Lithium Advanced Training Sessions

Objetivos

Explicar e documentar as várias funcionalidades da Framework Lithium e das tecnologias associadas.

Partilhar experiências do desenvolvimento dos micro serviços que estão em produtivo.

Promover boas práticas na implementação dos micro serviços.

Evitar a propagação de erros.

Promover a evolução da framework.

Formato

Sessões de 30 minutos de dois em dois dias sobre um ou mais temas do menu (em função do tempo).

A participação das equipas INT.CORE.I e INT.R&D é obrigatória. Para as restantes equipas é opcional.

O brainstorming sobre determinada sessão terá que ocorrer offline, entre reuniões.

Temas (uma lista em evolução)

- ✓ Arquitetura base de um micro serviço
- ✓ Funcionalidades principais do SDK
- ✓ Hydrogen
- ✓ Organização do source code de um micro serviço
- ✓ Código gerado e código custom
- ✓ Package references
- ✓ Funcionalidades out-of-the-box de um micro serviço
- ✓ Dependências (cache REDIS, table storage, blob storage, etc.)
- ✓ Hosting da aplicação (startup)
- ✓ App settings (configuração)
- ✓ Models
- ✓ Controllers (e Managers)
- ✓ Tratamento de erros na API
- ✓ Tratamento de erros nas livrarias cliente
- ✓ Console client e Postman
- ✓ Pipelines

Temas (uma lista em evolução)

- ✓ Autorização OAuth (e Identity Server)
- ✓ Autenticação OIDC (dos utilizadores)
- ✓ Background services (e workers)
- ✓ Custom Web UI
- ✓ Testes unitários
- ✓ Testes de integração
- × Code analysis
- ✓ Deployment de um micro serviço
- ✓ Monitorização de um micro serviço

#1

Princípios fundamentais, arquitetura base, funcionalidades do SDK, Service Designer

14/11/2019

Princípios fundamentais

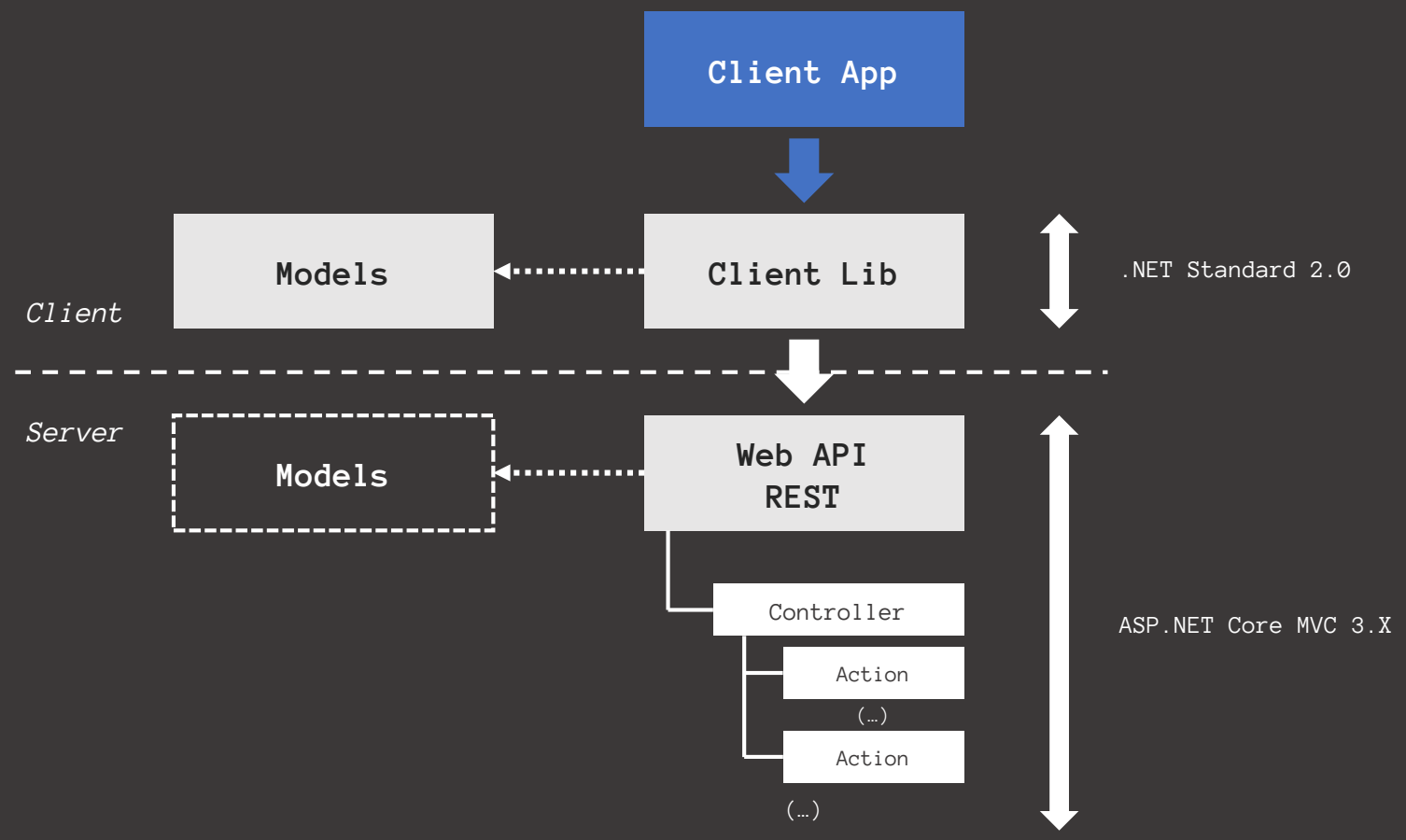
Keep it simple.

Implementamos aquilo que usamos, mais nada. Usamos aquilo que implementamos.

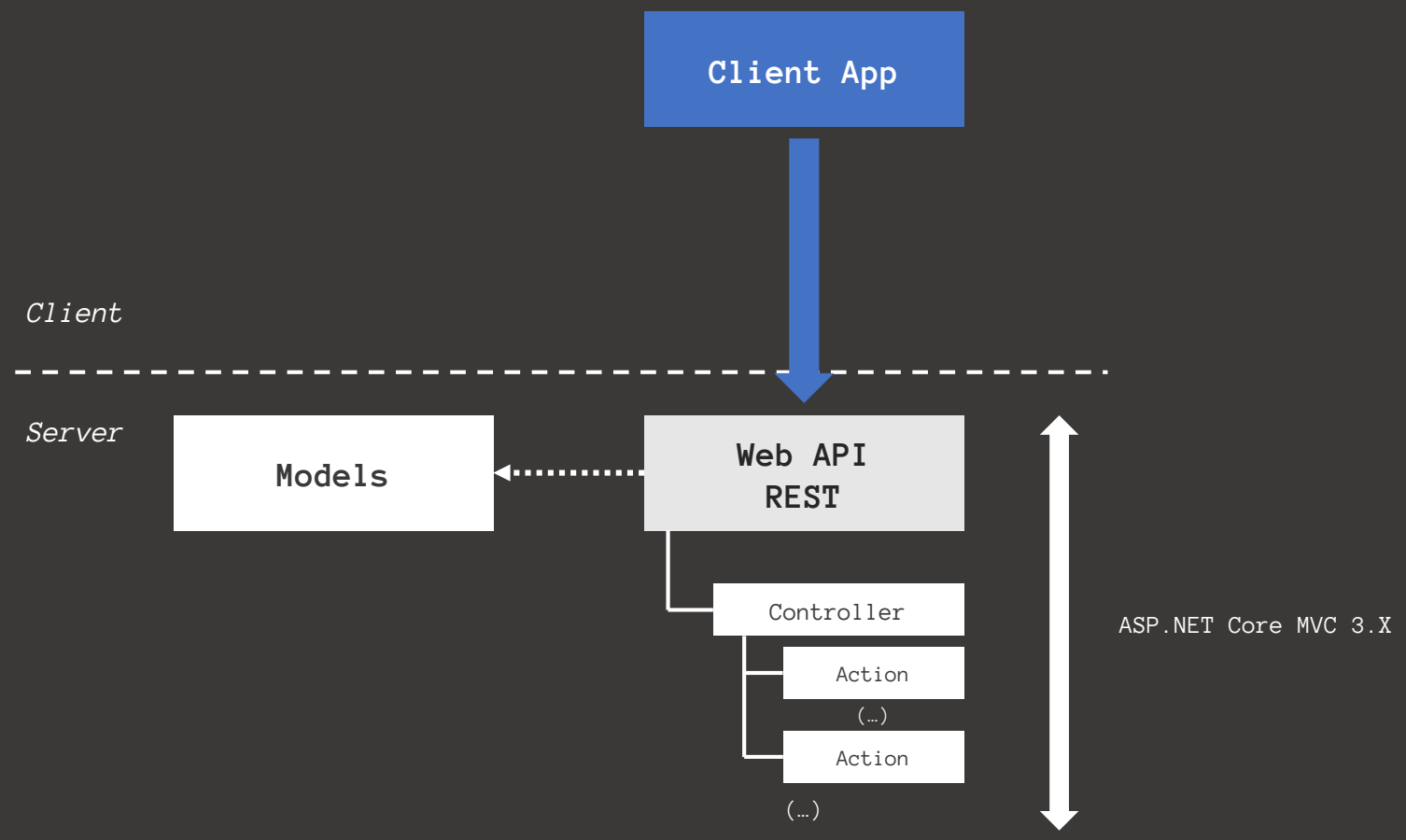
100% alinhado com o .NET Core.

