

# Использование OpenID и OAuth в ASP.NET Identity

[ASP.NET](#) --- [ASP.NET Identity](#) --- [Использование OpenID и OAuth](#)

1 Одним из преимуществ системы разрешений платформы ASP.NET Identity является то, что информация о пользователях может поступать из внешних источников, даже та информация, которая используется для их идентификации. Это означает, что другие системы могут проверять подлинность пользователей от имени приложения и ASP.NET Identity использует эту идею для создания легкой *системы аутентификации на третьей стороне (Third-Party Authentication)*, через такие сервисы, как Google, Microsoft, Facebook, Twitter, ВКонтакте и т.д. Вы наверняка видели подобные системы аутентификации на многих сайтах, где не нужно проходить процесс регистрации, а достаточно войти в приложение через свой профиль в социальной сети.

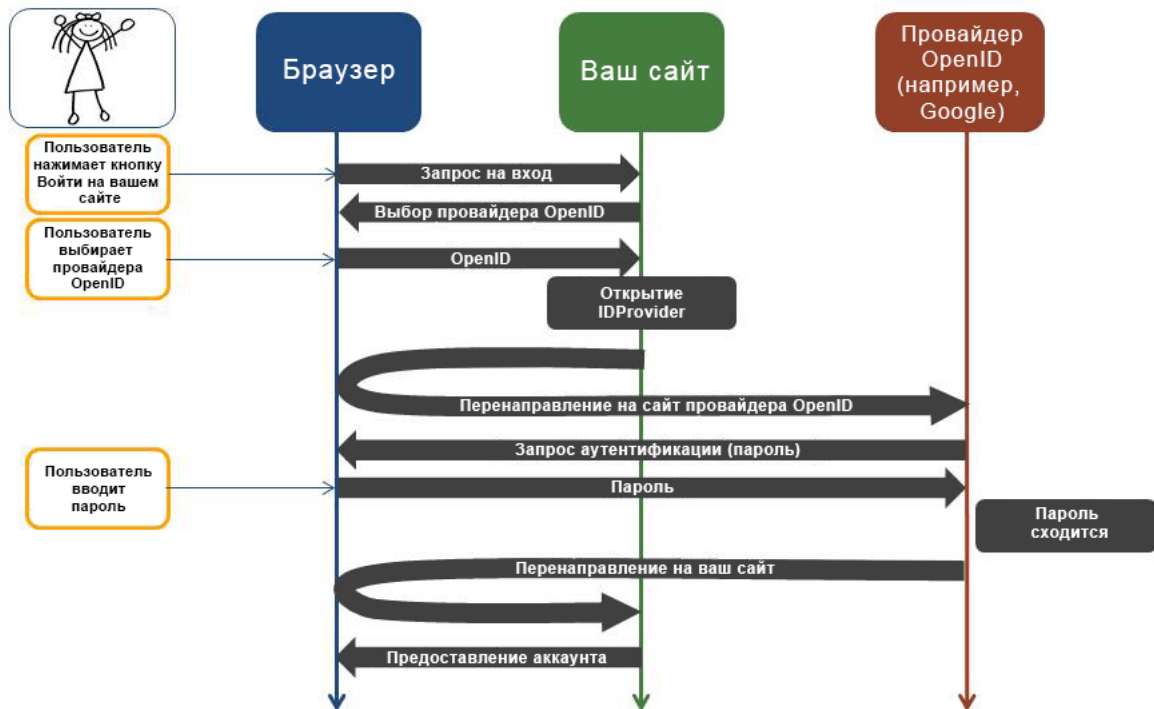
Есть множество преимуществ такой системы аутентификации: многие пользователи уже имеют профили в различных социальных сетях, пользователи могут использовать двухфакторную аутентификацию и вы не должны управлять учетными записями пользователя в приложении.

В современных приложениях, такие системы аутентификации реализуются на основе протоколов OpenID и OAuth, описывающих взаимодействие приложения со сторонней системой. Ниже кратко описаны оба этих протокола и приведены схемы их работы:

## OpenID

Этот протокол используется для аутентификации пользователей через другие интернет-ресурсы и проверяет, действительно ли пользователь тот — за кого себя выдает. На рисунке ниже приведена схема работы OpenID:

