Brothersoft

Informe técnico aplicativo WebApi

Contenido

[Sistema de autenticación usado 1](#_Toc47428730)

[Controladores 3](#_Toc47428731)

[Modelos 7](#_Toc47428732)

[Ayudas (Helper) 7](#_Toc47428733)

[Configuración 8](#_Toc47428734)

# Sistema de autenticación usado

**JSON Web Token** (abreviado **JWT**) es un estándar abierto basado en JSON propuesto por IETF (RFC 7519) para la creación de tokens de acceso que permiten la propagación de identidad y privilegios o *claims* en inglés.

Estructura

Los JSON Web Tokens generalmente están formados por tres partes: un encabezado o *header*, un contenido o *payload*, y una firma o *signature*. El encabezado identifica qué algoritmo fue usado para generar la firma y normalmente se ve de la siguiente forma:

|  |
| --- |
| header = '{"alg":"HS256","typ":"JWT"}' |

HS256 indica que este token está firmado utilizando HMAC-SHA256.

El contenido contiene la información de los privilegios o *claims* del token:

|  |
| --- |
| payload = '{"loggedInAs":"admin","iat":1422779638}' |

El estándar sugiere incluir una marca temporal o timestamp en inglés, llamado iat para indicar el momento en el que el token fue creado.

La firma está calculada codificando el encabezamiento y el contenido en base64url, concatenándose ambas partes con un punto como separador:

|  |
| --- |
| key = 'secretkey'  unsignedToken = encodeBase64Url(header) + '.' + encodeBase64Url(payload)  signature = HMAC-SHA256(key, unsignedToken) |

En el token, las tres partes -encabezado, contenido y firma- están concatenadas utilizando puntos de la siguiente forma:

|  |
| --- |
| token = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' + encodeBase64Url(signature) # token es:  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSraSYS8EXBxLN\_oWnFSRgCzcmJmMjLiuyu5CSpyHI |

# Controladores

Por cada Requerimiento de aplicativo móvil vamos a tener un controlador con una acción por defecto que puede ser POST o GET.

* Estructura de un controlador tipo Api

|  |
| --- |
| [Route("api/[controller]")]  [ApiController]  [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]  public class CambioContrasenaController : ControllerBase  {  private IPrincipal \_principal;  private readonly ElectrosurContext \_context;  public CambioContrasenaController(  ElectrosurContext context,  IPrincipal principal)  {  \_context = context;  \_principal = principal;  }  [HttpPost]  public IActionResult Index([FromBody] CambioContrasena recuperar)  {  IActionResult response = Unauthorized();  try  {  var uniqueid = new Guid(recuperar.usruniqueid);  var usuario = \_context  .PdpUsrtUsuarioDelSistema  .Where(m => m.UsruniqueId == uniqueid)  .FirstOrDefault();  usuario.Usrcontrasena =  PasswordHash.GetMd5Hash(recuperar.contrasena);  usuario.Usrmodificado = DateTime.Now;  usuario.UsrultimoAcceso = DateTime.Now;  \_context.Update(usuario);  \_context.SaveChanges();  response = Ok(new { Estado = "OK" });  }  catch (Exception ex)  {  response = Ok(new  {  Estado = "Error",  Mensaje = ex.Message,  });  }  return response;  }  } |

Anotaciones

|  |
| --- |
| [Route("api/[controller]")]  [ApiController]  [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)] |

Las anotaciones las usuamos para describir en forma general el comportamiento del controlador API.

Aquí nos indica los siguiente

* + La ruta que tiene
  + Tipo de controlador
  + Qué sistema de autenticación usa, de ser anónimo estaría sin esta línea.

Constructor

|  |
| --- |
| public CambioContrasenaController(  ElectrosurContext context,  IPrincipal principal)  {  \_context = context;  \_principal = principal;  } |

A partir de la versión ASP Core, la forma mas adecuada de usar clases es por medio de “Inyección de dependencia” aquí agregamos los objetos que utilizaríamos en cada controlador

Acción

|  |
| --- |
| [HttpPost]  public IActionResult Index([FromBody] CambioContrasena recuperar)  {  IActionResult response = Unauthorized();  response = Ok(new { Estado = "OK" });  }  catch (Exception ex)  {  response = Ok(new  {  Estado = "Error",  Mensaje = ex.Message,  });  }  return response;  } |

La acción es la que realiza el proceso, tiene anotación que nos indicar que tipo de método es GET o POST, si no hay anotación por defecto es GET.

Como parámetros va a tener una clase, la cual se emparejará con los datos de envío por su nombre, los atributos de la clase deben estar en minúsculas.

Toda acción de un Api Controller debe devolver un objeto IActionResult en el cual le adjuntamos una respuesta tipo JSON

# CambioContrasenaController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Cambia la contraseña del usuario |
| **Ruta** | Api/cambiocontrasena |
| **Autorizacion** | JwtBearerDefaults |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto CambioContrasena  Cambia la contraseña del usuario por medio de su UsruniqueId |

# HistorialController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Devuelve la lista de pagos ordenamos por año y mes |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormHistorial  Recibimos UniqueId del usuario y buscamos sus pagos realizamos por año y mes. |
|  |  |  |

# HomeController

|  |  |
| --- | --- |
| **Tipo** | Controller |
| **Descripción** | Solo usamos a ver si la aplicación esta funcionando. |

# LoginController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Para el ingreso al sistema. |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormLogin  Recibimos un usuario y password si estamos en la base de datos, generamos un token tipo JWT. |
|  |  |  |

# PagoActualizarController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Una vez realizado el pago en el móvil actualizamos el pago ahora vamos a actualizar en la base datos y el sistema comercial. |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormPagoActualizar  Actualizamos el pago según el número de contrato. |
|  |  |  |

# PagoActualizarInvitadoController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Una vez realizado el pago en el móvil actualizamos el pago ahora vamos a actualizar en la base datos y el sistema comercial. |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormPagoActualizarInvitado  Actualizamos el pago según el número de contrato. |
|  |  |  |

# PagosController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Buscamos un contrato para pagarlo |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormPagos  Retorna una lista de recibos. |
|  |  |  |

# PagosInvitadoController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Buscamos un contrato para pagarlo |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
| Index | POST | Parámetro de entrada:  Objeto FormPagosInvitado  Retorna una lista de recibos. |
|  |  |  |

# RecuperarContrasenaController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Enviamos un correo una contraseña temporal. |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
|  |  | Parámetro de entrada:  Objeto FormRecuperarContrasena  Enviamos un correo con una clave temporal |
|  |  |  |

# RegistrarseController

|  |  |
| --- | --- |
| **Tipo** | Api/Controller |
| **Descripción** | Registramos un nuevo usuario al sistema |

Acciones

|  |  |  |
| --- | --- | --- |
| Acción | Método | Descripción |
|  |  | Parámetro de entrada:  Objeto FormRegistro  Registramos a un nuevo usuario y enviamos un correo de confirmación de correo. |
|  |  |  |

# Modelos

Estas son clases generadas por Entity Framework Core que es ORM de la base de datos.

Para la generación de estas clases vamos a la “Consola de Administrador de paquetes” y ejecutamos este comando:

|  |
| --- |
| Scaffold-DbContext "Server=ELSX007;Database=dbPagosVISA;user id=UsuarioPagosVISA;password=PagosVIS@;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Force |

* ElectrosurContext
* PdpOpepOperacion
* PdpPagpPago
* PdpUsrtUsuarioDelSistema

ModelViews

* ErrorViewModel
* FormRegistro .- Modelos usados en las peticiones, son clases contenedoras.
* Recibos

# Ayudas (Helper)

* PasswordHash

|  |  |
| --- | --- |
| **Tipo** | Clase genérica |
| **Descripción** | Esta clase encripta la contraseña en un hash para guardarlo en la base de datos |

Acciones

|  |  |
| --- | --- |
| Acción | Descripción |
| GetMd5Hash | Encripta la contraseña en un hash |

* SendEmailOutlook

|  |  |
| --- | --- |
| **Tipo** | Clase genérica |
| **Descripción** | Para en envió de correos |
| **Ubicación de plantillas de correo** | /wwwroot/plantilla\_email |

Acciones

|  |  |
| --- | --- |
| Acción | Descripción |
| Send | Envió de correo |

# Configuración

* ILoginService

|  |  |
| --- | --- |
| **Tipo** | Interface |
| **Descripción** | Con una función Generate JWT |

* LoginService

|  |  |
| --- | --- |
| **Tipo** | Clase Genérica |
| **Descripción** | Genera JWT |

Acciones

|  |  |
| --- | --- |
| Acción | Descripción |
| GenerateJWT | Generamos JWT y resgistramos el nombre de la persona que ingresa. |

* Register

|  |  |
| --- | --- |
| **Tipo** | Clase Genérica |
| **Descripción** | Registramos el servicio de LoginService |

* RegisterJWT

|  |  |
| --- | --- |
| **Tipo** | Clase Genérica |
| **Descripción** | Registramos el servicio de JWT |

* Program

|  |  |
| --- | --- |
| **Tipo** | Genérica |
| **Descripción** | Clase propia de la aplicación sin cambio alguno |

* Startup

|  |  |
| --- | --- |
| **Tipo** | Genérica |
| **Descripción** | Esta clase es la que inicia la aplicación y en donde configuramos los servicios |

Acciones

|  |  |
| --- | --- |
| Acción | Descripción |
| ConfigureServices | Configuramos lo siguiente   * Sesión * Dase de datos y cadena de conexión * Tipo aplicación en este caso es MVC que sería AddControllersWithViews * Configuraciones genéricas como AddSingleton , AddTransient |
| Configure | Activa las configuraciones anteriores como:   * Página de error * Uso de archivos estáticos * Uso de enrutamiento * Uso de autorizaciones * Uso de sesiones * Mapa de rutas |