Procuramos independência de serviços de terceiros (em particular, o Azure).

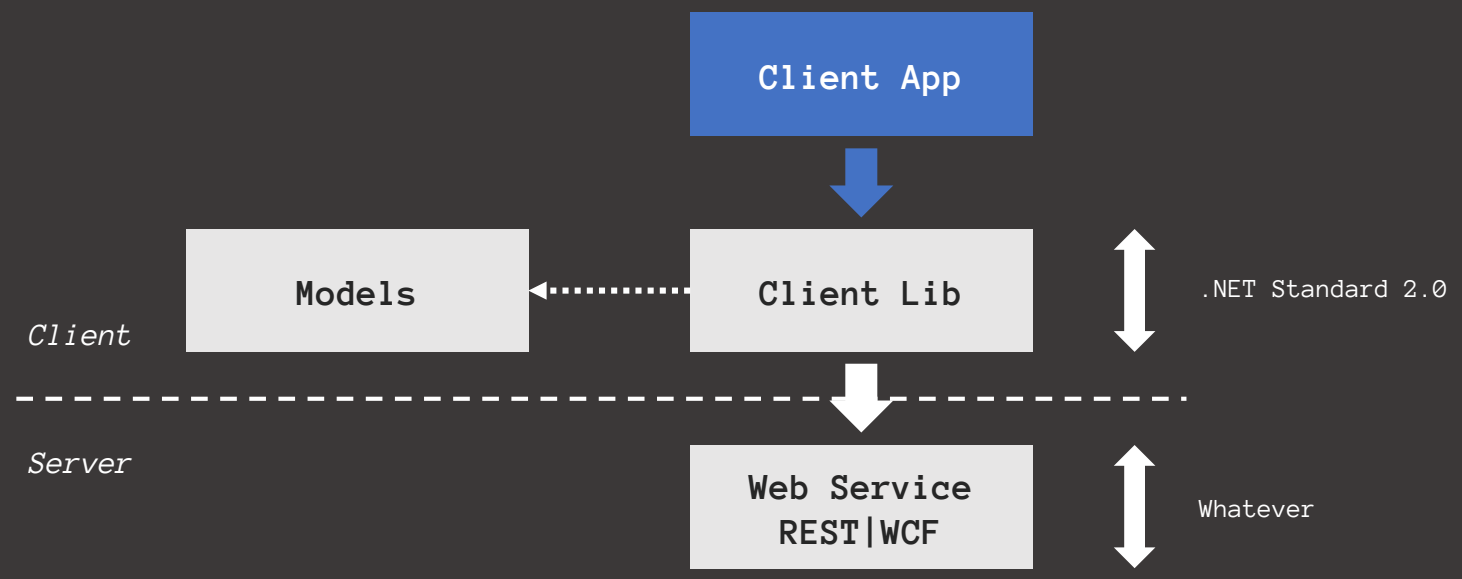
Arquitetura base de um micro serviço



WebAPI only (or MVC only or MVC + WebAPI)



ClientLib only (not really a microservice)



Funcionalidades do SDK

- Solution templates
- Service designer
- Geradores de código
- Upgrade de soluções
- Verificação de dependências (packages)
- https://github.com/PrimaveraDeveloper/lithium

Service Designer

Models e Enumerations

Controllers e Actions

Background Services

Dependencies

Herança

Validation rules

Authorization mode

Authorization scopes

Code comments

#2

Hydrogen, source code, código gerado, package references

18/11/2019

Hydrogen

https://github.com/PrimaveraDeveloper/lithium/tree/master/ref

v1.0 e v2.0

“Abstractions pattern”

Exemplo da Table Storage:

```
services.AddAzureTableStorage();
(...)
ITableStorageService service = provider.GetRequiredService<ITableStorageService();
ITableReference table = service.GetTable(“MyTable”);
IList<TableRecords> records = await table.RetrieveRecordsAsync(“key1”).ConfigureAwait(false);
IList<MyEntity> entities = await table.RetrieveEntitiesAsync(“key1”).ConfigureAwait(false);
(...)
public class MyEntity : TableEntity
(...)
```

Organização do source code

Team project Lithium

$Lithium\Core\Hydrogen.DesignTime

$Lithium\Core\Hydrogen

$Lithium\Microservices

Linhas de código (Development | Mainline)

Orinoco

Merge

Releases

Nomenclaturas (soluções e projetos)

Projetos de uma solução

Código gerado e código custom

Aspeto de uma solução de um micro serviço novo

GeneratedCode | CustomCode

Transformação dos templates

Dificuldades comuns

Lithium Settings

Double derived, override do código gerado

Qualidade do código custom

Package references

Primavera.Hydrogen.DesignTime.Configuration (assinatura, assembly info, code analyzers, etc.)

Primavera.Hydrogen.*

http://nuget.primaverabss.com:82/feeds/public-lithium-general

Pseudo metapackages

Dependências de terceiros

Validação no SDK

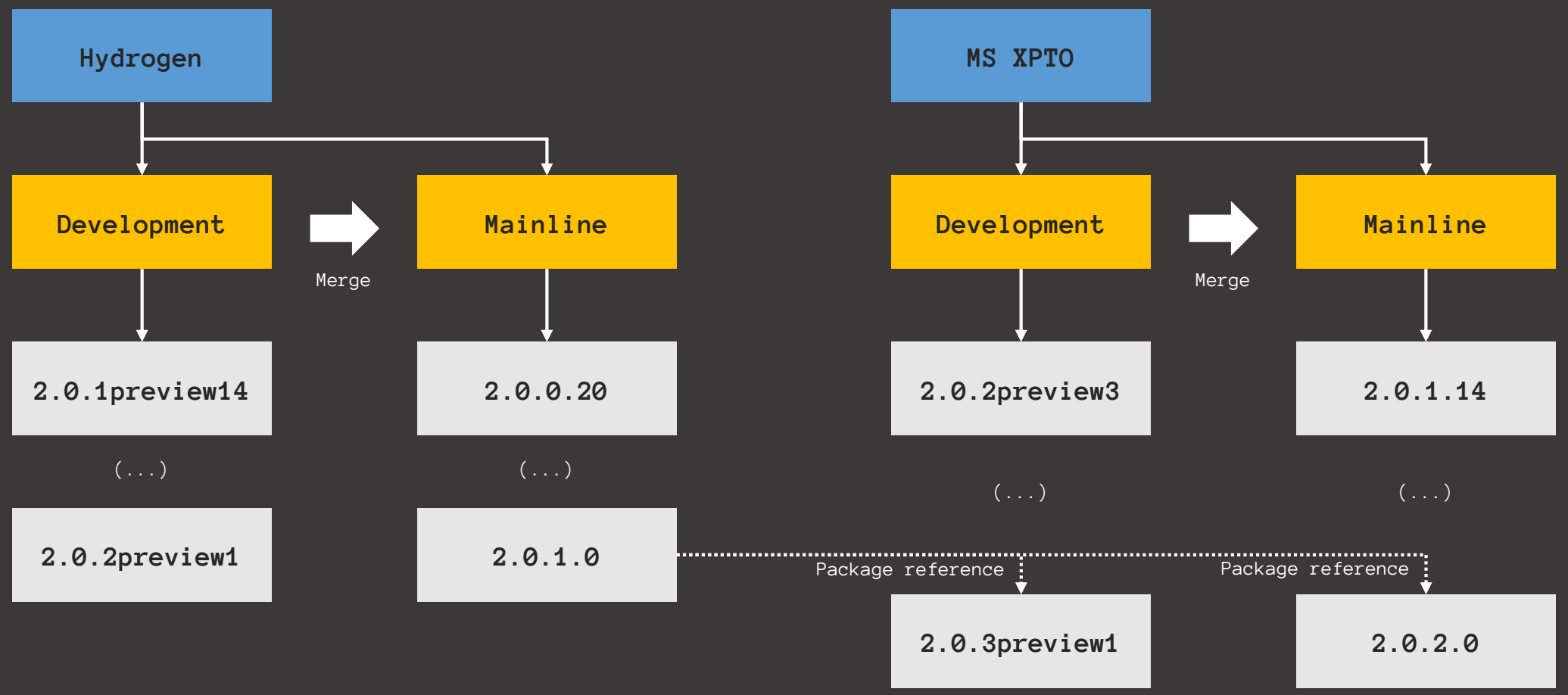
Update de packages nas soluções dos serviços

