**DESARROLLO E IMPLEMENTACIÓN DE UNA API WEB PARA GESTIONAR LOS PROCESOS DE COMPRAS EN TIENDAS VIRTUALES**

**AUTORES**

Eduar Stiven Zamora Luna: Estudiante de 5 semestre de Tecnología en Gestión de Base de Datos

Alejandro Fierro Molina: Estudiante de 5 semestre de Tecnología en Gestión de Base de Datos

**ABSTRACT**

Hereby we try to explain the importance of an API since it is the most useful mechanism to connect two softwares to each other for the exchange of messages or data in standard format such as XML or JSON. This is how it becomes an instrument to seek income, open to talent, innovate and automate processes.

**RESUMEN**

Las API (Application Programming Interface en inglés) permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API (Interfaz de Programación de Aplicaciones en español) le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales).

Debido a que simplifican la forma en que los desarrolladores integran los elementos de las aplicaciones nuevas en una arquitectura actual, las API permiten la colaboración entre el equipo comercial y el de TIC. Las necesidades comerciales suelen cambiar rápidamente en respuesta a los mercados digitales en constante cambio, donde la competencia puede modificar un sector entero con una aplicación nueva. Para seguir siendo competitivos, es importante admitir la implementación y el desarrollo rápidos de servicios innovadores. El desarrollo de aplicaciones nativas de la nube es una forma identificable de aumentar la velocidad de desarrollo y se basa en la conexión de una arquitectura de aplicaciones de microservicios a través de las API.

Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas representan un valor comercial único porque simplifican y amplían la forma en que se conecta con sus partners y, además, pueden rentabilizar sus datos (API de Google Maps, Youtube API y Calendar API son ejemplos de API más usadas actualmente).

**PALABRAS CLAVE**

API, ASP.NET Core 2, Lenguaje XAML, Lenguaje C#, Sql Server

**INTRODUCCIÓN**

Uno de los principales objetivos de una API consiste en proporcionar un conjunto de funciones y procedimientos que puedan ser usados por otro software, dándoles un uso general.

De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, y evitando tener que programar todo desde el principio.

Es por eso que se decidió desarrollar una API para el área de negocios específicamente en las tiendas virtuales pues con la ayuda de esta se ahorra tiempo, costos y gastos en el proceso, inclusive se trabajará con tecnología innovable, usando archivos JSON sin referencias circulares.

**OBJETIVOS**

**Objetivo General**

Desarrollo e implementación de una API web para gestionar los procesos de compras en tiendas virtuales.

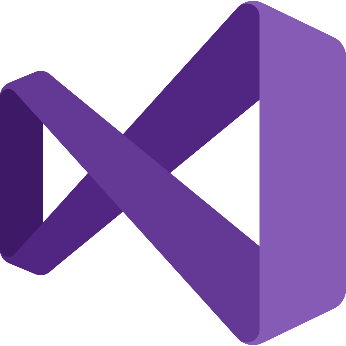
**Objetivos Específicos**

* Crear el prototipo de la aplicación, con el fin de identificar falencias para su corrección.
* Desarrollar una interfaz que sea agradable y de fácil utilización, que permita generar una aceptación por parte del usuario.
* Desarrollar una API que actúe como intermediario virtual que remite información de una interfaz a otra mediante archivos JSONs.
* La API debe de tener una comunicación con servicios y que dicha acción sea segura.

**RECURSOS**

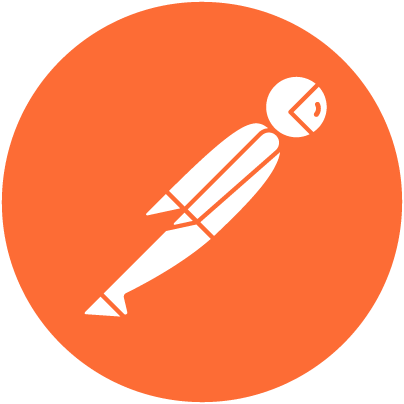
Para el desarrollo eficiente de la aplicación, hicimos uso de diferentes herramientas, intentando optimizar y sacarle el mejor provecho a cada una de ellas. Entre ellas tenemos:

**Visual Studio IDE**



Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Monaco.

**Postman**



Se trata de una herramienta dirigida a desarrolladores web que permite realizar peticiones HTTP a cualquier API. Postman es muy útil a la hora de programar y hacer pruebas, puesto que nos ofrece la posibilidad de comprobar el correcto funcionamiento de nuestros desarrollos.

Fuera de su objetivo principal, hacer peticiones a servicios, nos ofrece un conjunto de funcionalidades que nos ayudarán a organizar las peticiones en colecciones, hacer y automatizar pruebas, mantener equipos sincronizados y crear Mocks de APIs.

**JSON**



(acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript») es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera (año 2019) un formato independiente del lenguaje.

Si bien se tiende a considerar JSON como una alternativa a XML, lo cierto es que no es infrecuente el uso de JSON y XML en la misma aplicación; así, una aplicación de cliente que integra datos de Google Maps con datos meteorológicos en SOAP necesita hacer uso de ambos formatos.

**AutoMapper**



Es un asignador entre dos objetos. Es decir, AutoMapper es un mapeador de objetos de objeto. Asigna las propiedades de dos objetos diferentes transformando el objeto de entrada de un tipo al objeto de salida de otro tipo.

También proporciona algunos datos interesantes para eliminar el trabajo sucio de averiguar cómo asignar el objeto de tipo A con un objeto de tipo B, siempre y cuando el objeto de tipo B siga la convención establecida de AutoMapper. A medida que avancemos en este curso, verá el poder de la automapper.

**METODOS**

Para empezar a desarrollar la API primero creamos una base de datos con el motor de búsqueda SQL Server, en el momento de la creación de las tablas se puede agregar unos valores para verificar más adelante si funcionan los métodos de la API.

Acto siguiente, en el Visual Studio se crea una aplicación ASP.NET core versión 2.0, luego indicamos que vamos a crear una API, hecho esto se procede a instalar los paquetes necesarios (Microsoft.EntityFrameworkCore.Tools, Microsoft.EntityFrameworkCore.SQLSever, Microsoft.EntityFrameworkCore.SQLSever.Design, Microsoft.EntityFrameworkCore. Design, AutoMapper).

Teniendo listo nuestro proyecto con todos los recursos necesarios, seguimos con los modelos para eso usaremos un generador de código (scaffolding) en el que creará en una carpeta models los modelos basándose en las tablas que contenga la base de datos antes creadas, inclusive nos genera el contexto de datos.

De manera obligatoria se debe configurar la cadena de conexión pues ahí es donde sucede la relación con la base de datos (ante un cambio de un campo).

Se sabe que al trabajar con JSON que se comuniquen de manera directa con los modelos se generan archivos con un gran peso por tener propiedades demás, esto ocasiona una lentitud en el proceso de interacción de servicios.

Por esa razón se decidió trabajar con DTOs (Data Transfer Objects) estos son a veces copias de los modelos (por cada modelo debe haber un DTO) pero son un poco menos pesados (pues se cambia los elementos Icollection por List) y más eficaces.

Ahora llega el turno del AutoMapper, creando un archivo AutoMapperConfiguration dentro de la carpeta DTOs, para configurar y realizar el mapeo entre los modelos y los DTOs.

Estando dentro del archivo startup.cs se agrega la configuración antes hecha y se agrega unas instrucciones para ignora las propiedades de navegación a fin de que los JSON generados por los controladores no tengan referencias circulares.

Ya acercándonos al final siguen los controladores quienes tendrán los métodos impuestos por la API (nos referimos a operaciones CRUD, los métodos serian Put, Post, Get, Delete), por cada modelo se debe tener un controlador, entonces, para crearlo usaremos la creación automática por Visual Studio, damos al botón agregar, seleccionamos Controlador y luego indicamos que tipo de controlador vamos a crear, este es Controlador con acciones que usan EntityFramework.

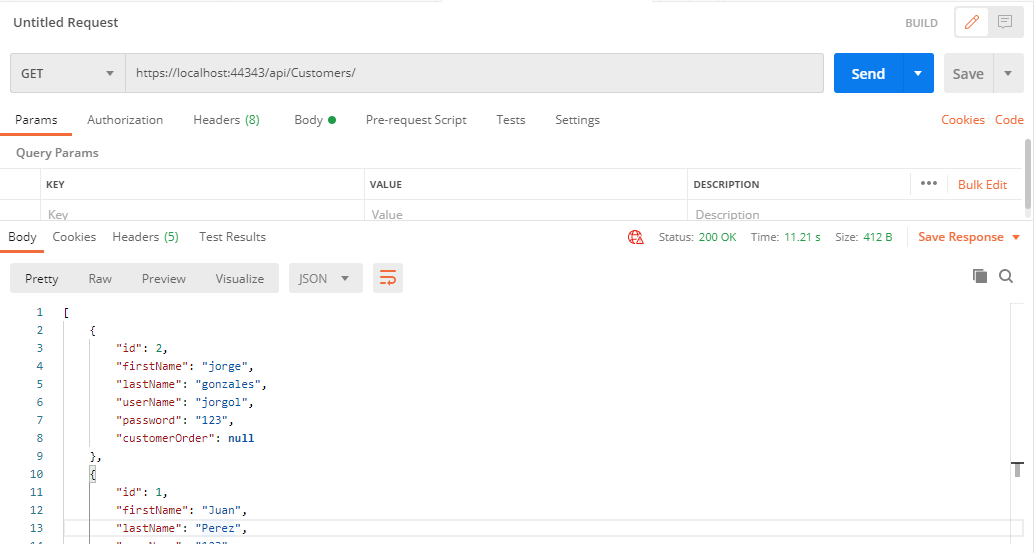
Como se plantea que la API contenga un servicio de login, entonces en CustomersController y en EmployeesController agregamos un método post que se encargará de esta acción usando los campos de ambos modelos Username y Password.

Otro detalle a añadir es que como los controladores se generaron de manera automática, entonces quedaron trabajando directamente con las entidades, por ende, procedemos a cambiarlas por los DTOs.

Por último, verificamos que la API es bien construida y para verificar su funcionamiento se usará la herramienta Postman para hacer peticiones.

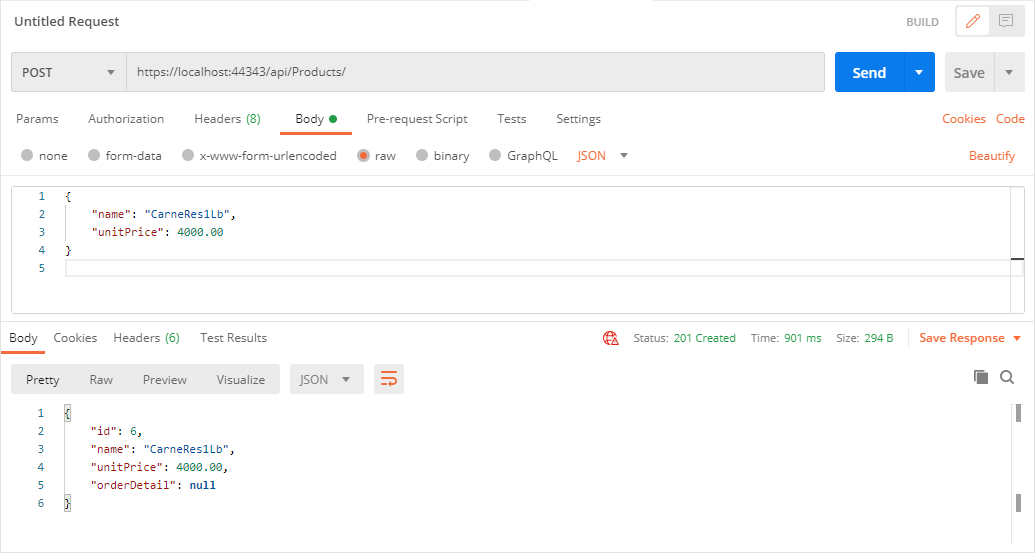
**RESULTADOS**

Lo más lógico es que exista un método en el que me arroje una lista de todos los clientes que estén registrados en la tienda. Pues este es el método Get:



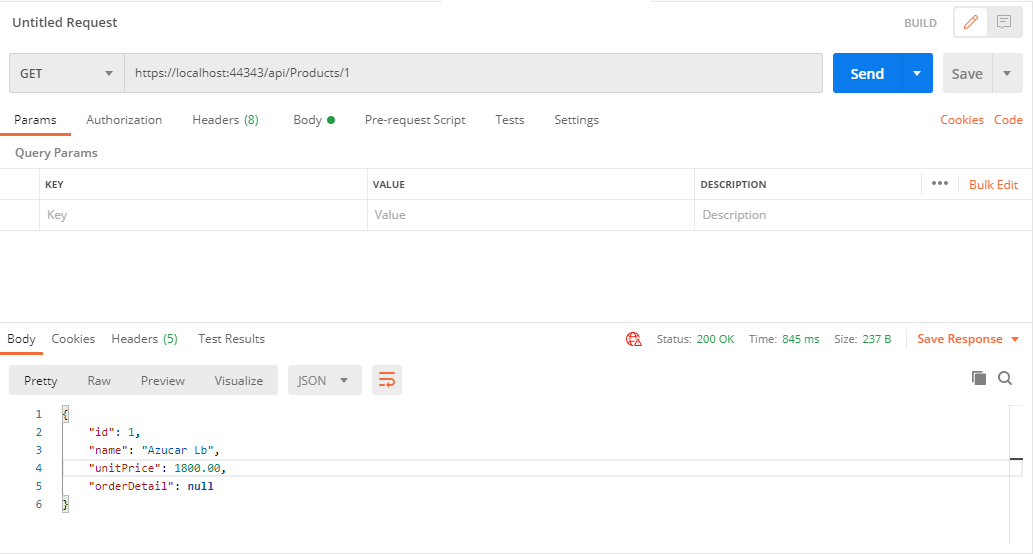
Otra acción relevante es que la API me permita agregar o trabajar de diferentes maneras con los productos, vemos los siguientes ejemplos:

Podemos observar que el servicio trabajó de manera correcta pues aparte de verse como resultado la lista de clientes en la esquina derecha-superior se puede ver el estado arrojado, en este caso es el 200 que significa OK.



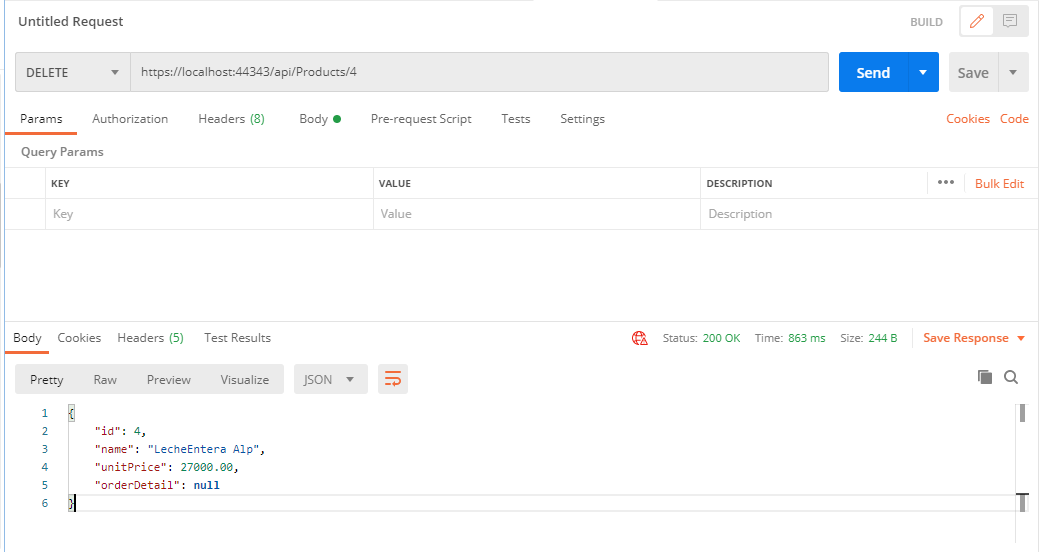
Podemos observar que el método funcionó de manera correcta pues aparte de verse en el resultado el elemento creado (n producto) en la esquina derecha-superior se puede ver el estado arrojado, en este caso es el 201 que significa Created.

En algún momento llegara la situación en el que el cliente o el empleado requiera de consultar la existencia de un producto en específico para conocer sus características, Entonces, aquí actúa el Get específico, recibiendo como parámetro el id del producto a buscar



