using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Options;

using Microsoft.IdentityModel.Tokens;

using NuGet.Common;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Security.Cryptography;

using System.Text;

using WebLearning.Application.Services;

using WebLearning.Contract.Dtos;

using WebLearning.Domain.Entites;

using WebLearning.Persistence.ApplicationContext;

namespace WebLearning.Api.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class LoginController : ApiBase

{

private readonly ILogger<LoginController> \_logger;

private readonly AppSetting \_appSettings;

private readonly WebLearningContext \_webLearningContext;

private readonly ILoginService \_loginService;

public LoginController(ILogger<LoginController> logger, ILoginService loginService, IOptionsMonitor<AppSetting> optionsMonitor, WebLearningContext webLearningContext)

{

\_logger = logger;

\_appSettings = optionsMonitor.CurrentValue;

\_webLearningContext = webLearningContext;

\_loginService = loginService;

}

[HttpPost("Login")]

public async Task <ApiResponse> Validate(LoginDto loginDto)

{

//var account = \_webLearningContext.Accounts.SingleOrDefault(x => x.Email == loginDto.UserName && loginDto.Password == x.Password);

//var role = \_webLearningContext.Roles.SingleOrDefault(x => x.Id == account.RoleId);

//if (account == null)

//{

// return Ok(new ApiResponse()

// {

// Success = false,

// Message = "Invalid username/password"

// });

//}

//var token = await GenerateToken(account, role);

//return Ok(new ApiResponse()

//{

// Success = true,

// Message = "Authenticate success",

// Data = token

//});

return await \_loginService.Validate(loginDto);

}

private async Task<TokenModel> GenerateToken(Account account, Role role)

{

var jwtTokenHandler = new JwtSecurityTokenHandler();

var secretKeyBytes = Encoding.UTF8.GetBytes(\_appSettings.SecretKey);

var tokenDescription = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(new[] {

new Claim(ClaimTypes.Name, account.Email),

new Claim(JwtRegisteredClaimNames.Email, account.Email),

new Claim(JwtRegisteredClaimNames.Sub, account.Email),

new Claim(ClaimTypes.Role, role.RoleName),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),

new Claim("UserName", account.Email),

new Claim("Id", account.Id.ToString()),

//roles

}),

Expires = DateTime.UtcNow.AddMinutes(5),

SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(secretKeyBytes), SecurityAlgorithms.HmacSha512Signature)

};

var token = jwtTokenHandler.CreateToken(tokenDescription);

var accessToken = jwtTokenHandler.WriteToken(token);

var refreshToken = GenerateRefreshToken();

//Lưu database

var refreshTokenEntity = new RefreshToken

{

Id = Guid.NewGuid(),

JwtId = token.Id,

UserId = account.Id,

Token = refreshToken,

IsUsed = false,

IsRevoked = false,

IssuedAt = DateTime.UtcNow,

ExpiredAt = DateTime.UtcNow.AddHours(1)

};

await \_webLearningContext.AddAsync(refreshTokenEntity);

await \_webLearningContext.SaveChangesAsync();

return new TokenModel

{

AccessToken = accessToken,

RefreshToken = refreshToken

};

}

private string GenerateRefreshToken()

{

var random = new byte[32];

using (var rng = RandomNumberGenerator.Create())

{

rng.GetBytes(random);

return Convert.ToBase64String(random);

}

}

[HttpPost("RenewToken")]

public async Task<ApiResponse> RenewToken(TokenModel model)

{

//var jwtTokenHandler = new JwtSecurityTokenHandler();

//var secretKeyBytes = Encoding.UTF8.GetBytes(\_appSettings.SecretKey);

//var tokenValidateParam = new TokenValidationParameters

//{

// //tự cấp token

// ValidateIssuer = false,

// ValidateAudience = false,

// //ký vào token

// ValidateIssuerSigningKey = true,

// IssuerSigningKey = new SymmetricSecurityKey(secretKeyBytes),

// ClockSkew = TimeSpan.Zero,

// ValidateLifetime = false //ko kiểm tra token hết hạn

//};

//try

//{

// //check 1: AccessToken valid format

// var tokenInVerification = jwtTokenHandler.ValidateToken(model.AccessToken, tokenValidateParam, out var validatedToken);

// //check 2: Check alg

// if (validatedToken is JwtSecurityToken jwtSecurityToken)

// {

// var result = jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha512, StringComparison.InvariantCultureIgnoreCase);

// if (!result)//false

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Invalid token"

// });

// }

// }

// //check 3: Check accessToken expire?

// var utcExpireDate = long.Parse(tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Exp).Value);

// var expireDate = ConvertUnixTimeToDateTime(utcExpireDate);

// if (expireDate > DateTime.UtcNow)

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Access token has not yet expired"

// });

// }

// //check 4: Check refreshtoken exist in DB

// var storedToken = \_webLearningContext.RefreshTokens.FirstOrDefault(x => x.Token == model.RefreshToken);

// if (storedToken == null)

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Refresh token does not exist"

// });

// }

// //check 5: check refreshToken is used/revoked?

// if (storedToken.IsUsed)

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Refresh token has been used"

// });

// }

// if (storedToken.IsRevoked)

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Refresh token has been revoked"

// });

// }

// //check 6: AccessToken id == JwtId in RefreshToken

// var jti = tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Jti).Value;

// if (storedToken.JwtId != jti)

// {

// return Ok(new ApiResponse

// {

// Success = false,

// Message = "Token doesn't match"

// });

// }

// //Update token is used

// storedToken.IsRevoked = true;

// storedToken.IsUsed = true;

// \_webLearningContext.Update(storedToken);

// await \_webLearningContext.SaveChangesAsync();

// //create new token

// var account = await \_webLearningContext.Accounts.SingleOrDefaultAsync(nd => nd.Id == storedToken.UserId);

// var role = \_webLearningContext.Roles.SingleOrDefault(x => x.Id == account.RoleId);

// var token = await GenerateToken(account,role);

// return Ok(new ApiResponse

// {

// Success = true,

// Message = "Renew token success",

// Data = token

// });

//}

//catch (Exception ex)

//{

// return BadRequest(new ApiResponse

// {

// Success = false,

// Message = "Something went wrong"

// });

//}

return await \_loginService.RenewToken(model);

}

private DateTime ConvertUnixTimeToDateTime(long utcExpireDate)

{

var dateTimeInterval = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc);

dateTimeInterval.AddSeconds(utcExpireDate).ToUniversalTime();

return dateTimeInterval;

}

}

}