

Documentación prueba técnica

Objetivo

Crear API con autenticación, la cual reciba en formato Json un conjunto de fichas de dominó y organizarlas según las condiciones requeridas.

Tecnologías Utilizadas

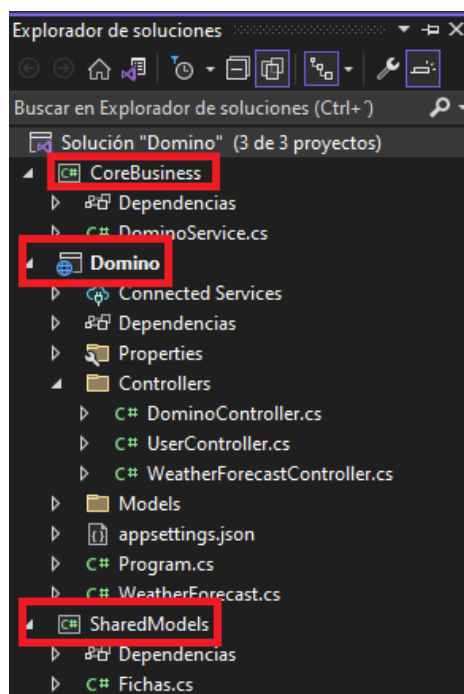
ASP.NET core Web Api versión 6.0, JWT (Json Web Tokens)

Estructura del Proyecto

El proyecto internamente consta de 3 proyectos. El proyecto principal es Web Api en ASP.NET core llamado “Domino”. Este contiene un controlador llamado “UserController” el cual se encarga de generar el token por medio de JWT. También tiene un controlador llamado “DominoController” el cual realiza la validación del token y organiza las fichas.

Los otros 2 proyectos son bibliotecas de clases, una llamada “CoreBusiness”, la cual contiene una clase llamada “DominoService”, en ella se encuentra el método que se encargará de ordenar las fichas.

La otra biblioteca de clases se llama “SharedModels”, esta contiene un Modelo llamado Fichas, el cual se comparte tanto en el proyecto principal, como en la biblioteca “CoreBusiness”, la intención es que en esta biblioteca se agreguen los modelos compartidos entre los proyectos de la solución y permitir un proyecto más ordenado.



El proyecto “Domino” contiene 2 modelos, el primero llamado “JWT” el cual se llena con la información de JWT que se encuentra en “appsetting.json”:

```

using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using Microsoft.IdentityModel.Tokens;

namespace Domino.Models
{
    1 referencia
    public class Jwt
    {
        0 referencias
        public string Key { get; set; }
        1 referencia
        public string Issuer { get; set; }
        1 referencia
        public string Audience { get; set; }
        1 referencia
        public string Subject { get; set; }
        1 referencia
        public int ExpiresInMinutes { get; set; }
    }
}

```

Y un modelo que servirá para obtener la información del usuario que está iniciando sesión llamado "UserLogin":

```

namespace Domino.Models
{
    7 referencias
    public class UserLogin
    {
        3 referencias
        public string IdUser { get; set; }
        4 referencias
        public string? UserName { get; set; }
        4 referencias
        public string? Password { get; set; }

        2 referencias
        public static UserLogin UserBD() {
            UserLogin user = new UserLogin();
            user.IdUser = "1";
            user.UserName = "Inalambria";
            user.Password = "password";
            return user;
        }
    }
}

```

Autenticación:

He realizado la autenticación con JWT (Json Web Tokens), esta permite generar un token el cual contendrá información para el inicio de sesión y atributos del token:

```

var token = new JwtSecurityToken
(
    jwt.Issuer,
    jwt.Audience,
    claims,
    expires: DateTime.Now.AddMinutes(jwt.ExpiresInMinutes),
    signingCredentials: signIn
);

```

Funcionalidades Principales:

Controlador "UserController":

- Recibe los datos de inicio de sesión en formato JSON y los deserializa para obtener el nombre de usuario y la contraseña.
- Verifica las credenciales del usuario con los datos del usuario demo creado.
- Si las credenciales son válidas, genera un token JWT que contiene la información del usuario autenticado, como su id, nombre de usuario y contraseña.
- El tiempo de expiración del token se configura utilizando el valor proporcionado en el archivo de configuración "appsettings.json".

El token generado se devuelve como respuesta al cliente, lo que le permite al usuario autenticado acceder a la funcionalidad de ordenar las fichas mientras el token permanezca activo.

Parámetros del Método Login

- **"optData"**: Objeto que contiene los datos de inicio de sesión del usuario en formato JSON.

Respuestas del Método Login

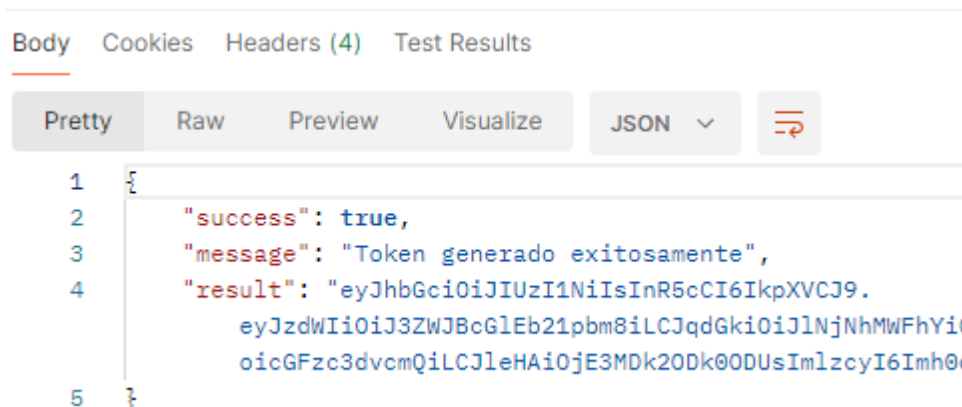
En caso de que las credenciales del usuario no sean válidas, se devuelve un objeto JSON con el siguiente formato:



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (4), and Test Results. The Body tab is selected, and the response is displayed in a 'Pretty' format. The JSON response indicates a failed login attempt.

```
1 {
2   "success": false,
3   "message": "Credenciales Incorrectas",
4   "result": ""
5 }
```

Si las credenciales del usuario son válidas y se genera el token correctamente, se devuelve un objeto JSON con el siguiente formato:



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (4), and Test Results. The Body tab is selected, and the response is displayed in a 'Pretty' format. The JSON response indicates a successful login and token generation.

```
1 {
2   "success": true,
3   "message": "Token generado exitosamente",
4   "result": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJ3ZWJBcG1Eb21pbm8iLCJqdGkiOiJlNjNhMWFhYi0oicGFzc3dvcmQiLCJleHAiOjE3MDk0ODU5Im1zcyI6Imh0c
5 }
```

Controlador "DominoController":

Se encarga de la validación de fichas de dominó.

Método "EnviarFichas"

Recibe un conjunto de fichas de dominó en formato JSON como cuerpo de la solicitud y verifica la autenticación del usuario utilizando un token JWT proporcionado en el encabezado de autorización. Si el usuario está autenticado y el token no ha expirado, el método procede a procesar las fichas de dominó enviadas por el usuario.

Parámetros del Método "EnviarFichas"

- **fichas:** Lista de fichas de dominó proporcionadas en formato JSON.

Respuestas del Método "EnviarFichas"

- Si el token de autenticación no se proporciona o es inválido, se devuelve una respuesta HTTP 401 (Unauthorized), indicando que el usuario no está autorizado para acceder al recurso protegido.
- Si el token de autenticación ha expirado, se devuelve una respuesta HTTP 401 (Unauthorized), con un mensaje indicando al usuario que su sesión ha expirado y que debe generar un nuevo token.
- Si las credenciales del usuario son válidas y las fichas de dominó proporcionadas son correctas, se procesan las fichas y se devuelve una respuesta HTTP 200 (OK), con un cuerpo de respuesta en formato JSON que contiene las fichas de dominó ordenadas correctamente, junto con un mensaje indicando que las fichas fueron recibidas correctamente.
- Si las fichas de dominó proporcionadas no cumplen con los criterios de validez (por ejemplo, si la cantidad de fichas es incorrecta), se devuelve una respuesta HTTP 400 (Bad Request).

Servicio "DominoService":

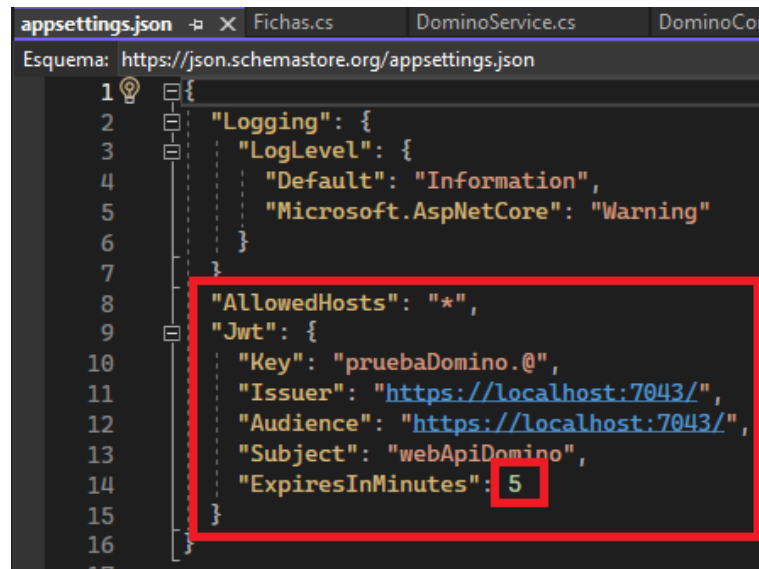
Método VerificarFichas

Este método recibe una lista de fichas de dominó y las organiza de manera que los números en los lados de las fichas coincidan correctamente.

- La primera ficha en la lista se toma como punto de partida.
- Se verifica si el lado A o el lado B de la ficha actual coincide con el lado B de la ficha anterior en la lista.
- Si coincide, se agrega la ficha actual a la lista de fichas ordenadas y se actualiza el lado actual.
- Si ninguno de los lados coincide, se pasa a la siguiente ficha en la lista.
- Si se agrega una ficha a la lista, se reinicia el ciclo para verificar si las fichas restantes encajan correctamente.
- Al final, se verifica si el último lado coincide con el lado inicial de la primera ficha. Si es así, se devuelve la lista de fichas ordenadas.

Configuración

En el archivo “appsettings.json” se encuentra la configuración básica de JWT. Para este proyecto, el único ítem configurable será "ExpiresInMinutes", el cual indica el tiempo de duración en minutos que durará el token activo, si se desea aumentar la cantidad de tiempo en la duración del token no se hace necesario ingresar a la solución del proyecto, se puede realizar modificando directamente el archivo ubicado en la carpeta del proyecto.



```
1 {
2   "Logging": {
3     "LogLevel": {
4       "Default": "Information",
5       "Microsoft.AspNetCore": "Warning"
6     }
7   },
8   "AllowedHosts": "*",
9   "Jwt": {
10    "Key": "pruebaDomino.",
11    "Issuer": "https://localhost:7043/",
12    "Audience": "https://localhost:7043/",
13    "Subject": "webApiDomino",
14    "ExpiresInMinutes": 5
15  }
16 }
```

Nombre	Fecha de modificación	Tipo	Tamaño
bin	29/02/2024 6:42 p. m.	Carpeta de archivos	
Controllers	5/03/2024 7:22 p. m.	Carpeta de archivos	
Models	5/03/2024 6:52 p. m.	Carpeta de archivos	
obj	4/03/2024 8:00 p. m.	Carpeta de archivos	
Properties	29/02/2024 6:42 p. m.	Carpeta de archivos	
appsettings.Development.json	29/02/2024 6:42 p. m.	Archivo JSON	1 KB
appsettings.json	5/03/2024 7:16 p. m.	Archivo JSON	1 KB
Domino.csproj	4/03/2024 8:02 p. m.	C# Project File	1 KB
Domino.csproj.user	3/03/2024 8:19 p. m.	Per-User Project O...	1 KB
Program.cs	5/03/2024 5:51 p. m.	C# Source File	2 KB
WeatherForecast.cs	29/02/2024 6:42 p. m.	C# Source File	1 KB

Existe también un modelo en el proyecto principal “Domino”, en el cual creé un usuario predeterminado con el cual hacer las pruebas:

```

namespace Domino.Models
{
    7 referencias
    public class UserLogin
    {
        3 referencias
        public string IdUser { get; set; }
        4 referencias
        public string? UserName { get; set; }
        4 referencias
        public string? Password { get; set; }

        2 referencias
        public static UserLogin UserBD() {
            UserLogin user = new UserLogin();
            user.IdUser = "1";
            user.UserName = "Inalambria";
            user.Password = "password";
            return user;
        }
    }
}

```

USO DE LA API:

Credenciales de acceso: Estas serán las credenciales que se deben usar para realizar la prueba

Usuario = "Inalambria";

Contraseña = "password";

En postman, crear 2 métodos "POST", el primero será para obtener el token, el segundo para validar el token y ordenar las fichas.

Método 1:

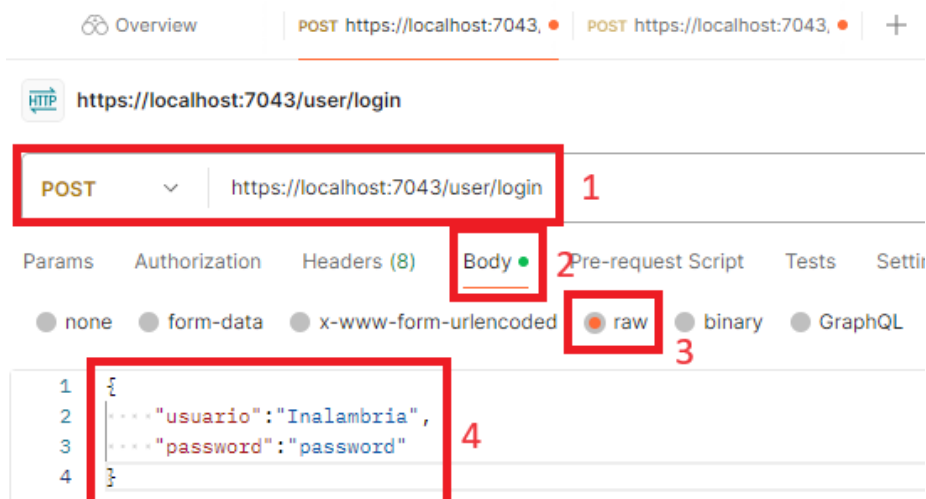
la URL es: <https://localhost:puerto/user/login> .En la pestaña "Body" seleccionar la opción "raw", y escribir el siguiente JSON:

```

{
    "usuario":"Inalambria",
    "password":"password"
}

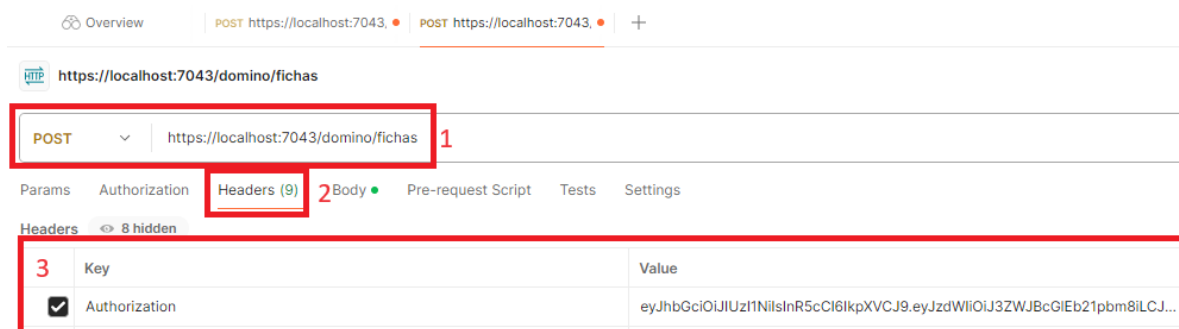
```

Ejemplo:



Método 2:

la URL es: <https://localhost:puerto/domino/fichas> . En la pestaña “Headers” agregar Key llamada "Authorization" y marcar el check, en Value se agregará el token generado, Ejemplo:

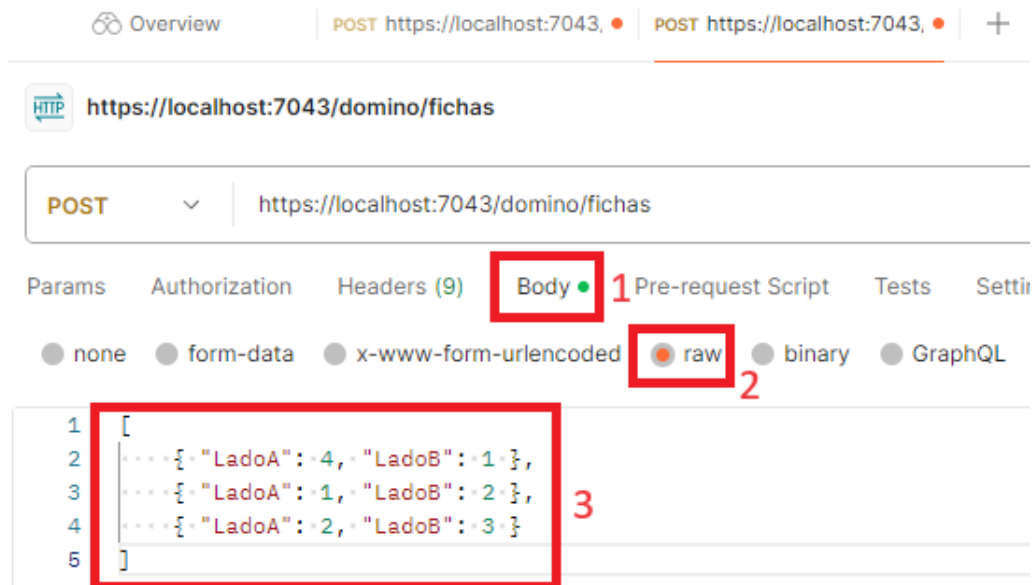


Cambiar a pestaña “Body”, seleccionar “raw”, y agregar el siguiente formato Json:

```

[
  { "LadoA": 4, "LadoB": 1 },
  { "LadoA": 1, "LadoB": 2 },
  { "LadoA": 2, "LadoB": 3 }
]
  
```

Ejemplo:



En esta sección del Body es donde se agregarán las fichas.

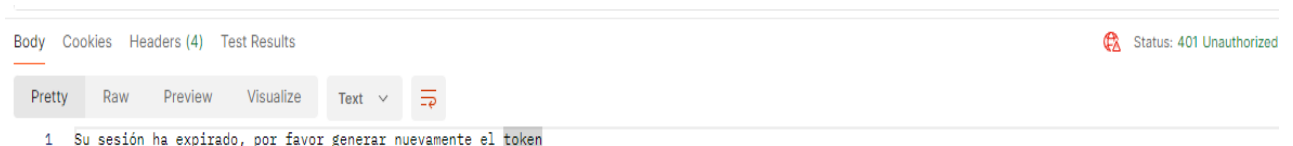
Seguido a esto es necesario clonar el repositorio e iniciar la solución desde visual studio para permitir a postman realizar las operaciones.

Ejemplos


Obtención exitosa del token 200:




Token expirado 401:




Error interno 400:


Body Cookies Headers (4) Test Results 

Pretty Raw Preview Visualize Text 

```
1 error al verificar las fichasIDX12729: Unable to decode the header '[PII of type 'System.String' is hidden. For more details, see https://aka.ms/Identity'
```

Fichas ordenadas correctamente 200:

Body Cookies Headers (4) Test Results 

Pretty Raw Preview Visualize JSON 

```
1 {
2   "success": true,
3   "message": "Fichas ordenadas correctamente",
4   "fichas": [
5     {
6       "ladoA": 3,
7       "ladoB": 2
8     },
9     {
10      "ladoA": 2,
11      "ladoB": 1
12    },
13    {
14      "ladoA": 1,
15      "ladoB": 3
16    }
17  ]
18 }
```

Fichas en formato inválido 400:

Body Cookies Headers (4) Test Results 

Pretty Raw Preview Visualize Text 

```
1 Las fichas no forman una cadena válida
```

Autor

Creado por Miguel Angel Perez

Contacto

Celular: 3125960639

Correo: migueitoo@hotmail.com

Muchas gracias por la oportunidad!