## OpenID

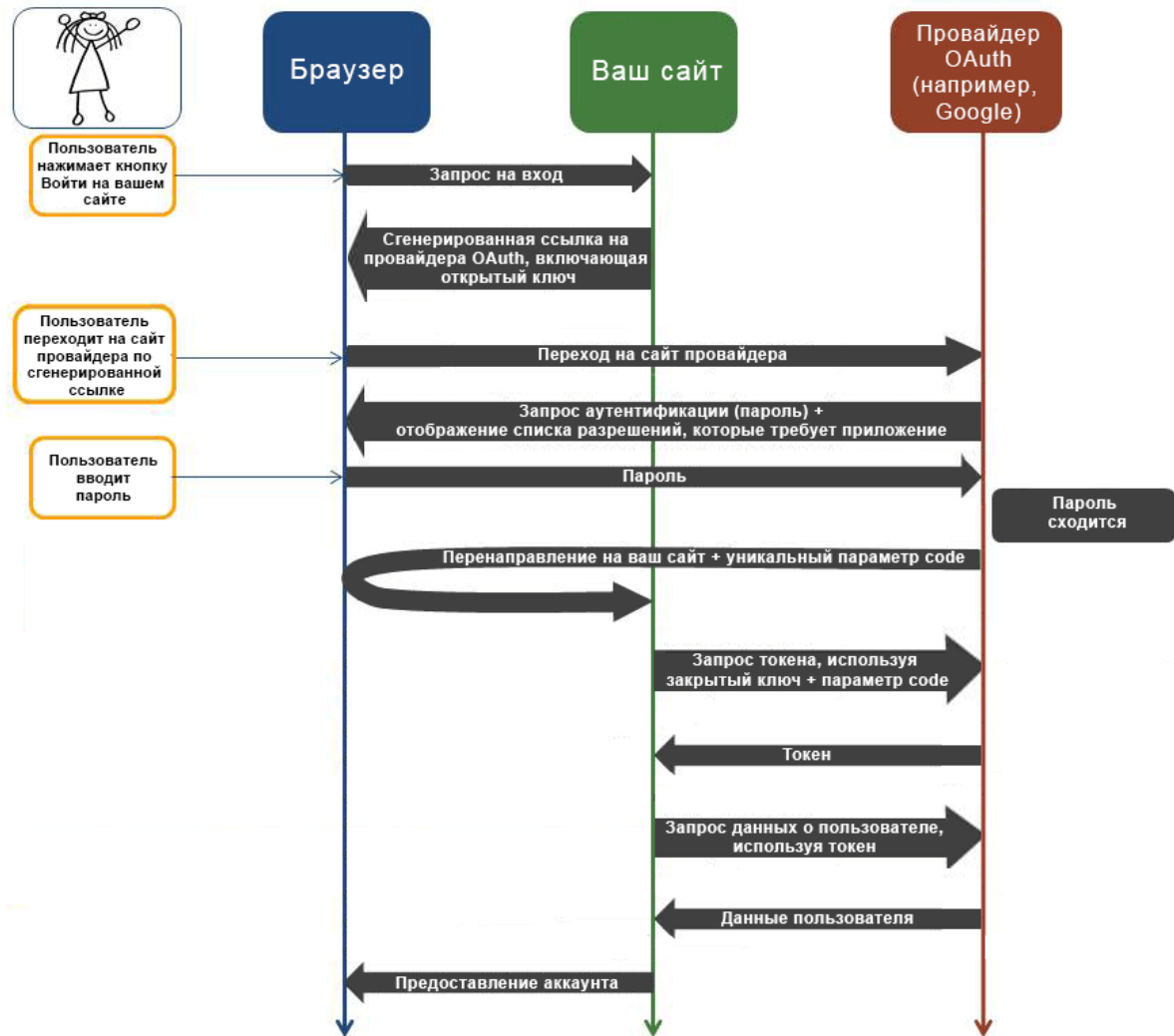


В двух словах, OpenID позволяет входить на множество сайтов используя один аккаунт провайдера, например профиль в социальной сети Facebook или Google+. Например, когда пользователь заходит на ваш сайт, как показано на схеме выше, ему предоставляется выбор провайдера, после чего он перенаправляется на сайт этого провайдера и вводит свои учетные данные. Если он успешно проходит аутентификацию у провайдера, провайдер возвращает вашему сайту успешный OpenID пользователя и вы уже можете его аутентифицировать в своем приложении.

### OAuth

Этот протокол используется для авторизации пользователей через другие интернет-ресурсы и, в отличие от OpenID, позволяет предоставить права на использование какого-то ресурса (например, загрузить аватарку пользователя, его email, поменять его стену Вконтакте или использовать другие API-интерфейсы, доступные у провайдера). На рисунке ниже показана схема работы OAuth:

## OAuth



После того, как пользователь прошел аутентификацию в другом сервисе, для него генерируется уникальный токен. Благодаря этому токenu вы можете загрузить данные пользователя из этого сервиса, либо выполнить другие манипуляции в аккаунте пользователя (список разрешений, которые требует ваше приложение, отображается при аутентификации).

