**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | TECNICO EN PROGRAMACIÓN DE VIDEOJUEGOS |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | Elaborar la lógica del contenido digital de acuerdo con métodos de desarrollo y propuesta de diseño. | RESULTADOS DE APRENDIZAJE | 220501127-01. Identificar los requerimientos del videojuego de acuerdo con el documento de diseño |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | 03 |
| NOMBRE DEL COMPONENTE FORMATIVO | API de videojuegos y repositorios |
| BREVE DESCRIPCIÓN | Durante el desarrollo de este componente formativo, el aprendiz aprenderá los conceptos de API y su relación con el desarrollo de videojuegos. Asimismo, conocerá cuále  s son los dispositivos que se pueden utilizar como entrada y/o salida para que el usuario interactúe con ellos. Además, como tema final, se explicará qué es el versionamiento en *software* y cómo se manejan los repositorios basados en GIT. |
| PALABRAS CLAVE | API, Versionamiento, GIT, *Hardware,* Motor de videojuego |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 5 - ARTE, CULTURA, ESPARCIMIENTO Y DEPORTES |
| IDIOMA | Español |

1. **TABLA DE CONTENIDOS:**
2. Sistemas Operativos, API y librerías
   1. Sistemas Operativos
   2. Concepto de API
   3. Motores para videojuegos multiplataforma
   4. Librerías para videojuegos
   5. Librerías de videojuegos web
   6. Librerías de videojuegos móviles
3. Periféricos para videojuegos
4. Repositorios GIT
   1. Versiones en *software*
   2. Definición de GIT
   3. Configuración de GIT
   4. Operaciones en repositorios
5. **DESARROLLO DE CONTENIDOS**

**INTRODUCCIÓN**

Estimado aprendiz bienvenido al componente formativo “APIs de videojuegos y repositorios” Para comenzar, le invitamos a ingresar al siguiente video para obtener más información:

|  |
| --- |
| **Video introducción**  **CF03\_Introducción** |

**1. Sistemas Operativos, API y librerías**

Los sistemas operativos, las API y las librerías son componentes esenciales en el ámbito de la informática y la programación. Los sistemas operativos, como Windows, macOS o Linux, son fundamentales para gestionar los recursos del *hardware* y proporcionar una interfaz entre los usuarios y la máquina. Las API, o interfaces de programación de aplicaciones, son cruciales para permitir que diferentes programas y servicios interactúen y compartan funcionalidades o datos. Por último, las librerías, que son colecciones de funciones y procedimientos preescritos, facilitan enormemente el proceso de desarrollo de *software* al ofrecer bloques de construcción estandarizados, lo que ahorra tiempo y esfuerzo en la programación. Estos tres elementos trabajan conjuntamente para crear un ecosistema robusto y eficiente, esencial para el desarrollo moderno de *software* y aplicaciones.

**1.1 Sistemas Operativos**

Un Sistema Operativo (SO) es el programa que, después de ser inicialmente cargado en la computadora mediante un programa de arranque, administra todos los demás programas de aplicación en una computadora. Los programas de aplicación utilizan el sistema operativo haciendo solicitudes de servicios a través de una Interfaz de Programación de Aplicaciones (API) definida. Además, los usuarios pueden interactuar directamente con el sistema operativo a través de una interfaz de usuario, como una Interfaz de Línea de Comandos (CLI) o una Interfaz Gráfica de Usuario (GUI).

Los Sistemas Operativos se clasifican en:

**Escritorio**

Los sistemas operativos de escritorio son aquellos diseñados para ser utilizados en computadoras personales y estaciones de trabajo. En el siguiente video se explican los más comunes:

|  |
| --- |
| **Video**  **CF03\_1.1 Sistemas Operativos** |

**Móvil**

Los sistemas operativos móviles están diseñados para satisfacer las necesidades específicas de la informática móvil y los dispositivos centrados en la comunicación, como teléfonos inteligentes y tabletas. Estos dispositivos generalmente ofrecen recursos computacionales limitados en comparación con las PC tradicionales, y el sistema operativo se optimiza en tamaño y complejidad para minimizar el uso de recursos, asegurando al mismo tiempo suficientes recursos para una o más aplicaciones en ejecución en el dispositivo. Los sistemas operativos móviles se enfocan en un rendimiento eficiente, capacidad de respuesta y manejo avanzado de tareas de datos, como el soporte para transmisión de medios.

|  |  |
| --- | --- |
| Ejemplos de sistemas operativos móviles son Apple iOS y Google Android. | Apple and android smartphones. Iphone IOS versus Android operating system |

**Integrados**

No todos los dispositivos informáticos son de propósito general. Una amplia variedad de dispositivos especializados, incluyendo asistentes digitales domésticos, cajeros automáticos (ATM), sistemas de aeronaves, terminales de punto de venta (POS) y dispositivos del Internet de las Cosas (IoT), incorporan computadoras que requieren un sistema operativo. La principal diferencia es que estos dispositivos están diseñados para realizar una función principal, lo que permite un sistema operativo altamente especializado, enfocado en rendimiento y robustez. El sistema operativo debe ser rápido, confiable y manejar errores adecuadamente para funcionar bajo cualquier circunstancia. Comúnmente, se incorpora en un chip integrado en el dispositivo.

|  |  |
| --- | --- |
| Por ejemplo, un dispositivo médico en un equipo de soporte vital empleará un sistema operativo integrado que debe operar de manera confiable para mantener al paciente con vida. Linux integrado es un ejemplo de este tipo de sistema operativo. | A close-up of an ECG monitor displaying a patient's heart rhythm in real-time |

**De red**

Un Sistema Operativo de Red (NOS) es un tipo especializado destinado a facilitar la comunicación entre dispositivos en una Red de Área Local (LAN). Un NOS proporciona la pila de comunicación necesaria para manejar los protocolos de red, creando, intercambiando y descomponiendo paquetes de red. Hoy, la idea de un NOS especializado se ha vuelto en gran parte obsoleta, ya que otros tipos de sistemas operativos ahora manejan la mayoría de las comunicaciones de red.

|  |  |
| --- | --- |
| Por ejemplo, Windows 10 y Windows Server 2019 incluyen capacidades avanzadas de red. Aunque el concepto de NOS sigue siendo relevante en dispositivos de red como enrutadores, conmutadores y *firewalls,* donde fabricantes como Cisco utilizan NOS propietarios como Cisco Internetwork Operating System (IOS), RouterOS y ZyNOS. | Tula, Russia - AUGUST 18, 2019: Microsoft Windows Server 2019 displayed on a modern laptop |

**En tiempo real**

Cuando un dispositivo informático necesita interactuar con el mundo real dentro de límites de tiempo estrictos y predecibles, se puede optar por un Sistema Operativo en Tiempo Real (RTOS).

|  |  |
| --- | --- |
| Por ejemplo, un sistema de control industrial puede gestionar las operaciones de una fábrica o planta de energía. Estos sistemas reciben señales de numerosos sensores y envían señales para operar válvulas, actuadores, motores y otros dispositivos. En estas situaciones, el sistema debe responder rápidamente y de manera predecible a cambios en el mundo real para evitar desastres. Un RTOS debe operar sin almacenamiento en búfer, latencias de procesamiento y otros retrasos que son aceptables en otros tipos de sistemas operativos. FreeRTOS y VxWorks son ejemplos de RTOS. | Engineer working at control room,Manager control system,Technician man monitoring program from a lot of monitor |

**1.2 Concepto de API**

API significa “interfaz de programación de aplicaciones”. (*Application Program Interface*) En el contexto de las API, la palabra aplicación se refiere a cualquier *software* con una función distinta. La interfaz puede considerarse como un “contrato” de servicio entre dos aplicaciones. Este “contrato” define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de su API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas. Las API son mecanismos que permiten a dos componentes de *software* comunicarse entre sí mediante un conjunto de definiciones y protocolos.

|  |  |
| --- | --- |
| Application programming interface illustration concept. Illustration for websites, landing pages, mobile applications, posters and banners. Trendy flat vector illustration | Por ejemplo, el sistema de *software* del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono “habla” con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono. |

**¿Cómo funcionan las API?**

La arquitectura de las API suele explicarse en términos de cliente y servidor. La aplicación que envía la solicitud se llama cliente, y la que envía la respuesta se llama servidor. En el ejemplo del tiempo, la base de datos meteorológicos del instituto es el servidor y la aplicación móvil es el cliente.

Las API pueden funcionar de cuatro maneras diferentes, según el momento y el motivo de su creación:

|  |
| --- |
| **Slide**  **CF03\_1.2. Concepto de API** |

**1.3** **Motores para videojuegos multiplataforma**

Los motores para videojuegos multiplataforma son herramientas de desarrollo de *software* diseñadas para crear videojuegos que se pueden ejecutar en varios sistemas operativos y dispositivos, como PC, consolas y dispositivos móviles. Algunos de los motores de videojuegos multiplataforma más populares incluyen:

|  |
| --- |
| **Acordeón**  **CF03\_1.3\_Motores para videojuegos multiplataforma** |

Estos motores varían en términos de capacidades gráficas, facilidad de uso, y modelos de licencia, pero todos ofrecen la capacidad de desarrollar juegos para múltiples plataformas, lo que es esencial en el mercado de juegos actual.

**1.4.** **Librerías para videojuegos**

Las bibliotecas o librerías para el desarrollo de videojuegos en Python ofrecen una variedad de herramientas y funciones para facilitar la creación de juegos. A continuación, se presentan algunas de las librerías:

|  |
| --- |
| **Pestañas**  **CF03\_1.4\_Librerías para videojuegos** |

Cada una de estas librerías tiene sus propias fortalezas y casos de uso ideales, desde la creación de juegos simples hasta el desarrollo de aplicaciones 3D complejas.

**1.5** **Librerías de videojuegos web**

Las bibliotecas de videojuegos para la web son herramientas esenciales para desarrollar juegos que se ejecutan en navegadores web. A continuación, se presentan algunas de las librerías:

|  |
| --- |
| **Pestañas**  **CF03\_1.5\_Librerías de videojuegos web** |

Estas bibliotecas y herramientas varían en términos de capacidades gráficas, facilidad de uso y enfoque (2D vs 3D), pero todas ofrecen potentes opciones para desarrollar juegos accesibles a través de navegadores web.

**1.6** **Librerías de videojuegos móviles**

Para el desarrollo de videojuegos móviles, existen varias bibliotecas y herramientas que facilitan la creación de juegos para plataformas como Android e iOS. Estas pueden ser:

|  |
| --- |
| **Carrusel de tarjetas**  **CF03\_1.6\_Librerías de videojuegos móviles** |

**2. Periféricos para videojuegos**

La industria del juego y la tecnología ha evolucionado significativamente en los últimos años. Hoy en día, los videojuegos tienen como característica ser más inmersivos e interactivos. A medida que avanza la tecnología, también lo hacen los periféricos que la acompañan.

Desde controladores de alta tecnología hasta cascos de realidad virtual, los periféricos para juegos se han vuelto esenciales para la experiencia de juego sin importar el nivel del jugador. Entre ellos destaca:

|  |
| --- |
| **Video**  **CF03\_2\_Periféricos para videojuegos** |

Se debe considerar los siguientes elementos al momento de desarrollar el videojuego:

|  |
| --- |
| **Infografia**  **CF03\_2\_Periféricos para videojuegos (2)** |

Si el videojuego que se quiere construir tiene como esencia ser orientada a Realidad Virtual, aumentada o Mixta igualmente debe considerarse las diferentes marcas y tecnologías disponibles en el mercado que permitan al usuario tener una buena experiencia en el momento de utilizar este tipo de tecnologías.

**3. Repositorios GIT**

Los repositorios GIT son una parte esencial del sistema de control de versiones Git, utilizado en el desarrollo de *software* para manejar el código fuente. Cada repositorio Git contiene todo el historial de cambios del proyecto, conocidos como *commits,* así como las distintas ramas y etiquetas. Estas características permiten a los desarrolladores registrar cada cambio en los archivos, trabajar en distintas características de forma aislada mediante las ramas, y marcar versiones específicas del proyecto con etiquetas. Su naturaleza distribuida significa que cada desarrollador trabaja con una copia completa del repositorio, facilitando la colaboración y el seguimiento detallado de los cambios.

Además de almacenar el código y su historial, los repositorios Git apoyan en la colaboración entre desarrolladores y en el seguimiento de errores o *bugs.* Plataformas de alojamiento como GitHub, GitLab y Bitbucket extienden las funcionalidades de Git, proporcionando herramientas para la revisión de código, la gestión de proyectos y la colaboración en equipo. Estos repositorios son indispensables en el mundo del desarrollo de *software,* ofreciendo un control detallado y eficiente sobre las diferentes versiones y cambios realizados en un proyecto.

**3.1 Versiones en *software***

El control de versiones es el proceso de gestionar múltiples versiones de un mismo producto, las cuales, aunque cumplen con la misma función general, pueden diferir en términos de mejoras, actualizaciones o personalizaciones. Aunque el término se utiliza en diversos contextos, es más comúnmente aplicado a sistemas operativos, *software* y servicios web.

|  |  |
| --- | --- |
| Por ejemplo, los proveedores pueden asignar nombres únicos o identificadores numéricos a diferentes versiones de un producto, como es el caso de Windows 10 y Windows 11. Estos números representan distintas versiones del mismo *software* o aplicación, incrementándose para reflejar desarrollos y actualizaciones recientes. | Viersen, Germany - January 9. 2022: Closeup of mobile phone with logo lettering of windows 10 and 11 operating system on computer keyboard |

Los lanzamientos sucesivos de un producto también pueden tener identificadores numéricos, generalmente consistiendo en dos o tres cifras separadas por puntos.

* **El primer número,** conocido como número mayor, se incrementa con mejoras significativas o cambios en la funcionalidad. Por ejemplo, el cambio de Internet Explorer 4 a Internet Explorer 5 por parte de Microsoft.
* **El segundo número,** denominado número menor, aumenta con cambios menores en las funciones o correcciones de errores importantes. Por ejemplo, Internet Explorer 5.2 indica cambios menores respecto a la versión anterior, Internet Explorer 5.1.
* **El tercer número***,* si se incluye, se conoce como número de revisión y se añade o incrementa para correcciones de errores menores. Así, el '7' en Internet Explorer 5.1.7 indica la corrección de errores menores en la versión anterior

**3.2 Definición de GIT**

GIT es una plataforma de desarrollo de *software* en línea. Se utiliza para almacenar, rastrear y colaborar en proyectos de *software***.** Los usuarios de GIT crean cuentas, cargan archivos y crean proyectos de codificación, permitiendo, de esta manera que los usuarios comienzan a colaborar.

 Si bien cualquiera puede codificar de forma independiente, la mayoría de los proyectos de desarrollo los construyen equipos de personas. A veces, estos equipos están todos en un lugar a la vez, pero lo más frecuente es que trabajen de forma asincrónica. Adicionalmente es el sistema de control de versiones más utilizado en el desarrollo de *software.*

**Utilidad de Git en el Versionamiento**

Al crear *software*, los desarrolladores actualizan el código con frecuencia y simultáneamente para agregar funciones y corregir errores. No tendría sentido realizar estos cambios directamente en el código fuente, ya que cualquier problema afectaría a los usuarios. En cambio, los desarrolladores trabajan con sus propias copias del código y luego, después de que el código se haya probado exhaustivamente, lo agregan a la base de código principal.

**3.3 Configuración de GIT**

Desde la instalación de Git hasta la creación de su primer repositorio en GitHub, a continuación, se proporcionará los conocimientos básicos necesarios para comenzar. Siga estos pasos para embarcarse en su viaje hacia una gestión de proyectos de *software* exitosa y eficiente.

|  |
| --- |
| **Pasos**  **CF03\_3.3\_Configuración de GIT** |

Para obtener una visión más detallada de la instalación, puede revisarla en el siguiente enlace: [**https://www.youtube.com/playlist?list=PLQxX2eiEaqby-qh4raiKfYyb4T7WyHsfW**](https://www.youtube.com/playlist?list=PLQxX2eiEaqby-qh4raiKfYyb4T7WyHsfW)

**3.4 Operaciones en repositorios**

Las operaciones básicas de Git, como *init, add, commit, y push,* permiten a los usuarios crear repositorios, gestionar cambios, guardar progresos y compartir su trabajo con otros. A continuación, se explora las operaciones fundamentales en repositorios Git, proporcionando una guía esencial para cualquiera que busque entender o mejorar su flujo de trabajo en el desarrollo de *software.*

* ***git INIT:*** el comando *git init* se utiliza para crear un repositorio Git vacío. Después de usar el comando *git init,* se crea una carpeta. git en el directorio con algunos subdirectorios. Una vez que se inicializa el repositorio, comienza el proceso de creación de otros archivos.
* ***git ADD:*** el comando Agregar se usa después de verificar el estado de los archivos, para agregar esos archivos al área de preparación. Antes de ejecutar el comando de confirmación, se utiliza "*git add*" para agregar archivos nuevos o modificados.
* ***git COMMIT:*** el comando de confirmación asegura que los cambios se guarden en el repositorio local. El comando "*git commit –m <mensaje>"* le permite describir a todos y ayudarlos a comprender lo que sucedió.
* ***git STATUS*:** el comando *git status* indica el estado actual del repositorio. El comando proporciona la rama de trabajo actual. Si los archivos están en el área de preparación, pero no confirmados, se mostrará en el estado de git. Además, si no hay cambios, mostrará el mensaje No hay cambios para confirmar, directorio de trabajo limpio.
* ***git CONFIG*:** el comando *git config* se usa inicialmente para configurar el nombre de usuario y el correo electrónico de usuario. Esto especifica qué ID de correo electrónico y nombre de usuario se utilizarán desde un repositorio local. Cuando se usa *git config* con el indicador --global, escribe la configuración en todos los repositorios de la computadora.
* ***git BRANCH*:** el comando *git branch* se utiliza para determinar en qué rama se encuentra el repositorio local. El comando permite agregar y eliminar una rama.
* ***git CHECKOUT:*** el comando *git checkout* se utiliza para cambiar de rama, siempre que el trabajo deba iniciarse en una rama diferente. El comando funciona en tres entidades separadas: archivos, confirmaciones y ramas.
* ***git MERGE*:** el comando *git merge* se utiliza para integrar las ramas. El comando combina los cambios de una rama a otra. Se utiliza para fusionar los cambios en la rama provisional con la rama estable.
* ***git REMOTE:*** el comando git remote se utiliza para crear, ver y eliminar conexiones a otros repositorios. Las conexiones aquí no son como enlaces directos a otros repositorios, sino como marcadores que sirven como nombres convenientes para usar como referencia.
* ***git CLONE*:** el comando git clone se utiliza para crear una copia de trabajo local de un repositorio remoto existente. El comando descarga el repositorio remoto a la computadora. Es equivalente al comando Git init cuando se trabaja con un repositorio remoto.
* ***git PULL*:** el comando *git pull* se utiliza para recuperar y fusionar cambios del repositorio remoto al repositorio local. El comando "*git pull origin master"* copia todos los archivos de la rama maestra del repositorio remoto al repositorio local.
* ***git PUSH****:* el comando *git push* se usa para transferir las confirmaciones o enviar el contenido del repositorio local al repositorio remoto. El comando se utiliza después de que se haya modificado un repositorio local y las modificaciones se compartirán con los miembros del equipo remoto.

1. **SÍNTESIS**

Diagrama

Descripción generada automáticamente con confianza media

1. **ACTIVIDADES DIDÁCTICAS**

|  |  |
| --- | --- |
| Nombre de la Actividad | Prueba de Conocimiento sobre Sistemas Operativos, Apis y Librerías |
| Objetivo de la actividad | Determinar el nivel de aprendizaje sobre los conceptos básicos acerca de los Sistemas Operativos, APIs y Librerías en un entorno de desarrollo de videojuegos para garantizar la adecuada apropiación de los conceptos |
| Tipo de actividad sugerida | Cuestionario de Preguntas Opción Múltiple |
| **Archivo de la actividad**  **(Anexo donde se describe la actividad propuesta)** | A01-CF03-Cuestionario\_de\_Preguntas\_CF03 |

1. **MATERIAL COMPLEMENTARIO:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tema** | **Referencia APA del Material** | **Tipo de material**  **(Video, capítulo de libro, artículo, otro)** | **Enlace del Recurso o**  **Archivo del documento o material** |
| Sistemas Operativos | ElSalondeInformatica. (2021, febrero 18). SISTEMAS OPERATIVOS | ¿Que son?¿Como Funcionan? | TIPOS de SISTEMAS OPERATIVOS | Explicación Facil. Youtube. | Video | <https://www.youtube.com/watch?v=fsuroRYmagw> |
| Motores para videojuegos multiplataforma | GogoGames07. (2023, agosto 1). Descubriendo los Motores de Videojuegos: ¿Cuál es el Mejor? | Video | <https://www.youtube.com/watch?v=XAoSRQKo2ug> |
| Periféricos videojuegos | YANPOL (s.f). Accesorios que DEBES TENER EN TU SETUP. Youtube. | Video | <https://www.youtube.com/playlist?list=PLsdJhEyi2JJCB0TWpvoSmxI4rP19kOMx6> |
| Versiones en *software* | MyLearny Platform. (2023, junio 13). Curso de Git y GitHub - 1. Introducción. Youtube. | Video | <https://www.youtube.com/watch?v=SZpz-Sd326M&list=PLCWlDaiOqlV2WVBRfmDp-T8KIfkmH6hAP> |
| Configuración de GIT | Todo code. (2021). Curso GIT y GITHUB - Tutorial desde CERO. Youtube. | Video | <https://www.youtube.com/playlist?list=PLQxX2eiEaqby-qh4raiKfYyb4T7WyHsfW> |

1. **GLOSARIO:**

|  |  |
| --- | --- |
| **TÉRMINO** | **SIGNIFICADO** |
| **API** | es una abreviatura de *Application Programming Interfaces*, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el *software* de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de *software* a través de un conjunto de reglas. |
| **Dispositivo de Escritorio (Informática)** | se denomina computadora de escritorio, computador de escritorio, ordenador de sobremesa u ordenador fijo1​ a un tipo de computadora personal, diseñada y fabricada para ser instalada en una ubicación estática, como un escritorio o mesa, a diferencia de otras computadoras similares, como la computadora portátil, cuya ubicación es dinámica |
| **Dispositivo Móvil (Informática)** | es un pequeño dispositivo de computación portátil que generalmente incluye una pantalla y un método de entrada (ya sea táctil o teclado en miniatura). Muchos dispositivos móviles tienen sistemas operativos que pueden ejecutar aplicaciones. Las aplicaciones hacen posible para los dispositivos móviles y teléfonos celulares se utilicen como dispositivos para juegos, reproductores multimedia, calculadoras, navegadores, etc. |
| **GIT** | Git es un sistema de control de versiones distribuido que te permite registrar los cambios que haces en tus archivos y volver a versiones anteriores si algo sale mal. Fue diseñado por Linus Torvalds para garantizar la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones que tienen un gran número de archivos de código fuente. |
| **Librería (*Software)*** | es un conjunto de archivos que se utiliza para desarrollar *software*. Suele estar compuesta de código y datos, y su fin es ser utilizada por otros programas de forma totalmente autónoma. |
| **Motor de Videojuegos** | un motor de juegos, o *game engine*, como está llamado en inglés, es un conjunto de herramientas de *software* o API creadas para optimizar el desarrollo de un videojuego. |
| **Página web** | se conoce como página Web, página electrónica o página digital a un documento digital de carácter multimediático (es decir, capaz de incluir audio, video, texto y sus combinaciones), adaptado a los estándares de la *World Wide Web* (WWW) y a la que se puede acceder a través de un navegador Web y una conexión activa a Internet. Se trata del formato básico de contenidos en la red. |
| **Periférico** | un periférico de ordenador es un dispositivo externo al ordenador que está conectado a él pero que no es parte del equipo principal y que permite la entrada y salida de información desde o hacia el propio ordenador. |
| **Sistema Operativo** | es el programa encargado de administrar y gestionar de manera eficiente todos los recursos de un ordenador y otros dispositivos. También se le conoce como *software* de sistema, y su función comienza nada más encender el dispositivo en el que están instalados. |

1. **REFERENCIAS BIBLIOGRÁFICAS:**

Fasheh, I. (2023). *Programando Videojuegos 2.0.*

Lopez Sandoval, Carlo (2022). *UNITY aprende a desarrollar videojuegos*. Edicion Actualizada a Unity 2022.

Sauco, A, y Lozano, E. (2021). El Viaje del Jugador: Guía de Diseño de Videojuegos.

1. **CONTROL DEL DOCUMENTO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia**  ***(Para el SENA indicar Regional y Centro de Formación)*** | **Fecha** |
| **Auror** | Carlos Andrés Cortes | Experto temático | Centro de Diseño e Innovación Tecnológica Industrial- Regional Risaralda | Noviembre 2023 |
|  | Paola Alexandra Moya Peralta | Diseñadora instruccional | Centro Industrial del Diseño y la Manufactura - Regional Santander | Noviembre 2023 |
|  | Rafael Neftalí Lizcano Reyes | Responsable de Línea de Producción | Centro Industrial del Diseño y la Manufactura - Regional Santander | Noviembre 2023 |

1. **CONTROL DE CAMBIOS**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nombre** | **Cargo** | **Dependencia** | **Fecha** | **Razón del Cambio** |
| **Autor (es)** |  |  |  |  |  |