

Construcción API

**Breve descripción:**

En este componente formativo se abordarán los conceptos relacionados con la configuración del entorno de desarrollo como espacio de trabajo para la API REST. Se revisarán las buenas prácticas de los estándares de codificación para el desarrollo de la API, la cual será construida con las tecnologías de Node.js, Express y MongoDB y se abordará el renting de la API creada, a través de la herramienta Postman.

**Junio 2024**

Tabla de contenido

[Introducción 3](#_Toc168655595)

[1. La internet y la web 4](#_Toc168655596)

[1.1. Conceptos sobre el entorno de desarrollo integrado - IDE 4](#_Toc168655597)

[1.2. Instalación del Visual Studio Code 6](#_Toc168655598)

[1.3. Instalación y configuración del Node.js y npm 6](#_Toc168655599)

[1.4. Instalación de módulo 7](#_Toc168655600)

[2. Codificación de la API REST con Node.js 10](#_Toc168655601)

[2.1. Buenas prácticas en la codificación de Node.js 10](#_Toc168655602)

[3. Testing de la API REST con Postman 13](#_Toc168655603)

[Síntesis 14](#_Toc168655604)

[Material complementario 15](#_Toc168655605)

[Glosario 16](#_Toc168655606)

[Referencias bibliográficas 18](#_Toc168655607)

[Créditos 20](#_Toc168655608)

Introducción

En este componente formativo se estudiarán los conceptos y pasos para el desarrollo de una API REST, utilizando las tecnologías del back-end como Node.js, Express y el sistema de bases de datos no relacional como lo es MongoDB. De esta forma se inicia con la puesta a punto del entorno de desarrollo con Visual Studio Code, el cual será el espacio de trabajo donde se construirá la API. Adicionalmente, se comprenderá la importancia del uso de los comandos (en la terminal) para la instalación y configuración de paquetes necesarios para el correcto funcionamiento del proyecto; de igual manera, se ilustran algunas buenas prácticas de desarrollo y herramientas que harán más prolijo el código.

Una vez construida la API se inicia con un testing a través de la herramienta Postman, que es muy utilizada para este tipo de desarrollos. Así, se corroboran las respuestas y solicitudes de un tipo de proyecto de esta naturaleza.

# La internet y la web

Para todo proyecto software es recomendable alistar las herramientas necesarias y mínimas para iniciar con un desarrollo. En ese sentido, dependiendo del alcance, la tecnología y/o lenguaje se sugiere definir muy bien este tipo de recursos para realizar el ejercicio. En este orden de ideas se abordarán algunos conceptos fundamentales.

## Conceptos sobre el entorno de desarrollo integrado - IDE

Como primera medida, es necesario tener en cuenta la definición de un entorno de desarrollo integrado IDE, que es un espacio de trabajo basado en tecnologías y software, que sirve de plataforma para el diseño de aplicaciones que componen una serie de herramientas para el desarrollador de software. Un IDE cuenta con las siguientes características:

* **Editor de código fuente**

Es un editor de texto que ayuda a escribir en el lenguaje de programación definido; esta herramienta apoya la sintaxis o construcción de instrucciones, para que se realicen de la manera correcta. Una de sus funcionalidades más importantes es resaltar ciertas partes del código que se están desarrollando, para tener una referencia sobre el flujo o línea lógica del desarrollo de la aplicación. De manera adicional, hay editores que ayudan a autocompletar ciertas funciones ya preestablecidas del lenguaje.

* **Automatización de compilaciones locales**

Es un recurso que permite automatizar tareas sencillas e iterativas como parte de la creación de una compilación local del software para su uso por parte del desarrollador.

* **Depurador**

“Programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica” (Red Hat, 2021). Además, hay entornos que ayudan con varias opciones de la sintaxis para corregir el código.

Para el caso de este componente formativo se utilizará el IDE Visual Studio Code por las siguientes razones:

* Visual Studio Code resalta las palabras clave del código en diferentes colores para ayudar a identificar fácilmente los patrones de codificación y aprender más rápido.
* A medida que se codifica, Visual Studio Code brinda sugerencias para completar líneas de código y soluciones rápidas para errores comunes.
* Se aprovecha porque este entorno tiene incorporada la terminal, lo cual hace mucho más fácil el desarrollo, sobre todo con las tecnologías de Node.js, porque trae los paquetes necesarios para el proyecto.

En el proceso de alistamiento del entorno para iniciar el proyecto de desarrollo de la API, se deberá contar con los siguientes recursos:

* Visual Studio Code
* Node.js
* Express.js
* MongoDB

## Instalación del Visual Studio Code

Para la instalación se deben seguir los siguientes pasos (para este caso se ha tomado como plataforma el sistema operativo Windows 10):

* **Paso 1**

Descargar el instalador de la página principal para Visual Studio Code en Windows.

Enlace: <https://code.visualstudio.com/docs?dv=win>

* **Paso 2**

Una vez descargado, se debe correr el instalador y seguir con el wizard o paneles de instalación. Luego dar clic en finalizar.

Por defecto, el Visual Studio Code se instala en la siguiente ruta:

C:\users\{username}\AppData\Local\Programs\Microsoft VS Code

## Instalación y configuración del Node.js y npm

Node.js está considerado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, de igual manera, está diseñado para crear aplicaciones network escalables y pensado en principio como tecnología back-end (node.js.org, 2021).

Los pasos para instalar el Node.js son los siguientes:

* **Paso 1**

Se ingresa a la página oficial, se selecciona el sistema operativo, y hace clic en descargar.

Enlace: <https://nodejs.org/en/download>

* **Paso 2**

Una vez descargado, se debe correr el instalador y seguir con el wizard o paneles de instalación.

* **Paso 3**

Posterior a la instalación, se verifica que tanto el Node.js como el npm estén instalados, para eso se utiliza la línea de comandos en la terminal del Visual Studio Code: node -v o npm – v.

## Instalación de módulo

Un paquete en Node.js contiene todos los archivos que se necesitan para instalar un módulo; los módulos son bibliotecas de JavaScript que se pueden incluir en el proyecto. Instalar módulos es una de las principales tareas que debe aprender a hacer al comenzar con el administrador de paquetes Node. El comando para instalar un módulo en el directorio actual es el siguiente:

$ npm install < módulo >

$ npm i < módulo >

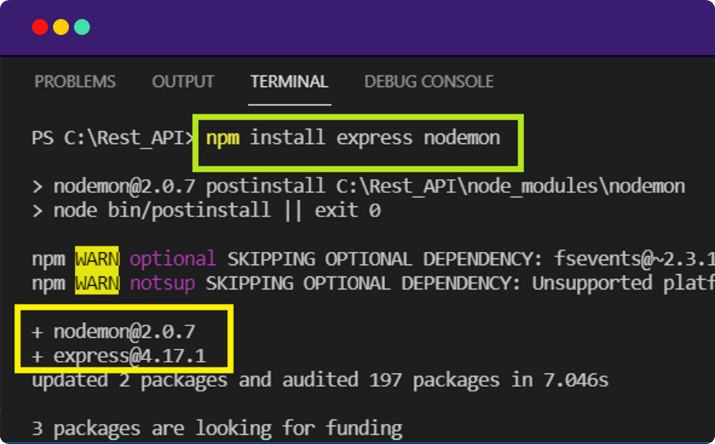
El primer módulo a instalar es Express, el cual es el framework web más popular de Node y proporciona mecanismos para (Mozilla, 2021).

* Escritura de manejadores de peticiones con diferentes verbos HTTP, en diferentes caminos URL (rutas).
* Integración con motores de renderización de “vistas”, para generar respuestas mediante la introducción de datos en plantillas.
* Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
* Añadir el procesamiento de peticiones “middleware” adicional en cualquier punto dentro de la tubería de manejo de la petición.

El segundo módulo para instalar es nodemon, que es una utilidad que monitorea los cambios en el código fuente que se está desarrollando y de manera automática reinicia el servidor. Es una herramienta muy útil para el desarrollo de aplicaciones en Node.js (Gómez, 2017).

A continuación, se muestra la línea de comando que se debe ejecutar desde la terminal del Visual Studio Code.

1. Comando para la instalación de Express y nodemon



Hasta esta parte se ha abordado la instalación y configuración del espacio de trabajo para iniciar con el desarrollo de la API. No obstante, es necesario revisar las buenas prácticas de codificación para que el desarrollo no solo sea prolijo en su estructura, sino que también siga estándares de codificación, permitiendo que el código tenga un alto desempeño.

# Codificación de la API REST con Node.js

En este tema, se inicia con el desarrollo de la API REST, por lo que se verá la codificación de cada uno de los archivos que componen el proyecto. Para esto, es importante tener en cuenta dos cosas, la primera es realizar buenas prácticas de codificación, permitiendo que el código quede legible y mucho más limpio al momento de documentar y la segunda, es entender la estructura de directorios del proyecto, la cual va a dar el orden al proceso de búsqueda e indexación de las rutas y recursos que se necesitarán al momento del desarrollo.

## Buenas prácticas en la codificación de Node.js

El objetivo de seguir estándares de codificación, es alcanzar un código mantenible y se considera mantenible cuando el código es legible, consistente, predictivo y documentado. En la práctica, la corrección de errores y modificación del código desarrollado de un producto software toma demasiado tiempo y se llega algunas veces a que no es posible realizar del todo el desarrollo, por lo que se toma la decisión de iniciar de cero, debido a las malas prácticas. (Choque, 2016).

A continuación, se indican las buenas prácticas enunciadas por uno de los mejores servicios de control de versiones y desarrollo de software colaborativo basado en Git, llamada GitLab. Esta comunidad sigue de igual manera estándares como Google JavaScript Style Guide (Whyte, 2021).

1. **Código en general**

* Utilizar el punto y coma (;) como final de instrucción. Mejora la lectura y evita conflictos con herramientas de verificación de código.
* Para delimitar y diferenciar cadenas de texto, se recomienda utilizar comillas simples con excepción de JSON.
* La codificación en la sintaxis para los nombres de variables o procedimientos es lowerCamelCase, la cual establece que la letra inicial debe ser minúscula y las palabras siguientes deben tener la inicial en mayúscula, y el resto en minúscula, por ejemplo, var logFileName.
* La codificación en la sintaxis de las clases e interfaces es UpperCamelCase que establece que la letra inicial debe ser mayúscula y las palabras siguientes deben tener la inicial en mayúscula, y el resto en minúscula.
* Evitar escribir prefijos del tipo de dato a las variables. Se recuerda que las variables en JavaScript son sin tipo, es decir, que toman su tipo de dato dependiendo del valor.

1. **Bloques de código**

* Las líneas de código que tienen alguna relación, deben estar agrupadas sin espacio entre sus líneas.
* La cantidad máxima de caracteres que debe tener una línea de código, debería ser de 120.
* Los bloques de código que son alusivos a las clases, procedimientos y estructuras de control, deben estar identados con 4 espacios, que es lo mismo a una tabulación.

1. **Procedimientos**

La cantidad de líneas que se recomiendan para un procedimiento es de 20, salvo casos excepcionales. En el caso que un procedimiento supere las 20 líneas, se debe evaluar la división del proceso en procesos más pequeños. Se debe tener en cuenta que un procedimiento debe realizar una tarea.

1. **Manejo de errores y excepciones**

* Como una buena práctica los errores y/o excepciones posibles, deben ser capturados (catch) y se debe lanzar (throw) un mensaje preciso, con el fin de evitar la finalización abrupta del programa.
* En caso que el error requiera alguna acción por parte del usuario, como puede ser continuar, cancelar u otras opciones, el sistema debe informar al usuario sobre las consecuencias de cada elección.

1. **Comentarios**

* Los comentarios forman parte de la documentación de un proyecto, de hecho, esta buena práctica hace parte de la base del manual técnico de un proyecto de software.
* Su principal ventaja es que, a partir de ellos, se logra reducir el costo de transferencia de conocimientos entre equipos de desarrollo, dado que es posible intervenir de manera más rápida en la continuación del desarrollo.

**Desarrollo de la API REST**

Lo invitamos a revisar el PDF “**Desarrollo API REST**”, en el cual se describen de manera sucinta, los pasos requeridos para el desarrollo de la API REST con Node.js, que encuentra en la carpeta Anexos.

# Testing de la API REST con Postman

En la definición de la arquitectura de despliegue de aplicaciones y servicios, es muy importante la definición de los componentes desde el punto de vista de la plataforma tecnológica que serán utilizados en el momento de iniciar el proceso de desarrollo y en el de hacer el despliegue o puesta en producción, para lo cual el equipo de desarrollo deberá elegir qué tipo de plataforma tecnológica será la utilizada.

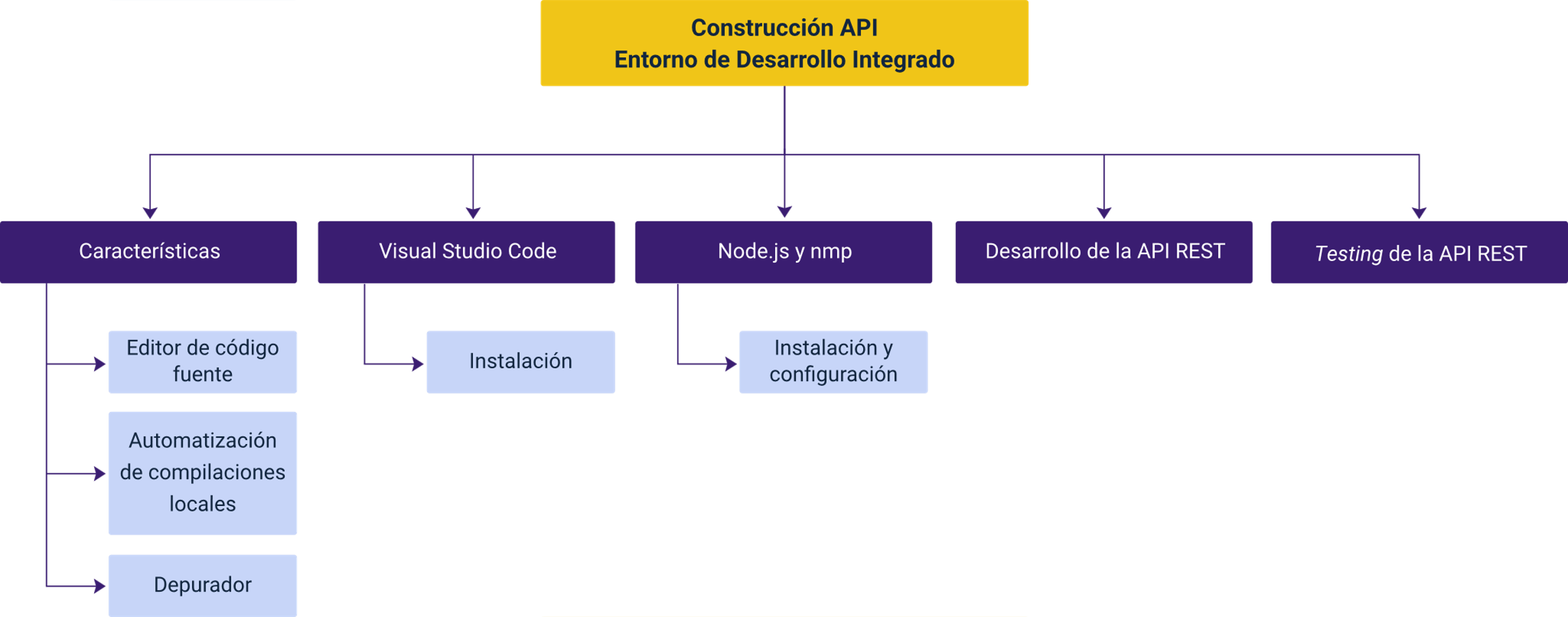
Si bien la plataforma tecnológica exige la selección de varios componentes como sistema operativo, servidor web, gestor de base de datos y lenguaje de programación, en la actualidad se debe tener claro qué tipo de despliegue se requiere para la construcción local del producto en las máquinas de los equipos de desarrollo y para el despliegue en producción (generalmente debe ser el mismo).

**Instalación Postman**

Lo invitamos a revisar el PDF “**Instalación herramienta Postman”**, en el cual se explica su instalación, que encuentra en la carpeta Anexos.

Síntesis

A continuación, se presenta una síntesis de la temática estudiada en el componente formativo:



Material complementario

| Tema | Referencia | Tipo de material | Enlace del recurso |
| --- | --- | --- | --- |
| Entidades de información digital | Coding, G. (2020). Cómo instalar Node.js en Windows 2024 [video]. YouTube. | Video | <https://www.youtube.com/watch?v=BgtB31gXkoA> |
| Información digital | Dominicode. (2018). "Postman en Español: ¡El tutorial definitivo para el éxito en las pruebas API! [video]. YouTube. | Video | <https://www.youtube.com/watch?v=qsejysrhJiU> |
| Entidades de información digital | DiSimone, K. (2020). mLab to Atlas for existing NS users [video]. YouTube. | Video | <https://www.youtube.com/watch?v=hXSLBACjYQo> |

Glosario

**Identación**: dentro del contexto de desarrollo de software identar es un término que significa mover un bloque de código hacia la derecha insertando espacios o tabulaciones, para así separarlo del margen izquierdo y distinguirlo mejor del texto adyacente, es decir, organizar mejor el código. La identación se considera un principio fundamental de buena práctica de la programación de software.

**JSON**: Java Script Object Notation (JSON) es un formato basado en texto estándar que se utiliza para representar datos estructurados en la sintaxis de objetos de JavaScript. Se utiliza principalmente para transmitir datos en aplicaciones web (como es el caso de enviar algunos datos desde el servidor al cliente, por lo tanto, estos datos pueden ser mostrados en páginas web o viceversa).

**Localhost**: en el contexto de las redes de computadoras, localhost es una computadora que ejecuta un programa. La computadora que se tiene al frente funciona también como un servidor virtual; este modo es comúnmente utilizado para realizar pruebas en el desarrollo web.

**Manejadores o handlers de eventos**: en la programación con JavaScript un evento por sí mismo carece de utilidad; para que los eventos resulten útiles o aplicables se debe asociar funciones o código JavaScript a cada evento. De esta forma, cuando se produce un evento de manera inmediata se ejecuta el código indicado, por lo que la aplicación responde ante cualquier evento que se produzca durante su ejecución. Las funciones o código JavaScript que se realizan para cada evento se denominan "manejador de eventos”.

**Middleware**: es un bloque de código que se ejecuta entre la petición que hace el usuario (request) hasta que esa petición llega al servidor. De esta manera, son funciones que tienen acceso al objeto de solicitud (req), el objeto de respuesta (res) y la siguiente función de middleware en el ciclo de solicitud-respuesta de la aplicación.

Referencias bibliográficas

Choque, J. C. (2016). Adición de estándares de codificación node.js. GitLab. <https://gitlab.agetic.gob.bo/dbarra/bolivialibre/commit/1ff84b6ae70f0e9436f6e6ddc7f3e060bc9556ad>

Chrome web store. (2021). Postman. <https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=es-419>

Express. (2021). Direccionamiento básico. <https://expressjs.com/es/starter/basic-routing.html>

Gómez, D. (2017). Nodemon. Vortexbird. <https://vortexbird.com/nodemon/#:%7E:text=Nodemon%20es%20una%20utilidad%20que,desarrollo%20de%20aplicaciones%20en%20nodojs>

López, A. (2021). Qué es Postman y para qué sirve. OpenWebinars. <https://openwebinars.net/blog/que-es-postman/>

Mozilla. (2021). Await. MDN Web Docs. <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/await>

Mozilla. (2021). Introducción a Express/Node. MDN Web Docs. <https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction>

Nodejs.org. (2021). Acerca de Node.js. Node.js. <https://nodejs.org/es/about/>

npm. (2021). About npm. <https://www.npmjs.com/about>

Red Hat. (2021). El concepto de IDE. <https://www.redhat.com/es/topics/middleware/what-is-ide>

Whyte, A. (2021). Google JavaScript Style Guide. <https://google.github.io/styleguide/javascriptguide.xml>

Créditos

| Nombre | Cargo | Centro de Formación y Regional |
| --- | --- | --- |
| Milady Tatiana Villamil Castellanos | Responsable del Ecosistema | Dirección General |
| Olga Constanza Bermúdez Jaimes | Responsable de Línea de Producción | Centro de Servicios de Salud - Regional Antioquia |
| Zulema Yidney León Escobar | Experta Temática | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Jonathan Guerrero Astaiza | Experto Temático | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Ana Catalina Córdoba Sus | Evaluadora Instruccional | Centro de Servicios de Salud - Regional Antioquia |
| Yerson Fabián Zárate Saavedra | Diseñador de Contenidos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Luis Jesús Pérez Madariaga | Desarrollador Fullstack | Centro de Servicios de Salud - Regional Antioquia |
| Edgar Mauricio Cortés García | Actividad Didáctica | Centro de Servicios de Salud - Regional Antioquia |
| Jaime Hernán Tejada Llano | Validador de Recursos Educativos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Margarita Marcela Medrano Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |
| Daniel Ricardo Mutis Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |