**AuthController Sınıfı**

Bu sınıf, Music Tracker projesinde kulllanıcıların Spotify hesabı ile uygulamaya yetkilendirilmesini sağlamak amacıyla geliştirilmiştir. ASP.NET Core mimarisinde ControllerBase sınıfından türetilmiştir ve ISpotifyService ile IJwtService bağımlılıklarıyla çalışmaktadır. AuthController, kullanıcının Spotify ile bağlantısını kurmak ve oturum durumunu yönetmek için çeşitli uç noktalar sunar: login, callback, auth-status ve logout.

**Login Uç Noktası**

Bu uç nokta, Spotify OAuth 2.0 akışını başlatır. Kullanıcıya gerekli izinleri içeren bir Spotify yetkilendirme bağlantısı hazırlar ve kullanıcıyı bu bağlantıya yönlendirir.

Kodun Genel Akışı:

1. Spotify’dan istenen izinler (scope) belirlenir.
2. base\_url, response\_type, client\_id, scope ve redirect\_uri birleştirilerek yetkilendirme bağlantısı oluşturulur.

Kod Bloğu:

|  |
| --- |
| [HttpGet("login")]  public IActionResult Login()  {  var scopes = "user-read-recently-played playlist-modify-public playlist-modify-private";  var authUrl = $"https://accounts.spotify.com/authorize" +  $"?response\_type=code" +  $"&client\_id={ClientId}" +  $"&scope={Uri.EscapeDataString(scopes)}" +  $"&redirect\_uri={Uri.EscapeDataString(RedirectUri)}";  return Redirect(authUrl);  } |

**Callback Uç Noktası**

Bu uç nokta, Spotify OAuth 2.0 yetkilendirme sürecinin ikinci aşamasını gerçekleştiri. Kullanıcı Spotify tarafından yetkilendirildikten sonra, Spotify platformu redirect\_uri parametresi aracılığıyla bu uç noktaya bir yetkilendirme kodu (authorization code) gönderir. Bu kod, kullanıcıya ait erişim ve yenileme jetonlarını almak için kullanılır.

Kodun Genel Akışı:

1. Spotify’dan dönen authorization code alınır.
2. Bu kod client\_id, client\_secret ve redirect\_uri ile birlikte Spotify’a gönderilir.
3. Spotifydan alınan erişim (access\_token) ve yenileme (refresh\_token) jetonları ayrıştırılır.
4. Erişim jetonu kullanılarak Spotify’dan kullanıcı profili çekilir.
5. Kullanıcıya ait bilgiler (id, email, display\_name, profile\_image vb.) SpotifyUser nesnesi aracılığıyla bir araya getirilir.
6. Bu SpotifyUser, kullanıcının daha önce sisteme dahil olup olmaması durumuna göre bilgileri, veritabanında güncellenir veya veritabanına eklenir.
7. Uygulama içerisinde kullanılmak üzere bir JWT token oluşturulur ve güvenli bir çerezde (HttpOnly, Secure) saklanır.
8. Kullanıcı, ön yüze (localhost:5173) yönlendirilir.

Kod Bloğu:

|  |
| --- |
| [HttpGet("callback")]  public async Task<IActionResult> Callback(string code)  {  var client = \_httpClientFactory.CreateClient();  var postData = new Dictionary<string, string>  {  { "grant\_type", "authorization\_code" },  { "code", code },  { "redirect\_uri", RedirectUri },  { "client\_id", ClientId },  { "client\_secret", ClientSecret }  };  var response = await client.PostAsync("https://accounts.spotify.com/api/token", new FormUrlEncodedContent(postData));  var content = await response.Content.ReadAsStringAsync();  var tokenResponse = JsonSerializer.Deserialize<SpotifyTokenResponse>(content);  if (tokenResponse == null || string.IsNullOrEmpty(tokenResponse.AccessToken))  {  return BadRequest($"{tokenResponse?.AccessToken}, Failed to retrieve access token from Spotify.");  }  Console.WriteLine($"Token Scopes: {tokenResponse.Scope}");  client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", tokenResponse.AccessToken);  var userInfoResponse = await client.GetAsync("https://api.spotify.com/v1/me");  if (!userInfoResponse.IsSuccessStatusCode)  {  var errorContent = await userInfoResponse.Content.ReadAsStringAsync();  Console.WriteLine($"Spotify API Error: {errorContent}");  return StatusCode((int)userInfoResponse.StatusCode, errorContent);  }  var userJson = await userInfoResponse.Content.ReadAsStringAsync();  using var doc = JsonDocument.Parse(userJson);  var user = new SpotifyUser  {  Id = doc.RootElement.GetProperty("id").GetString(),  Email = doc.RootElement.TryGetProperty("email", out var emailEl) ? emailEl.GetString() : null,  DisplayName = doc.RootElement.TryGetProperty("display\_name", out var displayNameEl) ? displayNameEl.GetString() : null,  SpotifyToken = tokenResponse,  };  if (doc.RootElement.TryGetProperty("images", out var imagesEl)  && imagesEl.ValueKind == JsonValueKind.Array &&  imagesEl.GetArrayLength() > 0  )  {  var img = imagesEl[0];  user.ProfileImage = new ProfileImage  {  Url = img.GetProperty("url").GetString(),  Height = img.TryGetProperty("height", out var heightEl) ? heightEl.GetInt32() : 300,  Width = img.TryGetProperty("width", out var widthEl) ? widthEl.GetInt32() : 300,  };  }  await \_databaseService.AddOrUpdateUserAsync(user);  string token = \_jwtService.GenerateToken(user.Id);  Response.Cookies.Append("jwt", token, new CookieOptions  {  HttpOnly = true,  Secure = true,  SameSite = SameSiteMode.Strict,  Expires = DateTime.UtcNow.AddMinutes(double.Parse(\_config["JwtSettings:ExpiryMinutes"]))  });  return Redirect("http://localhost:5173/");  } |

**Auth-Status Uç Noktası**

Bu uç nokta, istemcinin oturum açma durumunu kontrol etmek için kullanılır. Kullanıcı arayüzü tarafından sıkça çağırılan bu API, kullanıcı sistemde aktif olarak giriş yapmış mı (JWT çerezi hala geçerli mi) ve geçerli bir Spotify erişim jetonuna sahip mi (ve jeton hala aktif mi), gibi durumları değenlendirerek uygun bir yanıt döner.

Kodun Genel Akışı:

1. İstemciden gelen istekle birlikte gönderilen JWT çerezi kontrol edilir.
2. JWT içinden kullanıcı kimliği (userId) çıkarılır.
3. Kullanıcıya ait veriler veri tabanından alınır.
4. Spotify erişim jetonunun süresi kontrol edilir.
   * Eğer jeton süresi dolmuşşa, refresh\_token ile Spotify API’ye bir çağrı yapılarak yeni bir jeton talep edilir.
   * Yeni jeton başarıyla alınırsa, kullanıcı giriş yapmış sayılır ve yeni jeton veri tabanında saklanır.
5. Tüm kontroller geçildiyse, loggedIn = true cevabı döndürülür. Aksi durumlarda loggedIn = false cevabı döndürülür.

Kod Bloğu:

|  |
| --- |
| [HttpGet("auth-status")]  public async Task<IActionResult> IsUserLoggedIn()  {  var token = Request.Cookies["jwt"];  if (string.IsNullOrEmpty(token))  return Ok(new { loggedIn = false });  var userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;  if (string.IsNullOrEmpty(userId))  return Ok(new { loggedIn = false });  var user = await \_databaseService.GetSpotifyUserAsync(userId);  if (user == null || string.IsNullOrEmpty(user.SpotifyToken.AccessToken))  return Ok(new { loggedIn = false });  var accessToken = user.SpotifyToken.AccessToken;  if (user.SpotifyToken.IsExpired())  {  var newAccessToken = await \_spotifyService.RefreshSpotifyAccessToken(user.SpotifyToken.RefreshToken);  if (!string.IsNullOrEmpty(newAccessToken))  {  user.SpotifyToken.AccessToken = newAccessToken;  await \_databaseService.AddOrUpdateUserAsync(user);  return Ok(new { loggedIn = true });  }  else  {  return Ok(new { loggedIn = false });  }  }  return Ok(new { loggedIn = true });  } |

**Logout Uç Noktası**

Bu uç nokta, kullanıcı oturumunu sonlandırmak amacıyla kullanılır. Oturum doğrulaması için kullanılan JWT çerezi, sunucu tarafından silinir ve kullanıcı sistemden güvenli bir çıkış yapmış olur. İstemci tarafında kullanıcı oturumu pasif hale gelir.

Kodun genel akışı:

1. Sunucuda tanımlı jwt adlı çerez silinir.
2. Kullanıcıya başarıyla çıkış yapıldığını belirten bir yanıt (200 OK) döndürülür.

Kod Bloğu:

|  |
| --- |
| [HttpPost("logout")]  public IActionResult Logout()  {  Response.Cookies.Delete("jwt");  return Ok(new { message = "Logged out successfully" });  } |