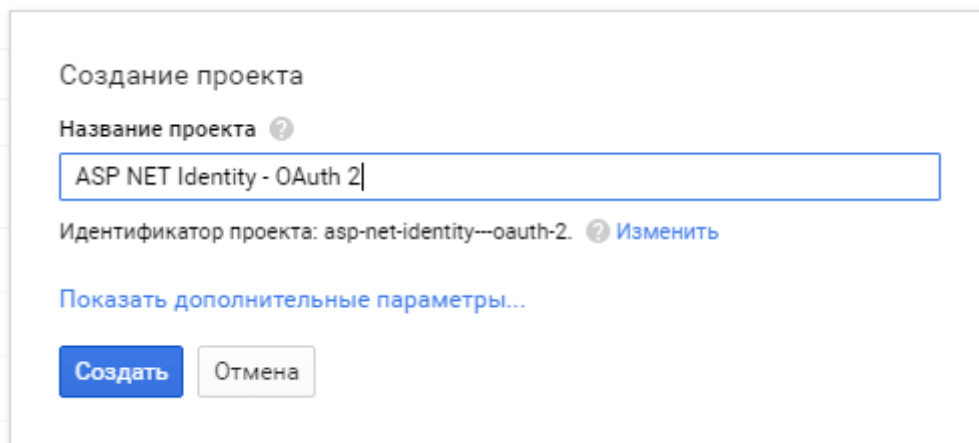
## Пример использования аутентификации Google

ASP.NET Identity имеет встроенную поддержку для работы с протоколами OpenID и OAuth для таких сервисов, как Google, Microsoft, Facebook и Twitter, а также содержит более обобщенную поддержку OAuth для интеграции с любыми другими сервисами. В этой статье в качестве примера мы рассмотрим аутентификацию через учетную запись в социальной сети Google Plus.

### Создание приложения в Google Developer

Для начала работы нам необходимо зайти в [консоль разработчика Google](#) и создать новый проект, который будет использоваться для авторизации пользователей:

Пройди тесты    C# тест (легкий)    .NET тест (средний)



Создание проекта

Название проекта ?

ASP NET Identity - OAuth 2

Идентификатор проекта: asp-net-identity---oauth-2. ? Изменить

[Показать дополнительные параметры...](#)

После создания нового проекта вас перенаправит на его начальную страницу, где нужно будет перейти в пункт меню «Учетные данные», после чего перейти во вкладку «Окно запроса доступа OAuth». В этой вкладке указывается ваш email-адрес, название приложения и его логотип (эта информация будет отображаться пользователю при аутентификации на стороне Google).

Окно запроса доступа - X

https://console.developers.google.com/apis/credentials/consent?project=asp-net-identity---oauth-2

Google APIs ASP NET Identity - OAuth 2

API Диспетчер API

Панель управления

Библиотека

Учетные данные

Учетные данные

Учетные данные Окно запроса доступа OAuth Подтверждение прав на домен

Адрес электронной почты ?

alexerohinzz@gmail.com

Название продукта, которое видят пользователи

Приложение авторизации через ASP.NET Identity

URL главной страницы (Необязательно)

https:// или http://

URL логотипа продукта (Необязательно) ?

http://professorweb.ru/downloads/logo-aspnetmvc.png

Так ваш логотип видят пользователи.  
Максимальный размер: 120x120 пикселей

URL политики конфиденциальности

Необходимо при развертывании приложения

https:// или http://

URL условий использования (Необязательно)

https:// или http://

Сохранить Отмена

Это окно будет отображаться каждый раз, когда зарегистрированные в этом проекте приложения будут запрашивать доступ к личным данным пользователя.

Для аутентификации OAuth необходимо указать адрес электронной почты и название продукта.

Нажмите кнопку «Сохранить». Теперь перейдите в левом меню в раздел «Библиотека» и найдите Google+ API, перейдите в меню этого API:

Библиотека API - ASP.NET Identity - OAuth 2

https://console.developers.google.com/apis/library?project=asp-net-identity---oauth-2&q=Google%2B%20API

Google APIs ASP NET Identity - OAuth 2

API Диспетчер API

Панель управления

Библиотека

Учетные данные

Библиотека

Google API

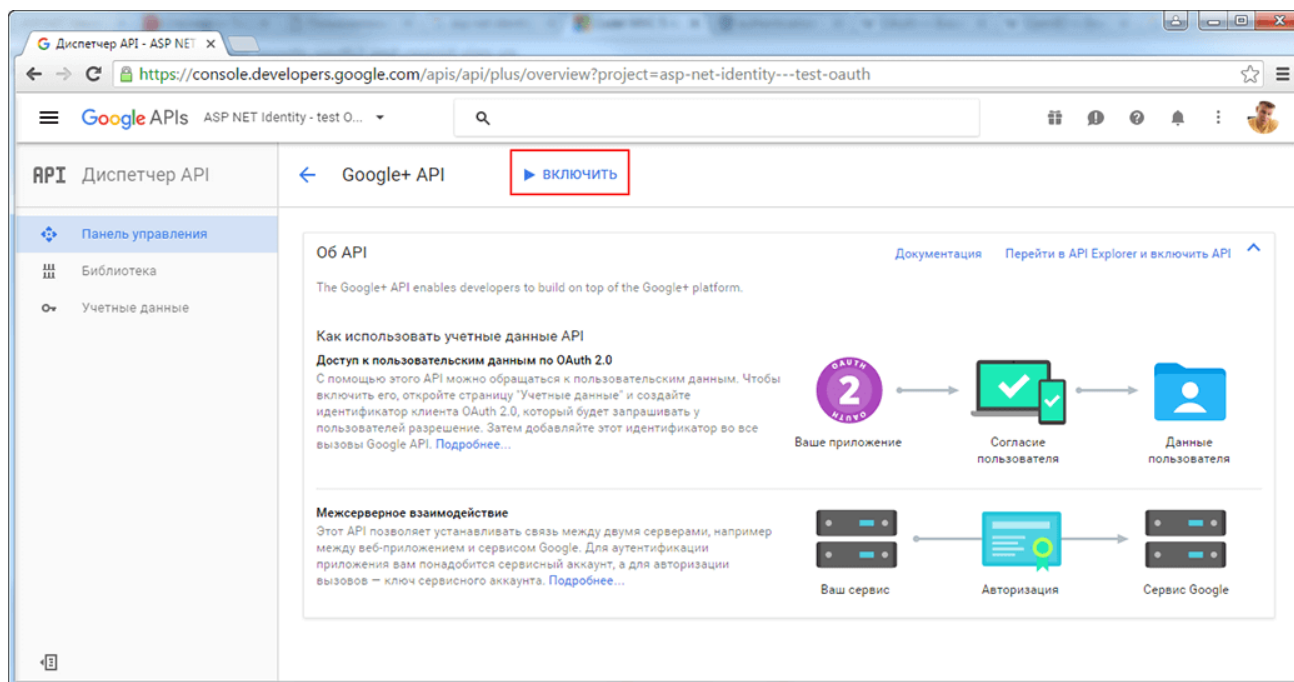
Google+ API

Вернуться к списку популярных API

Название	Описание
Google+ API	The Google+ API enables developers to build on top of the Google+ platform.
Google+ Domains API	The Google+ Domains API enables developers to build on top of the Google+ platform for Google Apps domains.
Google+ Hangouts API	The Google+ Hangouts API enables developers to build applications that run inside of Google+ Hangouts.
Google Places API Web Service	Find detailed information about places across a wide range of categories. Backed by the same database used by Google Maps and Google+, the Google Places API Web Service features about 100 million businesses and points of interest that are updated frequently through

Пройди тесты C# тест (легкий) .NET тест (средний)

В открывшемся окне необходимо щелкнуть по кнопке «Включить», как показано на рисунке ниже:



После этого отобразится следующее сообщение:

← Google+ API ■ ОТКЛЮЧИТЬ

⚠ Этот API включен, но чтобы использовать его, вам нужно создать учетные данные.

[Создать учетные данные](#)

Вам необходимо нажать на кнопку «Создать учетные данные», для настройки доступа OAuth. После этого откроется окно добавления учетных данных, состоящее из трех этапов. На первом этапе указывается тип API (в проекте может использоваться несколько API) и контекст его использования. В нашем примере мы используем Google+ API и вызываем его из веб-браузера (JavaScript):

## Учетные данные

### Добавление учетных данных

#### 1 Выбор типа учетных данных

Эта форма поможет определить, какой тип учетных данных необходим для вашего проекта.

If you wish you can skip this step and create an [API key](#), [client ID](#), or [service account](#)

##### Какой API вы используете?

От этого зависит тип учетных данных.

Google+ API

##### Откуда вы будете вызывать API?

От этого зависят точные настройки.

Веб-браузер (JavaScript)

##### К каким данным вы будете обращаться?

- ☒ Данные пользователя  
Данные пользователя Google (с запросом доступа)
- ☐ Данные приложения  
Данные ваших приложений

Выбрать тип учетных данных

#### 2 Создание учетных данных

Отмена

На втором этапе указываются ограничения по адресу сайта, из которого будет вызываться Google+ API:

## Добавление учетных данных

- ✓ Выбор типа учетных данных  
Вызовы к: Google+ API, источник: веб-браузер

### 2 Создание идентификатора клиента OAuth 2.0

#### Название

#### Ограничения

Задайте источники JavaScript и/или URI перенаправления.

##### Разрешенные источники JavaScript

URI источника клиентского приложения для запросов из браузера. В этом поле не допускаются подстановочные знаки (например, `http://*.example.com`) и пути (например, `http://example.com/subdir`). Если используется нестандартный порт, обязательно укажите его.



##### Разрешенные URI перенаправления

Используется для запросов с сервера. Здесь необходимо указать путь к разделу приложения, в который пользователи будут перенаправлены после аутентификации в Google. В конце к нему будет автоматически добавлен код авторизации для доступа. Не может содержать общедоступные IP-адреса, относительные пути и неполные URL.



