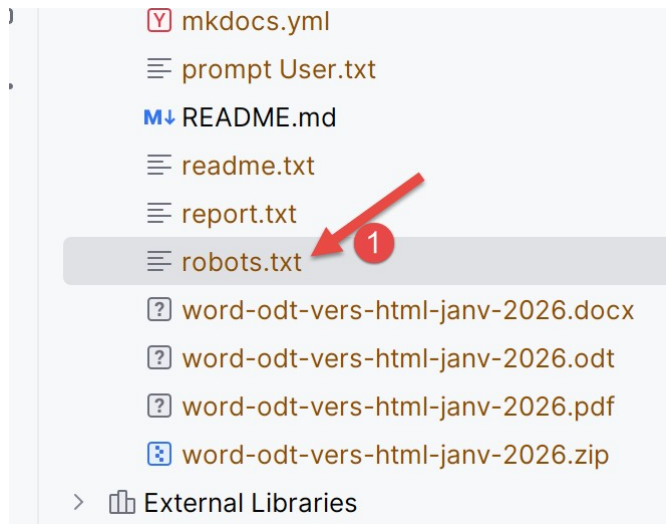
- en [6], passez à l'étape suivante;

Normalement, à cette étape, Google Search Console vous propose un fichier [googlexxxxx.html] que vous devez télécharger. À moi, cela ne m'est pas apparu car j'avais déjà ce fichier sur le site web implémenté. C'est l'un des deux fichiers déclarés dans [config.py]

```
1. "files_to_copy": [  
2.     "google5179xxxxx.html",  
3.     "robots.txt"  
4. ]
```

Placez le fichier [googlexxxxx.html] à la racine de votre dossier de travail et ajoutez son nom à la ligne 2 précédente du fichier [config.py]. Le script [convert] s'occupe de placer les deux fichiers précédents à la racine du site MkDocs que génère. Le script [build] les placera à la racine du site HTML généré.

Après avoir copié le fichier [googlexxxxx.html] à la racine de votre dossier de travail, vérifiez le contenu du fichier [robots.txt]:



El contenido del archivo [robots.txt] es el siguiente:

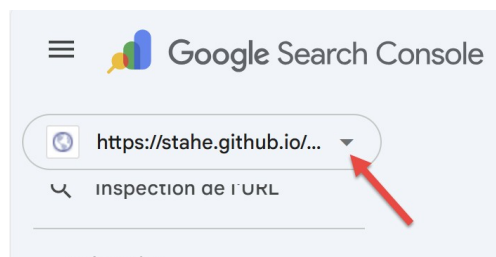
```
1. User-agent: *
2. Allow: /
3. Sitemap: https://stahe.github.io/word-odt-vers-html-janv-2026/sitemap.xml
```

En la línea 3, comprueba que la URL sea la de tu sitio web, la misma que figura en el archivo [config.py].

Vuelva a publicar su sitio web en GitHub escribiendo los tres comandos siguientes en su terminal de Python:

```
1. python .\convert.py .\word-odt-vers-HTML.odt config.py
2. python .\build.py
3. python -m mkdocs gh-deploy
```

Una vez hecho esto, vuelve a [Google Search Console] y selecciona la propiedad que has creado anteriormente.



- En [1], selecciona la opción [Sitemaps];
- En [2], escribe [sitemap.xml];
- En [3], valida la URL;

El archivo [sitemap.xml] es un archivo que Mkdocs ha colocado en la raíz del sitio HTML exportado a GitHub. Puedes comprobar que está ahí escribiendo directamente su URL:

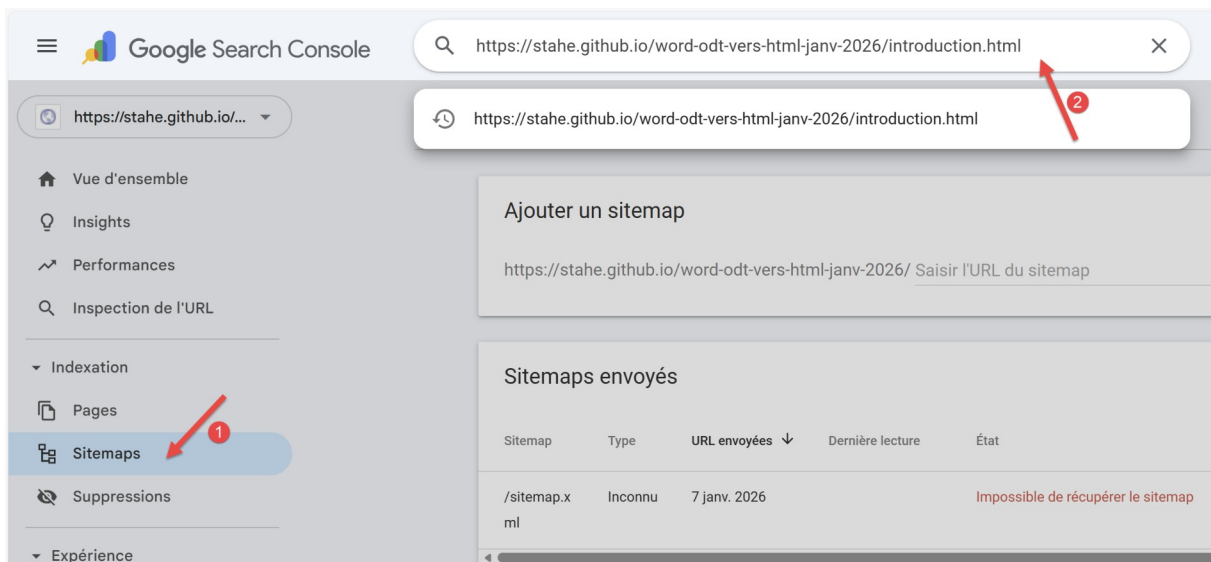
El archivo [sitemap.xml] enumera todas las páginas HTML del sitio.

Si todo va bien, aparecerá la siguiente pantalla:

Google le traitera régulièrement pour chercher des modifications. Si un problème se pose à l'avenir, vous en serez averti.

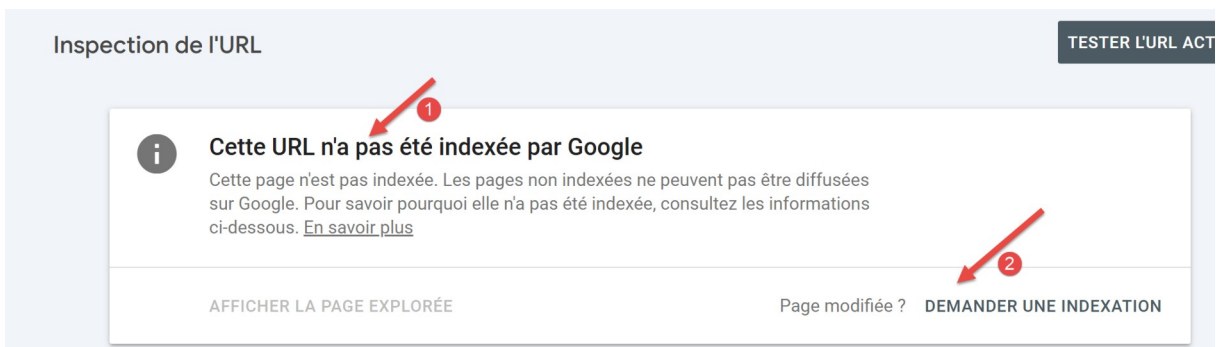
Masquer

Eso es todo. Deberás indexar las páginas del sitio tú mismo:



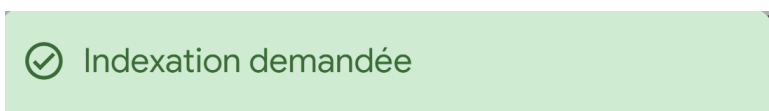
- En [2], écrive la URL de una de las páginas de tu sitio;

Aquí, el resultado es el siguiente:



- En [1], Google indica que la página aún no está indexada;
- En [2], puedes solicitar su indexación;

Google Search Console comprueba entonces si la página solicitada puede indexarse. Si es así, aparece el siguiente mensaje:



Cette URL a été ajoutée à une file d'attente d'exploration prioritaire. Le fait d'envoyer une page à plusieurs reprises ne modifiera pas sa priorité ni sa position dans la file d'attente. [En savoir plus](#)

Masquer

Puede hacer esto con todas las páginas de su sitio web para asegurarse de que está correctamente indexado por el motor de búsqueda de Google.

14. Conclusión

En este documento hemos mostrado un conversor de documentos ODT generados por [LibreOffice](#) o de documentos DOCX de Word a un sitio web HTML estático de muy buena calidad.

En el capítulo [[Ejemplos de este documento](#)] hemos presentado las estructuras que el conversor Gemini / ChatGPT gestiona correctamente.

También hemos mostrado cómo exportar a GitHub los sitios HTML creados por los convertidores ODT y DOCX (véase el capítulo [Alojamiento del sitio HTML en GitHub](#))

Aparte de las estructuras del capítulo de ejemplos, no hay nada garantizado. Es probable que el conversor muestre anomalías. En ese caso, habrá que iniciar de nuevo un chat con Gemini o ChatGPT para resolverlas. Lo más sencillo es proporcionar al conversor actual a la IA junto con su archivo de configuración. A continuación, preguntarle a la IA si entiende los códigos que le hemos dado. Esto la obliga a analizar el código. Siempre responderá que entiende perfectamente el código. Será el momento de pedirle los cambios deseados. Tanto para Word como para LibreOffice, puedes ayudar a la IA con macros. La IA no «ve» el documento ODT o DOCX utilizado. Por lo tanto, le falta información. Así, el conversor DOCX -> HTML no conseguía gestionar correctamente la numeración de los códigos enriquecidos. No sabía leer el número de la primera línea de un bloque de código para convertirlo a HTML. Siempre empezaba el código por 1, aunque el código en el DOCX empezara por 12. Entonces recurrí a una macro. Colocé el cursor del documento en la línea numerada y le pedí a ChatGPT que generara una macro VBA que le proporcionara las características de esa línea, lo que le permitiría generar correctamente la numeración HTML. ChatGPT lo hizo y le di el resultado de su macro. Eso lo desbloqueó: la información recuperada le mostraba que el número de línea se propagaba a través de su estilo y no a través del icono de numeración. Entonces cambió su algoritmo y funcionó. Así que podemos ayudar a la IA a avanzar.

Otra forma más sencilla es modificar el documento. Cuando el conversor genere un error, intenta modificar tu documento para que utilice únicamente las estructuras del capítulo de ejemplos.