.NET 8 ve FastEndpoints ile Google OAuth 2.0 Entegrasyonu

Bu proje, modern .NET teknolojileri kullanılarak FastEndpoints çerçevesiyle Google OAuth 2.0 kimlik doğrulamasının nasıl entegre edileceğini gösteren çalışan bir demodur.

1. Amaç ve Motivasyon

Projenin temel amacı, bir .NET REST API'sinde harici oturum açma entegrasyonunu basitleştirmek ve göstermektir. FastEndpoints, minimal şablon yapısı ve performansa odaklı yönlendirme özellikleri nedeniyle tercih edilmiştir.

2. İncelenen Alanlar ve Alt Başlıklar

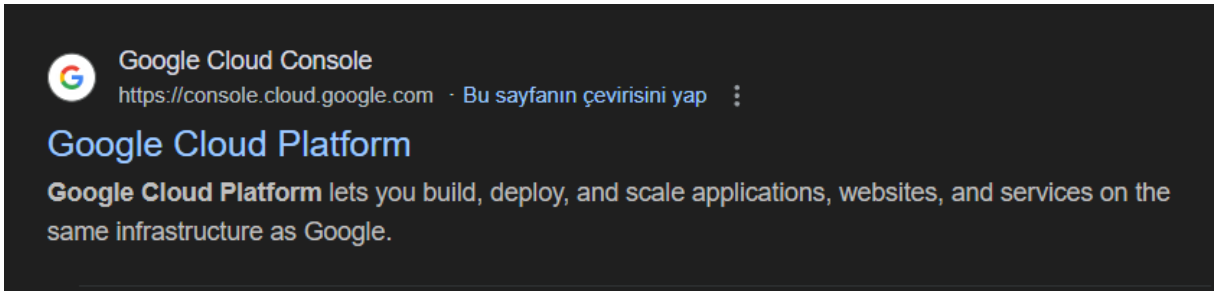
a. Google OAuth 2.0 yönlendirme ve signin işlemi
b. FastEndpoints kullanımının entegrasyonu ve ara yazılımlar
c. appsettings.json dosyasının güvenliğinin sağlanması

3. Uygulama Detayları

Proje, kullanıcıların Google hesaplarıyla kimlik doğrulaması yapmalarını ve ardından JWT (JSON Web Token) alarak API'ye erişim sağlamalarını mümkün kılar. FastEndpoints çerçevesi, uç noktaların ve ara yazılımların yapılandırılmasında kullanılır.

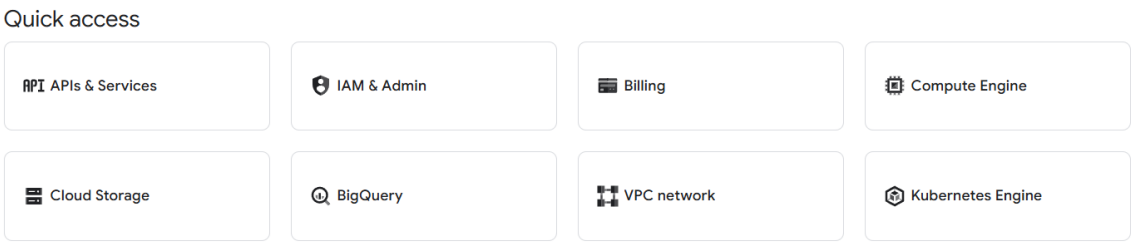
A. Google Konsol Kurulumu

Öncelikli olarak uygulamamızın çalışması için ***Client Id*** ve ***Client Secret*** değerlerine ihtiyacımız var. Bunu alabilmek için de öncelikli olarak Google Konsol üzerinden projemizi oluşturup gerekli adımları uygulamamız gerekiyor.



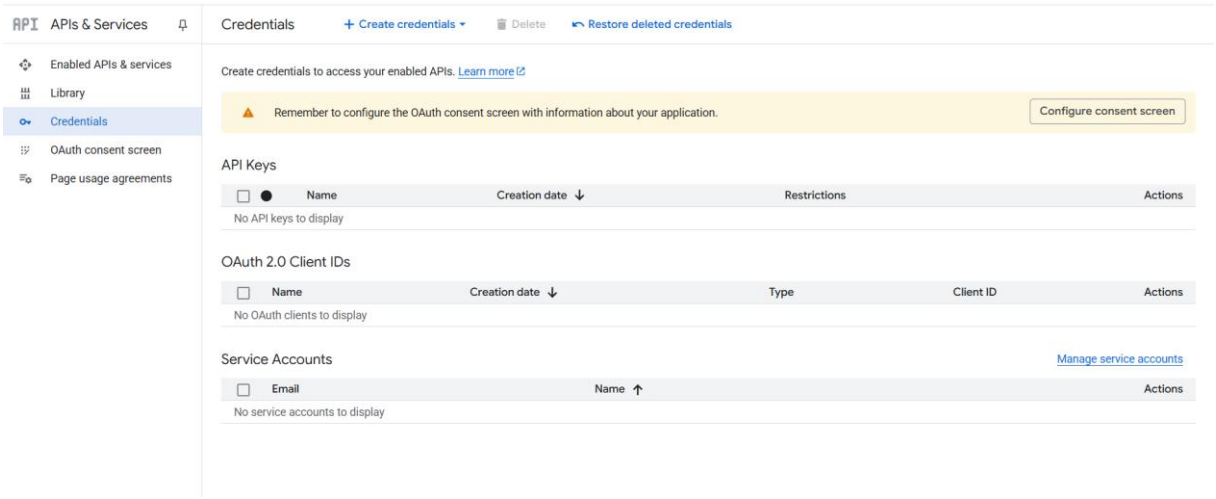
Şekil 1

Şekil 1 deki Google Cloud Platform'a giriş yapıyoruz. Daha sonra açılan sayfada Quick Access kısmı bizi sayfanın altında şekil 2'deki gibi karşılıyor.



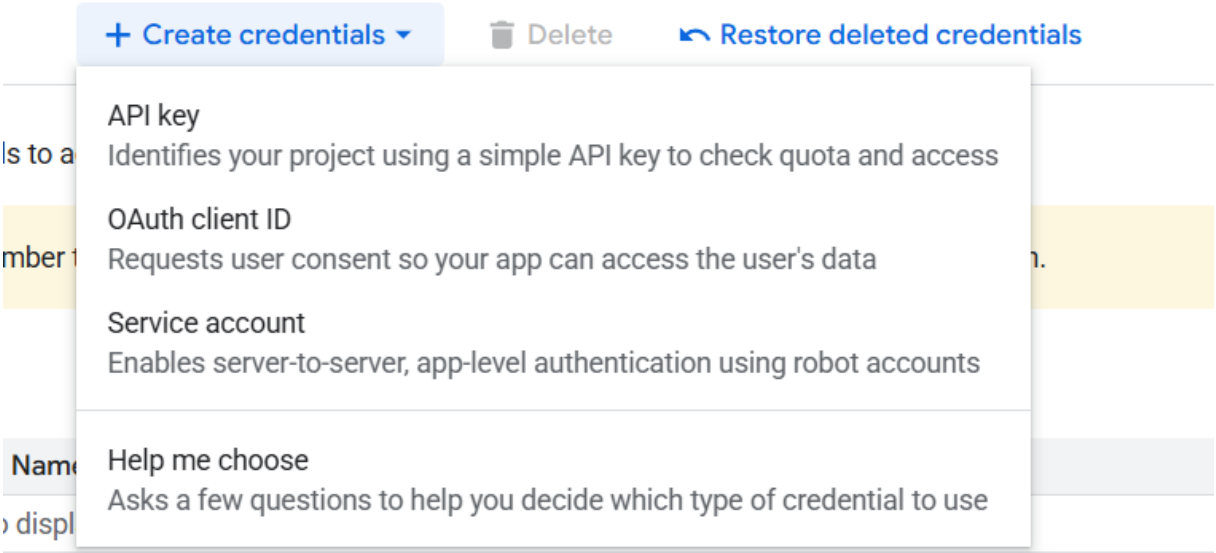
Şekil 2

Şekil 2'deki gibi olan sonuçta API's & Services kısmına tıklamamız gerekiyor.



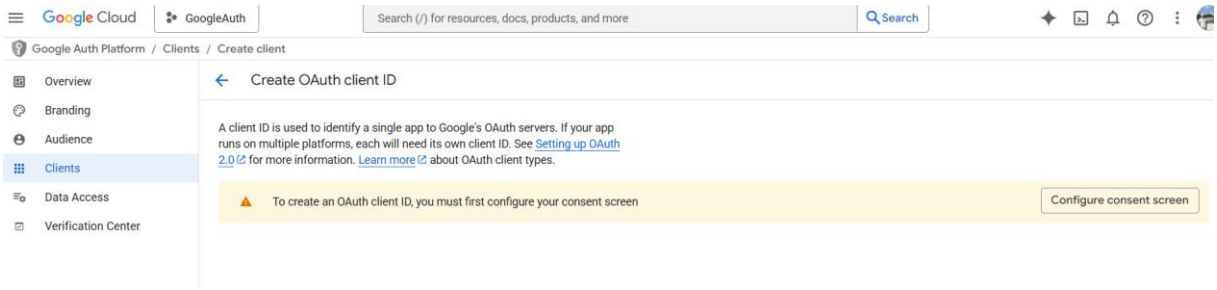
Şekil 3

Tıkladığımızda Şekil 3'deki sayfa bizleri karşılamakta. Burada ekranın üst kısmında yer alan + **Create Credentials** butonuna tıklamamız gerekli.



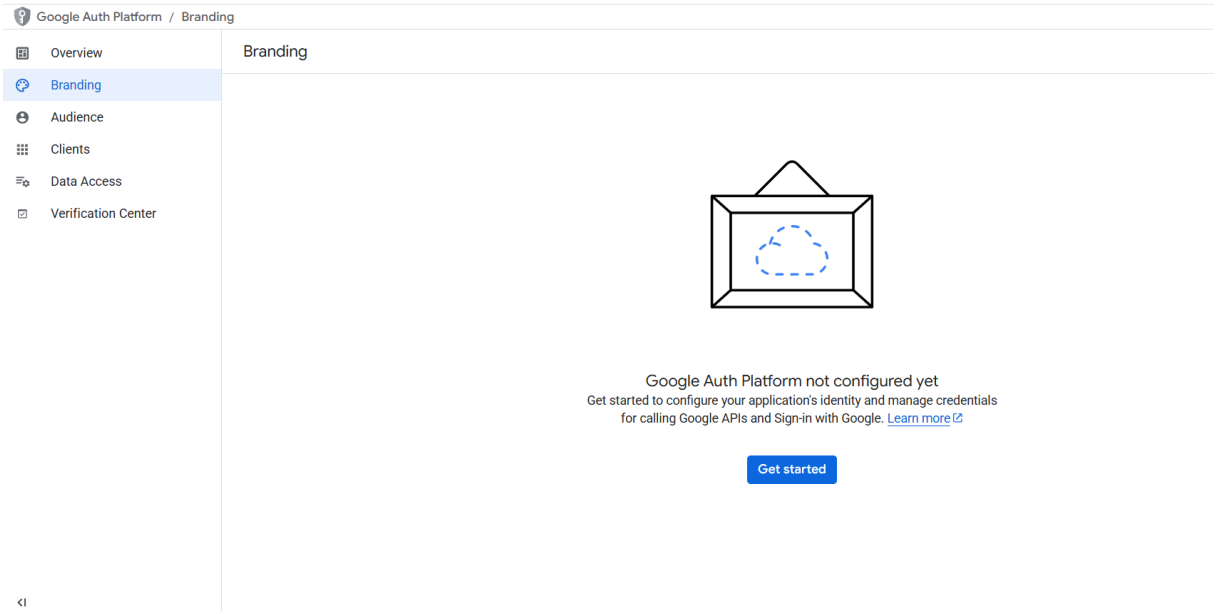
Şekil 4

Şekil 4'deki gibi bir pencere açılacak. Daha sonra burada **OAuth Client ID** Butonuna tıklamamız gerekiyor.



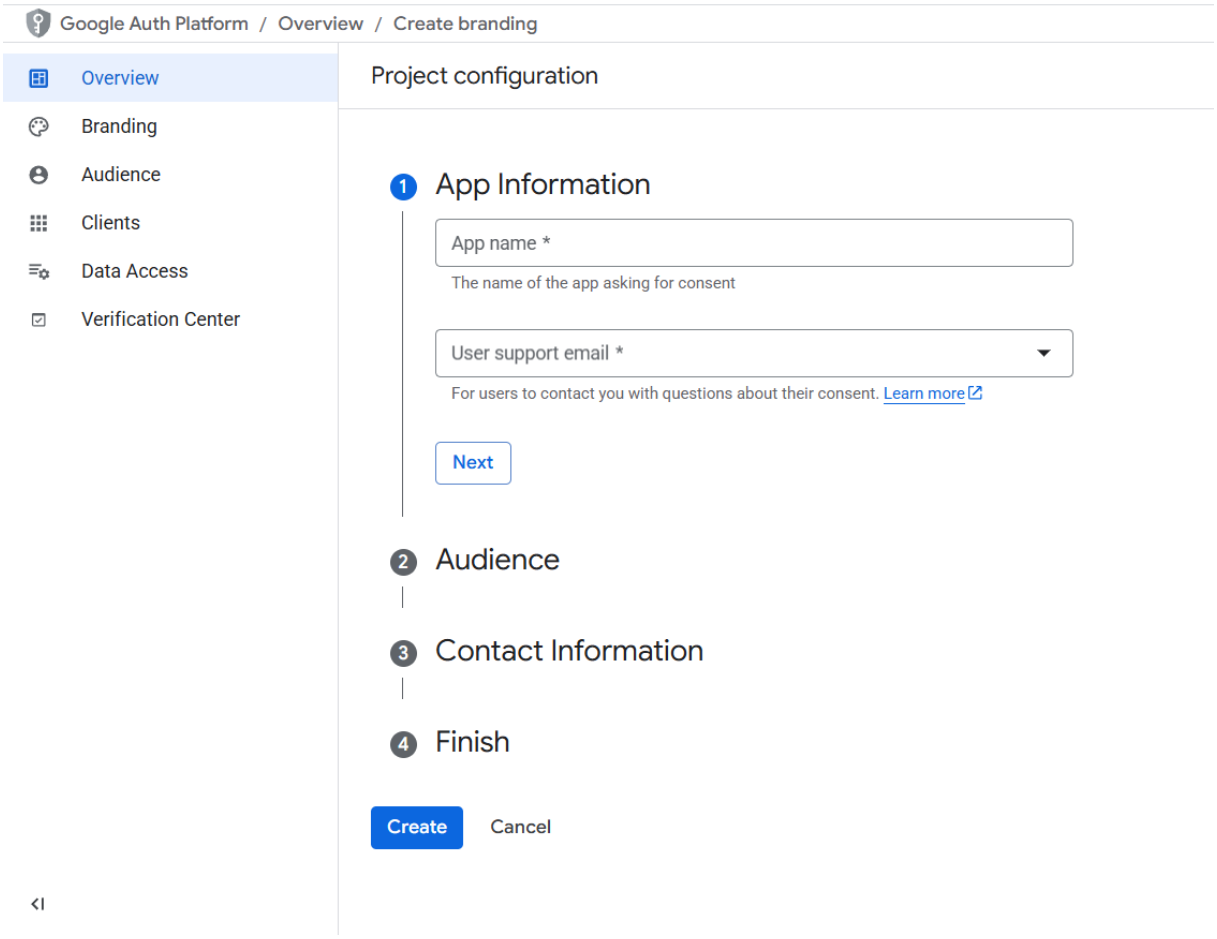
Şekil 5

Burada öncelikli olarak **Configure Consent Screen**'a tıklayarak gerekli ayarlamaları yapmamız gerekiyor.



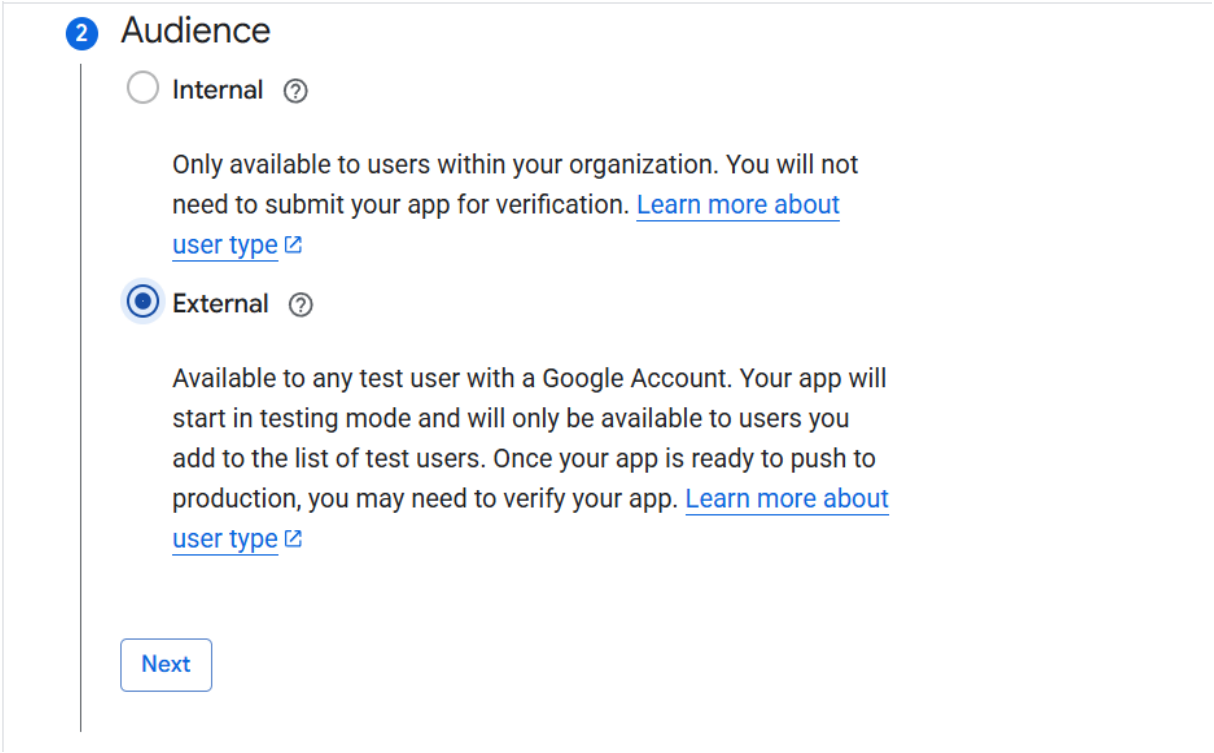
Şekil 6

Tıklayınca Şekil 6'daki bir sayfa açılacak. Burada da **Get Started** diyerek de gerekli kurulumları gerçekleştireceğiz.



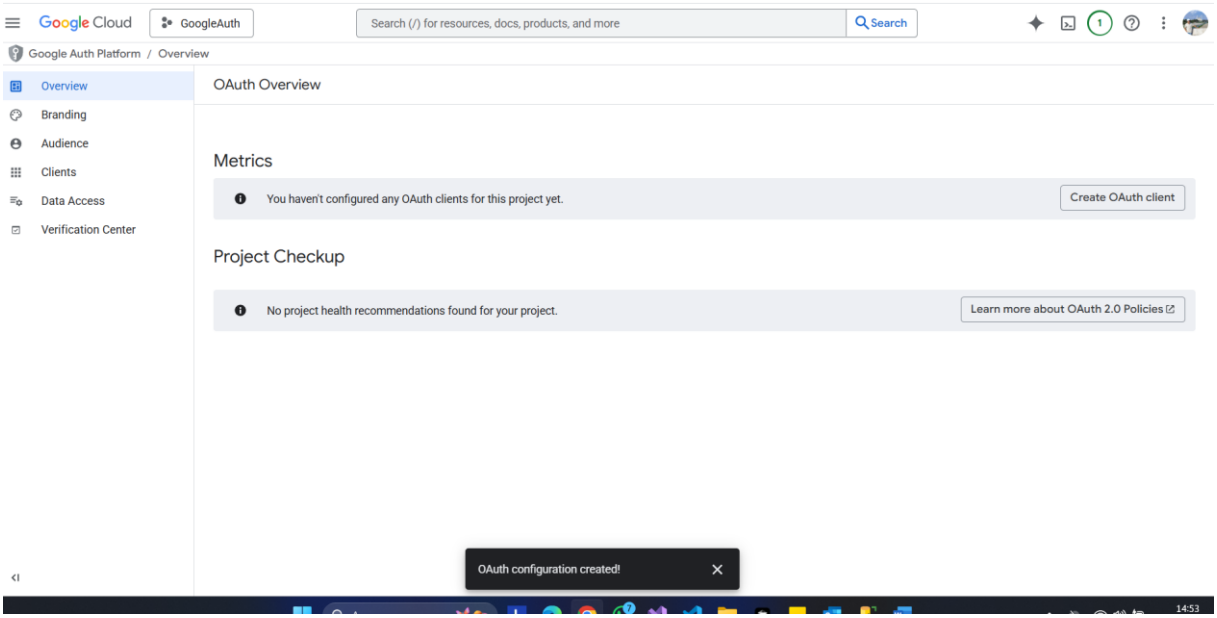
Şekil 7

Şekil 7 ‘deki gibi bir kısım açılıyor burayı projemize uygun bir vaziyette doldurmamız gerekli. Burada dikkat edilmesi gereken nokta ikinci adımdaki **Audience** kısmında **External** seçmemiz gerekiyor. Şekil 8’de bu kısmı görebilirsiniz.



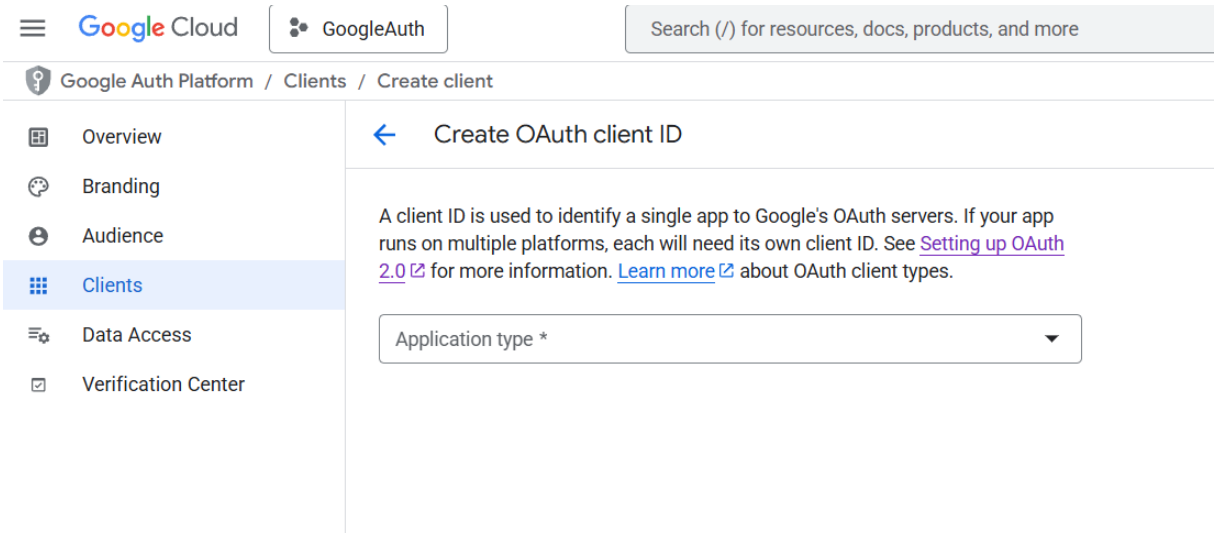
Şekil 8

Daha sonra diğer adımları da doldurarak finish butonuna tıklayarak bu kısmın kurulumu bitiriyoruz.



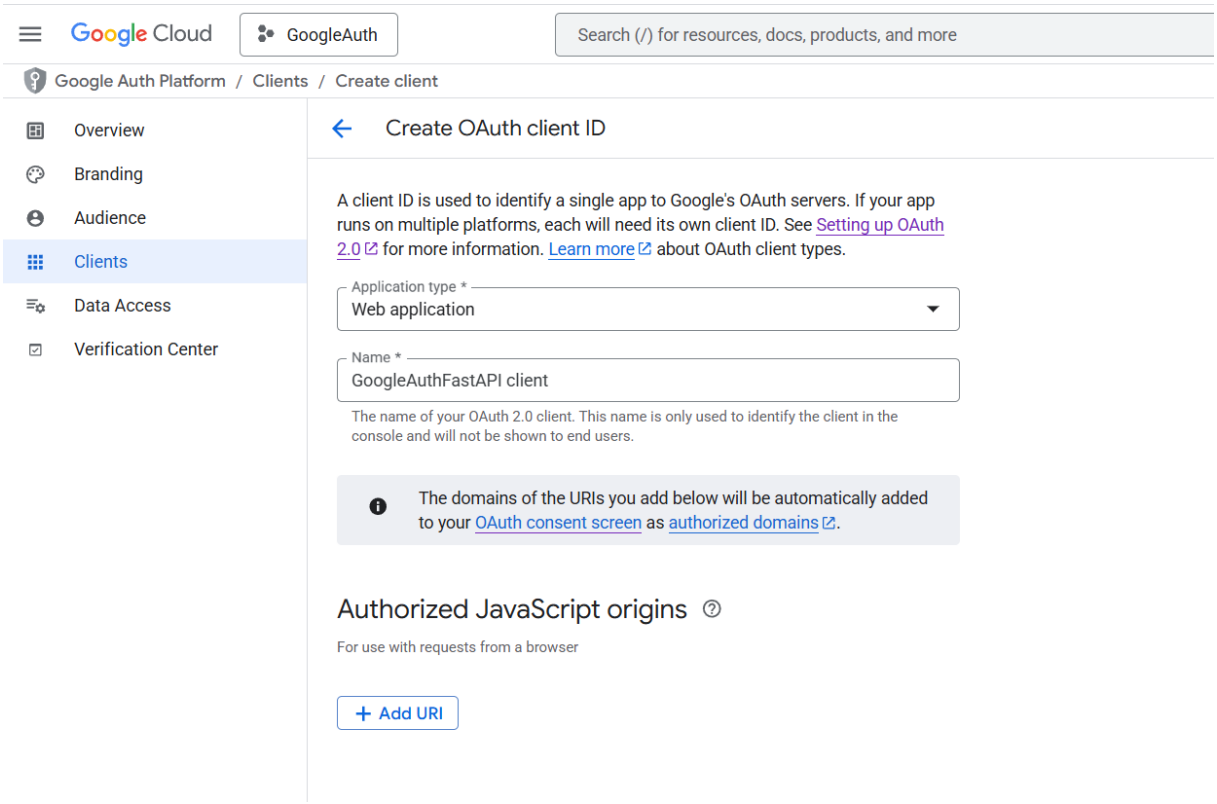
Şekil 9

Şekil 9'da görüldüğü üzere en altta bildirim olarak kurulumun başarılı olduğunu ve başarıyla ayarların oluşturulduğunu görebilmekteyiz. Şimdi Şekil 9'da gözüken **Create OAuth Client** butonuna tıklayalım.



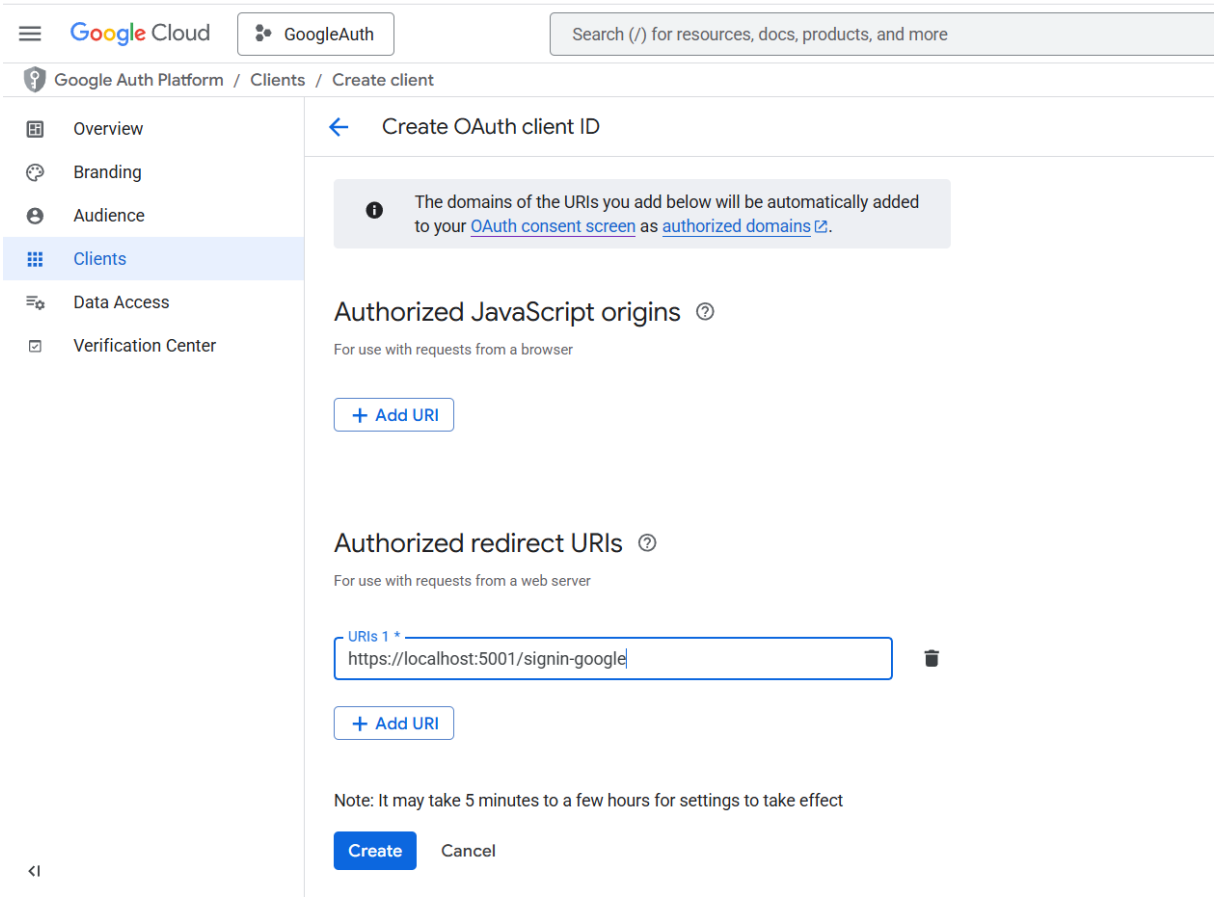
Şekil 10

Şekil 10'daki gibi bir pencere karşılayacak **Application Type'a** tıklayarak **Web Application** seçelim.



Şekil 11

Web Application seçildiğinde diğer kısımlarda gözükecek. Burada Name'de girdikten sonra sayfanın altına inelim.



Şekil 12

Şekil 12'deki gibi Authorized redirect URIs kısmı gözükecek. Buraya bir adres girmemiz gerekiyor. Ben bu adres için /signin-google seçtim. Bu adresin amacı da, Google Sign-In işleminde başarılı giriş sonrası **Google'ın kullanıcıyı tekrar uygulamaya yönlendireceği endpointtir**. Bu endpoint üzerinden Google, doğrulama kodunu gönderir ve uygulama bu kodu kullanarak **erişim token'ı alır ve kullanıcı bilgilerini elde eder.** Daha sonra **Create** butonuna tıklayarak işlemimizi sonlandırıyoruz. Bu işlem sonrasında bize Client Id ve Client Secret bilgilerimizi içeren bir pop-up ekran açılıyor. Bu bilgilerimizi ise ***appsetting.json*** dosyamızda kullanacağız.

B. .NET 8 & Fast-Endpoints Kurulumu

Google OAuth 2.0 için gerekli kimlik bilgilerini aldıktan sonra, bu bilgileri .NET 8 projemizde kullanabilmek için bazı yapılandırmalar yapmamız gerekiyor. Bu adımda **appsettings.json, Program.cs ve FastEndpoints** yapılandırmaları üzerinde duracağız.

1.1.1. Appsetting.json

Appsettings dosyamızın yapısı Şekil 13'deki gibi olmalıdır. Burada ClientId ve ClientSecret yerlerine az önce Google Konsol üzerinden aldığımız bilgileri girmemiz gerekiyor. Ayrıca Jwt kısmına da 256 bit'lik bir token ekledik. **Bu token, sistemimizin oluşturacağı JWT'leri imzalamak için kullanılan gizli bir anahtardır. Her kullanıcı için dinamik olarak üretilen JWT'lerin doğruluğu, bu key sayesinde sistem tarafından kontrol edilir. Böylece gelen isteklerin gerçekten bizim tarafımızdan verilen token'larla yapılıp yapılmadığı anlaşılır. Bu key, güvenlik nedeniyle sadece sunucu tarafında tutulmalı ve asla dışarıya paylaşılmamalıdır.**

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "Authentication": {
    "Google": {
      "ClientId": "YOUR_GOOGLE_CLIENT_ID",
      "ClientSecret": "YOUR_GOOGLE_CLIENT_SECRET"
    },
    "Jwt": {
      "Key": "YOUR_JWT_SECRET_KEY",
      "Issuer": "YOUR_ISSUER",
      "Audience": "YOUR_AUDIENCE"
    }
  }
}
```

Şekil 13

1.1.2. Program.cs Yapılandırması

Google OAuth ile kimlik doğrulama ve JWT üretimi işlemlerinin sorunsuz çalışabilmesi için Program.cs dosyamızda gerekli servis yapılandırmalarını gerçekleştirmemiz gerekiyor. Aşağıda Şekil 14'te bu yapılandırmanın detaylarını görebilirsiniz:

```
using FastEndpoints;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication.Google;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;

var builder = WebApplication.CreateBuilder(args);

//FastEndpoints
builder.Services.AddFastEndpoints();
builder.Services.AddAuthentication(options =>
{
    // Tarayıcı yönlendirmesi için cookie kullanılması gerekli
    options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = GoogleDefaults.AuthenticationScheme;
})
.AddCookie()
.AddGoogle(options =>
{
    options.ClientId = builder.Configuration["Authentication:Google:ClientId"];
    options.ClientSecret = builder.Configuration["Authentication:Google:ClientSecret"];
})
.AddJwtBearer(options =>
{
    //API isteklerinde JWT token doğrulama kuralları belirliyoruz
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateIssuer = true,
        ValidateAudience = true,
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = builder.Configuration["Authentication:Jwt:Issuer"],
        ValidAudience = builder.Configuration["Authentication:Jwt:Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configur
    };
});

builder.Services.AddAuthorization();

builder.Services.ConfigureApplicationCookie(options =>
{
    options.Cookie.SameSite = SameSiteMode.None;
    options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
});

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}

app.UseHttpsRedirection();
app.UseAuthentication();
app.UseAuthorization();
app.UseFastEndpoints();

app.Run();
```

Şekil 14

1.1.3.Fast-Endpoints

Bu projede endpoint tanımlamaları ve yönlendirmeler için geleneksel Controller tabanlı yapı yerine modern ve minimal bir yaklaşım sunan FastEndpoints kütüphanesi tercih edilmiştir.
FastEndpoints, .NET 8 ile tam uyumlu, performansa odaklı ve az boilerplate kod ile REST API geliştirmeye olanak tanıyan bir yapıdır. Login işlemi için ve login sonrası işlemimiz için iki tane endpoint adresi belirledik. Bunlar **"/api/auth/google-login" ve "/auth/callback"** endpointleridir. İlk tanımlanan enpoint adresine girince :

```
using FastEndpoints;
using Microsoft.AspNetCore.Authentication.Google;
using Microsoft.AspNetCore.Authentication;

namespace GoogleAuthFastAPI.Endpoints.Google
{
    public class GoogleLoginEndpoint : EndpointWithoutRequest<object>
    {
        public override void Configure()    //Endpoint noktası
        {                                   //Burada oluşturduğumuz endpointe istek geldiğinde google signin gelecek.
            Get("/api/auth/google-login");
            AllowAnonymous();
        }

        public override async Task<object> ExecuteAsync(CancellationToken ct)
        {
            var props = new AuthenticationProperties    //Login işleminden sonra hangi URL'e yönleneceğimizi seçtik
            {
                RedirectUri = "/auth/callback"
            };


            var challengeResult = Results.Challenge(props, new[] { GoogleDefaults.AuthenticationScheme });


            await challengeResult.ExecuteAsync(HttpContext);

            return new { }; // Buraya uğramıyor zaten
        }
    }
}
```

Şekil 15

Şekil 15'deki görebileceğiniz gibi Google tarafından sağlanan signin sürecine yönlendiriliyor. Bu login işleminden sonra ise sistemimizde tanımladığımız diğer endpoint olan **"/auth/callback"** adresine yönlendirme yapılıyor. Şekil 16'da da bu süreci görebilirsiniz.

```
using FastEndpoints;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.IdentityModel.Tokens;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;

namespace GoogleAuthFastAPI.Endpoints.Google
{
    public class GoogleCallbackEndpoint : EndpointWithoutRequest
    {
        private readonly IConfiguration _config;

        public GoogleCallbackEndpoint(IConfiguration config)
        {
            _config = config;
        }

        public override void Configure()
        {
            Get("/auth/callback");
            AllowAnonymous();
        }

        public override async Task HandleAsync(CancellationToken ct)    // Burada işlem son
        {
            var result = await HttpContext.AuthenticateAsync(CookieAuthenticationDefaults.A

            if (!result.Succeeded)
            {
                await SendAsync(new { error = "Login failed" }, 400, ct);
                return;
            }

            var email = result.Principal?.FindFirstValue(ClaimTypes.Email); //Google'dan ge
            var name = result.Principal?.FindFirstValue(ClaimTypes.Name);

            // JWT üretiyoruz
            var token = GenerateToken(email!, name ?? "");

            await SendAsync(new { token, email, name }, 200, ct);
        }

        private string GenerateToken(string email, string name)
        {
            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Authenticati
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var claims = new[]
            {
            new Claim(ClaimTypes.Email, email),
            new Claim(ClaimTypes.Name, name),
            new Claim("email", email)
        };

            var token = new JwtSecurityToken(
                issuer: _config["Authentication:Jwt:Issuer"],
                audience: _config["Authentication:Jwt:Audience"],
                claims: claims,
                expires: DateTime.UtcNow.AddHours(1),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }

}
```

Şekil 16

1.1.4.Giriş Yapan Kullanıcının Bilgilerini Okuma

Giriş yapan kullanıcının bilgilerini HttpContext.User üzerinden veya doğrudan User nesnesi ile erişebiliyoruz

```
var result = await HttpContext.AuthenticateAsync(CookieAuthenticationDefaults.AuthenticationScheme);

if (!result.Succeeded)
{
    await SendAsync(new { error = "Login failed" }, 400, ct);
    return;
}

var email = result.Principal?.FindFirstValue(ClaimTypes.Email); //Google'dan gelen email, name bilgisi
var name = result.Principal?.FindFirstValue(ClaimTypes.Name);
```

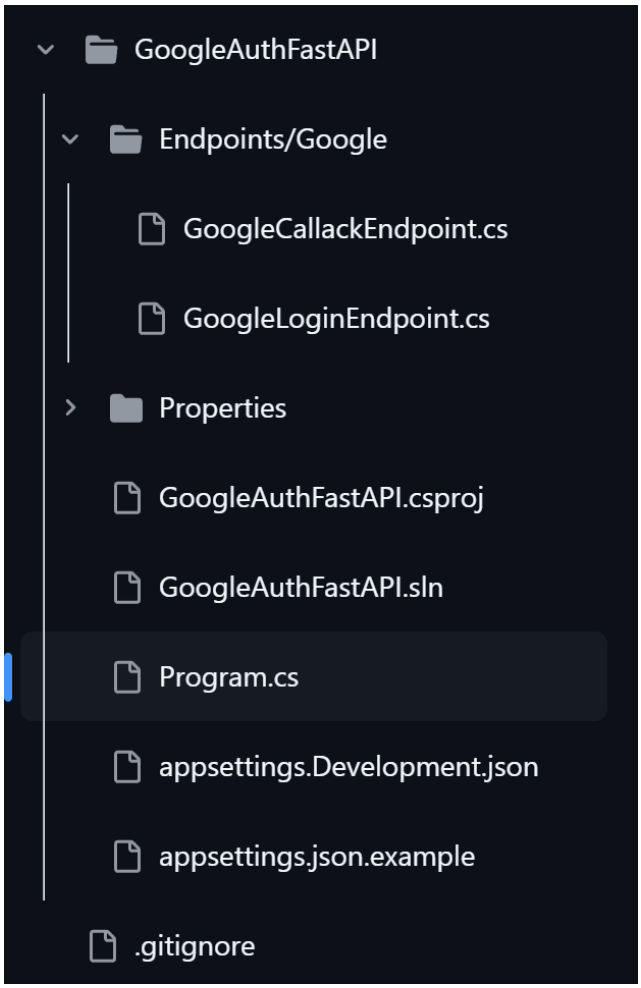
4. Güvenlik Önlemleri

Bu projede kullanıcı verilerinin korunması ve kimlik doğrulama süreçlerinin güvenli şekilde yürütülmesi amacıyla bazı temel güvenlik önlemleri uygulanmıştır:

- **JWT Token Güvenliği:** Her kullanıcıya özel olarak oluşturulan **JWT'ler**, 256 bit uzunluğunda gizli bir anahtar ile imzalanmaktadır. Bu anahtar yalnızca sunucu tarafında saklanır ve token doğrulama işlemleri için kullanılır.
- **Google OAuth Güvenliği:** Yalnızca Google Console üzerinden tanımlanmış olan yönlendirme adresleri (Authorized Redirect URI) kullanılmakta, böylece dış müdahalelere karşı koruma sağlanmaktadır.
- **Çerez (Cookie) Güvenliği:** Kullanıcı oturumları HTTPS üzerinden iletilen güvenli çerezlerle yönetilmektedir. Gerekli ayarlamalar ile tarayıcılar arası güvenli iletişim sağlanmıştır.
- **Erişim Kontrolü:** Uç noktalara erişim, kullanıcı doğrulamasına göre sınırlandırılmıştır. Yalnızca kimliği doğrulanmış kullanıcılar belirli endpoint’lere erişebilmektedir.

5. Proje Yapısı

Proje, .**NET 8** ve **FastEndpoints** kullanılarak sade ve işlevsel bir klasör yapısıyla oluşturulmuştur. Kodların daha okunabilir ve yönetilebilir olması adına ilgili bileşenler klasörler altında gruplandırılmıştır. Proje yapısı Şekil 17'te görülmektedir



Şekil 17

6. Sonuç

Bu projede, **.NET 8** ve **FastEndpoints** kullanılarak **Google OAuth 2.0** kimlik doğrulama süreci başarılı bir şekilde entegre edilmiştir. Kullanıcıların Google hesapları ile sisteme güvenli bir şekilde giriş yapmaları sağlanmış, ardından her kullanıcı için özel olarak **JWT** üretimi gerçekleştirilmiştir. Uygulama yapısı, modüler bir mimariyle kurgulanmış olup, FastEndpoints ile geliştirilen endpoint'ler sayesinde sade ve yönetilebilir bir kod düzeni elde edilmiştir. Aynı zamanda appsettings dosyaları üzerinden yapılan yapılandırmalar ile uygulama güvenliği ön planda tutulmuş, hem doğrulama hem de yetkilendirme süreçleri sağlıklı bir şekilde işletilmiştir. Geliştirilen bu yapı, benzer kimlik doğrulama ihtiyaçları olan modern web uygulamaları için esnek ve yeniden kullanılabilir bir çözüm sunmaktadır.