DigitAll

Formación en
Competencias
Digitales

3
Creación de contenidos digitales

DigitAll

Formación en
Competencias
Digitales

Creación de contenidos digitales

Nivel C1

ÍNDICE

DigitAll

Creación de
contenidos digitales

3.1
DESARROLLO DE CONTENIDOS

Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

Generación automática de texto a partir de técnicas de IA. GPT un caso práctico.

Generación automática de texto a partir de técnicas de IA. GPT un caso práctico

Introducción

La generación automática de texto ha experimentado avances significativos gracias al desarrollo de técnicas de inteligencia artificial (IA). Estas técnicas permiten a las máquinas comprender y producir texto de manera coherente y contextualmente relevante, en algunos casos, de forma muy similar a como lo haría un humano. En este texto, exploraremos algunas de las técnicas que se han empleado para desarrollar estas IAs y pondremos algunos ejemplos con GPT, probablemente la IA más empleada para generar texto.

Modelos de IAs para la generación automática de texto

Existen diferentes aproximaciones a la hora de conseguir que una IA genere texto, lógicamente, cada una de ellas propone el uso de una serie de herramientas y produce un texto de unas determinadas características. A continuación, enumeramos algunas de ellas y después pasaremos a hablar más específicamente del modelo GPT.

- **Modelos basados en reglas:** estos modelos emplean algoritmos predefinidos y reglas específicas para generar texto. Aunque pueden producir resultados aceptables en escenarios simples, su capacidad para adaptarse a contextos más complejos es limitada. Un tipo de tecnología basada en reglas es ChatScript, que se diseñó inicialmente para crear chatbots.
- **Modelos de lenguaje estadísticos:** estos modelos emplean técnicas estadísticas para analizar patrones y estructuras en grandes conjuntos de datos textuales. A través del aprendizaje automático, estos modelos generan texto basándose en la probabilidad de ocurrencia de palabras y secuencias. Spacy, una librería de software para procesamiento de lenguajes naturales, pertenece a este tipo de modelos.

- **Redes neuronales recurrentes (RNN):** las RNN son modelos de IA que tienen en cuenta la secuencia y el contexto del texto. Estas redes neuronales procesan la información de manera secuencial, lo que les permite capturar dependencias a largo plazo y generar texto coherente y fluido. Por ejemplo, TensorFlow ofrece una amplia gama de funcionalidades para el entrenamiento y despliegue de modelos de RNN, lo que facilita la generación de texto secuencial y contextualmente coherente.

¿Qué es GPT-4?

GPT-4 (*Generative Pre-trained Transformer 4*) es una de las últimas incorporaciones en el campo de la generación automática de texto. Este modelo se basa en la arquitectura *Transformer*, que ha demostrado su eficacia en una amplia gama de tareas de procesamiento del lenguaje natural.

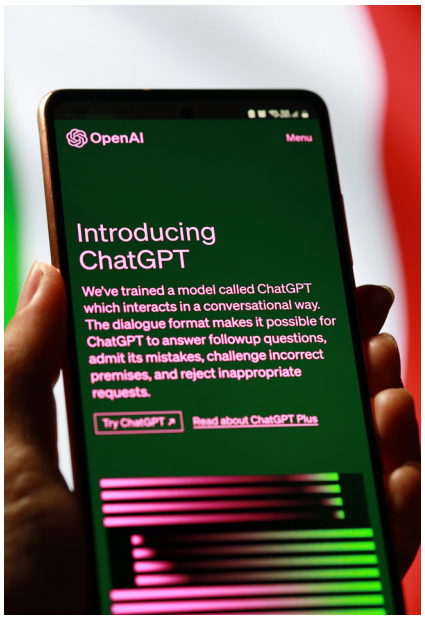
GPT-4 utiliza técnicas de **aprendizaje automático** para predecir la siguiente palabra o frase en función del contexto proporcionado. A través de un entrenamiento intensivo en grandes conjuntos de datos, GPT-4 ha adquirido un conocimiento profundo del lenguaje y puede generar texto coherente y relevante en diversos dominios. A diferencia de las RNN, los modelos basados en *Transformer* utilizan una arquitectura que permite capturar relaciones a largo plazo en el texto sin depender exclusivamente del orden de las palabras.

Dos aclaraciones importantes a la hora de usar GPT-4 de manera efectiva. Por un lado, es importante destacar que es necesario proporcionar al sistema una serie de instrucciones, que estudiaremos en el siguiente nivel, que nos van a permitir optimizar el resultado. Por otro lado, no debemos olvidar que este sistema tiene algunas limitaciones si accedemos a él desde ChatGPT, la más destacada es que las bases de datos que se usaron para su entrenamiento son anteriores a 2021, por lo que su conocimiento del mundo es limitado a partir de ese año. No obstante, en su versión de pago es posible incorporar *plugins* con conectividad a internet o también podemos recurrir al chat de Bing, que está conectado a internet.

¿Qué se puede hacer con GPT-4?

Esta herramienta nos va a proporcionar un sinfín de posibilidades, incluso si solo la usamos como generador de texto. Lo más sencillo es pedirle que nos explique cualquier cosa: un hecho histórico, cómo funciona un dispositivo tecnológico, cómo se prepara una receta, etc. Pero también podemos obtener textos más complejos como canciones o poemas.

En relación con los temas que hemos trabajado en esta competencia, GPT-4 nos puede ayudar a generar mucho contenido multimedia, por ejemplo, le podemos pedir que nos indique cómo aplicar una máscara de capa en una imagen, cómo aplicar efectos sobre un texto o cómo generar una web usando *WordPress*.



Saber más

Plataformas como Coursera, Udemy y edX ofrecen una amplia variedad de cursos relacionados con la inteligencia artificial, el procesamiento del lenguaje natural y la generación de texto. Algunos de los cursos con más éxito son "Natural Language Processing" de la Universidad de Stanford en Coursera o "Deep Learning Specialization" también en Coursera.

Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

Creación y edición de rutas en diseño gráfico

Creación y edición de rutas en diseño gráfico

La creación y edición de rutas es una de las herramientas más potentes y versátiles en el campo del diseño gráfico. Aunque puede parecer compleja a primera vista, con una comprensión clara de sus capacidades y cómo se aplican, puede abrir un mundo de posibilidades para el diseñador gráfico.

¿Qué es una ruta de diseño gráfico y cuál es su utilidad?

Las rutas, también conocidas como "*paths*" en inglés, es una herramienta fundamental en el diseño gráfico digital. Son trazos vectoriales que se pueden manipular para crear formas, líneas y curvas complejas. Las rutas son muy versátiles y pueden ser utilizadas tanto para definir zonas de selección sobre las que aplicar efectos y cambios, como para el dibujo de formas.

Las rutas son esenciales porque definen exactamente qué parte de una imagen se selecciona para aplicar un cambio de color, un efecto de desenfoque, entre otros. Además, las formas basadas en rutas tienen la ventaja de ser totalmente editables y escalables sin perder calidad, lo que las convierte en una herramienta valiosa para el diseño de logotipos, iconos, y cualquier elemento gráfico que pueda necesitar ajustes de tamaño.

Creación y edición

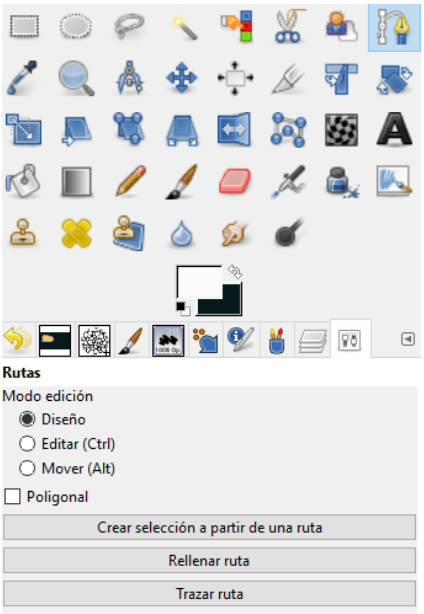
Usaremos el software GIMP para explicar paso a paso la creación y edición de una ruta de manera Inicial.

1 | Abre GIMP y tu imagen

Abre GIMP y carga la imagen con la que deseas trabajar. Puedes hacer esto haciendo clic en "Archivo" en la barra de menú superior, luego selecciona "Abrir" y busca la imagen que quieres usar.

2 | Abre la herramienta de rutas

Haz clic en la herramienta de rutas en el panel de herramientas de GIMP. Esta se representa con un icono que parece un bolígrafo/pluma del que sale un trazo con vectores. Si no encuentras la caja de herramientas, puedes activarla yendo a "Ventanas" en el menú, luego selecciona "Herramientas recién cerradas" y finalmente "Herramientas".



3 | Crea tu ruta

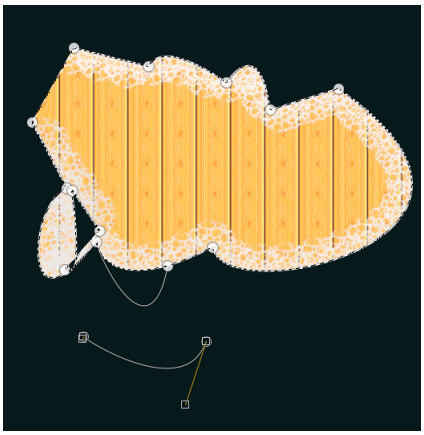
Haz clic en el lugar donde te gustaría que comience tu ruta. Esto creará el primer nodo o ancla. Luego, haz clic en el lugar donde te gustaría que termine tu línea o curva. Esto creará un segundo nodo. Si quieres crear una línea recta, puedes dejarlo así. Pero si quieres una curva, haz clic con el botón Izquierdo del ratón y arrastra el segundo nodo. Verás aparecer dos líneas de color amarillo desde el nodo que puedes arrastrar para ajustar la curva.

4 | Crea formas más complejas

Para crear formas más complejas, simplemente continúa haciendo clic para agregar más nodos. Puedes hacer clic en el primer nodo cuando hayas terminado para cerrar la ruta.

5 | Edita tu ruta

Puedes ajustar la ruta en cualquier momento haciendo clic y arrastrando cualquier nodo. También puedes hacer clic y arrastrar las líneas entre los nodos para ajustar las curvas, moverla, rellenarla con color sólido o patrón y ponerle un trazo o no.



6 | Aplica tu ruta

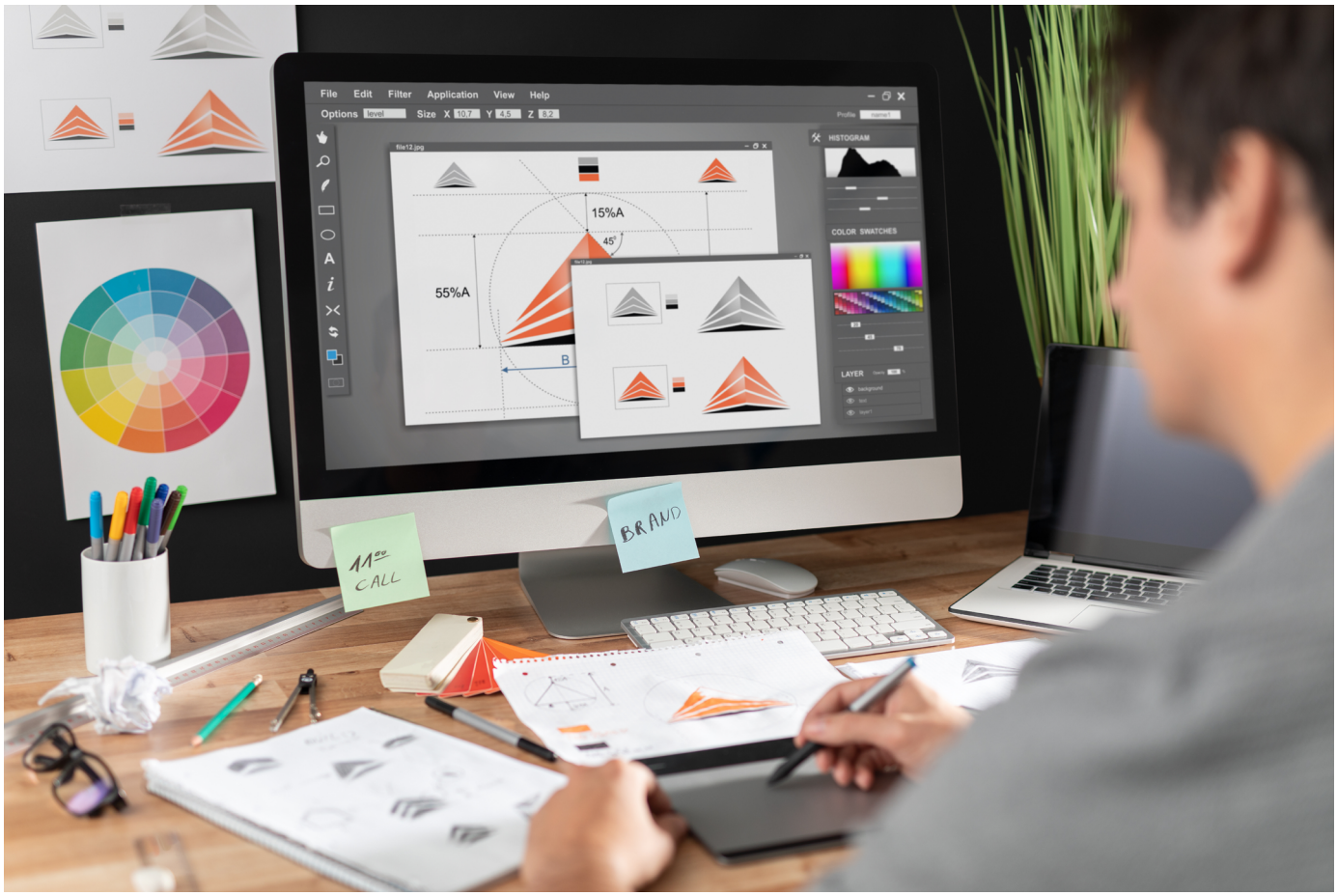
Una vez que estés satisfecho con tu ruta, puedes usarla para varias cosas, como hacer una selección. Para hacerlo, ve al panel de rutas (si no está visible, puedes encontrarlo en "Ventanas" > "Herramientas recién cerradas" > "Rutas"), haz clic derecho en la ruta que acabas de crear y selecciona "De ruta a selección".

Este ha sido un vistazo general a la creación y edición de rutas en diseño gráfico. En niveles superiores, profundizaremos en cómo maximizar el uso de esta herramienta, con ejemplos prácticos y técnicas más detalladas.

Saber más

La herramienta de rutas tiene una curva de aprendizaje lenta, pero una vez que se domina, puede ser increíblemente útil. Al principio, puede ser útil trabajar con rutas simples y prácticas de dibujo básicas para acostumbrarse a cómo se comportan las rutas. Gradualmente, puede experimentar con formas más complejas y selecciones más detalladas.

e.digitall.org.es/ruta-gimp

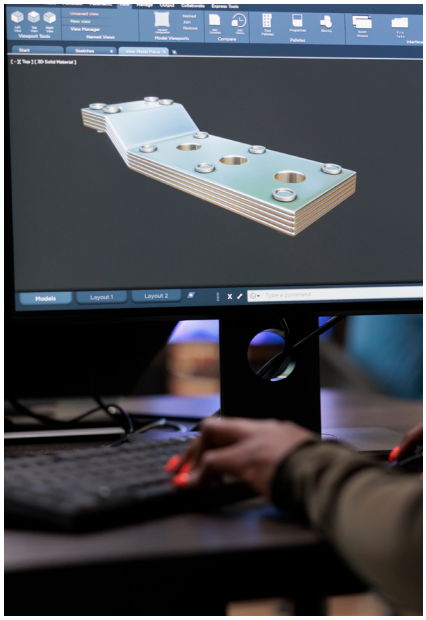


Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

Herramientas para el diseño 3D: libres y privativas

Herramientas para el diseño 3D: libres y privativas



Introducción

El diseño 3D ha revolucionado la forma en que conceptualizamos y creamos objetos virtuales. Gracias a las capacidades avanzadas de modelado, animación y renderizado, el diseño 3D se ha convertido en una herramienta esencial en disciplinas como la arquitectura, el diseño industrial, la medicina, la ingeniería y el arte digital. En este texto, exploraremos la importancia del diseño 3D y las herramientas disponibles, tanto libres como privativas, que permiten materializar nuestras ideas en el mundo virtual.

Herramientas para el diseño 3D

Software CAD

Veremos a continuación que muchas de las herramientas que vamos a mencionar incluyen las letras CAD en su nombre. Cuando hablamos de software CAD (del inglés *Computer-Aided Design*, en español diseño asistido por ordenador) nos referimos a una categoría de software especializado que proporciona las herramientas para crear, modificar y optimizar diseños en formato digital.

Así, el software CAD permite a los diseñadores crear modelos virtuales en 2D o 3D de objetos o sistemas. De este modo, utilizando herramientas y funciones específicas, los usuarios pueden dibujar geometría precisa, aplicar restricciones y dimensiones, realizar simulaciones o analizar propiedades físicas de los diseños.

El software CAD, como los programas que vamos a mencionar a continuación, ofrece una serie de ventajas sobre los métodos de diseño tradicionales. Por un lado, al trabajar en un entorno digital, los diseñadores pueden realizar modificaciones rápidas y precisas en los diseños, explorar diferentes opciones y evaluar su viabilidad antes de la producción física. Por otro lado, dado que esta tecnología sustituye el dibujo manual por un proceso automatizado, los proyectos desarrollados con estos programas se desarrollan en menos tiempo del que necesitaría un diseño clásico.

Herramientas libres

El software libre, como en tantos otros campos, ha democratizado el acceso al diseño 3D al ser gratuito y de código abierto. Probablemente, el software libre de modelado y animación 3D más empleado sea Blender. De hecho, es tan potente que es el preferido de muchos profesionales.

Aunque puede resultar un poco intimidante al principio, Blender es bastante sencillo de usar y tiene una interfaz intuitiva. Su popularidad se debe a varias ventajas, por un lado, es un software multiplataforma que nos va a permitir controlar todo el proceso de creación (modelado, animación, simulación, renderizado…). Por otro lado, su amplia comunidad de usuarios contribuye con actualizaciones constantes, recursos adicionales y tutoriales en línea que pueden ayudarte a aprender a usar Blender.

No obstante, además de Blender, existen otras herramientas libres para hacer diseño 3D. Algunas de las más populares son:

- **FreeCAD:** es un modelador 3D paramétrico de código abierto multiplataforma hecho principalmente para diseñar objetos de la vida real de cualquier tamaño.
- **OpenSCAD:** es un programa CAD de código abierto que crea modelos 3D a partir de scripts o guiones. Una vez nos hemos familiarizado con el lenguaje de programación que usa, a partir de estos guiones podremos crear modelos 3D complejos.
- **3D Builder:** es una aplicación adecuada para usuarios sin experiencia desarrollada por Microsoft que permite ver, crear y personalizar objetos 3D.
- **Tinkercad:** es una herramienta gratuita en línea para crear diseños en 3D. Es fácil de usar y no requiere experiencia previa en diseño 3D. De hecho, se usa en educación ya que se basa en combinar formas simples para generar modelos complejos, como si de un LEGO se tratara.

NOTA

La película de animación Next Gen, de Netflix (2018), fue creada con Blender en su totalidad.

Herramientas privativas

Las herramientas privativas suelen ofrecer mayor cantidad de herramientas y características avanzadas que los programas libres, pero sobre todo, un soporte técnico especializado para resolver los problemas del usuario. Además, por lo general, suelen tener una interfaz más amigable.

A continuación, se describirán algunos de los programas de diseño 3D privativos más populares:

- **Autodesk 3ds Max:** es un software ampliamente utilizado en la industria del diseño y la visualización 3D ya que sus versiones originales datan de los 90. Ofrece una amplia gama de herramientas para modelado, animación, simulación y renderizado. Es popular en la industria del entretenimiento y se ha empleado en películas como Blade: Trinity, Misión Imposible o Hellboy.
- **Autodesk Maya:** es otro programa desarrollado por Autodesk y es muy utilizado en la industria del cine y la animación. Es conocido por su capacidad para crear personajes 3D realistas y efectos especiales impresionantes. De hecho, se ha empleado en películas tan populares como Avatar o algunas secuelas de la saga The Matrix.
- **Cinema 4D:** es un software de diseño 3D desarrollado por Maxon. Es utilizado tanto por profesionales como por principiantes debido a su interfaz intuitiva y fácil de usar. Cinema 4D se utiliza en la industria del cine, la televisión y los medios digitales para crear efectos visuales, animaciones y gráficos en movimiento. Dos ejemplos de su potencial son las películas Pacific Rim y Tron: Legacy.
- **Houdini:** es un software que destaca por su enfoque procesal, cubriendo todas las áreas de la producción 3D. Es responsable de planos de películas tan populares como Frozen o Zootopia.

Estos son solo algunos ejemplos de los programas de diseño 3D privativos más utilizados. Cada uno de ellos tiene sus propias características y se utiliza según necesidades específicas.

Impresión 3D: La materialización de los objetos virtuales

La impresión 3D ha revolucionado aún más el diseño 3D al permitir la materialización física de los objetos virtuales creados. Es decir, podemos usar programas de diseño 3D para crear modelos digitales que se pueden imprimir en 3D. Al utilizar tecnologías de fabricación aditiva, las impresoras 3D pueden transformar los diseños en objetos tangibles de manera rápida y precisa. Esto ha ampliado aún más las posibilidades de aplicación del diseño 3D en diversos campos, como la creación de prototipos, la medicina regenerativa, la arquitectura sostenible o la personalización de productos.

Conclusión

El diseño 3D se ha convertido en una herramienta esencial debido a su capacidad para visualizar ideas de manera eficiente y precisa. Tanto las herramientas libres como las privativas desempeñan un papel importante en este proceso, brindando opciones adaptadas a sus necesidades y recursos disponibles. Con la integración de la impresión 3D, el diseño 3D ha dado un paso más allá al permitir la materialización de los objetos virtuales, abriendo nuevas posibilidades y oportunidades en multitud de ámbitos.

Saber más

Un ejemplo de la importante relación que existe entre el diseño 3D y la impresión 3D de los materiales diseñados la podemos encontrar en los trabajos de Ayúdame 3D. Ayúdame3D es una entidad española que fomenta el valor social de la tecnología a través de programas de concienciación tecnológico-social con el fin de ayudar a colectivos vulnerables de todo el mundo. Gracias a ello crea y entrega brazos impresos en 3D, denominados trésdesis, de manera gratuita a personas con discapacidad.

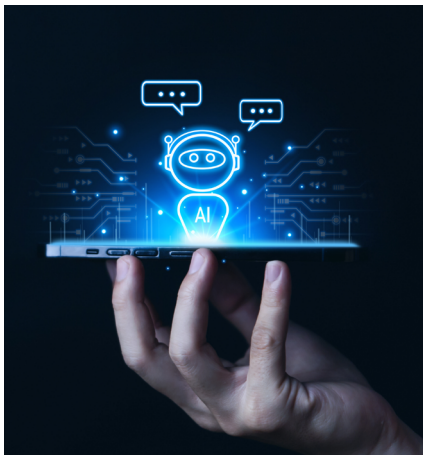
ayudame3d.org

Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

Uso de la inteligencia artificial en la nube para crear nuevos vídeos

Uso de la inteligencia artificial en la nube para crear nuevos vídeos



La creación de vídeos con Inteligencia Artificial (IA) es un campo en constante evolución, emergiendo nuevas técnicas y herramientas todo el tiempo. La ayuda de la IA se está convirtiendo en un componente integral de la producción de medios digitales ya que esta forma de crear y editar vídeos ofrece oportunidades para ahorrar tiempo, aumentar la eficiencia y abrir nuevas posibilidades creativas.

Usos habituales de IA en generación de vídeos

Entre los usos actuales que poseen estas utilidades se encuentran:

1 | Automatización del proceso de edición

Las herramientas de edición de vídeo impulsadas por IA permiten a los usuarios editar vídeos de la misma manera que editarían un documento de texto. Al generar transcripciones automáticas del contenido del vídeo, los usuarios pueden eliminar fácilmente las partes no deseadas, simplemente eliminando el texto correspondiente de la transcripción.

2 | Mejora de la calidad del vídeo

Gracias a estos instrumentos se pueden realizar tareas como eliminar audio, el seguimiento del movimiento, la eliminación del fondo y la detección de silencios. Estas funciones permiten a los usuarios mejorar la calidad de su contenido sin necesidad de tener habilidades técnicas avanzadas.

3 | Creación de contenido generativo

Con ellas se puede crear nuevos contenidos a partir de texto o imágenes, basándose en las indicaciones dadas por el usuario, abriendo nuevas posibilidades para la creación de contenidos visuales únicos.

4 | Personalización y *branding*

Las herramientas de vídeo de IA también pueden ayudar a las empresas a personalizar sus vídeos para reflejar su marca o crear vídeos utilizando avatares digitales. Esto puede ser particularmente útil para la creación rápida de vídeos explicativos, vídeos de capacitación o anuncios, sin la necesidad de contratar actores o personal de cámara.

Herramientas de generación de vídeo

A continuación, presentamos algunos ejemplos de herramientas disponibles que puedes usar de manera gratuita o de pago.

1 | Descript

Te permite generar una transcripción de todo lo que dices junto con un conjunto de escenas, separando la pista de vídeo automáticamente. Puedes resaltar partes de la transcripción que quieres eliminar, y Descript las edita por ti. Además, puedes dividir tu vídeo en escenas y agregar material de archivo de alta calidad a tu proyecto sin salir de la ventana del editor.

DESCRIPT

descript.com

2 | Wondershare Filmora

Esta es una herramienta de edición de vídeo tradicional con utilidades basadas en IA. Algunas de las funcionalidades basadas en IA incluyen recorte inteligente, eliminación del ruido de audio, estiramiento de audio, seguimiento de movimiento, eliminación de fondo y detección de silencio. También puedes hacer cambios en la configuración de tu vídeo al exportar, eligiendo el formato, la resolución, la calidad e incluso la velocidad de fotogramas. Además, tiene agregado ChatGPT, por lo que se puede también generar guiones, subtítulos, descripciones, etc... mediante IA.

WONDERSHARE FILMORA

filmora.wondershare.net/es

3 | Runway

Runway ofrece características de IA para pulir vídeos y generar de texto a vídeo, de imagen a imagen y la posibilidad de entrenar tus propios modelos de IA para la generación de imágenes. También ofrece características como cambio de fondo de pantalla verde (croma), eliminación y reemplazo de objetos y reemplazo de secciones de imágenes.

RUNAWAY

runwayml.com

4 | Peech

Está diseñada para equipos de marketing de contenido. Te permite añadir tu kit de marca para que Peech pueda marcar automáticamente todos tus vídeos. Cada vez que añades un nuevo vídeo, Peech añade estos elementos junto con subtítulos personalizables.

PEECH

peech-ai.com

5 | Synthesia

Esta herramienta te permite generar vídeos utilizando avatares digitales de IA. Soporta múltiples idiomas, pero los avatares pueden no ser completamente creíbles cuando se ven en pantallas grandes.

SYNTHESIA

synthesia.io

CARACTERÍSTICAS DE GENERADORES DE VÍDEO MEDIANTE IA

Nombre	Indicado para	Plataforma	Gratuidad
Descript	Edición de vídeo mediante la edición del guión	Windows, Mac (Web para algunasfunciones)	Sí, 1 hora de transcripción y marca de agua
Wondershare Filmora	Pulir vídeo con herramientas de IA	Windows, Mac, iOS, Android	Sí, marca de agua
Runway	Experimentar con IA generativa	Web	Sí, con 125 créditos, 3 projectos
Peech	Equipos de marketing de contenidos	Web	Sí, 2 vídeos/mes
Synthesia	Uso de avatares digitales	Web	No

Saber más

Existen cursos en línea (Coursera, machine learning A-Z, ...) que proporcionan conocimientos detallados sobre IA, sus herramientas, tecnología y tendencias. Sin embargo, dependiendo de la herramienta específica que quieras aprender, es posible que necesites buscar tutoriales o cursos específicos para esa herramienta. En el siguiente enlace encuentras las mejores alternativas en el momento de la redacción de este documento.

unite.ai/best-ai-vídeo-generators

Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

¿Tu sitio web y sus contenidos digitales son accesibles?

¿Tu sitio web y sus contenidos digitales son accesibles?

Protocolo para evaluar si tu sitio web y tus contenidos son accesibles

Una vez familiarizados con los criterios expuestos en los módulos anteriores, se propone un protocolo o procedimiento a seguir para comprobar si un sitio web cumple con los principales requisitos de accesibilidad (que afecten a texto, imagen, vídeo y sonido).

1 | Verificación de texto:

- Verifica que el tamaño de fuente sea legible y se pueda ajustar según las preferencias del usuario.
- Comprueba que no se utilicen demasiados colores o colores pastel o muy llamativos para transmitir la información, ya que esto puede dificultar la comprensión para personas con discapacidades visuales o daltonismo.
- Examina la estructura de encabezados adecuada (H1, H2, etc.) para facilitar la navegación y la comprensión del contenido.
- Asegúrate de que existe un alto contraste entre el color del texto y el color del fondo.
- Elige una tipografía sencilla que se lea con facilidad, pues la elección de una con excesivos adornos puede dificultar la lectura.

2 | Verificación de imágenes:

- Asegúrate de que todas las imágenes importantes tengan texto alternativo (atributo ALT) descriptivo. Esto permitirá que las personas con discapacidad visual comprendan el contenido de las imágenes a través de lectores de pantalla u otras tecnologías de asistencia.
- Comprueba que los enlaces o botones que utilicen solo imágenes tengan texto alternativo o etiquetas de título explicativas.

3 | Verificación de videos:

- Asegúrate de que los vídeos incluyan subtítulos para personas con discapacidad auditiva. Los subtítulos deben ser precisos, sincronizados y representar adecuadamente el diálogo y los efectos de sonido relevantes.

- Verifica que se proporcione una versión de vídeo con audio descriptivo para personas con discapacidad visual. El audio descriptivo es una narración adicional que describe las acciones visuales importantes que ocurren en el vídeo.
- Comprueba que los controles de reproducción de vídeo sean accesibles y se puedan operar con facilidad utilizando el teclado.
- Valora si es posible incorporar un vídeo superpuesto de una persona o avatar empleando el lenguaje de signos.

4 | Verificación de sonido:

- Asegúrate de que los elementos de audio tengan controles de reproducción que permitan pausar, detener o ajustar el volumen del sonido.
- Comprueba que se proporcione una alternativa de texto o una transcripción para el contenido de audio, permitiendo que las personas con discapacidad auditiva accedan al contenido.

5 | Pruebas de compatibilidad con tecnología de asistencia:

- Realiza pruebas utilizando tecnología de asistencia, como lectores de pantalla (como *JAWS*, *NVDA* o *VoiceOver*) y teclados en lugar de ratones. Esto te permitirá evaluar cómo se presenta y se interactúa con el contenido del sitio web para personas con discapacidades visuales o de movilidad.

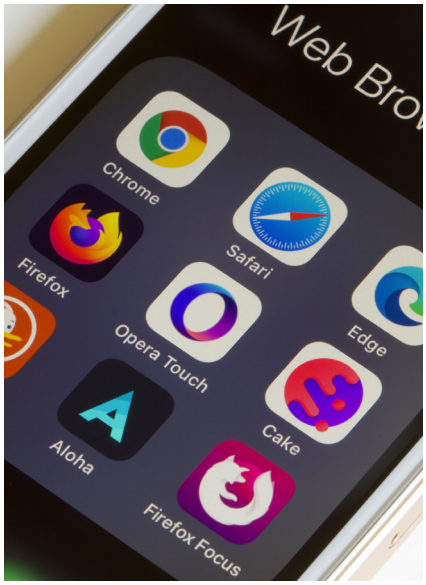
6 | Verificación de navegación y estructura:

- Asegúrate de que la navegación del sitio web sea clara, coherente y fácil de entender. Utiliza etiquetas adecuadas para los elementos de navegación y proporciona indicadores visuales claros de la ubicación actual del usuario en el sitio.
- Comprueba que se utilicen enlaces descriptivos que indiquen claramente el destino del enlace, en lugar de enlaces genéricos como “clic aquí”.
- Verifica que el orden de tabulación sea lógico y que se pueda navegar por el sitio web utilizando solo el teclado, sin depender del uso del ratón.
- Asegúrate de que los formularios sean accesibles, incluyendo etiquetas claras y asociaciones adecuadas entre los campos de entrada y las etiquetas correspondientes.

7 | Verificación de contraste de color:

- Asegúrate de que el contraste de color entre el texto y el fondo sea suficiente para facilitar la legibilidad. Verifica que se cumplan los criterios de contraste establecidos en las pautas de accesibilidad web.
- Comprueba que los elementos interactivos, como los botones o enlaces, tengan un contraste adecuado para facilitar su identificación y uso.

8 | Pruebas de compatibilidad con diferentes navegadores y dispositivos:



- Realiza pruebas de accesibilidad en diferentes navegadores web populares, como Chrome, Firefox, Safari y Edge, para asegurarte de que el sitio web sea accesible en todas las plataformas.
- Verifica la accesibilidad en diferentes dispositivos, como computadoras de escritorio, tablets y dispositivos móviles, para garantizar una experiencia accesible en todas las pantallas.

9 | Validación del código:

- Utiliza herramientas de validación de HTML y CSS para asegurarte de que el código del sitio web cumpla con los estándares y las mejores prácticas recomendadas. Un código limpio y bien estructurado es fundamental para la accesibilidad.

10 | Pruebas de rendimiento:

- Realiza pruebas de rendimiento del sitio web para garantizar una carga rápida y una experiencia fluida. Los retrasos en la carga pueden afectar negativamente a las personas con discapacidades, especialmente a aquellos que utilizan tecnologías de asistencia.

11 | Evaluación de errores de lenguaje y legibilidad:

- Verifica que no haya errores de gramática, ortografía o puntuación que puedan dificultar la comprensión del contenido por parte de las personas con discapacidades cognitivas o de lectura.
- Asegúrate de que la estructura de las oraciones y los párrafos sea clara y concisa para facilitar la lectura y comprensión.

12 | Pruebas de interactividad y funcionalidad:

- Verifica que los elementos interactivos, como formularios, botones y menús desplegables, sean accesibles y se puedan operar fácilmente utilizando solo el teclado.
- Asegúrate de que las animaciones y las actualizaciones dinámicas de contenido sean accesibles y no provoquen distracción o confusión para los usuarios.

13 | Verificación de la etiqueta de idioma:

- Asegúrate de que el sitio web utilice la etiqueta de idioma HTML correctamente. Esto es especialmente importante para las personas que utilizan tecnologías de asistencia, ya que les permite adaptar la pronunciación y el lenguaje del contenido.

14 | Evaluación de la navegación por tabulación:

- Comprueba que el orden de tabulación de los elementos interactivos sea lógico y predecible. Esto garantiza que las personas que utilizan el teclado para navegar puedan acceder y operar correctamente todos los elementos de la página.

15 | Pruebas de contraste en diferentes niveles de zoom:

- Realiza pruebas de contraste de color en diferentes niveles de zoom del navegador. Asegúrate de que el contraste cumpla con los estándares de accesibilidad incluso cuando el usuario aumente o disminuya el nivel de zoom de la página.

16 | Verificación de la legibilidad del contenido en diferentes tamaños de pantalla:

- Evalúa la legibilidad del contenido en diferentes tamaños de pantalla, asegurándote de que el texto sea lo suficientemente grande y legible en dispositivos móviles y pantallas más pequeñas.

17 | Pruebas de teclado:

- Verifica que todos los elementos y acciones interactivas en el sitio web puedan ser accedidos y utilizados a través del teclado. Esto es esencial para las personas que no pueden utilizar un ratón u otros dispositivos de entrada similares.

18 | Pruebas de redimensionamiento de texto:

- Asegúrate de que el contenido del sitio web se pueda redimensionar hasta 200% sin que se pierda información, funcionalidad o diseño. Esto es especialmente importante para las personas con discapacidades visuales que necesitan aumentar el tamaño del texto para leerlo con claridad.

19 | Validación de formularios y mensajes de error:

- Verifica que los formularios en el sitio web incluyan etiquetas claras, mensajes de error descriptivos y validación adecuada para facilitar la interacción y la corrección de errores por parte de los usuarios.

20 | Documentación de hallazgos y mejoras:

- Documenta todos los hallazgos, errores y mejoras identificados durante el proceso de verificación de accesibilidad. Registra los cambios implementados y realiza un seguimiento de las actualizaciones realizadas en el sitio web para mejorar su accesibilidad.

La accesibilidad web es un proceso continuo y dinámico. Mantén un plan de acción para abordar las barreras de accesibilidad identificadas y realiza actualizaciones regulares para mejorar la accesibilidad del sitio web en base a las nuevas pautas y estándares que se publiquen.

Sitios webs que evalúan automáticamente otras webs

Para evaluar si un sitio web cumple con los criterios de accesibilidad necesarios, existen webs que hacen análisis automáticos de otras webs a través de su url. Algunos ejemplos son:

- **WebAIM's WAVE** (*Web Accessibility Evaluation Tool*)**:** WAVE es una herramienta en línea gratuita que proporciona análisis automatizado de accesibilidad web. Simplemente ingresa la URL de un sitio web y WAVE generará un informe detallado que destaca los problemas de accesibilidad detectados.

WEBAIM'S WAVE

wave.webaim.org

- **Axe by Deque Systems:**
  Axe es una suite de herramientas de accesibilidad desarrollada por Deque Systems. Proporciona una extensión para navegadores web que permite analizar sitios web en busca de problemas de accesibilidad. También se ofrece una API para la integración en flujos de trabajo de desarrollo.

AXE BY DEQUE SYSTEMS
deque.com/axe

- **Lighthouse by Google:**
  Lighthouse es una herramienta de código abierto desarrollada por Google que ofrece auditorías automáticas para la calidad de las páginas web. Además de las auditorías generales, Lighthouse incluye un conjunto de auditorías específicas para la accesibilidad web.

LIGHTHOUSE BY GOOGLE
e.digitall.org.es/lighthouse

- **Tenon.io:**
  Tenon.io es una plataforma de análisis de accesibilidad web que permite analizar sitios web en busca de problemas de accesibilidad. Ofrece una API para la integración con flujos de trabajo de desarrollo y también proporciona una interfaz de usuario en línea para realizar análisis rápidos.

TENON.IO
tenon.io

Saber más

Si deseas saber más sobre los estándares internacionales sobre accesibilidad, puedes consultar la W3C Web Accessibility Initiative (WAI).

e.digitall.org.es/wai

Creación de
contenidos digitales

Nivel C1 3.1 Desarrollo de contenidos

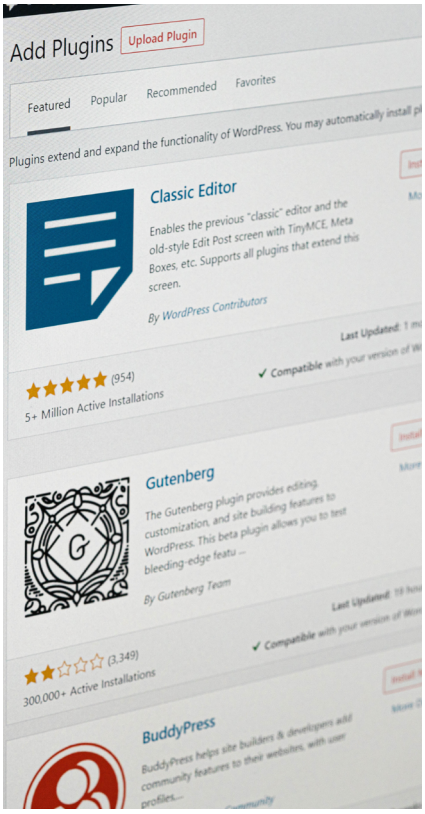
Aumentando la funcionalidad de los gestores de contenidos para la creación de sitios web

Aumentando la funcionalidad de los gestores de contenidos para la creación de sitios web

Introducción

Los gestores de contenidos desempeñan un papel fundamental en la creación de sitios web, permitiendo a los usuarios administrar y publicar su contenido de manera eficiente. Sin embargo, la funcionalidad básica de estos gestores puede no ser suficiente para satisfacer todas las necesidades de un sitio web en particular. Es aquí donde entran en juego los **complementos o *plugins***, herramientas que pueden instalarse en los gestores de contenidos para aumentar su complejidad y capacidad. O, dicho de otra forma, a través de los *plugins* vamos a poder adaptar nuestro sitio web a nuestras necesidades particulares. En este texto, exploraremos la importancia de estos complementos y examinaremos ejemplos de uso con uno de los gestores de contenidos más populares, WordPress.

Aumentando la funcionalidad con *plugins*

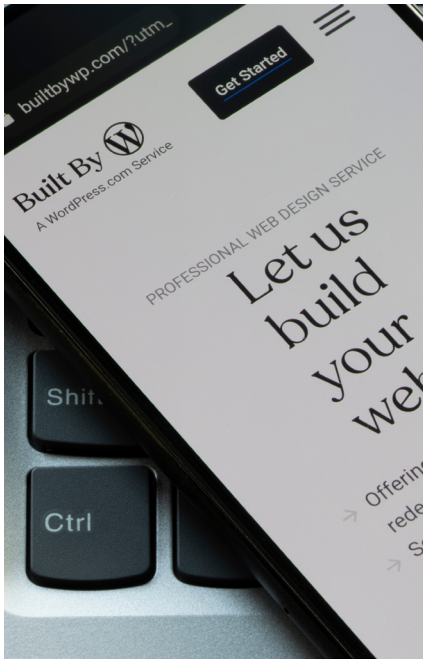


Los complementos o *plugins* son pequeñas aplicaciones que se integran con los gestores de contenidos, agregando nuevas funcionalidades y características. Estas herramientas ofrecen una forma conveniente de **personalizar y ampliar** las capacidades de un gestor de contenidos según las **necesidades** del usuario. Los complementos pueden variar desde la simple incorporación de formularios de contacto hasta la creación de complejos sistemas de comercio electrónico.

Vamos a describir a continuación dos tipos de generales de *plugins*, los de pago y los gratuitos. Por lo general, los *plugins* de pago ofrecen funcionalidades más avanzadas y especializadas. Estas soluciones suelen ser desarrolladas por expertos y empresas que invierten tiempo y recursos en su desarrollo. Los complementos de pago suelen ofrecer un soporte técnico más sólido, actualizaciones regulares y características exclusivas.

Por otro lado, también hay una amplia gama de complementos gratuitos disponibles para los gestores de contenidos. Estos complementos son desarrollados por una comunidad de programadores y diseñadores que contribuyen de forma voluntaria. Aunque los complementos gratuitos pueden no ofrecer tantas funciones como los de pago, pueden ser una excelente opción para quienes tienen un presupuesto limitado o necesitan funcionalidades básicas.

Uso de *plugins* en WordPress



En el universo de WordPress, existen *plugins* para prácticamente cualquier funcionalidad que podamos imaginar. Al igual que comentamos en niveles anteriores cuando describimos las características de WordPress, los *plugins* nos van a permitir esta personalización sin que necesitemos conocimientos de programación.

La **instalación** de *plugins* en WordPress es un proceso sencillo que se puede hacer mediante dos métodos: bien a través del repositorio de WordPress o bien a través de una instalación manual. En este texto describiremos cómo instalar *plugins* a través del repositorio de WordPress.

Para instalar un *plugin* en WordPress seleccionaremos desde nuestro panel de administración la pestaña "*Plugins*" en el menú lateral (Figura 1). Podemos navegar por las diferentes categorías de *plugins*, realizar búsquedas por palabras clave o explorar las listas de los más populares o recomendados. Cada *plugin* en el repositorio tiene su propia página de detalles, donde se proporciona información sobre su funcionalidad, características, requisitos y reseñas de usuarios.

Una vez hemos encontrado el *plugin* a instalar, solo tenemos que pulsar en "Instalar ahora" y automáticamente WordPress descargará e instalará el *plugin* en nuestro sitio web. En la Figura 2 vemos como ejemplo el plugin "Asesor de Cookies RGPD para normativa europea" que nos va a permitir adaptarnos al Reglamento General de Protección de Datos.

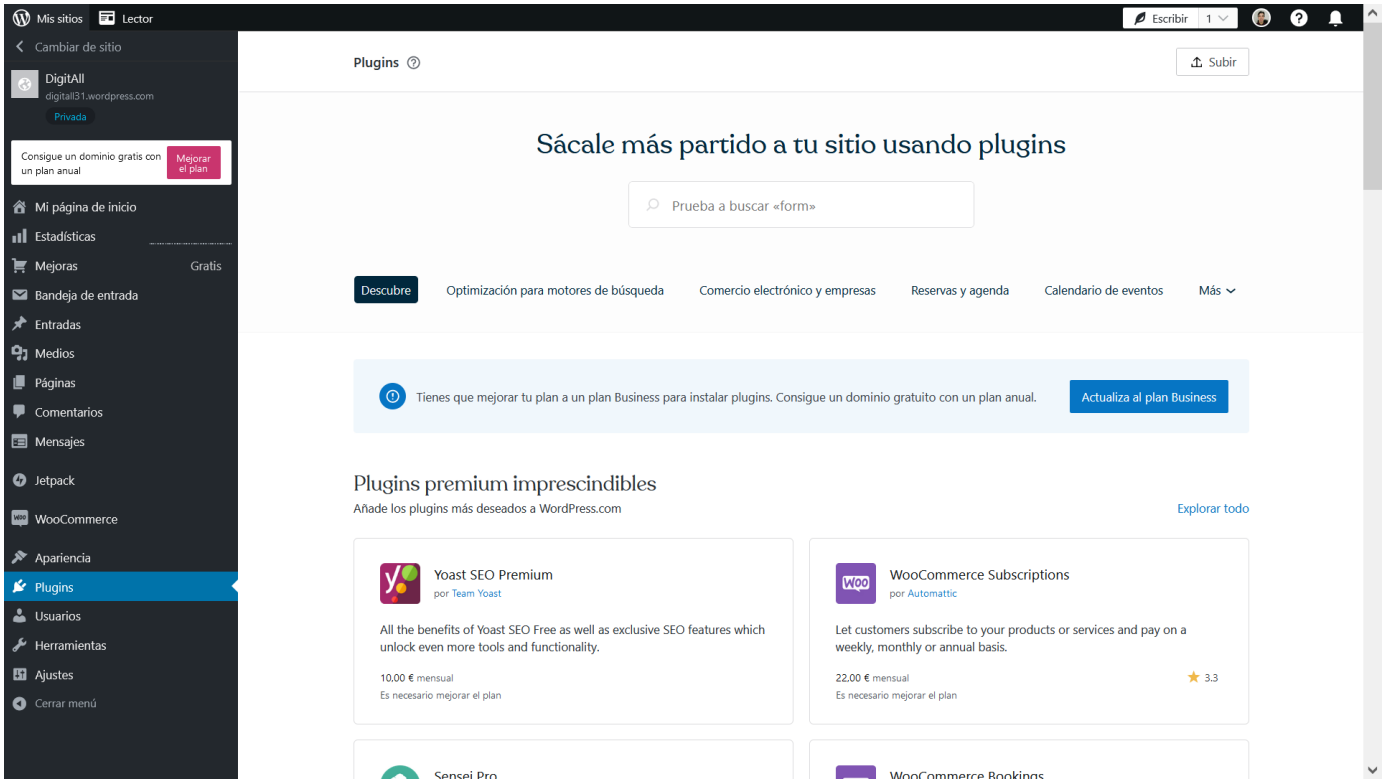


Figura 1. Visión general del menú "*Plugins*" en el panel de administración de WordPress.

Figura 2. Detalle del *plugin* "Asesor de Cookies RGPD para normativa europea". Podemos instalarlo en nuestra web simplemente pulsando el botón de abajo a la derecha y nos ayudará a generar el famoso botón de "Aceptar cookies".

Como vemos en la Figura 1, una de las opciones es buscar por categorías populares. Algunas de ellas son:

- **Optimización para motores de búsqueda (SEO):** para optimizar la visibilidad y el posicionamiento de los sitios web en los motores de búsqueda
- **Comercio electrónico y empresas:** para crear tiendas en línea y gestionar pagos.
- **Seguridad:** para proteger el sitio web contra ataques y amenazas.
- **Formularios:** para crear formularios de contacto personalizados y encuestas.
- **Galerías de imágenes:** para mostrar imágenes de forma atractiva en el sitio web.

Plugins populares en WordPress

Algunos de los *plugins* gratuitos más populares para WordPress son Yoast SEO, Contact Form 7, Elementor Website Builder, WooCommerce o Akismet Anti-Spam. Por otro lado, algunos de los *plugins* de pago más populares para WordPress son las versiones de pago de Yoast y los *plugins* para gestionar un comercio electrónico basado en WooCommerce.

Conclusión

Los gestores de contenidos son herramientas poderosas para la creación de sitios web, pero a veces su funcionalidad básica no es suficiente para satisfacer todas las necesidades. Es aquí donde los *plugins* entran en juego, permitiendo a los usuarios ampliar las capacidades de los gestores de contenidos de manera fácil y personalizada. Ya sea optando por complementos de pago o gratuitos, la variedad de opciones disponibles en el universo de WordPress permite a los usuarios personalizar sus sitios web y mejorar la experiencia tanto para ellos como para los visitantes. Explorar y experimentar con *plugins* es una forma efectiva de aumentar la funcionalidad de los gestores de contenidos y llevar los sitios web al siguiente nivel.

Saber más

Puedes aprender mucho sobre cómo funcionan *plugins* específicos en el directorio de *plugins* de WordPress en español. Éste es un sitio web que contiene una gran cantidad de *plugins* gratuitos y algunos de código abierto para WordPress. Para cada *plugin* puedes ver tanto sus características, las valoraciones de otros usuarios, la compatibilidad con la versión actual de WordPress instalada, así como el número de instalaciones activas. Dado que el repositorio de WordPress es mantenido y administrado por el equipo de WordPress.org, todos los *plugins* incluidos en el repositorio son revisados y aprobados por el equipo de WordPress antes de ser publicados, lo que ayuda a garantizar la calidad y seguridad.

es.wordpress.org/plugins

DigitAll

Creación de contenidos digitales

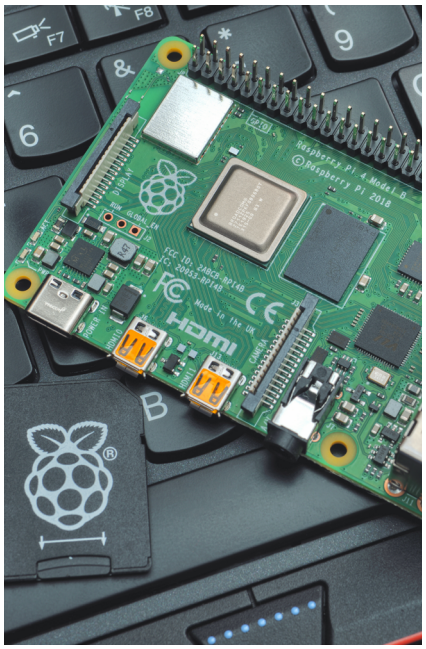
3.2 INTEGRACIÓN Y REELABORACIÓN DE CONTENIDO DIGITAL

Creación de
contenidos digitales

Nivel C1 3.2 Integración y reelaboración de contenido digital

Tipos de sensores y actuadores

Tipos de sensores y actuadores



Sensores

La construcción de un sistema software-hardware requiere el uso de microcontroladores entre los que destacan Arduino y el ordenador Raspberry Pi. La elección de un tipo de microcontrolador depende del proyecto electrónico a construir. La ventaja de Arduino es que ejecuta inmediatamente la tarea. Raspberry Pi, por disponer de más potencia de cómputo y un sistema operativo, es más adecuado para ser usado como un microrordenador funcional portátil. Ambos tipos de placas permiten la integración en el sistema de otros dispositivos llamados **sensores** y **actuadores**. La figura nº1 muestra un ejemplo de estos dispositivos electrónicos.

Los sensores son dispositivos que son capaces de medir magnitudes físicas o químicas, denominadas **variables de instrumentación**, y transformarlas en **variables eléctricas** mediante un **transductor**.

La conversión de la variable de instrumentación en variable eléctrica se puede realizar mediante una resistencia térmica (por ejemplo, la temperatura), un fotodiodo (intensidad de luz) o mediante un cristal piezoeléctrico (presión) que adquiere polaridad cuando es sometido a una presión mecánica. Las variables eléctricas pueden ser resistencia, capacidad, voltaje, corriente, etc. Las salidas del sensor pueden ser analógicas, digitales o conectadas a través de buses de comunicación.

Los sensores realizan mediciones y este proceso está caracterizado por varios parámetros que se describen a continuación. El **dominio** es el conjunto de posibles valores que el dispositivo es capaz de medir. La **resolución** es la mínima variación de la variable de instrumentación que el sensor puede detectar. La **precisión** es el máximo error que puede cometer. Las **derivas** son otras magnitudes, además de la deseada, que afectan a las medidas resultantes. La desviación de cero (*offset*) es el valor de la variable eléctrica cuando la variable de instrumentación es nula.

A continuación, se enumeran los principales tipos de sensores:

Tipo	Descripción	Características
Desplazamiento y distancia	Miden desplazamientos lineales, posiciones o distancias lineales. El rango de estos dispositivos va desde milímetros a cientos de metros.	Existen con contacto físico (con rozamiento) y sin contacto que se basan en láser o ultrasonidos.
Aceleración	Miden la aceleración o la vibración desde unos pocos g's hasta miles de g's.	Existen de diversos tipos como piezoresitivos, piezoeléctricos y acelerómetros capacitivios.
Giróscopos y referencias inerciales	Permiten medir o mantener la orientación en el espacio cuando se produce el movimiento de un objeto. Las referencias inerciales son sistemas de medida de posición y velocidad angular. Este tipo de sensores se denominan sensores giroscópicos.	Integran giróscopos y acelerómetros triaxiales.
Sensores de par y torsión	Miden la fuerza de torsión a la que se somete un eje. Se emplean en el estudio de elementos de rotación.	Son estáticos o dinámicos. El transductor transforma la torsión en una variación de voltaje.
Presión y caudal	Transforman una fuerza por unidad de superficie en un voltaje equivalente. Destacan para medir la presión de líquidos como agua, aceite, o líquidos de frenos. Los de caudal miden la cantidad de material (en peso o volumen) que circula por un canal por unidad de tiempo.	Tienen aplicaciones en la construcción de interruptores de presión o monitorización de niveles.
Parámetros ambientales	Permiten medir la temperatura, humedad, presión barométrica, CO2, NOx, PM10, PM2.5, etc.	Tienen aplicaciones en agricultura para el control inteligente del riego o para el control de emisiones en ciudades.
Sensores de presencia	Detectan la presencia de un objeto y su cercanía con el punto de referencia. Tienen aplicaciones en vehículos no tripulados o vigilancia autónoma.	Constan de un par emisor/receptor. Se usan en aplicaciones para automáticamente abrir/cerrar puertas, encender/apagar luces, encendido automático de pantallas.
Sensores acústicos	Tiene aplicaciones en la detección de comandos de voz o en la vigilancia inteligente.	Transforman ondas acústicas en impulsos eléctricos.
Sensor de luz	Tienen una superficie sensible a la luz, produciendo valores diferentes de resistencia.	Se pueden utilizar para medir el nivel de luz y poder detectar si es de día o de noche o corregir la iluminación.

La Figura 1 muestra un ejemplo de un dispositivo Raspberry Pi conectado a un pequeño circuito eléctrico que integra dos sensores medioambientales que miden gases GPL y monóxido de carbono, respectivamente. La Figura 2 muestra el resultado real.

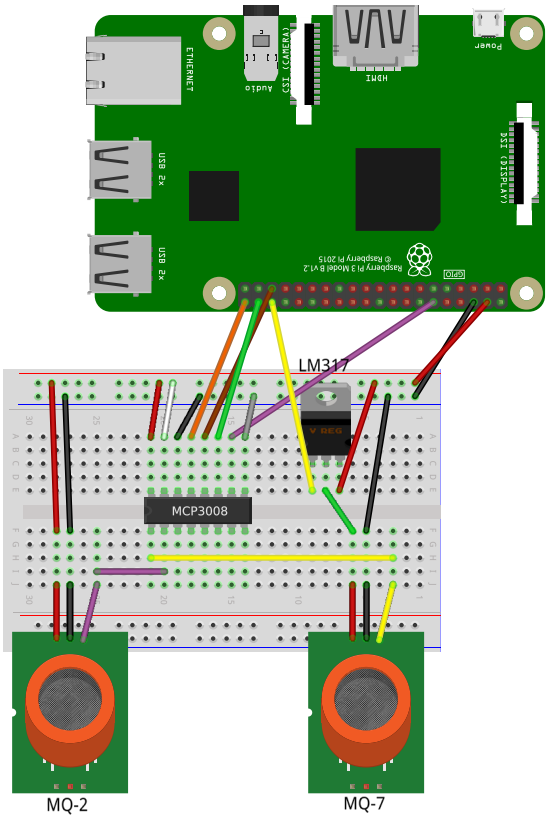


Figura 1. Ejemplo de dispositivo Raspberry Pi conectado a dos sensores de parámetros ambientales.

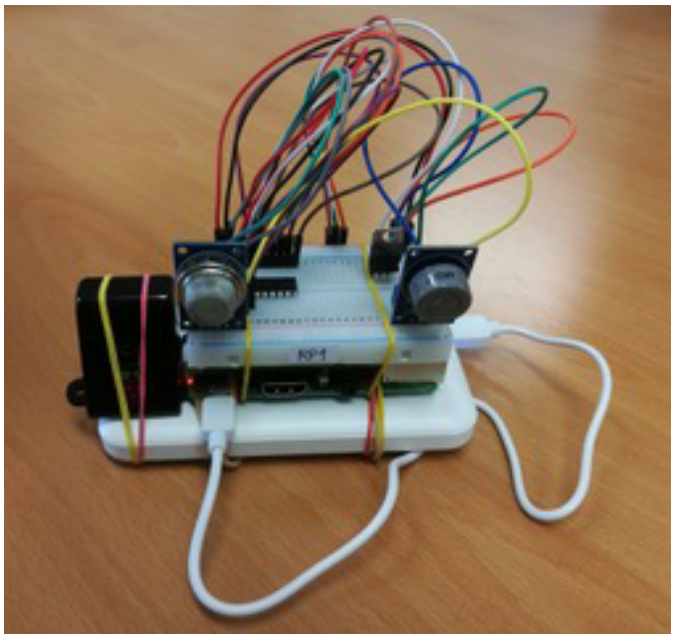


Figura 2. Resultado real del esquema mostrado en la Figura 1..

Actuadores

Los **actuadores** son dispositivos que transforman energía hidráulica, neumática o eléctrica en la activación de un proceso que genera una acción sobre un elemento externo al sistema.

Por ejemplo, un actuador puede recibir una orden de un microcontrolador, como Arduino o Raspberry Pi, y poner en funcionamiento el motor de una bomba.

La siguiente tabla resume los principales tipos de actuadores:

Tipo	Descripción	Ejemplos de uso
Mecánicos	Transforman la acción rotativa de una entrada en un movimiento lineal.	Gatos mecánicos. Se emplean para la colocación exacta y repetible de objetos.
Neumáticos	Transforman aire comprimido en una fuerza motriz que produce una acción mecánica.	Pinzas neumáticas que permiten recoger objetos de un lugar y dejarlos en otro sitio.
Hidráulicos	Emplean una fuerza hidráulica para empujar y obtener una fuerza externa. Se usan cuando se necesita una potencia elevada.	Se emplean en grúas y en excavadoras, así como para la automatización de válvulas.
Eléctricos	La fuente de la fuerza es la energía eléctrica.	Válvulas multi-vueltas, motores, o relés.
Térmicos	Obtienen la fuerza a través del movimiento y la amplificación por la expansión térmica.	Automatización de sistemas de calefacción y refrigeración.

Los sensores van conectados a las entradas de las placas de Arduino o de la Raspberri Pi mientras que los actuadores se conectan en las salidas. Además, se pueden conectar los **periféricos** que son dispositivos hardware que permiten almacenar o intercambiar información con el exterior del sistema. Ejemplos de periféricos son las pantallas LCD, teclados, memorias externas, impresoras, cámaras, micrófonos, displays numéricos, etc.

El documento "***Paradigmas de la programación. Visión general***" (competencia 3.4 del Nivel C1) es clave para el control de sensores y actuadores en diferentes campos, como la robótica y la automatización industrial. Mediante el uso de lenguajes de programación, se pueden crear programas que interactúen con los sensores para recopilar información del entorno y enviarla a los actuadores para realizar acciones específicas. Además, la información capturada por los sensores se puede utilizar para generar contenido multimedia, como imágenes y videos, que proporcionen una representación visual de los datos recopilados. Esto es útil en aplicaciones de monitoreo ambiental o seguridad.

PARADIGMAS DE LA PROGRAMACIÓN. VISIÓN GENERAL

Documento referenciado:
A3C34C1D01

⚠ ATENCIÓN

A la hora de elegir un sensor, debemos revisar detenidamente las características y elegir uno que sea compatible con nuestra placa (tensión/voltaje e intensidad de corriente/amperaje) y que su dominio sea acorde a las magnitudes de nuestro sistema. Otro parámetro que influye en la elección es su facilidad de uso, disponibilidad de librerías y documentación

Creación de
contenidos digitales

Nivel C1 3.2 Integración y reelaboración de contenido digital

Autenticidad y verificación de contenidos digitales

Autenticidad y verificación de contenidos digitales

En la actualidad, el mundo vive en la denominada **sociedad de la información**. En ella las personas pueden crear, modificar, acceder y compartir la información y el conocimiento para mejorar sus procesos productivos y, en definitiva, su calidad de vida.

La **sociedad de la información** es aquella en la que las **Tecnologías de la Información y la Comunicación (TIC)** permiten el acceso, la creación y la distribución de la información en las actividades económicas, sociales y culturales.

Paradójicamente, una de las consecuencias de la sociedad de la información es la **desinformación**. El acceso a la información no implica necesariamente estar informado, especialmente cuando la información es **ingente, heterogénea** (cualquier usuario puede producir contenido) y, a menudo, **contradictoria**. En este sentido, la sociedad de la información plantea a los ciudadanos el reto de filtrar y discriminar con **sentido crítico** la información útil y veraz de aquella que no lo es.

El **análisis forense informático** es el área de las ciencias de la computación que engloba el conjunto de técnicas que permiten **identificar, preservar, recuperar y analizar** medios digitales (hardware y software).

Existen multitud de herramientas de análisis forense informático encaminadas a la toma de decisiones, así como a la presentación de hechos y evidencias respecto a la información digital. Algunas de ellas son ***FotoForensics*** (***fotoforensics.com***), que permite realizar diferentes análisis sobre una imagen proveniente de una URL o del dispositivo local para comprobar si ha sido o no modificada, ***AmpedAuthenticate*** (***ampedsoftware.com***) en la que, tomando una imagen como referencia, es posible realizar diferentes análisis y revelar el historial de procesamiento de una imagen digital para determinar si se trata o no de

contenido manipulado, o ***ForensicToolkit (FTK)*** (***exterro.com/forensic-toolkit***), que ofrece una solución integrada que permite recuperar la información de un dispositivo digital y realizar el análisis forense de su contenido.

Un ejemplo claro de contenido que provoca desinformación son las **fake news**. Se tratan de **noticias falsas** publicadas en la red con el objetivo de engañar, manipular e inducir a error a los lectores además de enaltecer o desprestigiar a una persona o institución. Estas noticias se presentan con apariencia de veracidad en medios digitales, especialmente en redes sociales.

Recientemente, están apareciendo en redes sociales y numerosos medios digitales contenidos audiovisuales falsos. Algunos de estos contenidos tienen un propósito humorístico, pero otros tienen objetivos similares a los de las fake news. En estos contenidos, conocidos como **deepfakes**, destacan vídeos en los que se integran caras y voces de personajes reales para elaborar contenidos falsos.

Deepfake, también conocido como **falsificación profunda**, engloba a aquellos contenidos digitales como imágenes y vídeos ficticios generados a través de técnicas de inteligencia artificial.

Para la generación de deepfakes se utiliza una técnica de inteligencia artificial, concretamente de **aprendizaje profundo**, basada en redes neuronales y llamada **Red Generativa Antagónica (RGA)**. En ella, coexisten dos redes. Una red se encarga de aprender de una gran base de datos de imágenes y vídeos existentes para generar otros nuevos. La segunda red se encarga de discriminar si las imágenes y vídeos generados por la primera son falsos o no. Cuando un contenido generado por la primera red no es detectado como falso por la segunda, entonces el deepfake está preparado.

Por todo lo anterior, garantizar la **autenticidad** de un contenido digital y **verificar** si ha sido **manipulado** o no es fundamental para evitar la desinformación y otros delitos relacionados con el fraude, el honor o la imagen. En ocasiones, es posible detectar que el contenido ha sido manipulado a través de la observación de incorrecciones en imágenes y vídeos. Para una detección más formal y rigurosa, además de las herramientas

de análisis forense vistas con anterioridad, existe tecnología **anti–deep fake** que trata de detectarlos para autenticar el contenido y también evitar que un contenido se utilice para generar nuevos deepfakes.

Sin embargo, existen muchos más recursos destinados a la tecnología para crear deepfakes que a las herramientas para detectarlos. Además, los desarrolladores de estos contenidos aprovechan las investigaciones publicadas sobre cómo detectarlos para mejorar su tecnología y seguir generando contenido falso que escape a estos sistemas. Por este motivo, el software disponible para la detección de deepfakes no está generalmente abierto a los usuarios.

NOTA

Algunas de las comprobaciones visuales que se pueden realizar para detectar un deepfake en imágenes y vídeos son: inconsistencias en la iluminación y sombreado en la imagen o en un frame, alteración de la resolución en algunas zonas y presencia de imperfecciones o incorrecciones visibles.

Entre el **software anti–deep fake** más utilizado y disponible para el público general, se encuentran los siguientes:

- **Sensity:** se trata de una plataforma web dedicada a la detección de contenido falso. Permite **subir archivos en diferentes formatos**, incluidos documentos, para detectar si se trata de contenido fraudulento. Entre sus herramientas disponibles, destacan la **verificación de identidad**, **detección de deepfakes, reconocimiento facial, o reconocimiento de documentos fraudulentos**. Para acceder a las herramientas, se debe completar un **formulario de contacto** con la compañía para solicitar el acceso.

SENSITY

sensity.ai/deepfakes-detection

- **Deepware Scanner:** se trata de un proyecto de código abierto para la **detección de deepfakes**. Su ejecución se basa en una **interfaz de línea de comandos** y permite escanear los vídeos de un directorio del dispositivo para detectar cuáles de ellos son un deepfake. Al contrario que Sensity, cuyo sistema se basa en RGA, Deepware Scanner utiliza modelos basados en **redes neuronales convolucionales**. Al tratarse de un proyecto de código abierto, es posible conocer cómo está diseñado este sistema, así como consultar el código fuente de los modelos pre-entrenados que se utilizan para la detección de deep fakes.

DEEPWARE SCANNER

github.com/deepware

- **Fake Profile Detector:** es una extensión gratuita de Google Chrome que permite identificar si una **imagen** se trata o no de un deepfake. Su instalación es muy sencilla y para utilizarla simplemente hay que hacer clic con el botón derecho del ratón en la imagen que se quiere analizar y después hacer clic en la opción "**check fake profile picture**". En ese momento aparecerá una notificación en la esquina superior derecha que indica la **probabilidad** de que se trate de una imagen falsa.

FAKE PROFILE DETECTOR
e.digitall.org.es/fake-profile

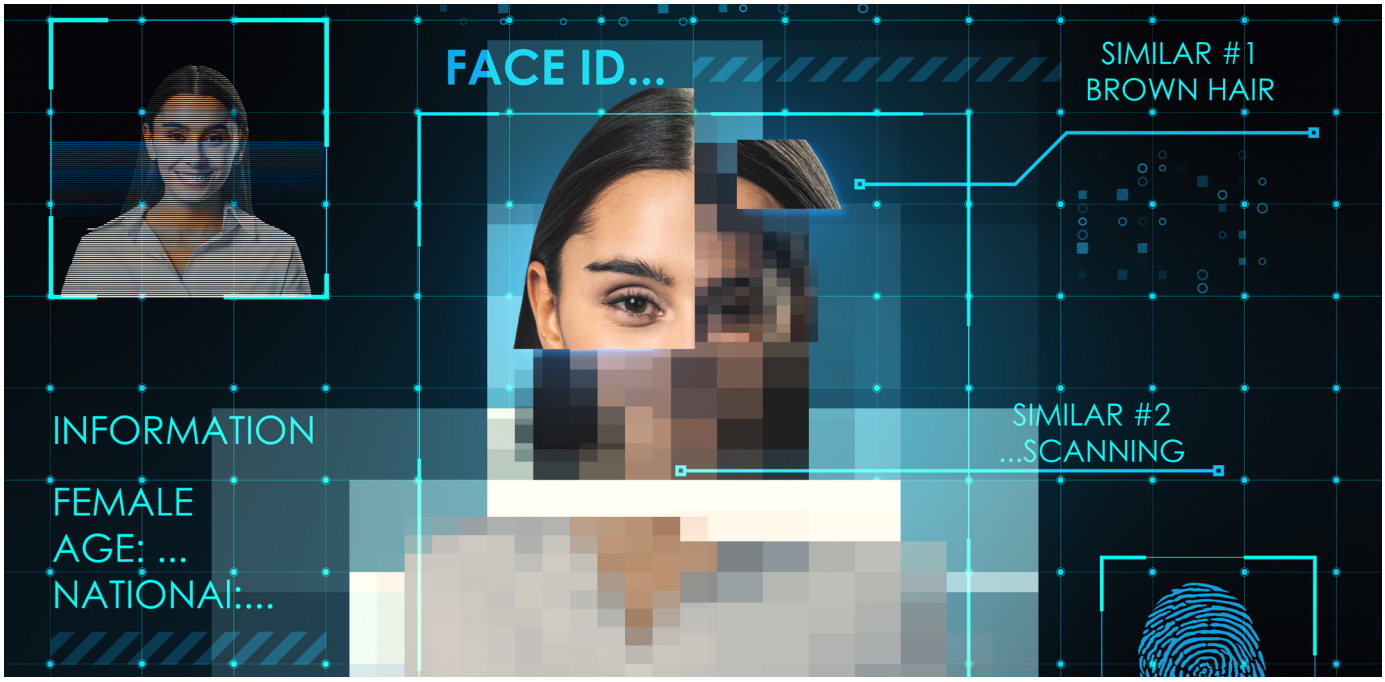
En resumen, esta sección recoge una serie de recomendaciones prácticas para autenticar contenido digital a partir de la inspección visual, así como un conjunto de herramientas propias del análisis forense informático y otra tecnología anti-deep fake, que permita a cualquier usuario autenticar y verificar la originalidad de un contenido digital.

Saber más

Otra herramienta de análisis forense es ***TinEye*** (tineye.com).

Saber más

Software de detección deepfakes: e.digitall.org.es/deepfake



DigitAll

Creación de contenidos digitales

3.3 DERECHOS DE AUTOR Y LICENCIAS DE PROPIEDAD INTELECTUAL

Creación de
contenidos digitales

Nivel C1 3.3 Derechos de autor y licencias de propiedad intelectual

Registrando el copyright y explotando una obra

Registrando el copyright y explotando una obra

El contenido de este documento brinda conocimientos sobre el mecanismo de registro de la propiedad intelectual de manera telemática para un caso concreto en la Comunidad de Castilla-La Mancha, aunque será igual para cualquier otra comunidad a excepción de Andalucía y la Comunidad de Madrid que tienen sistemas propios.

Para solicitar el registro de manera telemática, a través de la ***sede electrónica del Ministerio de Cultura y Deporte*** (***cultura.sede.gob.es/procedimientos***), hay que acceder al listado de procedimientos y seleccionar la categoría "Propiedad Intelectual", dentro de ella habrá que ir a la subcategoría "Registro de la Propiedad Intelectual" y seleccionar la ***"Solicitud de primera inscripción en el Registro de la Propiedad Intelectual"*** (***e.digitall.org.es/primera-inscripcion***). A continuación, se describirá el proceso para el registro de una Obra artística, científica, ect., de "PRIMERA INSCRIPCIÓN".

⚠ ATENCIÓN

La Solicitud de "**PRIMERA INSCRIPCIÓN**" se realiza cuando se quiere solicitar la inscripción de una obra que NO ha sido inscrita con anterioridad en el Registro de la Propiedad Intelectual (IPR), incluyendo los siguientes casos:

1 | Solicitudes de obras no inscritas presentadas por los autores, productores, etc. de las mismas.
2 | Solicitudes de obras no inscritas, cuyos derechos han sido transmitidos inter vivos por los autores, productores, etc. a un tercero.
3 | Solicitudes de obras no inscritas, cuyos derechos han sido transmitidos mortis causa por los autores, productores, etc. a sus herederos.

Para iniciar este procedimiento de solicitud de primera inscripción en el Registro de la Propiedad Intelectual, una vez en la web antes indicada, el interesado deberá clickear en el botón "Registrar" que aparece en la página. Como el procedimiento pertenece a varios ámbitos, posteriormente hay que seleccionar la comunidad autónoma para el que se solicita el acceso, en este caso particular se selecciona la Comunidad Autónoma de Castilla-La Mancha.

⚠ ATENCIÓN

Al acceder se informa sobre los requisitos generales y técnicos necesarios para poder realizar el registro. Se debe elegir el método de identificación que se utilizará para firmar la solicitud que podrá ser mediante **FIRMA BÁSICA** (mediante *Cl@ve*) o **FIRMA CON CERTIFICADO** (mediante *Autofirm@*, por lo que deberá tener instalada esta aplicación).

Una vez elegido el método de identificación, se abre la ventana con el formulario online para el registro, que contendrá 4 pestañas: Datos del solicitante, Obra, Autores y Documentación Adicional. La pestaña Datos de contacto del solicitante, ya contiene algunos datos, sin embargo, habrá que completar "Otros datos del solicitante", como son el Sexo y la Dirección. Para el registro se deben completar datos en las otras pestañas con los datos de la "Obra", los "Autores" de la misma y "Documentación Adicional", tal como se muestra en las siguientes figuras:

Figura 1. Pestaña: "Datos del solicitante".

Figura 2. Pestaña: "Obra".

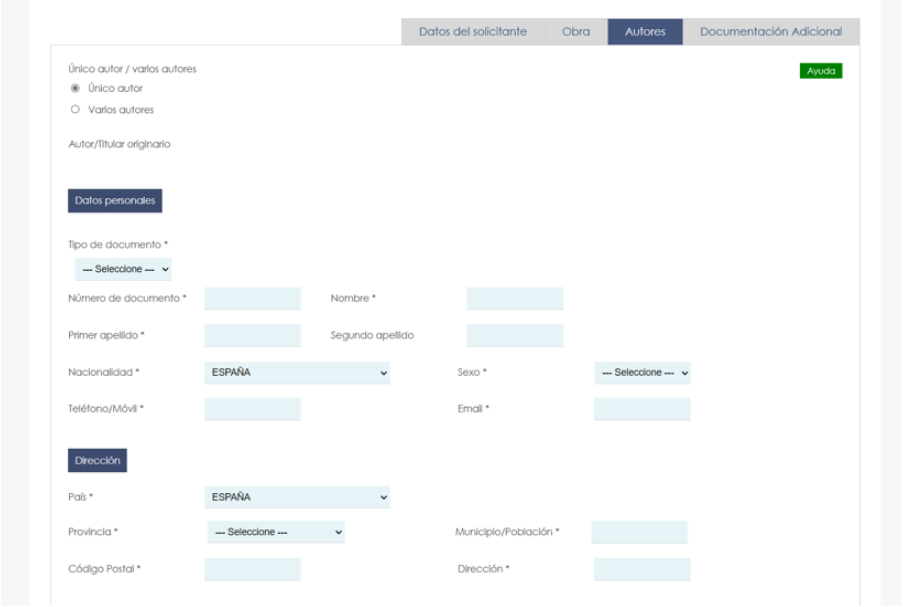


Figura 3. Pestaña: "Autores".

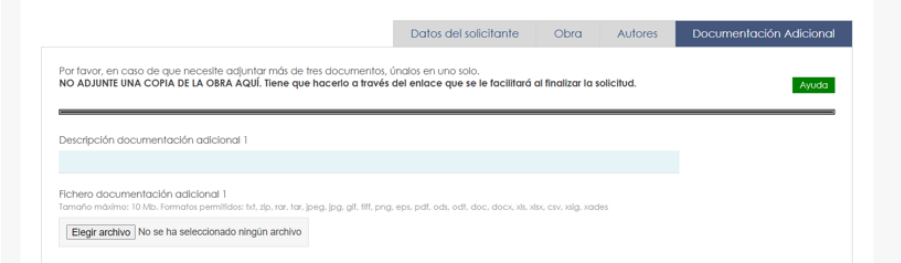


Figura 4. Pestaña: "Documentación Adicional".

Una vez rellenados todos los datos, al pie de la página se encuentra la información sobre la Tasa del Registro de la Propiedad Intelectual, se deberá indicar la "Opción de pago" seleccionada, que puede ser adjuntando el justificante de pago que previamente se ha realizado en alguna entidad bancaria o por pago telemático. En este último caso se podrá seleccionar la entidad bancaria en un desplegable y luego el método de pago que podrá ser a través de una cuenta o tarjeta de crédito. Una vez introducidos todos los datos requeridos para el registro, para hacerlo efectivo se deberá pulsar el botón "Enviar", que se encuentra en la parte inferior del formulario.

Si todo está correcto, aparecerá una ventana emergente en la que se le solicitará al interesado que firme la solicitud antes de enviarla. Si no indicará los errores encontrados solicitando que se corrijan.

Una vez firmada la solicitud, le aparecerá una pantalla en la que se le indica que la solicitud ha sido registrada con éxito y se le muestran todos los datos introducidos en la solicitud, incluyendo también el número del expediente que se ha creado, el número de registro y la fecha. Se podrá pulsar descargar el justificante del registro de la solicitud en formato PDF y también un documento con los datos de la Solicitud en formato PDF.

Una vez que el interesado haya registrado una obra, el sistema le ofrecerá una dirección web para proceder a la aportación de la copia del ejemplar identificativo.
El o los documentos que deberán adjuntarse variarán según la tipología de la obra que deben cumplir los requisitos de tipo y tamaño de fichero permitidos.

Saber más

El documento completo con la descripción detallada de cómo se debe presentar una solicitud telemática de primera inscripción se puede descargar desde: *e.digitall.org.es/manual-primera-inscripcion*

DigitAll

Creación de
contenidos digitales

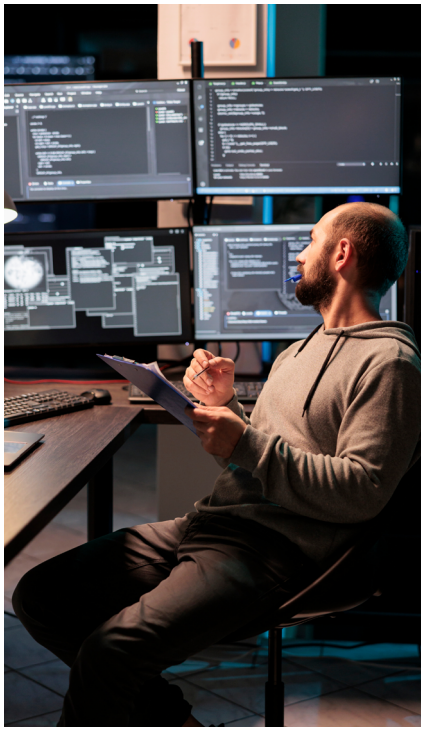
3.4
PROGRAMACIÓN

Creación de
contenidos digitales

Nivel C1 3.4 Programación

Paradigmas de programación. Visión general

Paradigmas de Programación. Visión General



A lo largo de la historia de la programación, la forma de desarrollar los programas ha ido variando en función de la necesidad de resolver problemas cada vez más complejos y diversos. Al enfoque o estilo que determina cómo abordar el diseño y la escritura de un programa se le denomina *paradigma de programación*, y consiste en un conjunto de reglas, técnicas y principios que guían todo el proceso. Los paradigmas de programación están estrechamente relacionados con los lenguajes de programación, ya que cada lenguaje de programación está diseñado para soportar uno o más paradigmas de programación. De hecho, algunos lenguajes de programación son específicos de un solo paradigma, mientras que otros son multiparadigma, pudiendo soportar varios paradigmas diferentes.

En la actualidad, existen distintos paradigmas y cada uno posee características específicas que lo diferencian del resto, relacionados con la forma en que se representan los datos y se manejan las operaciones y el flujo de control. Esta variedad es fruto de la evolución que se ha producido desde el considerado como el paradigma más antiguo, el de la **programación imperativa**, cuyo origen se remonta al nacimiento de los primeros lenguajes de programación, a finales de los años 50 del siglo XX. La idea sobre la que se fundamenta este paradigma es la de que un programa es una secuencia de instrucciones que se ejecutan en un orden específico. Así, bajo este enfoque, los programas consisten en la descripción precisa y detallada de la secuencia de pasos que hay que realizar para resolver un problema; es decir, el foco se pone en el "cómo" se resuelve el problema. Lenguajes típicos, aunque algunos ya en desuso, son Fortran, C, Pascal, Basic o Cobol.

A finales de la década de 1960, los programas se habían hecho más complejos y surgió la necesidad de mejorar su claridad, calidad y tiempo de desarrollo, lo que derivó en el paradigma de la **programación estructurada**. El principio en el que se basa es el de que cualquier programa se debe escribir haciendo uso exclusivamente de las tres estructuras de control básicas (secuencia, selección e iteración) y de módulos o procedimientos. Ada, Algol o Modula-2 son lenguajes de

programación estructurada. Por otro lado, el lenguaje C es principalmente un lenguaje de programación imperativo, pero también tiene algunas características que lo hacen adecuado para la programación estructurada.

De forma casi paralela, se desarrolla el paradigma de la **programación declarativa**, según el cual, al contrario que en el paradigma imperativo, los programas se escriben especificando "qué" es lo que debe hacer en vez de "cómo" hacerlo. Para ello, en lugar de instrucciones, se utilizan reglas y propiedades. Algunos lenguajes de programación modernos, como SQL para bases de datos o HTML para el diseño de páginas web, se basan en este paradigma. Dentro de este enfoque, existen dos paradigmas importantes:

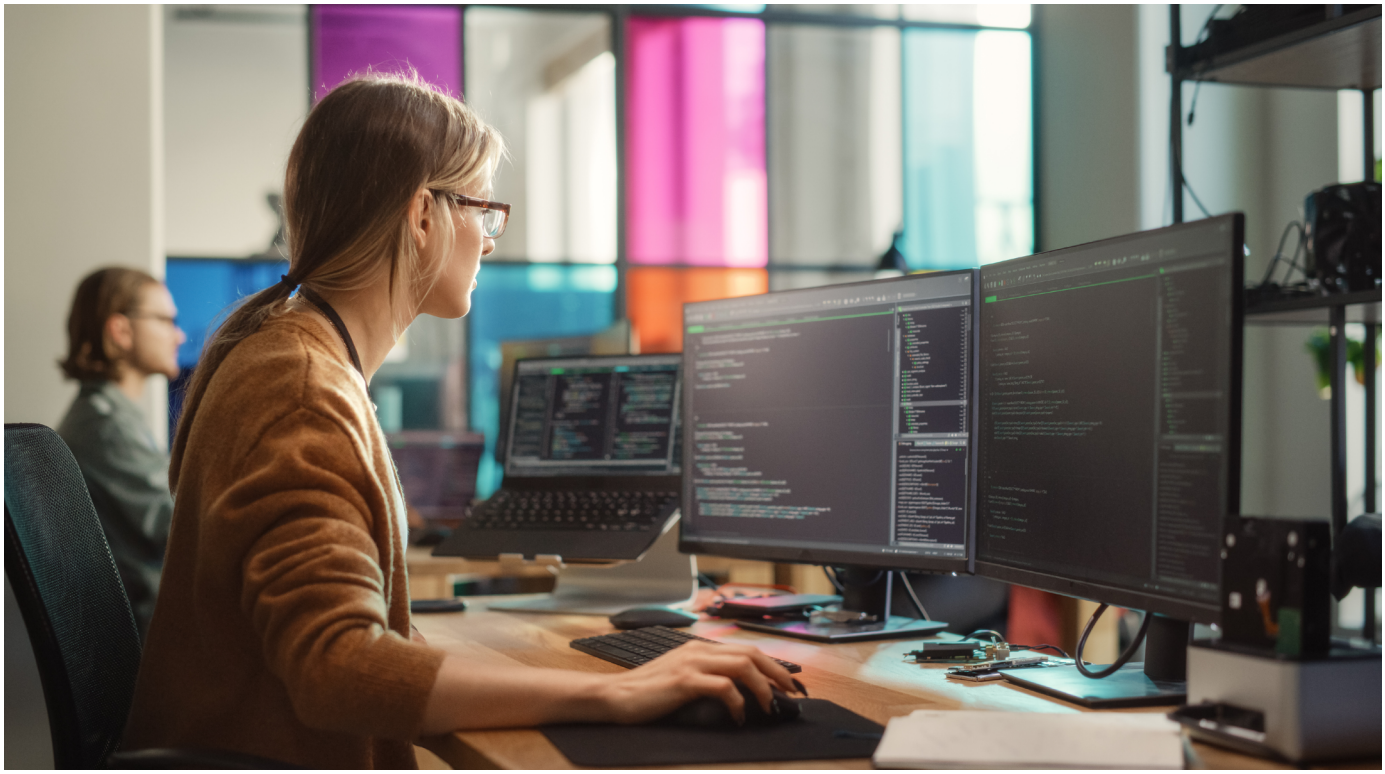
- ***Programación lógica***, según el cual el programa se construye a partir de declaraciones lógicas. El programa establece una serie de hechos y reglas, y utiliza la lógica matemática para deducir conclusiones. El lenguaje de programación Prolog, que se desarrolló en la década de 1970, fue uno de los primeros lenguajes en implementar este paradigma.
- ***Programación funcional***, que se basa en el uso exclusivamente de funciones matemáticas y en la evaluación de expresiones. Este enfoque no ganó popularidad hasta la década de los 80 del siglo XX. Lisp, Haskell y Scheme son lenguajes funcionales usados básicamente para la investigación y la enseñanza. A pesar de que se ha usado fundamentalmente en la comunidad académica, en los últimos años ha ganado popularidad en la industria del software, gracias a su capacidad para manejar de manera eficiente grandes cantidades de datos y procesos concurrentes. Lenguajes como Clojure, Scala y F# son ejemplos de lenguajes de programación funcional que se utilizan en la industria.

Una década más tarde, durante los años 90 del siglo XX, se popularizó el paradigma de la **programación orientada a objetos (POO)** como forma de representar la información de forma más parecida a como ocurre en la vida real, así como para permitir la reutilización de código. Bajo este enfoque, un programa se considera una colección de objetos que interactúan entre sí; incluso el propio programa sería un

objeto. Algunos de los lenguajes de programación orientada a objetos más populares son Java, Python y C++. Estos lenguajes modernos y multiparadigma permiten, además, la programación imperativa y estructurada.

Otro paradigma que es ampliamente utilizado en la actualidad para el desarrollo de aplicaciones web o móviles es el de la **programación orientada a eventos**, a pesar de que surgió a principios de la década de 1980. Nace de la necesidad de manejar interacciones de forma asíncrona en sistemas distribuidos y en tiempo real, como sistemas de control industrial y de telecomunicaciones. Sin embargo, ha sido el desarrollo de las interfaces gráficas de usuario lo que le ha hecho ganar popularidad. Los lenguajes de programación Java y C++ soportan este tipo de programación, aunque ganan terreno otros lenguajes más actuales como JavaScript o C#.

Todos estos paradigmas se encuadran dentro de la programación clásica. En la actualidad, empieza a extenderse un paradigma completamente distinto, el de la **programación cuántica**. Este se basa en hacer uso de los principios de la mecánica cuántica para desarrollar programas, en los que las operaciones pueden superponerse y entrelazarse, permitiendo realizar varias en paralelo. Un ejemplo de lenguaje es Q#.



Creación de
contenidos digitales

Nivel C1 3.4 Programación

Entornos de desarrollo integrado (IDE). Visión general

Entornos de desarrollo integrado (IDE). Visión general

Introducción

Los Entornos de Desarrollo Integrados, comúnmente conocidos como IDEs por su nomenclatura en inglés (Integrated Development Environments), son herramientas fundamentales para programadores y desarrolladores. Estos entornos proporcionan un conjunto de herramientas y funcionalidades integradas que simplifican y mejoran el proceso de desarrollo de software.

En este documento, analizaremos las características principales de los IDEs, así como las ventajas e inconvenientes asociados con su uso. Además, compararemos su enfoque con el del intérprete interactivo de Python.

Antes de continuar, es aconsejable que consultes el vídeo:

INTRODUCCIÓN A LOS FRAMEWORKS DE DESARROLLO

Este vídeo contiene información elemental sobre el concepto de IDE y sus principales características.

e.digitall.org.es/A3C34B1V03

¿Qué es un IDE?

Un IDE es un entorno de programación que agrupa diferentes herramientas enfocadas a la creación de software. Dentro de las cuales, cabe destacar las siguientes:

1 | Editor de código
Proporciona una interfaz para escribir y editar código fuente. Se trata de una potente herramienta que permite tanto una escritura más ágil como la reducción de errores. Algunas de las características que incluye son el resaltado de sintaxis, autocompletado y sugerencias contextuales.

2 | Depurador
La depuración de código permite identificar errores y solucionarlos una vez que han sido localizados. Los IDEs ofrecen al desarrollador diversas opciones de depuración, como la inserción de puntos de ruptura o la inspección de variables.

3 | Navegación en el código
Facilita una navegación rápida y una búsqueda sencilla y eficiente de determinados elementos dentro del código. Permite realizar búsquedas de funciones, clases y variables. Además, ofrece vistas jerárquicas del proyecto para mejorar la comprensión de la estructura del código.

4 | Integración con sistemas de control de versiones
El uso de sistemas de control de versiones es una práctica habitual entre los programadores. Por ello, los IDEs integran los sistemas de control de versiones más extendidos. Un sistema de control de versiones permite realizar un seguimiento de los cambios realizados en el código a lo largo del tiempo. De esta manera, los desarrolladores pueden gestionar y controlar el versionado de su código sin tener que acudir a aplicaciones externas.

5 | Pruebas
Las pruebas de software son una etapa clave en la creación de cualquier aplicación. Tan importante como el diseño y desarrollo de nuevas funcionalidades, es el diseño y desarrollo de pruebas. Los IDEs ofrecen herramientas y funcionalidades que facilitan el proceso de prueba y aseguran la calidad del software.

6 | Integración con herramientas externas
Los IDEs permiten la integración de una amplia variedad de herramientas externas que complementan el proceso de desarrollo de aplicaciones. Estas herramientas abarcan desde el diseño y la planificación hasta el desarrollo, optimización y pruebas de software.

Un entorno de desarrollo integrado (IDE) es un programa informático enfocado a la creación de software y que integra las herramientas más utilizadas por los desarrolladores en una misma interfaz gráfica.

Ventajas de los IDEs

Una vez descritas las principales herramientas que integran un IDE, es fácil identificar las ventajas que aportan a los desarrolladores de software. La ventaja más importante y que impacta en otros aspectos es el aumento de la productividad en todas las etapas del proceso de desarrollo. Los desarrolladores cuentan con una única interfaz gráfica que reúne las herramientas necesarias para cubrir todas las etapas del ciclo de vida del software, desde el análisis y el diseño, hasta la codificación, las pruebas y su mantenimiento posterior.

Esta ventaja se debe a que los desarrolladores pueden alternar de manera sencilla entre diferentes tareas o fases del proyecto, sin necesidad de tener que cambiar constantemente de aplicación. Además, los IDEs son altamente configurables, lo que se traduce en una adaptación a las necesidades y gustos particulares de cada desarrollador.

Cabe destacar que el uso de un IDE no solo aumenta la productividad al reducir el tiempo necesario para codificar programas, sino que también contribuye a generar un código más confiable y robusto. Estas dos ventajas vienen derivadas del uso de funciones como el autocompletado, el resaltado de sintaxis y la corrección de errores en tiempo real, características que se encuentran disponibles en el editor de los IDE.

Dado que la mayoría de los desarrolladores trabajan en equipo, es imprescindible establecer unos estándares comunes de trabajo. Los IDEs proporcionan plantillas que pueden seguir todos los miembros del equipo, facilitando el trabajo conjunto. Además, la integración de herramientas de control de versionas facilitan la compartición de código de manera segura.

En resumen, los IDEs aumentan la productividad de los desarrolladores, reducen el tiempo de instalación y agilizan las tareas de desarrollo.

IDE vs intérprete de Python

En temas anteriores se ha estudiado el concepto de intérprete interactivo, en particular el intérprete interactivo de Python. El intérprete interactivo de Python es un programa que lee instrucciones escritas en Python, las evalúa o procesa y las ejecuta. Sin embargo, el concepto de IDE es mucho más complejo y además entre sus herramientas cuenta con intérpretes o compiladores.

INTÉRPRETES INTERACTIVOS

Vídeo en el que se introduce el concepto de intérprete interactivo, así como las ventajas que ofrece como herramienta de programación. En concreto, se particulariza con el intérprete interactivo de Python, incluyendo las nociones básicas de instalación y de ejecución.

e.digitall.org.es/A3C34B2V02

Los IDEs presentan multitud de ventajas, pero a la hora de desarrollar un programa hay que sopesar las ventajas y los principales inconvenientes. Uno de ellos es la complejidad de su curva de aprendizaje, en concreto para aquellos usuarios poco expertos. Además, no sólo hay que tener en cuenta la experiencia del desarrollador, sino también el tipo de programa que se quiere desarrollar.

A continuación, se muestran una tabla en la que se compara el uso, características y curva de aprendizaje de los IDEs frente al intérprete interactivo de Python.

	IDEs	Intérprete de Python
Uso	Recomendado en proyectos complejos.	Recomendado para ejecutar pruebas rápidas y desarrollos sencillos.
Características	- Editor de código con resaltado de sintaxis y autocompletado. - Integración con herramientas externas y librerías. - Mayor productividad y agilidad en el desarrollo de software.	- Editor sencillo. - Acceso a la biblioteca estándar de Python. - Fácil acceso y ejecución rápida de código Python.
Curva de aprendizaje	Compleja.	Sencilla.

En resumen, la elección entre utilizar un IDE u otra herramienta, como el intérprete interactivo de Python, depende de diversos factores como la complejidad del proyecto, las necesidades específicas y la experiencia del desarrollador. Los IDEs son más adecuados para proyectos complejos que requieren características avanzadas, mientras que el intérprete interactivo de Python es más adecuado para desarrollos más simples y pruebas rápidas.

NOTA

Python, incluye en su instalación básica un IDE básico llamado IDLE (Integrated Development and Learning Environment). Ofrece características básicas de un IDE, como un editor de código con resaltado de sintaxis, autocompletado, búsqueda de texto, y capacidad para ejecutar código de forma interactiva. Permite además crear y ejecutar scripts de Python, depurar código paso a paso, acceder a la documentación integrada y ejecutar pruebas unitarias.

Saber más

A continuación, se listan algunos de los IDEs más populares entre los programadores de Python:

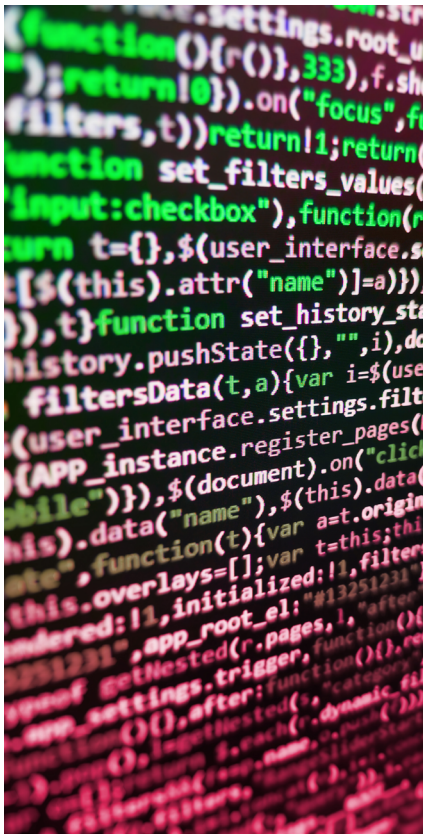
- **PyCharm:** *jetbrains.com/pycharm*
- **Spyder:** *spyder-ide.org*
- **Pydev:** *pydev.org*

Creación de
contenidos digitales

Nivel C1 3.4 Programación

Excepciones. ¿Qué son y para qué sirven?

Excepciones. ¿Qué son y para qué sirven?



Con carácter general, una excepción se puede entender como un evento que ocurre durante la ejecución de un programa y que altera su flujo de ejecución predefinido. En este sentido, la ocurrencia de una excepción es algo no deseado, ya que se aleja del comportamiento esperado de un programa. Sin embargo, es posible capturar y manejar las excepciones que puedan ocurrir en tiempo de ejecución, es decir, cuando un programa se está ejecutando, con el objetivo de recuperar la normalidad y continuar con la ejecución del mismo.

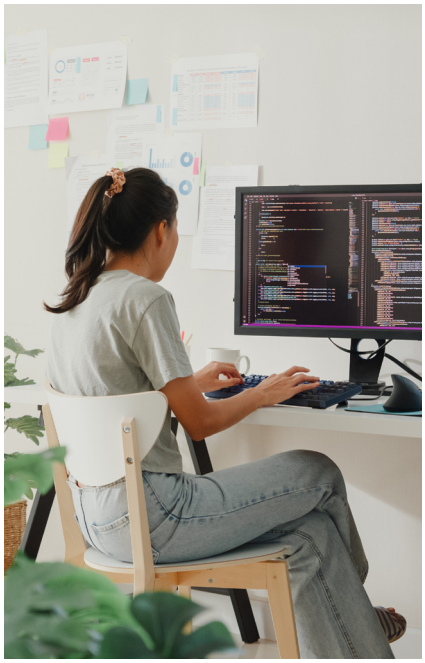
La mayoría de los lenguajes de programación modernos, como Python, incluyen soporte nativo para la definición, captura y gestión de excepciones. Esto quiere decir que el propio lenguaje ofrece instrucciones específicas para tratar excepciones, facilitando así el desarrollo de programas robustos ante errores en tiempo de ejecución. Este es el principal cometido de la gestión de excepciones.

Por otra parte, es posible clasificar las excepciones en: i) excepciones contempladas por el lenguaje de programación y ii) excepciones definidas por el usuario. Un ejemplo clásico de excepción contemplada por el lenguaje aparecería cuando una instrucción trata de realizar una división por cero. En Python, está excepción está definida por *ZeroDivisionError*. Un ejemplo de excepción definida por el usuario podría contemplar el intento de almacenamiento de un número de teléfono que no tenga exactamente 9 dígitos. Con respecto a este último ejemplo, el usuario sería responsable de la definición de este nuevo tipo de excepción, del mismo modo que ocurriría con la definición de una nueva estructura de datos.

⚠ ATENCIÓN

Cuando se manejan excepciones, existen tres tipos de operaciones representativas: lanzamiento de excepciones, captura de excepciones y manejo de excepciones. El lanzamiento se refiere a la capacidad del programador para disparar una excepción, que típicamente será recogida y manejada por otra parte del programa. La captura implica la detección explícita de una instrucción a nivel de código. Finalmente, el manejo está vinculado al código fuente definido para tratar la excepción previamente capturada.

Ejemplo de uso de excepciones en Python



En esta sección se muestra un sencillo ejemplo de uso de excepciones en Python. El objetivo no es el de profundizar sobre cómo se definen y manejan excepciones en Python. Al contrario, simplemente se pretende ejemplificar el concepto de excepción, introducido anteriormente, a través de este lenguaje.

En este sentido, el siguiente fragmento de código muestra cómo se puede obtener un número por teclado para, posteriormente, almacenarlo en una variable denominada valor. Después, se muestra el contenido de dicha variable.

```
>>> valor = int(input("Introduzca un número..."))
Introduzca un número...7
>>> valor
7
```

Desafortunadamente, este código fallará cuando el usuario introduzca, por ejemplo, una cadena de texto. En este caso, el intérprete de Python estará esperando un valor (debido a la conversión a entero realizada), por lo que arrojará la excepción ValueError.

```
>>> valor = int(input("Introduzca un número..."))
Introduzca un número...Ciudad Real
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'Ciudad Real'
```

En Python, como descubrirás más adelante con más detalle, es posible envolver un segmento de código para capturar las potenciales excepciones que puedan ocurrir. Para ello, puedes utilizar los bloques try-except. El siguiente fragmento de código captura y gestiona el posible error del usuario a la hora de introducir por teclado, en nuestro ejemplo anterior, algo que no sea un valor entero.

```
>>> try:
...     valor = int(input("Introduzca un número..."))
... except ValueError:
...     print("Debe introducir un número")
...
Introduzca un número...Madrid
Debe introducir un número
```

Creación de
contenidos digitales

Nivel C1 3.4 Programación

Procesamiento de archivos en Python

Procesamiento de archivos en Python

Como se ha visto anteriormente, Python incorpora una serie de funciones para abrir, leer, escribir y cerrar archivos. Estas funciones se pueden aplicar a cualquier tipo de archivo, aunque en esta guía se darán indicaciones para trabajar concretamente con archivos de texto plano.

Gestión Básica de Errores

La gestión de errores es una parte crucial de cualquier programa. En Python, los errores se gestionan utilizando bloques **try/except**.

Si se intenta abrir un archivo que no existe, Python lanzará un error `FileNotFoundError`. Este error puede ser capturado y gestionado usando un bloque **try/except**, para que la ejecución del programa no finalice abruptamente. Por ejemplo:

```
# Ejemplo de gestión de errores
try:
    with open("archivo_inexistente.txt", "r") as archivo:
        print(archivo.read())
except FileNotFoundError:
    print("El archivo no fue encontrado.")
```

En este caso, si el archivo no se encuentra, se mostrará el mensaje «El archivo no fue encontrado.» por pantalla, en lugar de terminar el programa con un error y continuando la ejecución del programa en caso de que hubiera más sentencias a continuación del bloque.

Búsqueda de una cadena de texto en el contenido de un archivo

Una tarea común al procesar archivos es buscar una cadena de texto específica dentro del contenido del archivo. A continuación, se presenta un ejemplo de cómo hacer esto, incluyendo el manejo de posibles errores que puedan surgir:

```
# Ejemplo de búsqueda de texto en un archivo
nombre_archivo = "archivo.txt"
cadena_busqueda = "texto a buscar"
try:
    with open(nombre_archivo, "r") as archivo:
        contenido = archivo.read()
        if cadena_busqueda in contenido:
            print("La cadena de búsqueda fue encontrada en el archivo.")
        else:
            print("La cadena de búsqueda no fue encontrada en el archivo.")
except FileNotFoundError:
    print(f"El archivo {nombre_archivo} no fue encontrado.")
```

En este código, se abre el archivo indicado por la variable `nombre_archivo`, se lee su contenido y, posteriormente, se comprueba si la cadena de búsqueda está en el contenido del archivo utilizando el operador `in`. Si la cadena de búsqueda está en el archivo, se imprime un mensaje indicándolo. Si no está, se imprime un mensaje diferente. Si el archivo no se encuentra, se imprime un mensaje de error.

Este ejemplo funciona correctamente para archivos que tengan poco contenido. En caso de que tengamos archivos con una enorme cantidad de contenido (del orden de gigabytes), esta técnica probablemente no funcionará, ya que el programa no podrá cargar el archivo al completo en la memoria del sistema para poder hacer la búsqueda.

Para resolver esto, una posible solución sería leer el contenido del archivo línea a línea, de tal forma que solo se cargue en memoria, como mucho, una línea del archivo a la vez. Por ejemplo:

```
# Ejemplo de búsqueda de texto en un archivo (versión mejorada)
nombre_archivo = "archivo.txt"
cadena_busqueda = "texto a buscar"
try:
    with open(nombre_archivo, "r") as archivo:
        linea = archivo.readline()
        while linea:
            if cadena_busqueda in linea:
                print("La cadena de búsqueda fue encontrada en el archivo.")
                break
            linea = archivo.readline()
        else:
            print("La cadena de búsqueda no fue encontrada en el archivo.")
except FileNotFoundError:
    print(f"El archivo {nombre_archivo} no fue encontrado.")
```

En este caso, se abre el archivo y se lee línea a línea. Si la cadena de búsqueda está en alguna línea del archivo, se imprime un mensaje indicándolo y se interrumpe el ciclo, de tal forma que no haya que seguir procesando el archivo. Si no está, y se han leído todas las líneas, se imprime un mensaje diferente. Si el archivo no se encuentra, se imprime un mensaje de error.

Como se ha comentado, esta versión del código puede ser más eficiente en términos de memoria cuando se trabaja con archivos muy grandes, ya que solo mantiene una línea del archivo en memoria a la vez, en lugar del contenido completo del archivo.

Conclusión

El procesamiento de archivos puede ser utilizado para una amplia variedad de tareas, desde el análisis de datos hasta la automatización de tareas. Previamente, se han introducido técnicas para abrir y leer archivos, manejar errores, y buscar cadenas de texto en el contenido de un archivo.

Saber más

Para saber más sobre la gestión de errores en Python, puedes consultar la documentación oficial en castellano: *e.digitall.org.es/excepciones*

Creación de
contenidos digitales

Nivel C1 3.4 Programación

Programación orientada a objetos. Principios y conceptos fundamentales

Programación orientada a objetos. Principios y conceptos fundamentales

A la hora de programar, los lenguajes de programación pueden seguir distintas filosofías o paradigmas de programación, según la realidad que pretenden modelar y el tipo de problemas que pretenden solucionar.

La realidad que nos rodea está repleta de objetos como sillas, mesas, coches, entre otros, que pueden plantear problemas que pueden solucionarse mediante programación. Con el fin de representar esta realidad, el paradigma de **Programación Orientada a Objetos (POO)** hace especial hincapié en el diseño y la manipulación de objetos como elementos fundamentales para diseñar la solución para cualquier programa. A continuación, se describirán los principios y conceptos fundamentales de este paradigma.

Conceptos fundamentales

A hora de modelar la solución a un problema de la vida real, es necesario trabajar con objetos tan diversos como una silla o una mesa que pueden formar parte del problema. Por tanto, es necesario describir formalmente estos objetos, pero no todas sus características, sino las que son más importantes para el programa.

Abstracción

La abstracción es uno de los pilares de la POO. Consiste en un proceso de reconocimiento de aquellas características que son fundamentales para la realidad que se quiere modelar. Debe permitir obviar aquellos detalles que son irrelevantes para un programa, centrándose únicamente en el número mínimo de detalles que se deben implementar.

Supongamos que se quiere implementar un programa en el que intervengan personas. Lo primero que se debe hacer es modelar el concepto de ***Persona***. Los elementos básicos que pueden definir a dicha persona pueden ser de dos tipos: **atributos y acciones**. Los **atributos** son las características o variables que la definen, como podrían ser su nombre y edad. Las **acciones** serían los métodos que actuarían sobre dichos atributos.

Tipo	Persona
Atributos	nombre edad
Acciones	preguntarNombre cumplirAños

Una persona puede tener más características como el sexo o el DNI. Sin embargo, haciendo un proceso de abstracción, es necesario determinar cuáles son los atributos y acciones mínimas que son necesarios para el programa a implementar. En esta figura podría verse el esquema general de un tipo Persona. Considerando esta definición o esquema general de lo que sería una Persona en un programa, en POO se correspondería con el concepto de **clase**.

Concepto de clase

Hasta ahora, para la gestión de datos en Python, se han usado tipos de datos primitivos como *int*, *float* o *string*. Estos permiten representar un tipo de dato sencillo como es el caso de un entero o una cadena de texto. Sin embargo, hay situaciones en las que es necesario modelar tipos de datos más complejos que se forman como composición de otros datos más simples. Para ello, se puede recurrir al concepto de estructuras de datos más complejas como vectores o matrices. Estas normalmente suelen ser homogéneas, es decir, contienen el mismo tipo de datos, todos los valores son numéricos o todos son cadenas de texto, por ejemplo. También, existe la posibilidad de que sean heterogéneas, es decir, pueden combinarse valores enteros, con valores reales y/o cadenas texto, por ejemplo.

Si además de trabajar sobre distintos tipos de datos, se añaden operaciones que permitieran manipular dichos datos, se estaría ante la definición de un **Tipo Abstracto de Datos**. Por lo tanto, el concepto de **clase** es un tipo de abstracto de datos, es decir, un nuevo tipo, aparte de los básicos ya conocidos como enteros o cadenas de texto.

Resumiendo, una **clase** representa una entidad cuyos atributos serán implementados mediante las variables de dicha clase y las acciones que pueda llevar a cabo mediante un conjunto funciones. Puede considerarse como un Tipo Abstracto de Datos, que incluye los datos o variables sobre los que operar y las acciones o métodos que se pueden realizar sobre dichos datos. Un ejemplo de implementación en Python de la clase **Persona** vista anteriormente sería:

```
#CÓDIGO FUENTE DE LA CLASE PERSONA

class Persona(object):
   def __init__(self, nombre, edad):
      self.nombre = nombre
      self.edad = edad
   def preguntarNombre(self):
      return self.nombre
   def cumplirAños (self):
      self.edad += 1
```

La función `preguntarNombre` permite consultar el atributo nombre de cada objeto Persona implementado, mientras que la función `cumplirAños` permite actualizar el valor de la variable edad de cada objeto Persona. De esta forma se aplica el principio de encapsulación, otro de los pilares de la POO. Los datos `edad` y `nombre` están protegidos, no se accede a ellos directamente, solo a través de métodos especialmente diseñados para su consulta y manipulación.

Además, cabe destacar el método `__init__` que recibe el nombre de **método constructor** o método de inicialización de atributos (edad y nombre) de un objeto.

Concepto de objeto

Así, una clase es un nuevo tipo de datos, igual que ya se había visto anteriormente el tipo *boolean* o *int*, que de forma abstracta representan un valor lógico o entero. Sin embargo, para poder operar sobre un valor concreto, era necesaria la definición de una variable. En POO existe el concepto análogo, que es el concepto de referencia a un objeto.

Razonando por analogía, la forma de crear una variable es darle un nombre y un valor inicial. La forma de crear un objeto es la misma, y consiste en darle un nombre a la referencia y asignar los valores de todos sus atributos.

```
nombreVariable = valorInicial
nombreReferencia = valores iniciales atributos
```

Como un objeto tiene varios atributos, no habrá un único valor inicial sino varios. Por esta razón, el método constructor es el encargado de inicializar todos los atributos a la vez.

```
pepe = Persona(nombre = "José", edad = "20")  #inicialización de un objeto
```

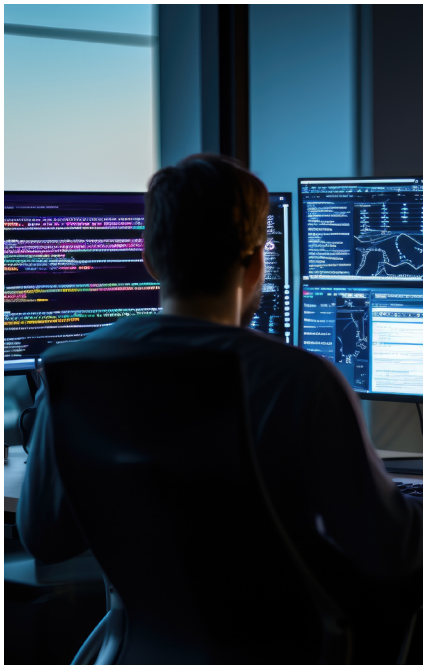
Una clase se define solamente una vez, sin embargo, se crearán tantos objetos, es decir, tantas personas, como se requieran en un programa.

Una de las características principales de los objetos es su **estado**. El **estado** representa la situación actual de sus atributos, es decir, el valor de sus variables. El estado de un objeto es dinámico. Así, una persona hoy puede tener 20 años, y mañana 21, es decir, su estado puede evolucionar con el tiempo.

Para invocar cualquier acción o método de un objeto, en Python, se utilizará la referencia creada junto al operador punto ".". Así, si se quiere que una persona cumpla un año más o preguntar cómo se llama, simplemente es necesario invocar los métodos de esta forma:

```
pepe.cumplirAños()
print(pepe.preguntarNombre())
```

Relaciones entre objetos/clases



Las clases y objetos no son elementos estancos que existen sin más. Al igual que en la vida real, los objetos interacción entre sí estableciéndose relaciones. Existen distintos tipos de relaciones que se puede caracterizar por distintos tipos de predicados.

1 | Relaciones de asociación

Se identifican por predicados del tipo: "tiene uno o varios..."

Representan una relación estructural entre objetos, es decir, una relación que perdura en el tiempo. Un ejemplo sería un coche **tiene varias** ruedas. El coche puede representarse como una clase que puede tener atributos marca y matrícula, y una rueda puede representarse como una clase con atributos como marca, presión, y diámetro.

2 | Relaciones de dependencia

Se identifican por predicados del tipo: "usa/conoce a uno o varios..."

Representan una relación temporal entre objetos. Un ejemplo sería una persona **usa un** bolígrafo. Al contrario que el ejemplo

anterior, el coche siempre tiene ruedas, sin embargo, una persona no siempre tiene un bolígrafo consigo, simplemente lo usa cuando es necesario y pierde su relación tras usarlo.

3 | Relaciones de herencia

La **herencia** es otro de los pilares básicos de la POO. Las relaciones de herencia se identifican por predicados del tipo: "es un ..."

Representan relaciones en las que los objetos comparten características y comportamientos, es decir, métodos y atributos, aunque también haya ciertas diferencias entre ellos que los hacen únicos.

Un ejemplo sería las clases Animal, Pez y Perro. El pez **es un** tipo de animal y el perro **es un** tipo de animal, por lo que tendrán cosas en común, por ejemplo, ambos tienen una edad o se les puede poner un nombre. Sin embargo, hay características que no son comunes, como la forma de moverse, uno nada y el otro camina.

La herencia es, además, la base de **polimorfismo**, otro de los pilares de la POO. Permite, por ejemplo, que una matriz de la clase Animal pueda contener varios objetos tanto de la clase Pez o de la clase Perro.

Cada una de estas relaciones tiene su propia implementación en el lenguaje de programación en el que se trabaje.

Saber más

Python.org. Definición de clases: *e.digitall.org.es/python-clases*

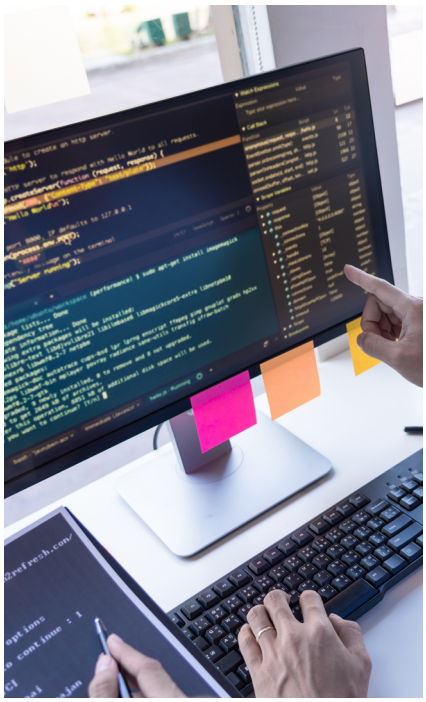
Microsoft Company. ¿Qué es la POO?: *e.digitall.org.es/poo*

Creación de
contenidos digitales

Nivel C1 3.4 Programación

Pruebas de Código. Aspectos Fundamentales

Pruebas de código. Aspectos fundamentales



En el **ciclo de desarrollo de software**, se siguen varias etapas que van desde el análisis de los requisitos hasta el mantenimiento del sistema en funcionamiento. Una de las etapas más críticas en este ciclo es la de pruebas. Las pruebas de código, o tests, son una parte integral de cualquier proceso de desarrollo de software. Proporcionan una forma de asegurar que el código escrito funcione como se esperaba y cumpla con los requisitos especificados.

Ciclo de Desarrollo de Software

El ciclo de desarrollo de software se puede dividir en las siguientes etapas:

1 | **Requisitos:** se recogen y analizan los requisitos del sistema a desarrollar.
2 | **Diseño:** se decide la arquitectura del sistema y se diseñan los diferentes componentes.
3 | **Desarrollo:** se implementa el sistema basándose en los diseños.
4 | **Pruebas:** se verifica que el sistema cumpla con los requisitos y se comporta correctamente.
5 | **Integración:** se combinan todos los componentes del sistema y se verifican como un todo.
6 | **Mantenimiento:** se realiza un seguimiento del sistema en funcionamiento, corrigiendo errores y adaptándolo a nuevas necesidades.

Las **pruebas** se realizan para identificar errores, fallos o discrepancias entre el sistema construido y los requisitos originales. No solo implican encontrar y arreglar errores en el código, sino también verificar que el sistema cumple con los requisitos y validar que es lo que el usuario necesita. Las pruebas son una forma de garantizar la calidad del software y reducir la cantidad de problemas que se encuentran en la etapa de integración.

En las siguientes páginas, vamos a explorar el enfoque de desarrollo guiado por pruebas (Test-Driven Development o *TDD*), su relación con las pruebas de código y cómo se pueden implementar en Python.

Desarrollo Guiado por Pruebas (TDD)

El Desarrollo Guiado por Pruebas (Test-Driven Development) es una metodología de desarrollo de software que gira en torno a la repetición de un ciclo de desarrollo muy corto: primero el desarrollador escribe un caso de prueba automatizado que define una mejora deseada o una nueva función, luego produce el código mínimo necesario para pasar esa prueba y finalmente refina el nuevo código al nivel aceptable.

Enfoque TDD vs. Tradicional

En el enfoque tradicional, el desarrollo de software tiende a seguir este orden: i) se escribe el código, ii) se realiza una prueba para verificar que funcione correctamente y iii) si se encuentran errores, se corrigen y se vuelve a probar. Este proceso continúa hasta que el código pasa todas las pruebas.

En contraste, TDD cambia completamente este enfoque. Antes de escribir cualquier código de producción, el desarrollador primero escribe una prueba para el nuevo código. Inicialmente, esta prueba fallará porque el código que está probando aún no existe. Luego, el desarrollador escribe el código mínimo necesario para que la prueba pase. Finalmente, se refina el código, mejorándolo sin cambiar su comportamiento.

TDD en Python

Python, al ser un lenguaje de programación de alto nivel y de propósito general, proporciona un conjunto de herramientas que permiten implementar la metodología TDD de manera efectiva.

Por ejemplo, supongamos que se quiere implementar una función que sume dos números. En un enfoque TDD, primero se escribiría una prueba para esta función:

```
import unittest

class TestSuma(unittest.TestCase):
    def test_suma(self):
        resultado = sumar(1, 2)
        self.assertEqual(resultado, 3)

if __name__ == '__main__':
    unittest.main()
```

Al ejecutar este código, la prueba fallará porque aún no se ha definido la función `sum()`. Tras esto, se implementaría la función para hacer que la prueba pase:

```
def sum(a, b):
    return a + b
```

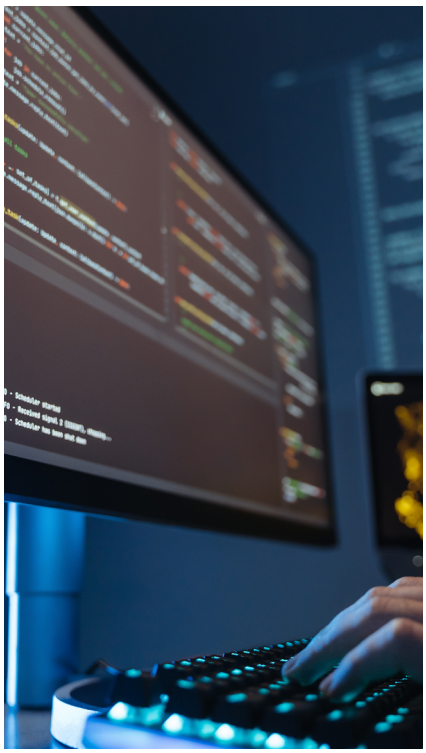
Y se volverían a ejecutar las pruebas. Ahora deberían pasar, indicando que nuestra función cumple con la expectativa que hemos definido en la prueba.

Conclusión

Las pruebas de código son una parte integral de cualquier proceso de desarrollo de software. Proporcionan una manera de asegurar que el software desarrollado se comporta de acuerdo con las expectativas y cumple con los requisitos definidos. Por otro lado, el Desarrollo Guiado por Pruebas (TDD) es una metodología que pone las pruebas en el centro del desarrollo de software, promoviendo la escritura de código de alta calidad que cumple con las necesidades del usuario y minimiza la aparición de errores.

Python, con su conjunto de herramientas de prueba y su sintaxis clara y concisa, es una excelente opción para implementar la metodología TDD. El enfoque de escribir pruebas antes que el código de producción ayuda a definir claramente las expectativas para el código, facilita la detección de errores y promueve la escritura de código limpio y mantenible.

Las pruebas son una inversión que puede ahorrar mucho tiempo y esfuerzo en el futuro, evitando la aparición de errores inesperados y facilitando la detección y corrección de estos cuando ocurren. Adoptar un enfoque de desarrollo centrado en las pruebas puede mejorar la calidad del software y hacer que el proceso de desarrollo sea más eficiente y efectivo.



Saber más

Para saber más sobre el desarrollo guiado por pruebas en Python, puedes consultar la documentación oficial en castellano de los siguientes módulos:

- **unittest:** *e.digitall.org.es/unittest*
- **doctest:** *e.digitall.org.es/doctest*

DigitAll

Formación en
Competencias
Digitales

https://www.digitall.org.es/

DigitAll

Coordinación General

Universidad de Castilla-La Mancha
Carlos González Morcillo
Francisco Parreño Torres

Coordinadores de área

Área 1. Búsqueda y gestión de información y datos

Universidad de Zaragoza
Francisco Javier Fabra Caro

Área 2. Comunicación y colaboración

Universidad de Sevilla
Francisco Javier Fabra Caro
Francisco de Asís Gómez Rodríguez
José Mariano González Romano
Juan Ramón Lacalle Remigio
Julio Cabero Almenara
María Ángeles Borrueco Rosa

Área 3. Creación de contenidos digitales

Universidad de Castilla-La Mancha
David Vallejo Fernández
Javier Alonso Albusac Jiménez
José Jesús Castro Sánchez

Área 4. Seguridad

Universidade da Coruña
Ana M. Peña Cabanas
José Antonio García Naya
Manuel García Torre

Área 5. Resolución de problemas

UNED
Jesús González Boticario

Coordinadores de nivel

Nivel A1

Universidad de Zaragoza
Ana Lucía Esteban Sánchez
Francisco Javier Fabra Caro

Nivel A2

Universidad de Córdoba
Juan Antonio Romero del Castillo
Sebastián Rubio García

Nivel B1

Universidad de Sevilla
Francisco de Asís Gómez Rodríguez
José Mariano González Romano
Juan Ramón Lacalle Remigio
Montserrat Argandoña Bertran

Nivel B2

Universidad de Castilla-La Mancha
María del Carmen Carrión Espinosa
Rafael Casado González
Víctor Manuel Ruiz Penichet

Nivel C1

UNED
Antonio Galisteo del Valle

Nivel C2

UNED
Antonio Galisteo del Valle

Maquetación

Universidad de Salamanca
Fernando De la Prieta Pintado
Pilar Vega Pérez
Sara Alejandra Labrador Martín

Creadores de contenido

Área 1. Búsqueda y gestión de información y datos

1.1 Navegar, buscar y filtrar datos, información y contenidos digitales

Universidad de Huelva
Ana Duarte Hueros (coord.)
Arantxa Vizcaíno Verdú
Carmen González Castillo
Dieter R. Fuentes Cancell
Elisabetta Brandi
José Antonio Alfonso Sánchez
José Ignacio Aguaded
Mónica Bonilla del Río
Odiel Estrada Molina
Tomás de J. Mateo Sanguino (coord.)

1.2 Evaluar datos, información y contenidos digitales

Universidad de Zaragoza
Ana Belén Martínez Martínez
Ana María López Torres
Francisco Javier Fabra Caro
José Antonio Simón Lázaro
Laura Bordonaba Plou
María Sol Arqued Ribes
Raquel Trillo Lado

1.3 Gestión de datos, información y contenidos digitales

Universidad de Zaragoza
Ana Belén Martínez Martínez
Francisco Javier Fabra Caro
Gregorio de Miguel Casado
Sergio Ilarri Artigas

Área 2. Comunicación y colaboración

2.1 Interactuar a través de tecnología digitales

Iseazy

2.2 Compartir a través de tecnologías digitales

Universidad de Sevilla
Alién García Hernández
Daniel Agüera García
Jonatan Castaño Muñoz
José Candón Mena
José Luis Guisado Lizar

2.3 Participación ciudadana a través de las tecnologías digitales

Universidad de Sevilla
Ana Mancera Rueda
Félix Biscarri Triviño
Francisco de Asís Gómez Rodríguez
Jorge Ruiz Morales
José Manuel Sánchez García
Juan Pablo Mora Gutiérrez
Manuel Ortigueira Sánchez
Raúl Gómez Bizcocho

2.4 Colaboración a través de las tecnologías digitales

Universidad de Sevilla
Belén Vega Márquez
David Vila Viñas
Francisco de Asís Gómez Rodríguez
Julio Barroso Osuna
María Puig Gutiérrez
Miguel Ángel Olivero González
Óscar Manuel Gallego Pérez
Paula Marcelo Martínez

2.5 Comportamiento en la red

Universidad de Sevilla
Ana Mancera Rueda
Eva Mateos Núñez
Juan Pablo Mora Gutiérrez
Óscar Manuel Gallego Pérez

2.6 Gestión de la identidad digital

Iseazy

Área 3. Creación de contenidos digitales

3.1 Desarrollo de contenidos

Universidad de Castilla-La Mancha
Carlos Alberto Castillo Sarmiento
Diego Cordero Contreras
Inmaculada Ballesteros Yáñez
José Ramón Rodríguez Rodríguez
Rubén Grande Muñoz

3.2 Integración y reelaboración de contenido digital

Universidad de Castilla-La Mancha
José Ángel Martín Baos
Julio Alberto López Gómez
Ricardo García Ródenas

3.3 Derechos de autor (copyright) y licencias de propiedad intelectual

Universidad de Castilla-La Mancha
Gabriela Raquel Gallicchio Platino
Gerardo Alain Marquet García

3.4 Programación

Universidad de Castilla-La Mancha
Carmen Lacave Rodero
David Vallejo Fernández
Javier Alonso Albusac Jiménez
Jesús Serrano Guerrero
Santiago Sánchez Sobrino
Vanesa Herrera Tirado

Área 4. Seguridad

4.1 Protección de dispositivos

Universidade da Coruña
Antonio Daniel López Rivas
José Manuel Vázquez Naya
Martiño Rivera Dourado
Rubén Pérez Jove

4.2 Protección de datos personales y privacidad

Universidad de Córdoba
Aida Gema de Haro García
Ezequiel Herruzo Gómez
Francisco José Madrid Cuevas
José Manuel Palomares Muñoz
Juan Antonio Romero del Castillo
Manuel Izquierdo Carrasco

4.3 Protección de la salud y del bienestar

Universidade da Coruña
Javier Pereira Loureiro
Laura Nieto Riveiro
Laura Rodríguez Gesto
Manuel Lagos Rodríguez
María Betania Groba González
María del Carmen Miranda Duro
Nereida María Canosa Domínguez
Patricia Concheiro Moscoso
Thais Pousada García

4.4 Protección medioambiental

Universidad de Córdoba
Alberto Membrillo del Pozo
Alicia Jurado López
Luis Sánchez Vázquez
María Victoria Gil Cerezo

Área 5. Resolución de problemas

5.1 Resolución de problemas técnicos

Iseazy

5.2 Identificación de necesidades y respuestas tecnológicas

Iseazy

5.3 Uso creativo de la tecnología digital

Iseazy

5.4 Identificar lagunas en las competencias digitales

Iseazy

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia