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 cuáles 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.

Diciembre 2023

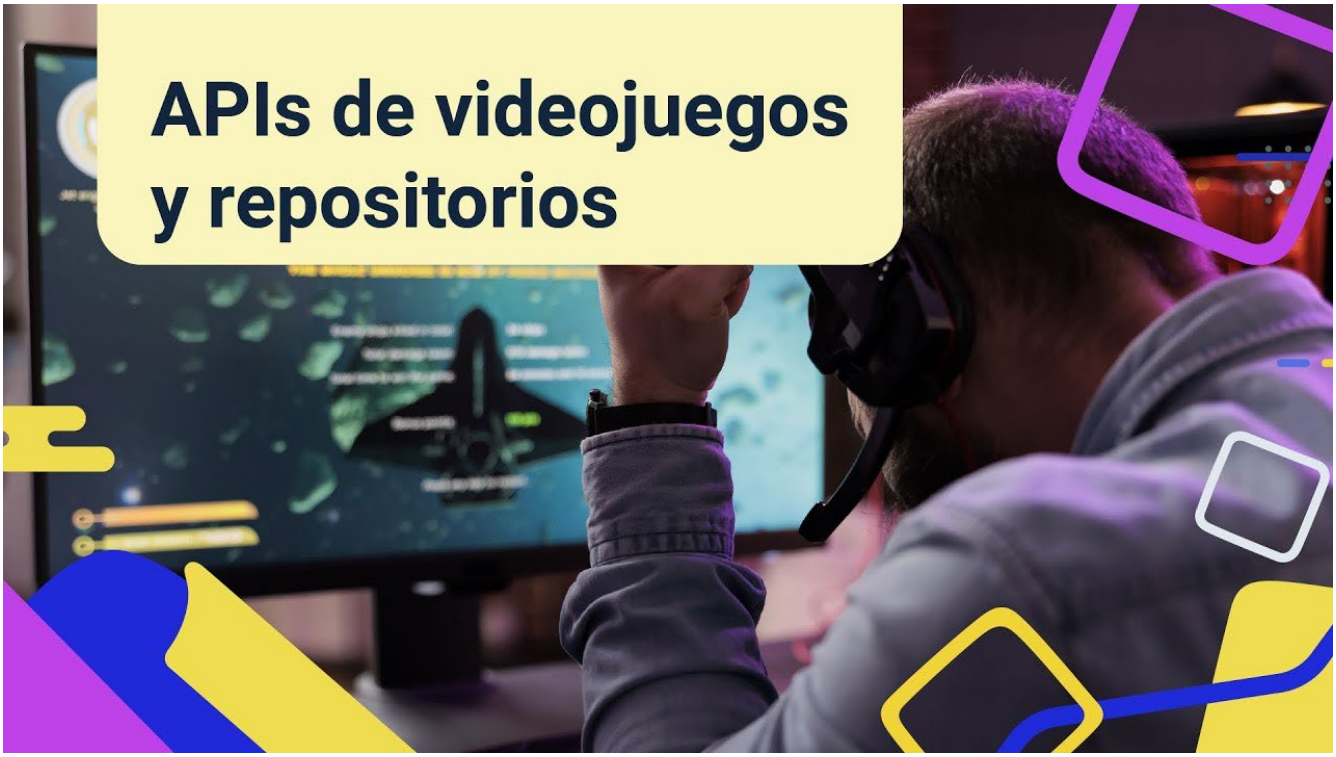
Grupo de Ejecución de la Formación Virtual

Tabla de contenido

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 1. APIs de videojuegos y repositorios



Enlace de reproducción del video

Síntesis del video: APIs de videojuegos y repositorios
Este componente formativo aborda el desarrollo de videojuegos, trabajando en los sistemas operativos, APIs y librerías que hacen posible la creación de juegos. Los

sistemas operativos se convierten en la base de todo desarrollo de “software”, incluyendo los juegos.

Las APIs, esas herramientas vitales que permiten la interacción entre diferentes “software”, hacen posible que los juegos funcionen de manera fluida y eficiente. Asimismo, las librerías específicas para videojuegos son cruciales, ya que ofrecen funciones que se pueden reutilizar en diversos proyectos, tanto en plataformas de escritorio como en dispositivos móviles.

También se habla sobre la instalación de estas herramientas, un paso esencial para cualquier desarrollador de juegos. Se verán los periféricos para videojuegos, esos dispositivos de “hardware” que hacen que nuestra experiencia de juego sea aún más envolvente.

Finalmente, se abordan los repositorios GIT. Aquí, profundizaremos en cómo gestionar las versiones de nuestro “software”, cómo configurar GIT y cómo realizar operaciones básicas en repositorios. Esto es vital para mantener nuestro código organizado y eficiente.

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 prescritos, 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.

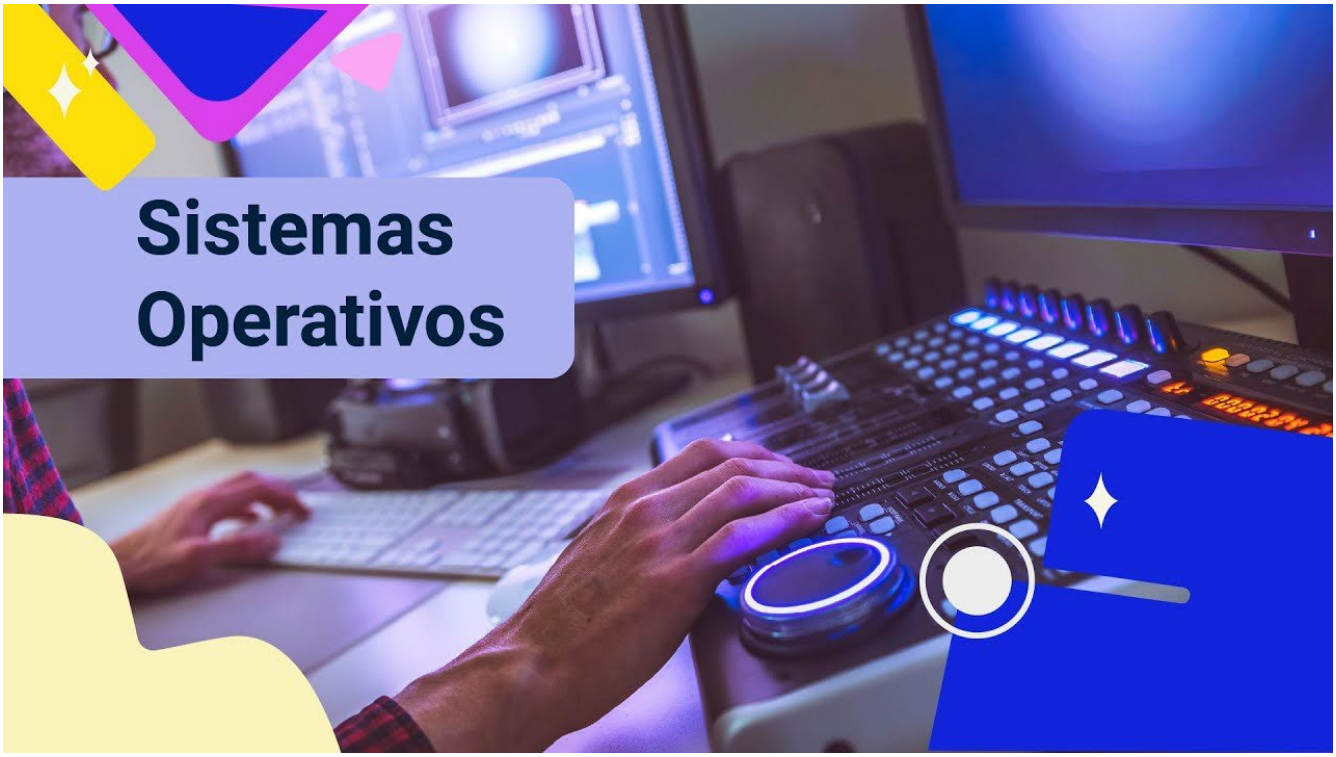
1.1. Sistemas Operativos

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 2. Sistemas Operativos



Enlace de reproducción del video

Síntesis del video: Sistemas Operativos
En el mundo de la informática, los sistemas operativos juegan un papel fundamental como la interfaz principal entre el “hardware” de una computadora y las aplicaciones que se usan diariamente. Los sistemas operativos más populares son:

- Windows es el sistema operativo insignia de Microsoft, el estándar de facto para computadoras domésticas y comerciales. Introducido en 1985, el sistema operativo basado en GUI se ha lanzado en muchas versiones desde entonces.
- Mac OS es el sistema operativo para la línea Macintosh de PC y estaciones de trabajo de Apple.
- Unix es un sistema operativo multiusuario diseñado para brindar flexibilidad y adaptabilidad. Desarrollado originalmente en la década de 1970, Unix fue uno de los primeros sistemas operativos escrito en lenguaje C.
- Linux es un sistema operativo similar a Unix que fue diseñado para brindar a los usuarios de PC una alternativa gratuita o de bajo costo. Linux tiene la reputación de ser un sistema eficiente y de rápido rendimiento.

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

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.

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.

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.

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.

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:

- **API de SOAP.** Estas API utilizan el protocolo simple de acceso a objetos. El cliente y el servidor intercambian mensajes mediante XML. Se trata de una API menos flexible que era más popular en el pasado.
- **API de RPC.** Estas API se denominan llamadas a procedimientos remotos. El cliente completa una función (o procedimiento) en el servidor, y el servidor devuelve el resultado al cliente.
- **API de "WebSocket".** La API de "WebSocket" es otro desarrollo moderno de la API web que utiliza objetos JSON para transmitir datos. La API de "WebSocket" admite la comunicación bidireccional entre las aplicaciones cliente y el servidor. El servidor puede enviar mensajes de devolución de llamada a los clientes conectados, por lo que es más eficiente que la API de REST.

- **API de REST.** Estas son las API más populares y flexibles que se encuentran en la web actualmente. El cliente envía las solicitudes al servidor como datos. El servidor utiliza esta entrada del cliente para iniciar funciones internas y devuelve los datos de salida al cliente.

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:

- **Unity.** Es ampliamente reconocido por su versatilidad y facilidad de uso. Permite a los desarrolladores crear juegos para múltiples plataformas, incluyendo Windows, macOS, Linux, iOS, Android, y consolas. Enlace de Unity:
- **Unreal Engine.** Desarrollado por Epic Games, es conocido por su potente renderizado gráfico y se utiliza en muchos juegos AAA. Es compatible con una variedad de plataformas. Enlace de Unreal Engine:
- **Gdevelop.** Es un motor de juegos de código abierto, enfocado en ser fácil de usar para principiantes, y permite la creación de juegos sin necesidad de conocimientos de programación. Compatible con varias plataformas, incluyendo Windows, Linux y dispositivos móviles. https://gdevelop.io/es-es

- **O3DE (Open 3D Engine).** Un motor de juego de código abierto que se centra en la creación de mundos 3D y juegos inmersivos. Ofrece una amplia gama de herramientas y es compatible con múltiples plataformas. https://o3de.org/
- **GameMaker.** Similar a GameMaker Studio, es una plataforma de desarrollo que destaca por su sencillez y eficacia, permitiendo la creación rápida de juegos para diversas plataformas. https://gamemaker.io/es

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:

Python Arcade. Una biblioteca moderna, fácil de usar, para el desarrollo de juegos 2D en Python. Ofrece soporte para gráficos, animaciones y sonido, siendo una opción ideal para principiantes y educadores. https://api.arcade.academy/en/latest/

Panda3D. Un motor de juego y una biblioteca de renderizado 3D para Python y C++. Es potente y flexible, adecuado para juegos más complejos y aplicaciones 3D.

https://docs.panda3d.org/1.10/python/index

Harfang. Una plataforma de alto nivel para la creación de aplicaciones 2D y 3D. Ofrece un amplio conjunto de características para gráficos, física y más, ideal para proyectos avanzados. https://www.harfang3d.com/en_US/

Ren'Py. Especializado en la creación de novelas visuales y juegos basados en narrativas. Es muy popular entre los desarrolladores de novelas visuales por su facilidad de uso y flexibilidad. https://www.renpy.org/

Kivy. Una biblioteca de Python para desarrollar aplicaciones multitáctiles. Aunque no es específicamente para juegos, su soporte para gráficos y UI la hace útil para ciertos tipos de juegos. https://kivy.org/

Pygame. Una de las bibliotecas más populares para el desarrollo de videojuegos en Python. Es ideal para juegos 2D y es amigable para principiantes, con una gran comunidad y recursos de aprendizaje. https://www.pygame.org/news

Cocos2d. Un marco de trabajo para construir juegos, interfaces gráficas y proyectos multimedia en Python. Es bien conocido por su facilidad de uso y su capacidad para crear juegos 2D. https://pypi.org/project/cocos2d/

Cochinillo (Pyglet). Una biblioteca para el desarrollo de juegos y otras aplicaciones visuales en Python. Es una base sólida para juegos 2D/3D, aplicaciones multimedia y otros proyectos gráficos. https://pyglet.org/

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:

- **Phaser.** Una biblioteca de juegos rápida y gratuita para la creación de juegos HTML5 tanto en escritorio como en dispositivos móviles. Es especialmente popular por su flexibilidad y facilidad de uso, ideal para juegos 2D.
- **Three.js.** Una biblioteca JavaScript para crear y mostrar gráficos 3D animados en un navegador web, usando WebGL. Es ampliamente utilizada para juegos, visualizaciones y experiencias interactivas en 3D.
- **Pixi.js.** Un motor de renderizado 2D que permite crear contenido interactivo y hermoso. Es conocido por su velocidad y su capacidad para trabajar con WebGL y Canvas.
- **Babylon.js.** Un potente motor de juego 3D para la web, que facilita la creación de experiencias interactivas y juegos en 3D con soporte para WebGL, WebXR y otras tecnologías web modernas.
- **PlayCanvas.** Un motor de juego 3D basado en HTML5 y WebGL. Ofrece un entorno de desarrollo integrado en la nube y es conocido por su rendimiento y accesibilidad.
- **Construct 3.** Un editor de juegos basado en la web que permite a los usuarios crear juegos sin escribir código. Es muy accesible para principiantes y es popular para la creación de juegos 2D.

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:

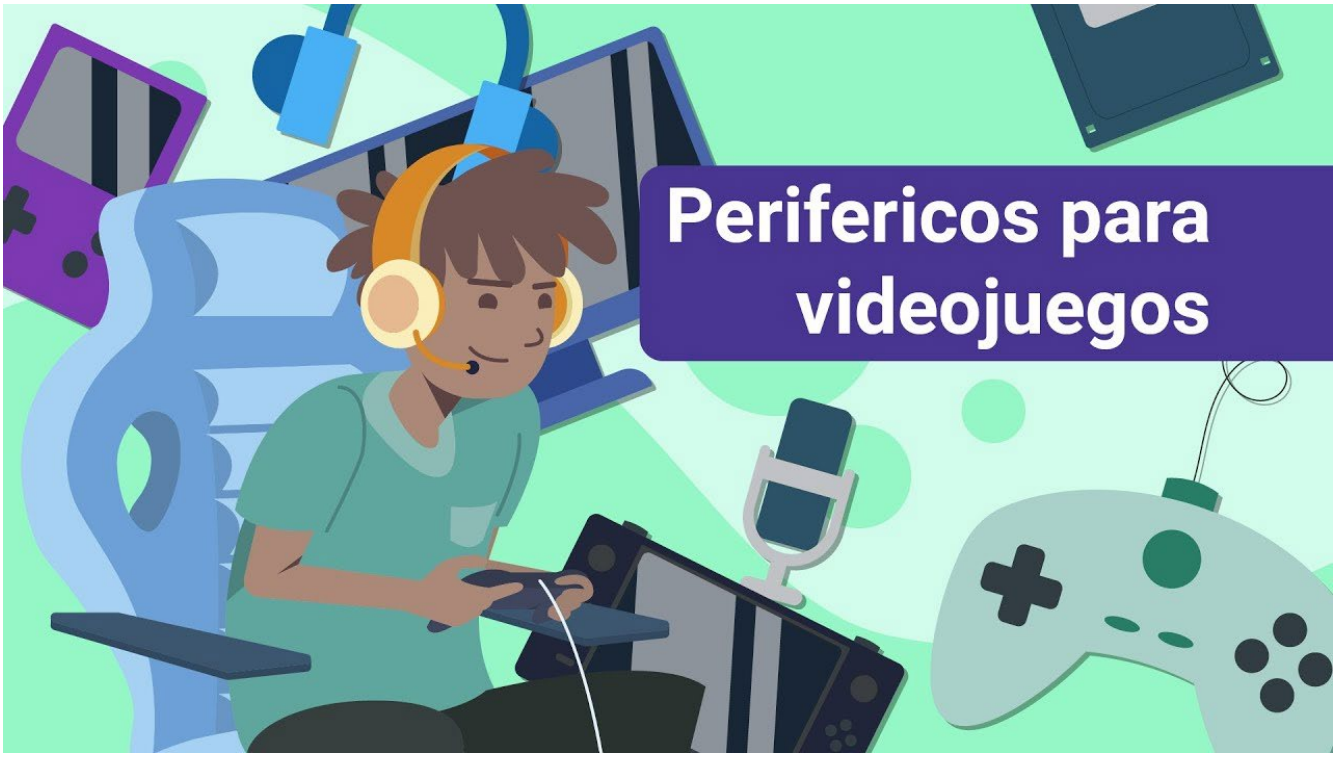
- **Android Game Development Kit (AGDK).** Una colección de herramientas y bibliotecas proporcionadas por Google específicamente para el desarrollo de juegos en Android. Incluye soporte para C++ y Kotlin, y ofrece herramientas para mejorar el rendimiento y la calidad de los juegos en dispositivos Android.
- **GameSalad.** Una plataforma para el desarrollo de juegos que permite a los usuarios crear juegos para iOS, Android, HTML5 y más, sin necesidad de conocimientos de programación avanzados. Es popular entre educadores y principiantes por su interfaz de arrastrar y soltar.
- **Stencyl.** Similar a GameSalad, Stencyl es una herramienta que permite a los desarrolladores crear juegos sin necesidad de escribir código, utilizando un sistema de 'arrastrar y soltar'. Soporta la creación de juegos para iOS, Android, Windows, Mac, y Linux.

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 3. Periféricos para videojuego



Enlace de reproducción del video

Síntesis del video: Periféricos para videojuego

Los periféricos para videojuegos son dispositivos que se conectan a una computadora o consola de juegos para mejorar la experiencia de juego, en el que se incluyen:

- **Auriculares:** estos mejoran significativamente la experiencia de audio en diversas situaciones. Por ello, al desarrollar un videojuego, es crucial enfocarse en ofrecer una calidad de audio que permita a los jugadores sumergirse plenamente en él.
- **Teclado:** se utilizan convencionalmente para introducir comandos básicos o manejar programas comunes en un computador, sin importar su tecnología. Sin embargo, en los videojuegos, los teclados pueden convertirse en el principal dispositivo de entrada. Esto hace que sea importante permitir a los jugadores configurar los controles de teclado según consideren que les permitirá un mejor rendimiento en el juego.
- **Ratón:** en el mundo de los videojuegos, la precisión es clave. En este contexto, el ratón se convierte en el dispositivo más utilizado. Al igual que con el teclado, se debe permitir a los jugadores configurar el ratón. Es importante considerar que los ratones "gamer" suelen tener más de tres botones, lo cual es superior a lo tradicional, y debe permitirse su configuración en el videojuego.
- **"Joystick"**: la integración de controladores mejora la experiencia de juego para los usuarios. Los mandos de PC se han convertido en un periférico esencial para muchos jugadores, especialmente aquellos acostumbrados a jugar en consolas.

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

- **Compatibilidad:** no todos los controladores funcionan con PC para juegos.
- **Con cable o inalámbrico:** los controladores inalámbricos ofrecen mayor libertad de movimiento, pero los controladores con cable generalmente tienen menos retraso de entrada y una conexión más confiable.
- **Ergonomía:** la comodidad es fundamental, sobre todo si se piensa utilizar el mando durante muchas horas.
- **Diseño de botones:** algunos controladores tienen diferentes configuraciones de botones y algunos pueden ser más útiles para tipos específicos de juegos.
- **Marca:** considere comprar marcas reconocidas que produzcan controladores de juegos de alta calidad.

Si el videojuego que se quiere construir tiene como esencia ser orientado 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 utilizarlas.

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.

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.

a. **Instalar GIT.** Instale la última versión de GIT en su dispositivo. Necesitará tener GIT instalado para funcionar con su repositorio de GitHub. Hay varias

formas de hacerlo, así que siga las recomendaciones del sitio web de GIT. El "software" GIT es gratuito.

b. **Regístrese en GitHub.** Después de instalar GIT, vaya al sitio web de GitHub y cree una cuenta con su dirección de correo electrónico.
c. **Inicie un repositorio.** Una vez que su cuenta de GITHub esté configurada, accederá a su panel de control. Para iniciar su primer repositorio, haga clic en Crear repositorio. Esto le permite mantener todo el código de su nuevo proyecto de GITHub en un solo lugar.
d. **Nombre el proyecto.** En la pantalla Crear un nuevo repositorio, ingrese el nombre de su repositorio y una descripción opcional (puede cambiar ambos más adelante).
e. **Agregue detalles del proyecto.** En la misma pantalla, agregue un archivo README (un archivo de texto que describe su proyecto y una de las mejores prácticas de desarrollo), un .gitignore (que elimina archivos irrelevantes como .DS_Store) y una licencia para su proyecto.
f. **Crear el repositorio.** Crear el repositorio. Haga clic en Crear repositorio. Será llevado a la página de su repositorio principal, que enumera sus archivos.
g. **Crear una copia local de su repositorio.** Ahora creará una copia local de su repositorio de GITHub (o en términos de GITHub, "clonará" su repositorio) donde editará sus archivos y enviará sus cambios. En la página principal de su repositorio, haga clic en el botón verde Código y luego copie la URL HTTPS de su repositorio.
h. **Elegir un directorio.** Abra su terminal y navegue hasta el directorio donde desea colocar su copia del repositorio.

i. **Pegar la URL de su repositorio.** En la terminal, ingresa GIT clone. Después de esto, pegue la URL del repositorio que copió anteriormente.

j. **Clonar y verificar el repositorio copiado.** Presione Enter para clonar el repositorio. Verá un nuevo archivo agregado a su sistema de archivos local con el nombre de su repositorio. Si abre este archivo, verá que contiene los archivos en su repositorio de GitHub. Estas son versiones copiadas de los archivos de su repositorio que puede editar y luego enviar de regreso a su repositorio.

Nota. Para obtener una visión más detallada de la instalación, puede revisarla en el siguiente recurso. **Enlace Curso GIT y GITHUB.**

3.4. Operaciones en repositorio

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

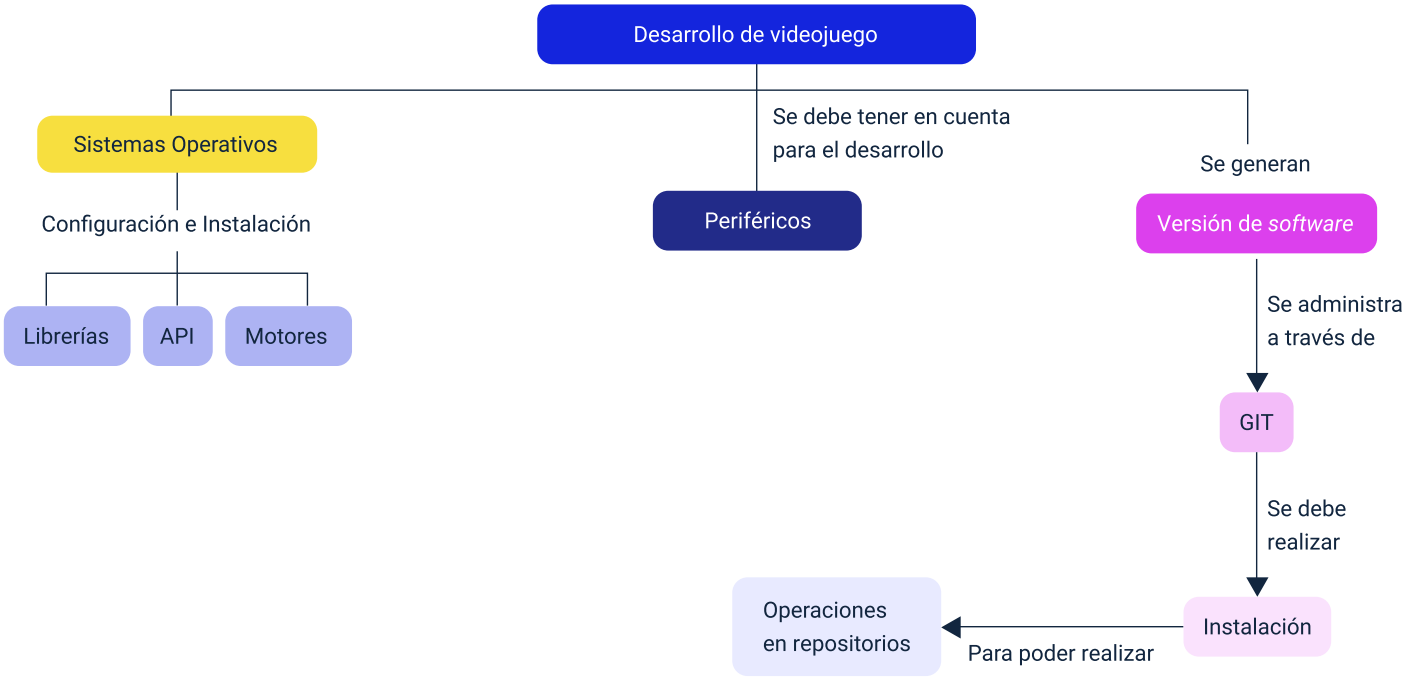
ejecutar el comando de confirmación, se utiliza “git add” para agregar archivos nuevos o modificados.

- **“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.

Síntesis

A continuación, se muestra un mapa conceptual con los elementos más importantes desarrollados en este componente.



Descripción del esquema: Síntesis
La síntesis representa un diagrama relacionado con el desarrollo de videojuegos. Se destacan elementos clave como los sistemas operativos, que requieren configuración e instalación, y se subdividen en categorías como librerías, API y motores. Los periféricos son componentes importantes a tener en cuenta en el desarrollo. Además, se enfatiza en la generación y administración de versiones de software, donde se utiliza GIT para la gestión. Las operaciones en repositorios son presentadas como una actividad necesaria, vinculada con el proceso de instalación.

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
Sistemas Operativos	El Salón de Informática. (2021, febrero 18). Sisemas operativos | ¿Qué son?¿Cómo Funcionan? | Tipos de Sistemas Operativos | Explicación Fácil. 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=PLCWlDaiO
Configuración de GIT	Todo code. (2021). Curso GIT y GITHUB - Tutorial desde CERO. Youtube.	Video	https://www.youtube.com/playlist?list=PLQxX2eiEaqby-qh4raiKfYyb4T7WyHsfW

Glosario

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 fijo 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 que 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.

Referencias bibliográficas

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

López Sandoval, Carlo (2022). UNITY aprende a desarrollar videojuegos. Edición Actualizada a Unity 2022.

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

Créditos

Nombre	Cargo	Regional y Centro de Formación
Claudia Patricia Aristizábal	Responsable del Ecosistema	Dirección General
Rafael Neftalí Lizcano Reyes	Responsable de Línea de Producción	Centro Industrial del Diseño y la Manufactura - Regional Santander
Carlos Andrés Cortes	Experto Temático	Centro de Diseño e Innovación Tecnológica Industrial- Regional Risaralda
Paola Alexandra Moya Peralta	Diseñador Instruccional	Centro Industrial del Diseño y la Manufactura - Regional Santander
Blanca Flor Tinoco	Diseñador de Contenidos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Emilsen Alfonso Bautista	Desarrollador Fullstack	Centro Industrial del Diseño y la Manufactura - Regional Santander
Carmen Alicia Martínez Torres	Animador y Productor Audiovisual	Centro Industrial del Diseño y la Manufactura - Regional Santander
Carlos Eduardo Garavito Parada	Animador y Productor Audiovisual	Centro Industrial del Diseño y la Manufactura - Regional Santander
Camilo Andrés Bolaño Rey	Locución	Centro Industrial del Diseño y la Manufactura - Regional Santander
Zuleidy María Ruíz Torres	Validación de Recursos Educativos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Luis Gabriel Urueta Álvarez	Validación de Recursos Educativos Digitales	Centro Industrial del Diseño y la Manufactura - Regional Santander
Daniel Ricardo Mutis Gómez	Evaluador para Contenidos Inclusivos y Accesibles	Centro Industrial del Diseño y la Manufactura - Regional Santander