#3

Package references e builds (recap), funcionalidades out-of-the-box, startup, app settings

20/11/2019

Package references (recap)



Builds (recap)

As builds development produzem packages pre-release

As builds mainline produzem packages release

As builds que interessam são as da mainline

A linha development existe apenas para suportar a correção de bugs de serviços em produtivo (na mainline)

O único código que interessa é aquele que é utilizado por alguém ou alguma coisa

- Quando o binário produzido pela mainline vai ser utilizado
- Quando a release vai ser utilizada (para testes ou em produtivo)

Só se faz check-in de código terminado (ainda que seja uma versão intermédia)

Garantimos que a build development funciona

Garantimos que o check-in é merged para a mainline

Garantimos que a build mainline funciona

No processo são executados testes unitários e testes de integração

Qualquer erro imprevisto é detetado muito mais cedo

No final do processo, a regressão tenderá para ZERO!

Funcionalidades out-of-the-box

Hosting da aplicação Web (o host da Web API e do UI MVC)

Logging

Application Insights (trace, requests, exceptions, monitoring)

Configuration options (HostConfiguration, etc.)

Configuration secrets

MVC (API e UI)

Autenticação (dos utilizadores, OIDC)

Autorização (das client apps, OAuth) (dos utilizadores, OIDC)

Error handling (status code, exceptions, development mode)

HTTPS e HSTS

Request localization (accept-language)

Throttling

API versioning

Route analyzer

Home View

Next: Doc (API e ClientLib), WebHooks

Dependências (funcionalidades configuráveis)

Blob storage

Table storage

Isolated storage

Secrets storage

Search

Pipelines

Data protection

Startup da aplicação

Program

HostBuilderBase e HostBuilder

StartupBase e Startup

StartupBase.ConfigureServices()

StartupBase.Configure()

Overrides no startup (e.g. AddAdditionalServices())

App settings

Service.gen.lsspec.json e Service.lsspec.json (custom)

GeneratedCode/appsettings.gen.json

GeneratedCode/appsettings-{Environment}.gen.json

CustomCode/appsettings.json

CustomCode/appsettings-{Environment}.json

HostConfigurationBase e HostConfiguration

Customização da configuração do host

#4

Documentação (recap), models, controllers, managers, tratamento de erros no serviço

27/11/2019

Documentação (recap)

Home view: http://localhost:20000/

Client Library: http://localhost:20000/.doc/clientlib

Web API: http://localhost:20000/.doc/webapi

Diretoria: https://github.com/PrimaveraDeveloper/lithium/blob/master/dir/common/cs.md

Spec: https://github.com/PrimaveraDeveloper/lithium/blob/master/dir/common/specs/cs-spec-2.0.md

Models

Model = Resource (na teoria REST)

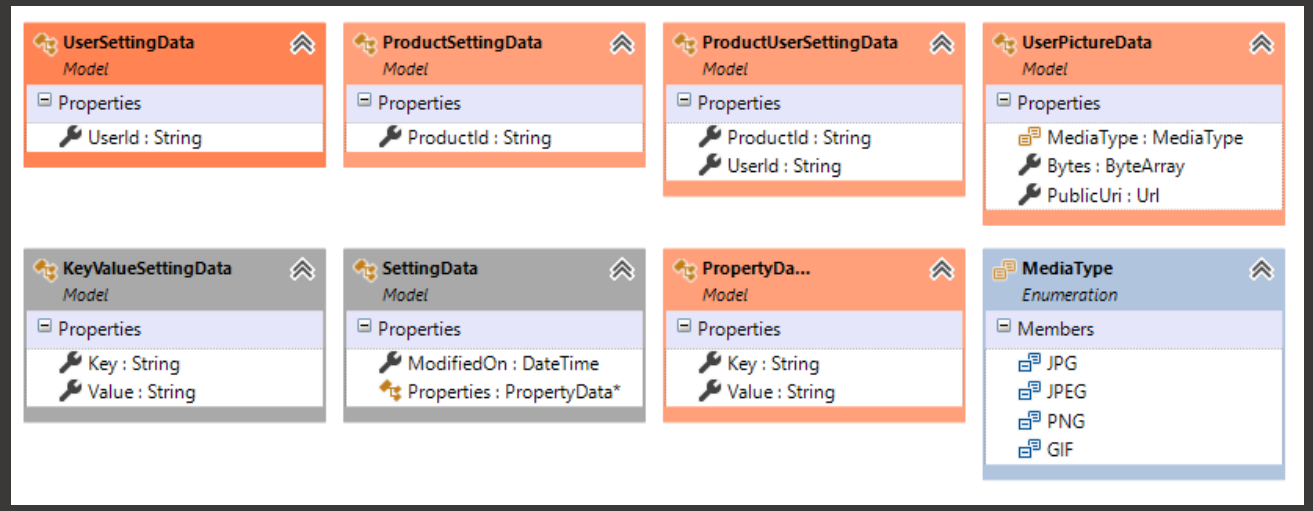
Tipos de dados:

Boolean, Guid, ByteArray, DateTime, Integer, Long, Double, String, Email, PhoneNumber, Url, Password, Enum, Model, List (*)

Regras de validação (conforme o tipo):

Required, GreaterThan, LessThan, MinLen, MaxLen, RegEx

Base models (herança)



Controllers (e managers)

No modelo...

- O Controller é um agregador lógico de operações (Actions) sobre o mesmo resource (Model) (não estrito)
- Define o modo de autorização e o scope de todas as operações
- Define a rota base de todas as operações (ex.: /api/v{version:apiVersion}/usersettings)

No código, o controller e as actions...

- São o ponto de entrada da Web API
- Devem ser o mais simples possível
- Devem definir a “shape” da API, não a lógica de negócio detalhada
- Daí resulta a necessidade de implementar essa lógica de negócio num “Manager”
- Esta separação Controller/Manager facilita/simplifica muito a geração do código, que se pretende que trate apenas do “esqueleto” da API (para dar liberdade/agilidade ao desenvolvimento)

O controller trabalha ao nível do layer HTTP/MVC => produz status code (devolve IActionResult)

O manager trabalha ao nível do layer da lógica de negócio => produz resultados (devolve OperationResult)

Exemplo: Manager e OperationResult<Model>

```
/// <summary>
/// Provides actions to send email notifications.
/// </summary>
/// <remarks>
/// This is the base class of the MVC controller.
/// </remarks>
[GeneratedCode("Lithium", "2.0")]
[Authorize(Constants.Policies.Notifications)]
[SuppressMessage("Maintainability Rules", "SA1402:FileMayOnlyContainASingleType", Justification = "Because of code generation design.")]
4 references | Hugo Ribeiro, 2 days ago | 2 authors, 3 changes
public abstract partial class EmailNotificationsControllerBase : ApiControllerBase, IEmailNotificationsController
{
    #region Code

    Protected Properties

    Protected Constructors

    #region Public Methods

    /// <inheritdoc />
    [HttpDelete(Primavera.Lithium.Notifications.Models.Metadata.Routes.EmailNotifications.CancelEmailNotification)]
    [ProducesResponseType(typeof(ServiceError), (int)HttpStatusCode.BadRequest)]
    [ProducesResponseType(typeof(void), (int)HttpStatusCode.NoContent)]
    1 reference | Hugo Ribeiro, 2 days ago | 2 authors, 2 changes
    public virtual Task<IActionResult> CancelEmailNotificationAsync([FromRoute] Guid id)
    {
        // Begin Validation

        if (!this.Validate()
            .Result(ErrorCodes.RequestArgsInvalid, ValidationResources.RES_Error_RequestArgsInvalid, out ServiceError validationError))
        {
            return Task.FromResult<IActionResult>(this.BadRequest(validationError));
        }

        // Result

        return this.CancelEmailNotificationCoreAsync(id);
    }
```