Создать идентификатор клиента

### 3 Скачайте учетные данные

Отмена

Обратите внимание, что я указал ссылки на `localhost:4989`, в вашем приложении порт будет другим. Также, при разворачивании приложения, необходимо будет добавить еще URL-адрес реального веб-сайта. На третьем этапе отобразится сгенерированный ключ:

Пройди тесты

C# тест (легкий)

.NET тест (средний)



## Добавление учетных данных

- ✓ Выбор типа учетных данных  
Вызовы к: Google+ API, источник: веб-браузер

- ✓ Создание идентификатора клиента OAuth 2.0  
Клиент OAuth 'App Client' создан.

### 3 Скачайте учетные данные

Client ID	219070456750-ue5ci1k1k9htpqqpp70k1qteqfojllpq.apps.googleusercontent.com
-----------	--

Файл учетных данных всегда можно скачать на странице "Учетные данные". Он имеет формат JSON.

[Скачать](#)[Не сейчас](#)[Готово](#)[Отмена](#)

Этот ключ является открытым (ClientId). Когда вы нажмете на кнопку «Готово», браузер перенаправит вас на страницу идентификатора клиента в веб-приложении, где отображен также закрытый ключ (ClientSecret). Скопируйте куда-нибудь оба этих ключа, т.к. позже они нам понадобятся.

[Пройди тесты](#)[C# тест \(легкий\)](#)[.NET тест \(средний\)](#)

←
Скачать файл JSON
Сбросить секрет клиента
Удалить

### Идентификатор клиента для Веб-приложение

Идентификатор клиента	<b>ClientId</b> 219070456750-ue5ci1k1k9htpqqpp70k1qteqfojjlpq.apps.googleusercontent.com
Секрет клиента	<b>ClientSecret</b> XHbC3srTbNAXsCUhRQ_GDbMB
Дата создания	31 авг. 2016 г., 17:12:33

**Название**

App Client

**Ограничения**  
Задайте источники JavaScript и/или URI перенаправления.

**Разрешенные источники JavaScript**  
URI источника клиентского приложения для запросов из браузера. В этом поле не допускаются подстановочные знаки (например, http://\*.example.com) и пути (например, http://example.com/subdir). Если используется нестандартный порт, обязательно укажите его.

http://localhost:4989 ×

http://www.example.com

**Разрешенные URI перенаправления**  
Используется для запросов с сервера. Здесь необходимо указать путь к разделу приложения, в который пользователи будут перенаправлены после аутентификации в Google. В конце к нему будет автоматически добавлен код авторизации для доступа. Не может содержать общедоступные IP-адреса, относительные пути и неполные URL.

http://localhost:4989 ×

http://localhost:4989/GoogleLoginCallback ×

http://www.example.com/oauth2callback

Сохранить Отмена

## Настройки приложения

После того, как вы настроили проект OAuth на сайте Google Developer, можно приступить к интеграции сторонней системы аутентификации в своем приложении. Для начала необходимо установить пакет NuGet, добавляющий библиотеки для работы с аутентификацией Google через OAuth, используя Identity. Выполните следующую команду в окне Package Manager Console:

```
Install-Package Microsoft.Owin.Security.Google -version 3.0.1
```

Есть также еще несколько пакетов NuGet для работы с другими популярными социальными сетями. Они перечислены в табл

Пройди тесты

C# тест (легкий)

.NET тест (средний)

### Пакеты аутентификации NuGet

Пакет	Описание
<i>Microsoft.Owin.Security.Google</i>	Проверяет подлинность пользователей через учетную запись Google
<i>Microsoft.Owin.Security.Facebook</i>	Аутентификация пользователей через аккаунт Facebook
<i>Microsoft.Owin.Security.Twitter</i>	Аутентификация пользователей через аккаунт Twitter
<i>Microsoft.Owin.Security.MicrosoftAccount</i>	Аутентификация пользователей через аккаунт Microsoft
<i>Microsoft.Owin.Security.OAuth</i>	Аутентификация через любые сервисы, поддерживающие OAuth 2.0

После установки пакета, необходимо включить поддержку аутентификации через стороннее приложение в классе запуска OWIN. Для этого отредактируйте файл IdentityConfig.cs в папке App\_Start нашего приложения, как показано в примере ниже:

```
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.Google;
using Owin;
using System;
using Users.Infrastructure;

namespace Users
{
    Пройди тесты    C# тест (легкий)    .NET тест (средний)
    public class IdentityConfig
    {

```

```

public void Configuration(IAppBuilder app)
{
    app.CreatePerOwinContext<AppIdentityDbContext>(AppIdentityDbContext.Create);
    app.CreatePerOwinContext<AppUserManager>(AppUserManager.Create);

    app.CreatePerOwinContext<AppRoleManager>(AppRoleManager.Create);

    app.UseCookieAuthentication(new CookieAuthenticationOptions
    {
        AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
        LoginPath = new PathString("/Account/Login"),
    });

    app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);
    app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions
    {
        AuthenticationType = "Google",
        ClientId = "879075641925-ne4oeb5uoji2q5ofcpva5vfnlravp3e6.apps.googleusercontent.com",
        ClientSecret = "cQicsLYWtYJEE09EdQbxWBQk",
        Caption = "Авторизация через Google+",
        CallbackPath = new PathString("/GoogleLoginCallback"),
        AuthenticationMode = AuthenticationMode.Passive,
        SignInAsAuthenticationType = app.GetDefaultSignInAsAuthenticationType(),
        BackchannelTimeout = TimeSpan.FromSeconds(60),
        BackchannelHttpHandler = new System.Net.Http.WebRequestHandler(),
        BackchannelCertificateValidator = null,
        Provider = new GoogleOAuth2AuthenticationProvider()
    });
}
}
}
}

```

Каждый из пакетов, которые были перечислены в таблице выше, содержит расширяющий метод, который включает функциональность аутентификации через соответствующий сервис в OWIN. При аутентификации через Google используется метод `UseGoogleAuthentication`, которому передается объект `GoogleOAuth2AuthenticationOptions`. В свойствах этого объекта передаются параметры аутентификации. В частности в свойствах `ClientId` и `ClientSecret` указываются открытый и закрытый ключи приложения, которые мы сгенерировали ранее.

В свойстве `AuthenticationType` указывается уникальный идентификатор для экземпляра поставщика аутентификации. Этот <sup>Пройди тесты</sup> C# тест (легкий) и .NET тест (средний) при запросе свойства `Issuer` класса утверждения `Claim`, который мы подробно рассмотрели в предыдущей статье. Благодаря этому параметру можно использовать

несколько разных аутентификаций Google, например, используя разные проекты в Google Developer.

**Свойство *CallbackPath*** указывает ссылку, на которую будет возвращен пользователь после успешной аутентификации в Google. Важно, чтобы это свойство совпадало с URL перенаправления, который мы указали на втором этапе добавления учетных данных.

Теперь давайте добавим кнопку аутентификации через Google в представление Login.cshtml, находящееся в папке /Views/Account:

```
@model AuthUsers.Models.LoginViewModel
@{
    ViewBag.Title = "Авторизация на сайте";
}

<h2>Войти</h2>

@Html.ValidationSummary()
@using (Html.BeginForm()) {
    ...
}

<br />

@using (Html.BeginForm("GoogleLogin", "Account", FormMethod.Post, new { style = "width: 100%; border: 1px solid #ccc; padding: 10px; text-align: center;" }))
{
    <input type="hidden" name="returnUrl" value="@ViewBag returnUrl" />
    <button class="btn btn-primary" type="submit">
        Войти с помощью аккаунта Google
    </button>
}
```

Новая кнопка отправляет данные формы методу действия GoogleLogin контроллера Account. Ниже я показал, какие изменения необходимо внести в этот контроллер:

```
// ...

namespace AuthUsers.Controllers
{
    // Пройди тесты C# тест (легкий) .NET тест (средний)
    [Authorize]
    public class AccountController : Controller
    {
    }
}
```

1

// ...

[HttpPost]

[AllowAnonymous]

**public** ActionResult **GoogleLogin**(**string** returnUrl)

{

**var** properties = **new** AuthenticationProperties

{

RedirectUri = Url.Action("GoogleLoginCallback",

**new** { returnUrl = returnUrl })

};

HttpContext.GetOwinContext().Authentication.Challenge(properties, "Go

**return new** HttpUnauthorizedResult();

}

[AllowAnonymous]

**public async** Task<ActionResult> **GoogleLoginCallback**(**string** returnUrl)

{

    ExternalLoginInfo loginInfo = **await** AuthManager.GetExternalLoginInfoAsync(returnUrl);    AppUser user = **await** UserManager.FindAsync(loginInfo.Login);    **if** (user == **null**)

{

        user = **new** AppUser

{

Email = loginInfo.Email,

UserName = loginInfo.DefaultUserName,

City = Cities.LONDON,

Country = Countries.ENG

};

    IdentityResult result = **await** UserManager.CreateAsync(user);    **if** (!result.Succeeded)

{

**return** View("Error", result.Errors);

}

**else**

{

        result = **await** UserManager.AddLoginAsync(user.Id, loginInfo.LoginProviderName);        **if** (!result.Succeeded) **return** View("Error", result.Errors);

}

**return** View("Error", result.Errors);

```
        }
    }

    ClaimsIdentity ident = await UserManager.CreateIdentityAsync(user,
        DefaultAuthenticationTypes.ApplicationCookie);

    ident.AddClaims(loginInfo.ExternalIdentity.Claims);

    AuthManager.SignIn(new AuthenticationProperties
    {
        IsPersistent = false
    }, ident);

    return Redirect(returnUrl ?? "/");
}
}
```

Метод `GoogleLogin` создает экземпляр класса `AuthenticationProperties` и устанавливает *свойство* `RedirectUri` с адресом перенаправления, который вызывает метод действия `GoogleLoginCallback` в этом же контроллере. Следующая часть кода магическим образом вызывает ASP.NET Identity, чтобы перенаправлять пользователя на страницу аутентификации Google, а не ту, которая определена в приложении:

```
// ...

HttpContext.GetOwinContext().Authentication.Challenge(properties, "Google");
return new HttpUnauthorizedResult();

// ...
```

Это означает, что когда пользователь нажимает кнопку «Войти с помощью аккаунта Google», его браузер будет перенаправлен на страницу аутентификации Google, а затем обратно перенаправлен на страницу `GoogleLoginCallback` после того, как он пройдет проверку достоверности.

В методе действия `GoogleLoginCallback` мы получаем данные о пользователе от Google, используя *метод* `GetExternalLoginInfoAsync`, вызываемый на реализации *интерфейса* `IAuthenticationManager`:

Пройди тесты    C# тест (легкий)    .NET тест (средний)

```
// ...  
ExternalLoginInfo loginInfo = await AuthManager.GetExternalLoginInfoAsync();  
// ...
```

Класс **ExternalLoginInfo** содержит следующие свойства:

*Свойства класса ExternalLoginInfo*

Свойство	Описание
<i>DefaultUserName</i>	Возвращает имя пользователя
<i>Email</i>	Возвращает адрес электронной почты пользователя
<i>ExternalIdentity</i>	Возвращает объект ClaimsIdentity, связанный с идентификатором пользователя
<i>Login</i>	Возвращает объект UserLoginInfo, который описывает данные при внешнем входе в приложение

Мы используем метод FindAsync класса управления пользователями, для поиска на основе свойства ExternalLoginInfo.Login. Этот метод возвращает объект AppUser, если пользователь ранее уже прошел проверку достоверности в приложении:

```
// ...  
AppUser user = await UserManager.FindAsync(loginInfo.Login);  
// ...
```

Если метод FindAsync не возвращает объект AppUser — это означает, что пользователь впервые аутентифицируется в приложении, поэтому мы создаем новый объект AppUser с данными, полученными от Google и сохраняем нового пользователя в базе данных.

```
// ...  
Пройди тесты    C# тест (легкий)    .NET тест (средний)  
result = await UserManager.AddLoginAsync(user.Id, loginInfo.Login);  
// ...
```



Все что остается для аутентификации — добавить утверждения (claims), ассоциированные с данным пользователем и предоставляемые Google, а также создать аутентифицирующий файл cookie, позволяющий использовать учетные данные в приложении между сессиями:

```
// ...
ClaimsIdentity ident = await UserManager.CreateIdentityAsync(user,
    DefaultAuthenticationTypes.ApplicationCookie);
ident.AddClaims(loginInfo.ExternalIdentity.Claims);
// ...
```

## Тестирование Google-аутентификации

Перед тестированием аутентификации через Google необходимо внести еще одно изменение в приложение: нам нужно отключить пользовательскую проверку пароля, которую мы включили в статье [«Валидация данных»](#). Для этого прокомментируйте следующие строки в файле класса AppUserManager:

```
// ...

namespace Users.Infrastructure
{
    public class AppUserManager : UserManager<AppUser>
    {
        public AppUserManager(IUserStore<AppUser> store)
            : base(store)
        { }

        public static AppUserManager Create(IdentityFactoryOptions<AppUserManager> options,
            IOwinContext context)
        {
            AppIdentityDbContext db = context.Get<AppIdentityDbContext>();
            AppUserManager manager = new AppUserManager(new UserStore<AppUser>(db));

            /*manager.PasswordValidator = new CustomPasswordValidator
            {
                RequiredLength = 6,
                RequireNonLetterOrDigit = false,
                RequireDigit = false,
                RequireLowercase = true,
                RequireUppercase = true
            };*/

            manager.UserValidator = new CustomUserValidator();

            return manager;
        }
    }
}
```

Проверку данных пользователя необходимо оставить, т.к. по умолчанию, в ASP.NET Identity в логине можно указывать только буквы английского алфавита. Напомню, ранее мы добавили класс CustomUserValidator, где отключили эту проверку:

Пройди тесты    C# тест (легкий)    .NET тест (средний)

```
// ...

namespace Users.Infrastructure
{
    public class CustomUserValidator : IIdentityValidator<AppUser>
    {
        public async Task<IdentityResult> ValidateAsync(AppUser item)
        {
            List<string> errors = new List<string>();

            if (String.IsNullOrEmpty(item.UserName.Trim()))
                errors.Add("Вы указали пустое имя.");

            string userNamePattern = @"^[a-zA-Z0-9a-яА-Я]+$";

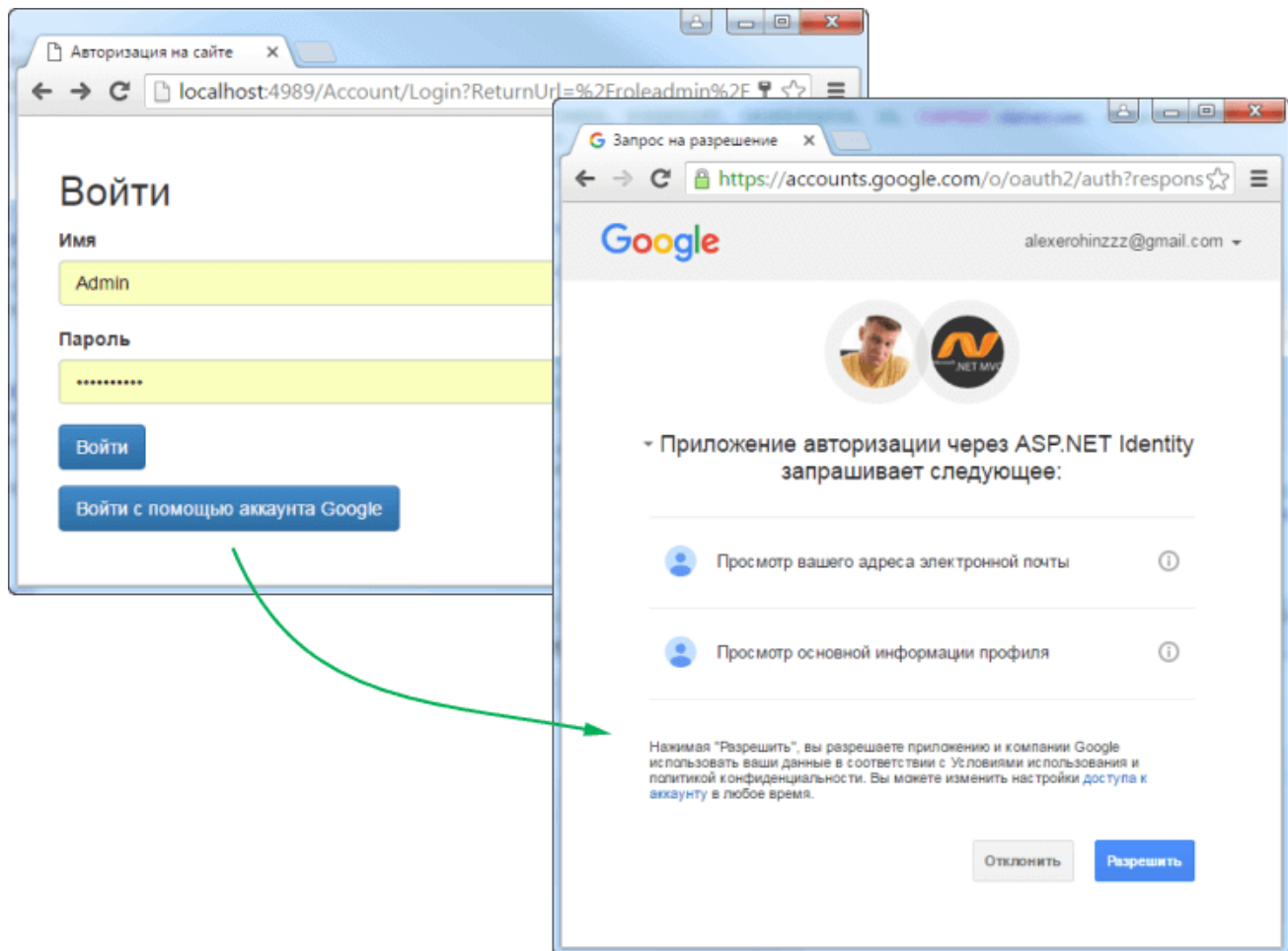
            if (!Regex.IsMatch(item.UserName, userNamePattern))
                errors.Add("В имени разрешается указывать буквы английского и");

            if (errors.Count > 0)
                return IdentityResult.Failed(errors.ToArray());

            return IdentityResult.Success;
        }
    }
}
```

Запустите приложение и нажмите кнопку «Войти с помощью аккаунта Google». Браузер откроет вам страницу Google, где необходимо ввести свои учетные данные:

[Пройди тесты](#)[C# тест \(легкий\)](#)[.NET тест \(средний\)](#)



Когда вы завершите процесс аутентификации, браузер перенаправит вас обратно в приложение. Если теперь открыть страницу с данными по утверждениям /Claims/Index, вы сможете увидеть, как утверждения из системы Google были добавлены к личности пользователя Identity: