|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | | | | |
| Documentación  Ejercicio técnico para cargo ingeniero de desarrollo | | | | |
|  | | | | |
|  | | | | |
| Estado: **Creacion** | Creación: **Andres Felipe Arevalo Moreno** |  | Identificador: | **123** |
|  |  |  | Versión: | **1,0** |
|  |  |  | Fecha: | **Febrero 28 de 2023** |
|  | | | | |
|  | | | | |
|  | | | | |
| LAS COPIAS IMPRESAS NO ESTÁN CONTROLADAS | | | | |
| Este documento y su contenido son de uso interno de Inalambria Internacional S.A.S  Cualquier copia u otro uso deben ser autorizados expresamente por Inalambria Internacional S.A.S  © Copyright INALAMBRIA Internacional S.A.S Todos los derechos reservados. | | | | |

**Tabla de contenido**

[1. Propósito 3](#_Toc115184903)

[2. Error Reportado 3](#_Toc115184904)

[3. Descripción de la Solución 3](#_Toc115184905)

[Contexto 3](#_Toc115184906)

[Causa raíz 4](#_Toc115184907)

[Tipo de problema 4](#_Toc115184908)

[Diagnóstico 4](#_Toc115184909)

[Impacto 4](#_Toc115184910)

[Solución Presentada 4](#_Toc115184911)

[Solución general 4](#_Toc115184912)

[Solución Particular 4](#_Toc115184913)

[4. Evidencia de la Solución 5](#_Toc115184914)

[5. Observaciones 17](#_Toc115184915)

[6. Revisiones y aprobaciones 17](#_Toc115184916)

# Proposito

El propósito de este documento es documentar las evidencias y explicar el desarrollo de la prueba técnica.

# Error Reportado

No Aplica

# Descripción de la Solución

## Contexto

* ***Prueba Técnica – Ejercicio Practico***

Este ejercicio consiste en construir una cadena con fichas de dominó. Implemente una forma de ordenar un conjunto dado de fichas de dominó de tal manera que se construya una cadena correcta de fichas (los puntos en una de las mitades de una ficha concuerdan con los puntos que tiene la mitad vecina de la ficha adyacente) y que los puntos de las mitades de aquellas fichas que no tengan vecino (fichas primera y última) concuerden uno con el otro.

*Ejemplo 1:*

Dadas las fichas [2|1], [2|3] y [1|3], el resultado del computo fruto de su construcción debe retornar algunas de las siguientes opciones:

✓ [1|2] [2|3] [3|1] ó

✓ [3|2] [2|1] [1|3] ó

✓ [1|3] [3|2] [2|1], etc.

Como es posible observar, los números primero y último son los mismos

.

*Ejemplo 2:*

Dadas las fichas [1|2], [4|1] y [2|3] la cadena resultado no es válida:

* [4|1] [1|2] [2|3]

Los números primero y último no son los mismos.

Deberá construir un API que permita enviar en notación JSON el conjunto de fichas de dominó, conjunto que puede tener mínimo dos fichas y máximo seis fichas; así mismo el resultado debe ser retornado en notación JSON indicando una cadena correcta que cumpla lo informado al inicio de la descripción de Ejercicio Práctico –

este ejercicio, o en determinado caso, informando la imposibilidad de generar una cadena que cumpla los criterios definidos con el conjunto de fichas de dominó enviado.

En esta API debe implementar un método de autenticación, el que sea de su agrado; así mismo debe implementar códigos de respuesta HTTP.

*Puntos para tener en cuenta*

✓ El ejercicio debe realizarse en lenguaje C#; puede utilizar las librerías de su elección.

✓ Dejamos la actividad abierta para que exprese su creatividad; siéntase libre para incluir en la prueba lo que considere necesario para cumplir las expectativas relacionadas con la demostración de sus conocimientos.

✓ Documentación relacionada y componentes adicionales a los solicitados serán un plus que será tenido en cuenta.

✓ Como entregable de esta prueba deberá publicar dicho API y el código fuente (GitHub es una opción) y enviar cualquier documentación que considere necesaria de tal manera que sea posible su visualización y prueba por nuestra parte, con lo cual agendaremos un espacio para que se sustente lo implementado.

## Causa raíz

No aplica

## Tipo de problema

API

## Diagnóstico

Para la solución se debe recibir en un json las fichas que sean mínimo 2 fichas y máximo 6 fichas en forma de string e internamente se deben separar los números de forma individual en una lista y volverlos enteros. Después agregarlos a un arreglo dimencional 2D (es decir, casi una matriz) y el calculo de la diagonal invertida debe ser igual al número de las posiciones, para poder acomodar las fichas, debe desplazarse por filas y columnas, validando las posiciones internamente debe tener dos ciclos internos, uno que se vaya desplazando por filas y columnas para hacer el intercambio de filas y columnas verificando si cumple con la condición. Cuando cumpla, se saldrá del ciclo while y avanzara en el ciclo interno de las columnas (es decir la variable j) y cuando termine la ejecución se saldrá del los dos ciclos for y se procederá a almacenar el resultado en una variable acumulativa de tipo string para devolverla en el resultado json. También se agrega la autorización de token y la autenticacion de validación mediante roles con usuario y contraseña, los cuales son ”administrador” y “usuario”. Solo el administrador podrá insertar fichas y así protegiendo a la API basada con algunas librerias, en la arquitectura de MVC (Modelo Vista Controlador).

## Impacto

No Aplica

## Solución Presentada

Se hace una API con .net core 6 con #c visual studio 2022 y se realiza la API con peticiones desde Postman, realiza el calculo mediante arreglos dimencionales, autenticación y autorización de la API en el mismo aplicativo (colocar el token dentro de la API en vez de ir al postman, se puede hacer de las dos maneras). Artefacto publish y código fuente publicado en github “pipear07” y ambos comprimidos y enviados por correo para tener el back-up

## Solución general

Se hace una API con .net core 6 con visual studio 2022 y se realiza la API con peticiones desde Postman, realiza el cálculo mediante arreglos dimencionales, autenticación y autorización de la API en el mismo aplicativo (colocar el token dentro de la API en vez de ir al postman, se puede hacer de las dos maneras). Artefacto publish y código fuente publicado en github “pipear07” y ambos comprimidos y enviados por correo para tener el back-up

## Solución particular

No Aplica

# Evidencias de la Solucion

**Librerias Usadas (paquetes Nuget):**

* Microsoft.AspNetCore.Authentication.JwtBearer
* Microsoft.EntityFrameworkCore.Tools
* Newtonsoft.Json
* Swashbuckle.AspNetCore
* Swashbuckle.AspNetCore.Swagger

**Logica de la solucion del código**

Para la lógica del programa, se ingresa una cadena de fichas como se muestra a continuación "[2|1]", "[2|3]", "[1|3]"

*Código Json*

{

  "ficha": [

    "[2|1]", "[2|3]", "[1|3]"

  ]

}

Internamente el programa le quita los corchetes y separador a las fichas para poder trabajar en ellos. Se agregan a una lista y se vuelven enteros

*Lista* **=**

Después se agregan a un arreglo dimencional de 2d (casi como ver una matriz) y se colocan en orden de ingreso.

*Separador* ***= 2x3***

*2* ***= numero de caras por ficha - filas***

*3* ***= cantidad de fichas insertadas - columnas***

Recordemos que en algebra lineal uno resolvía matrices por método de Gauus Jordan, en que consiste en resolver las diagonales principales y volverlas 1 y 0.

*1* ***= pivote de la matriz, ósea el numero que voy a interactuar***

Para el ejercicio, es lo mismo, pero en diagonal invertida, y validando los números iguales en diagonal e ir avanzando, intercambiando filas y columnas hasta cumplir con la penúltima columna.

*1* ***= pivote de la matriz, ósea el número que voy a interactuar***

En el ejercicio, aquí se muestra la matriz y sus posiciones, para hacer los cálculos en el ciclo for y while para poder avanzar e intercambiar filas y columnas. Y después mostrar el resultado final.

*Inicio*

*Separador* ***= Necesitamos poner diagonal el 1***

*Separador* ***= Intercambie filas y sino son iguales***

*Separador* ***= cambie filas e intercambia columnas, verifique si son iguales***

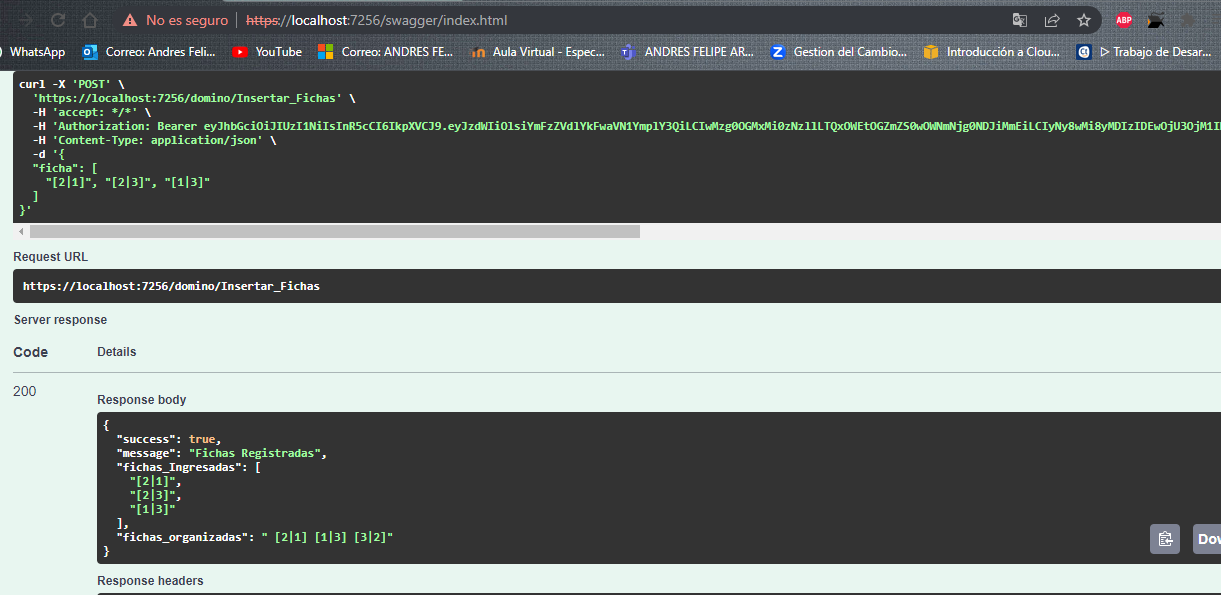
*Separador* ***= si son iguales cambie de pivote y si son iguales termine ciclo***

*Final*

Después guardamos con una variable llamada “Fichas\_organizadas” la cual almacenará en un ciclo for por orden, el resultado de las fichas sin alterar su estructura y el resultado nos dará algo como esto:

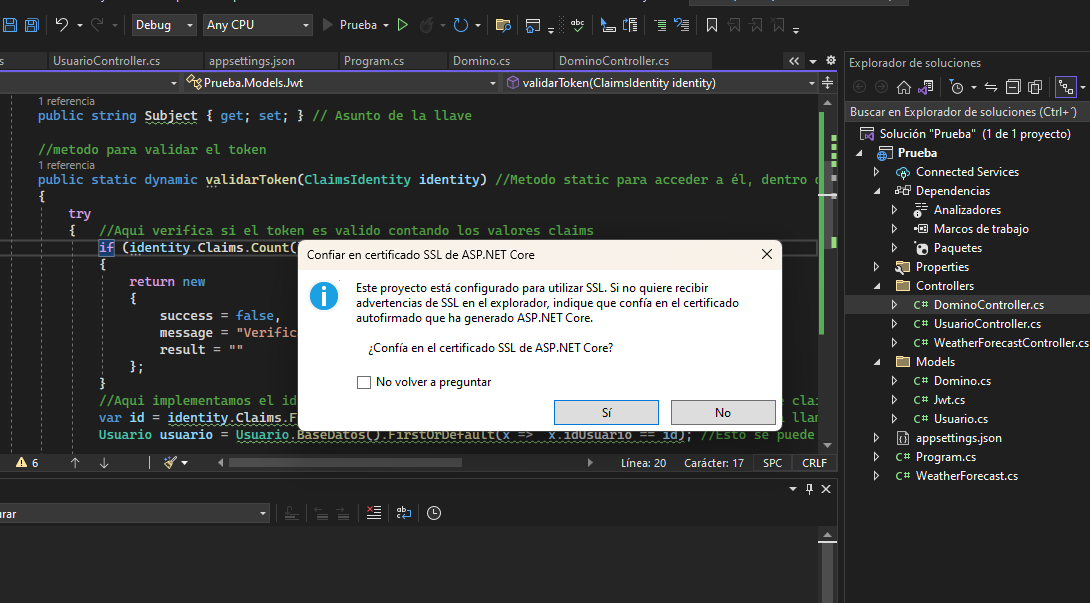


Entorno de postman



Entorno de la API corriendo en .net core 6

**Ejecutar la API**



Cuando la ejecutamos, nos preguntara que si deseamos confiar en el certificado SSL, le decimos que “No” en visual studio 2022 con sus paquetes nugets y librerias.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Dominio** = La petición post para recibir las fichas.

**Usuario** = Es para iniciar sesión y que genere el token y se pueda generar en el candado de “Authorize”.

**WeatherForecast** = Es el ejemplo que viene por default en la API

Primero debemos iniciar sesión con el usuario con rol de “administrador”, porque es el único que puede insertar fichas de dominó.

**Lista de usuarios:**

* **Usuario =** pipear

**Rol =** Usuario

**Clave =** 12345

* **Usuario =** admin

**Clave =** 12345

**Rol =** Administrador

En la clase Usuario dentro de Models (modelos) esta la lista de usuarios\*

Codigo Json

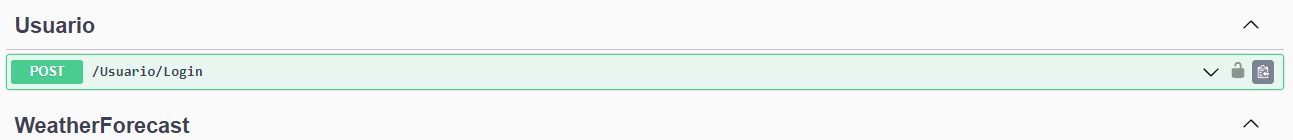
{

"usuario": "admin",

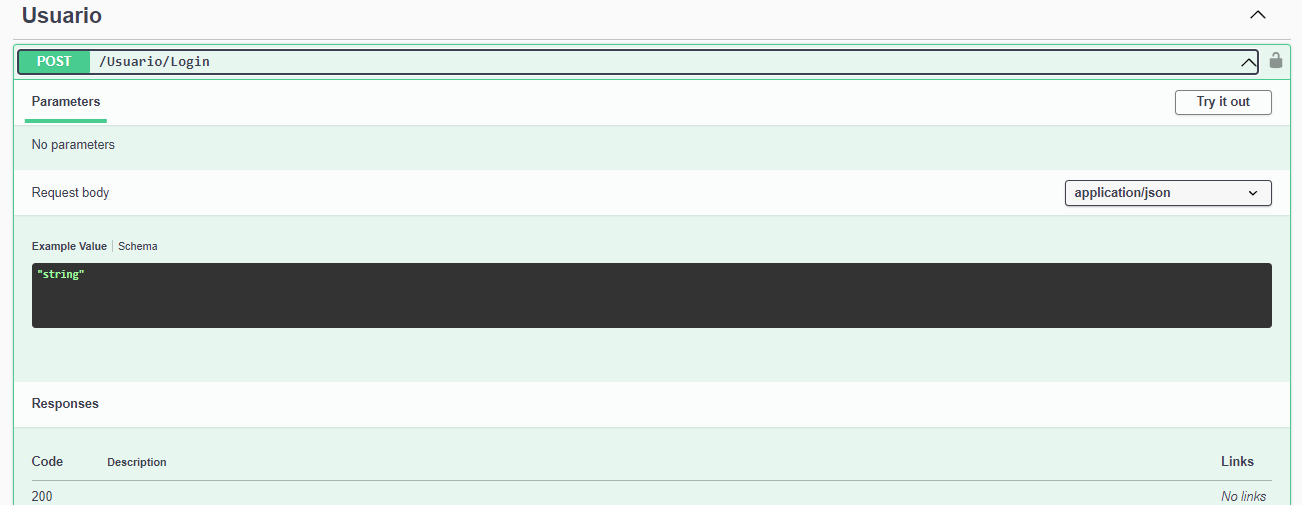
"clave": "12345"

}

Después le damos click en /Usuario/Login

****

Luego se abrirá el panel en la pare superior derecha le damos click al boton “try it out”



Después, se nos abrirá la pestaña en blanco mostrando la palabra “string”, la borramos

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Copiamos el código json dejado arriba y nos debe quedar de esta forma.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Luego le damos en el botón azul de “execute” y nos debe aparecer un token encriptado con esa sesión, lo copiamos y procedemos. Ya que tenemos 4 minutos de vigencia

Captura de pantalla de computadora

Descripción generada automáticamente

Luego copiamos la variable “result” que contiene el token, sin comillas, debe ser algo parecido a esto:

**eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOlsiYmFzZVdlYkFwaVN1YmplY3QiLCJlZTQyMmE2OC04Y2Y4LTQ5YTEtOWQyYS0wNDQ2ZTY1NjRiZjQiLCIyNy8wMi8yMDIzIDExOjU3OjQ0IHAuwqBtLiJdLCJpZCI6IjIiLCJ1c3VhcmlvIjoiYWRtaW4iLCJleHAiOjE2Nzc1NDI1MDQsImlzcyI6Imh0dHBzOi8vbG9jYWxob3N0OjcyNTYvIiwiYXVkIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6NzI1Ni8ifQ.X9GZ9chptUnVxIAGTyY4H\_LbQQzbC1F1sohL0i9DJ2U**

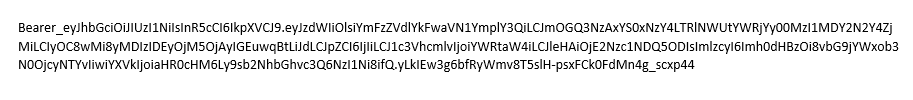
(algoritmo de encriptación sha256)

Después en la parte superior derecha, nos metemos al candado verde que dice “Authorize” le damos click a ese boton

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente



 Después, escribimos “Bearer” y después un espacio y pegamos el token que copiamos y le damos en el botón de “Authorize” y después los candados cambiaran de abierto a cerrado, es significa que se ha iniciado sesión con éxito, debe quedar asi:



Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Luego le damos en “Close”

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

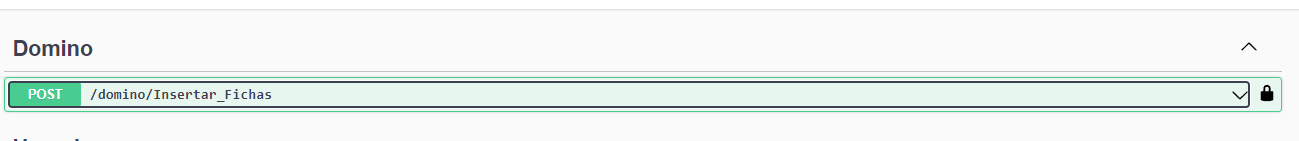


Por lo tanto, los candados cambiaron a cerrado

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Después, nos vamos a la parte de “Domino” y hacemos click en /domino/Insertar\_Fichas



Nos va aparecer algo como esto, le damos click en el botón “try it out” en la parte superior derecha

Patrón de fondo

Descripción generada automáticamente

Luego nos debe aparecer algo como esto, quitamos el “string” y colocamos el valor json de las fichas

Patrón de fondo

Descripción generada automáticamente con confianza media

La entrada de fichas correcto de la cadena es el siguiente:

*Código Json*

{

  "ficha": [

    "[2|1]", "[2|1]"

  ]

}

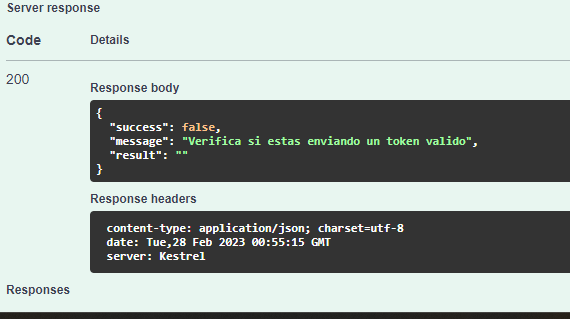
La entrada de fichas correcto de la cadena es el siguiente:

Texto

Descripción generada automáticamente

Las fichas aparecen organizadas en la variable json “fichas\_organizadas”

Nota: Recordemos que tenemos 4 minutos de vigencia del token, si el tiempo se llega a pasar, tendrá que volver a iniciar sesión, por si el token llega a ser invalido



**Fichas de prueba:**

**2x2**

"[2|1]","[2|1]"

**2x3**

"[2|1]", "[2|3]", "[1|3]"

**2x4**

"[2|1]", "[4|3]", "[3|2]", "[1|4]"

**2x5**

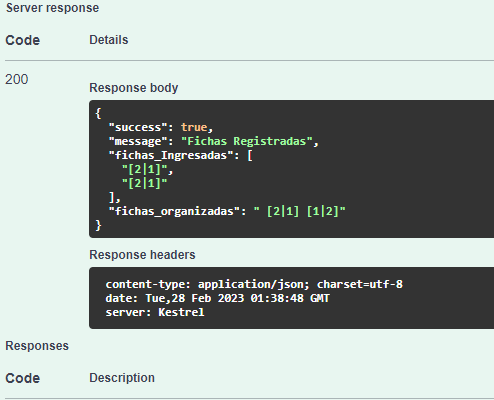
"[2|1]", "[5|2]", "[4|5]", "[4|3]" ,"[1|3]"

**2x6**

"[1|3]", "[1|2]", "[6|4]", "[3|4]" ,"[2|5]","[6|5]"

**Fichas de prueba con evidencia:**

**2x2**



**2x3**

Texto

Descripción generada automáticamente

**2x4**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**2x5**

Texto

Descripción generada automáticamente

**2x6**

Interfaz de usuario gráfica, Texto

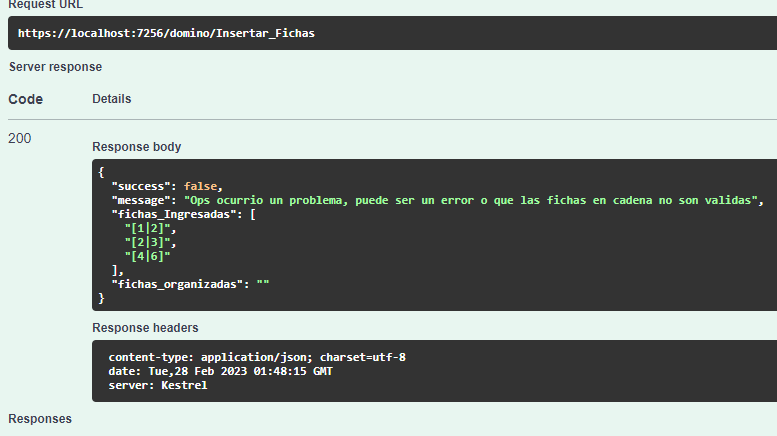
Descripción generada automáticamente

**Validaciones:**

Texto

Descripción generada automáticamente

Si se digita una sola ficha o se digita mas de 6 fichas, saldrá un mensaje con success en false, que dice que mínimo debe ser 2 y máximo 6 fichas respetando los criterios de los requerimientos



Codigo Json

{

"ficha": [

"[1|2]", "[2|3]", "[4|6]"

]

}

Aquí valido si la cadena de entrada es válida, ósea si se puede organizar de acuerdo con la lógica planteada, en este caso no es válida, entonces emitirá este tipo de mensaje, y también estoy validando la excepción y las cadenas invalidas en una sola. Según requerimientos.

**Manejo desde postman:**

**Texto

Descripción generada automáticamente con confianza media**

Primero, tenemos que estar ejecutando la API para que desde el postman pueda ejecutarla, después entramos al postman y abrimos una pestaña nueva, en petición POST e insertamos esta URL:

<https://localhost:7256/Usuario/Login>

Seleccionamos la sección “Body” y la opción “raw” con la extension JSON

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

Copiamos y pegamos el json con las credenciales del usuario administrador

*Codigo Json*

{

"usuario": "admin",

"clave": "12345"

}

Y le damos en el botón “Send” (enviar)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Aquí vemos que arrojo un “Exito” para que genere el token de result y lo copiamos sin comillas, nos debe quedar así:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOlsiYmFzZVdlYkFwaVN1YmplY3QiLCI4N2FhZGYyZS0yYThmLTQ5NTItOTBhYS01ZDliNTI2M2FiYTEiLCIyOC8wMi8yMDIzIDI6MTU6MTEgYS7CoG0uIl0sImlkIjoiMiIsInVzdWFyaW8iOiJhZG1pbiIsImV4cCI6MTY3NzU1MDc1MSwiaXNzIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6NzI1Ni8iLCJhdWQiOiJodHRwczovL2xvY2FsaG9zdDo3MjU2LyJ9.W\_DjTPYPx72r7cDT0h1iVVPlmD8uzKaoq0UFrsDlE\_A

Ahora tenemos 4 minutos de vigencia del token, entonces creamos una nueva pestaña en postman que se parezca a esto:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Abrimos una pestaña nueva, en petición POST e insertamos esta URL:

<https://localhost:7256/domino/Insertar_Fichas>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Después, en la pestaña “Authorization” buscamos Type en la izquierda y seleccionamos la opcion “Bearer Token”

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Luego pegamos el token que tenemos copiado así como esta subrayado en azul. Lo pegamos y nos vamos a la pestaña de “Body”

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Insertamos el siguiente codigo Json:

{

  "ficha": [

    "[2|1]", "[2|3]", "[1|3]"

  ]

}

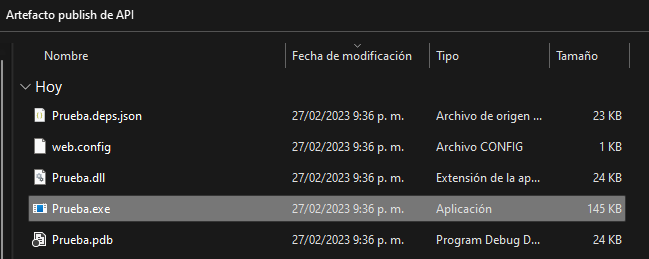
Y nos dice que el token no es válido, ahora debemos iniciar sesión otra vez y generar el token otra vez, lo copiamos, lo pegamos en bearer y en la pestaña “Body” pegamos este json y le damos en el botón “Send” y nos organizara los fichas de domino.

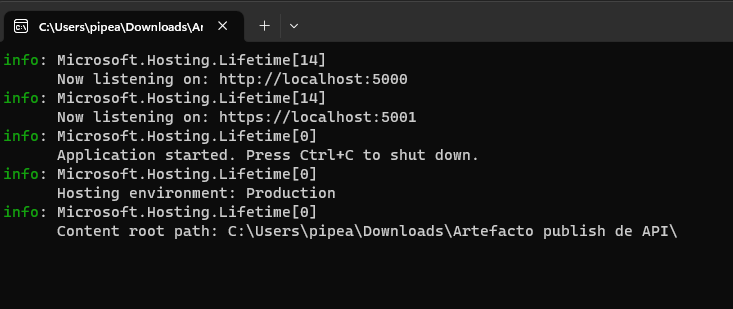
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Artefacto publish (ejecutable .exe)**

Se genero él .exe de la API llamado “Prueba.exe” esta dentro de la carpeta llamada “Artefacto publish de API”, una vez que este ejecutando nos mostrara una pantalla negra, eso significa que esta en ejecución.





Entonces, el puerto es el 5001 en este caso, entonces ejecutamos ese mismo puerto en el postman como se hizo en la explicación del “manejo en postman”, solo que la diferencia es que el puerto cambiara, y quedara algo como así:

<https://localhost:5001/Usuario/Login>

<https://localhost:5001/domino/Insertar_Fichas>

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Funcionará sin ningún problema

**Repositorio GitHub:**

**Link** [**https://github.com/pipear07/Prueba\_Tecnica\_API\_AndresFelipeArevaloMoreno.git**](https://github.com/pipear07/Prueba_Tecnica_API_AndresFelipeArevaloMoreno.git)

También por seguridad, se envió el código, el artefacto publish y la documentación comprimidos por correo y en el github esta todo también.

# 5. Observaciones

Para que corra el proyecto de la API, deben tener visual studio 2022 e instalar las librerias mencionadas anteriormente. También se puede correr por el artefacto publish, pero tener cuidado con los puertos y el tiempo del token y los roles de los usuarios (que vendría siendo parte del plus)

# Revisiones y aprobaciones

| Versión | Fecha | Autor | Estado | Revisado por | Aprobado por | Descripción modificación |
| --- | --- | --- | --- | --- | --- | --- |
| 1,0 | 28/02/2022 | Andrés Felipe Arevalo Moreno | Creación |  |  | Creación del Documento |
|  |  |  |  |  |  |  |