Exemplo: Manager e OperationResult<Model>

```
[SuppressMessage("StyleCop.CSharp.DocumentationRules", "SA1601: PartialElementsMustBeDocumented")]
6 references | Hugo Ribeiro, 15 days ago | 2 authors, 2 changes
public partial class EmailNotificationsController
{
    Private Properties

    #region Protected Methods

    #region CancelEmailNotification

    /// <inheritdoc />
    2 references | Hugo Lourenço | 1 author, 1 change
    protected override async Task<IActionResult> CancelEmailNotificationCoreAsync(Guid id)
    {
        // Invoke the manager

        OperationResult result = await this.NotificationsManager.CancelAsync(id, CancellationToken.None).ConfigureAwait(false);

        // Success?

        if (result.IsSuccess)
        {
            // Logging

            this.Logger.LogDebug($"Canceled email notification with id '{id}'.");

            // Result

            return this.NoContent();
        }

        // Not found?

        if (result.IsFailureWith(ErrorCodes.EmailNotificationNotFound))
        {
            ServiceError error1 = new ServiceError(ErrorCodes.EmailNotificationNotFound, Properties.Resources.RES_Error_EmailNotificationNotFound);
            return this.NotFound(error1);
        }

        // Failed

        ServiceError error2 = new ServiceError(ErrorCodes.EmailNotificationNotCanceled, Properties.Resources.RES_Error_EmailNotificationNotCanceled);
        error2.Details.Add(ServiceErrorDetail.FromOperationResult(result));

        return this.BadRequest(error2);
    }
```

Exemplo: Manager e Resource|Null

```
/// <summary>
/// Provides operations to manage user settings.
/// </summary>
/// <remarks>
/// This is the base class of the MVC controller.
/// </remarks>
[GeneratedCode("Lithium", "2.0")]
[Authorize(Constants.Policies.Settings)]
[SuppressMessage("Maintainability Rules", "SA1402:FileMayOnlyContainASingleType", Justification = "Because of code generation design.")]
14 references | Hugo Ribeiro, 1 day ago | 2 authors, 5 changes | 1 incoming change
public abstract partial class UserSettingsControllerBase : ApiControllerBase, IUserSettingsController
{
    #region Code

    Protected Properties

    Protected Constructors

    #region Public Methods

    /// <inheritdoc />
    [HttpGet(Primavera.Lithium.Settings.Models.Metadata.Routes.UserSettings.GetUserSetting)]
    [ProducesResponseType(typeof(Primavera.Lithium.Settings.Models.UserSettingData), (int)HttpStatusCode.OK)]
    [ProducesResponseType(typeof(ServiceError), (int)HttpStatusCode.BadRequest)]
    [ProducesResponseType(typeof(ServiceError), (int)HttpStatusCode.NotFound)]
    11 references | Hugo Ribeiro, 1 day ago | 2 authors, 3 changes | 1 incoming change
    public virtual Task<IActionResult> GetUserSettingAsync([FromRoute] string userId, [FromRoute] string key)
    {
        // Begin Validation

        if (!this.Validate()
            .Required(userId, ErrorCodes.GetUserSettingUserIdRequired, ValidationResources.RES_Error_GetUserSetting_UserId_Required)
            .MaxLength(userId, 200, ErrorCodes.GetUserSettingUserIdInvalid, ValidationResources.RES_Error_GetUserSetting_UserId_Invalid)
            .RegularExpression(userId, @"^(\d|[a-z]|[A-Z]|\.|-)*$", ErrorCodes.GetUserSettingUserIdInvalid, ValidationResources.RES_Error_GetUserSetting_UserId_Invalid)
            .Required(key, ErrorCodes.GetUserSettingKeyRequired, ValidationResources.RES_Error_GetUserSetting_Key_Required)
            .MaxLength(key, 200, ErrorCodes.GetUserSettingKeyInvalid, ValidationResources.RES_Error_GetUserSetting_Key_Invalid)
            .RegularExpression(key, @"^(\d|[a-z]|[A-Z]|\.|-|_)*$", ErrorCodes.GetUserSettingKeyInvalid, ValidationResources.RES_Error_GetUserSetting_Key_Invalid)
            .Result(ErrorCodes.RequestArgsInvalid, ValidationResources.RES_Error_RequestArgsInvalid, out ServiceError validationError))
        {
            return Task.FromResult<IActionResult>(this.BadRequest(validationError));
        }

        // Result

        return this.GetUserSettingCoreAsync(userId, key);
    }
```

Exemplo: Manager e Resource|Null

```
[SuppressMessage("StyleCop.CSharp.DocumentationRules", "SA1601:PartialElementsMustBeDocumented")]
20 references | Hugo Lourenço | 1 author, 1 change | 1 incoming change
public partial class UserSettingsController
{
    Private Properties

    #region Protected Methods

    /// <inheritdoc />
    2 references | Hugo Lourenço | 1 author, 1 change | 1 incoming change
    protected override async Task<IActionResult> GetUserSettingCoreAsync(string userId, string key)
    {
        // Get the user setting

        UserSettingData result = await this.Manager.GetSettingAsync(
            new string[]
            {
                userId,
                key
            })
            .ConfigureAwait(false);

        if (result == null)
        {
            return this.NotFound(
                new ServiceError(ErrorCodes.UserSettingNotFound, Properties.Resources.RES_Error_UserSettingNotFound));
        }

        // Result

        return this.Ok(result);
    }
```

Tratamento de erros (no serviço)

As Web API não devolvem erros ou exceptions, devolvem **respostas com um status code e um body (opcional)**

As aplicações cliente precisam de implementar lógica de tratamento de erros ao invocar serviços Lithium

Queremos padronizar o comportamento dos nossos serviços e das livrarias cliente (logo das aplicações cliente)

Mas as API Lithium também podem ser invocadas diretamente (ex.: HttpClient)

Os erros são de dois tipos: da **responsabilidade do cliente** ou da **responsabilidade do servidor**

No caso dos erros do cliente, queremos que sejam claros e incluam a informação necessária para que este possa corrigir o seu pedido (mas não mais que isso)

No caso dos erros do servidor, queremos informar o cliente sempre, mas sem lhe dar qualquer informação sobre a implementação interna ou mesmo a causa do erro e, importante, queremos recolher (no servidor) toda a informação necessária para os detetar (insights) e corrigir (insights + debug info) rapidamente

O developer confunde muitas vezes o seu role (como cliente ou como servidor)

Status Code HTTP

1xx – informacional

2xx – sucesso

3xx – redirecção

4xx – erro do cliente

5xx – erro do servidor

Status Code HTTP (2xx)

200 = OK

201 = Created

202 = Accepted

203 = Non-Authoritative Information

204 = No Content

205 = Reset Content

206 = Partial Content

207 = Multi-Status (WebDAV)

208 = Already Reported (WebDAV)

Status Code HTTP (5xx)

500 = Internal Server Error

501 = Not Implemented

502 = Bad Gateway

503 = Service Unavailable

504 = Gateway Timeout

505 = HTTP Version Not Supported

506 = Variant Also Negotiates (Experimental)

507 = Insufficient Storage (WebDAV)

508 = Loop Detected (WebDAV)

509 = Bandwidth Limit Exceeded (Apache)

510 = Not Extended

511 = Network Authentication Required

Status Code HTTP (4xx)

400 = Bad Request
401 = Unauthorized
402 = Payment Required
403 = Forbidden
404 = Not Found
405 = Method Not Allowed
406 = Not Acceptable
407 = Proxy Authentication Required
408 = Request Timeout
409 = Conflict
410 = Gone
411 = Length Required
412 = Precondition Failed
413 = Request Entity Too Large
414 = Request-URI Too Long
415 = Unsupported Media Type
416 = Requested Range Not Satisfiable
417 = Expectation Failed
418 = I'm a teapot (RFC 2324)
420 = Enhance Your Calm (Twitter)
422 = Unprocessable Entity (WebDAV)
423 = Locked (WebDAV)
424 = Failed Dependency (WebDAV)
425 = Reserved for WebDAV
426 = Upgrade Required
428 = Precondition Required
429 = Too Many Requests
431 = Request Header Fields Too Large
444 = No Response (Nginx)
449 = Retry With (Microsoft)
450 = Blocked by Windows Parental Controls (Microsoft)
451 = Unavailable For Legal Reasons
499 = Client Closed Request (Nginx)

Error Handling Middleware

Unhandled exceptions API => Azure Application Insights

Unhandled exceptions UI MVC => Status Code (500) Reexecute (error view)

Status code errors UI MVC => Status Code Reexecute (error view)

Development environment => Developer Exception Page

```
/// <summary>
/// Called by the runtime when the application starts to allow the application to configure itself.
/// </summary>
/// <param name="app">The current application.</param>
/// <param name="hostConfiguration">The current host configuration.</param>
/// <param name="logger">The logger.</param>
14 references | Hugo Ribeiro, 1 day ago | 2 authors, 3 changes | 1 incoming change
public virtual void Configure(IApplicationBuilder app, HostConfiguration hostConfiguration, ILogger<Startup> logger)
{
    // Validation

    SmartGuard.NotNull(() => app, app);
    SmartGuard.NotNull(() => hostConfiguration, hostConfiguration);
    SmartGuard.NotNull(() => logger, logger);

    // Logger

    this.Logger = logger;

    // Start

    this.Logger.LogDebug("Application configuration starting...");

    // Validate host configuration

    this.ValidateConfiguration(app, hostConfiguration);

    // Error handling

    this.UseErrorHandling(app, hostConfiguration);

    // HTTPS

    this.UseHttps(app, hostConfiguration);
```

Error Handling Middleware

```
/// <summary>
/// Called to activate error handling.
/// </summary>
/// <param name="app">The application builder.</param>
/// <param name="hostConfiguration">The current host configuration.</param>
/// <remarks>
/// The method is called from <see cref="Configure(IApplicationBuilder, HostConfiguration, ILogger{Startup})"/>.
/// </remarks>
1 reference | Hugo Ribeiro, 17 days ago | 2 authors, 2 changes | 1 incoming change
protected virtual void UseErrorHandling(IApplicationBuilder app, HostConfiguration hostConfiguration)
{
    // Add exceptions middleware

    if (this.UseDevelopmentSettings || this.CurrentEnvironment.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseWhen(
            context =>
            {
                return context.Request.PathIsNotApi() && context.Request.PathIsNotContent();
            },
            builder =>
            {
                builder.UseExceptionHandler(Constants.Controllers.Home.Routes.Error);
            });
    }

    // Add status code errors middleware

    // NOTE:
    // This middleware handles simple status code responses (e.g. 404)
    // Exceptions (status code 500) are handled by the exception handler above

    app.UseWhen(
        (context) =>
        {
            return context.Request.PathIsNotApi() && context.Request.PathIsNotContent();
        },
        builder =>
        {
            builder.UseStatusCodePagesWithReExecute(Constants.Controllers.Home.Routes.Error, "?statusCode={0}");
        });
}
```

Resumo

O manager:

- Devolve OperationResult ou OperationResult<T>
- Success ou Failure + OperationError
- OperationError tem um ErrorCode
- Trata as exceções expetáveis
- Todas as outras não são tratadas

O controller:

- Devolve IActionResult (não StatusCode())
- Trata a OperationResult devolvida pelo manager
- Success => Success Status Code (OK|Accepted|Created|NoContent)
- Failure => Error Status Code (BadRequest|NotFound|Conflict)
- O status code de erro deve depender do error code da OperationResult

#5

Tratamento de erros nas livrarias cliente, console client, postman e pipelines

29/11/2019

Client Library

Class Library .NET Standard 2.0 C#

Compatibilidade com .NET Framework (ERP), .NET Core (MS) e .NET Standard (Jasmin, Rose)

OAuth2 Client Credentials

Funcionalidades out-of-the-box:

- Autorização
- Versionamento da API
- Retry
- Localização (Accept-Language)

Tratamento de erros (nas livrarias cliente)

ServiceOperationResult e ServiceOperationResult<T> (inspirado em AzureOperationResponse)

- Body: T
- Request: HttpRequestMessageSurrogate
- Response: HttpResponseMessageSurrogate

ServiceException

- Body: ServiceError
- Request: HttpRequestMessageSurrogate
- Response: HttpResponseMessageSurrogate

ServiceError

- Code: string => ErrorCodes.XXX
- Message: string
- Details : IList<ServiceErrorDetail> (code + description)

Tratamento de erros (nas livrarias cliente)

```
ServiceOperationResult SaveUserSettingAsync(UserSettingData setting, CancellationToken token);
ServiceOperationResult<UserSettingData> GetUserSettingAsync(string userId, string key, CancellationToken token);
```

```
try
{
  UserSettingData setting = await this.Client.GetUserSettingAsync("U1", "K1").ConfigureAwait(false);
  // (...)
}
catch (ServiceException ex)
{
  if (ex.Body.Code.EqualsNoCase("UserSettingNotFound"))
  {
    // (...)
  }
  if (ex.Response.StatusCode == HttpStatusCode.NotFound)
  {
    // (...)
  }
}
```

Client.Console e Postman.Collection

Debugging em desenvolvimento

Pipelines

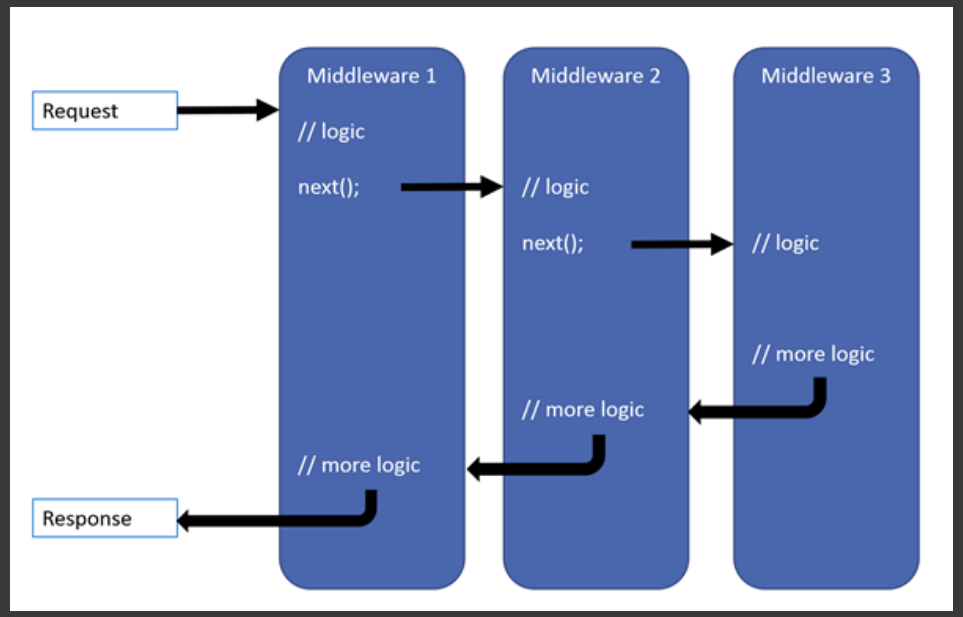
Componente inspirado na pipeline de middlewares ASP.NET Core

Facilita em particular a implementação de cenários de caching

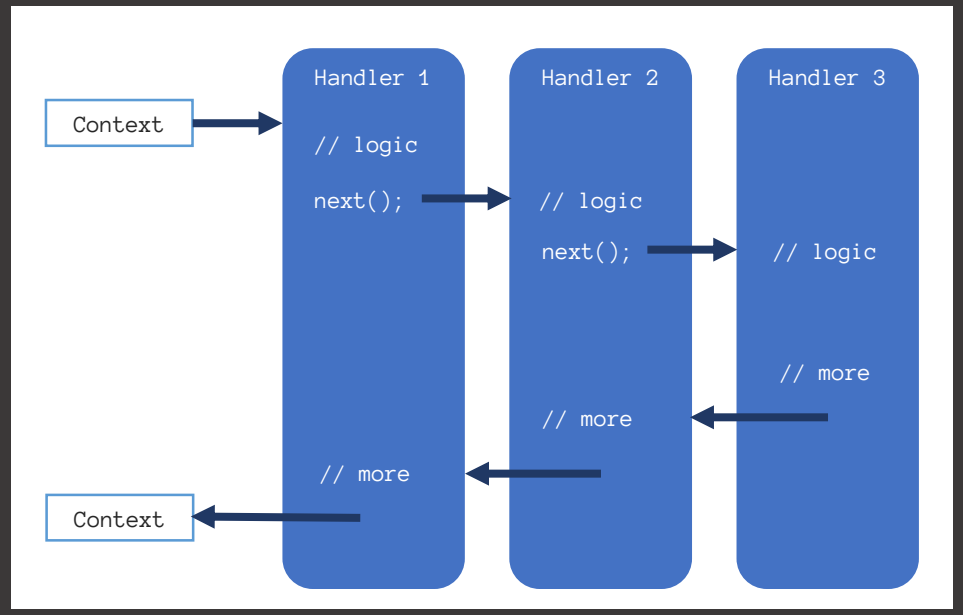
Separation of concerns – cada handler implementa uma parte da lógica do manager

Facilita a configuração (handlers executados em função do contexto e/ou da configuração)

