Back End Development 1
sesi 11

Generate API Docs : Swagger

Swagger

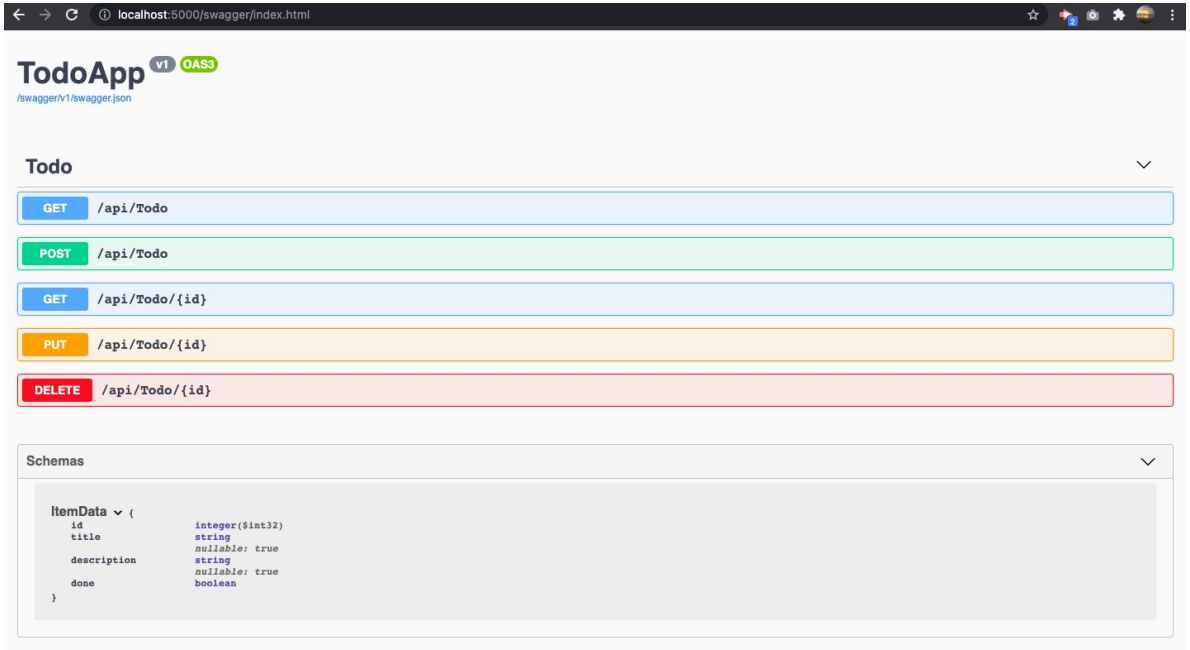
Bagi seorang Back End, selain berurusan dengan data juga harus bisa membuat dokumentasi sistem yang menaungi data-data nya entah itu nantinya di consume oleh pihak ke-3 (third party) ataupun oleh Front End,

Maka dari itu untuk membuat dokumentasi API itu susah - susah gampang. Kenapa ? karena banyak details yang harus di check terlebih dahulu apakah system ataupun API yang sudah dibuat itu akan menghasilkan data yang sama.

Untuk mempermudah hal ini ada tools bernama SWAGGER yang merupakan open source yang bisa di optimalkan oleh seorang back end. Selain memiliki interface yang menarik untuk ukuran dokumentasi, akan sangat membantu seorang Back End dalam menyajikan setiap endpoint yang akan/hendak di consume nantinya.

Good NEWS :
Sejak menggunakan .NET 5 ketika kita membuat dokumentasi webapi project Swagger yang sudah ter-integrasi dengan Aplikasi kita, nah cara melihat tampilan swagger bisa di cek pada :

`http://localhost:5000/swagger/index.html`



HACKTIV8

LIVE CASE : REFRESH JWT TOKEN

Nah karena kita memakai asp.net 5 maka kita tidak perlu pusing lagi untuk menyajikan dokumentasi yang kita buat nantinya,

Jadi pada LIVE SESSION kali ini kita akan membuktikan sepowerful apa swagger ini untuk output hasil akhir. Jadi pada project kali ini masih akan melanjutkan project sesi 10 sebelumnya tentang JWT.

Sebelumnya kita berhasil membuat otentikasi dengan JWT,

Nah disini kita harus paham tentang cara kerja token JWT ini, pada dasarnya Token JWT ini memiliki expire time. SEMAKIN PENDEK EXPIRE TIME ITU SEMAKIN AMAN.

Ada 2 Pilihan ketika hal ini terjadi :
1. Meminta User untuk login kembali, tentu ini bukan user-experience yang baik
2. Gunakan Refresh Token yang secara otomatis akan men-otorisasi user dengan generate Token JWT baru.

Tadi sempat di mention jika SEMAKIN PENDEK EXPIRE TIME token JWT maka ITU SEMAKIN AMAN.

Analogi :

- Ketika ada seorang hacker mencuri token JWT dan melakukan request ke server maka data kita akan dengan mudah di consume oleh hacker.

Lalu bagaimana dengan refresh token ?
Pada case yang sama token JWT yang berhasil dicuri akan menyusahkan untuk seoarang Hacker untuk melakukan request bahkan mengambil data di server kita. Kenapa ? karena Token JWT kita sudah EXPIRE / KADALUWARSA dan menjadi token yang tidak berguna.

Nah Pada Project ini kita akan belajar implementasi refresh token sesuai point 2 dengan melanjutkan project CRUD REST API kita kemarin.

LIVE CASE : REFRESH JWT TOKEN

Hal pertama yang kita lakukan adalah pada class startup dengan membuat TokenValidationParameters tersedia di seluruh aplikasi dengan menambahkan ke Dependency Injection.

```
45          var key = Encoding.ASCII.GetBytes(Configuration["JwtConfig:Secret"]);
46
47          var tokenValidationParams = new TokenValidationParameters {
48                  ValidateIssuerSigningKey = true,
49                  IssuerSigningKey = new SymmetricSecurityKey(key),
50                  ValidateIssuer = false,
51                  ValidateAudience = false,
52                  ValidateLifetime = true,
53                  RequireExpirationTime = false,
54                  ClockSkew = TimeSpan.Zero
55              };
56
57          services.AddSingleton(tokenValidationParams);
58
59          services.AddAuthentication(options => {
60              options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
61              options.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
62              options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
63          })
64          .AddJwtBearer(jwt => {
65              jwt.SaveToken = true;
66              jwt.TokenValidationParameters = tokenValidationParams;
67          });
```

startup.cs

Selanjutnya adalah kita butuh update function GenerateJwtToken() di AuthManagementController, dimana kita lihat ada Expire value pada TokenDesriptor yang diperbaiki dari ExpiryTimeFrame.

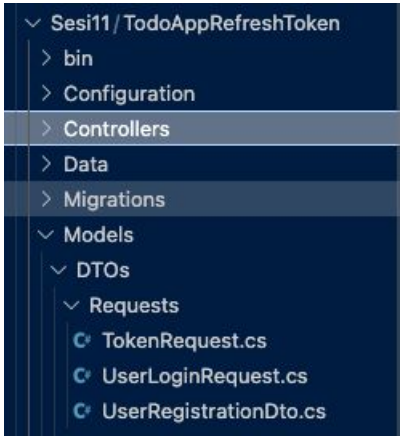
```
122         private string GenerateJwtToken(IdentityUser user)
123         {
124             var jwtTokenHandler = new JwtSecurityTokenHandler();
125
126             var key = Encoding.ASCII.GetBytes(_jwtConfig.Secret);
127
128             var tokenDescriptor = new SecurityTokenDescriptor
129             {
130                 Subject = new ClaimsIdentity(new []
131                 {
132                     new Claim("Id", user.Id),
133                     new Claim(JwtRegisteredClaimNames.Email, user.Email),
134                     new Claim(JwtRegisteredClaimNames.Sub, user.Email),
135                     new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
136                 }),
137                 Expires = DateTime.UtcNow.AddSeconds(30),
138                 SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signatur
139             };
140
141             var token = jwtTokenHandler.CreateToken(tokenDescriptor);
142             var jwtToken = jwtTokenHandler.WriteToken(token);
143
144             return jwtToken;
145         }
146     }
147 }
```

HACKTIV8

Next, pada AuthResult.cs di dalam folder konfigurasi, kita butuh menambahkan properti yang akan nge-update refresh token.

```
5   public class AuthResult
6   {
7       public string Token { get; set; }
8       public string RefreshToken { get; set; }
9       public bool Success { get; set; }
10      public List<string> Errors { get; set; }
11  }
```

Tambahkan class baru bernama TokenRequest didalam folder Models/DTOs/Requests dimana akan bertanggung jawab menerima request baru untuk mengelola refresh token.




```
1   using System.ComponentModel.DataAnnotations;
2
3   namespace TodoAppRefreshToken.Models.DTOs.Requests
4   {
5       public class TokenRequest
6       {
7           [Required]
8           public string Token { get; set; }
9
10          [Required]
11          public string RefreshToken { get; set; }
12      }
13  }
```

HACKTIV8

Buat model baru bernama RefreshToken.cs


```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using Microsoft.AspNetCore.Identity;

namespace TodoAppRefreshToken.Models
{
    public class RefreshToken
    {
        public int Id { get; set; }
        public string UserId { get; set; }
        public string Token { get; set; }
        public string JwtId { get; set; }
        public bool IsUsed { get; set; }
        public bool IsRevorked { get; set; }
        public DateTime AddedDate { get; set; }
        public DateTime ExpiryDate { get; set; }

        [ForeignKey(nameof(UserId))]
        public IdentityUser User {get;set;}
    }
}
```

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TodoAppRefreshToken.Models;

namespace TodoAppRefreshToken.Data
{
    public class ApiDbContext : IdentityDbContext
    {
        public virtual DbSet<ItemData> Items {get;set;}
        public virtual DbSet<RefreshToken> RefreshTokens {get;set;}

        public ApiDbContext(DbContextOptions<ApiDbContext> options)
            : base(options)
        {

        }
    }
}
```

Model sudah ditambahkan, selanjutnya kita update ApiDbContext.cs

HACKTIV8

Migrasi ApiDbContext kita

```
dotnet ef migrations add "Added refresh tokens table"
```

```
Maliks-MacBook-Air:TodoAppRefreshToken swijaya$ dotnet ef migrations add "Added refresh tokens table"
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
```

```
dotnet ef database update
```

```
Maliks-MacBook-Air:TodoAppRefreshToken swijaya$ dotnet ef database update
Build started...
Build succeeded.
Done.
```

HACKTIV8

Buat Endpoint baru : RefreshToken di AuthManagementController, Hal pertama yang kita harus lakukan adalah inject the TokenValidationParameters

```
27        private readonly TokenValidationParameters _tokenValidationParams;
28        private readonly ApiDbContext _apiDbContext;
29
30        public AuthManagementController(
31            UserManager<IdentityUser> userManager,
32            IOptionsMonitor<JwtConfig> optionsMonitor,
33            TokenValidationParameters tokenValidationParams,
34            ApiDbContext apiDbContext)
35        {
36            _userManager = userManager;
37            _jwtConfig = optionsMonitor.CurrentValue;
38            _tokenValidationParams = tokenValidationParams;
39            _apiDbContext = apiDbContext;
40        }
```

Setelah kita inject parameter yang di-require, kita harus update function GenerateToken termasuk didalamnya refresh token (masih pada file AuthManagementController)

```
154         private async Task<AuthResult> GenerateJwtToken(IdentityUser user)
155         {
156             var jwtTokenHandler = new JwtSecurityTokenHandler();
157 
158             var key = Encoding.ASCII.GetBytes(_jwtConfig.Secret);
159 
160             var tokenDescriptor = new SecurityTokenDescriptor
161             {
162                 Subject = new ClaimsIdentity(new []
163                 {
164                     new Claim("Id", user.Id),
165                     new Claim(JwtRegisteredClaimNames.Email, user.Email),
166                     new Claim(JwtRegisteredClaimNames.Sub, user.Email),
167                     new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())
168                 }),
169                 Expires = DateTime.UtcNow.AddSeconds(30), // 5-10
170                 SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
171             };
172 
173             var token = jwtTokenHandler.CreateToken(tokenDescriptor);
174             var jwtToken = jwtTokenHandler.WriteToken(token);
175 
176             var refreshToken = new RefreshToken()
177             {
178                 JwtId = token.Id,
179                 IsUsed = false,
180                 IsRevorked = false,
181                 UserId = user.Id,
182                 AddedDate = DateTime.UtcNow,
183                 ExpiryDate = DateTime.UtcNow.AddMonths(6),
184                 Token = RandomString(35) + Guid.NewGuid()
185             };
```

```
187            await _apiDbContext.RefreshTokens.AddAsync(refreshToken);
188            await _apiDbContext.SaveChangesAsync();
189
190            return new AuthResult() {
191                Token = jwtToken,
192                Success = true,
193                RefreshToken = refreshToken.Token
194            };
195        }
196        private string RandomString(int length)
197        {
198            var random = new Random();
199            var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
200            return new string(Enumerable.Repeat(chars, length)
201                .Select(x => x[random.Next(x.Length)]).ToArray());
202        }
```

Sekarang mari kita perbarui return kedua action yang telah kita ubah untuk GenerateJwtToken

```
var jwtToken  = await GenerateJwtToken(existingUser);

return Ok(jwtToken);
```

Action Login

```
var jwtToken = await GenerateJwtToken( newUser);

return Ok(jwtToken);
```

Action Register

Sekarang kita bisa mulai build RefreshToken action

```
125         [HttpPost]
126         [Route("RefreshToken")]
127         public async Task<IActionResult> RefreshToken([FromBody] TokenRequest tokenRequest)
128         {
129             if(ModelState.IsValid)
130             {
131                 var result = await VerifyAndGenerateToken(tokenRequest);
132 
133                 if(result == null) {
134                     return BadRequest(new RegistrationResponse() {
135                         Errors = new List<string>() {
136                             "Invalid tokens"
137                         },
138                         Success = false
139                     });
140                 }
141 
142                 return Ok(result);
143             }
144 
145             return BadRequest(new RegistrationResponse() {
146                 Errors = new List<string>() {
147                     "Invalid payload"
148                 },
149                 Success = false
150             });
151         }
```

HACKTIV8

```
197         private async Task<AuthResult> VerifyAndGenerateToken(TokenRequest tokenRequest)
198         {
199             var jwtTokenHandler = new JwtSecurityTokenHandler();
200
201             try
202             {
203                 // Validation 1 - Validation JWT token format
204                 var tokenInVerification = jwtTokenHandler.ValidateToken(tokenRequest.Token, _tokenValidationParams, out var validatedToken);
205
206                 // Validation 2 - Validate encryption alg
207                 if(validatedToken is JwtSecurityToken jwtSecurityToken)
208                 {
209                     var result = jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256, StringComparison.InvariantCultureIgnoreCase);
210
211                     if(result == false) {
212                         return null;
213                     }
214                 }
215
216                 // Validation 3 - validate expiry date
217                 var utcExpiryDate = long.Parse(tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Exp).Value);
218
219                 var expiryDate = UnixTimeStampToDateTime(utcExpiryDate);
220
221                 if(expiryDate > DateTime.UtcNow) {
222                     return new AuthResult() {
223                         Success = false,
224                         Errors = new List<string>() {
225                             "Token has not yet expired"
226                         }
227                     };
228                 }
229
```

HACKTIV8

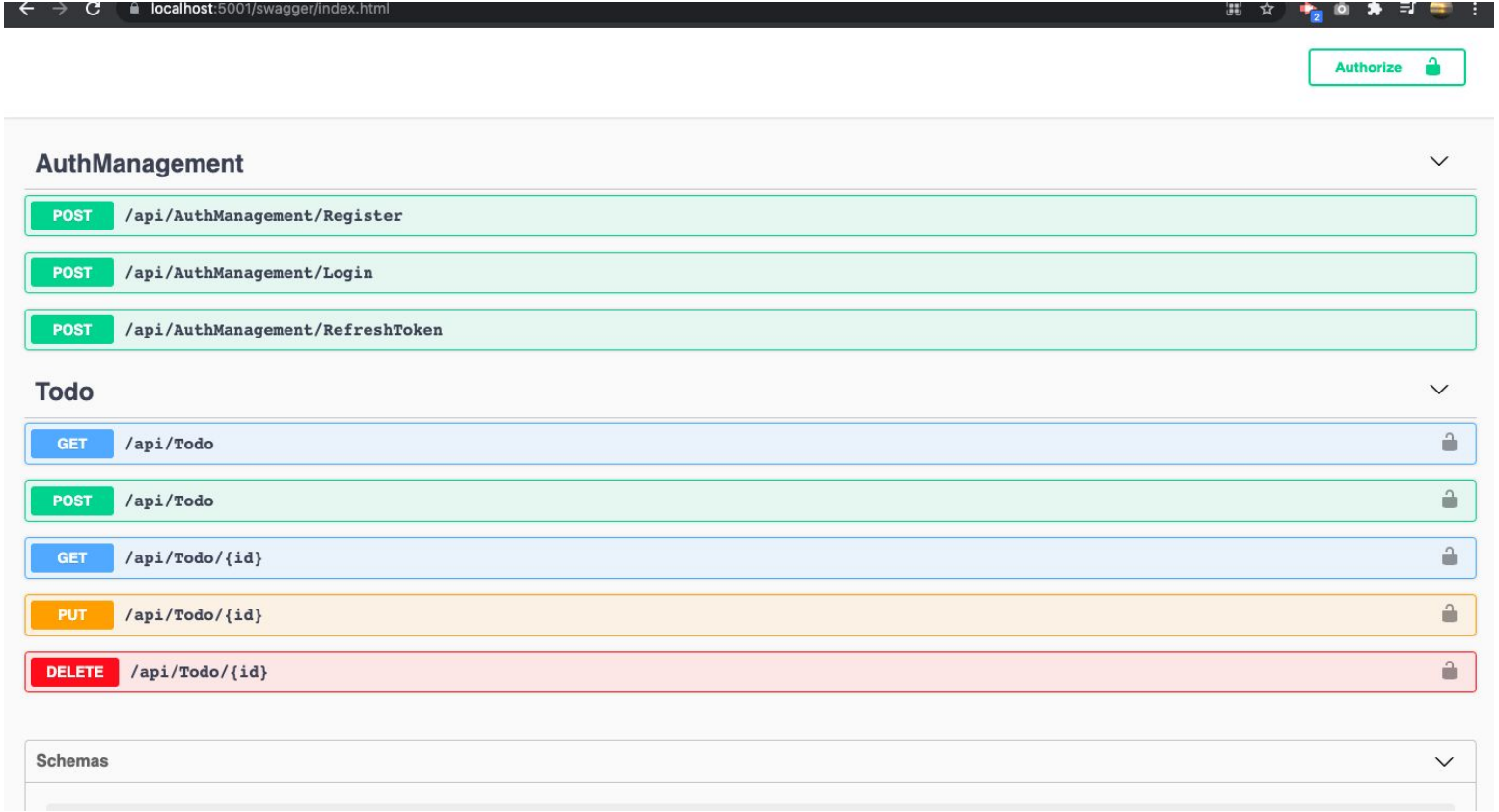
```
230            // validation 4 - validate existence of the token
231            var storedToken = await _apiDbContext.RefreshTokens.FirstOrDefaultAsync(x => x.Token == tokenRequest.RefreshToken);
232
233            if(storedToken == null)
234            {
235                return new AuthResult() {
236                    Success = false,
237                    Errors = new List<string>() {
238                        "Token does not exist"
239                    }
240                };
241            }
242
243            // Validation 5 - validate if used
244            if(storedToken.IsUsed)
245            {
246                return new AuthResult() {
247                    Success = false,
248                    Errors = new List<string>() {
249                        "Token has been used"
250                    }
251                };
252            }
253
254            // Validation 6 - validate if revoked
255            if(storedToken.IsRevorked)
256            {
257                return new AuthResult() {
258                    Success = false,
259                    Errors = new List<string>() {
260                        "Token has been revoked"
261                    }
262                };
263            }
264
```

```
265                 // Validation 7 - validate the id
266                 var jti = tokenInVerification.Claims.FirstOrDefault(x => x.Type == JwtRegisteredClaimNames.Jti).Value;
267
268                 if(storedToken.JwtId != jti)
269                 {
270                     return new AuthResult() {
271                         Success = false,
272                         Errors = new List<string>() {
273                             "Token doesn't match"
274                         }
275                     };
276                 }
277
278                 // update current token
279
280                 storedToken.IsUsed = true;
281                 _apiDbContext.RefreshTokens.Update(storedToken);
282                 await _apiDbContext.SaveChangesAsync();
283
284                 // Generate a new token
285                 var dbUser = await _userManager.FindByIdAsync(storedToken.UserId);
286                 return await GenerateJwtToken(dbUser);
287             }
288             catch(Exception ex)
289             {
290                 if(ex.Message.Contains("Lifetime validation failed. The token is expired.")) {
291
292                     return new AuthResult() {
293                         Success = false,
294                         Errors = new List<string>() {
295                             "Token has expired please re-login"
296                         }
297                     };
```

```
298
299                 } else {
300                     return new AuthResult() {
301                         Success = false,
302                         Errors = new List<string>() {
303                             "Something went wrong."
304                         }
305                     };
306                 }
307             }
308         }
309
310         private DateTime UnixTimeStampToDateTime(long unixTimeStamp)
311         {
312             var dateTimeVal = new DateTime(1970, 1,1,0,0,0,0, DateTimeKind.Utc);
313             dateTimeVal = dateTimeVal.AddSeconds(unixTimeStamp).ToUniversalTime();
314
315             return dateTimeVal;
316         }
```

Silahkan dijalankan dan dicoba dengan swagger ya :

https://localhost:5001/swagger/index.html



HACKTIV8

Kita coba dari modul register

Request :

AuthManagement

POST /api/AuthManagement/Register

Parameters

Cancel

No parameters

Request body

application/json

```
{
  "username": "test@gmail.com",
  "email": "test@gmail.com",
  "password": "PasswOrd1@1"
}
```

Execute

Clear

Response :

Responses

Curl

```
curl -X POST "https://localhost:5001/api/AuthManagement/Register" -H  "accept: */*" -H  "Content-Type: application/json" -d "
{\"username\":\"test@gmail.com\",\"email\":\"test@gmail.com\",\"password\":\"Passw0rd1@1\"}"
```

Request URL

```
https://localhost:5001/api/AuthManagement/Register
```

Server response

Code	Details
200	Response body


```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6ImVhNzZkNTRhLWU3NzAtNDIzMC1hZTlmLWFmYzRhYzMzN2Q4YiIsImVtYWlsIjoidGVzdEBnbWFpbC5jb20iLCJzdWIiOiJ0ZXN0QGdtYWlsLmNvbSIs-Imp0aSI6IjkyYmNjZWQ3LTM3YTAtNGYxOC1hYjZhLTE5MZQ1N2IyNTE4MSIsIm5iZiI6MTYzMjg5ODQxNiwiZXhwIjoxNjMyODk4NDQ2LCJpYXQiOjE2MzI4OTg0MTZ9.dm38hd059miFGQVszTRtBTvDYw-NS8rUMGWGjutNoWY",
  "refreshToken": "6MHM2FA592CHAL7IMIVXG6Z3IZOF1HJZ7OY5ce055e1-99fa-4c3a-80dd-11127612a52d",
  "success": true,
  "errors": null
}
```


Download

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Wed29 Sep 2021 06:53:36 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

HACKTIV8

Lanjut : login

Request :

POST /api/AuthManagement/Login

Parameters

Cancel

No parameters

Request body

application/json

```
{
  "email": "test@gmail.com",
  "password": "Passw0rd1@1"
}
```

HACKTIV8

Response :

Responses

Curl

```
curl -X POST "https://localhost:5001/api/AuthManagement/Login" -H  "accept: */*" -H  "Content-Type: application/json" -d "{\"email\":\"test@gmail.com\",\"password\":\"Passw0rd1@1\"}"
```

Request URL

```
https://localhost:5001/api/AuthManagement/Login
```

Server response

Code	Details
200	Response body


```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6ImVhNzZkNTRhLWU3NzAtNDIzMC1hZTlmLWFmYzRhYzMzN2Q4YiIsImVtYWlsIjoidGVzdEBnbWFpbC5jb20iLCJzdWIiOiJ0ZXN0QGdtYWlsLmNvbSIs-Imp0aSI6ImQ4YTZmOTUzLTYyYjYtNDcwZC1hNDgwLWM5NWM3OGJiYzg0NiIsIm5iZiI6MTYzMjg5ODU5NSwiZXhwIjoxNjMyODk4NjI1LCJpYXQiOjE2MzI4OTg1OTV9.VXqv4N8K99A6Zt51nbJKDpYIfP9fniE--A00F9hVd2k",
  "refreshToken": "XHCBUP1BU01OTJD22JRIKIK7PFVVQJZ7SP0fd27393a-6649-471e-813b-9454c703e3f8",
  "success": true,
  "errors": null
}
```


Download

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Wed29 Sep 2021 06:56:35 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

HACKTIV8

Coba Endpoint baru *RefreshToken

Request :

POST /api/AuthManagement/RefreshToken

Parameters

Cancel

No parameters

Request body

application/json


```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZCI6ImVhNzZkNTRhLWU3NzAtNDIzMC1hZTlmLWFmYzRhYzMzN2Q4YiIsImVtYWlsIjoidGVzdEBnbWFpbC5jb20iLCJzdWIiOiJ0ZXN0QGdtYWlsLmNvbSIsImp0aSI6ImQ4YTZmOTUzLTY
yYjYtNDcwZC1hNDgwLWM5NWM3OGJiYzg0NiIsIm5iZiI6MTYzMjg5ODU5NSwiZXhwIjoxNjMyODk4NjI1LCJpYXQiOjE2MzI4OTglOTV9.VXqv4N8K99A62t5lnbJKDpYIfP9fniE--A00F9hVd2k",
  "refreshToken": "XHCBUPlBU0lOTJD22JRIKIK7PFVVQJZ7SP0fd27393a-6649-471e-813b-9454c703e3f8"
}
```


Execute

Clear

HACKTIV8

Response :

Responses

Curl

```
curl -X POST "https://localhost:5001/api/AuthManagement/RefreshToken" -H  "accept: */*" -H  "Content-Type: application/json" -d "{\"token\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZ-
CI6ImVhNzZkNTRhLWU3NzAtNDIzMC1hZTlmLWFmYzRhYzMzN2Q4YiIsImVtYWlsIjoidGVzdEBnbWFpbC5jb20iLCJzdWIiOiJ0ZXN0QGdtYWlsLmNvbSIsImp0aSI6ImQ4YTZmOTUzLTYyYjYtNDcwZC1hNDgwLWM5NWM3OGJiYzg0NiIsIm5iZiI
6MTYzMjg5ODU5NSwiZXhwIjoxNjMyODk4NjI1LCJpYXQiOjE2MzI4OTg1OTV9.VXqv4N8K99A62t51nbJKDpYIfP9fniE--A00F9hVdZk\",\"refreshToken\":\"XHCBUP1BU01OTJD22JRIKIK7PFVVQJZ7SP0fd27393a-6649-471e-813b-
9454c703e3f8\"}"
```

Request URL

```
https://localhost:5001/api/AuthManagement/RefreshToken
```

Server response

Code	Details
200	Response body

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Token has expired please re-login"
  ]
}
```

Download

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Wed29 Sep 2021 06:58:23 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

HACKTIV8

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Token has expired please re-login"
  ]
}
```

Ini berjalan, tapi perlu diingat pada authManagementController expire time yang kita set adalah 30 detik.

```
169            Expires = DateTime.UtcNow.AddSeconds(30), // 5-10
170            SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
171        };
```

Lakukan login ulang dengan cepat (sebelum 30 detik) dan lihat apa response yang akan kalian terima.

Responses

Curl

```
curl -X POST "https://localhost:5001/api/AuthManagement/RefreshToken" -H  "accept: */*" -H  "Content-Type: application/json" -d "{\"token\":\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJJZ-
CI6ImVhNzZkNTRhLWU3NzAtNDIzMC1hZTlmLWFmYzRhYzMzN2Q4YiIsImVtYWlsIjoidGVzdEBnbWFpbC5jb20iLCJzdWIiOiJ0ZXN0QGdtYWlsLmNvbSIsImp0aSI6IjBkMjc4MDVkLWUyNjctNDI3MS1hMzNiLWI2MDgyMWU0MzMxNSIsIm5iZiI
6MTYzMjg5ODk5MSwiZXhwIjoxNjMyODk5MDIxLCJpYXQiOjE2MzI4OTg5OTF9.WgiVKkCglHl8ANu2pcbAm6enhn3-OtHbHgHZSDoreco\",\"refreshToken\":\"7C6AC5PF5CDDRDDFRTG4S72E820XF3TGRPD249cbcea-9082-4163-80b7-
25cd2f89c990\"}"
```

Request URL

```
https://localhost:5001/api/AuthManagement/RefreshToken
```

Server response

Code	Details
200	Response body

```
{
  "token": null,
  "refreshToken": null,
  "success": false,
  "errors": [
    "Token has not yet expired"
  ]
}
```

Download

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Wed29 Sep 2021 07:03:22 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Success	*No links*

HACKTIV8