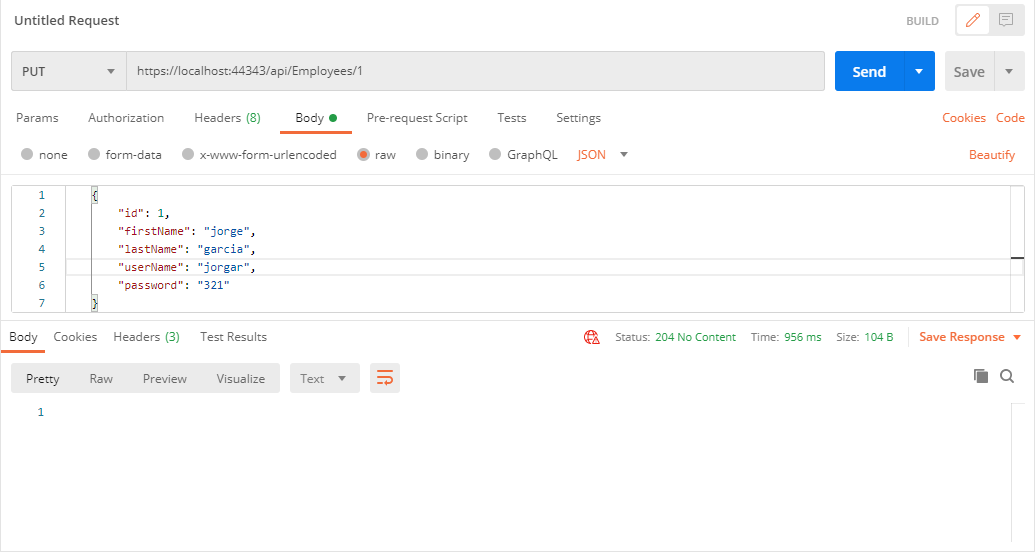
Podemos observar que el servicio trabajó de manera correcta pues aparte de ver el producto solicitado en la esquina derecha-superior se puede ver el estado arrojado, en este caso es el 200 que significa OK.

Otra operación que puede surgir es en el momento en el que el empleado decide borrar un prodcuto pues dado sea la situacion ya no venderan o surtiran ese artículo, en el API estamos hablando del método Delete.



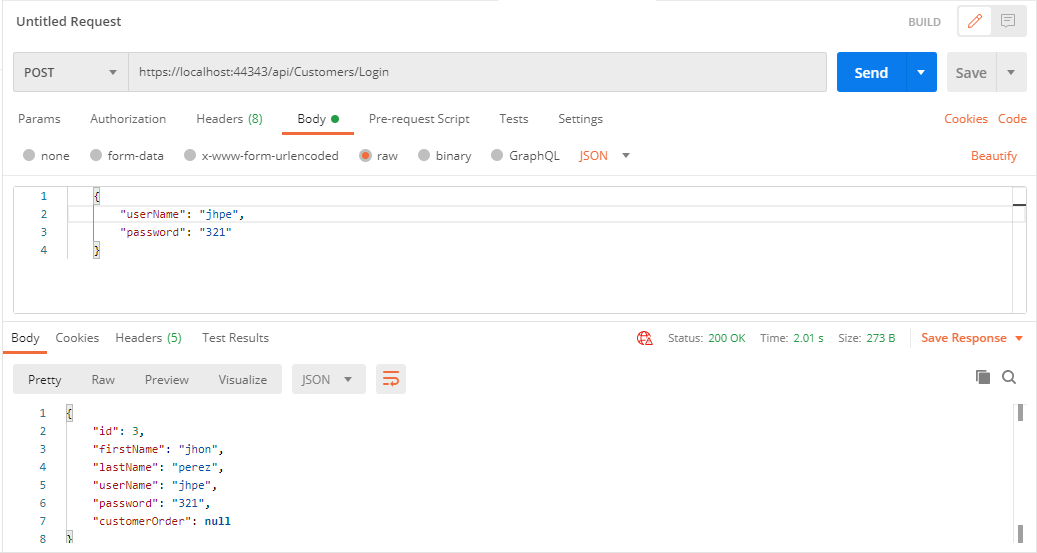
Podemos observar que el método arrojó un estado exitoso pues aparte de verse como resultado la lista de clientes en la esquina derecha-superior se puede ver el estado arrojado, en este caso es el 200 que significa OK.

Otra posible situación y más común es en el que la persona, ya sea el cliente o el empleado deciden cambiar algunos de sus datos por ejemplo su contraseña.

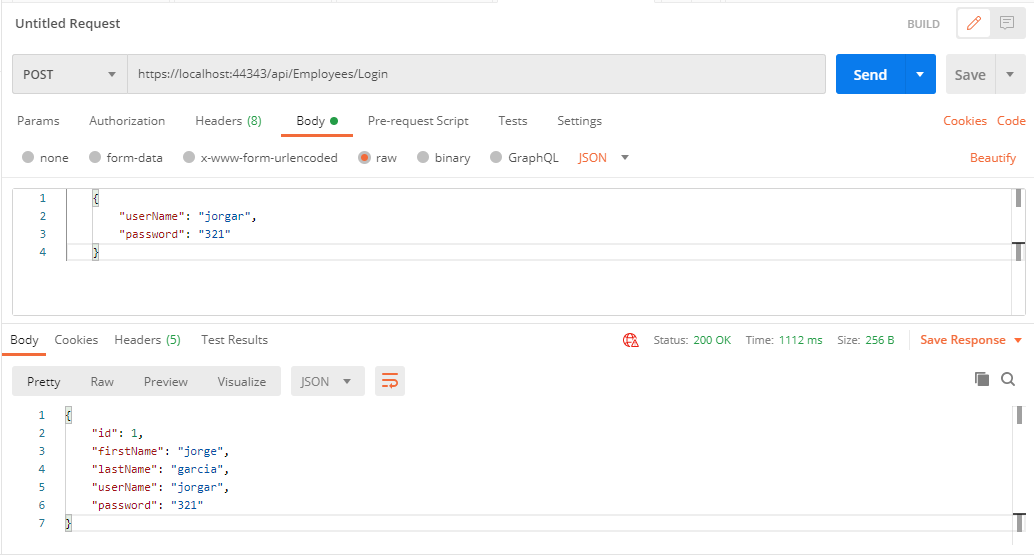


Podemos observar que el método arrojó un estado exitoso, eso lo podemos verificar observando en la esquina derecha-superior se puede ver el estado arrojado, en este caso es el 204 que significa No Content.

Por último, observaremos cómo funciona el servicio de Login.



Aquí esta el ejemplo cuando un usuario ingresa, en esa ocasión el API arroja un archivo JSON con todos los datos del usuario para verificar su información y continuar con los demás servicios.



Lo mismo sucede cuando un empleado va a consumir el servicio de Login.

**CONCLUSIONES**

1. Caracteristica principal detectada en la API fue que al tratarla de hacer lo más segura y eficientemente aumentará el tiempo en el desarrollo y en el precio a la hora de desplegarla.
2. Gracias a esta investigación nos dejo claro un panorama en la creación e implementación de las APIs, con todo ese conocimiento adquirido lo Podemos aplicar a proyectos futuros que tenga una cierta relación con este tema
3. Hemos notado que esta investigación puede seguirse mejorando aun más pues ya consumiéndose en una Aplicación Web con ASP.NET, agregándole cualidades diferentes al MD5 y añadiendo la biblioteca SignalR.
4. Las API Igual que una interfaz de usuario permite la interacción y comunicación entre un *software* y una persona, una API facilita la relación entre dos aplicaciones para el intercambio de mensajes o datosm por ejemplo pueden existir dos aplicaciones web de tienda virtual desde dos perspectivas. Una del cliente y otra del empleado.

Colocar como mínimo tres conclusiones

**AGRADECIMIENTOS**

* Se agradece principalmente al ingeniero José Omar Mayorga Pabón que nos brindó los espacios de aprendizaje para el desarrollo y construcción de la aplicación, inclusive tuvo paciencia a todos ante el proceso de aprendizaje.
* Se agradece a la universidad del Tolima por el esfuerzo de permitir ala generación de espacios de investigación y construir espacios de interacción con los compañeros de cursos, los cuales permitieron el intercambio de información, sobre todo ante la situacion en la que nos encontramos actualmente por covid-19.
* Se agradece a Dios por permitirnos estudiar en una modalidad universitaria que ayuda a que el estudiante sea autodidacto en la consecución de conocimientos para el mejoramiento de nuestra carrera y poder afianzarnos en nuestro campo.

**REFERENCIAS**

* <https://www.wrike.com/es/blog/que-es-una-api-necesitas-saber/>
* <https://bbvaopen4u.com/es/actualidad/que-es-una-api-y-que-puede-hacer-por-mi-negocio>
* <https://www.reviso.com/es/blog/aun-no-sabes-para-que-sirve-una-api/>
* <https://lamadriguerabit.com/articulos/que-es-postman/>
* [AutoMapper in C# with Examples - Dot Net Tutorials](https://dotnettutorials.net/lesson/automapper-in-c-sharp/)
* [JSON - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/JSON)
* [Microsoft Visual Studio - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Microsoft_Visual_Studio)