ASP.NET Core Pipeline



Hydrogen Pipeline



Exemplo (pipeline no manager)

```
// Create the pipeline
// When the country code is PT we no dot enable the VIES service handler

IPipeline<VatNumberLookupContext> pipeline = null;

if (countryCode.EqualsNoCase("PT"))
{
    pipeline = new PipelineBuilder<VatNumberLookupContext>()
        .Use<VatNumberLookupHandlerMemory>()
        .Use<VatNumberLookupHandlerStorage>()
        .Build(this.ServiceProvider);
}
else
{
    pipeline = new PipelineBuilder<VatNumberLookupContext>()
        .Use<VatNumberLookupHandlerMemory>()
        .Use<VatNumberLookupHandlerStorage>()
        .Use<VatNumberLookupHandlerViesService>()
        .Build(this.ServiceProvider);
}

// Execute the pipeline

VatNumberLookupContext context = new VatNumberLookupContext(countryCode, vatNumber);
await pipeline.InvokeAsync(context).ConfigureAwait(false);

// Result

VatNumberInfo result = context.Result;
```

Exemplo (handler de caching)

```
/// <inheritdoc />
2 references | Hugo Ribeiro, 10 hours ago | 1 author, 1 change
async Task IPipelineHandler<VatNumberLookupContext>.InvokeAsync(VatNumberLookupContext context, PipelineDelegate<VatNumberLookupContext> next)
{
    // Check the cache

    VatNumberInfo result = await this.GetFromCacheAsync(
        context.CountryCode,
        context.VatNumber)
        .ConfigureAwait(false);

    if (result != null)
    {
        // Set additional properties

        SetAdditionalProperties(result);

        // Result

        context.Result = result;

        // Return

        return;
    }

    // Call next handler

    await next.Invoke(context).ConfigureAwait(false);

    // Has result?

    if (context.Result != null && context.Result.State != VatNumberState.Inconclusive)
    {
        // Add to cache

        await this.AddToCacheAsync(
            context.Result).ConfigureAwait(false);

        // Return

        return;
    }
}
```

```
/// <summary>
/// Defines the context for the pipeline that allows looking up
/// a VAT number.
/// </summary>
22 references | Hugo Ribeiro, 10 hours ago | 1 author, 1 change
internal sealed partial class VatNumberLookupContext
{
    #region Public Properties

    /// <summary>
    /// Gets the VAT number country code.
    /// </summary>
    14 references | Hugo Ribeiro, 10 hours ago | 1 author, 1 change
    public string CountryCode...

    /// <summary>
    /// Gets the VAT number.
    /// </summary>
    13 references | Hugo Ribeiro, 10 hours ago | 1 author, 1 change
    public string VatNumber...

    /// <summary>
    /// Gets or sets the result.
    /// </summary>
    12 references | Hugo Ribeiro, 10 hours ago | 1 author, 1 change
    public VatNumberInfo Result...

    #endregion

    Constructors
}
```

#6

Autorização (OAuth2) e autenticação (OIDC)

02/12/2019

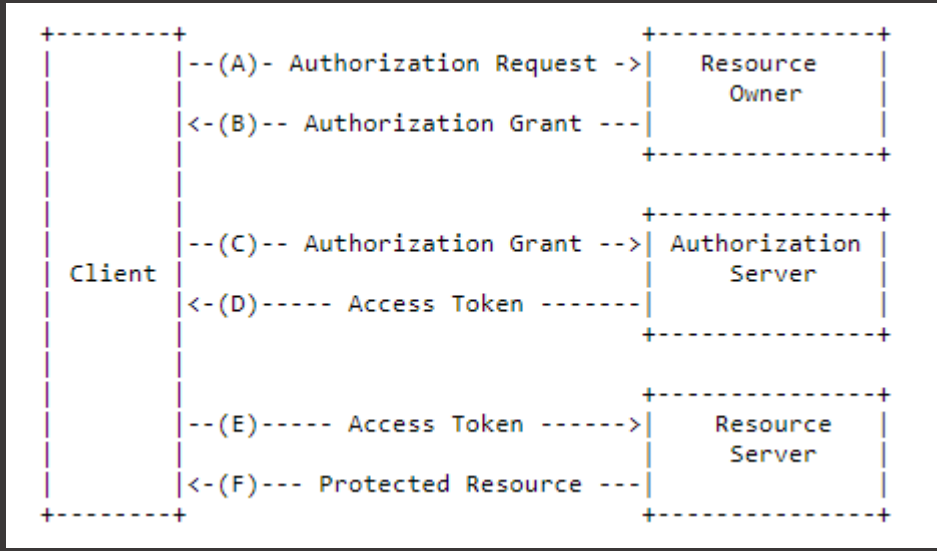
Autorização – OAuth2

https://oauth.net/2/

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server, and is issued a different set of credentials than those of the resource owner.

Instead of using the resource owner's credentials to access protected resources, the client obtains an access token – a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner. The client uses the access token to access the protected resources hosted by the resource server.



Authorization grant types:

- Authorization code
- Implicit
- Resource owner password credentials
- Client credentials

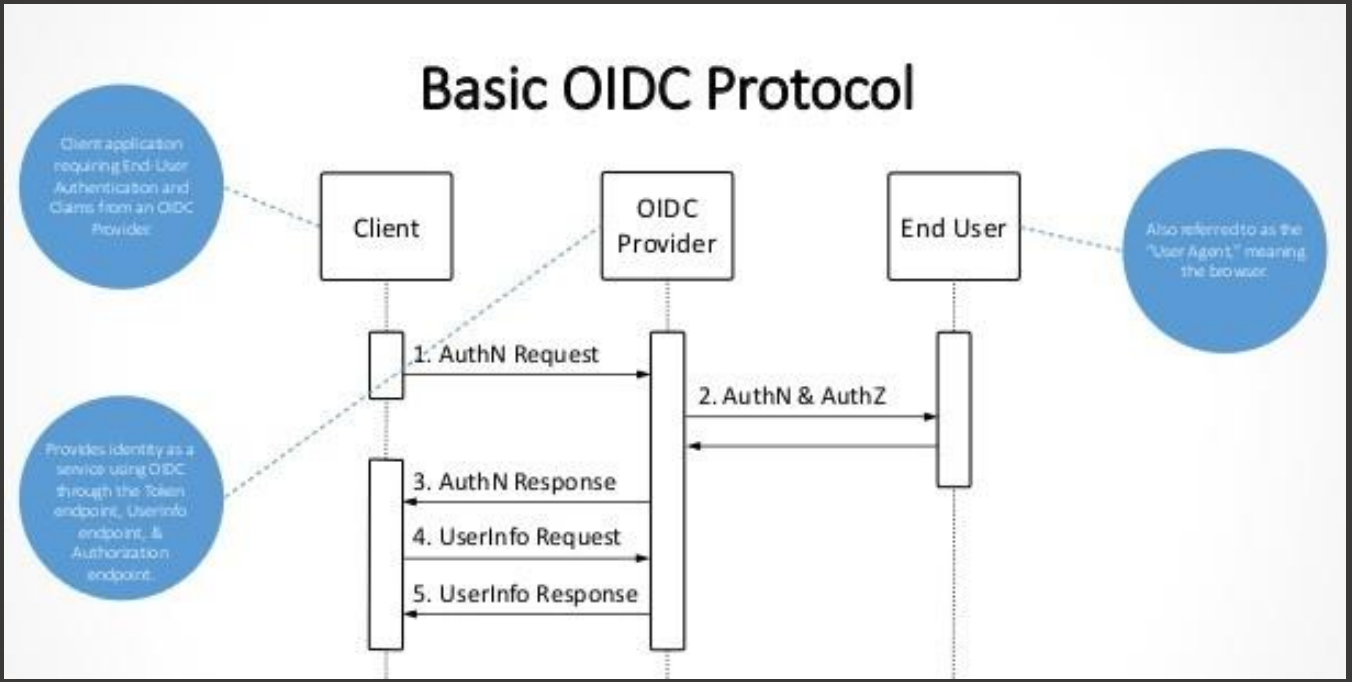
Autenticação – OIDC

https://openid.net/connect/

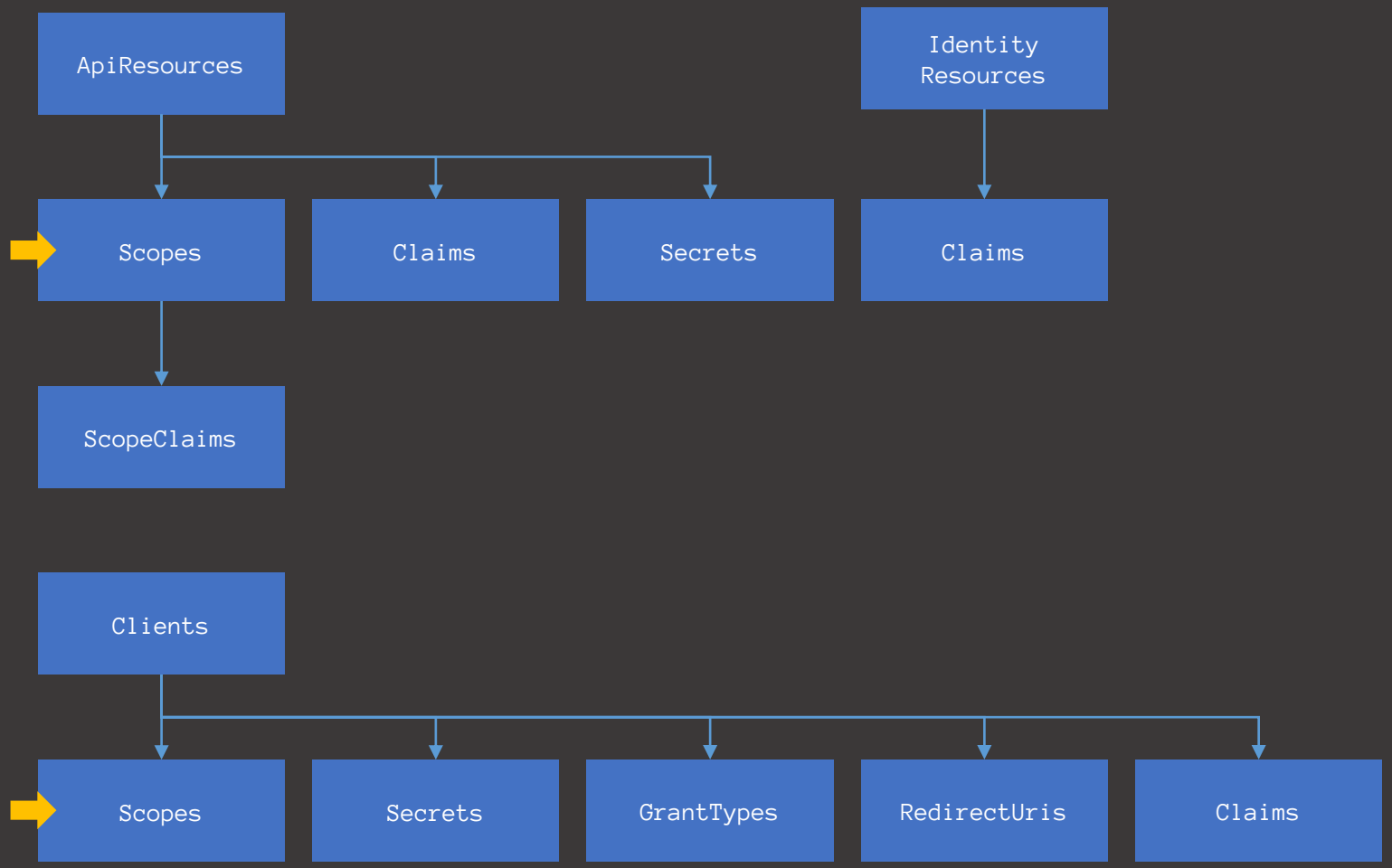
OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

Response types:

- code token
- code id_token
- code id_token token



Conceitos + dados



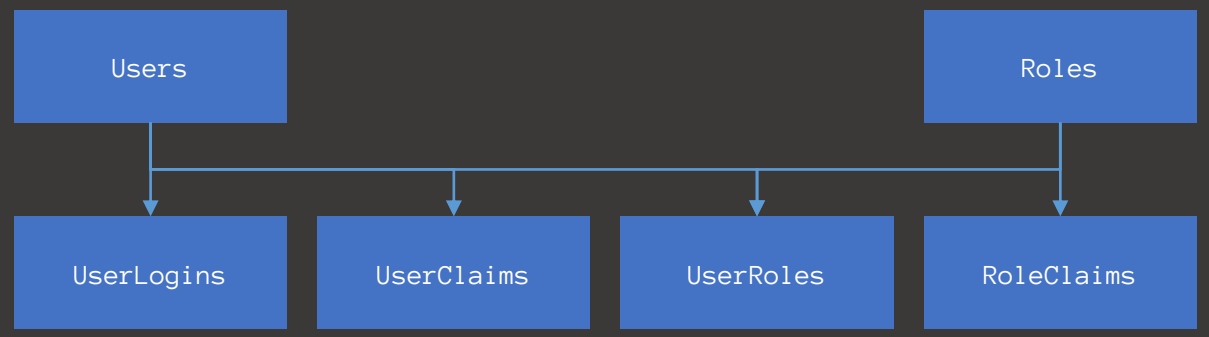
Exemplos:

Identity Resource:
email
profile

Api Resource:
Name = lithium-datalookup
Scope = lithium-datalookup-vatnumber
Scope = lithium-datalookup-exchrate

Client:
Name = lithium-datalookup-clientcredentials
Secret = xpto
Scope = lithium-datalookup-vatnumber

Conceitos + dados



Identity Server

IdentityServer4 (OAuth2 + OIDC + ASP.NET Core)

https://github.com/IdentityServer/IdentityServer4

ASP.NET Core Identity

https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity

PRIMAVERA Identity Server (OAuth2 + OIDC + Identity Database + Backoffice)

https://identity.primaverabss.com

Implementação no serviço (OAuth)

```
/// <summary>
/// Called by the runtime when the application starts to allow the applicati
/// </summary>
/// <param name="services">The services collection.</param>
10 references | Hugo Ribeiro, 10 days ago | 2 authors, 2 changes
public virtual void ConfigureServices(IServiceCollection services)
{
    // Validation

    SmartGuard.NotNull(() => services, services);

    // Configuration

    HostConfiguration hostConfiguration = this.AddConfiguration(services);

    // Validate host configuration

    this.ValidateConfiguration(hostConfiguration);

    // MVC

    this.AddMvc(services, hostConfiguration);

    // Authorization

    this.AddAuthorization(services, hostConfiguration);

    // Authentication

    this.AddAuthentication(services, hostConfiguration);

    // Telemetry

    this.AddTelemetry(services, hostConfiguration);

    // Background services

    this.AddBackgroundServices(services, hostConfiguration);
```

Implementação no serviço (OAuth)

```
/// <summary>
/// Called to add authorization to the service collection.
/// </summary>
/// <param name="services">The service collection.</param>
/// <param name="hostConfiguration">The host configuration.</param>
/// <remarks>
/// The method is called from <see cref="ConfigureServices(IServiceCollection)"/>.
/// </remarks>
4 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected virtual void AddAuthorization(IServiceCollection services, HostConfiguration hostConfiguration)
{
    // Add authorization and authorization policies

    services
        .AddAuthorization(
            (options) =>
            {
                this.AddAuthorizationPolicies(options, hostConfiguration);
            });
}
```

```
/// <summary>
/// Called to add authorization policies to the service collection.
/// </summary>
/// <param name="options">The service collection.</param>
/// <param name="hostConfiguration">The host configuration.</param>
/// <remarks>
/// The method is called from <see cref="AddAuthorization(IServiceCollection, HostConfiguration)"/>.
/// </remarks>
3 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected virtual void AddAuthorizationPolicies(AuthorizationOptions options, HostConfiguration hostConfiguration)
{
    options.AddPolicy(
        Constants.Policies.Notifications,
        (policy) =>
        {
            policy.AuthenticationSchemes = new List<string>
            {
                OidcConstants.AuthenticationSchemes.Bearer
            };

            policy.RequireClaim(JwtClaimTypes.Scope, Primavera.Lithium.Notifications.Models.Metadata.Scopes.Notifications);
        });
}
```

Implementação no serviço (OAuth)

```
/// <summary>
/// Called to add authentication to the service collection.
/// </summary>
/// <param name="services">The service collection.</param>
/// <param name="hostConfiguration">The host configuration.</param>
/// <remarks>
/// The method is called from <see cref="ConfigureServices(IServiceCollection)"/>.
/// </remarks>
3 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected virtual void AddAuthentication(IServiceCollection services, HostConfiguration hostConfiguration)
{
    // Validation

    SmartGuard.NotNull(() => services, services);

    // Add authentication

    AuthenticationBuilder builder = services
        .AddAuthentication(OidcConstants.AuthenticationSchemes.Bearer);

    // Add JWT bearer

    this.AddJwtBearer(services, builder, hostConfiguration);
}
```

Implementação no serviço (OAuth)

```
/// <summary>
/// Called to add JWT bearer authentication to the service collection.
/// </summary>
/// <param name="services">The service collection.</param>
/// <param name="builder">The authentication builder.</param>
/// <param name="hostConfiguration">The host configuration.</param>
/// <remarks>
/// The method is called from <see cref="AddAuthentication(IServiceCollection, HostConfiguration)"/>.
/// </remarks>
3 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected virtual void AddJwtBearer(IServiceCollection services, AuthenticationBuilder builder, HostConfiguration hostConfiguration)
{
    // Validation

    SmartGuard.NotNull(() => services, services);
    SmartGuard.NotNull(() => builder, builder);
    SmartGuard.NotNull(() => hostConfiguration, hostConfiguration);

    // Add

    builder
        .AddJwtBearer(
            (options) =>
            {
                // Standard configuration

                options.Authority = hostConfiguration.IdentityServerBaseUri?.Trim();
                options.Audience = Scopes.Notifications;
                options.RequireHttpsMetadata = false;
                options.IncludeErrorDetails = true;
                options.RefreshOnIssuerKeyNotFound = true;
                options.SaveToken = true;
                options.Events = new HttpBearerChallengeEvents()
                {
                    OnAuthenticationFailed = this.OnJwtBearerAuthenticationFailed,
                    OnForbidden = this.OnJwtBearerForbidden,
                    OnMessageReceived = this.OnJwtBearerMessageReceived,
                    OnTokenValidated = this.OnJwtBearerTokenValidated,
                    OnChallenge = this.OnJwtBearerChallenge
                };

                // Custom configuration

                this.ConfigureJwtBearerOptions(services, builder, options, hostConfiguration);
            });
}
```

Implementação no serviço (OIDC)

```
/// <inheritdoc />
[SuppressMessage("Microsoft.Maintainability", "CA1506:AvoidExcessiveClassCoupling")]
3 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected override void AddAuthentication(IServiceCollection services, HostConfiguration hostConfiguration)
{
    // Validation

    SmartGuard.NotNull(() => hostConfiguration, hostConfiguration);

    // Add authentication

    AuthenticationBuilder builder = services
        .AddAuthentication(
            (options) =>
            {
                options.DefaultScheme = OidcConstants.AuthenticationSchemes.Cookies;
                options.DefaultChallengeScheme = OidcConstants.AuthenticationSchemes.Oidc;
            });

    // Add cookies

    builder
        .AddCookie(
            (options) =>...);

    builder
        .AddOpenIdConnect(
            OidcConstants.AuthenticationSchemes.Oidc,
            (options) =>
            {
                options.SignInScheme = OidcConstants.AuthenticationSchemes.Cookies;
                options.Authority = hostConfiguration.IdentityServerBaseUri?.Trim();
                options.RequireHttpsMetadata = false;
                options.ClientId = Constants.Credentials.Hybrid.ClientId;
                options.ClientSecret = Constants.Credentials.Hybrid.ClientSecret;
                options.ResponseType = OpenIdConnectResponseType.CodeIdTokenToken;
                options.SaveTokens = true;
                options.GetClaimsFromUserInfoEndpoint = true;

                options.Scope.Add(Metadata.Scopes.Notifications);
                options.Scope.Add(JwtClaimTypes.Email);
                options.Scope.Add(JwtClaimTypes.Profile);

                options.TokenValidationParameters = new TokenValidationParameters
                {
                    NameClaimType = "name",
                    RoleClaimType = "role"
                };

                options.Events = new OpenIdConnectEvents()...;
            });

    // Add JWT Bearer

    this.AddJwtBearer(services, builder, hostConfiguration);
}
```

```
/// <inheritdoc />
4 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected override void AddAuthorization(IServiceCollection services, HostConfiguration hostConfiguration)
{
    // Default behavior

    base.AddAuthorization(services, hostConfiguration);

    // Policy handlers

    services.AddSingleton<IAuthorizationHandler, ManagerAuthorizationHandler>();
}

/// <inheritdoc />
3 references | Hugo Ribeiro, 10 days ago | 1 author, 1 change
protected override void AddAuthorizationPolicies(AuthorizationOptions options, HostConfiguration hostConfiguration)
{
    // Validation

    SmartGuard.NotNull(() => options, options);

    // Default behavior

    base.AddAuthorizationPolicies(options, hostConfiguration);

    // Custom policy to prevent non-managers to access the site

    options.AddPolicy(
        Constants.Policies.NotificationsManager,
        (policy) =>
        {
            policy.Requirements.Add(new ManagerAuthorizationRequirement());
        });
}
```

Utilização no cliente

ServiceClientCredentials

- NoCredentials
- AccessTokenCredentials
- ClientCredentials
- AuthenticationCallbackCredentials

HttpBearerChallenge (WWW-Authenticate)

IdentityModel.Client.TokenClient

IdentityModel.Client.DiscoveryClient

```
/// <summary>
/// Defines the service client that allows accessing the Data Lookup Service API.
/// </summary>
[GeneratedCode("Lithium", "2.0")]
[SuppressMessage("Maintainability Rules", "SA1402:FileMayOnlyContainASingleType", Justification = "Because of code generation des
99+ references | Hugo Lourenço | 1 author, 1 change
public partial class DataLookupClient : DataLookupClientBase
{
    #region Code

    #region Public Constructors

    /// <summary> Initializes a new instance of the DataLookupClient class.
    3 references | 0/2 passing | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, ServiceClientCredentials credentials)        ...

    /// <summary> Initializes a new instance of the DataLookupClient class.
    11 references | 0/4 passing | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, ServiceClientCredentials credentials, HttpMessageHandler handler)        ...

    /// <summary> Initializes a new instance of the DataLookupClient class.
    0 references | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, ServiceClientCredentials credentials, HttpMessageHandler handler, bool disposeHandler)

    /// <summary> Initializes a new instance of the DataLookupClient class.
    6 references | 0/2 passing | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, AuthenticationCallback callback)        ...

    /// <summary> Initializes a new instance of the DataLookupClient class.
    6 references | 0/3 passing | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, AuthenticationCallback callback, HttpMessageHandler handler)        ...

    /// <summary> Initializes a new instance of the DataLookupClient class.
    0 references | Hugo Lourenço | 1 author, 1 change
    public DataLookupClient(Uri baseUri, AuthenticationCallback callback, HttpMessageHandler handler, bool disposeHandler)

    #endregion

    Private Methods

    #endregion
}
```

Utilização no cliente

```
/// <summary>
/// Creates an instance of the service client.
/// </summary>
/// <returns>
/// The <see cref="DataLookupClient"/> instance.
/// </returns>
3 references | Hugo Lourenço | 1 author, 1 change
protected virtual DataLookupClient GetServiceClient()
{
    ConsoleHelper.WriteInformationLine("Creating the service client for base URI '{0}'...", this.Configuration.ServiceBaseUri);

    string clientId = Constants.ClientId;
    string clientSecret = Constants.ClientSecret;

    return new DataLookupClient(
        new Uri(this.Configuration.ServiceBaseUri),
        async (args) =>
        {
            // Retrieve the access token

            string accessToken = await ClientCredentials
                .ForAllScopes(
                    new Uri(args.Authority),
                    clientId,
                    clientSecret)
                .RetrieveAccessTokenAsync()
                .ConfigureAwait(false);

            // Result

            return accessToken;
        });
}
```

#7

Background services, custom Web UI, testes, deployment e monitorização

06/12/2019

Background services

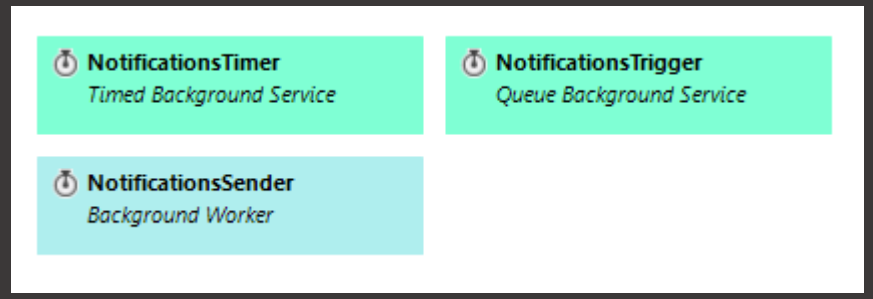
IHostedService

Timed Background Services

Queued Background Services

Background Workers

Locking



Custom Web UI (MVC)

https://lithium-notification.primaverabss.com/

OIDC - autenticação do utilizador

Authorization policies - roles do utilizador

Session

Localization

Controllers + views

Testes unitários e testes de integração

WebApi.Tests, Models.Tests, ClientLib.Tests

Integration.Tests

MonitoringTests – testes de integração gerados automaticamente

Primavera.Hydrogen.DesignTime.UnitTesting

Primavera.Hydrogen.DesignTime.IntegrationTesting

Releases

https://tfs.primaverabss.com/tfs/P.TEC.Elevation/Lithium/_release

Ambientes:

- Development
- Staging
- Preview
- Production

Fluxos de aprovação

Monitorização

https://portal.azure.com/#blade/HubsExtension/BrowseResourceBlade/resourceType/microsoft.insights%2Fcomponents

https://portal.azure.com/#blade/HubsExtension/BrowseResource/resourceType/Microsoft.Web%2Fsites

Failures

Availability (probe)

Logs