SSO на базе OpenId Connect

В корпоративной системе. Организация и fine-tuning.

Дмитрий Федоров

DOTNEXT

2018 Piter

Вячеслав Михайлов
DataArt

Построение SSO на примере Identity Server 4.0 (.NET Core 2.0)

А что же будет?

- OIDC + MVC + SPA
- Single Sign Out
- Разное

Кому всё это нужно?

Ленивым пользователям

- **не нужно логиниться на каждом сайте отдельно**
- **меньше паролей – лучше один, но надёжный**

Хитрым разработчикам

- **проще инфраструктура**
- **меньше персональных данных**

Payroll

Inbox

Trip & Expanses

Time reporting

lelp desk

HR

Competences

Salary review

System administration

SSO – Это…

Когда удобно пользователю!

Идеальный случай: Windows identity

- **протоколы NTLM/Kerberos**
- **логин и пароль только при входе в систему**
- **но есть ложка дёгтя**

SSO – Это…

WIF (ex- project Geneva, представлен в ‘09 совместно с ADFS)

- **протокол WS-Federation, токен в формате SAML 1.0 (SOAP)**
- **работает в вебе, но можно подружить с Windows identity**
- **но есть ложка дёгтя**

Альтернативные протоколы

- **2005 Brad Fitzpatrick: Yadis**

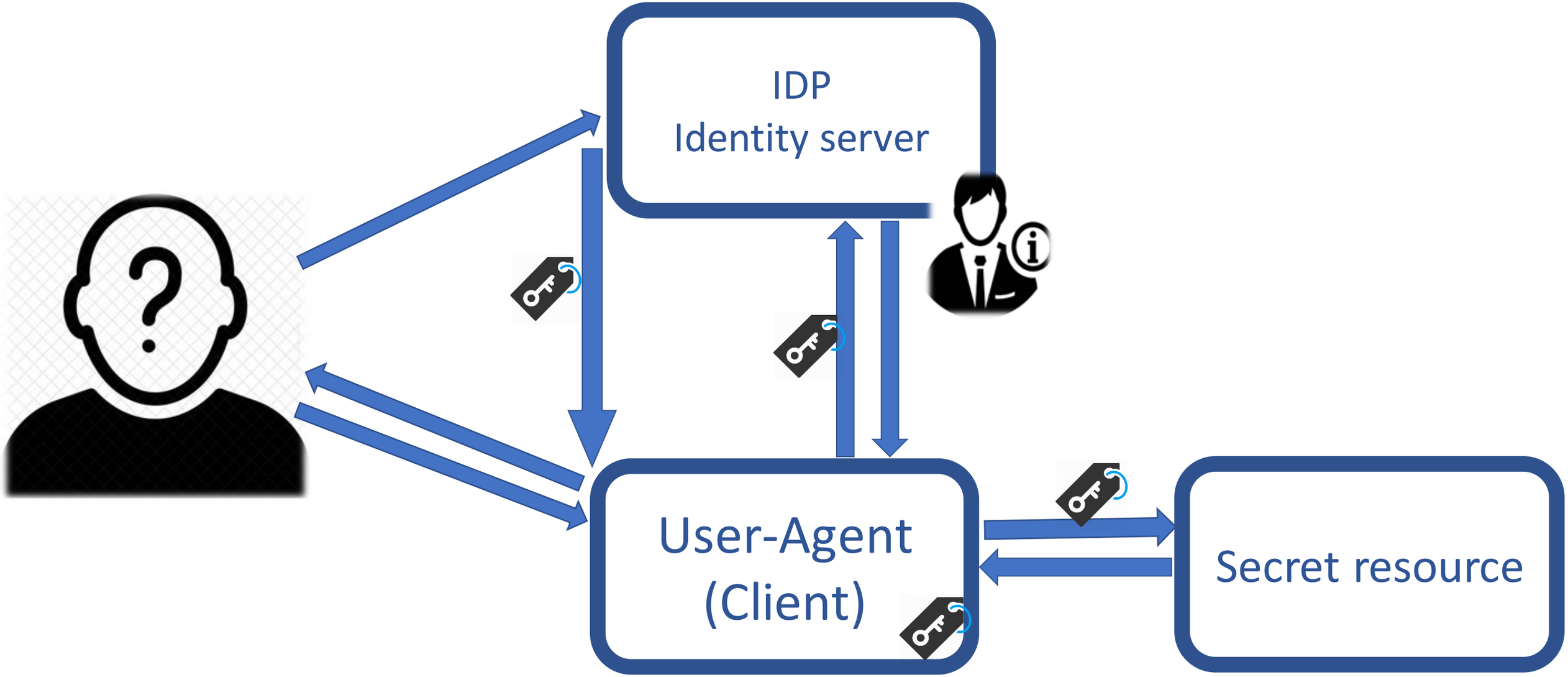
  Yet Another Distributed

  Identity System

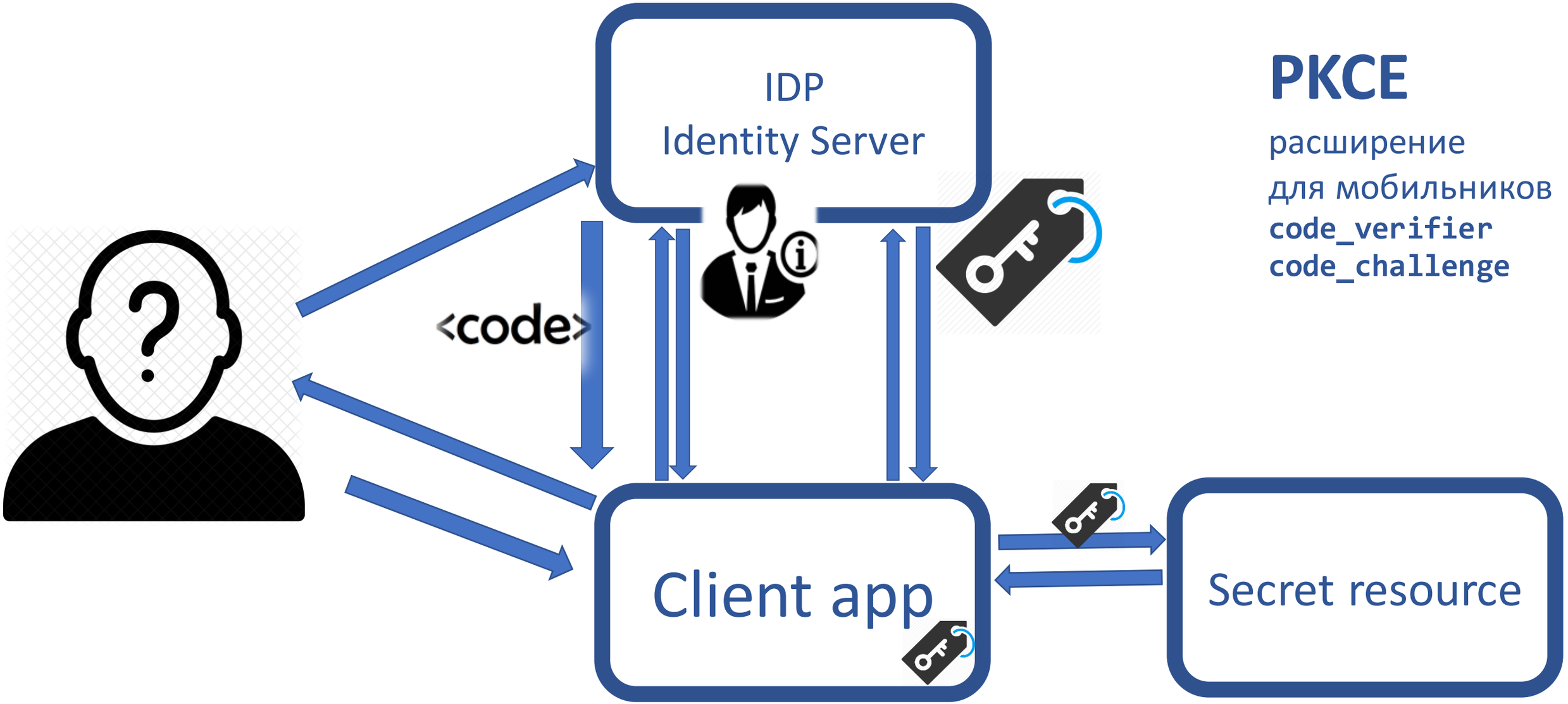
- **2006…2012 Blaine Cook, Chris Messina,**
  Twitter: Oauth


OpenID
Connect

Implicit Flow



Authorization Code Flow



PKCE

расширение
для мобильников
`code_verifier`
`code_challenge`

identity SERVER

Dominick Baier

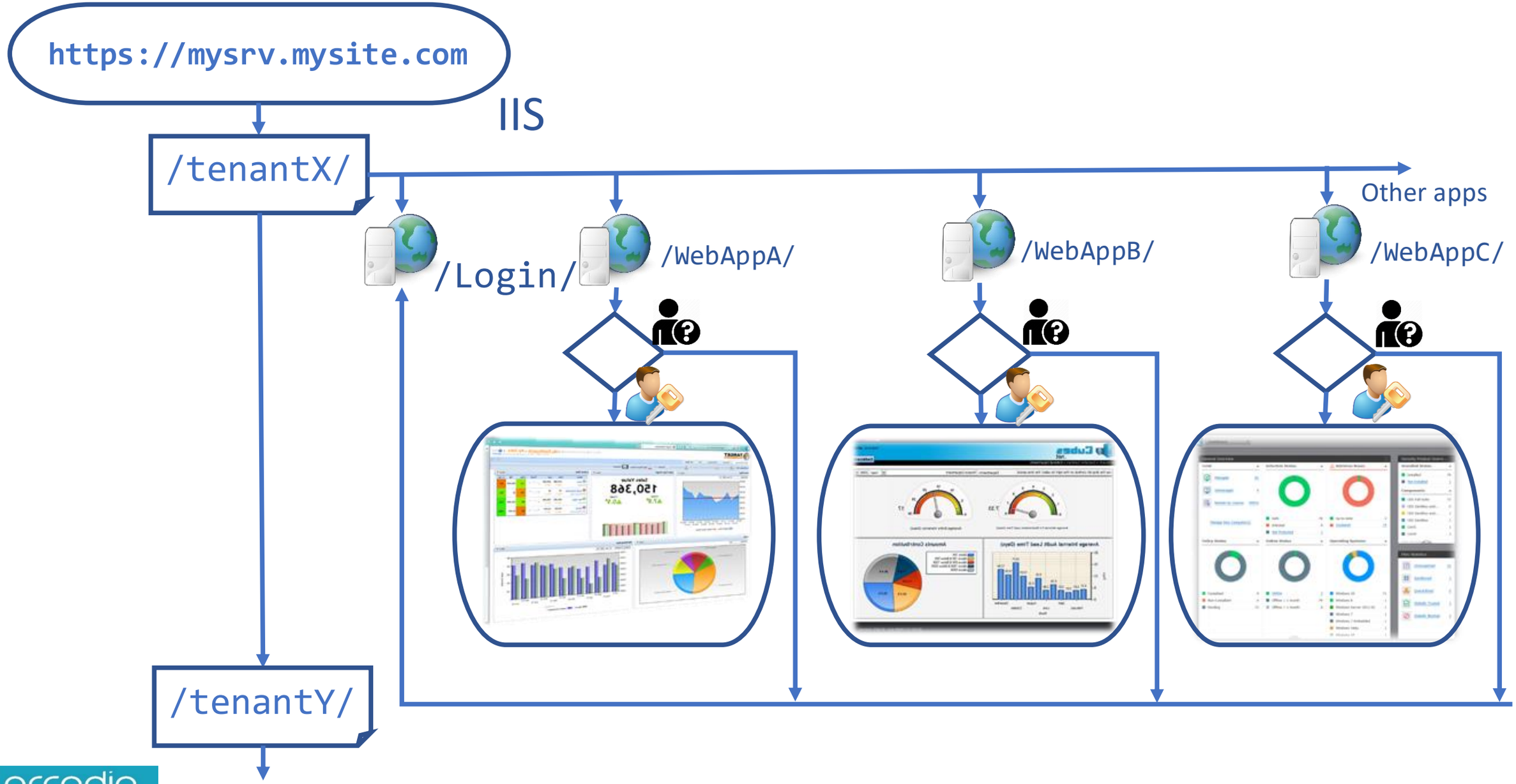
aka leastprivilege

Brock Allen

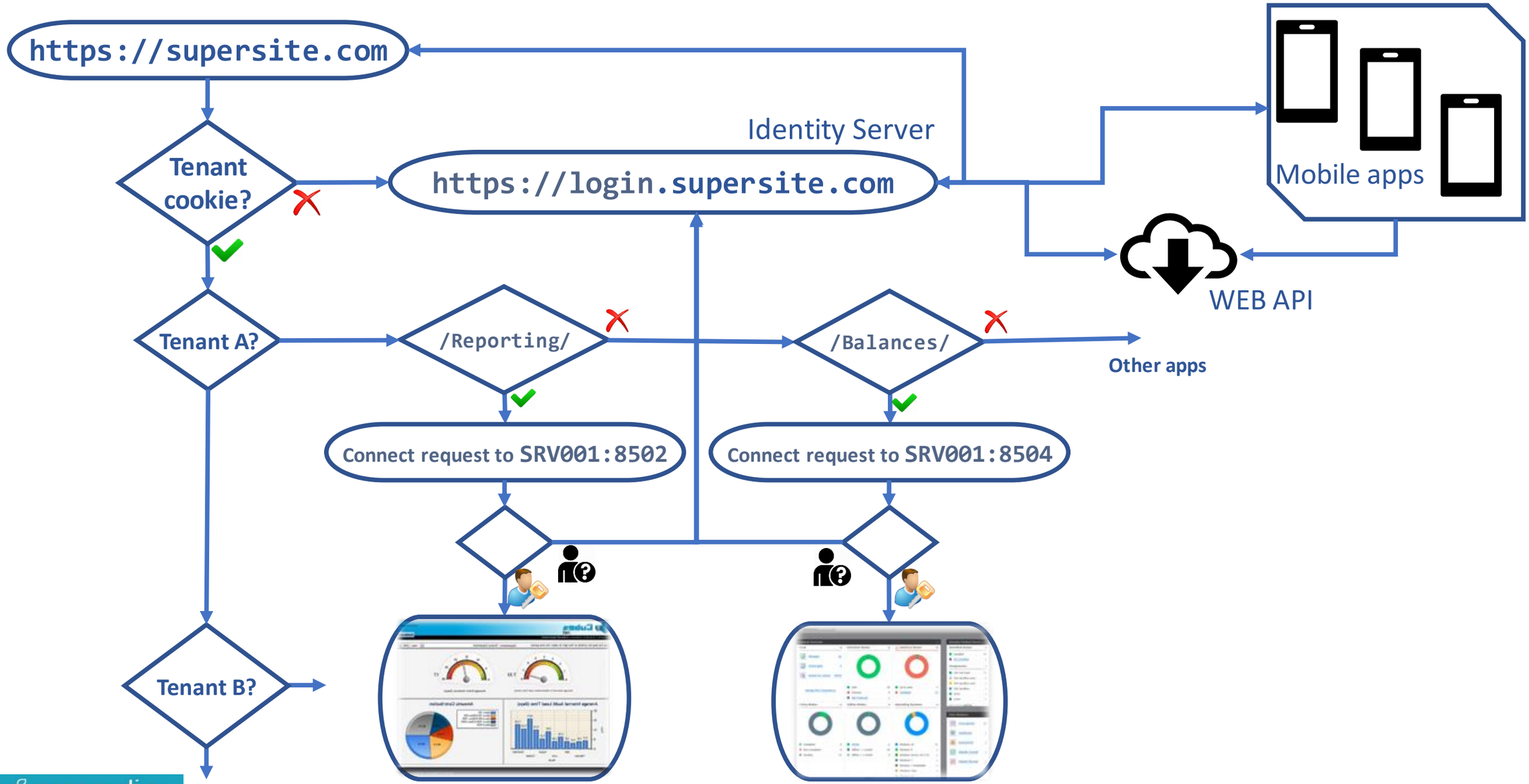
aka BrockAllen

http://identityserver.io/

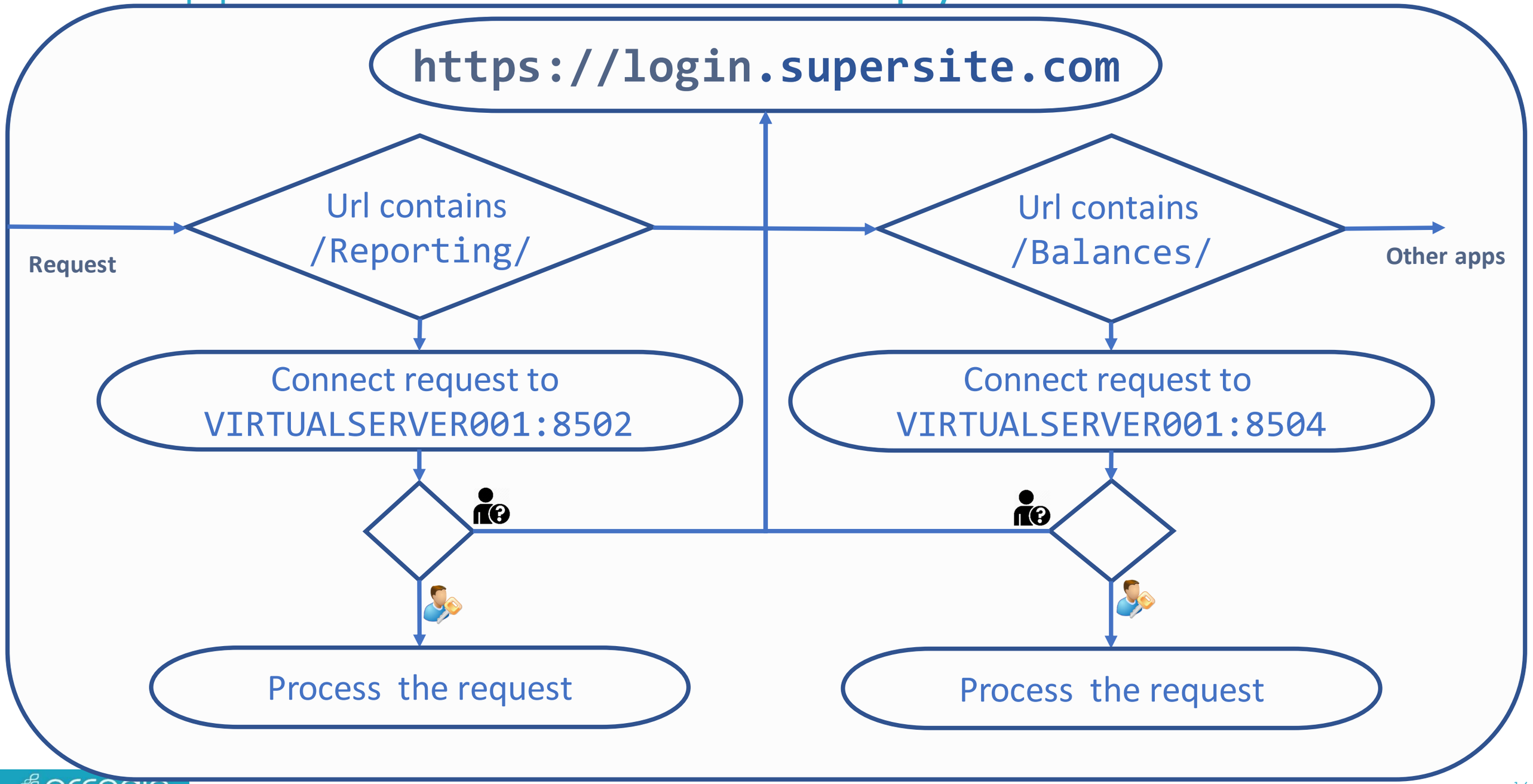
Пример из практики. Есть рабочее приложение…



А хотим: общий IdP + тенанты на докер-хостах.



Отдельно взятый тенант крупным планом.



Переключаемся с WIFa на OIDC

```
<system.identityModel>
      <identityConfiguration>
            <audienceUris>
                  <add value="http://dev.domain.com/ClientApp/" />
            </audienceUris>
            <issuerNameRegistry>
                  <trustedIssuers>
                  <!-- Thumbprint for the singing certificate -->
                        <add
                     thumbprint="FC6BB85A57488178C90161DE55A7948F00C756E6"
                     name="ssl-test" />
                  </trustedIssuers>
            </issuerNameRegistry>
      </identityConfiguration>
</system.identityModel>
```

В ASP.NET classic используем OIDC-клиента для OWIN

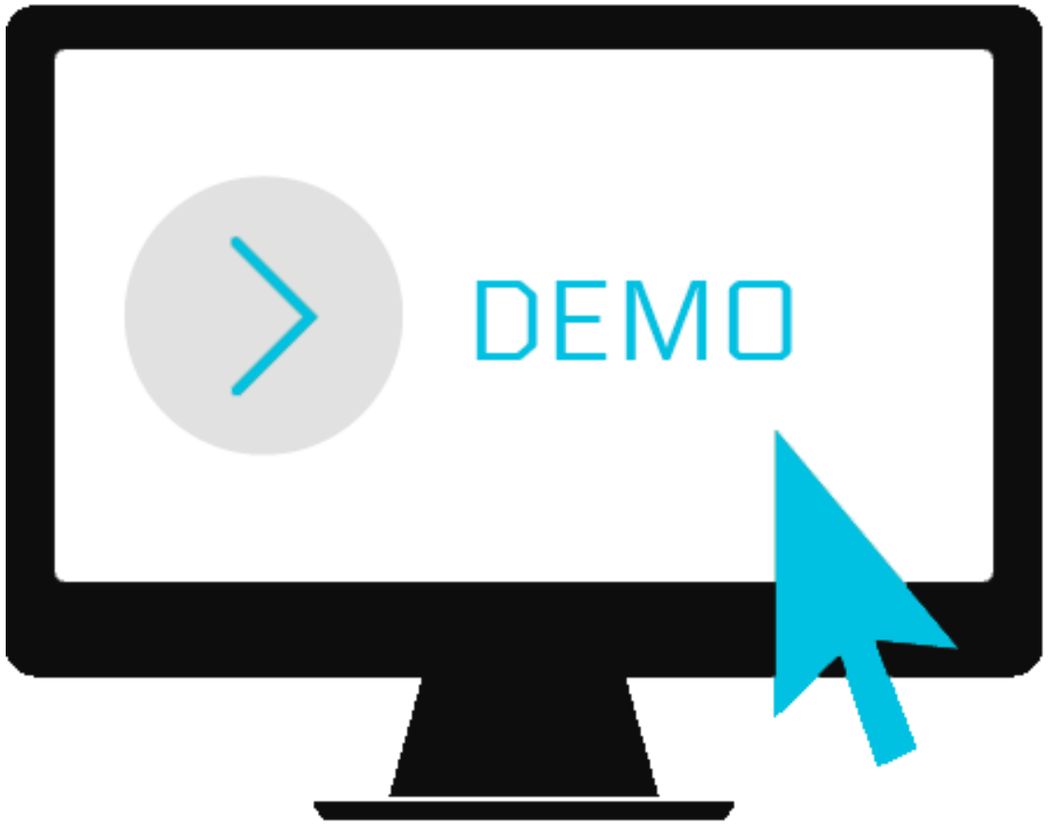
Пропишем в Application_Start:

```
RegisterGlobalFilters.(GlobalFiltersCollection filters){
    filters.Add(new HandleErrorAttribute());
    filters.Add(new AuthorizeAttribute());
}
```

А в конфиге у нас теперь:

```
<applicationSettings>
    <MyCompany.Owin.OidcHelper.OpenIdConnectAuthentication>
      <setting name="Authority" serializeAs="String">
        <value>http://localhost:5100</value>
      </setting>
      <setting name="RedirectUri" serializeAs="String">
        <value>http://localhost/IdSrvTestApp/</value>
      </setting>
      <setting name="ClientId" serializeAs="String">
        <value>localhost-testapp</value>
      </setting>
    </MyCompany.Owin.OidcHelper.OpenIdConnectAuthentication>
</applicationSettings>
```

Попробуем на практике



А если у нас даже не MVC, а ASP.NET WEB Forms?

```
public enum PipelineStage{
    Authenticate,
    PostAuthenticate,
    Authorize,
    PostAuthorize,
    ResolveCache,
    PostResolveCache,
    MapHandler,
    PostMapHandler,
    AcquireState,
    PostAcquireState,
    PreHandlerExecute,
 }
```

Troubleshooting: как избежать зацикливания

- `CookieManager = new SystemWebChunkingCookieManager()`
  - в `OpenIdConnectAuthenticationOptions`
  - и `CookieAuthenticationOptions`
- Следим, что return URI совпадает с исходным,

в части протокола и хоста, например:

```
app.Use(async (context, next) =>
{
        if (<какой-то сеттинг>)
                context.Request.Scheme = Uri.UriSchemeHttps;
        await next.Invoke();
});
```

Пара слов про SPA, в частности Angular

https://github.com/damienbod/angular-auth-oidc-client

- Разработан специально для Angular 2+
- Implicit, Code + PKCE flow
- Api call interceptor included

С вебом всё замечательно

Теперь не забыть защитить API

Например,

```
services.AddAuthorization(options =>
{
    options.AddPolicy("OfficeNumberUnder200", policy =>
                                        policy.Requirements.Add(
            new MaximumOfficeNumberRequirement(200)));
});

services.AddSingleton<IAuthorizationHandler,
                      MaximumOfficeNumberAuthorizationHandler>();
```

И далее:

```
[Authorize(Policy = "OfficeNumberUnder200")]
```

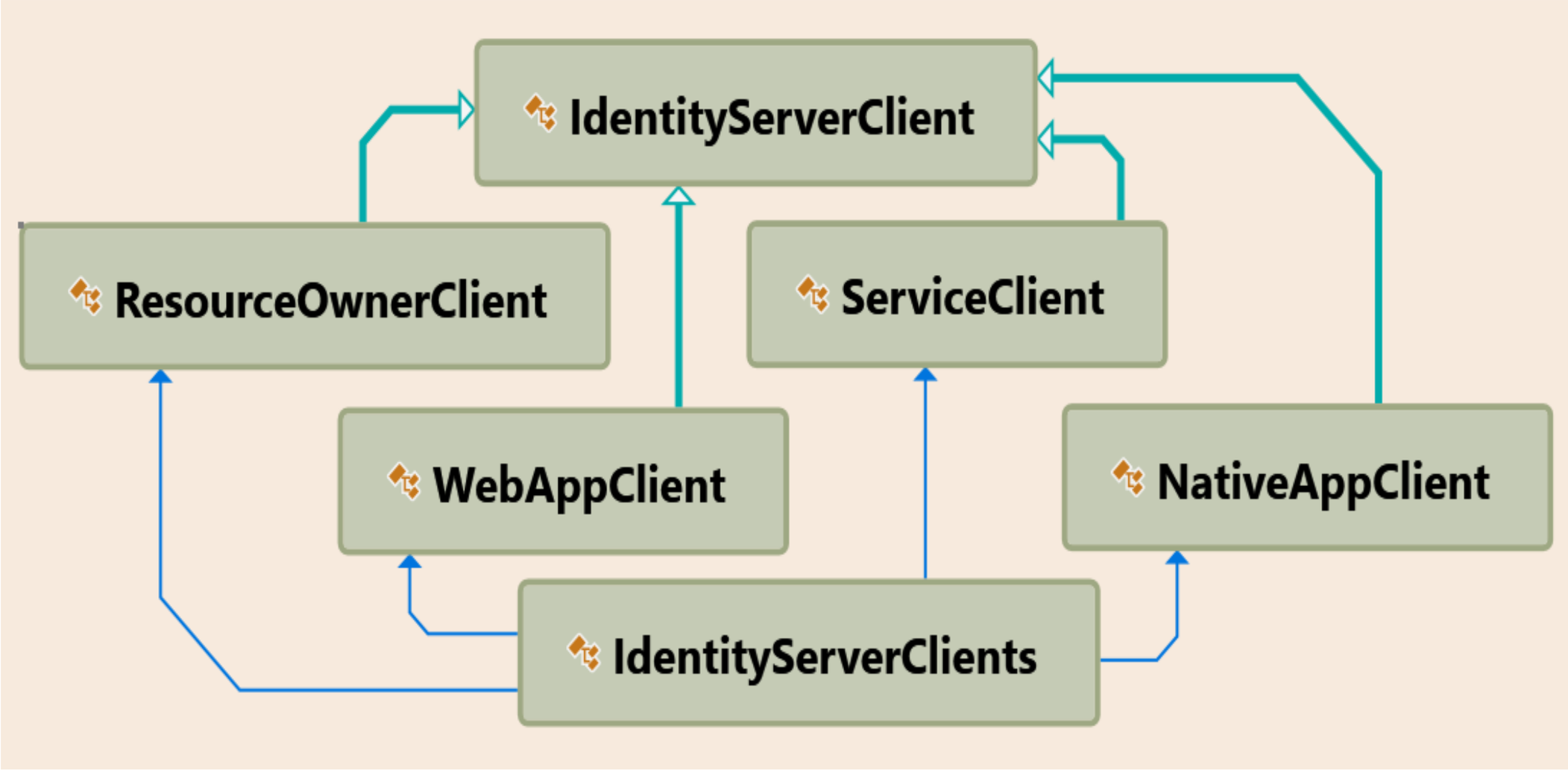
Cloud ready:

- OIDC через OWIN в MVC 5
- Или нативно в ASP.NET Core
- Или SPA на Angular 6
- `Bearer token` в WEB API

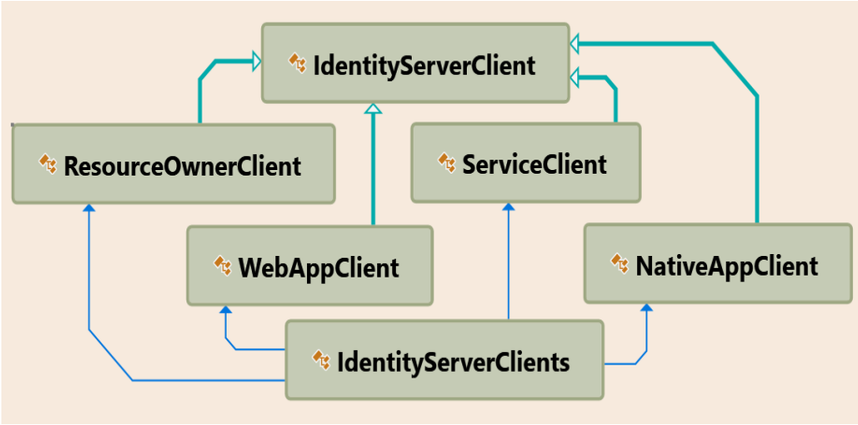
Identity Server Configuration

```
public void ConfigureServices(IServiceCollection services)
{
    Config.GetSection(nameof(IdentityServerClients)).Bind(myClients);
    services.AddSingleton(myClients);

    var cert = new X509Certificate2(“cert.pfx", "the-password");

    services.AddIdentityServer()
        .AddInMemoryClients(myClients.Clients)
        .AddInMemoryApiResources(GetApiResources())
        .AddSigningCredential(cert)
        .AddInMemoryIdentityResources(GetIdentityResources())
        .AddProfileService<UserProfileService>()
}
```

Если клиентов не слишком много,
сконфигурируем их в appsettings.json



```
"IdentityServerClients": {
   "WebAppClients": [
     {"ClientId": "super-app1",
       "RedirectUri": "https://app.example/super-app1/",
       "AllowedScopes": ["graph-api","graph-api.backend"] }],
   "NativeAppClients": [
     {"ClientId": "simple-native-app",
       "RedirectUri": "com.company.app:/oauth2callback",
       "AllowedScopes": [ "graph-api" ]      }    ],
   "ServiceClients": [
     { "ClientId": "xxx-cli-client",
       "ClientSecret": "xxx-secret-2018",
       "AllowedScopes": [
         "config-api",
         "config-api.admin"
       ]}]
 }
```



Кастомизируем Claims через Scope
Для аутентификации…

```
public static List<IdentityResource> GetIdentityResources()
{
    var openIdScope = new IdentityResources.OpenId();
    openIdScope.UserClaims.Add(JwtClaimTypes.Locale);

    return new List<IdentityResource>
    {
        openIdScope,
        new IdentityResources.Profile(),
        new IdentityResources.Email(),
        new IdentityResource(Constants.RolesScopeType, Constants.RolesScopeType,
            new List<string> {JwtClaimTypes.Role, Constants.TenantIdClaimType})
     };
}
```

Кастомизируем Claims через Scope
И для авторизации

```
public static IEnumerable<ApiResource> GetApiResources(){
        return new List<ApiResource>{
            new ApiResource{
                Name = "some-test-api",
                Scopes = {
                    new Scope{
                        Name = "some-api",
                        UserClaims = {
                            JwtClaimTypes.SessionId,
                            JwtClaimTypes.Role,
                            Constants.TenantIdClaimType,
                            JwtClaimTypes.Email,
                            JwtClaimTypes.Locale
        }};};};
}
```

Grant types

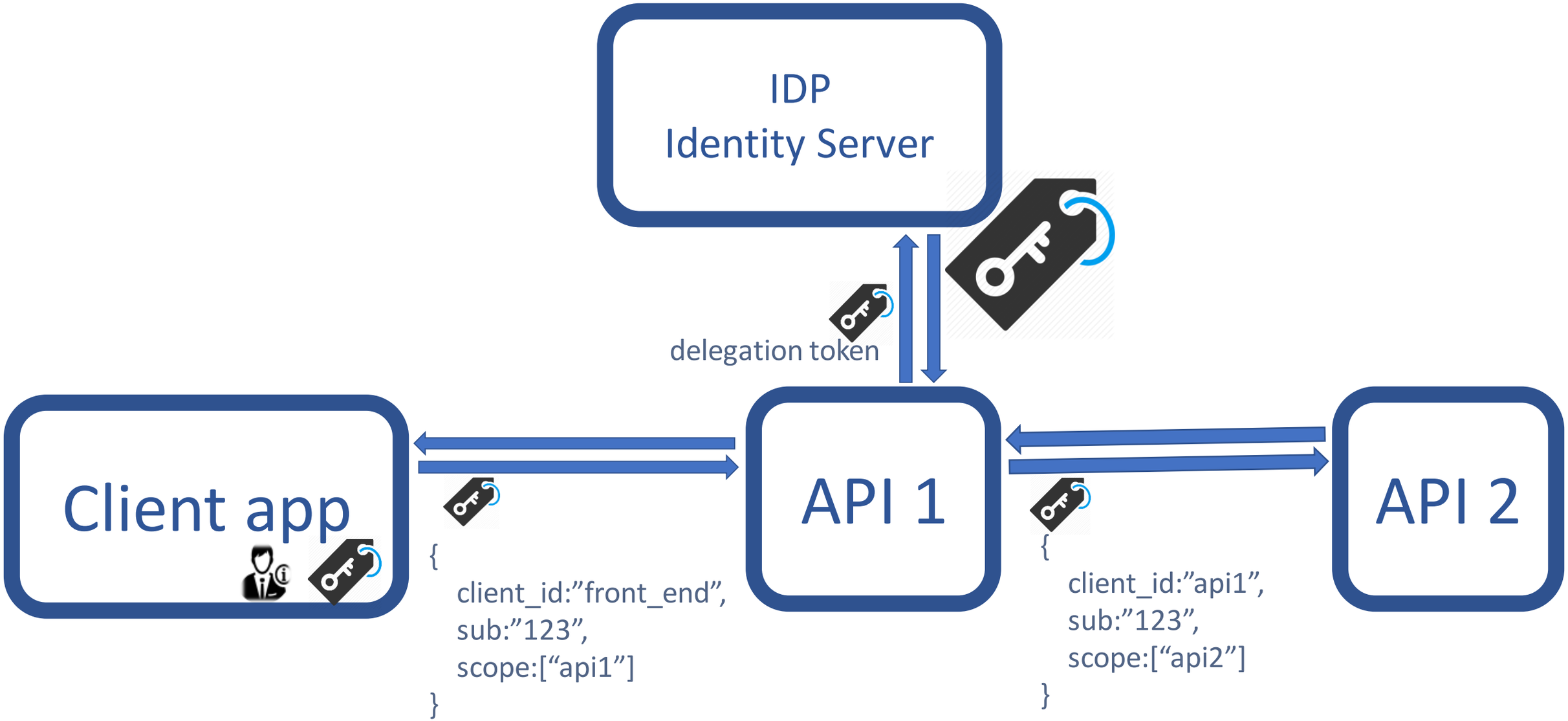
- Implicit
- Authorization code
- Hybrid
- Client credentials
- Resource owner password
- Refresh tokens
- Extension grants

Grant types

при регистрации клиента:

```
Client.AllowedGrantTypes =
    GrantTypes.HybridAndClientCredentials;

Client.AllowedGrantTypes =
{
    GrantType.Hybrid,
    GrantType.ClientCredentials,
    "my_custom_grant_type“
 };
```

Extension Grant для делегирования прав



Extension Grant для делегирования прав

```
public class DelegationGrantValidator:IExtensionGrantValidator
{
    public string GrantType => "delegation";
    public async Task ValidateAsync(ExtensionGrantValidationContext context)
    {
        var userToken = await ValidateAccessTokenAsync(
                                        context.Request.Raw.Get("token"));

        // get user's identity
        var sub = userToken.Claims.FirstOrDefault(c =>
                                c.Type == JwtClaimTypes.Subject)?.Value;

        context.Result = new GrantValidationResult(
                    sub, GrantType, userToken.Claims, Constants.MyIdIdpName);
    }
}
```

Чего нам не хватает – мультиинстантности задействуем REDIS

- **Сертификат для подписи токенов**

```
services.AddSigningCredential(cert);
```

- **Замена MachineKey, с версии 2.0 появилась поддержка Redis**

```
services.AddDataProtection()
    .SetApplicationName(typeof(Startup).Namespace)
    .PersistKeysToRedis(redis, "DataProtection-Keys");
```

- **PersistedGrantStore для хранения токенов, Consens**

```
services.AddSingleton<IPersistedGrantStore,
                      RedisPersistedGrantStore>().Configure(
    (Action<PersistedGrantStoreOptions>) (options =>
       options.DatabaseFactory = () => redis.GetDatabase(-1, null)));
```

Persistent token store, всего 5 методов

```
public interface IPersistedGrantStore {

    Task StoreAsync(PersistedGrant grant);

    Task<PersistedGrant> GetAsync(string key);

    Task<IEnumerable<PersistedGrant>> GetAllAsync(string subjectId);

    Task RemoveAsync(string key);

    Task RemoveAllAsync(string subjectId, string clientId);

}
```

Масштабирование:

- Общий сертификат
- Хранилище ключей
- Хранилище грантов

Single sign out

Просто sign out и Single sign out

В чём разница?

Из всех клиентов сразу!

Single sign out

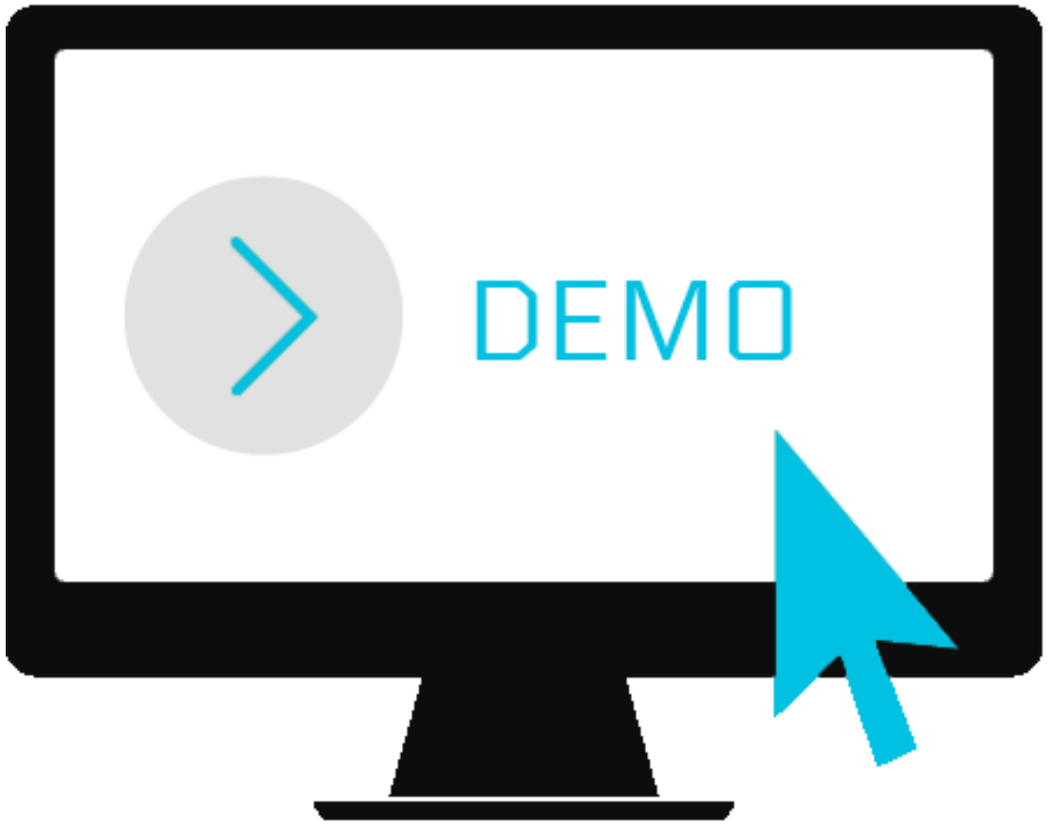
- **Front-Channel Logout 1.0 - draft 02**

  IFrame для каждого клиента (RP)

- **Back-Channel Logout 1.0 - draft 04**

  OP посылает JWT (Logout Token)

Попробуем на практике



Single sign out + External IdP? Легко!

- Стандартный

  `SignOutAsync(ExternalAuthenticationScheme)`

- Вызов внешнего LogoutEndpoint’a со странички LoggedOut


HERE
WE
ARE

Локализация

Задействуем RequestLocalizationMiddleware

```
app.UseRequestLocalization(
        new RequestLocalizationOptions{
            DefaultRequestCulture = new RequestCulture("en-GB"),
            SupportedCultures = Constants.SupportedCultures,
             SupportedUICultures = Constants.SupportedCultures
        });
```

Добавим культуру в токены

```
if (HttpContext?.Features.Get<IRequestCultureFeature>()?.Provider != null) {
    var requestedCulture = rcf.RequestCulture.Culture.TextInfo.CultureName;
    additionalClaims.Add(new Claim(JwtClaimTypes.Locale, requestedCulture));
} else // залогируем, что не нашли культуры в реквесте
```

Доступ к микросервисам из самомго IS-хоста.

```
private readonly IdentityServerTools idSrvTools;
var apiAccessToken = await
            idSrvTools.IssueClientJwtAsync(
                        "internal-identity-client",
                        12 * 60 * 60,
  new[] {Constants.ApiName,
            $"{Constants.ApiName}.admin"
            }, //scopes
  new[] {Constants.ConfigApiName} //audiences
);
apiClient.SetBearerToken(apiAccessToken);
```

Нужно вывесить API на IS-хосте?

Валидация токена без сетевого обмена

github.com/Kahbazi/IdentityServer4.Contrib.LocalAccessTokenValidation

```
services.AddAuthentication()
        .AddLocalAccessTokenValidation(
                                "token“
                                ,isAuth => { });
```

- SSO
- OIDC
- Группа бизнес приложений
- Identity Server 4
- Static Client config
- Grant Types
- Custom Grants
- Single Sign Out

Вопросы и ресурсы

- https://github.com/dfrunet/ – примеры из этого доклада
- https://openid.net/connect/
- https://github.com/IdentityServer
- https://leastprivilege.com/
- https://www.scottbrady91.com/Identity-Server/
- https://docs.microsoft.com/en-us/aspnet/aspnet/overview/owin-and-katana/owin-middleware-in-the-iis-integrated-pipeline
- https://blogs.msdn.microsoft.com/webdev/2017/04/06/jwt-validation-and-authorization-in-asp-net-core/

Contact

Dmitry Fedorov

dmitriy.fedorov@arcadia.spb.ru